Armadillo-IoT ゲートウェイ A6E
製品マニュアル

AG6251-C03D0
AG6251-C03Z
AG6251-U03Z
AG6251-U00Z
AG6241-C01Z
AG6241-U01Z
AG6241-U00Z
AG6211-C02D0
AG6211-C02Z
AG6211-U02Z
AG6211-U00Z
AG6201-C00Z
AG6201-U00Z
AG6253-C03D0
AG6253-C03Z
AG6243-C01Z
AG6213-C02D0
AG6213-C02Z
AG6203-C00Z

Version 3.8.0
2026/01/28

株式会社アットマークテクノ [https://www.atmark-techno.com]

Armadillo サイト [https://armadillo.atmark-techno.com]

Armadillo-IoT ゲートウェイ A6E 製品マニュアル

株式会社アットマークテクノ

製作著作 © 2025-2026 Atmark Techno, Inc.

Version 3.8.0
2026/01/28

目次

1. はじめに 30
1.1. 本書について 30
1.1.1. 本書で扱うこと 30
1.1.2. 本書で扱わないこと 31
1.1.3. 本書で必要となる知識と想定する読者 31
1.1.4. 本書の構成 31
1.1.5. フォント 32
1.1.6. コマンド入力例 32
1.1.7. アイコン 32
1.1.8. ユーザー限定コンテンツ 33
1.1.9. 本書および関連ファイルのバージョンについて 33
1.2. 注意事項 33
1.2.1. 安全に関する注意事項 33
1.2.2. 取扱い上の注意事項 35
1.2.3. 製品の保管について 36
1.2.4. ソフトウェア使用に関しての注意事項 37
1.2.5. 本製品を廃棄する場合について 38
1.2.6. 電波障害について 38
1.2.7. 無線モジュールの安全規制について 38
1.2.8. LED について 39
1.2.9. 保証について 39
1.2.10. 輸出について 40
1.2.11. 商標について 40
1.3. 謝辞 40
2. 製品概要 41
2.1. 製品の特長 41
2.1.1. Armadillo とは 41
2.1.2. Armadillo-IoT ゲートウェイ A6E とは 41
2.1.3. Armadillo Base OS とは 43
2.1.4. Armadillo Base OS のメンテナンスポリシーとアップデートの推奨 45
2.1.5. Armadillo Twin とは 46
2.2. 製品ラインアップ 47
2.2.1. Armadillo-IoT ゲートウェイ A6E 開発セット 49
2.2.2. Armadillo-IoT ゲートウェイ A6E 量産用 50
2.2.3. Armadillo-IoT ゲートウェイ A6E 量産ボード 51
2.2.4. Armadillo-IoT ゲートウェイ A6E オプション品 52
2.2.5. Armadillo-IoT ゲートウェイ A6E BTO サービス 52
2.3. 仕様 53
2.3.1. スタンダードタイプ 53
2.3.2. +Di8+Ai4 タイプ 54
2.4. インターフェースレイアウト 55
2.4.1. スタンダードタイプ 55
2.4.2. +Di8+Ai4 タイプ 57
2.5. ブロック図 59
2.5.1. スタンダードタイプ 59
2.5.2. +Di8+Ai4 タイプ 60
2.6. 使用可能なストレージデバイス 62
2.7. ストレージデバイスのパーティション構成 62
2.8. ソフトウェアのライセンス 63
3. 開発編 64

3.1. 開発の準備 64
3.1.1. 準備するもの 64
3.1.2. 仮想環境のセットアップ 65
3.1.3. VS Code のセットアップ 71
3.1.4. Armadillo の初期化と ABOS のアップデート 73
3.1.5. Armadillo に初期設定をインストールする 76
3.1.6. Python アプリケーションで動作確認する 86
3.1.7. シリアルコンソールを使用する 94
3.1.8. ユーザー登録 100
3.2. アプリケーション開発の流れ 101
3.3. 開発前に知っておくべき Armadillo Base OS の機能・特徴 103
3.3.1. 一般的な Linux OS 搭載組み込み機器との違い 104
3.3.2. Armadillo Base OS 搭載機器のソフトウェア開発手法 105
3.3.3. アップデート機能について 105
3.3.4. SWUpdate とは 105
3.3.5. ファイルの取り扱いについて 111
3.3.6. インストールディスクについて 113
3.4. ハードウェアの設計 114
3.4.1. 信頼性試験データについて 114
3.4.2. 放射ノイズ 114
3.4.3. ESD/雷サージ 114
3.4.4. 拡張基板の設計 115
3.4.5. 電気的仕様 121
3.4.6. 各動作モードにおける電源供給状況 125
3.4.7. reboot コマンドによる再起動時の電源供給について 125
3.4.8. 基板形状図 125
3.4.9. ケース形状図 131
3.4.10. アンテナ形状図 134
3.5. ケースの組み立てと分解方法 135
3.5.1. ケースのパーツ構成 136
3.5.2. ケースの組み立て手順 141
3.5.3. ケースの分解 141
3.6. WLAN アンテナの取り付けと取り外し 145
3.6.1. WLAN 基板アンテナの取り付け 146
3.6.2. WLAN 基板アンテナの取り外し 148
3.6.3. WLAN 基板アンテナのケースへの取り付け 148
3.7. インターフェースの使用方法とデバイスの接続方法 153
3.7.1. インターフェース仕様 153
3.7.2. 周辺装置との接続 156
3.7.3. 電源を入力する 158
3.7.4. Ethernet を使用する 159
3.7.5. 無線 LAN を使用する 160
3.7.6. LTE を使用する 162
3.7.7. SD カードを使用する 164
3.7.8. USB デバイスを使用する 168
3.7.9. UART を使用する 174
3.7.10. GPIO を使用する 178
3.7.11. 接点入力を使用する 181
3.7.12. 接点出力を使用する 187
3.7.13. アナログ入力を使用する 190
3.7.14. I2C デバイスを使用する 194
3.7.15. リアルタイムクロックを使用する 196
3.7.16. ユーザースイッチを使用する 199

3.7.17. LED を使用する 201
3.7.18. Bluetooth® を使用する 204
3.7.19. Wi-SUN デバイスを使用する 206
3.7.20. EnOcean デバイスを使用する 207
3.7.21. CAN デバイスを使用する 208
3.7.22. 入力電圧を監視する 209
3.7.23. 拡張インターフェースを使用する 211
3.7.24. 起動デバイスを設定する 213
3.7.25. 外部電源制御出力を使用する 214
3.8. ソフトウェアの設計 216
3.8.1. 開発者が開発するもの、開発しなくていいもの 216
3.8.2. ユーザーアプリケーションの設計 218
3.8.3. 省電力・間欠動作の設計 218
3.8.4. ログの設計 220
3.9. ネットワーク設定 225
3.9.1. ABOS Web とは 225
3.9.2. ABOS Web へのアクセス 227
3.9.3. ABOS Web のパスワード登録 229
3.9.4. ABOS Web の設定操作 232
3.9.5. ログアウト 233
3.9.6. ABOS Web のパスワード変更 233
3.9.7. WWAN 設定 233
3.9.8. WLAN 設定 237
3.9.9. 接続詳細設定 241
3.9.10. DHCP サーバー設定 243
3.9.11. NAT 設定 244
3.9.12. VPN 設定 251
3.9.13. 状態一覧 253
3.10. USB デバイスの接続を許可する 253
3.11. ABOS Web をカスタマイズする 260
3.12. ABOS Web から Armadillo の電源を操作する 263
3.13. Network Time Protocol (NTP, ネットワーク・タイム・プロトコル) の設定 264
3.14. Armadillo Twin を体験する 265
3.15. ABOSDE によるアプリケーションの開発 265
3.15.1. ABOSDE の対応言語 265
3.15.2. 参照する開発手順の章の選択 266
3.16. ゲートウェイコンテナアプリケーションの開発 267
3.16.1. ゲートウェイコンテナアプリケーション開発の流れ 267
3.16.2. ATDE 上でのセットアップ 268
3.16.3. アプリケーション開発 269
3.16.4. ゲートウェイコンテナアプリケーションの設定 271
3.16.5. ゲートウェイコンテナのディストリビューション 282
3.16.6. Armadillo に転送するディレクトリ及びファイル 282
3.16.7. Armadillo 上でのセットアップ 283
3.16.8. SBOM 生成に関する設定 288
3.16.9. リリース版のビルド 288
3.16.10. 製品への書き込み 289
3.16.11. Armadillo 上のゲートウェイコンテナイメージの削除 289
3.16.12. クラウドを含めた動作確認 289
3.17. CUI アプリケーションの開発 289
3.17.1. CUI アプリケーション開発の流れ 289
3.17.2. ATDE 上でのセットアップ 290
3.17.3. アプリケーション開発 291

3.17.4. Armadillo に転送するディレクトリ及びファイル 296
3.17.5. コンテナ内のファイル一覧表示 296
3.17.6. Armadillo 上でのセットアップ 308
3.17.7. SBOM 生成に関する設定 313
3.17.8. リリース版のビルド 313
3.17.9. 製品への書き込み 313
3.17.10. Armadillo 上のコンテナイメージの削除 313
3.18. C 言語によるアプリケーションの開発 313
3.18.1. C 言語によるアプリケーション開発の流れ 314
3.18.2. ATDE 上でのセットアップ 314
3.18.3. アプリケーション開発 315
3.18.4. コンテナ内のファイル一覧表示 321
3.18.5. Armadillo に転送するディレクトリ及びファイル 333
3.18.6. Armadillo 上でのセットアップ 333
3.18.7. SBOM 生成に関する設定 338
3.18.8. リリース版のビルド 338
3.18.9. 製品への書き込み 338
3.18.10. Armadillo 上のコンテナイメージの削除 338
3.19. SBOM 生成に関わる設定を行う 338
3.19.1. SBOM 生成に必要なファイルを確認する 339
3.20. 生成した SBOM をスキャンする 339
3.20.1. OSV-Scanner のインストール 340
3.20.2. OSV-Scanner でソフトウェアの脆弱性を検査する 341
3.21. システムのテストを行う 341
3.21.1. ランニングテスト 341
3.21.2. 異常系における挙動のテスト 342
3.22. ユーザー設定とユーザーデータを一括削除する 342
4. 量産編 345
4.1. 概略 345
4.1.1. Armadillo Twin を契約する 345
4.1.2. リードタイムと在庫 346
4.1.3. Armadillo 納品後の製造・量産作業 346
4.2. BTO サービスを使わない場合と使う場合の違い 347
4.2.1. BTO サービスを利用しない(標準ラインアップ品) 347
4.2.2. BTO サービスを利用する 347
4.3. 量産時のイメージ書き込み手法 347
4.4. インストールディスクを用いてイメージ書き込みする 348
4.4.1. /etc/swupdate_preserve_file への追記 348
4.4.2. Armadillo Base OS の更新 349
4.4.3. パスワードの確認と変更 349
4.4.4. 開発したシステムをインストールディスクにする 350
4.4.5. VS Code を使用して生成する 350
4.4.6. インストールディスクの動作確認を行う 354
4.4.7. コマンドラインから生成する 354
4.4.8. インストールの実行 361
4.5. SWUpdate を用いてイメージ書き込みする 361
4.5.1. SWU イメージの準備 361
4.5.2. desc ファイルの記述 362
4.6. イメージ書き込み後の動作確認 362
5. 運用編 363
5.1. Armadillo Twin に Armadillo を登録する 363
5.1.1. Armadillo の設置前に登録する場合 363
5.1.2. Armadillo の設置後に登録する場合 363

5.2. Armadillo を設置する 363
5.2.1. 設置場所 363
5.2.2. ケーブルの取り回し 363
5.2.3. WLAN+BT コンボモジュール用アンテナの指向性 363
5.2.4. LTE 外付け用アンテナの指向性 364
5.2.5. LTE の電波品質に影響する事項 365
5.2.6. サージ対策 365
5.2.7. Armadillo の状態を表すインジケータ 365
5.2.8. 個体識別情報の取得 365
5.2.9. 電源を切る 367
5.3. ABOSDE で開発したアプリケーションをアップデートする 367
5.3.1. アプリケーションのアップデート手順 367
5.4. Armadillo のソフトウェアをアップデートする 368
5.4.1. SWU イメージの作成 369
5.4.2. mkswu の desc ファイルを作成する 369
5.4.3. desc ファイルから SWU イメージを生成する 370
5.4.4. イメージのインストール 371
5.5. Armadillo Twin から複数の Armadillo をアップデートする 371
5.6. Armadillo Twin を利用してソフトウェアの脆弱性チェックを行う 371
5.7. eMMC の寿命を確認する 371
5.7.1. eMMC について 371
5.7.2. eMMC 予備領域の確認方法 371
5.8. 本製品の部品変更情報を知る 372
5.9. Armadillo を廃棄する 372
6. 応用編 374
6.1. 省電力・間欠動作機能 374
6.1.1. シャットダウンモードへの遷移と起床 374
6.1.2. スリープモードへの遷移と起床 375
6.1.3. スリープ(SMS 起床可能)モードへの遷移と起床 378
6.1.4. 状態遷移トリガにコンテナ終了通知を利用する 380
6.1.5. コンテナ終了後、指定した秒数だけスリープしてコンテナを再始動する 382
6.1.6. スリープ動作の禁止と禁止解除 383
6.2. persist_file について 384
6.3. swupdate を使用してアップデートする 388
6.3.1. swupdate で可能なアップデート 388
6.3.2. コンテナのアップデート、ユーザーデータディレクトリや Armadillo Base OS の差分アップデート 390
6.3.3. Armadillo Base OS の一括アップデート 393
6.3.4. ブートローダーのアップデート 397
6.3.5. swupdate がエラーする場合の対処 397
6.4. mkswu の .desc ファイルを編集する 397
6.4.1. インストールバージョンを指定する 397
6.4.2. Armadillo へファイルを転送する 399
6.4.3. Armadillo 上で任意のコマンドを実行する 400
6.4.4. Armadillo にファイルを転送し、そのファイルをコマンド内で使用する 401
6.4.5. 動作中の環境でのコマンドの実行 401
6.4.6. Armadillo にコンテナイメージを転送する 402
6.4.7. Armadillo のブートローダーを更新する 402
6.4.8. SWU イメージの設定関連 402
6.4.9. Armadillo 上のコンテナイメージと自動起動用 conf ファイルを削除する 402
6.4.10. SWUpdate 実行中/完了後の挙動を指定する 403
6.4.11. desc ファイル設定例 403
6.5. swupdate_preserve_files について 405

6.6. SWU イメージの内容の確認 406
6.7. SWUpdate と暗号化について 406
6.8. SWUpdate の署名鍵と証明書の更新 406
6.8.1. 署名鍵と証明書の追加 406
6.8.2. 署名鍵と証明書の削除 408
6.9. コンテナの概要と操作方法を知る 408
6.9.1. Podman - コンテナ仮想化ソフトウェアとは 408
6.9.2. コンテナの基本的な操作 408
6.9.3. コンテナとコンテナに関連するデータを削除する 423
6.9.4. コンテナ起動設定ファイルを作成する 425
6.9.5. アットマークテクノが提供するイメージを使う 433
6.9.6. alpine のコンテナイメージをインストールする 435
6.9.7. コンテナのネットワークを扱う 436
6.9.8. コンテナ内にサーバを構築する 438
6.9.9. コンテナからの poweroff 及び reboot 441
6.9.10. 異常検知 442
6.10. ゲートウェイコンテナを動かす 443
6.10.1. ゲートウェイコンテナの概要 443
6.10.2. ゲートウェイコンテナ利用の流れ 444
6.10.3. ゲートウェイコンテナ起動確認 444
6.10.4. 接続先の クラウド 環境を構築 (AWS) 444
6.10.5. 接続先の クラウド 環境を構築 (Azure) 455
6.10.6. 接続先の クラウド 環境を構築 (SORACOM) 460
6.10.7. ゲートウェイコンテナの設定ファイル 464
6.10.8. コンテナ起動・実行 464
6.10.9. クラウドからの操作 483
6.10.10. コンテナの終了 498
6.10.11. ログ内容確認 499
6.10.12. ゲートウェイコンテナの構成 499
6.11. ゲートウェイコンテナアプリケーションを改造する 500
6.12. Web UI から Armadillo をセットアップする (ABOS Web) 500
6.12.1. ABOS Web ではできないこと 500
6.12.2. ABOS Web の設定機能一覧と設定手順 500
6.12.3. コンテナ管理 501
6.12.4. SWU インストール 502
6.12.5. 時刻設定 503
6.12.6. アプリケーション向けのインターフェース (Rest API) 505
6.12.7. カスタマイズ 526
6.12.8. ユーザー設定とユーザーデータの削除 526
6.12.9. ABOS Web を停止する 526
6.12.10. ABOS Web を起動する 526
6.12.11. ABOS Web のセキュリティ対策 527
6.13. ABOSDE から ABOS Web の機能を使用する 528
6.13.1. Armadillo の SWU バージョンを取得する 529
6.13.2. Armadillo のコンテナの情報を取得する 530
6.13.3. Armadillo のコンテナを起動・停止する 531
6.13.4. Armadillo のコンテナのログを取得する 533
6.13.5. Armadillo に SWU をインストールする 533
6.14. ssh 経由で Armadillo Base OS にアクセスする 534
6.15. 入力電圧監視サービス (power-alertd) を使用する 534
6.15.1. 入力電圧監視サービス (power-alertd) の設定 535
6.15.2. 入力電圧監視サービス (power-alertd) の有効・無効化 535
6.16. コマンドラインからネットワーク設定を行う 536

6.16.1. 接続可能なネットワーク 536
6.16.2. ネットワークの設定方法 536
6.16.3. nmcli の基本的な使い方 536
6.16.4. 有線 LAN の接続を確認する 540
6.16.5. LTE 541
6.16.6. 無線 LAN 550
6.16.7. 無線 LAN アクセスポイント (AP) として設定する 551
6.16.8. ファイアウォールの設定方法 554
6.17. コマンドラインからストレージを使用する 556
6.17.1. ストレージのパーティション変更とフォーマット 557
6.18. コマンドラインから CPU の測定温度を取得する 558
6.18.1. 温度を取得する 558
6.19. アナログ入力インターフェースの電源制御を行う 558
6.20. SMS を利用する 559
6.20.1. 初期設定 559
6.20.2. SMS を送信する 559
6.20.3. SMS を受信する 560
6.20.4. SMS 一覧を表示する 560
6.20.5. SMS の内容を表示する 561
6.20.6. SMS を削除する 561
6.20.7. SMS を他のストレージに移動する 561
6.21. ボタンやキーを扱う 562
6.21.1. SW1 の短押しと長押しの対応 562
6.21.2. USB キーボードの対応 563
6.21.3. Armadillo 起動時にのみボタンに反応する方法 564
6.22. 動作中の Armadillo の温度を測定する 564
6.22.1. 温度測定の重要性 564
6.22.2. atmark-thermal-profiler をインストールする 564
6.22.3. atmark-thermal-profiler を実行・停止する 565
6.22.4. atmark-thermal-profiler が出力するログファイルを確認する 565
6.22.5. 温度測定結果の分析 566
6.22.6. Armadillo Twin から Armadillo の温度を確認する 567
6.23. 電源を安全に切るタイミングを通知する 567
6.23.1. signal_indicator の設定 568
6.23.2. DTS overlays の設定 568
6.23.3. 動作確認 569
6.24. Armadillo Base OS をアップデートする 569
6.25. ロールバック状態を確認する 569
6.26. Armadillo 起動時にコンテナの外でスクリプトを実行する 570
6.27. u-boot の環境変数の設定 571
6.28. SD ブートの活用 573
6.28.1. ブートディスクの作成 574
6.28.2. SD ブートの実行 575
6.29. Device Tree をカスタマイズする 576
6.29.1. at-dtweb の更新 576
6.29.2. at-dtweb の起動 576
6.29.3. Device Tree をカスタマイズ 579
6.29.4. DTS overlays によるカスタマイズ 584
6.29.5. 独自の DTS overlay を追加する 586
6.30. Armadillo のソフトウェアをビルドする 587
6.30.1. ブートローダーをビルドする 587
6.30.2. Linux カーネルをビルドする 589
6.30.3. Alpine Linux ルートファイルシステムをビルドする 592

6.31. eMMC のデータリテンション 595
6.32. 動作ログ 595
6.32.1. 動作ログについて 595
6.32.2. 動作ログを取り出す 596
6.32.3. ログファイルのフォーマット 596
6.32.4. ログ用パーティションについて 596
6.32.5. /var/log/ 配下のログに関して 596
6.32.6. ABOS Web でログを確認する 597
6.33. ATDE のバックアップと移行 597
6.34. ATDE・Linux でインストールディスクを作成する 598
6.34.1. GUI でインストールディスクを作成する 599
6.34.2. CUI でインストールディスクを作成する 601
6.35. シリアル通信ソフトウェア(minicom) 602
6.35.1. シリアル通信ソフトウェア(minicom)のセットアップ 602
6.35.2. minicom の起動 604
6.35.3. minicom の終了 605
6.36. vi エディタを使用する 605
6.36.1. vi の起動 605
6.36.2. 文字の入力 606
6.36.3. カーソルの移動 606
6.36.4. 文字の削除 607
6.36.5. 保存と終了 607
6.37. セキュリティ 607
6.37.1. SWUpdate と暗号化について 607
6.37.2. SBOM の提供 608
6.37.3. 不正な USB デバイスの接続を拒否する 610
6.37.4. EdgeLock SE050 を利用したキーストレージ 616
6.37.5. デバッグ機能を閉じる 625
6.37.6. コンテナの最小化 628
6.38. オプション品 630
6.38.1. Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M) 630
6.38.2. +Di8+Ai4 拡張基板 636
6.38.3. +Di8+Ai4 拡張基板のカスケード接続 640

図目次

1.1. 製品化までのロードマップ 31
1.2. LTE モジュール: SIM7672G 認証マーク 39
1.3. LTE モジュール:SIM7672JP 認証マーク 39
1.4. WLAN+BT コンボモジュール:Sterling LWB5+ 認証マーク 39
2.1. Armadillo-IoT ゲートウェイ A6E とは 42
2.2. 様々なデバイスとの接続例 43
2.3. Armadillo Base OS とは 44
2.4. コンテナによるアプリケーションの運用 44
2.5. ロールバックの仕組み 45
2.6. Armadillo Twin　とは 46
2.7. スタンダートタイプと+Di8+Ai4 タイプ 48
2.8. Armadillo-IoT ゲートウェイ A6E 開発セット (スタンダードタイプ) 50
2.9. Armadillo-IoT ゲートウェイ A6E 量産用 51
2.10. Armadillo-IoT ゲートウェイ A6E 量産ボード 51
2.11. Armadillo-IoT ゲートウェイ A6EBTO サービス 52
2.12. Armadillo-IoT ゲートウェイ A6E インターフェースレイアウト (スタンダードタイプ) 56
2.13. Armadillo-IoT ゲートウェイ A6E インターフェースレイアウト (+Di8+Ai4 タイプ) 58
2.14. ブロック図 (スタンダートタイプ) 60
2.15. ブロック図 (+Di8+Ai4 タイプ) 61
3.1. GNOME 端末の起動 67
3.2. GNOME 端末のウィンドウ 67
3.3. ソフトウェアをアップデートする 68
3.4. ATDE にデバイスを接続する 68
3.5. 共有フォルダー設定を開く 69
3.6. 共有フォルダー設定 70
3.7. 共有フォルダーの追加 70
3.8. 「ファイル」に表示される共有フォルダー 71
3.9. VS Code を起動する 71
3.10. VS Code に開発用エクステンションをインストールする 72
3.11. zip ファイルを展開 73
3.12. Win32 Disk Imager Renewal 設定画面 74
3.13. Armadillo-IoT ゲートウェイ A6E を初期化する接続 75
3.14. 起動デバイス設定スイッチの操作 76
3.15. initial_setup.swu を作成する 77
3.16. initial_setup.swu 初回生成時の各種設定 77
3.17. ABOS にアクセスするための接続 79
3.18. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする 80
3.19. ABOSDE に表示されている Armadillo を更新する 80
3.20. ABOSDE を使用して ABOS Web を開く 81
3.21. パスワード登録画面 82
3.22. パスワード登録完了画面 82
3.23. ログイン画面 83
3.24. トップページ 83
3.25. SWU インストール 84
3.26. SWU インストールに成功した画面 84
3.27. プロジェクトを作成する 86
3.28. プロジェクト名を入力する 86
3.29. VS Code で初期設定を行う 87
3.30. VS Code のターミナル 87
3.31. SSH 用の鍵を生成する 87

3.32. VS Code でコンテナイメージの作成を行う 88
3.33. コンテナイメージの作成完了 89
3.34. ABOSDE で Armadillo に SWU をインストール 89
3.35. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する 90
3.36. ssh_config を編集する 90
3.37. Armadillo 上でアプリケーションを実行する 91
3.38. 実行時に表示されるメッセージ 91
3.39. アプリケーションを終了する 92
3.40. Armadillo 上のコンテナイメージを削除する 93
3.41. シリアルコンソールを使用する配線例 94
3.42. Armadillo を接続している COM ポートを指定 95
3.43. Armadillo を接続しているシリアルポートの設定 96
3.44. アプリケーション開発の流れ 102
3.45. persist_file コマンド実行例 112
3.46. chattr によって copy-on-write を無効化する例 113
3.47. 3-State バッファを使用した回路例 115
3.48. ピンを出力専用で使用した回路例 116
3.49. 垂直方向に拡張基板を配置した場合の接続例 117
3.50. 水平方向に拡張基板を配置した場合の接続例 118
3.51. Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M) 搭載例 118
3.52. 水平方向に拡張する場合の基板形状図 119
3.53. 水平方向に拡張する場合の部品搭載制限(A 面側) 120
3.54. 水平方向に拡張する場合の部品搭載制限(B 面側) 121
3.55. 電源回路の構成 124
3.56. 電源シーケンス 124
3.57. 基板形状および固定穴寸法(スタンダードタイプ) 126
3.58. コネクタ、スイッチ、LED 位置(スタンダードタイプ) 127
3.59. 部品高さ(スタンダードタイプ) 128
3.60. 基板形状および固定穴寸法(+Di8+Ai4 タイプ) 129
3.61. コネクタ、スイッチ、LED 位置(+Di8+Ai4 タイプ) 130
3.62. 部品高さ(+Di8+Ai4 タイプ) 131
3.63. ケース形状図(スタンダードタイプ) 132
3.64. ケース形状図(+Di8+Ai4 タイプ) 133
3.65. WLAN 基板アンテナ形状図(Cat.1 bis+WLAN モデル) 134
3.66. WLAN 基板アンテナ形状図(WLAN モデル) 134
3.67. LTE アンテナ形状図 135
3.68. ケース展開図 (スタンダートタイプ) 136
3.69. フック取り付け 1 (スタンダートタイプ) 137
3.70. フック取り付け 2 (スタンダートタイプ) 138
3.71. ケースモデル展開図 (+Di8+Ai4 タイプ) 139
3.72. フック取り付け 1 (+Di8+Ai4 タイプ) 140
3.73. フック取り付け 2 (+Di8+Ai4 タイプ) 141
3.74. WLAN 基板アンテナの位置 142
3.75. フックのツメ 143
3.76. ケースボトムのツメ(スタンダードタイプ) 144
3.77. ケースボトムのツメ(+Di8+Ai4 タイプ) 144
3.78. カバーのツメ 145
3.79. WLAN+BT コンボモジュールのアンテナコネクタの位置 (Cat.1 bis+WLAN モデル) 145
3.80. WLAN+BT コンボモジュールのアンテナコネクタの位置 (WLAN モデル) 146
3.81. WLAN 基板アンテナ(Cat.1 bis+WLAN モデル) 146
3.82. WLAN 基板アンテナ(WLAN モデル) 146
3.83. WLAN 基板アンテナの挿抜治具による取り付け 147
3.84. WLAN 基板アンテナの手による取り付け前の準備 147

3.85. WLAN 基板アンテナの手による取り付け 148
3.86. WLAN 基板アンテナの挿抜治具による取り外し 148
3.87. WLAN 基板アンテナの貼り付け位置 (スタンダードタイプ Cat.1 bis+WLAN モデル) 149
3.88. WLAN 基板アンテナの貼り付け位置 (+Di8+Ai4 タイプ Cat.1 bis+WLAN モデル) 150
3.89. WLAN 基板アンテナの貼り付け位置 (スタンダードタイプ WLAN モデル) 151
3.90. WLAN 基板アンテナの貼り付け位置 (+Di8+Ai4 タイプ WLAN モデル) 152
3.91. WLAN 基板アンテナのケーブル引き回し 152
3.92. Armadillo-IoT ゲートウェイ A6E のインターフェース (スタンダードタイプ) 154
3.93. Armadillo-IoT ゲートウェイ A6E のインターフェース (+Di8+Ai4 タイプ) 155
3.94. Armadillo-IoT ゲートウェイ A6E の接続例 157
3.95. AC アダプタの極性マーク 158
3.96. CON4 LAN LED 160
3.97. nanoSIM カードの接続例 163
3.98. ANT1 接続可能なアンテナコネクタ形状 163
3.99. ANT1 50Ω 同軸ケーブルでの延長例 163
3.100. LTE モデムをリセットまたは 再起動する 164
3.101. LTE モデムの電源を切る 164
3.102. カバーのロックを解除する 165
3.103. カバーを開ける 166
3.104. microSD カードの挿抜 166
3.105. カードマークの確認 166
3.106. カバーを閉める 167
3.107. カバーをロックする 167
3.108. SD カードをマウントする例 167
3.109. SD カードを使用するコンテナの作成例 168
3.110. SD カードをマウントする例 168
3.111. USB メモリをマウントする例 169
3.112. コンテナ内で USB メモリをマウントして使用するコンテナの作成例 170
3.113. コンテナ内で USB メモリをマウントする例 170
3.114. ABOS 上で USB メモリをマウントする例 171
3.115. ABOS 上でマウントした USB メモリを使用するコンテナの作成例 171
3.116. USB メモリに保存されているデータの確認例 171
3.117. USB シリアルデバイスを使用するコンテナの作成例 171
3.118. USB シリアルデバイスの設定の確認例 172
3.119. USB カメラを使用するコンテナの作成例 172
3.120. USB カメラの情報を確認する例 173
3.121. 電源の ON/OFF を切り替えるためのコンテナ作成例 173
3.122. 電源の制御例 174
3.123. RS-485 内部回路 176
3.124. スイッチの状態と終端抵抗の ON/OFF 176
3.125. シリアルインターフェースを使用するコンテナの作成例 177
3.126. シリアルインターフェイスの設定の確認例 177
3.127. GPIO を扱うためのコンテナ作成例 179
3.128. コンテナ内からコマンドで GPIO を操作する例 179
3.129. gpiochip をリスト表示 180
3.130. gpiochip の詳細情報を表示 180
3.131. 接点入力(CON6) 内部回路 182
3.132. 接点入力(CON22) 内部回路 184
3.133. 入力レベルの確認 185
3.134. 接点入力と接点出力をループバックする接続 185
3.135. 接点入力と接点出力をループバックする際の内部回路 186
3.136. DI1、DO1 をループバックした場合のコマンド実行例 186
3.137. 接点入力を使用するコンテナの作成例 186

3.138. 接点入力を操作する例 187
3.139. 接点出力(CON6) 内部回路 188
3.140. 出力レベルを "0" に設定する例 189
3.141. 接点出力を使用するためのコンテナの作成例 189
3.142. コンテナ内からコマンドで接点出力を操作する例 189
3.143. アナログ入力(CON21) 内部回路 192
3.144. アナログ入力(CON21) 接続例 192
3.145. アナログ入力を扱うためのコンテナ作成例 193
3.146. アナログ入力デバイスのラベル名の確認 193
3.147. アナログ入力デバイス名の確認 193
3.148. アナログ入力 raw の取得例 194
3.149. アナログ入力 scale の取得例 194
3.150. I2C を使用するコンテナの作成例 195
3.151. I2C デバイスのスレーブアドレスの確認例 196
3.152. システムクロックを設定 198
3.153. ハードウェアクロックを設定 198
3.154. リアルタイムクロックを使用するコンテナの作成例 198
3.155. リアルタイムクロックの時刻表示と設定例 199
3.156. ユーザースイッチのイベントを取得するコンテナの作成例 200
3.157. evtest コマンドによる確認例 200
3.158. LED を消灯させる 202
3.159. LED を点灯させる 202
3.160. LED の状態を表示する 202
3.161. 対応している LED トリガを表示 203
3.162. LED のトリガに timer を指定する 203
3.163. LED を使用するコンテナの作成例 203
3.164. LED の点灯/消灯の例 204
3.165. Bluetooth® を使用するコンテナの作成例 205
3.166. Bluetooth® 機能を有効にする例 205
3.167. 周辺機器のスキャンとペアリングの例 206
3.168. Wi-SUN デバイスで通信するコンテナの作成例 206
3.169. Wi-SUN デバイスの確認例 207
3.170. EnOcean デバイスで通信するコンテナの作成例 207
3.171. EnOcean デバイスの確認例 208
3.172. CAN 通信するコンテナの作成例 208
3.173. CAN の設定例 208
3.174. 入力電圧を監視するコンテナの作成例 209
3.175. 入力電圧監視デバイス名の確認 210
3.176. 入力電圧 raw の取得例 210
3.177. 入力電圧 scale の取得例 210
3.178. 入力電圧監視計算式 210
3.179. GPIO3_IO13 を High 出力にするノードの例 212
3.180. スイッチの状態と起動デバイス 214
3.181. CON22 外部電源制御出力内部回路 215
3.182. 出力レベルを "1" に設定する場合 215
3.183. 外部電源制御出力を扱うためのコンテナ作成例 216
3.184. 外部電源制御出力を操作する例 216
3.185. 開発者が開発するもの、開発しなくていいもの 217
3.186. ゲートウェイコンテナ使用時、開発者が開発するもの、開発しなくていいもの 217
3.187. 状態遷移図 219
3.188. podman_start コマンドでコンテナを起動する例 221
3.189. コンテナ内で syslog を使用してログを出力する例（シェルスクリプト） 221
3.190. コンテナ内で syslog を使用してログを出力する例（Python） 222

3.191. syslogd に渡す引数を変更 222
3.192. add_args を用いてコンテナに情報を渡すための書き方 224
3.193. add_args を用いてコンテナに情報を渡す例 224
3.194. avahi-daemon を停止する 224
3.195. avahi-daemon を起動する 225
3.196. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする 228
3.197. ABOSDE を使用して ABOS Web を開く 228
3.198. ABOSDE に表示されている Armadillo を更新する 229
3.199. パスワード登録画面 230
3.200. パスワード登録完了画面 231
3.201. ログイン画面 232
3.202. トップページ 232
3.203. パスワード変更画面 233
3.204. WWAN 設定入力画面 234
3.205. WWAN 設定画面 235
3.206. 「WWAN 再接続設定」画面 236
3.207. WLAN クライアント設定入力画面 238
3.208. WLAN クライアント設定画面 239
3.209. WLAN アクセスポイント設定入力画面 240
3.210. WLAN アクセスポイント設定画面 240
3.211. 現在の接続情報画面 241
3.212. LAN 接続設定で固定 IP アドレスに設定した画面 242
3.213. DHCP サーバー設定入力 243
3.214. eth1 に対する DHCP サーバー設定 244
3.215. LTE を宛先インターフェースに指定した設定 245
3.216. ポートフォワーディング設定入力 246
3.217. 全ての受信パケットに対するポートフォワーディング設定 246
3.218. 「IP アドレスフィルタリング設定」画面 247
3.219. 「デフォルトポリシー」画面 247
3.220. 「WAN/LAN インターフェースの設定」画面 248
3.221. 「現在のルール」画面 249
3.222. 「ルールの追加」画面 250
3.223. 「ルールの追加」不可画面 251
3.224. 「IP アドレスフィルタリング設定の初期化」画面 251
3.225. VPN 接続情報入力画面 252
3.226. VPN 設定確認画面 253
3.227. USB 接続制御の設定画面 255
3.228. 接続済みの USB デバイス 256
3.229. 接続済み USB デバイスの詳細情報 256
3.230. USB デバイスを許可する 257
3.231. 許可済み USB デバイス 257
3.232. 接続済みの個体と同じメーカー・製品の全ての個体を許可する 258
3.233. 全ての個体を許可する場合の表示 258
3.234. 許可ルールを削除する 259
3.235. USB デバイスクラス 259
3.236. USB デバイスクラスの追加 260
3.237. ABOS Web のカスタマイズ設定 261
3.238. メニュー変更画面 (一部) 263
3.239. 「電源制御」の画面 264
3.240. chronyd のコンフィグの変更例 264
3.241. 参照する開発手順の章を選択する流れ 266
3.242. ゲートウェイコンテナアプリケーション開発の流れ 268
3.243. プロジェクトを作成する 269

3.244. プロジェクト名を入力する 269
3.245. VS Code で my_project を起動する 269
3.246. 初期設定を行う 270
3.247. VS Code で初期設定を行う 270
3.248. VS Code のターミナル 270
3.249. SSH 用の鍵を生成する 271
3.250. /var/app/rollback/volumes/gw_container/config/cloud_agent.conf のフォーマット 271
3.251. /var/app/rollback/volumes/gw_container/config/sensing_mgr.conf のフォーマット 275
3.252. DO の出力タイミング 280
3.253. VS Code で開発用の SWU の作成を行う 282
3.254. 開発用の SWU の作成完了 282
3.255. Armadillo 上でゲートウェイコンテナアプリケーションを実行する 283
3.256. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする 284
3.257. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する 285
3.258. ABOSDE に表示されている Armadillo を更新する 286
3.259. ssh_config を編集する 286
3.260. Armadillo 上でゲートウェイコンテナアプリケーションを実行する 287
3.261. 実行時に表示されるメッセージ 287
3.262. ゲートウェイコンテナアプリケーションを終了する 288
3.263. リリース版をビルドする 288
3.264. CUI アプリケーション開発の流れ 290
3.265. プロジェクトを作成する 291
3.266. プロジェクト名を入力する 291
3.267. VS Code で my_project を起動する 291
3.268. 初期設定を行う 292
3.269. VS Code で初期設定を行う 293
3.270. VS Code のターミナル 293
3.271. SSH 用の鍵を生成する 293
3.272. ABOSDE でコンテナ OS を選択する 294
3.273. VS Code でコンテナイメージの作成を行う 295
3.274. コンテナイメージの作成完了 295
3.275. Bluetooth® Low Energy パッケージをインストールする 296
3.276. コンテナ内のファイル一覧を表示するタブ 297
3.277. コンテナ内のファイル一覧の例 297
3.278. resources ディレクトリ 298
3.279. コンテナ内のファイル一覧を再表示するボタン 299
3.280. container/resources 下にファイルを追加するボタン 300
3.281. ファイル名を入力 300
3.282. 追加されたファイルの表示 301
3.283. container/resources 下にフォルダーを追加するボタン 302
3.284. container/resources 下にあるファイルを開くボタン 303
3.285. container/resources 下にあるファイルを削除するボタン 304
3.286. コンテナ内のファイルを container/resources 下に保存するボタン 305
3.287. 編集前のファイルを示すマーク 306
3.288. 編集後のファイルを示すマーク 307
3.289. コンテナ内にコピーされないことを示すマーク 308
3.290. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする 309
3.291. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する 310
3.292. ABOSDE に表示されている Armadillo を更新する 311
3.293. ssh_config を編集する 311
3.294. Armadillo 上でアプリケーションを実行する 312
3.295. 実行時に表示されるメッセージ 312
3.296. アプリケーションを終了する 312

3.297. リリース版をビルドする 313
3.298. C 言語によるアプリケーション開発の流れ 314
3.299. プロジェクトを作成する 315
3.300. プロジェクト名を入力する 315
3.301. VS Code で my_project を起動する 316
3.302. 初期設定を行う 317
3.303. VS Code で初期設定を行う 317
3.304. VS Code のターミナル 317
3.305. SSH 用の鍵を生成する 317
3.306. ABOSDE でコンテナ OS を選択する 318
3.307. C 言語による開発における packages.txt の書き方 319
3.308. VS Code でコンテナイメージの作成を行う 320
3.309. コンテナイメージの作成完了 321
3.310. コンテナ内のファイル一覧を表示するタブ 321
3.311. コンテナ内のファイル一覧の例 322
3.312. resources ディレクトリ 323
3.313. コンテナ内のファイル一覧を再表示するボタン 324
3.314. container/resources 下にファイルを追加するボタン 325
3.315. ファイル名を入力 325
3.316. 追加されたファイルの表示 326
3.317. container/resources 下にフォルダーを追加するボタン 327
3.318. container/resources 下にあるファイルを開くボタン 328
3.319. container/resources 下にあるファイルを削除するボタン 329
3.320. コンテナ内のファイルを container/resources 下に保存するボタン 330
3.321. 編集前のファイルを示すマーク 331
3.322. 編集後のファイルを示すマーク 332
3.323. コンテナ内にコピーされないことを示すマーク 333
3.324. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする 334
3.325. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する 335
3.326. ABOSDE に表示されている Armadillo を更新する 336
3.327. ssh_config を編集する 336
3.328. Armadillo 上でアプリケーションを実行する 337
3.329. 実行時に表示されるメッセージ 337
3.330. アプリケーションを終了する 337
3.331. リリース版をビルドする 338
3.332. 必要パッケージの確認 339
3.333. OSV-Scanner の実行ファイルをダウンロード 340
3.334. OSV-Scanner をインストールする 340
3.335. OSV-Scanner がインストールされたことを確認する 340
3.336. OSV-Scanner を用いて SBOM をスキャンする 341
3.337. メモリの空き容量の確認方法 342
3.338. 削除されるユーザー設定とユーザーデータを確認 343
3.339. 実際にユーザー設定とユーザーデータを削除する 343
4.1. Armadillo 量産時の概略図 345
4.2. BTO サービスで対応する範囲 347
4.3. 任意のファイルパスを/etc/swupdate_preserve_files に追記する 349
4.4. Armadillo Base OS を最新にアップデートする 349
4.5. パスワードを変更する 349
4.6. make-installer.swu を作成する 351
4.7. 対象製品を選択する 351
4.8. make-installer.swu 生成時のログ 351
4.9. make-installer.swu インストール時のログ 352
4.10. JTAG と SD ブートを無効化する 355

4.11. JTAG と SD ブートの設定値を確認する 355
4.12. JTAG と SD ブートの設定値をリセットする 355
4.13. U-Boot のコマンドプロンプトを無効化する 356
4.14. U-Boot のコマンドプロンプトの設定値を確認する 356
4.15. 開発完了後のシステムをインストールディスクイメージにする 356
4.16. ip_config.txt の内容 359
4.17. IP アドレスの確認 360
4.18. allocated_ips.csv の内容 360
4.19. インストールログを保存する 360
4.20. インストールログの中身 361
4.21. Armadillo に書き込みたいソフトウェアを ATDE に配置 362
4.22. desc ファイルの記述例 362
5.1. Armadillo-IoT ゲートウェイ A6E WLAN+BT アンテナの指向性 364
5.2. LTE 外付け用アンテナの指向性 364
5.3. 個体番号の取得方法（device-info） 366
5.4. 個体番号の取得方法（get-board-info） 366
5.5. 個体番号の環境変数を conf ファイルに追記 366
5.6. コンテナ上で個体番号を確認する方法 366
5.7. MAC アドレスの確認方法 366
5.8. 出荷時の Ethernet MAC アドレスの確認方法 367
5.9. VS Code を起動 368
5.10. desc ファイルから Armadillo へ SWU イメージをインストールする流れ 369
5.11. コンテナイメージアーカイブ作成例 370
5.12. sample_container_update.desc の内容 370
5.13. sample_container_update.desc の内容 370
5.14. eMMC の予備領域使用率を確認する 371
6.1. aiot-alarm-poweroff コマンド書式 375
6.2. aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込み以外での起床のとき） 375
6.3. aiot-set-wake-trigger コマンド書式 (RTC アラーム割り込みでの起床の場合: 分指定) 376
6.4. aiot-set-wake-trigger コマンド書式 (RTC アラーム割り込みでの起床の場合: 秒指定) 377
6.5. /etc/atmark/ain-set-wake-triggers.conf の記載例 378
6.6. 状態遷移トリガにコンテナ終了通知を利用する場合の設定値を永続化する 381
6.7. 状態遷移トリガの対象コンテナを設定する 382
6.8. コンテナ終了後に指定した秒数だけスリープして再始動する場合のコンテナ設定 383
6.9. スリープやシャットダウンの動作を一時的に禁止する 383
6.10. スリープやシャットダウンの禁止を解除する 383
6.11. スリープやシャットダウンの動作を無効にする 384
6.12. スリープやシャットダウンの動作を有効に戻す 384
6.13. OverlayFS の構成 384
6.14. persist_file コマンドの説明 385
6.15. 書き込み時の OverlayFS の挙動 385
6.16. 書き込み後の persist_file コマンドの実行 386
6.17. persist_file のヘルプ 386
6.18. persist_file 保存・削除手順例 387
6.19. persist_file ソフトウェアアップデート後も変更を維持する手順例 387
6.20. persist_file 変更ファイルの一覧表示例 387
6.21. persist_file でのパッケージインストール手順例 388
6.22. Armadillo Base OS を B 面にコピー 390
6.23. desc ファイルに記述した swudesc_* コマンドを実行 391
6.24. アップデート完了後の挙動 392
6.25. B 面への切り替え 392
6.26. Armadillo Base OS とファイルを B 面にコピー 394
6.27. desc ファイルに記述した swudesc_* コマンドを実行 395

6.28. アップデート完了後の挙動 396
6.29. B 面への切り替え（component=base_os） 396
6.30. mkswu --genkey で署名鍵と証明書を追加する 406
6.31. mkswu --genkey により mkswu.conf に追加された内容 407
6.32. 新しい証明書が Armadillo に追加されていることを確認する 407
6.33. 署名鍵と証明書を削除する設定 408
6.34. 証明書がインストールされていることを確認する 408
6.35. コンテナを作成する実行例 409
6.36. イメージ一覧の表示実行例 410
6.37. podman images --help の実行例 410
6.38. コンテナー覧の表示実行例 410
6.39. podman ps --help の実行例 411
6.40. コンテナを起動する実行例 411
6.41. コンテナを起動する実行例(a オプション付与) 411
6.42. podman start --help 実行例 412
6.43. コンテナを停止する実行例 412
6.44. podman stop --help 実行例 412
6.45. my_container を保存する例 412
6.46. podman build の実行例 413
6.47. podman build でのアップデートの実行例 413
6.48. コンテナを削除する実行例 414
6.49. イメージを削除する実行例 415
6.50. podman rmi --help 実行例 415
6.51. Read-Only のイメージを削除する実行例 415
6.52. コンテナ内部のシェルを起動する実行例 416
6.53. コンテナ内部のシェルから抜ける実行例 416
6.54. podman exec --help 実行例 416
6.55. コンテナを作成する実行例 417
6.56. コンテナの IP アドレスを確認する実行例 417
6.57. ping コマンドによるコンテナ間の疎通確認実行例 417
6.58. pod を使うコンテナを自動起動するための設定例 418
6.59. network を使うコンテナを自動起動するための設定例 418
6.60. abos-ctrl podman-rw の実行例 420
6.61. abos-ctrl podman-storage のイメージコピー例 421
6.62. Armadillo 上のコンテナイメージを削除する 424
6.63. abos-ctrl container-clear 実行例 425
6.64. コンテナを自動起動するための設定例 425
6.65. ボリュームを shared でサブマウントを共有する例 427
6.66. /proc/devices の内容例 428
6.67. add_armadillo_env で設定した環境変数の確認方法 429
6.68. 上記の例でエラーを発生させた際の起動ログ 432
6.69. インストール用のプロジェクトを作成する 434
6.70. at-debian-image のコンテナイメージをインストールする SWU ファイルを作成する 434
6.71. Docker ファイルによるイメージのビルドの実行例 435
6.72. ビルド済みイメージを load する実行例 435
6.73. alpine のコンテナイメージをインストールする SWU ファイルを作成する 436
6.74. コンテナの IP アドレス確認例 436
6.75. ip コマンドを用いたコンテナの IP アドレス確認例 437
6.76. ユーザ定義のネットワーク作成例 437
6.77. IP アドレス固定のコンテナ作成例 437
6.78. コンテナの IP アドレス確認例 438
6.79. コンテナに Apache をインストールする例 438
6.80. コンテナに lighttpd をインストールする例 439

6.81. コンテナに vsftpd をインストールする例 439
6.82. ユーザを追加する例 439
6.83. 設定ファイルの編集例 440
6.84. vsftpd の起動例 440
6.85. コンテナに samba をインストールする例 440
6.86. ユーザを追加する例 440
6.87. samba の起動例 441
6.88. コンテナに sqlite をインストールする例 441
6.89. sqlite の実行例 441
6.90. コンテナから shutdown を行う 441
6.91. ハードウェアウォッチドッグタイマーを使うためのコンテナ作成例 442
6.92. コンテナ内からハードウェアウォッチドッグタイマーを起動する実行例 442
6.93. ハードウェアウォッチドッグタイマーをキックする実行例 443
6.94. ハードウェアウォッチドッグタイマーを停止する実行例 443
6.95. Armadillo-IoT ゲートウェイ A6E クラウド設定データをダウンロードする 457
6.96. コンフィグファイルを編集する 458
6.97. コンフィグファイル設定例 458
6.98. Azure IoT Hub と DPS の設定を実行する 459
6.99. ゲートウェイコンテナを終了する 464
6.100. 接点入力制御シャドウ設定例 487
6.101. 接点入力制御デバイスツイン設定例 488
6.102. 接点出力制御シャドウ設定例 488
6.103. 接点出力制御デバイスツイン設定例 489
6.104. RS-485 レジスタ読み出しシャドウ設定例 491
6.105. RS-485 レジスタ読み出しデバイスツイン設定例 491
6.106. 接点出力制御シャドウ設定例 495
6.107. RS-485 レジスタ書き込みシャドウ設定例 496
6.108. LED 点灯制御シャドウ設定例 498
6.109. ログファイルのフォーマット 499
6.110. ログファイルの Count_value の出力例 500
6.111. コンテナ管理 501
6.112. SWU インストール 502
6.113. 最新バージョンに更新 503
6.114. ネットワークタイムサーバーと同期されている場合の状況確認画面 504
6.115. ネットワークタイムサーバーと同期されていない場合の状況確認画面 504
6.116. ネットワークタイムサーバーの設定項目 504
6.117. タイムゾーンの設定項目 505
6.118. 設定管理の Rest API トークン一覧表示 506
6.119. ユーザ名とパスワード認証の例 521
6.120. 証明書認証の例 521
6.121. ABOS Web を停止する 526
6.122. ABOS Web を起動する 526
6.123. コンテナからのアクセスのみを許可する設定 527
6.124. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする 529
6.125. ABOSDE の ABOS Web パスワード入力画面 529
6.126. ABOSDE で Armadillo の SWU バージョンを取得 530
6.127. ABOSDE で Armadillo のコンテナ情報を取得 531
6.128. ABOSDE で Armadillo のコンテナを起動 532
6.129. ABOSDE で Armadillo のコンテナを停止 532
6.130. ABOSDE で Armadillo のコンテナのログを取得 533
6.131. ABOSDE で Armadillo に SWU をインストール 534
6.132. /etc/atmark/power-alertd.conf の記載例 535
6.133. /etc/atmark/power-alertd.conf の永続化 535

6.134. 入力電圧監視サービス (power-alertd) を有効にする 535
6.135. 入力電圧監視サービス (power-alertd) を無効にする 536
6.136. nmcli のコマンド書式 536
6.137. コネクションの一覧表示 537
6.138. コネクションの有効化 537
6.139. コネクションの無効化 537
6.140. コネクションの作成 537
6.141. コネクションファイルの永続化 537
6.142. コネクションの削除 538
6.143. コネクションファイル削除時の永続化 538
6.144. 固定 IP アドレス設定 539
6.145. DHCP の設定 539
6.146. DNS サーバーの指定 539
6.147. コネクションの修正の反映 539
6.148. デバイスの一覧表示 539
6.149. デバイスの接続 540
6.150. デバイスの切断 540
6.151. 有線 LAN の PING 確認 540
6.152. LTE のコネクションの作成 543
6.153. LTE のコネクションの設定の永続化 543
6.154. ユーザー名とパスワード設定が不要な LTE のコネクションの作成 544
6.155. MCC/MNC を指定した LTE コネクションの作成 544
6.156. PAP 認証を有効にした LTE コネクションの作成 544
6.157. LTE のコネクション確立 545
6.158. LTE の PING 確認 545
6.159. LTE コネクションを切断する 545
6.160. LTE 再接続サービスの設定値を永続化する 547
6.161. LTE 再接続サービスの状態を確認する 547
6.162. LTE 再接続サービスを停止する 547
6.163. LTE 再接続サービスを開始する 547
6.164. LTE 再接続サービスを無効にする 547
6.165. LTE 再接続サービスを有効にする 548
6.166. 認識されているモデムの一覧を取得する 548
6.167. モデムの情報を取得する 548
6.168. SIM の情報を取得する 549
6.169. 回線情報を取得する 549
6.170. 無線 LAN アクセスポイントに接続する 550
6.171. 無線 LAN のコネクションが作成された状態 550
6.172. 無線 LAN の PING 確認 551
6.173. bridge インターフェースを作成する 551
6.174. wlan0 インターフェースを NetworkManager の管理から外す 552
6.175. hostapd.conf を編集する 552
6.176. dnsmasq の設定ファイルを編集する 554
6.177. 特定のポートに対する IP アドレスのフィルタリング 554
6.178. 特定のポートに対する IP アドレスのフィルタリングの設定を削除 555
6.179. mount コマンド書式 556
6.180. ストレージのマウント 556
6.181. ストレージのアンマウント 557
6.182. fdisk コマンドによるパーティション変更 557
6.183. EXT4 ファイルシステムの構築 558
6.184. i.MX6ULL の測定温度を取得する 558
6.185. アナログ入力インターフェースの電源オフ 559
6.186. アナログ入力インターフェースの電源オン 559

6.187. アナログ入力インターフェースの再起動 559
6.188. 言語設定 559
6.189. SMS の作成 560
6.190. SMS 番号の確認 560
6.191. SMS の送信 560
6.192. SMS の一覧表示 560
6.193. SMS の内容を表示 561
6.194. SMS の削除 561
6.195. SIM カードのストレージに SMS を移動 561
6.196. LTE モジュールの内蔵ストレージに SMS を移動 562
6.197. buttond で SW1 を扱う 562
6.198. buttond で USB キーボードのイベントを確認する 563
6.199. buttond で USB キーボードを扱う 563
6.200. buttond で SW1 を Armadillo 起動時のみ受け付ける設定例 564
6.201. atmark-thermal-profiler をインストールする 565
6.202. atmark-thermal-profiler を実行する 565
6.203. atmark-thermal-profiler を停止する 565
6.204. ログファイルの内容例 565
6.205. サーマルシャットダウン温度の確認(Armadillo-IoT ゲートウェイ G4 を例に) 566
6.206. Armadillo-IoT ゲートウェイ G4 で取得した温度のグラフ 567
6.207. /etc/conf.d/indicator_signals の記述内容 568
6.208. /etc/conf.d/indicator_signals の永続化 568
6.209. indicator_signals のコンソール出力 569
6.210. abos-ctrl status の例 569
6.211. /var/at-log/atlog の内容の例 570
6.212. local サービスの実行例 571
6.213. uboot_env.d のコンフィグファイルの例 571
6.214. at-dtweb の起動開始 577
6.215. ボード選択画面 577
6.216. Linux カーネルディレクトリ選択画面 578
6.217. at-dtweb 起動画面 578
6.218. UART1(RXD/TXD) のドラッグ 579
6.219. CON8 8/9 ピンへのドロップ 580
6.220. 信号名の確認 580
6.221. プロパティの設定 581
6.222. プロパティの保存 582
6.223. 全ての機能の削除 582
6.224. ECSPI1 の削除 583
6.225. dtbo/desc ファイルの生成 583
6.226. dtbo/desc の生成完了 584
6.227. /boot/overlays.txt の変更例 585
6.228. armadillo-600-customize.dts の編集 586
6.229. 編集した dts ファイルのビルド 586
6.230. ビルドした DTS overlay ファイルを Armadillo に配置 586
6.231. ビルドした DTS overlay ファイルを永続化 587
6.232. /boot/overlays.txt の編集と永続化 587
6.233. ブートローダーのソースコードをダウンロードする 587
6.234. デフォルトコンフィギュレーションの適用 587
6.235. ブートローダーのビルド 588
6.236. ブートローダーを SWU でインストールする方法 588
6.237. Linux カーネルソースコードの展開 589
6.238. Linux カーネルデフォルトコンフィギュレーションの適用 589
6.239. Linux カーネルコンフィギュレーションの変更 589

6.240. Linux カーネルコンフィギュレーション設定画面 590
6.241. Linux カーネルのビルド 591
6.242. Linux カーネルを SWU でインストールする方法 591
6.243. Linux カーネルを build_rootfs でインストールする方法 592
6.244. 動作ログのフォーマット 596
6.245. ATDE のバックアップ用アーカイブの作成 598
6.246. アーカイブの展開 598
6.247. zip ファイルを展開 599
6.248. 展開したフォルダ内にある img ファイルをダブルクリック 599
6.249. ディスクイメージをリストア 599
6.250. microSD カードを指定 600
6.251. 確認のウィンドウ 600
6.252. パスワードの要求 601
6.253. minicom の設定の起動 602
6.254. minicom の設定 602
6.255. minicom のシリアルポートの設定 603
6.256. minicom のシリアルポートのパラメータの設定 603
6.257. minicom シリアルポートの設定値 604
6.258. minicom 起動方法 604
6.259. minicom 終了確認 605
6.260. vi の起動 606
6.261. 入力モードに移行するコマンドの説明 606
6.262. 文字を削除するコマンドの説明 607
6.263. desc ファイルの追加例 610
6.264. USB 接続制御機能を管理するコマンド 610
6.265. USB 接続制御機能の状態を確認する 611
6.266. USB 接続制御機能を有効化する 612
6.267. USB 接続制御機能を無効化する 612
6.268. 接続されている USB デバイスをリストする 612
6.269. 指定した USB デバイスを許可する 613
6.270. パラメータで指定した USB デバイスを許可する 613
6.271. Armadillo に接続している USB デバイスのパラメータを調べる 613
6.272. 指定した USB デバイスを拒否する 614
6.273. 指定した USB デバイスクラスを許可する 614
6.274. 指定可能な USB デバイスクラスを確認する 614
6.275. 定義済みの USB デバイス許可ルールを表示する 615
6.276. 定義済みの USB デバイス許可ルールを削除する 616
6.277. Plug & Trust Middleware の周辺のソフトウェアスタック 617
6.278. EdgeLock SE050 内に保存されているユーザーが保存した鍵をすべて削除する 624
6.279. 削除したユーザーが保存した鍵を読み出そうとしてみる 624
6.280. インストールディスクの JTAG と SD ブートを無効化する 626
6.281. JTAG と SD ブートの設定値を確認する 627
6.282. JTAG と SD ブートの設定値をリセットする 627
6.283. インストールディスクを作成する 627
6.284. Debian コンテナの Dockerfile の例 629
6.285. ケース外観図 631
6.286. 形状図 ケース外形(トップとボトムを組み合わせた状態) 632
6.287. 形状図 ケース内高さおよび開口部寸法 633
6.288. 形状図 カバーパーツ A (アンテナ穴有り) 633
6.289. 形状図 カバーパーツ A (アンテナ穴無し) 634
6.290. 形状図 カバーパーツ B (アンテナ穴有り) 634
6.291. 形状図 カバーパーツ B (アンテナ穴無し) 635
6.292. 形状図 カバーパーツ C 635

Armadillo-IoT ゲートウェイ A6E 製品マニュアル

6.293. 形状図 カバーパーツ D 636
6.294. +Di8+Ai4 拡張基板のブロック図 637
6.295. +Di8+Ai4 拡張基板のインターフェースレイアウト 637
6.296. +Di8+Ai4 拡張基板基板形状図 640
6.297. +Di8+Ai4 拡張基板のカスケード接続 641
6.298. +Di8+Ai4 拡張基板の組付け方法 642
6.299. 悪い組付け例 642
6.300. ピンヘッダの位置と設定 643

表目次

1.1. 使用しているフォント 32
1.2. 表示プロンプトと実行環境の関係 32
1.3. コマンド入力例での省略表記 32
1.4. 無線 LAN 機能の利用条件 36
1.5. 推奨温湿度環境について 37
1.6. LTE モジュール: SIM7672G 適合証明情報 38
1.7. LTE モジュール:SIM7672JP 適合証明情報 39
1.8. WLAN+BT コンボモジュール:Sterling LWB5+ 適合証明情報 39
2.1. スタンダードタイプの型番と搭載モジュールおよび付属品一覧 48
2.2. +Di8+Ai4 タイプの型番と搭載モジュールおよび付属品一覧 49
2.3. Armadillo-IoT ゲートウェイ A6E 開発セット一覧 (スタンダードタイプ) 49
2.4. Armadillo-IoT ゲートウェイ A6E 開発セット一覧 (+Di8+Ai4 タイプ) 49
2.5. Armadillo-IoT ゲートウェイ A6E 開発セットの内容物 49
2.6. Armadillo-IoT ゲートウェイ A6E 量産用一覧 (スタンダードタイプ) 50
2.7. Armadillo-IoT ゲートウェイ A6E 量産用一覧 (+Di8+Ai4 タイプ) 50
2.8. Armadillo-IoT ゲートウェイ A6E 量産ボード一覧 51
2.9. Armadillo-IoT ゲートウェイ A6E のオプション品一覧 52
2.10. 仕様 (スタンダートタイプ) 53
2.11. 仕様 (+Di8+Ai4 タイプ) 54
2.12. 各部名称と機能 56
2.13. 各部名称と機能 58
2.14. ストレージデバイス 62
2.15. eMMC の GPP の用途 62
2.16. eMMC メモリマップ 62
2.17. eMMC ブートパーティション構成 63
2.18. eMMC GPP 構成 63
3.1. ユーザー名とパスワード 66
3.2. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ) 112
3.3. CON8 3-State バッファを使用した回路の IO ピン仕様 115
3.4. 推奨コネクタ (垂直接続) 117
3.5. 推奨コネクタ (水平接続) 118
3.6. 絶対最大定格 121
3.7. 推奨動作条件 122
3.8. 電源出力仕様 122
3.9. 入出力インターフェース(CON6)の入出力仕様 122
3.10. 拡張インターフェース(CON8)の入出力仕様(OVDD = VCC_3.3V) 123
3.11. アナログ入力インターフェース(CON21)の入力仕様 123
3.12. 入出力インターフェース(CON6)の入出力仕様 123
3.13. 各動作モードにおける電源供給状況 125
3.14. reboot コマンドで再起動した場合の各電源供給状況 125
3.15. ケース展開図パーツ一覧 (スタンダートタイプ) 137
3.16. ケースモデル展開図パーツ一覧 (+Di8+Ai4 タイプ) 140
3.17. Armadillo-IoT ゲートウェイ A6E インターフェース一覧 (スタンダードタイプ) 154
3.18. Armadillo-IoT ゲートウェイ A6E インターフェース一覧 (+Di8+Ai4 タイプ) 156
3.19. 接続可能な電線(電源) 159
3.20. 電源入力(CON6) 信号配列 159
3.21. LAN インターフェース(CON4) 信号配列 160
3.22. CON4 LAN LED の動作 160
3.23. nanoSIM インターフェース(CON3) 信号配列 162
3.24. SD インターフェース(CON1) 信号配列 165

3.25. USB インターフェース(CON9) 信号配列 169
3.26. USB コンソール用インターフェース(CON7) 信号配列 174
3.27. 接続可能な電線(RS-485) 175
3.28. RS-485 インターフェース(CON6) 信号配列 176
3.29. GPIO に対応する CON8 ピン番号 178
3.30. 接続可能な電線(DI) 181
3.31. 接点入力(CON6) 信号配列 182
3.32. 接点入力に対応する GPIO 番号 182
3.33. 接続可能な電線(DI) 183
3.34. 接点入力(CON22) 信号配列 183
3.35. 接点入力に対応する GPIO 番号 184
3.36. 接続可能な電線(DO) 187
3.37. 接点出力(CON6) 信号配列 188
3.38. 接点出力に対応する CON6 ピン番号 188
3.39. アナログ入力(CON21) 接続可能な電線 190
3.40. アナログ入力(CON21) 信号配列 191
3.41. I2C デバイス 194
3.42. RTC バックアップインターフェース(CON10) 信号配列 197
3.43. 時刻フォーマットのフィールド 197
3.44. ユーザースイッチ(SW1) 信号配列 199
3.45. インプットデバイスファイルとイベントコード 200
3.46. LED の状態表示 201
3.47. LED の接続 201
3.48. LED トリガの種類 203
3.49. CON8 搭載コネクタと対向コネクタ例 211
3.50. 拡張インターフェース(CON8) 信号配列 211
3.51. スイッチの状態と起動デバイス 214
3.52. CON22 接続可能な電線 214
3.53. 外部電源制御出力(CON22) 信号配列 215
3.54. 外部電源制御出力に対応する CON22 ピン番号 215
3.55. 動作モード別デバイス状態 219
3.56. Armadillo Base OS のデフォルトで開放しているポート 224
3.57. WWAN 再接続設定項目 236
3.58. 「ルールの追加」画面の項目 250
3.59. OpenVPN 設定項目 252
3.60. 認証方式として「ユーザー名とパスワード」を選択した場合の設定項目 252
3.61. 認証方式として「証明書」を選択した場合の設定項目 253
3.62. デフォルトで許可されている USB デバイス 254
3.63. 用意する favicon 画像 262
3.64. 制御できる機能 264
3.65. ABOSDE の対応言語 266
3.66. [CLOUD] 設定可能パラメータ 272
3.67. [CLOUD] 設定可能パラメータ 273
3.68. [AWS] 設定可能パラメータ 273
3.69. [AZURE] 設定可能パラメータ 275
3.70. [SORACOM] 設定可能パラメータ 275
3.71. [DEFAULT] 設定可能パラメータ 277
3.72. [LOG] 設定可能パラメータ 278
3.73. [CPU_temp] 設定可能パラメータ 278
3.74. [DI1,DI2] 設定可能パラメータ 279
3.75. [DO1,DO2] 設定可能パラメータ 280
3.76. [RS485_Data1, RS485_Data2, RS485_Data3, RS485_Data4] 設定可能パラメータ 281
4.1. インストールディスクと SWUpdate によるソフトウェア書き込みの比較 348

4.2. インストール中に実行される関数 358
5.1. EXT_CSD_PRE_EOL_INFO の値の意味 372
6.1. aiot-set-wake-trigger TRIGGER 一覧 375
6.2. 設定パラメーター 380
6.3. 遷移先の動作モード 381
6.4. 起床条件 381
6.5. swudesc_* コマンドの種類 391
6.6. アップデート完了後の挙動の種類 392
6.7. swudesc_* コマンドの種類 395
6.8. アップデート完了後の挙動の種類 396
6.9. add_hotplugs オプションに指定できる主要な文字列 428
6.10. add_armadillo_env で追加される環境変数 429
6.11. 利用できるインターフェース・機能 443
6.12. 利用できるクラウドベンダー・サービス 443
6.13. デバイス情報データー覧 465
6.14. CPU 温度データー覧 465
6.15. 接点入力データー覧 465
6.16. RS-485 データー覧 465
6.17. ユーザースイッチ関連データー覧 465
6.18. Azure Stream Analytics ジョブ設定値 470
6.19. Azure Stream Analytics ジョブ入力設定値 472
6.20. 接点入力設定値 486
6.21. 接点出力設定値 488
6.22. RS-485 レジスタ読み出し設定値 489
6.23. 接点出力制御設定値 494
6.24. RS-485 レジスタ書き込みシャドウ設定値 495
6.25. LED 点灯制御設定値 497
6.26. POWER_ALERTD_ARGS に記載するオプションの説明 535
6.27. ネットワークとネットワークデバイス 536
6.28. 固定 IP アドレス設定例 538
6.29. ファームウェアのバージョンと相互接続性試験の完了状況 541
6.30. APN 設定情報 542
6.31. sim7672-boot.conf の設定内容 542
6.32. psm の tau と act-time に設定可能な値 543
6.33. edrx に設定可能な値 543
6.34. APN 情報設定例 543
6.35. 再接続サービス設定パラメーター 546
6.36. thermal_profile.csv の各列の説明 566
6.37. rollback-status の出力と意味 570
6.38. rollback-status 追加情報の出力と意味 570
6.39. u-boot の主要な環境変数 572
6.40. microSD カードのパーティション構成 574
6.41. build-rootfs のファイル説明 593
6.42. /var/log/ 配下のログ 597
6.43. シリアル通信設定 602
6.44. シリアルコンソールとして使用する取り外し可能デバイス 604
6.45. 入力モードに移行するコマンド 606
6.46. カーソルの移動コマンド 607
6.47. 文字の削除コマンド 607
6.48. 保存・終了コマンド 607
6.49. desc ファイルの設定項目 610
6.50. デバイスリストの各列の意味 612
6.51. 2 列目が device のときの許可ルールリストの各列の意味 615

6.52. Plug & Trust Middle のパッケージ 617
6.53. Plug & Trust Middleware のサポート 617
6.54. Debian と Alpine の比較 629
6.55. Python プロジェクトにおける Debian と Alpine コンテナのサイズ比較 629
6.56. Python プロジェクトにおける --no-install-recommends の有無による Debian コンテナのサイズ比較 630
6.57. Armadillo-IoT ゲートウェイ A6E 関連のオプション品 630
6.58. Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)について 631
6.59. ケース(トップ/ボトム)の仕様 631
6.60. フックの仕様 631
6.61. カバーパーツ A/B/C/D の仕様 631
6.62. +Di8+Ai4 拡張基板の仕様 636
6.63. +Di8+Ai4 拡張基板のインターフェース一覧 638
6.64. CON20 信号配列 638
6.65. CON23 信号配列 639
6.66. +Di8+Ai4 拡張基板のピンヘッダ設定 643
6.67. +Di8+Ai4 拡張基板の順番と gpiochip に続く数字の対応関係 643
6.68. "deviceX"と+Di8+Ai4 拡張基板の順番の関係 644

1. はじめに

このたびは Armadillo-IoT ゲートウェイ A6E をご利用いただき、ありがとうございます。

Armadillo-IoT ゲートウェイシリーズ は、各種センサーとネットワークとの接続を中継する IoT 向けゲートウェイの開発プラットフォームです。ハードウェアやソフトウェアをカスタマイズして、オリジナルのゲートウェイを素早く、簡単に開発することができます。

Armadillo-IoT ゲートウェイ A6E は、標準インターフェースとして RS-485、接点入出力、Ethernet、USB を搭載。様々なセンサー・デバイスを接続することができます。

Armadillo-IoT ゲートウェイシリーズの中でも、省電力や間欠動作機能に特化した IoT ゲートウェイです。自立型のシステムを構築する際には、ソーラーパネルや蓄電池をより小さなものにでき、システム全体のコストを大幅に低減することができます。 ゲートウェイを間欠動作させることで、さらに細かな節電が可能です。スリープ時はほとんど電力を消費せず、その状態からすぐに高速起動することができます。必要なときだけゲートウェイを起動しクラウドと通信し、データ送信後は再スリープといった運用を実現します。

Linux ベースのディストリビューションとして専用設計の Armadillo Base OS (以下、ABOS) を 搭載しています。ABOS はユーザーアプリケーションをコンテナとして管理する機能、ABOS 自体とコンテナの両方を安全にリモートアップデートする機能、ネットワークや HW セキュリティに関する機能を集約したコンパクトな Armadillo 専用 OS です。

ABOS とユーザーアプリケーションを含むコンテナはどちらも、ABOS の リモートアップデート機能で安全にアップデートすることができます。ABOS はアップデートの状態を二面化しているので電源やネットワークの遮断によって中断してもアップデート前の状態に復旧します。

ユーザーアプリケーションをコンテナとして管理できる機能を利用し、各種クラウド IoT サービス (Azure IoT や AWS IoT Core) に対応したゲートウェイコンテナを用意しました。これまでの Armadillo-IoT ゲートウェイシリーズでは、ユーザー自身が開発するアプリケーションソフトウェアで、センサーからのデータ取得、クラウドへのアップロード等のゲートウェイとしての機能の他、通信障害時の対応、セキュリティ対応、間欠動作時の挙動などの難しい課題を自ら解決する必要がありました。あらかじめ用意されたゲートウェイコンテナを活用することで、これらの課題に対処することができ、短期間に IoT システムを構築可能です。

以降、本書では他の Armadillo ブランド製品にも共通する記述については、製品名を Armadillo と表記します。

1.1. 本書について

1.1.1. 本書で扱うこと

本書では、Armadillo-IoT ゲートウェイ A6E の使い方、製品仕様(ソフトウェアおよびハードウェア)、オリジナルの製品を開発するために必要となる情報、その他注意事項について記載しています。Linux あるいは組み込み機器に不慣れな方でも読み進められるよう、コマンドの実行例なども記載しています。

また、本書では、アットマークテクノが運営する Armadillo サイトをはじめ、開発に有用な情報を得る方法についても、随時説明しています。

1.1.2. 本書で扱わないこと

本書では、一般的な Linux のプログラミング、デバッグ方法やツールの扱い方、各種モジュールの詳細仕様など、一般的な情報や、他に詳しい情報があるものは扱いません。また、(Armadillo-IoT ゲートウェイ A6E を使用した)最終製品あるいはサービスに固有な情報や知識も含まれていません。

1.1.3. 本書で必要となる知識と想定する読者

本書は、読者として Armadillo-IoT ゲートウェイ A6E を使ってオリジナルの機器を開発するエンジニアを想定して書かれています。また、「Armadillo-IoT ゲートウェイ A6E を使うと、どのようなことが実現可能なのか」を知りたいと考えている設計者・企画者も対象としています。Armadillo-IoT ゲートウェイ A6E は組込みプラットフォームとして実績のある Armadillo をベースとしているため、標準で有効になっている機能以外にも様々な機能を実現することができます。

ソフトウェアエンジニア	端末からのコマンドの実行方法など、基本的な Linux の扱い方を知っているエンジニアを対象読者として想定しています。プログラミング言語として C/C++を扱えることは必ずしも必要ではありませんが、基礎的な知識がある方が理解しやすい部分もあります。
ハードウェアエンジニア	電子工学の基礎知識を有したエンジニアを対象読者として想定しています。回路図や部品表を読み、理解できる必要があります。

1.1.4. 本書の構成

本書には、Armadillo-IoT ゲートウェイ A6E をベースに、オリジナルの製品を開発するために必要となる情報を記載しています。また、取扱いに注意が必要な事柄についても説明しています。

本書の章構成は「図 1.1. 製品化までのロードマップ」に示す流れを想定したものとなっています。

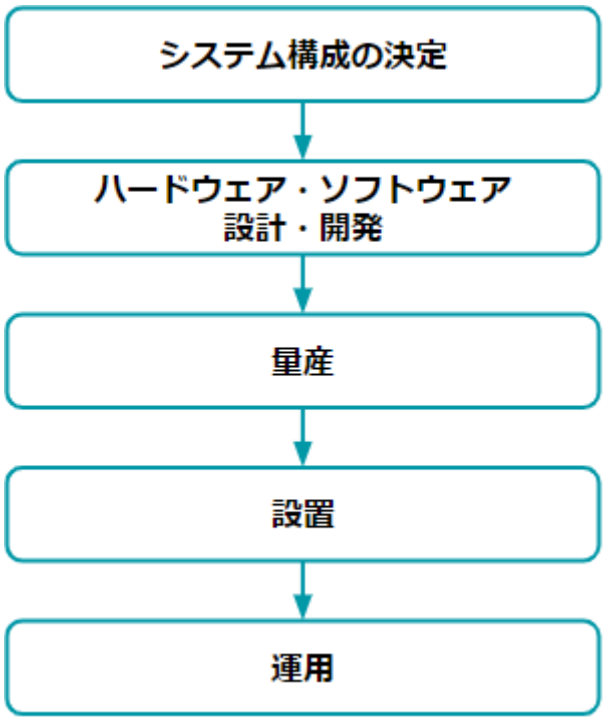


図 1.1 製品化までのロードマップ

- 「システム構成の決定」、「ハードウェア・ソフトウェア設計・開発」

システムが必要とする要件から使用するクラウド、デバイス、ソフトウェア仕様を決定および、ハードウェア・ソフトウェアの開発時に必要な情報について、「3. 開発編」で紹介します。

- **「量産」**

  開発完了後の製品を量産する方法について、「4. 量産編」で紹介します。

- **「設置」、「運用」**

  設置時の勘所や、量産した Armadillo を含めたハードウェアを設置し、運用する際に利用できる情報について、「5. 運用編」で紹介します。

また、本書についての概要を「1. はじめに」に、Armadillo-IoT ゲートウェイ A6E についての概要を「2. 製品概要」に、開発～運用までの一連の流れの中で説明しきれなかった機能についてを、「6. 応用編」で紹介します。

1.1.5. フォント

本書では以下のような意味でフォントを使いわけています。

表 1.1 使用しているフォント

フォント例	説明
本文中のフォント	本文
[PC ~]$ ls	プロンプトとユーザ入力文字列
text	編集する文字列や出力される文字列。またはコメント

1.1.6. コマンド入力例

本書に記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「 / 」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「 ~ 」で表します。

表 1.2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC の root ユーザで実行
[PC /]$	作業用 PC の一般ユーザで実行
[ATDE ~/]#	ATDE 上の root ユーザで実行
[ATDE ~/]$	ATDE 上の一般ユーザで実行
[armadillo /]#	Armadillo 上 Linux の root ユーザで実行
[armadillo /]$	Armadillo 上 Linux の一般ユーザで実行
[container /]#	Podman コンテナ内で実行
=>	Armadillo 上 U-Boot の保守モードで実行

コマンド中で、変更の可能性のあるものや、環境により異なるものに関しては以下のように表記します。適宜読み替えて入力してください。

表 1.3 コマンド入力例での省略表記

表記	説明
[VERSION]	ファイルのバージョン番号

1.1.7. アイコン

本書では以下のようにアイコンを使用しています。



注意事項を記載します。



役に立つ情報を記載します。



用語の説明や補足的な説明を記載します。

1.1.8. ユーザー限定コンテンツ

アットマークテクノ Armadillo サイトで購入製品登録を行うと、製品をご購入いただいたユーザーに限定して公開している限定コンテンツにアクセスできるようになります。主な限定コンテンツには、下記のものがあります。

・ 各種信頼性試験データ・納入仕様書等製造関連情報

限定コンテンツを取得するには、「3.1.8. ユーザー登録」を参照してください。

1.1.9. 本書および関連ファイルのバージョンについて

本書を含めた関連マニュアル、ソースファイルやイメージファイルなどの関連ファイルは最新版を使用することをおすすめいたします。本書を読み始める前に、Armadillo サイトで最新版の情報をご確認ください。

Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ドキュメントダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6e/resources/documents

Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6e/resources/software

1.2. 注意事項

1.2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。



- ご使用の前に必ず製品マニュアルおよび関連資料をお読みになり、使用上の注意を守って正しく安全にお使いください。
- マニュアルに記載されていない操作・拡張などを行う場合は、弊社 Web サイトに掲載されている資料やその他技術情報を十分に理解した上で、お客様自身の責任で安全にお使いください。
- 水・湿気・ほこり・油煙等の多い場所に設置しないでください。火災、故障、感電などの原因になる場合があります。
- 本製品に搭載されている部品の一部は、発熱により高温になる場合があります。周囲温度や取扱いによってはやけどの原因となる恐れがあります。本体の電源が入っている間、または電源切断後本体の温度が下がるまでの間は、基板上の電子部品、及びその周辺部分には触れないでください。
- 本製品を使用して、お客様の仕様による機器・システムを開発される場合は、製品マニュアルおよび関連資料、弊社 Web サイトで提供している技術情報のほか、関連するデバイスのデータシート等を熟読し、十分に理解した上で設計・開発を行ってください。また、信頼性および安全性を確保・維持するため、事前に十分な試験を実施してください。
- 本製品は、機能・精度において極めて高い信頼性・安全性が必要とされる用途(医療機器、交通関連機器、燃焼制御、安全装置等)での使用を意図しておりません。これらの設備や機器またはシステム等に使用された場合において、人身事故、火災、損害等が発生した場合、当社はいかなる責任も負いかねます。
- 本製品には、一般電子機器用(OA 機器・通信機器・計測機器・工作機械等)に製造された半導体部品を使用しています。外来ノイズやサージ等により誤作動や故障が発生する可能性があります。万一誤作動または故障などが発生した場合に備え、生命・身体・財産等が侵害されることのないよう、装置としての安全設計(リミットスイッチやヒューズ・ブレーカー等の保護回路の設置、装置の多重化等)に万全を期し、信頼性および安全性維持のための十分な措置を講じた上でお使いください。
- 電池をご使用の際は、極性(プラスとマイナス)を逆にして装着しないでください。また、電池の使用推奨期限を過ぎた場合や RTC の時刻を保持できなくなった場合には、直ちに電池を交換してください。そのまま使用すると、電池が漏液、発熱、破裂したり、ケガや製品の故障の原因となります。万一、漏れた液が身体に付着した場合は多量の水で洗い流してください。
- 無線 LAN 機能を搭載した製品は、心臓ペースメーカーや補聴器などの医療機器、火災報知器や自動ドアなどの自動制御器、電子レンジ、高度な電子機器やテレビ・ラジオに近接する場所、移動体識別用の構内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

1.2.2. 取扱い上の注意事項

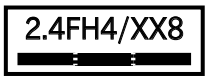
本製品に恒久的なダメージをあたえないよう、取扱い時には以下のような点にご注意ください。

破損しやすい箇所	microSD コネクタおよびそのカバーやフラットケーブルコネクタは、破損しやすい部品になっています。無理に力を加えて破損することのないよう十分注意してください。
本製品の改造	本製品に改造[1]を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設[2]を行う場合は、作業前に必ず動作確認を行ってください。
電源投入時のコネクタ着脱	本製品や周辺回路に電源が入っている状態で、活線挿抜対応インターフェース(LAN, USB) [3]以外へのコネクタ着脱は、絶対に行わないでください。
静電気	本製品には CMOS デバイスを使用しており、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。
ラッチアップ	電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等により、使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには、保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。
衝撃	落下や衝撃などの強い振動を与えないでください。
使用場所の制限	無線機能を搭載した製品は、テレビ・ラジオに近接する場所で使用すると、受信障害を招く恐れがあります。
振動	振動が発生する環境では、Armadillo が動かないよう固定して使用してください。
電池の取り扱い	電池の使用推奨期限を過ぎる前に電池の交換をしてください。使用推奨期限を超えて使用すると、電池の性能が充分に発揮できない場合や、電池を漏液させたり、製品を破損させるおそれがあります。
電波に関する注意事項(2.4GHz 帯無線)	2.4GHz 帯の電波を使用する機能(無線 LAN 等)は、自動ドアなどの自動制御電子機器に影響が出る場合、すぐに使用を中止してください。



[1]本書を含めた関連マニュアルで改造方法を記載している箇所および、コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2]改造やコネクタを増設する際にはマスキングを行い、周囲の部品に半田くず、半田ボール等付着しないよう十分にご注意ください。
[3]別途、活線挿抜を禁止している場合を除く

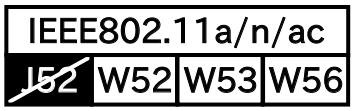
この無線機(Sterling LWB5+)は 2.4GHz 帯を使用します。全帯域を使用し、かつ移動体識別装置の帯域が回避可能です。変調方式として DS-SS および OFDM 方式を採用し、想定される与干渉距離は 40m 以下です。



この無線機(Sterling LWB5+)は 2.4GHz 帯を使用します。全帯域を使用し、かつ移動体識別装置の帯域が回避不可です。変調方式として FH-SS 方式を採用し、想定される与干渉距離は 40m 以下です。

電波に関する注意事項(5GHz 帯無線)

この無線機(Sterling LWB5+)は 5GHz 帯を使用します。



W52、W53 の屋外での利用は電波法により禁じられています。W53、W56 での AP モードは、現在工事設計認証を受けていないため使用しないでください。

上記の周波数帯域における無線 LAN 機能の利用条件を 「表 1.4. 無線 LAN 機能の利用条件」 に示します。

表 1.4 無線 LAN 機能の利用条件

場所＼周波数帯	2.4GHz 帯	W52	W53	W56
屋内	STA：○ AP：○	STA：○ AP：○	STA：○ AP：×	STA：○ AP：×
屋外	STA：○ AP：○	STA：△[a] AP：×	STA：× AP：×	STA：○ AP：×

[a]5.2GHz 帯高出力データ通信システムの基地局又は陸上移動中継局と通信する場合のみ可

電波に関する注意事項(LTE)

この無線機(SIM7672G、SIM7672JP)は LTE 通信を行います。LTE 通信機能は、 心臓ペースメーカーや除細動器等の植込み型医療機器の近く(15cm 程度以内)で使用しないでください。

電気通信事業法に関する注意事項について

本製品の有線 LAN を、電気通信事業者の通信回線(インターネットサービスプロバイダーが提供している通信網サービス等)に直接接続することはできません。接続する場合は、必ず電気通信事業法の認定を受けた端末設備(ルーター等)を経由して接続してください。

1.2.3. 製品の保管について



- 製品を在庫として保管するときは、高温・多湿、埃の多い環境、水濡れの可能性のある場所、直射日光のあたる場所、有毒ガス(特に腐食性ガス)の発生する場所を避け、精密機器の保管に適した状態で保管してください。
- 保管環境として推奨する温度・湿度条件は以下のとおりです。

表 1.5 推奨温湿度環境について

推奨温湿度環境	5～35℃/70%RH 以下[a] [b]

[a]半田付け作業を考慮した保管温度範囲となっております。半田付けを行わない、または、すべての半田付けが完了している場合の推奨温度・湿度条件は、製品の動作温度・湿度範囲となります。
[b]温度変化の少ない場所に保管してください。保管時の急激な温度変化は結露が生じ、金属部の酸化、腐食などが発生し、はんだ濡れ性に影響が出る場合があります。

- 製品を包装から取り出した後に再び保管する場合は、帯電防止処理された収納容器を使用してください。

1.2.4. ソフトウェア使用に関しての注意事項

製品に含まれるソフトウェアについて

製品の標準出荷状態でプリインストールされている Linux 対応ソフトウェアは、個別に明示されている（書面、電子データでの通知、口頭での通知を含む）場合を除き、オープンソースとしてソースコードが提供されています。再配布等の権利については、各ソースコードに記載のライセンス形態にしたがって、お客様の責任において行使してください。また、製品に含まれるソフトウェア（付属のドキュメント等も含む）は、現状有姿（AS IS）にて提供します。お客様ご自身の責任において、使用用途・目的の適合について事前に十分な検討と試験を実施した上でお使いください。アットマークテクノは、当該ソフトウェアが特定の目的に適合すること、ソフトウェアの信頼性および正確性、ソフトウェアを含む製品の使用による結果について、お客様に対し何らの保証も行いません。

パートナー等の協力により Armadillo ブランド製品向けに提供されているミドルウェア、その他各種ソフトウェアソリューションは、ソフトウェア毎にライセンスが規定されています。再頒布権等については、各ソフトウェアに付属する readme ファイル等をご参照ください。その他のバンドルソフトウェアについては、各提供元にお問い合わせください。



以下のソフトウェアは、オープンソースソフトウェアではありません。

ボード情報取得ツール(get-board-info)

パスワードの設定について

Armadillo Base OS にはシリアルコンソールや SSH (標準ソフトウェアでは無効)、 Web UI からパスワードを使用してログインできる機構が備わっています。

基本的には初回ログイン時または、専用の開発環境での開発開始時に機器のパスワードの再設定を求められます。この時、推測されやすい単純なパスワードを使用すると外部からの不正アクセスや機器の乗っ取りの原因になりますので、必ず推測されにくい複雑なパスワードを設定してください。

ソフトウェアのアップデートについて

弊社はソフトウェアのアップデートを定期的に実施します。詳細は「2.1.4. Armadillo Base OS のメンテナンスポリシーとアップデートの推奨」を参照してください。

製品のソフトウェアは常に最新版にアップデートした上でご使用ください。

お客様がアップデートを適用せずに運用した場合、セキュリティインシデント、利用できる機能の制限、サポートの制限が発生することがあります。

1.2.5. 本製品を廃棄する場合について

本製品を廃棄する場合は、必ず「5.9. Armadillo を廃棄する」に記載の注意事項をご一読の上、機器内に保存されているログや個人情報などのデータを正しく全て削除した上で廃棄してください。

情報資産が機器内に残留したまま廃棄することは、個人情報や機密情報が第三者に漏洩する可能性があります。また、法的な問題や責任を引き起こす可能性があります。データの削除手順について不明点がある場合は、弊社までお問い合わせください。

1.2.6. 電波障害について



この装置は、クラス B 情報技術装置です。この装置は、住宅環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。取扱説明書に従って正しい取り扱いをして下さい。VCCI-B

1.2.7. 無線モジュールの安全規制について

本製品に搭載されている LTE モジュール SIM7672G および SIM7672JP は、電気通信事業法に基づく設計認証を受けています。

また、本製品に搭載されている LTE モジュール SIM7672G および SIM7672JP 、WLAN+BT コンボモジュール Sterling LWB5+ は、電波法に基づく工事設計認証を受けています。

これらの無線モジュールを国内で使用するときに無線局の免許は必要ありません。



以下の事項を行うと法律により罰せられることがあります。

- 無線モジュールやアンテナを分解/改造すること。
- 無線モジュールや筐体、基板等に直接印刷されている証明マーク・証明番号、または貼られている証明ラベルをはがす、消す、上からラベルを貼るなどし、見えない状態にすること。

認証番号は次のとおりです。

表 1.6 LTE モジュール: SIM7672G 適合証明情報

項目	内容
型式又は名称	SIM7672G
電波法に基づく工事設計認証における認証番号	201-230741
電気通信事業法に基づく設計認証における認証番号	D230149201

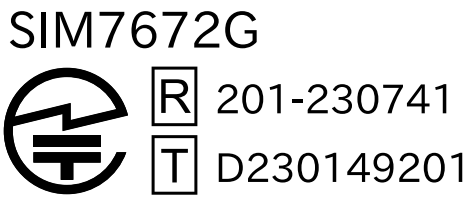


図 1.2 LTE モジュール: SIM7672G 認証マーク

表 1.7 LTE モジュール:SIM7672JP 適合証明情報

項目	内容
型式又は名称	SIM7672JP
電波法に基づく工事設計認証における認証番号	201-250329
電気通信事業法に基づく設計認証における認証番号	P250073201

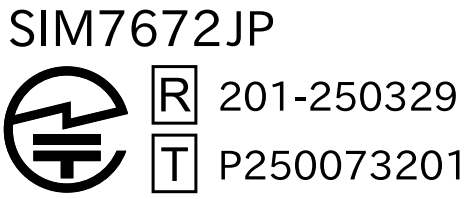


図 1.3 LTE モジュール:SIM7672JP 認証マーク

表 1.8 WLAN+BT コンボモジュール:Sterling LWB5+ 適合証明情報

項目	内容
型式又は名称	Sterling LWB5+
電波法に基づく工事設計認証における認証番号	201-200402

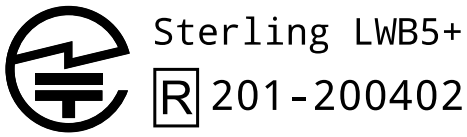


図 1.4 WLAN+BT コンボモジュール:Sterling LWB5+ 認証マーク

1.2.8. LED について

本製品に搭載されている LED は部品の特性上、LED ごとに色味や輝度の差が発生する場合がありますので、あらかじめご了承ください。

1.2.9. 保証について

本製品の本体基板は、製品に添付もしくは弊社 Web サイトに記載している「製品保証規定」に従い、ご購入から標準で 1 年間の交換保証を行っております。添付品およびソフトウェアは保証対象外となりますのでご注意ください。

また、製品を安心して長い期間ご利用いただくために、保証期間を 2 年または 3 年間に延長できる「延長保証サービス」をオプションで提供しています。詳細は「製品保証サービス」を参照ください。

製品保証サービス https://armadillo.atmark-techno.com/support/warranty

製品保証規定 https://armadillo.atmark-techno.com/support/warranty/policy

1.2.10. 輸出について

- 当社製品は、原則として日本国内での使用を想定して開発・製造されています。
- 海外の法令および規則への適合については当社はなんらの保証を行うものではありません。
- 当社製品を輸出するときは、輸出者の責任において、日本国および関係する諸外国の輸出関連法令に従い、必要な手続を行っていただきますようお願いいたします。
- 日本国およびその他関係諸国による制裁または通商停止を受けている国家、組織、法人または個人に対し、当社製品を輸出、販売等することはできません。
- 当社製品および関連技術は、大量破壊兵器の開発等の軍事目的、その他国内外の法令により製造・使用・販売・調達が禁止されている機器には使用することができません。

1.2.11. 商標について

- Armadillo は株式会社アットマークテクノの登録商標です。その他の記載の商品名および会社名は、各社・各団体の商標または登録商標です。™、®マークは省略しています。
- SD、SDHC、SDXC、microSD、microSDHC、microSDXC、SDIO ロゴは SD-3C, LLC の商標です。



- Bluetooth®のワードマークおよびロゴは、Bluetooth SIG, Inc.が所有する登録商標です。

1.3. 謝辞

Armadillo で使用しているソフトウェアの多くは Free Software / Open Source Software で構成されています。Free Software / Open Source Software は世界中の多くの開発者の成果によってなりたっています。この場を借りて感謝の意を表します。

2. 製品概要

2.1. 製品の特長

2.1.1. Armadillo とは

「Armadillo(アルマジロ)」は、Arm コアプロセッサ搭載・Linux 対応の組み込みプラットフォームのブランドです。Armadillo ブランド製品には以下の特長があります。

- Arm プロセッサ搭載・省電力設計

  Arm コアプロセッサを搭載しています。1〜数ワット程度で動作する省電力設計で、発熱が少なくファンを必要としません。

- 小型・手のひらサイズ

  CPU ボードは名刺サイズ程度の手のひらサイズが主流です。名刺の 1/3 程度の小さな CPU モジュールや無線 LAN モジュール等、超小型のモジュールもラインアップしています。

- 標準 OS として Linux をプリインストール

  標準 OS に Linux を採用しており、豊富なソフトウェア資産と実績のある安定性を提供します。ソースコードをオープンソースとして公開しています。

- 開発環境

  Armadillo の開発環境として、「Atmark Techno Development Environment (ATDE)」を無償で提供しています。ATDE は Oracle VirtualBox 用の仮想イメージファイルです。このイメージには、Linux デスクトップ環境をベースに GNU クロス開発ツールやその他の必要なツールが事前にインストールされています。ATDE を使うことで、開発用 PC の用意やツールのインストールなどといった開発環境を整える手間を軽減することができます。

2.1.2. Armadillo-IoT ゲートウェイ A6E とは

Armadillo-IoT ゲートウェイ A6E は、従来製品以上に省電力で動作する IoT ゲートウェイです。間欠動作にも対応しており、ハード・ソフトの両面で優れた省電力性能を有しています。

搭載する通信モジュールやインターフェースの有無でモデルが用意されているため、接続する機器やセンサーに合わせて選択することができます。

高い自由度と、開発のしやすさ、組み込み機器としての堅牢性をバランスよく兼ね備えており、オリジナルの商用 IoT ゲートウェイを市場のニーズに合わせてタイムリーに開発したい方に好適です。

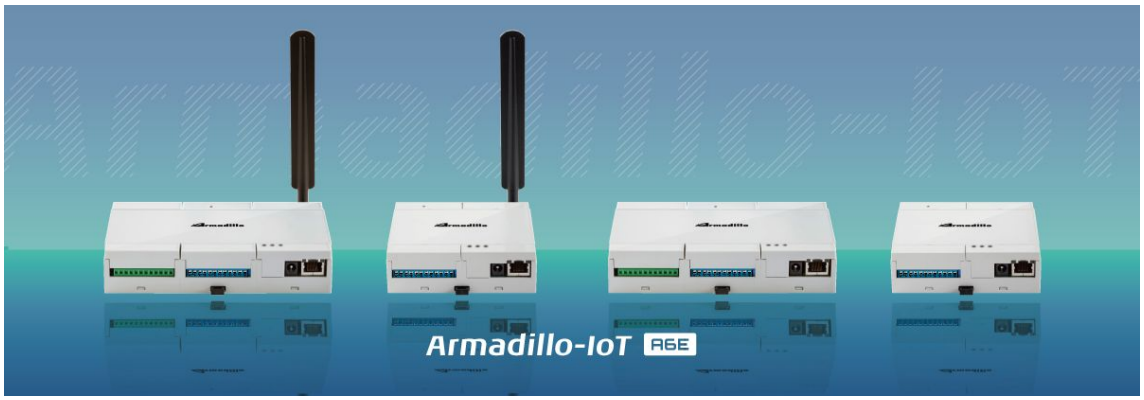


図 2.1 Armadillo-IoT ゲートウェイ A6E とは

- 省電力モード搭載・バッテリー駆動の機器に最適

  省電力モードを搭載し、「アプリケーションから本体の電源を OFF にする」「RTC(リアルタイムクロック)のアラームで決まった時間に本体の電源を ON にする」といった細かな電源制御や間欠動作が可能です。

  必要な時だけ本体を起動できるため、バッテリー駆動の機器に適しています。

- RS-485 や接点入出力、アナログ入力を搭載

  LAN、USB2.0 のインターフェースに加え、多くの事例で利用されている RS-485 シリアル通信(半二重)、接点入力 2ch、接点出力 2ch を標準搭載しました。また、アナログ入力 4ch、接点入力を 8ch 追加した製品も標準ラインアップに含まれています。

  センサーや機器とすぐに接続可能で、ハードウェアを拡張開発することなく、様々な製品を作ることができます。

- コンテナ型の Armadillo Base OS を搭載し、差分アップデートにも対応

  Linux をベースとした Armadillo Base OS (ABOS)は、コンパクトでセキュリティリスクが抑えられたコンテナアーキテクチャーの OS であり、標準でソフトウェアアップデート機能を有しています。アプリケーションソフトウェアはコンテナ上で動作し、コンテナのアップデートで新機能の追加やセキュリティ更新をすることができます。また差分アップデート機能にも対応しているため、アップデート時の通信容量を抑えることができます。

- 各種クラウド IoT サービスに対応したゲートウェイコンテナを提供

  各種クラウド IoT サービス(Azure IoT や AWS IoT Core)に対応したゲートウェイコンテナを用意しました。従来のモデルでは、ユーザー自身が開発するアプリケーションソフトウェア上で、ゲートウェイとしての機能の他、通信障害時の対応、セキュリティ対応、間欠動作時の挙動などの難しい課題を自ら解決する必要がありました。

  あらかじめ用意されたゲートウェイコンテナを活用することで、これらの課題に対処することができ、短期間に IoT システムを構築可能です。



図 2.2 様々なデバイスとの接続例

- 用途に合わせて複数のラインアップを用意

  Armadillo-IoT ゲートウェイ A6E は、モバイル通信モジュールを搭載し、WLAN モジュールの搭載有無を選べる「Cat.1 bis モデル」、WLAN モジュールのみを搭載した「WLAN モデル」、モバイル通信モジュールと WLAN を非搭載とした最もシンプルな「LAN モデル」をラインアップしています。

2.1.3. Armadillo Base OS とは

Armadillo Base OS は、アットマークテクノが提供する専用ディストリビューションです。Linux5.10 をベースに、コンテナ管理機能、ソフトウェアアップデート機能、ネットワークマネージャーなどに対応。機能を限定したコンパクトな OS で、安全性の高い運用を実現します。

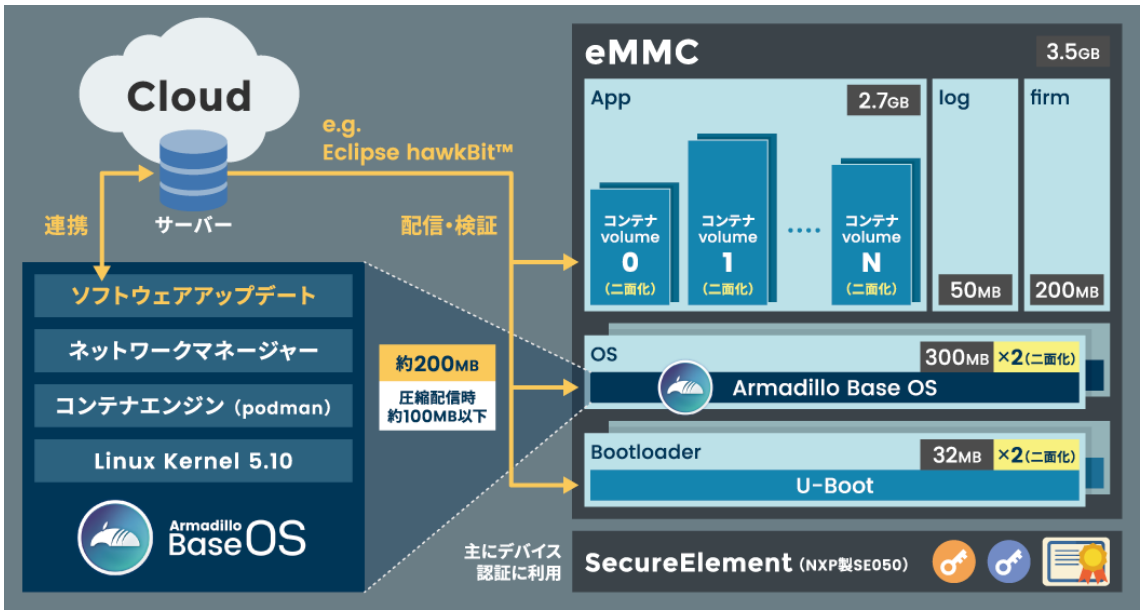


図 2.3 Armadillo Base OS とは

- OS のコンパクト化

  OS 基盤の機能を最小限にしたことで、セキュリティリスクを低減しています。アットマークテクノが継続的にアップデートを提供するため、高セキュリティな IoT 機器として長期間に渡り運用することができます。

- コンテナによるアプリケーション運用

  アプリケーションを「コンテナ」単位で OS から分離して管理できるため、コンテナごとのアップデートが可能です。サンドボックス化されることにより、悪意あるソフトウェアからの攻撃に対する機器全体の保護に有効性を発揮します。

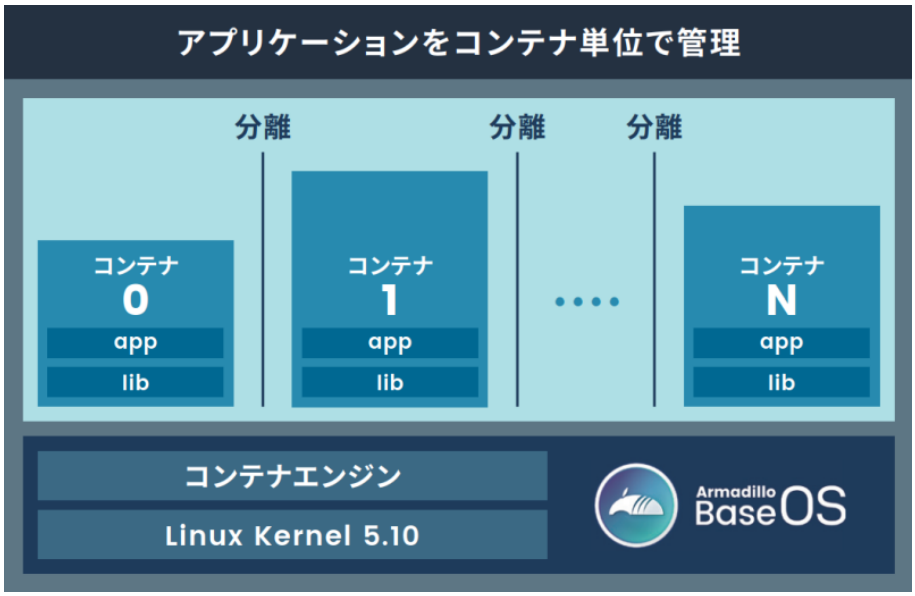


図 2.4 コンテナによるアプリケーションの運用

- アップデート機能を標準搭載

ネットワークや USB メモリ、microSD カード、Armadillo Twin によるアップデート機能を標準搭載しています。正しく署名されたソフトウェアのみアップデートできる仕組みや、差分アップデート機能も用意されています。OS・ブートローダー・コンテナ部分は、安全性を担保するため二面化し、リカバリー機能を備えています。万が一アップデートに失敗した場合でも、作業前の状態にロールバックすることができます。

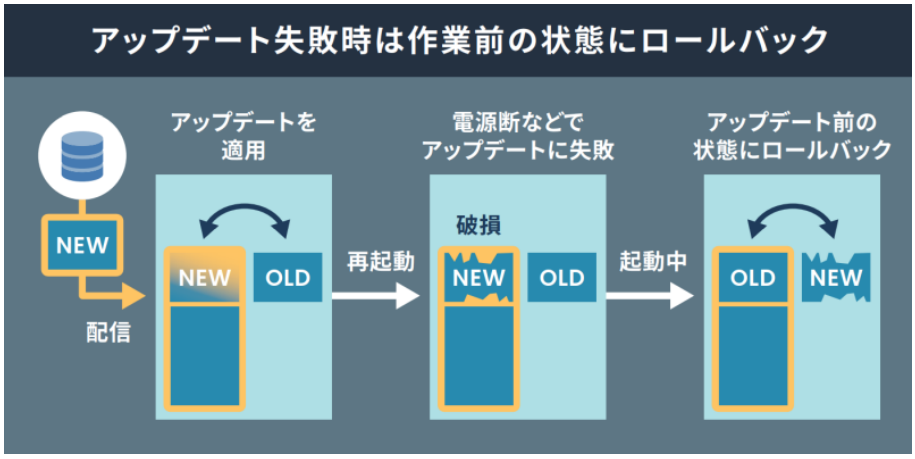


図 2.5 ロールバックの仕組み

- 堅牢性の向上

  安定性の高いファイルシステムで、ストレージへの書込みを減らして消耗を抑制するなど、高い堅牢性を有します。運用ログの記録機能も標準搭載しています。

- セキュリティ機能の向上

  コンテナにアクセス権限を設けて管理することができます。デバイス証明に利用できるセキュアエレメントを搭載するほか、セキュア環境「OP-TEE[1][2]」を利用可能な状態で提供しています。

2.1.4. Armadillo Base OS のメンテナンスポリシーとアップデートの推奨

Armadillo Base OS は Armadillo Base OS 搭載製品のサポート期限 [https://armadillo.atmark-techno.com/abos-support-timeline]にしたがって、アットマークテクノがセキュリティアップデートの提供、既存機能のバグ修正、今はない便利な機能の追加を継続的に行い、ユーザービリティの向上に努めます。緊急時を除き月末に "製品アップデート" としてこれらをリリースをし、Armadillo サイトから通知、変更内容の公開を行います。ユーザー登録を行うことで通知をメールで受け取ることもできます。

Armadillo を IoT 機器としてネットワークに接続し長期に運用を行う場合、継続的に最新バージョンを使用することを強く推奨いたします。最新バージョンを使用しない場合の注意点については「1.2.4. ソフトウェア使用に関しての注意事項」の「ソフトウェアのアップデートについて」を参照してください。

Armadillo サイト 製品アップデート

https://armadillo.atmark-techno.com/news/software-updates

[1]Arm コア向け TEE(Trusted Execution Environment)実装の一つです (https://optee.readthedocs.io/en/latest/)
[2]ソースコードは imx-boot の imx-optee-os ディレクトリに入っています

2.1.4.1. 後方互換性について

Armadillo Base OS は、原則、abos-ctrl コマンド等の各種機能や、sysfs ノード、コンテナ制御をするための podman コマンド等の API 後方互換を維持します。また、Armadillo Base OS とコンテナ間でサンドボックス化されていることもあり、互いの libc 等のライブラリや、各種パッケージなどの組み合わせによって互換性の問題は発生しません。このため、Armadillo Base OS をアップデートしても、これまで利用していたアプリケーションコンテナは原則的にそのまま起動・動作させることができます。

しかし、Armadillo Base OS 内の Linux-Kernel や alpine パッケージ変更によって、細かな動作タイミングが変更になる場合があるため、タイミングに大きく依存するようなアプリケーションをコンテナ内部に組み込んでいた場合に、動作に影響を与える可能性があります。まずは、テスト環境で Armadillo Base OS 更新を行い、アプリケーションコンテナと組み合わせた評価を行った後、市場で動作している Armadillo に対してアップデートを行うことを推奨します。

製品開発を開始するにあたり、Armadillo Base OS に関してより詳細な情報が必要な場合は、「3.3. 開発前に知っておくべき Armadillo Base OS の機能・特徴」 を参照してください。

2.1.5. Armadillo Twin とは

Armadillo Twin は、アットマークテクノが提供するクラウドサービスです。 Armadillo Base OS 搭載のデバイスを、リモートから運用管理することができます。様々なタスクをリモートから実行できるようになり、 OS アップデートもサービス画面からの操作で行えるため、稼働中のデバイスは常に最新の状態を維持することができます。また、バグ修正やセキュリティ対策などのメンテナンスのほか、機能追加や設定変更、アプリケーションのアップデートなども行えるため、デバイスの設置現場に出向くことなく、計画的で効率的な DevOps を実現することができます。

本書では、開発・量産・運用の各フェーズにおける Armadillo Twin の利用について記載しています。



図 2.6 Armadillo Twin　とは

2.1.5.1. サービスの特徴

- ソフトウェアアップデート (OTA)

  遠隔からデバイスのソフトウェアアップデートをすることで、長期的にセキュリティ性の高いシステムを保つと共に、新たな機能を提供することも可能です。本サービスで管理するデバイスに搭載されている Armadillo Base OS は、不正なソフトウェアへのアップデートを行わせない署名検証機能や、アップデートが失敗した際に自動で元の状態に戻るロールバック機能を備えています。そのため、安心してソフトウェアアップデートを利用することができます。

- 遠隔稼働監視

  登録されたデバイスの死活監視をはじめ、CPU の使用率や温度、メモリの使用量、モバイル回線の電波状況、ストレージの空き容量や寿命を監視することができます。各値にはアラートの設定を

行うことができ、異常を検知した場合はアラートメールを管理者に送信します。メールを受けた管理者は本サービスの遠隔操作機能を利用し、即座に対応を行うことができるため、システムの安定運用を行うことができます。そのほか、本サービスに登録したデバイスは、自由にラベル名を付けたりグループを作成して管理することができるため、どのデバイスをどの場所に設置したか画面上で把握することが容易になります。また、デバイス本体に搭載されているセキュアエレメントを利用した個体認証により、不正なデバイスの登録を防ぎます。

・遠隔操作

画面上で入力した任意のコマンドをデバイス上で実行することができます。本サービスは遠隔操作で一般的に使われる SSH(Secure Shell) のように固定グローバル IP アドレスの設定は不要です。そのため、通信回線の契約料金を安くできるだけではなく、インターネット上からのサイバー攻撃のリスクを抑制する効果も期待できます。任意のコマンドは単一のデバイスだけではなく、グループ単位、また複数のデバイスを選択して一括して実行したり、時刻を指定するスケジュール実行にも対応しています。

2.1.5.2. 提供する機能一覧

Armadillo Twin は、下記の機能を提供します。

遠隔稼働監視	死活監視、アプリケーションコンテナ稼働状況、 CPU 使用率・温度/メモリ使用率、ストレージ寿命、モバイル回線電波強度、モバイル回線基地局の位置情報[a]、アラートメール
遠隔操作	ソフトウェアアップデート(OTA)、任意コマンド実行、ソフトウェアバージョン確認、設定変更、グループ一括実行、スケジュール実行
個体管理	デバイス登録(デバイス証明書を利用)、ラベル付け、デバイスグループ化機能
ユーザ管理	ユーザーの追加/削除、ユーザー権限の設定
お知らせ	セキュリティアップデート、システム障害通知

[a]サービス開始時には非対応の機能です。今後のアップデートで対応予定です。

> **Armadillo Twin**
>
> サービス利用料金など、その他詳細については下記概要ページをご覧ください。
>
> https://armadillo.atmark-techno.com/guide/armadillo-twin

2.2. 製品ラインアップ

LAN、USB2.0 のインターフェース、RS-485 シリアル通信（半二重)、接点入力 2ch、接点出力 2ch を搭載したスタンダードタイプと、アナログ入力 4ch、接点入力 8ch を追加した+Di8+Ai4 タイプをラインアップしています。

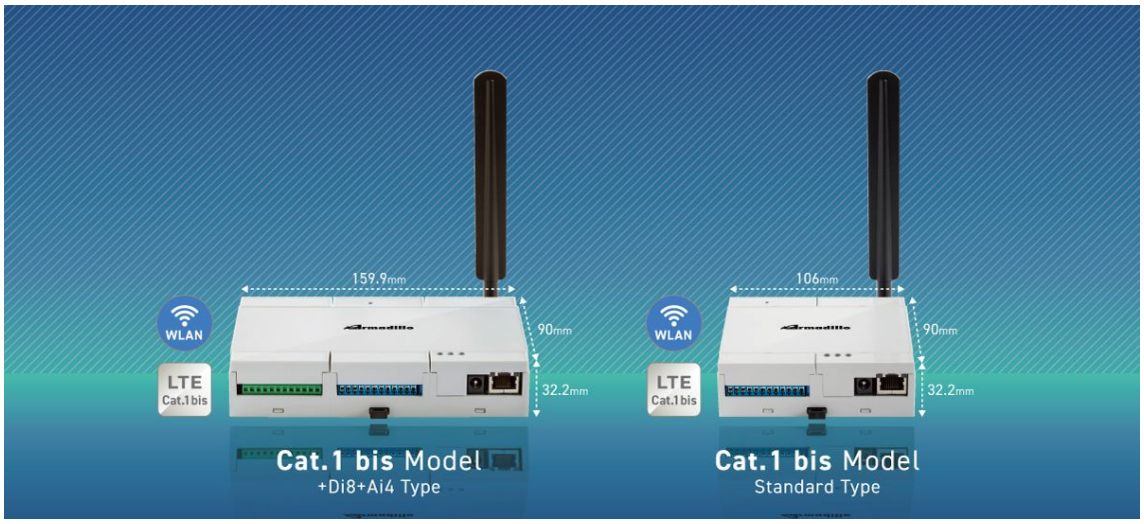


図 2.7 スタンダートタイプと+Di8+Ai4 タイプ

スタンダードタイプと+Di8+Ai4 タイプのそれぞれに、モバイル通信モジュールを搭載し、WLAN モジュールの搭載有無を選べる「Cat.1 bis モデル」、WLAN モジュールのみを搭載した「WLAN モデル」、モバイル通信モジュールと WLAN を非搭載とした「LAN モデル」を用意しています。

設計開発時には、開発に必要なものを一式含んだ「開発セット」、量産時には、必要最小限のセット内容に絞った「量産用」もしくは「量産ボード」をご購入ください。「量産用」はケース有、「量産ボード」はケース無となります。

表 2.1 スタンダードタイプの型番と搭載モジュールおよび付属品一覧

モデル	型番	LTE モジュール	LTE アンテナ	WLAN モジュール	WLAN アンテナ	ケース	その他付属品 [a]
Cat.1 bis +WLAN	AG6251-C03D0	○	○	○	○	○	○
	AG6251-C03Z	○	○	○	○	○	×
	AG6251-U03Z	○	○	○	○	×	×
	AG6251-U00Z	○	×	○	×	×	×
Cat.1 bis	AG6241-C01Z	○	○	×	×	○	×
	AG6241-U01Z	○	○	×	×	×	×
	AG6241-U00Z	○	×	×	×	×	×
WLAN	AG6211-C02D0	×	×	○	○	○	○
	AG6211-C02Z	×	×	○	○	○	×
	AG6211-U02Z	×	×	○	○	×	×
	AG6211-U00Z	×	×	○	×	×	×
LAN	AG6201-C00Z	×	×	×	×	○	×
	AG6201-U00Z	×	×	×	×	×	×

[a]AC アダプタ(12V/2.0A)、USB(A オス-microB)ケーブル

表 2.2 +Di8+Ai4 タイプの型番と搭載モジュールおよび付属品一覧

モデル	型番	LTE モジュール	LTE アンテナ	WLAN モジュール	WLAN アンテナ	ケース	その他付属品 [a]
Cat.1 bis +WLAN	AG6253-C03D0	○	○	○	○	○	○
	AG6253-C03Z	○	○	○	○	○	×
Cat.1 bis	AG6243-C01Z	○	○	×	×	○	×
WLAN	AG6213-C02D0	×	×	○	○	○	○
	AG6213-C02Z	×	×	○	○	○	×
LAN	AG6203-C00Z	×	×	×	×	○	×

2.2.1. Armadillo-IoT ゲートウェイ A6E 開発セット

Armadillo-IoT ゲートウェイ A6E の開発セットは、Armadillo-IoT ゲートウェイ A6E 本体に加え、AC アダプタや USB ケーブルなど、開発に必要なものが一式揃った、設計開発用のラインアップです。

モバイル通信が必要な場合は「Cat.1 bis モデル」、モバイル通信が不要な場合は WLAN のみの「WLAN モデル」を推奨いたします。

表 2.3 Armadillo-IoT ゲートウェイ A6E 開発セット一覧 (スタンダードタイプ)

名称	型番
Armadillo-IoT ゲートウェイ A6E Cat.1 bis+WLAN モデル 開発セット	AG6251-C03D0
Armadillo-IoT ゲートウェイ A6E WLAN モデル 開発セット	AG6211-C02D0

表 2.4 Armadillo-IoT ゲートウェイ A6E 開発セット一覧 (+Di8+Ai4 タイプ)

名称	型番
Armadillo-IoT ゲートウェイ A6E Cat.1 bis+WLAN モデル +Di8+Ai4 開発セット	AG6253-C03D0
Armadillo-IoT ゲートウェイ A6E WLAN モデル +Di8+Ai4 開発セット	AG6213-C02D0

開発セットの内容物は以下の通りです。

表 2.5 Armadillo-IoT ゲートウェイ A6E 開発セットの内容物

Armadillo-IoT ゲートウェイ A6E 本体
LTE 外付けアンテナ [a]
WLAN 基板アンテナ
USB(A オス-microB)ケーブル
AC アダプタ(12V/2.0A)

[a]Cat.1 bis モデルの場合

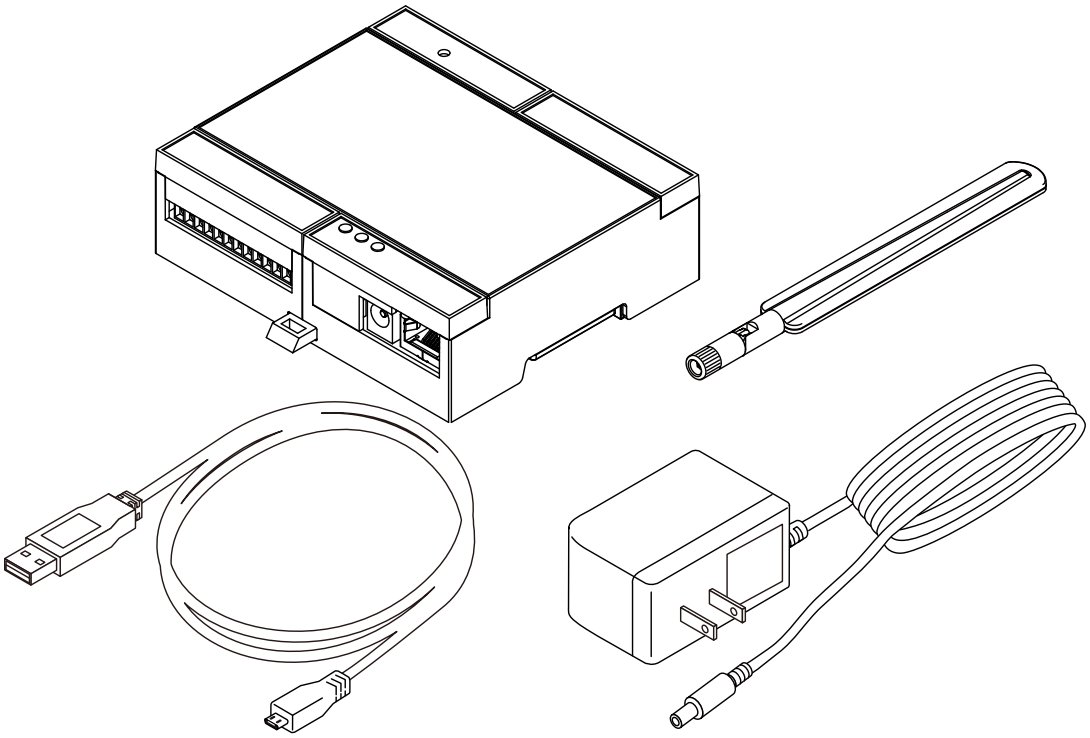


図 2.8 Armadillo-IoT ゲートウェイ A6E 開発セット (スタンダードタイプ)

2.2.2. Armadillo-IoT ゲートウェイ A6E 量産用

Armadillo-IoT ゲートウェイ A6E 量産用は、ケースに収めた Armadillo-IoT ゲートウェイ A6E 本体とアンテナのみに内容物を絞った、量産向けのラインアップです。

表 2.6 Armadillo-IoT ゲートウェイ A6E 量産用一覧 (スタンダードタイプ)

名称	型番
Armadillo-IoT ゲートウェイ A6E Cat.1 bis+WLAN モデル 量産用 (LTE アンテナセット付属、WLAN コンボ搭載、WLAN 基板アンテナ付属)	AG6251-C03Z
Armadillo-IoT ゲートウェイ A6E Cat.1 bis モデル 量産用 (LTE アンテナセット付属、WLAN コンボ非搭載)	AG6241-C01Z
Armadillo-IoT ゲートウェイ A6E WLAN モデル 量産用 (WLAN コンボ搭載、WLAN 基板アンテナ付属)	AG6211-C02Z
Armadillo-IoT ゲートウェイ A6E LAN モデル 量産用	AG6201-C00Z

表 2.7 Armadillo-IoT ゲートウェイ A6E 量産用一覧 (+Di8+Ai4 タイプ)

名称	型番
Armadillo-IoT ゲートウェイ A6E Cat.1 bis+WLAN モデル +Di8+Ai4 量産用 (LTE アンテナセット付属、WLAN コンボ搭載、WLAN 基板アンテナ付属)	AG6253-C03Z
Armadillo-IoT ゲートウェイ A6E Cat.1 bis モデル +Di8+Ai4 量産用 (LTE アンテナセット付属、WLAN コンボ非搭載)	AG6243-C01Z
Armadillo-IoT ゲートウェイ A6E WLAN モデル +Di8+Ai4 量産用 (WLAN コンボ搭載、WLAN 基板アンテナ付属)	AG6213-C02Z
Armadillo-IoT ゲートウェイ A6E LAN モデル +Di8+Ai4 量産用	AG6203-C00Z

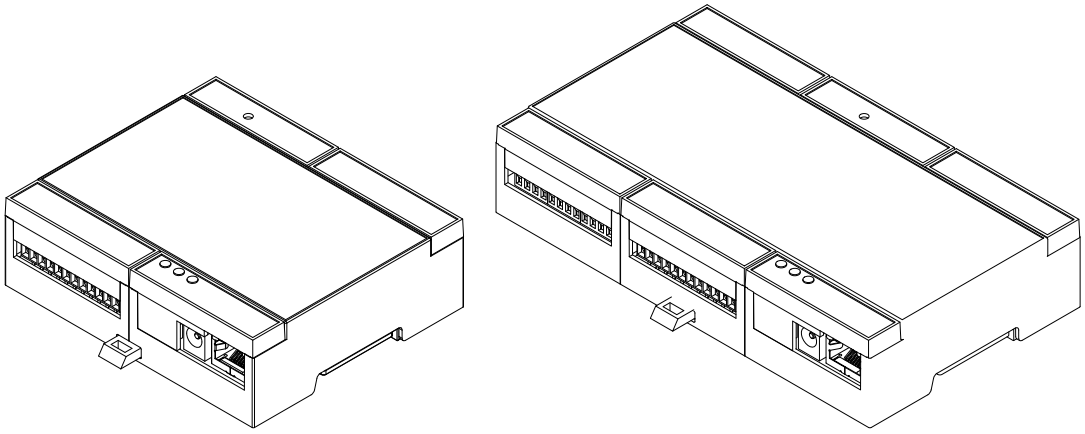


図 2.9 Armadillo-IoT ゲートウェイ A6E 量産用

2.2.3. Armadillo-IoT ゲートウェイ A6E 量産ボード

Armadillo-IoT ゲートウェイ A6E 量産ボードは、ケース無しの Armadillo-IoT ゲートウェイ A6E 本体とアンテナのみに内容物を絞った、量産向けのラインアップです。オリジナルのケースを使用する等、ケースが不要の場合はこちらをご検討ください。

表 2.8 Armadillo-IoT ゲートウェイ A6E 量産ボード一覧

名称	型番
Armadillo-IoT ゲートウェイ A6E Cat.1 bis+WLAN モデル 量産ボード (LTE アンテナセット付属、WLAN コンボ搭載、WLAN 基板アンテナ付属)	AG6251-U03Z
Armadillo-IoT ゲートウェイ A6E Cat.1 bis+WLAN モデル 量産ボード (LTE アンテナセット無、WLAN コンボ搭載、WLAN 基板アンテナ無)	AG6251-U00Z
Armadillo-IoT ゲートウェイ A6E Cat.1 bis モデル 量産ボード (LTE アンテナセット付属、WLAN コンボ非搭載)	AG6241-U01Z
Armadillo-IoT ゲートウェイ A6E Cat.1 bis モデル 量産ボード (LTE アンテナセット無、WLAN コンボ非搭載)	AG6241-U00Z
Armadillo-IoT ゲートウェイ A6E WLAN モデル 量産ボード (WLAN 基板アンテナ付属)	AG6211-U02Z
Armadillo-IoT ゲートウェイ A6E WLAN モデル 量産ボード (WLAN 基板アンテナ無)	AG6211-U00Z
Armadillo-IoT ゲートウェイ A6E LAN モデル 量産ボード	AG6201-U00Z

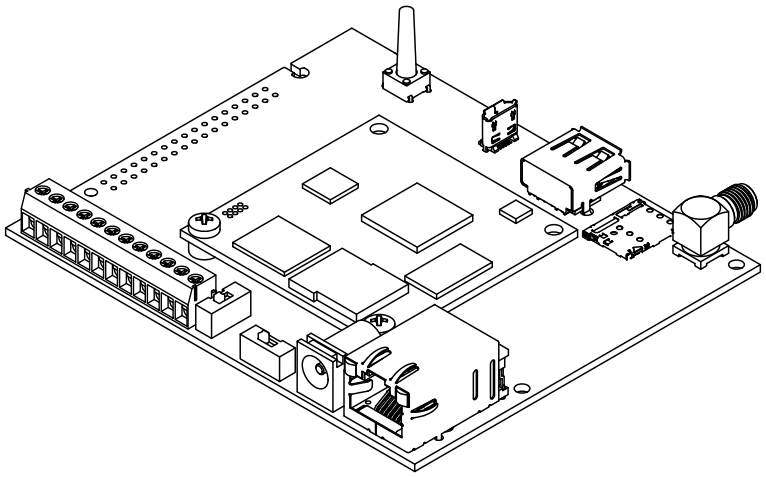


図 2.10 Armadillo-IoT ゲートウェイ A6E 量産ボード

2.2.4. Armadillo-IoT ゲートウェイ A6E オプション品

オプション品として、以下の製品を提供しています。

表 2.9 Armadillo-IoT ゲートウェイ A6E のオプション品一覧

名称	型番
AC アダプタ（12V/2.0A φ2.1mm）温度拡張品 効率レベル VI 品	OP-AC12V4-00
Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)	OP-CASEA6E-PLA-20
Armadillo-IoT ゲートウェイ A6E DI8AI4 拡張ボードセット 00	-[a]

[a]アットマークテクノ 営業部までお問い合わせください。

2.2.5. Armadillo-IoT ゲートウェイ A6E BTO サービス

セミオーダー式メニューから選択して Armadillo-IoT ゲートウェイ A6E の量産品を一括手配いただける有償サービスを提供しています。



図 2.11 Armadillo-IoT ゲートウェイ A6EBTO サービス

標準ラインアップ品の仕様をベースとして、搭載するモジュールの種類やケース、AC アダプタの有無、部品実装の一部変更、お客様支給品の SD カードや SIM カードの接続、お客様ご指定のソフトウェアイメージ書き込みなど、メニュー内から指定可能なキッティング項目を選択・指定することが可能です。販売代理店またはアットマークテクノの窓口からお申し込みいただけます。

詳しくは Armadillo サイトの 「アットマークテクノ BTO サービス」 [https://armadillo.atmark-techno.com/services/customize/bto] をご確認ください。

2.3. 仕様

2.3.1. スタンダードタイプ

Armadillo-IoT ゲートウェイ A6E スタンダードタイプの主な仕様を「表 2.10. 仕様 (スタンダートタイプ)」に示します。

表 2.10 仕様 (スタンダートタイプ)

モデル名	Cat.1 bis+WLAN	Cat.1 bis	WLAN	LAN
プロセッサ	NXP Semiconductors i.MX6ULL			
	ARM Cortex-A7 x 1 ・命令/データキャッシュ 32KByte/32KByte ・L2 キャッシュ 128KByte ・内部 SRAM 128KByte ・メディアプロセッシングエンジン(NEON)搭載 ・Thumb code(16bit 命令セット)サポート			
システムクロック	CPU コアクロック(ARM Cortex-A7): 528MHz DDR クロック: 396MHz 源発振クロック: 32.768kHz, 24MHz			
RAM	DDR3L: 512MByte バス幅: 16bit			
ROM	eMMC: 3.5GB [a]			
LAN(Ethernet)	100BASE-TX/10BASE-T x 1 AUTO-MDIX 対応			
無線 LAN/Bluetooth®	WLAN+BT コンボモジュール IEEE 802.11a/b/g/n/ac and Bluetooth® 5.2 [b] [c]	非搭載	WLAN+BT コンボモジュール IEEE 802.11a/b/g/n/ac and Bluetooth® 5.2 [b] [c]	非搭載
モバイル通信	LTE Cat.1 bis モジュール [d] SIM スロット: nanoSIM 対応		非搭載	
USB	USB 2.0 Host x 1 (High Speed)			
SD	microSD スロット x 1 [b] [e]			
入出力インターフェース	接点入力(電流シンク出力タイプに接続可能) x 2 接点出力(無極性) x 2			
シリアル(RS-485)	2 線式(Data+, Data-, GND) x 1 最大データ転送レート: 5Mbps 終端抵抗 120Ω 内蔵 [f]			
拡張インターフェース[g]	GPIO x 24、UART x 2、I2C x 1、SPI x 1、CAN x 2、I2S x 1、PWM x 4、A/D x 4、MQS x 1			
カレンダ時計	リアルタイムクロック搭載 バックアップ用電池 CR1220 接続可能 最大月差: 8 秒(周囲温度-20℃～60℃、経年変化除く)			
スイッチ	ユーザースイッチ x 1 設定用スイッチ x 2			
LED	SYS(Green) x 1 APP(Green) x 1 WWAN(Green) x 1 [h]			
メンテナンスポート	USB micro B シリアルコンソール			
セキュアエレメント	NXP Semiconductors SE050 搭載			
監視機能	入力電圧監視		非対応	
入力電源	DC 8～26.4V			

モデル名		Cat.1 bis+WLAN	Cat.1 bis	WLAN	LAN
消費電力	シャットダウン	約 4mW	約 3mW	約 3mW	約 3mW
	スリープ	約 150mW	約 140mW	約 130mW	約 130mW
	スリープ (SMS 起床可能)	約 175mW	約 165mW	非対応	非対応
	アクティブ	約 1250mW [i]	約 1000mW [i]	約 800mW	約 900mW [j]
	最大	約 3750mW [i]	約 2400mW [i]	約 1900mW	約 1300mW
動作温度範囲		-20〜+60℃ (結露なきこと)			
外形サイズ(基板)		103 x 87 mm (突起部、アンテナを除く)			
外形サイズ(ケース)		106.2 x 90 x 32.2 mm (突起部、アンテナを除く)			

[a]pSLC での数値です。出荷時 pSLC に設定しています。

[b]Cat.1 bis+WLAN モデルと WLAN モデルは、インストールディスク以外での SD 利用ができません。

[c]Bluetooth® の最大接続台数は Classic が 7 台、Bluetooth® Low Energy が 16 台です。

[d]モバイル通信を利用する時は、外付けアンテナを接続する必要があります。

[e]ケースに入れた状態で操作することはできません。

[f]ディップスイッチの操作で抵抗の切り離しが可能です。

[g]i.MX6ULL のピンマルチプクレスの設定で、優先的に機能を割り当てた場合に拡張可能な最大数を記載しています。

[h]WLAN モデルと LAN モデルは WWAN LED をユーザーが自由に使用することができます。

[i]LTE の signal quality が約 80%かつ周辺機器が未接続の時の参考値となります。電波環境や接続するデバイスにより消費電力は変化します。

[j]LAN ケーブル接続時

2.3.2. +Di8+Ai4 タイプ

Armadillo-IoT ゲートウェイ A6E +Di8+Ai4 タイプの主な仕様を「表 2.11. 仕様 (+Di8+Ai4 タイプ)」に示します。

表 2.11 仕様 (+Di8+Ai4 タイプ)

モデル名	Cat.1 bis+WLAN +Di8+Ai4	Cat.1 bis +Di8+Ai4	WLAN +Di8+Ai4	LAN +Di8+Ai4
プロセッサ	NXP Semiconductors i.MX6ULL			
	ARM Cortex-A7 x 1 ・命令/データキャッシュ 32KByte/32KByte ・L2 キャッシュ 128KByte ・内部 SRAM 128KByte ・メディアプロセッシングエンジン(NEON)搭載 ・Thumb code(16bit 命令セット)サポート			
システムクロック	CPU コアクロック(ARM Cortex-A7): 528MHz DDR クロック: 396MHz 源発振クロック: 32.768kHz, 24MHz			
RAM	DDR3L: 512MByte バス幅: 16bit			
ROM	eMMC: 3.5GB [a]			
LAN(Ethernet)	100BASE-TX/10BASE-T x 1 AUTO-MDIX 対応			
無線 LAN/Bluetooth®	WLAN+BT コンボモジュール IEEE 802.11a/b/g/n/ac and Bluetooth® 5.2 [b] [c]	非搭載	WLAN+BT コンボモジュール IEEE 802.11a/b/g/n/ac and Bluetooth® 5.2 [b] [c]	非搭載
モバイル通信	LTE Cat.1 bis モジュール [d] SIM スロット: nanoSIM 対応		非搭載	
USB	USB 2.0 Host x 1 (High Speed)			
SD	microSD スロット x 1 [b] [e]			

モデル名		Cat.1 bis+WLAN +Di8+Ai4	Cat.1 bis +Di8+Ai4	WLAN +Di8+Ai4	LAN +Di8+Ai4
入出力インターフェース		接点入力(電流シンク出力タイプに接続可能) x 10 [f] 接点出力(無極性) x 2 アナログ入力(分解能 12bit, 4-20mA/1-5V 入力に対応) x 4 [g] 外部電源制御出力 x 1 [h] [i]			
シリアル(RS-485)		2 線式(Data+, Data-, GND) x 1 最大データ転送レート: 5Mbps 終端抵抗 120Ω 内蔵 [j]			
拡張インターフェース		非対応			
カレンダ時計		リアルタイムクロック搭載 バックアップ用電池 CR1220 接続可能 最大月差: 8 秒(周囲温度-20℃～60℃、経年変化除く)			
スイッチ		ユーザースイッチ x 1 設定用スイッチ x 2			
LED		SYS(Green) x 1 APP(Green) x 1 WWAN(Green) x 1 [k]			
メンテナンスポート		USB micro B シリアルコンソール			
セキュアエレメント		NXP Semiconductors SE050			
監視機能		入力電圧監視			
入力電源		DC 8～26.4V			
消費電力	シャットダウン	約 4mW	約 3mW	約 3mW	約 3mW
	スリープ	約 200mW	約 190mW	約 180mW	約 180mW
	スリープ (SMS 起床可能)	約 225mW	約 215mW	非対応	非対応
	アクティブ	約 1300mW [l]	約 1050mW [l]	約 1200mW	約 950mW [m]
	最大	約 4000mW [l]	約 3000mW [l]	約 2150mW	約 1550mW
動作温度範囲		-20～+60℃ (結露なきこと)			
外形サイズ(基板)		156 x 87 mm (突起部、アンテナを除く)			
外形サイズ(ケース)		159.9 x 90 x 32.2 mm (突起部、アンテナを除く)			

[a]pSLC での数値です。出荷時 pSLC に設定しています。
[b]Cat.1 bis+WLAN モデルと WLAN モデルは、インストールディスク以外での SD 利用ができません。
[c]Bluetooth® の最大接続台数は Classic が 7 台、Bluetooth® Low Energy が 16 台です。
[d]モバイル通信を利用する時は、外付けアンテナを接続する必要があります。
[e]ケースに入れた状態で操作することはできません。
[f]+Di8+Ai4 拡張基板をカスケード接続することで最大 34 ポートまで増設が可能
[g]+Di8+Ai4 拡張基板をカスケード接続することで最大 16 ポートまで増設が可能
[h]デフォルトではシャットダウン時 OFF になり、アクティブ、スリープ時に ON になる無電圧接点出力です。
[i]+Di8+Ai4 拡張基板をカスケード接続することで最大 4 ポートまで増設が可能
[j]ディップスイッチの操作で抵抗の切り離しが可能です。
[k]WLAN モデルと LAN モデルは WWAN LED をユーザーが自由に使用することができます。
[l]LTE の signal quality が約 80%かつ周辺機器が未接続の時の参考値となります。電波環境や接続するデバイスにより消費電力は変化します。
[m]LAN ケーブル接続時

2.4. インターフェースレイアウト

2.4.1. スタンダードタイプ

Armadillo-IoT ゲートウェイ A6E スタンダードタイプのインターフェースレイアウト [3]です。一部のインターフェースを使用する際には、ケースを開ける必要があります。

[3]製品型番により、搭載部品が異なります。

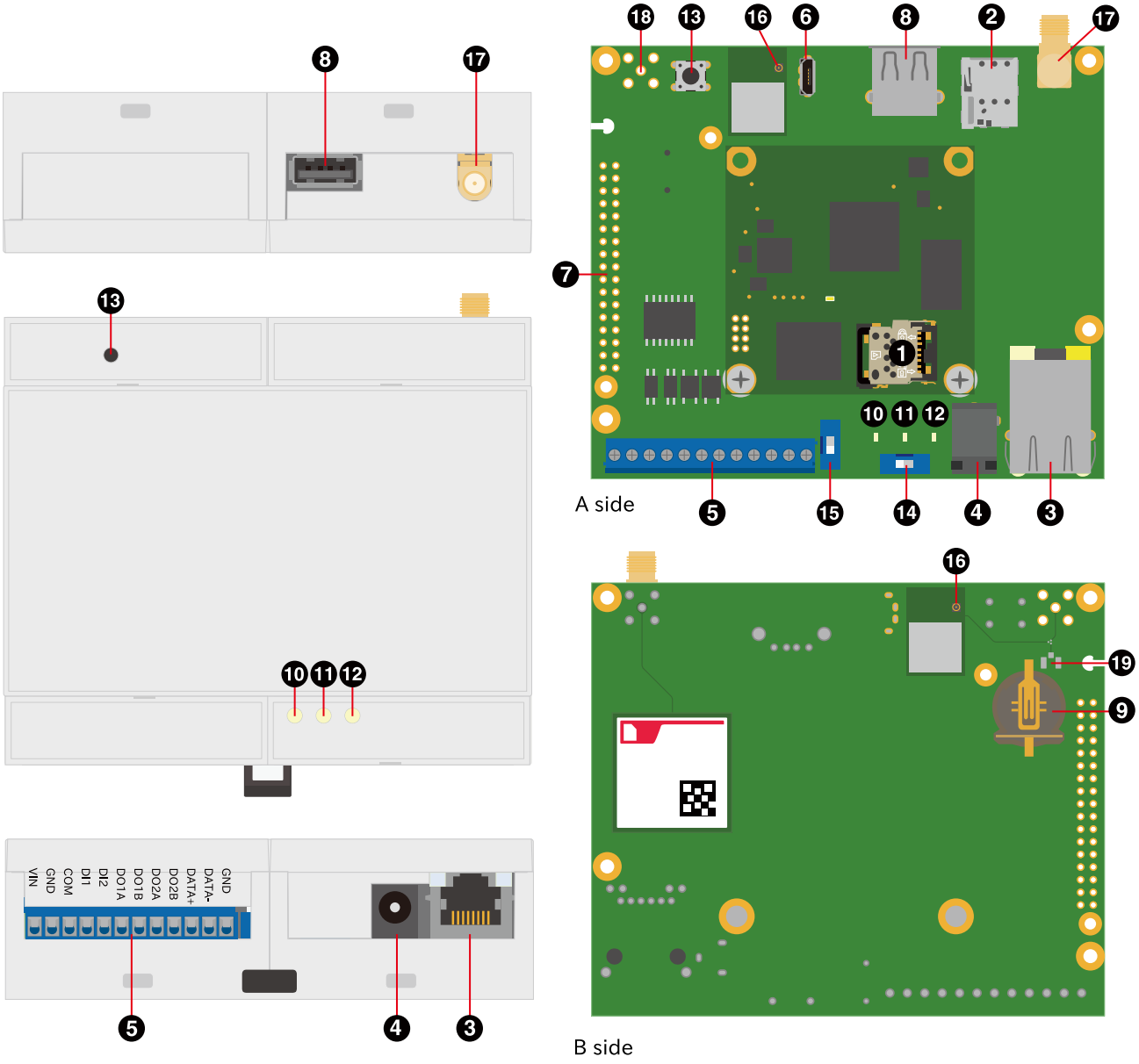


図 2.12 Armadillo-IoT ゲートウェイ A6E インターフェースレイアウト (スタンダードタイプ)

表 2.12 各部名称と機能

番号	名称	形状	説明
1	SD インターフェース	microSD スロット	外部ストレージが必要な場合 [a]や、ブートローダーを破壊してしまった時の復旧等で使用します。microSD カードを挿入します。
2	nanoSIM インターフェース	nanoSIM スロット	LTE データ通信を利用する場合に使用します。nanoSIM カードを挿入します。
3	LAN インターフェース	RJ-45 コネクタ	有線 LAN を利用する場合に使用します。LAN ケーブルを接続します。
4	電源入力インターフェース	DC ジャック	本体への電源供給で使用します [b]。付属の AC アダプタ(12V/2A)を接続します。対応プラグ:内径 2.1mm、外形 5.5mm
5	入出力インターフェース	端子台	絶縁入出力、RS-485 通信する場合に使用します。本体への電源供給も可能です [b]。

番号	名称	形状	説明
6	USB コンソールインターフェース	USB micro B コネクタ	コンソール入出力を利用する場合に使用します。USB micro B ケーブルを接続します。
7	拡張インターフェース	ピンヘッダ 34 ピン(2.54mm ピッチ)	機能拡張する場合に使用します。2.54mm ピッチのピンヘッダを実装することができます。
8	USB インターフェース	USB 2.0 Type-A コネクタ	外部ストレージが必要な場合等に使用します。USB メモリ等を接続します。
9	RTC バックアップインターフェース	電池ボックス	リアルタイムクロックのバックアップ給電が必要な場合に使用します。対応電池: CR1220 等
10	システム LED	LED(緑色、面実装)	電源の入力状態を表示する緑色 LED です。
11	アプリケーション LED	LED(緑色、面実装)	アプリケーションの状態を表示する緑色 LED です。
12	ワイヤレス WAN LED	LED(緑色、面実装)	LTE 通信の状態を表示する緑色 LED です。
13	ユーザースイッチ	タクトスイッチ	ユーザーが利用可能なタクトスイッチです。
14	起動デバイス設定スイッチ	DIP スイッチ	起動デバイスを設定する時に使用します。
15	RS-485 終端抵抗設定スイッチ	DIP スイッチ	RS-485 通信の終端抵抗を設定する時に使用します。
16	WLAN/BT アンテナインターフェース	MHF4 コネクタ	WLAN/Bluetooth®のデータ通信を利用する場合に使用します。付属の WLAN アンテナを接続します。 [c]
17	LTE アンテナインターフェース	SMA コネクタ	LTE データ通信を利用する場合に使用します。付属の LTE アンテナを接続します。
18	アンテナ中継インターフェース	SMA コネクタ	アンテナを外付けする場合に使用します。
19	アンテナインターフェース	U.FL コネクタ	アンテナを外付けする場合に使用します。

[a]Cat.1 bis+WLAN モデルと WLAN モデルは、SD をストレージとして使用できません。

[b]DC ジャックと端子台の両方から同時に電源供給することはできません。

[c]Cat.1 bis+WLAN モデルの場合は A 面側、WLAN モデルの場合は B 面側に搭載されます。

2.4.2. +Di8+Ai4 タイプ

Armadillo-IoT ゲートウェイ A6E +Di8+Ai4 タイプのインターフェースレイアウト [4]です。一部のインターフェースを使用する際には、ケースを開ける必要があります。

[4]製品型番により、搭載部品が異なります。

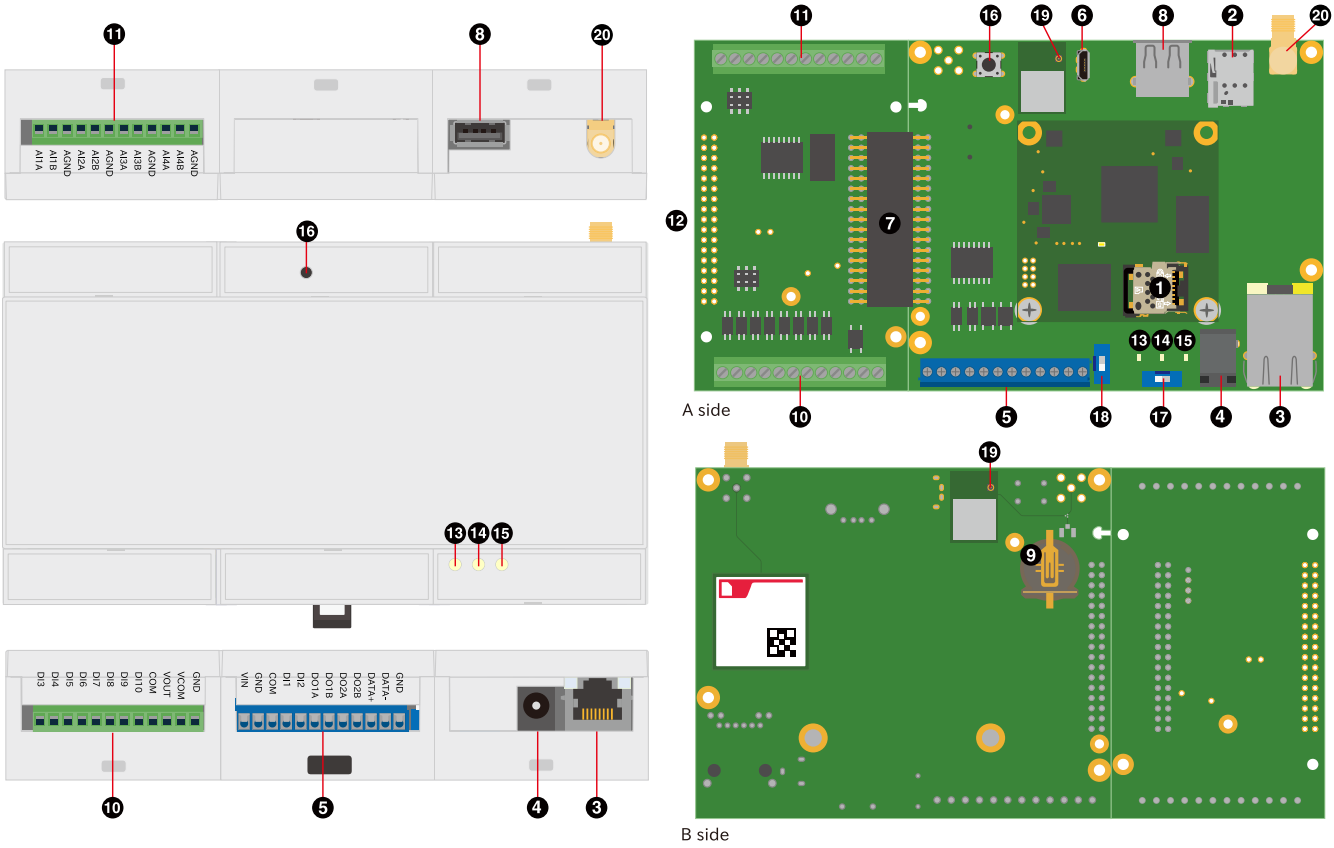


図 2.13 Armadillo-IoT ゲートウェイ A6E インターフェースレイアウト (+Di8+Ai4 タイプ)

表 2.13 各部名称と機能

番号	名称	形状	説明
1	SD インターフェース	microSD スロット	外部ストレージが必要な場合[a]や、ブートローダーを破壊してしまった時の復旧等で使用します。microSD カードを挿入します。
2	nanoSIM インターフェース	nanoSIM スロット	LTE データ通信を利用する場合に使用します。nanoSIM カードを挿入します。
3	LAN インターフェース	RJ-45 コネクタ	有線 LAN を利用する場合に使用します。LAN ケーブルを接続します。
4	電源入力インターフェース	DC ジャック	Armadillo-IoT ゲートウェイ A6E への電源供給で使用します [b]。付属の AC アダプタ (12V/2A)を接続します。
5	入出力インターフェース	端子台	絶縁入出力、RS-485 通信する場合に使用します。Armadillo-IoT ゲートウェイ A6E への電源供給も可能です [b]。
6	USB コンソールインターフェース	USB micro B コネクタ	コンソール入出力を利用する場合に使用します。USB micro B ケーブルを接続します。
7	拡張インターフェース	ピンヘッダ 34 ピン(2.54mm ピッチ)	+Di8+Ai4 拡張基板と接続しています。
8	USB インターフェース	USB 2.0 Type-A コネクタ	外部ストレージが必要な場合等に使用します。USB メモリ等を接続します。
9	RTC バックアップインターフェース	電池ボックス	リアルタイムクロックのバックアップ給電が必要な場合に使用します。対応電池: CR1220 等
10	入出力インターフェース	端子台	絶縁入力、外部電源制御出力を利用する場合に使用します。
11	アナログ入力インターフェース	端子台	アナログ入力を利用する場合に使用します。

番号	名称	形状	説明
12	拡張インターフェース	ピンヘッダ 34 ピン(2.54mm ピッチ)	+Di8+Ai4 拡張基板をカスケード接続する場合に使用します。
13	システム LED	LED(緑色、面実装)	電源の入力状態を表示する緑色 LED です。
14	アプリケーション LED	LED(緑色、面実装)	アプリケーションの状態を表示する緑色 LED です。
15	ワイヤレス WAN LED	LED(緑色、面実装)	LTE 通信の状態を表示する緑色 LED です。
16	ユーザースイッチ	タクトスイッチ	ユーザーが利用可能なタクトスイッチです。
17	起動デバイス設定スイッチ	DIP スイッチ	起動デバイスを設定する時に使用します。
18	RS-485 終端抵抗設定スイッチ	DIP スイッチ	RS-485 通信の終端抵抗を設定する時に使用します。
19	WLAN/BT アンテナインターフェース	MHF4 コネクタ	WLAN/Bluetooth®のデータ通信を利用する場合に使用します。付属の WLAN 基板アンテナを接続します。 [c]
20	LTE アンテナインターフェース	SMA コネクタ	LTE データ通信を利用する場合に使用します。付属の LTE アンテナを接続します。

[a]Cat.1 bis+WLAN モデルと WLAN モデルは、SD をストレージとして使用できません。

[b]DC ジャックと端子台の両方から同時に電源供給することはできません。

[c]Cat.1 bis+WLAN モデルの場合は A 面側、WLAN モデルの場合は B 面側に搭載されます。

2.5. ブロック図

2.5.1. スタンダードタイプ

Armadillo-IoT ゲートウェイ A6E スタンダードタイプのブロック図です。モデルにより、一部機能が異なります。

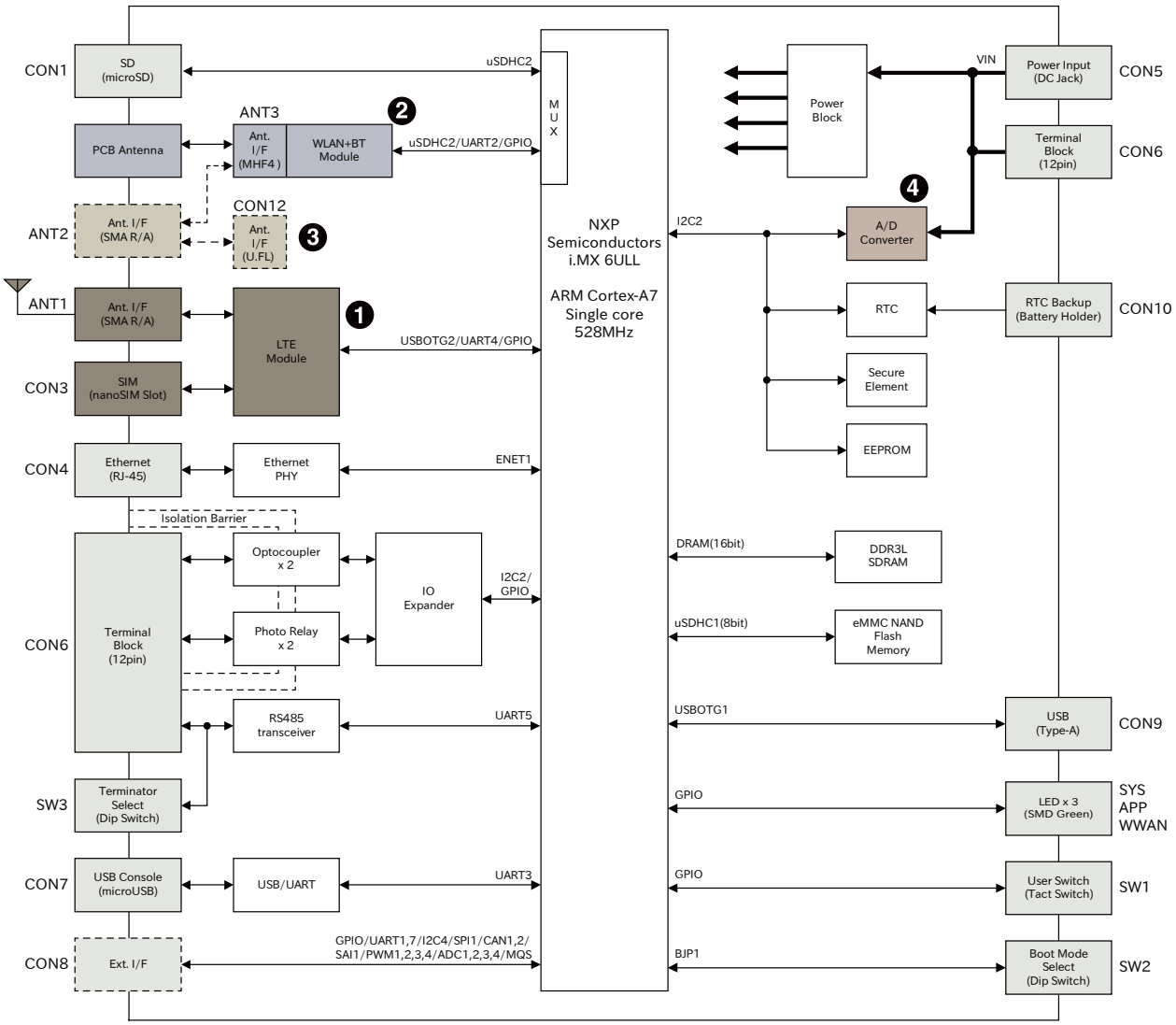


図 2.14 ブロック図 (スタンダートタイプ)

❶ Cat.1 bis モデルの場合、LTE モジュール、アンテナ、SIM スロットが搭載されます。

❷ Cat.1 bis+WLAN モデル、WLAN モデルの場合、WLAN モジュール、基板アンテナが搭載されます。

❸ ANT2 に WLAN アンテナを外付けするカスタマイズおよび拡張基板上の通信モジュールのアンテナケーブルを CON12 に接続して ANT2 にアンテナを外付けするカスタマイズが可能です。

❹ Cat.1 bis モデルの場合、A/D コンバータが搭載されます。

2.5.2. +Di8+Ai4 タイプ

Armadillo-IoT ゲートウェイ A6E +Di8+Ai4 タイプのブロック図です。モデルにより、一部機能が異なります。

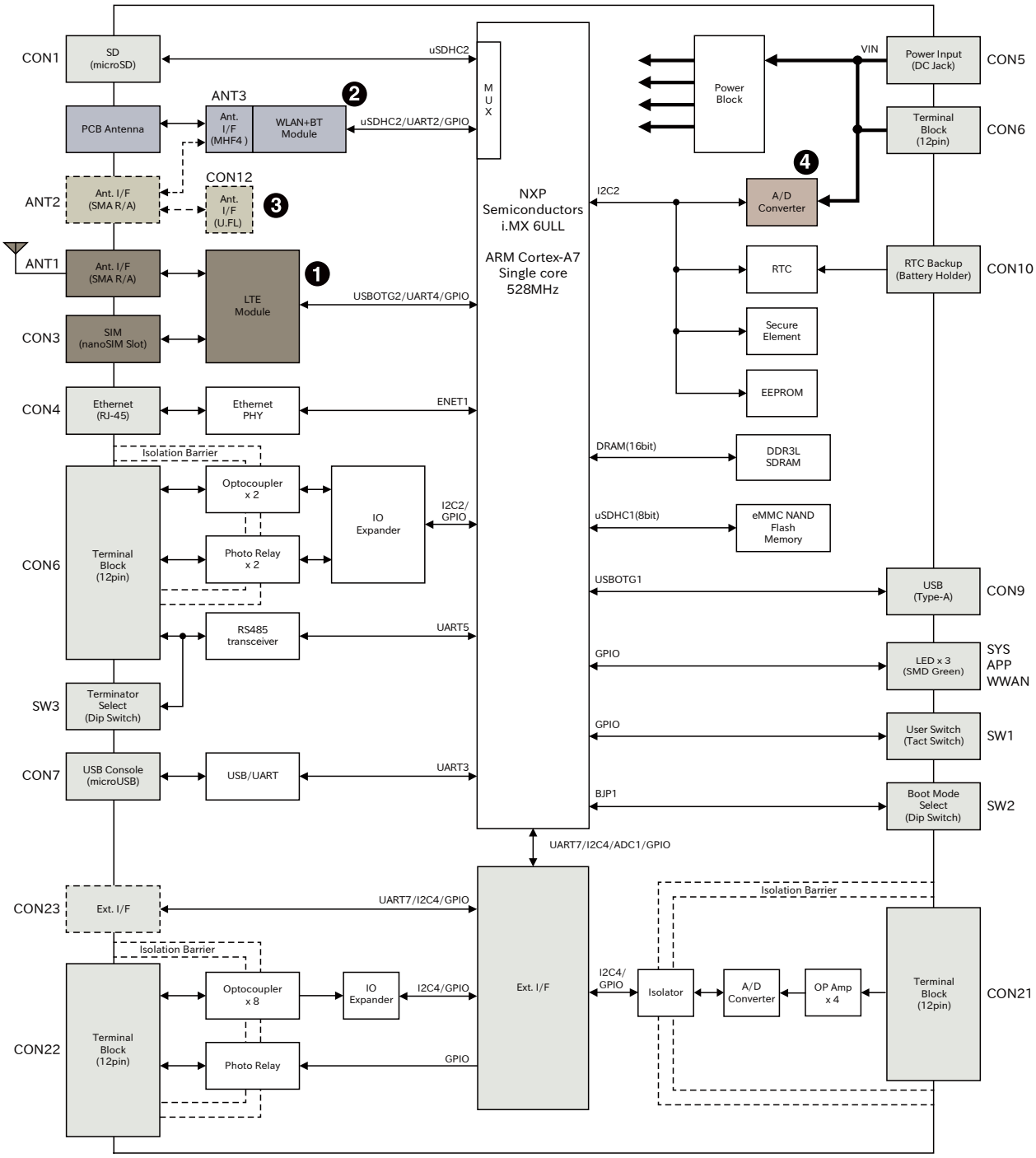


図 2.15 ブロック図 (+Di8+Ai4 タイプ)

❶ Cat.1 bis モデルの場合、LTE モジュール、アンテナ、SIM スロットが搭載されます。

❷ Cat.1 bis+WLAN モデル、WLAN モデルの場合、WLAN モジュール、基板アンテナが搭載されます。

❸ ANT2 に WLAN アンテナを外付けするカスタマイズおよび拡張基板上の通信モジュールのアンテナケーブルを CON12 に接続して ANT2 にアンテナを外付けするカスタマイズが可能です。

❹ Cat.1 bis モデルの場合、A/D コンバータが搭載されます。

2.6. 使用可能なストレージデバイス

Armadillo-IoT ゲートウェイ A6E でストレージとして使用可能なデバイスを次に示します。

表 2.14 ストレージデバイス

デバイス種類	ディスクデバイス	先頭パーティション	インターフェース
オンボード eMMC	/dev/mmcblk0	/dev/mmcblk0p1	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp2	なし	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp3	なし	オンボード
SD/SDHC/SDXC カード [a]	/dev/mmcblk1	/dev/mmcblk1p1	microSD スロット(CON1)
USB メモリ	/dev/sd* [b]	/dev/sd*1	USB ホストインターフェース (CON9)

[a]WLAN 搭載モデルでは SD のストレージとして使用はできません。量産用インストールディスクを WLAN 搭載モデルで作成する場合は、「4.4.4. 開発したシステムをインストールディスクにする」 をご覧ください。
[b]USB ハブを利用して複数の USB メモリを接続した場合は、認識された順に sda 、sdb 、 sdc … となります。



GPP(General Purpose Partition)について

GPP は、eMMC の通常の記憶領域を割譲して eMMC 内部に作られた記憶領域です。 eMMC の通常の記憶領域とはアドレス空間が異なるため、/dev/mmcblk0 および /dev/mmcblk0p* に対してどのような書き込みを行っても /dev/mmcblk0gp* のデータが書き換わることはありません。

Armadillo-IoT ゲートウェイ A6E では、8 MiB の GPP を 4 つ作成しています。各領域の用途を「表 2.15. eMMC の GPP の用途」に示します。

表 2.15 eMMC の GPP の用途

ディスクデバイス	用途
/dev/mmcblk0gp0	ライセンス情報等の為の予約領域
/dev/mmcblk0gp1	動作ログ領域
/dev/mmcblk0gp2	動作ログ予備領域[a]
/dev/mmcblk0gp3	ユーザー領域

[a]詳細は「6.32.4. ログ用パーティションについて」を参照ください。

2.7. ストレージデバイスのパーティション構成

Armadillo-IoT ゲートウェイ A6E の eMMC のパーティション構成を「表 2.16. eMMC メモリマップ」に示します。

表 2.16 eMMC メモリマップ

パーティション	サイズ	ラベル	説明
1	300MiB	rootfs_0	A/B アップデートの A 面パーティション(Linux カーネルイメージ, Device Tree Blob, Alpine Linux rootfs を含む)
2	300MiB	rootfs_1	A/B アップデートの B 面パーティション(Linux カーネルイメージ, Device Tree Blob, Alpine Linux rootfs を含む)
3	50MiB	logs	ログ書き込み用パーティション

パーティション	サイズ	ラベル	説明
4	200MiB	firm	ファームウェア用パーティション
5	2.5GiB	app	アプリケーション用パーティション

Armadillo-IoT ゲートウェイ A6E の eMMC のブートパーティションの構成を「表 2.17. eMMC ブートパーティション構成」に示します。

表 2.17 eMMC ブートパーティション構成

ディスクデバイス	サイズ	説明
/dev/mmcblk0boot0	4 MiB	A/B アップデートの A 面
/dev/mmcblk0boot1	4 MiB	A/B アップデートの B 面

Armadillo-IoT ゲートウェイ A6E の eMMC の GPP(General Purpose Partition)の構成を「表 2.18. eMMC GPP 構成」に示します。

表 2.18 eMMC GPP 構成

ディスクデバイス	サイズ	説明
/dev/mmcblk0gp0	8 MiB	ライセンス情報等の為の予約領域
/dev/mmcblk0gp1	8 MiB	動作ログ領域
/dev/mmcblk0gp2	8 MiB	動作ログ予備領域[a]
/dev/mmcblk0gp3	8 MiB	ユーザー領域

[a]詳細は「6.32.4. ログ用パーティションについて」を参照ください。

2.8. ソフトウェアのライセンス

Armadillo Base OS に含まれるソフトウェアのライセンスは、 Armadillo にログイン後に特定のコマンドを実行することで参照できます。

手順について、詳細は以下の Howto を参照してください。

Armadillo サイト - Howto インストール済みのパッケージのライセンスを確認する

https://armadillo.atmark-techno.com/howto_software-license-confirmation

3. 開発編

Armadillo-IoT ゲートウェイ A6E では基本的に ATDE という Armadillo 専用開発環境と、 Visual Studio Code (以降 VS Code と記載します) 向け Armadillo 開発用エクステンションを用いてアプリケーション開発を行っていきます。

3.1. 開発の準備

この節では、アプリケーション開発のために、はじめに開発環境のセットアップを行います。本節を完了すると、Armadillo を用いた製品の開発に即座に取り組むことができる状態になります。

開発環境のセットアップは、作業用 PC と Armadillo の両方に対して行います。本節では初めに作業用 PC についてのセットアップを行い、その後に Armadillo についてのセットアップを行います。そのため、新たに Armadillo を用意した場合や、Armadillo のセットアップをやり直したい方は本節の途中から行うことができます。後半では Armadillo による開発方法の勝手を大まかに把握したい方を想定して、Python アプリケーションによる LED 点滅の動作確認を行う項を用意しています。不要な方はこの項をスキップしてください。その後、 Armadillo のシリアルコンソールのセットアップ・操作方法について解説します。

3.1.1. 準備するもの

開発環境をセットアップする上で、まずは次のものを用意してください。

作業用 PC	Linux または Windows が動作し、ネットワークインターフェースと 1 つ以上の USB ポートを持つ PC です。
Armadillo-IoT ゲートウェイ A6E 開発セット一式	詳しくは「2.2. 製品ラインアップ」をご参照ください。
1 GB 以上の microSD カード	Armadillo の初期化・ABOS のアップデートの際に使用します。
マイナスドライバー	ドライバーはケースを取り外す際に使用しますので、ケースが無い場合は不要です。
ネットワーク環境	仮想化ソフトウェアや Armadillo の初期化インストールディスクイメージなどを作業用 PC にダウンロードする手順があります。また、 「3.1.5. Armadillo に初期設定をインストールする」 の手順から Armadillo と作業用 PC をネットワーク通信ができるようにする必要があります。

3.1.2. 仮想環境のセットアップ

作業用 PC をセットアップします。アットマークテクノでは、製品のソフトウェア開発や動作確認を簡単に行うために、Oracle VirtualBox 仮想マシンのデータイメージを提供しています。このデータイメージを ATDE(Atmark Techno Development Environment) と呼びます。ATDE の起動には仮想化ソフトウェアである VirtualBox を使用します。



Oracle VirtualBox には以下の特徴があります。

- 基本パッケージが GPLv3(GNU General Public License Version 3) で提供されている
- VMware 形式の仮想ディスク(.vmdk)ファイルに対応している

3.1.2.1. VirtualBox のインストール

ATDE を使用するために、作業用 PC に VirtualBox をインストールします。VirtualBox の Web ページ(https://www.virtualbox.org/) を参照してインストールしてください。

また、ホスト OS が Linux の場合、デフォルトでは VirtualBox で USB デバイスを使用することができません。ホスト OS（Linux）で以下のコマンドを実行後、ホスト OS を再起動してください。

```
[PC ~]$ sudo usermod -a -G vboxusers $USER
```

ホスト OS が Windows の場合はこの操作は必要ありません。

3.1.2.2. ATDE のアーカイブを取得

ATDE のアーカイブ(.ova ファイル)を Armadillo サイト(https://armadillo.atmark-techno.com/resources/software/atde/atde-v9)からダウンロードします。



アットマークテクノ製品の種類ごとに対応している ATDE のバージョンが異なります。本製品に対応している ATDE のバージョンは以下のとおりです。

VirtualBox	ATDE9 v20240925 以降
VMware	ATDE9 v20230123 から ATDE9 v20240826

ATDE9 は Debian GNU/Linux 11 (コードネーム bullseye) をベースに、Armadillo-IoT ゲートウェイ A6E のソフトウェア開発を行うために必要なクロス開発ツールや、Armadillo-IoT ゲートウェイ A6E の動作確認を行うために必要なツールが事前にインストールされています。



作業用 PC の動作環境(ハードウェア、VirtualBox、ATDE の対応アーキテクチャなど)により、 ATDE が正常に動作しない可能性があります。VirtualBox の Web ページ(https://www.virtualbox.org/) から、使用し

ている VirtualBox のドキュメントなどを参照して動作環境を確認してください。

3.1.2.3. ATDE のインポート

1. VirtualBox を起動し、[ファイル]-[仮想アプライアンスのインポート]を選択します。
2. [ソース]の項目で、ダウンロードした ATDE のアーカイブ(.ova ファイル)を選択します。
3. [設定]の項目で、[ハードドライブを VDI としてインポート]のチェックを外します。
4. [完了]をクリックします。インポートの処理が完了するまで数分程要します。
5. インポートの処理が完了したら、ホスト OS の環境に合わせた設定に更新するため仮想マシンを選択して[設定]をクリックした後に[OK]をクリックします。



ATDE に割り当てるメモリおよびプロセッサ数を増やすことで、 ATDE をより快適に使用することができます。仮想マシンのハードウェア設定の変更方法については、VirtualBox の Web ページ (https://www.virtualbox.org/) から、VirtualBox のドキュメントなどを参照してください。

3.1.2.4. ATDE の起動

1. 仮想マシンを選択して[起動]をクリックしてください。
2. ATDE のログイン画面が表示されます。

ATDE にログイン可能なユーザーを、「表 3.1. ユーザー名とパスワード」に示します [1]。

表 3.1 ユーザー名とパスワード

ユーザー名	パスワード	権限
atmark	atmark	一般ユーザー
root	root	特権ユーザー

3.1.2.5. コマンドライン端末(GNOME 端末)の起動

Armadillo を利用した開発では、 CUI (Character-based User Interface)環境を提供するコマンドライン端末を通じて、 Armadillo や ATDE に対して操作を行う場面が多々あります。コマンドライン端末にはいくつかの種類がありますが、ここでは GNOME デスクトップ環境に標準インストールされている GNOME 端末を起動します。

GNOME 端末を起動するには、「図 3.1. GNOME 端末の起動」のようにデスクトップ左上のアプリケーションの「ユーティリティ」カテゴリから「端末」を選択してください。

[1]特権ユーザーで GUI ログインを行うことはできません



図 3.1 GNOME 端末の起動

「図 3.2. GNOME 端末のウィンドウ」のようにウィンドウが開きます。

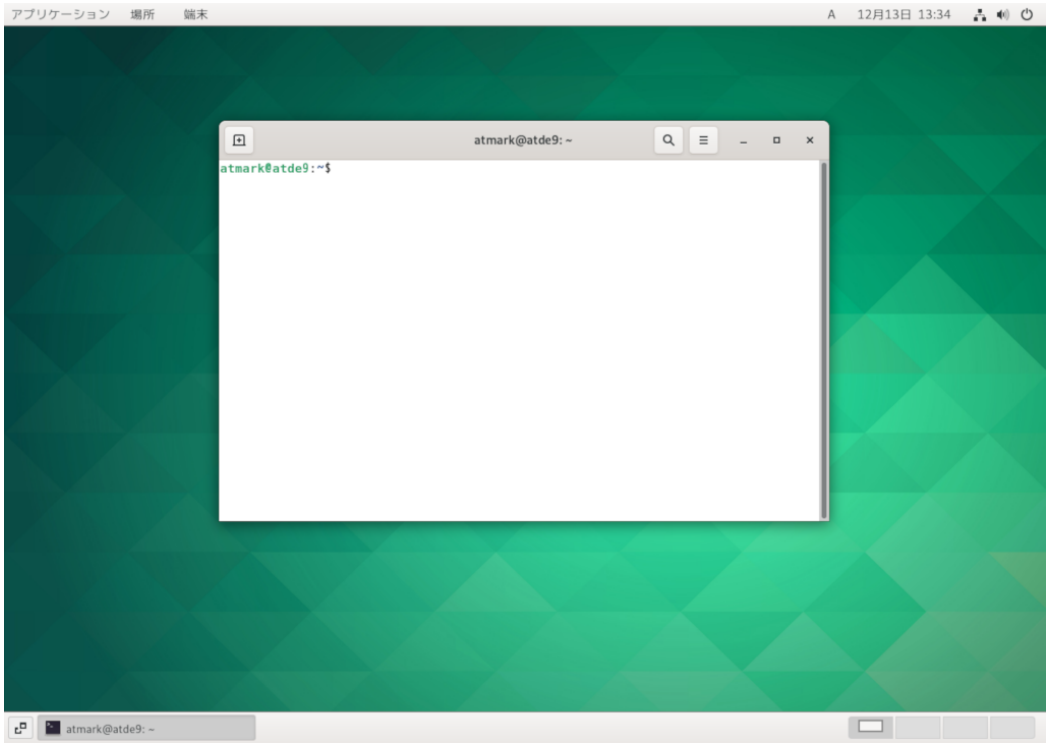


図 3.2 GNOME 端末のウィンドウ

3.1.2.6. ソフトウェアのアップデート

コマンドライン端末から次の操作を行い、ソフトウェアを最新版へアップデートしてください。

```
[ATDE ~]$ sudo apt update && sudo apt upgrade && sudo apt autoremove
```

図 3.3 ソフトウェアをアップデートする

3.1.2.7. 取り外し可能デバイスの使用

VirtualBox は、ゲスト OS (ATDE)による取り外し可能デバイス(USB デバイスや DVD など)の使用をサポートしています。デバイスによっては、ホスト OS (VirtualBox を起動している OS)と ATDE で同時に使用することができません。そのようなデバイスを ATDE で使用するためには、ATDE にデバイスを接続する 「図 3.4. ATDE にデバイスを接続する」 の操作が必要になります。

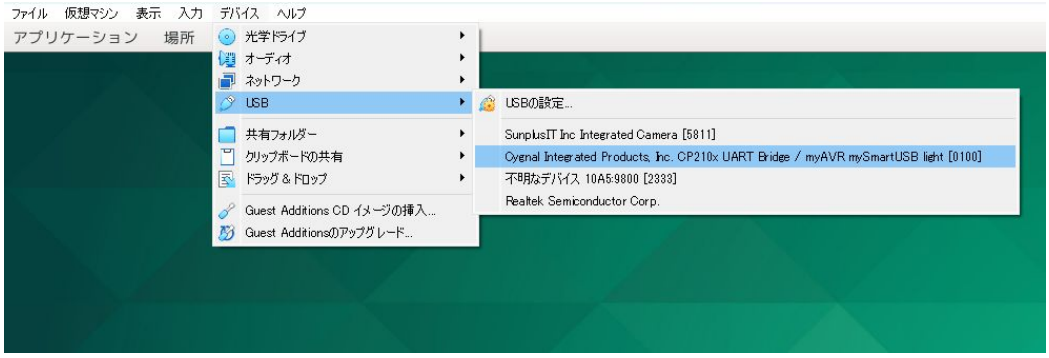


図 3.4 ATDE にデバイスを接続する

3.1.2.8. VirtualBox Guest Additions の再インストール

ATDE は VirtualBox 仮想マシン用ソフトである VirtualBox Guest Additions があらかじめインストールされた状態で配布されています。

Guest Additions のバージョンは VirtualBox 自体のバージョンと連動しているため、お使いの VirtualBox のバージョンと ATDE にインストール済みの Guest Additions のバージョンが異なる場合があります。

VirtualBox と Guest Additions のバージョンが異なることによって問題が起こる可能性は低いですが、これに起因すると思われる不具合（ATDE の画面・共有フォルダー・クリップボード等の不調）が発生した場合は、以下の手順を参考に Guest Additions を再インストールしてください。(実行前に ATDE のスナップショットを作成しておくことを推奨します)

1. ATDE を起動後、上部バーの[デバイス]-[Guest Additions CD イメージの挿入]を選択してください。
2. お使いの VirtualBox と同じバージョンの VBox_GAs_[VERSION] が「ファイル」上に表示されます。
3. VBox_GAs_[VERSION] をマウントするために、「ファイル」から VBox_GAs_[VERSION] を押下してください。
4. インストールする前に、以下のコマンドで既にインストール済みの Guest Additions をアンインストールします。

```
sudo /opt/VBoxGuestAdditions-[VERSION]/uninstall.sh
```

5. 以下のコマンドでお使いの VirtualBox のバージョンに合った Guest Additions がインストールされます。

```
cd /media/cdrom0
sudo sh ./autorun.sh
```

3.1.2.9. 共有フォルダーの作成

ホスト OS と ATDE 間でファイルを受け渡す手段として、共有フォルダーがあると大変便利です。ここでは、ホスト OS と ATDE 間の共有フォルダを作成する手順を紹介しますが、不要な方はこの手順をスキップしてください。

1. VirtualBox の上部バーから[デバイス]-[共有フォルダー]-[共有フォルダー設定]を選択します。(「図 3.5. 共有フォルダー設定を開く」)
2. 「図 3.6. 共有フォルダー設定」 の赤枠で示したアイコンをクリックします。
3. 「図 3.7. 共有フォルダーの追加」 のように、[フォルダーのパス]-[その他]を選択して、共有フォルダーに設定したいホスト OS 上のフォルダーを選択します。
4. 「図 3.7. 共有フォルダーの追加」 のように、[自動マウント]と[永続化する]にチェックを入れます。
5. [OK]をクリックして共有フォルダーを追加します。

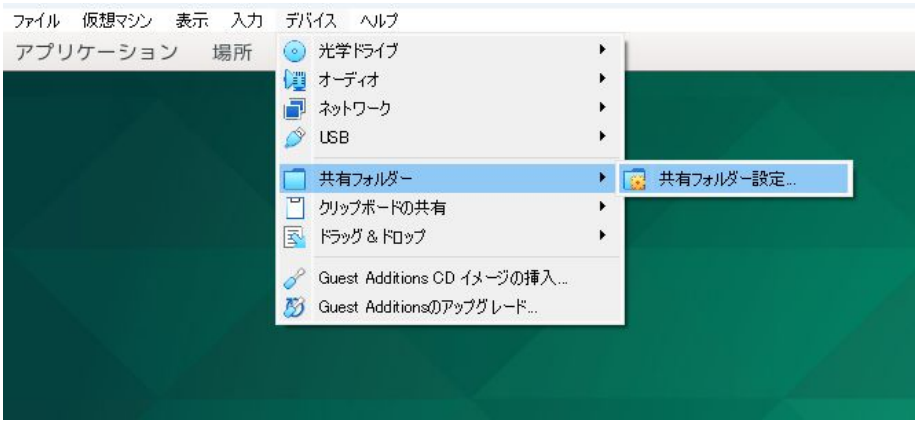


図 3.5 共有フォルダー設定を開く

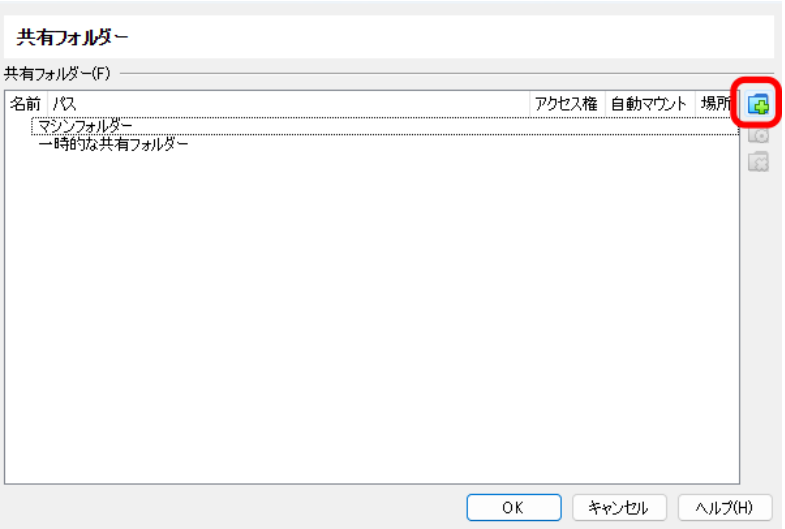


図 3.6 共有フォルダー設定

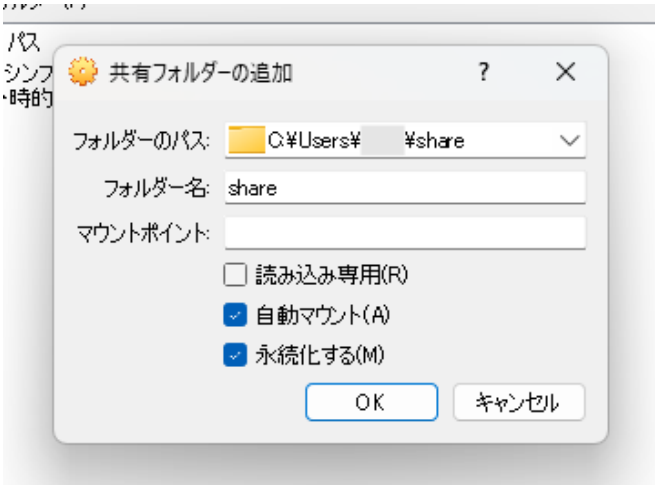


図 3.7 共有フォルダーの追加



図 3.8 「ファイル」に表示される共有フォルダー

追加した共有フォルダーは、「図 3.8. 「ファイル」に表示される共有フォルダー」のように「ファイル」からアクセスするか、または /media/sf_share（共有フォルダー名） からアクセスできます。(share というフォルダー名で作成すると、ATDE 上では sf_share として表示されます。)

3.1.3. VS Code のセットアップ

作業用 PC のセットアップです。Armadillo-IoT ゲートウェイ A6E 上で実行するアプリケーションの開発には、 VS Code を使用します。ATDE のバージョン v20230123 以上には、 VS Code がインストール済みのため新規にインストールする必要はありませんが、使用する前には「図 3.3. ソフトウェアをアップデートする」にしたがって最新版へのアップデートを行ってください。

以下の手順は全て ATDE 上で実施します。

3.1.3.1. VS Code を起動する

VS Code を起動するために code コマンドを実行するか、「アプリケーション」の中から「Visual Studio Code」を探して起動してください。

```
[ATDE ~]$ code
```

図 3.9 VS Code を起動する



VS Code を起動すると、日本語化エクステンションのインストールを提案してくることがあります。その時に表示されるダイアログに従ってインストールを行うと VS Code を日本語化できます。

3.1.3.2. VS Code に開発用エクステンションをインストールする

VS Code 上でアプリケーションを開発するために、ABOSDE (Armadillo Base OS Development Environment) というエクステンションをインストールします。

エクステンションはマーケットプレイスからインストールすることができます。VS Code を起動し、左サイドバーのエクステンションを選択して、検索フォームに「abos」と入力してください。

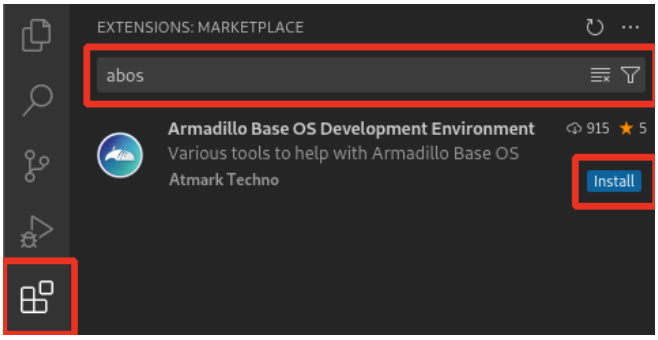


図 3.10 VS Code に開発用エクステンションをインストールする

表示された「Armadillo Base OS Development Environment」の 「Install」ボタンを押すとインストールは完了します。

3.1.4. Armadillo の初期化と ABOS のアップデート

Armadillo をセットアップします。まずは、お手元の Armadillo に搭載されている Armadillo Base OS(ABOS) を最新版にします。ABOS のバージョンが古い場合、本マニュアルで紹介されている重要な機能を使用できない可能性があります。そのため、以下の手順に従って、ABOS のアップデートを兼ねた Armadillo の初期化を行ってください。

3.1.4.1. 初期化インストールディスクの作成

初期化インストールディスクイメージを microSD カードに書き込み、初期化インストールディスクを作成します。ここでは、Windows で作成する方法を紹介します。(ATDE や Linux で実施する方法については「6.34. ATDE・Linux でインストールディスクを作成する」を参照してください)

1. 1 GB 以上の microSD カードを用意してください。

2. 標準のインストールディスクイメージをダウンロードします。Armadillo-IoT ゲートウェイ A6E インストールディスクイメージ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/disc-image] から「Armadillo Base OS インストールディスクイメージ」をダウンロードしてください。

3. ダウンロードした zip ファイルを展開します。図のとおり、zip ファイルを選択して右クリックし、「すべて展開…」をしてください。

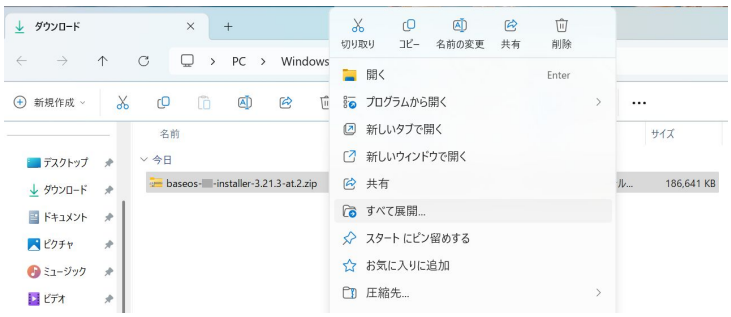


図 3.11 zip ファイルを展開

4. Win32 Disk Imager Renewal（ https://github.com/dnobori/DN-Win32DiskImagerRenewal/releases/）をダウンロードして起動します。

5. PC に microSD カードを接続します。

6. [Image File] に先ほど展開したフォルダ内の img ファイルを指定し、[Device] からは microSD カードに対応するものを選びます。



[Device]の選択には細心の注意を払ってください。他に接続されているデバイスに意図せず書き込んでしまった場合、そのデバイスのデータが上書きされてしまいます。選んだ[Device]が目的の microSD カードなのかどうか、しっかりと確認してください。microSD カードやカードリーダーの挿抜による[Device]の表示の変化で確かめることもできます。

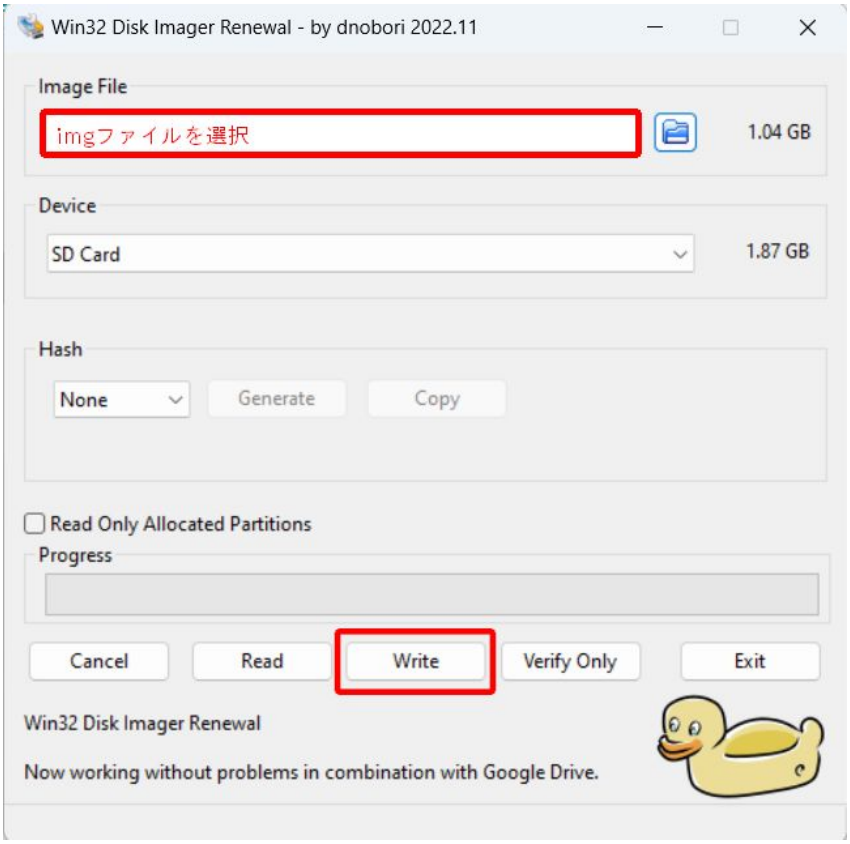


図 3.12 Win32 Disk Imager Renewal 設定画面

7. [Write] ボタンを押すと警告が出ますので、問題なければ[はい]を選択して書き込みます。

8. 書き込み終了ダイアログが表示されたらインストールディスクの作成は完了です。microSD カードを取り外してください。

3.1.4.2. インストールディスクを使用する

作成したインストールディスクを使用して、インストールを実施します。

以下の手順に沿って、 「図 3.13. Armadillo-IoT ゲートウェイ A6E を初期化する接続」 のとおりに接続します。なお、microSD カードを CON1 に挿入するために、ケースを外した状態にする必要があります。「3.5.3. ケースの分解」を参考にケースを外してください。

1. 起動デバイス設定スイッチ(SW2)を「SD」に切り替えます。

2. microSD カードを CON1 に挿入します。(方法は「3.7.7.2. microSD カードの挿抜方法」を参考にしてください)

3. AC アダプタを接続して電源を投入するとシステム LED(SYS) が点灯・点滅します。

4. 4 分程度で eMMC のソフトウェアの初期化が完了し、電源が切れます(システム LED(SYS) が消灯)。

5. システム LED(SYS) が消灯したら、電源を取り外し、続いて 起動デバイス設定スイッチ を eMMC に設定し、microSD カードを外してください。

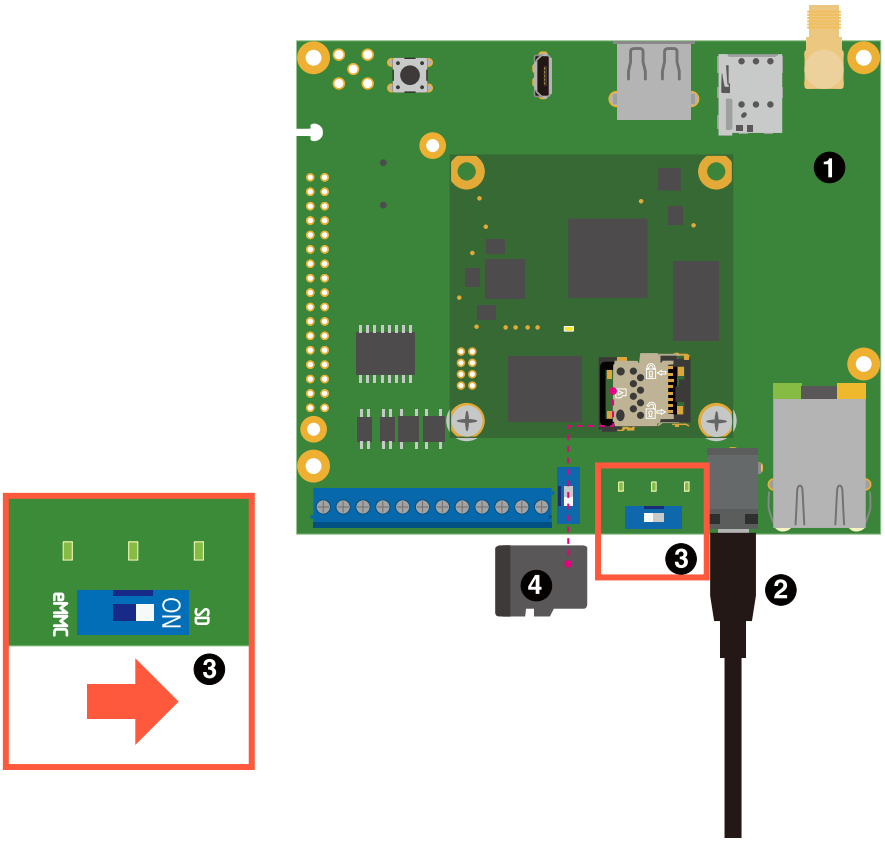


図 3.13 Armadillo-IoT ゲートウェイ A6E を初期化する接続

❶ Armadillo-IoT ゲートウェイ A6E

❷ AC アダプタ(12V/2.0A)

❸ 起動デバイス設定スイッチ

❹ microSD カード



起動デバイス設定スイッチについて

この手順では起動デバイス設定スイッチの操作が必要になります。起動デバイス設定スイッチを操作することで、起動デバイスを設定することができます。

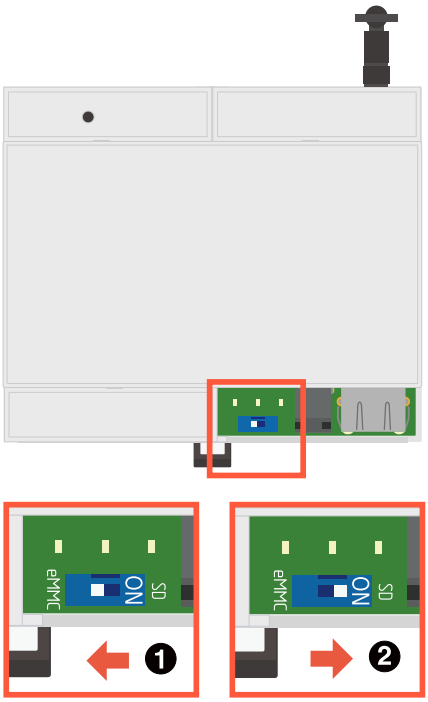


図 3.14 起動デバイス設定スイッチの操作

❶ 起動デバイスは eMMC になります。

❷ 起動デバイスは microSD になります。

起動デバイス設定スイッチの両脇の基板上に、白い文字で eMMC/SD とシルク記載しているので、操作の目印にご利用ください。

3.1.5. Armadillo に初期設定をインストールする

次に、 Armadillo に初期設定（ `initial_setup.swu` ）をインストールします。`initial_setup.swu` はログインパスワードやユーザー固有の証明書などの初期設定を Armadillo にインストールするためのファイルです。`initial_setup.swu` でインストールされるユーザー固有の証明書がない場合、ユーザーが開発したアプリケーションをインストール、またはアップデートすることができません。このため開発前に、初期化された Armadillo に `initial_setup.swu` をインストールする必要があります。初期化された Armadillo に対してユーザーが開発したアプリケーションのインストール・アップデートを行うために必須の手順になりますので、必ず行ってください。

ここでは、 `initial_setup.swu` を VS Code で作成し、 ABOS Web で Armadillo にインストールします。

3.1.5.1. `initial_setup.swu` の作成

「図 3.15. `initial_setup.swu` を作成する」 に示すように、VS Code の左ペインの [COMMON PROJECT COMMAND] から [Generate Initial Setup Swu] を実行してください。

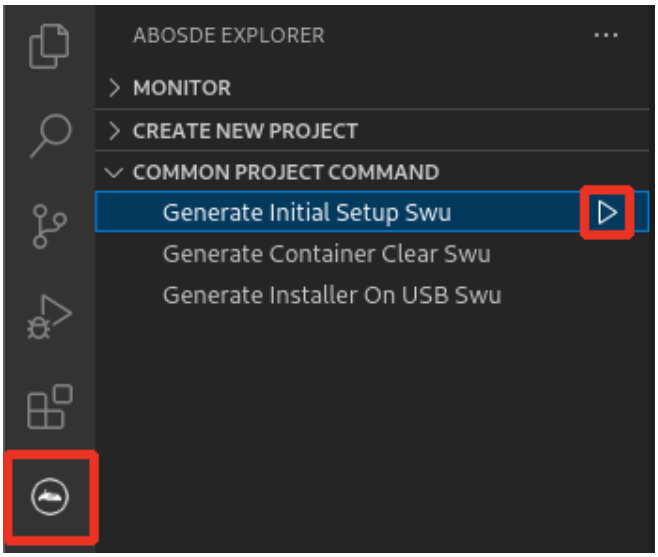


図 3.15 initial_setup.swu を作成する

初回実行時には各種設定の入力を求められます。入力する設定の内容を 「図 3.16. initial_setup.swu 初回生成時の各種設定」 に示します。

なお、この後の Python アプリケーション による動作確認では ABOS Web を使用した手順を記載しています。この後の手順通りに動作確認を行いたい場合は、ABOS Web のパスワードを設定してください。

```
Executing task: ./scripts/generate_initial_setup_swu.sh

mkdir: ディレクトリ '/home/atmark/mkswu' を作成しました
設定ファイルを更新しました：/home/atmark/mkswu/mkswu.conf
証明書のコモンネーム(一般名)を入力してください: [COMMON_NAME] ❶
証明書の鍵のパスワードを入力ください（4-1024 文字）❷
証明書の鍵のパスワード（確認）：
Generating an EC private key
writing new private key to '/home/atmark/mkswu/swupdate.key.tmp'
 -----
アップデートイメージを暗号化しますか？（N/y）❸
アットマークテクノが作成したイメージをインストール可能にしますか？（Y/n）❹
root パスワード: ❺
root のパスワード（確認）：
atmark ユーザのパスワード（空の場合はアカウントをロックします）：❻
atmark のパスワード（確認）：
BaseOS/プリインストールコンテナを armadillo.atmark-techno.com サーバーから自動アップデートしますか？ ⏎
（N/y）❼
abos-web のパスワードを設定してください。
abos-web のパスワード（空の場合はサービスを無効にします）：❽
abos-web のパスワード（確認）：
/home/atmark/mkswu/initial_setup.swu を作成しました。

"/home/atmark/mkswu/initial_setup.swu" をそのまま使うことができますが、
モジュールを追加してイメージを再構築する場合は次のコマンドで作成してください：
  mkswu "/home/atmark/mkswu/initial_setup.desc" [他の.desc ファイル]
```

```
インストール後は、このディレクトリを削除しないように注意してください。
鍵を失うと新たなアップデートはデバイスの /etc/swupdate.pem
を修正しないとインストールできなくなります。
 *  Terminal will be reused by tasks, press any key to close it.

[ATDE ~]$ ls ~/mkswu
initial_setup.desc  initial_setup.swu  mkswu.conf
swupdate.aes-key    swupdate.key       swupdate.pem ❾
```

図 3.16 initial_setup.swu 初回生成時の各種設定

❶ COMMON_NAME には証明鍵の「common name」として会社や製品が分かるような任意の名称を入力してください。

❷ 証明鍵を保護するパスフレーズを 2 回入力します。

❸ SWU イメージ自体を暗号化する場合に「y」を入力します。詳細は 「6.7. SWUpdate と暗号化について」 を参考にしてください。

❹ アットマークテクノのアップデートをインストールしない場合は「n」を入力します。

❺ root のパスワードを 2 回入力します。使用するパスワードは以下のルールに従ってください。

- 辞書に載っている言葉を使用しない
- 単調な文字列を使用しない
- 8 文字以上のパスワード長にする

❻ atmark ユーザーのパスワードを 2 回入力します。何も入力しない場合はユーザーをロックします。使用できるパスワードの制限は root と同様です。

❼ 自動アップデートを無効のままで進みます。ここで「y」を入れると、定期的にアットマークテクノのサーバーからアップデートの有無を確認し、自動的にインストールします。

❽ abos-web を使用する場合はパスワードを設定してください。ここで設定したパスワードは abos-web から変更できます。使用できるパスワードの制限は root と同様です。詳細は 「3.9.6. ABOS Web のパスワード変更」 を参考にしてください。

❾ 作成したファイルを確認します。「swupdate.aes-key」は暗号化の場合にのみ作成されます。

ファイルは ~/mkswu/initial_setup.swu に保存されます。

3.1.5.2. initial_setup.swu を Armadillo にインストール

上の手順で作成した SWU イメージ（initial_setup.swu）を Armadillo へインストールします。インストール方法は様々ありますが（「3.3.4.5. SWU イメージのインストール」）、ここでは ABOS Web を使用した手動インストールを行います。

ABOS には ABOS Web という機能が含まれています。この機能を活用することで、 Web ブラウザからネットワークの設定や、 SWU イメージのインストールなどを簡単に行うことができます。（ただし、Armadillo と作業用 PC が同一 LAN 内に存在している必要があります）

以下の手順に沿って、 ABOS Web へアクセスし、initial_setup.swu のインストールを行ってください。

まず、 「図 3.17. ABOS にアクセスするための接続」 のとおりに Armadillo に配線を行い、電源を入れてください。

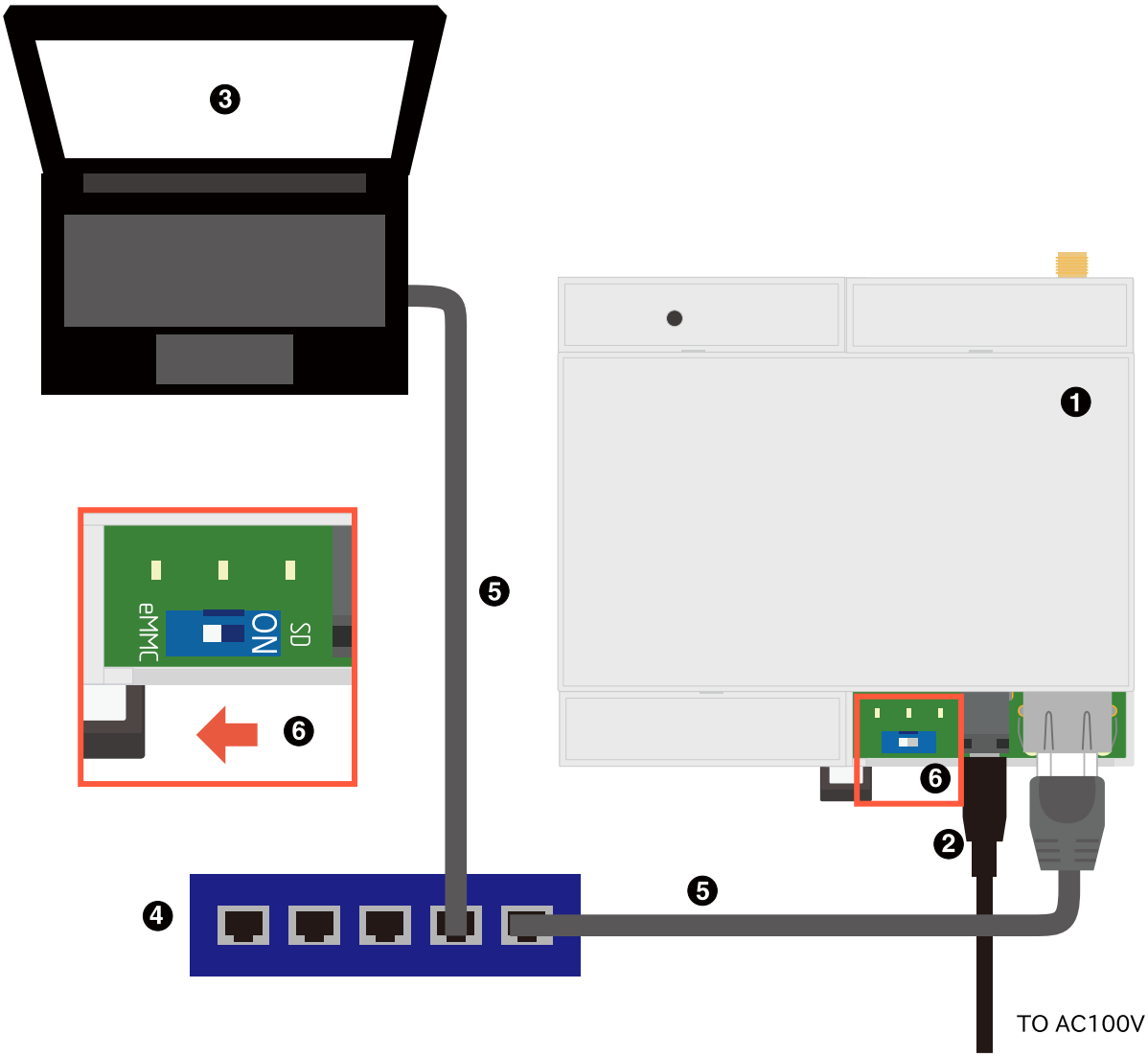


図 3.17 ABOS にアクセスするための接続

1. Armadillo-IoT ゲートウェイ A6E
2. AC アダプタ(12V/2.0A)
3. 作業用 PC
4. LAN HUB
5. Ethernet ケーブル
6. 起動デバイス設定スイッチ

1 分ほど待機して、ABOSDE でローカルネットワーク上の Armadillo をスキャンします。「図 3.18. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲われているボタンをクリックしてください。

Armadillo が正常に起動していた場合、「図 3.19. ABOSDE に表示されている Armadillo を更新する」の一覧に起動した Armadillo が armadillo.local という名称で表示されます。表示されない場合は 1 分

ほど待機してから「図 3.19. ABOSDE に表示されている Armadillo を更新する」の赤枠で囲われているマークをクリックしてスキャンを再度試みてください。

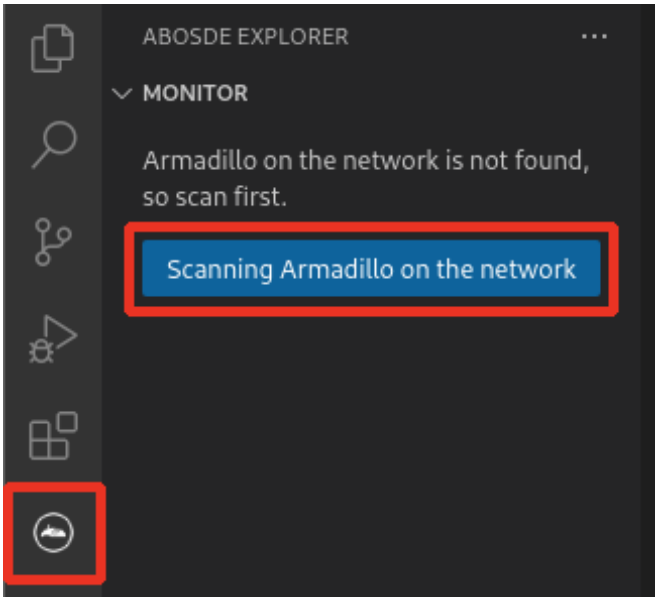


図 3.18 ABOSDE で ローカルネットワーク上の Armadillo をスキャンする

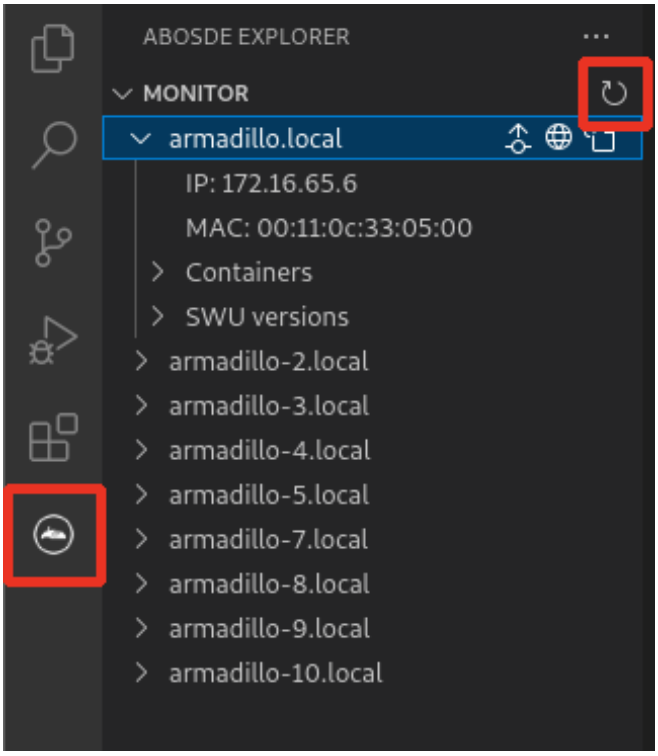


図 3.19 ABOSDE に表示されている Armadillo を更新する

ただし、ATDE のネットワークをブリッジ接続以外に設定している場合は Armadillo がリストに表示されない場合があります。表示するためには ATDE のネットワークをブリッジ接続に設定してください。また、ABOS Web が動作する Armadillo が同じ LAN 上に複数あると、ABOS Web に接続する URL のホスト名部分（armadillo.local）が、2 台目では armadillo-2.local、3 台目では armadillo-3.local の

ように、違うものが自動的に割り当てられます。目的の Armadillo がどのホスト名なのか不明な場合には、Armadillo のラベルに記載されている MAC アドレスと一致するもの（ 「図 3.20. ABOSDE を使用して ABOS Web を開く」 の赤枠に表示されます ）を探してください。

また、 Armadillo を社内 LAN やインターネットに直接に接続できないなどの場合は、ATDE を社内 LAN（インターネット）に接続しながら Armadillo とリンクローカルアドレスで 1 対 1 の接続を行うこともできます。詳細はブログ 「Armadillo Base OS：Armadillo をインターネットに繋げない環境でのソフトウェア開発を行うために」 [https://armadillo.atmark-techno.com/blog/10899/25537/]をご参照ください。

続いて、「図 3.20. ABOSDE を使用して ABOS Web を開く」 の赤枠で囲われているマークをクリックして、ABOS Web を Web ブラウザで開きます。

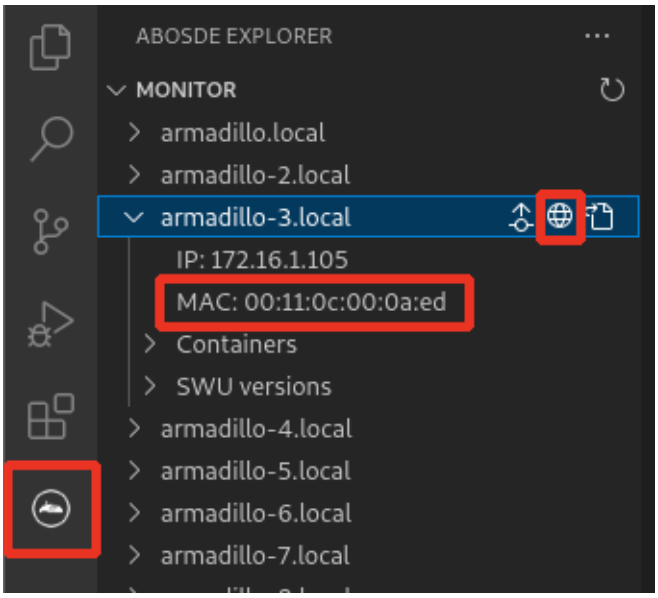


図 3.20 ABOSDE を使用して ABOS Web を開く

3.1.5.3. ABOS Web へアクセス

ABOS Web が正常に起動していれば、Web ブラウザに パスワード登録画面（ 「図 3.21. パスワード登録画面」 ）が表示されます。`initial_setup.swu` を作成する手順で設定したパスワードを入力して、ABOS Web のログイン用パスワードを設定します。



図 3.21 パスワード登録画面

パスワード登録画面で、"パスワード" フィールドと "パスワード(確認)" フィールドに、登録したいパスワードを入力してから、"登録" ボタンをクリックしてください。パスワード登録完了画面が表示されたら、パスワード登録の完了です。



図 3.22 パスワード登録完了画面

パスワード登録完了画面にある "トップページ" というリンクをクリックすると、「図 3.24. トップページ」が表示されます。

以降の接続の際は以下のログイン画面が表示されます。



図 3.23 ログイン画面

ログインに成功すると、ABOS Web の設定画面（ 「図 3.24. トップページ」 ）に表示が変わり、設定操作を行うことができます。これで、ABOS Web へのアクセスが完了しました。



図 3.24 トップページ

3.1.5.4. ABOS Web から initial_setup.swu をインストール

ABOS Web のトップページから"SWU インストール"をクリックして、 「図 3.25. SWU インストール」の画面に遷移します。



図 3.25 SWU インストール

"参照…"から ~/mkswu/initial_setup.swu を選択し、"インストール"をクリックしてください。数分ほど待機すると 「図 3.26. SWU インストールに成功した画面」 のように"インストールが成功しました。"と表示され、Armadillo が再起動します。(ABOS Web も再起動されるので、再起動完了後にページを更新するとログイン画面に戻ります)



図 3.26 SWU インストールに成功した画面

これで Armadillo に初期設定をインストールする手順が完了です。インストール完了後に ~/mkswu ディレクトリ以下にある `mkswu.conf` と、鍵ファイルの `swupdate.*` をなくさないようにしてください。



ABOS Web にブラウザから直接アクセスする

ABOSDE を使わずに、 直接 Web ブラウザのアドレスバーに ABOS Web の URL を入力することでも ABOS Web にアクセスできます。ATDE で Web ブラウザを起動した後、Web ブラウザのアドレスバーに次の URL を入力してください： https://armadillo.local:58080

複数台の Armadillo が接続されている場合には、armadillo.local の部分が armadillo-2.local や armadillo-3.local となっている可能性があります。これらは ABOSDE のリストに表示されているホスト名と同名ですので、目的の Armadillo と一致するホスト名を入力してください。

また、Web ブラウザから直接アクセスする方法では、ホスト名ではなく IP アドレスを指定することもできます。例えば、Armadillo の（ネットワークコネクタの）IP アドレスが 192.0.2.80 である場合は、次の URL を入力してください： https://192.0.2.80:5808

IP アドレスを固定している場合は IP アドレスを指定する方法が便利になる場面もあります。また、IP アドレスを指定する方法は ATDE のネットワークを NAT に設定している場合でも有効です。



ABOS Web からログアウトする

ログアウトを行う場合は、サイドメニューから "ログアウト" を選択してください。ログアウトすると、ログイン画面が再び表示されますので、ABOS Web をすぐに使わないのであれば、Web ブラウザを閉じてください。

3.1.6. Python アプリケーションで動作確認する

本項では LED を点滅させる Python のサンプルアプリケーションを使用して、Armadillo による開発方法の勝手を大まかに把握したい方を想定した簡単な動作確認を行います。なお、開発環境のセットアップに直接関わる手順ではないので、この動作確認が不要な方は本項をスキップしてください。

3.1.6.1. プロジェクトの作成

Armadillo でのアプリケーションの開発には ABOSDE を使用します。

VS Code の左ペインの [A6E] から [Python New Project] を実行（右に表示されている三角形ボタン）し、表示されるディレクトリ選択画面からプロジェクトを保存するディレクトリを選択してください。保存先を選択すると、プロジェクト名を入力するダイアログが表示されるので、任意のプロジェクト名を入力してエンターキーを押してください。この操作により、選択した保存先に、入力したプロジェクト名と同名のディレクトリが作成されます。

また、ここでは次のように設定しています。

- **保存先** : ホームディレクトリ
- **プロジェクト名** : `my_project`

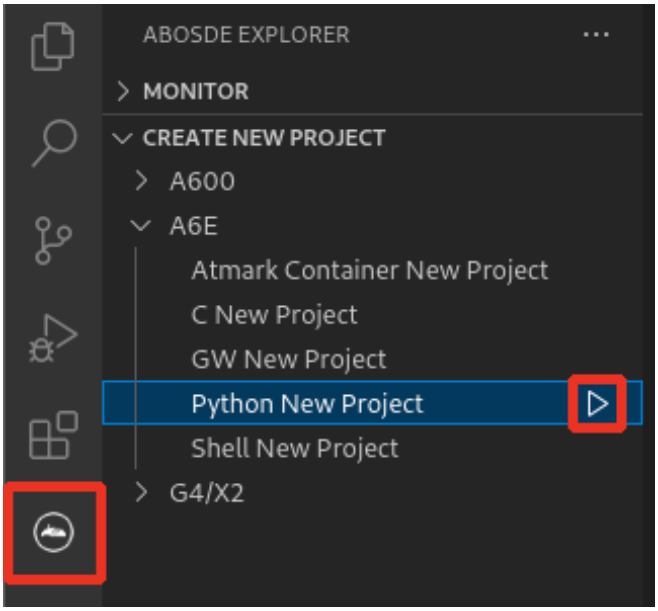


図 3.27 プロジェクトを作成する

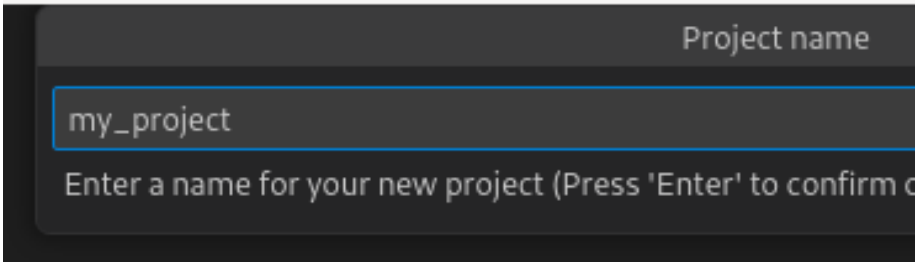


図 3.28 プロジェクト名を入力する

プロジェクトを作成したら、VS Code で `my_project` のディレクトリを開いてください。

3.1.6.2. 初期設定

プロジェクトを作成する度に、初期設定を行う必要があります。初期設定では主に Armadillo と SSH で接続するための秘密鍵と公開鍵の生成を行います。以下の手順を実施してください。

VS Code の左ペインの [my_project] から [Setup environment] を実行します。

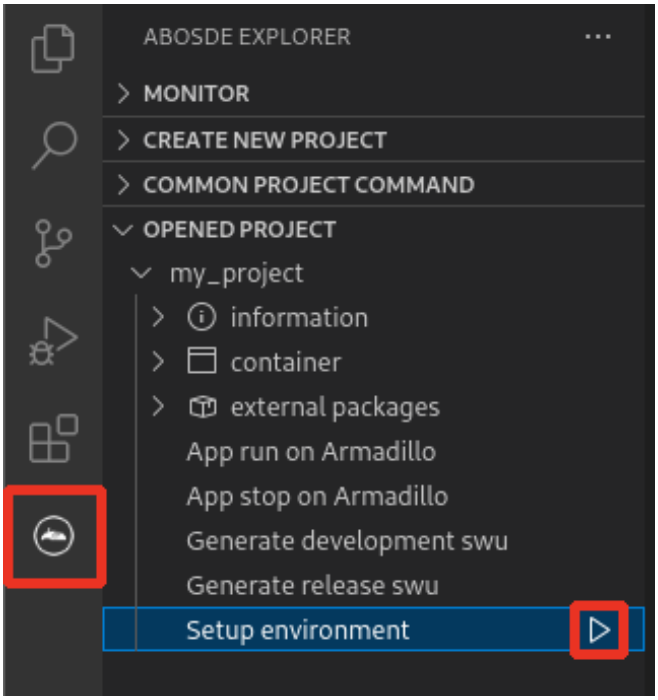


図 3.29 VS Code で初期設定を行う

選択すると、 VS Code の下部に以下のようなターミナルが表示されます。

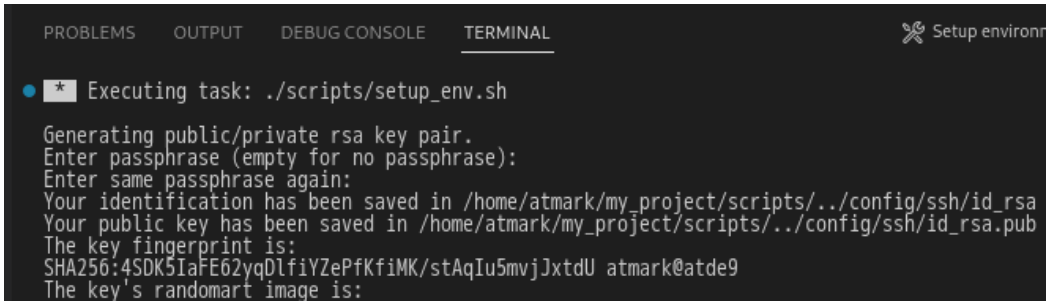


図 3.30 VS Code のターミナル

このターミナル上で以下のように入力してください。

```
 *  Executing task: ./scripts/setup_env.sh

Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase): ❶
Enter same passphrase again: ❷
Your identification has been saved in /home/atmark/.ssh/id_ed25519_vscode
:(省略)
```

```
*  Terminal will be reused by tasks, press any key to close it.  ❸
```

図 3.31 SSH 用の鍵を生成する

❶ パスフレーズを設定します。設定しない場合は何も入力せず Enter を押します。

❷ 1 でパスフレーズを設定した場合は、確認のため再度入力してください。

❸ ここで何か任意のキーを押すとターミナルが閉じます。

パスフレーズを設定した場合は、アプリケーションを Armadillo へ転送する時にパスフレーズの入力を求められることがあります。



ssh の鍵は $HOME/.ssh/id_ed25519_vscode (と id_ed25519_vscode.pub) に保存されていますので、プロジェクトをバックアップする時は $HOME/.ssh も保存してください。

3.1.6.3. アプリケーション実行用コンテナイメージの作成

Armadillo 上でアプリケーションを実行するためのコンテナイメージを作成します。ここで作成したコンテナイメージは SWU イメージを使用して Armadillo へインストールします。

コンテナイメージの作成および SWU イメージの作成も VS Code で行います。VS Code の左ペインの [my_project] から [Generate development swu] を実行します。

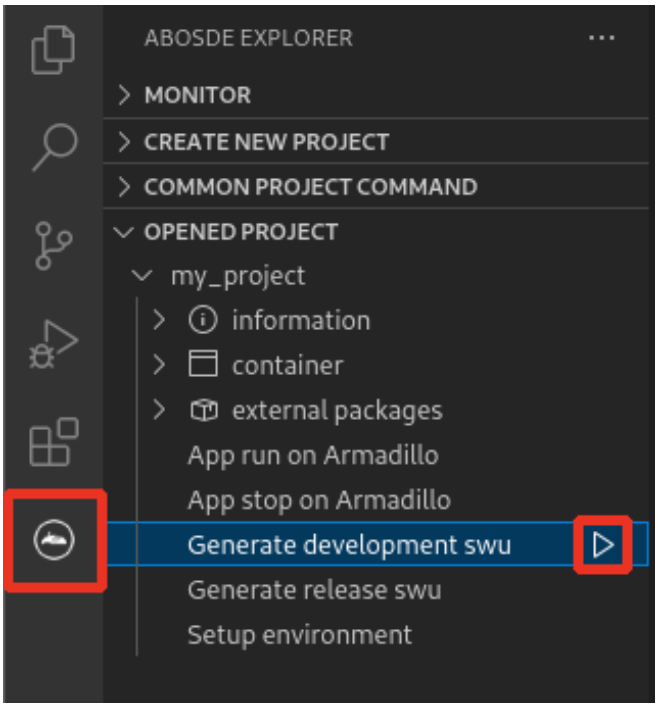


図 3.32 VS Code でコンテナイメージの作成を行う

コンテナイメージの作成にはしばらく時間がかかります。VS Code のターミナルに以下のように表示されるとコンテナイメージの作成は完了です。

```
コンテナイメージを ./swu/my_project.tar に保存しました。
./swu/app.desc のバージョンを 1 から 2 に変更しました。
./development.swu を作成しました。
次は Armadillo に ./development.swu をインストールしてください。
 *  Terminal will be reused by tasks, press any key to close it.
```

図 3.33 コンテナイメージの作成完了

作成した SWU イメージは my_project ディレクトリ下に development.swu というファイル名で保存されています。

3.1.6.4. アプリケーション実行用コンテナイメージのインストール

上で作成した development.swu を Armadillo へインストールします。initial_setup.swu をインストールしたときと同様に ABOS Web からインストールさせることも可能ですが、ここでは ABOSDE を使用してインストールする手順をご紹介します。

「図 3.34. ABOSDE で Armadillo に SWU をインストール」 のように、目的の Armadillo の隣にある赤枠で囲まれているボタンをクリックしてください。パスワードの入力を要求されますので、ABOS Web のパスワードを入力してください。その後、 ~/my_project/development.swu を選択してインストールを開始します。

インストールが成功すると、VS Code のターミナルに Successfully installed SWU と表示されます。

インストール後に自動で Armadillo が再起動し、1 分ほど待機すると LED が点滅します。

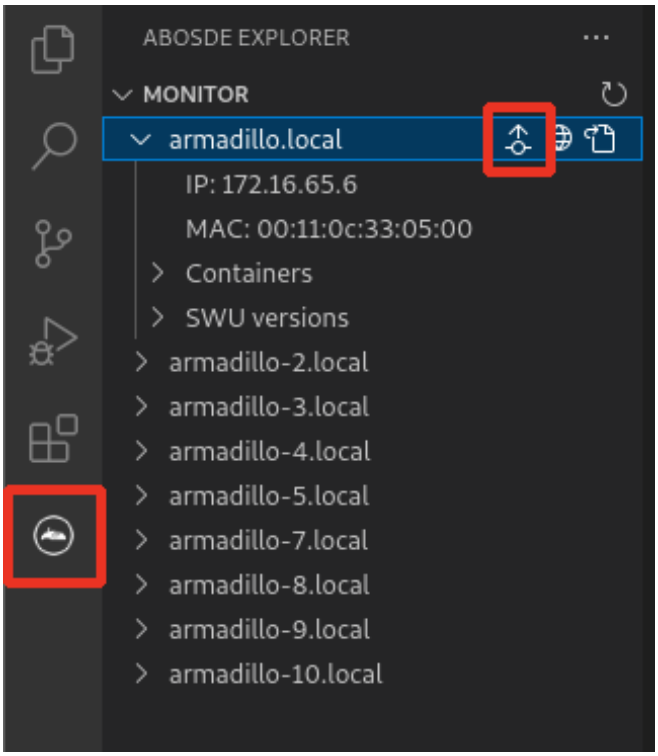


図 3.34 ABOSDE で Armadillo に SWU をインストール

3.1.6.5. ssh 接続に使用する IP アドレスの設定

以下の手順にしたがい、ABOS Web が動作している Armadillo の一覧を確認し、ssh 接続に使用する Armadillo の IP アドレスを指定してください。なお、この手順は Armadillo の IP アドレス が変更される度に行う必要があります。

「図 3.18. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲われているボタン、または 「図 3.19. ABOSDE に表示されている Armadillo を更新する」 の赤枠で囲われているマークをクリックして、ローカルネットワーク上で ABOS Web が実行されている Armadillo をスキャンしてください。

その後、目的の Armadillo について、「図 3.35. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する」 の赤枠で囲われているマークをクリックしてください。

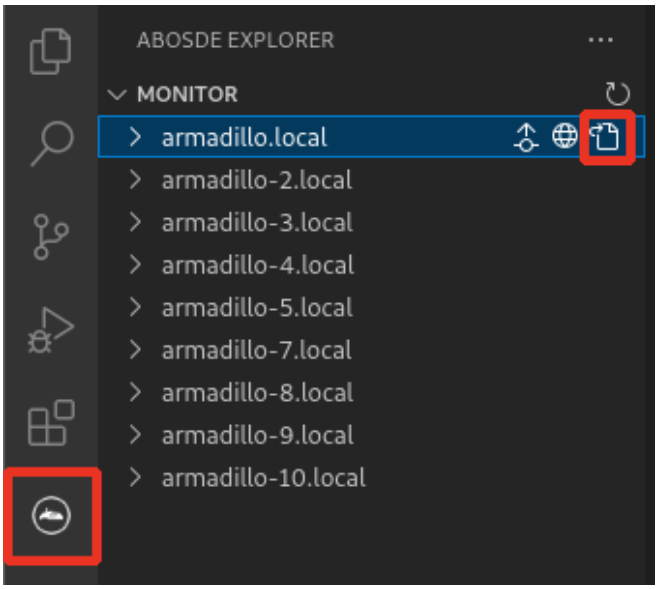


図 3.35 ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する

これにより、指定した Armadillo の IP アドレスを ssh 接続に使用する IP アドレスに設定することができます。また、プロジェクトディレクトリ内の `config/ssh_config` ファイルに指定した Armadillo の IP アドレスが記載されます。ATDE のネットワークを NAT に設定している場合や、ABOS Web を起動していない場合等、ABOSDE のリストに Armadillo が表示されない場合は、プロジェクトディレクトリに入っている `config/ssh_config` ファイルを編集して IP アドレスを書き換えてください。

```
[ATDE ~/my_project]$ code config/ssh_config
Host Armadillo
    Hostname x.x.x.x ❶
    User root
    IdentityFile ${HOME}/.ssh/id_ed25519_vscode
    UserKnownHostsFile config/ssh_known_hosts
    StrictHostKeyChecking accept-new
```

図 3.36 ssh_config を編集する

❶ Armadillo の IP アドレスに置き換えてください。



Armadillo を初期化した場合や、プロジェクトを実行する Armadillo を変えた場合は,プロジェクトの `config/ssh_known_hosts` に保存されている公開鍵で Armadillo を認識できなくなります。その場合はファイルを削除するか、「Setup environment」タスクを再実行してください。

3.1.6.6. アプリケーションの実行

VS Code の左ペインの [my_project] から [App run on Armadillo] を実行すると、アプリケーションが Armadillo へ転送されて起動します。

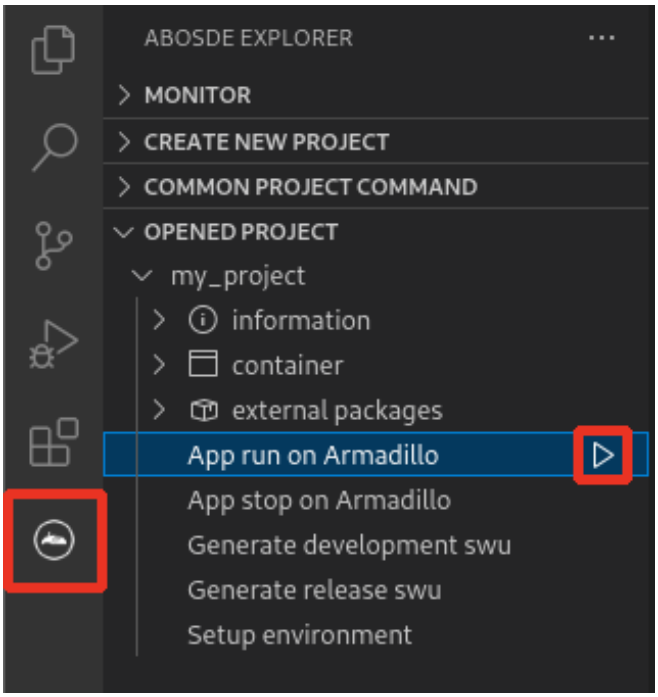


図 3.37 Armadillo 上でアプリケーションを実行する

VS Code のターミナルに以下のメッセージが表示されることがあります。これが表示された場合は yes と入力して下さい。

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

図 3.38 実行時に表示されるメッセージ

アプリケーションを終了するには VS Code の左ペインの [my_project] から [App stop on Armadillo] を実行してください。

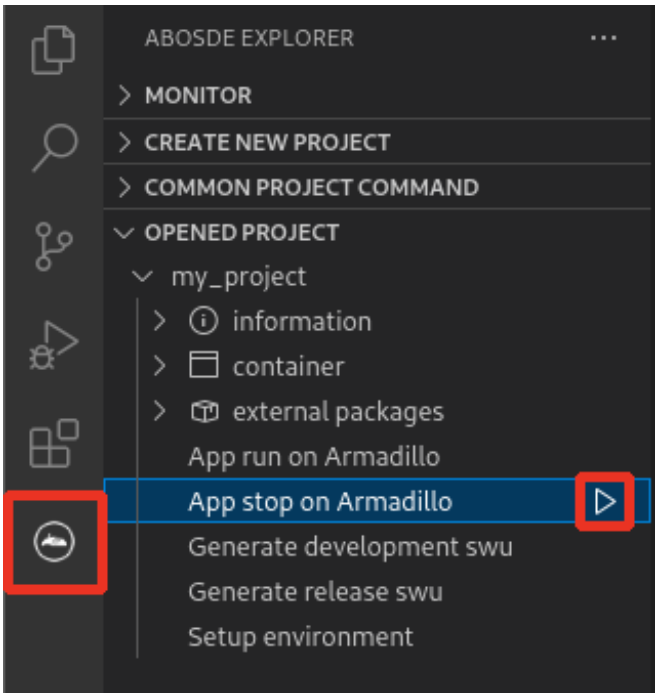


図 3.39 アプリケーションを終了する

3.1.6.7. アプリケーションの削除

動作確認として使用した Python アプリケーションを削除します。ABOSDE から Armadillo のコンテナイメージを全て削除する SWU イメージを作成します。この方法はコンテナイメージを全て削除する方法ですので、開発中に複数のコンテナイメージを使用している場合などはそれらも削除されることに注意してください。

VS Code の左ペインの [COMMON PROJECT COMMAND] から [Generate Container Clear Swu] を実行すると、SWU イメージが作成されます。SWU イメージは `~/mkswu/container_clear.swu` に保存されます。

この SWU イメージを Armadillo へインストールします。

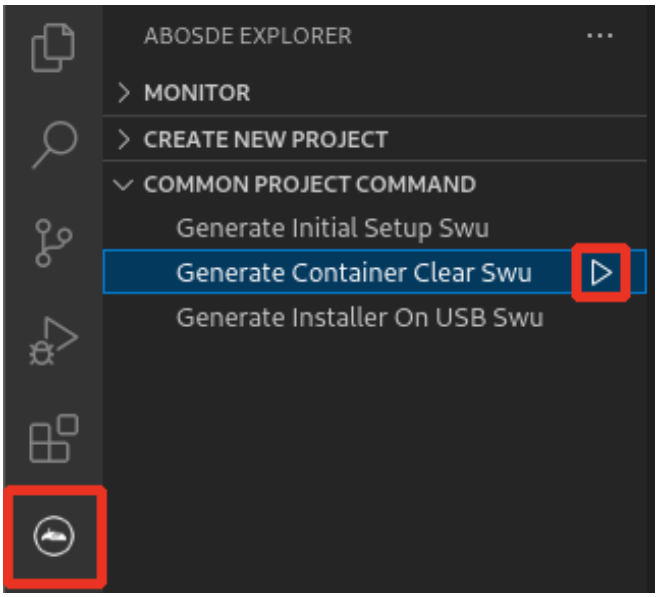


図 3.40 Armadillo 上のコンテナイメージを削除する

「図 3.34. ABOSDE で Armadillo に SWU をインストール」 のように、目的の Armadillo の隣にある赤枠で囲まれているボタンをクリックしてください。パスワードの入力を要求されますので、ABOS Web のパスワードを入力してください。その後、 `~/mkswu/container_clear.swu` を選択してインストールを開始します。

インストール後に自動で Armadillo が再起動し、LED が点滅しなくなります。

3.1.7. シリアルコンソールを使用する

Armadillo ではシリアルコンソールを通じて Linux コマンドを直接実行することができます。シリアルコンソールを活用することで、ABOS Web や ABOSDE からではできない多くのことが可能になるため、より応用的な開発やメンテナンス・デバッグの際に重宝します。また、この章以降ではシリアルコンソールを使用した手順が多々登場します。

本項ではシリアル通信ソフトウェア(Tera Term)を使用したシリアルコンソールの操作方法について記載しています。

3.1.7.1. Armadillo と開発用 PC を接続

Armadillo-IoT ゲートウェイ A6E のシリアルコンソールを使用するために、 「図 3.41. シリアルコンソールを使用する配線例」 のとおりに配線を行ってください。この配線図は Armadillo-IoT ゲートウェイ A6E のシリアルコンソールを使用するための最低限の配線ですので、これに加えて他のインターフェースを接続しても問題ありません。

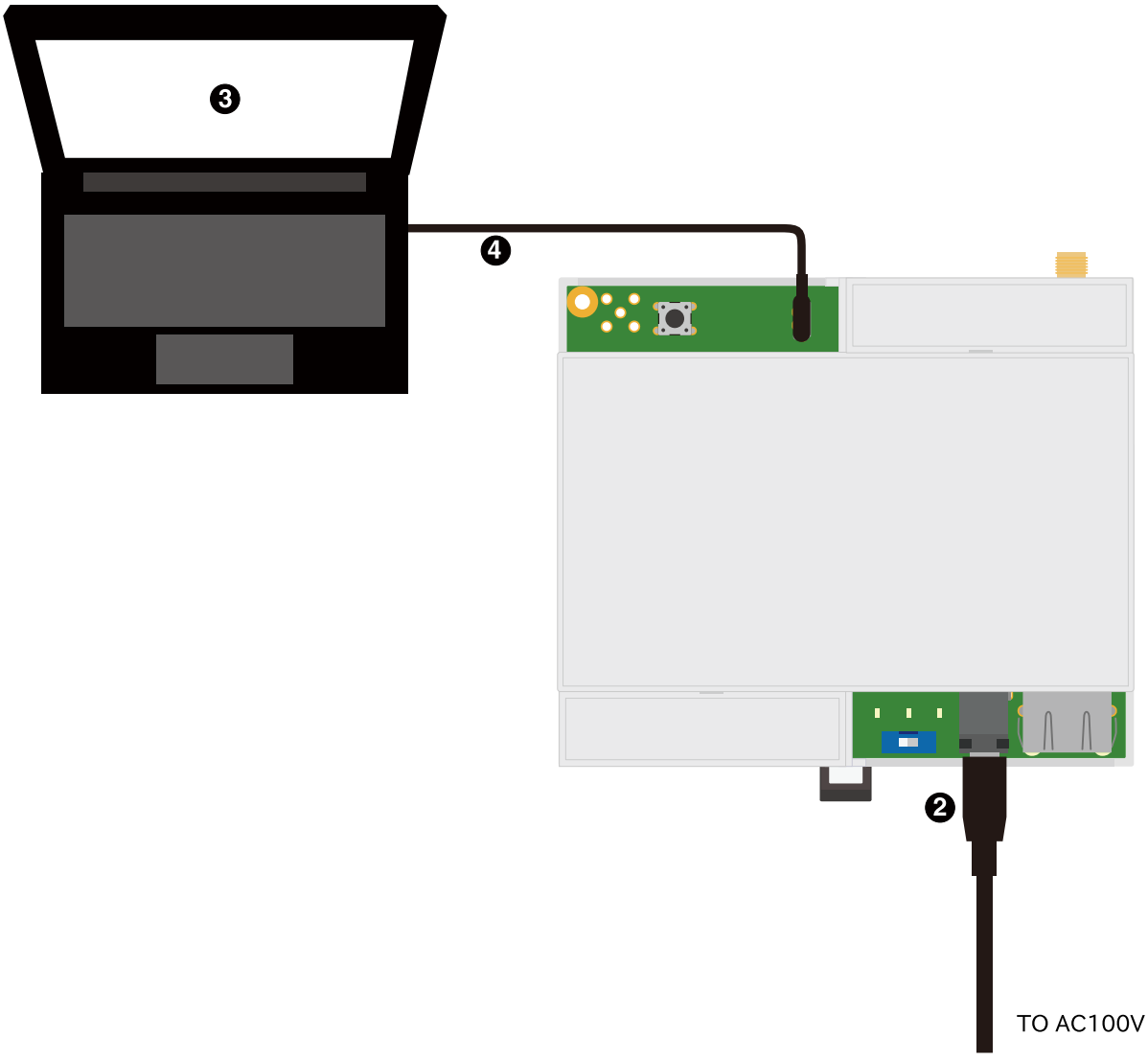


図 3.41 シリアルコンソールを使用する配線例

❶ Armadillo-IoT ゲートウェイ A6E

❷ AC アダプタ(12V/2.0A)

❸ 作業用 PC

❹ シリアル通信用 USB ケーブル(A-microB)

3.1.7.2. Tera Term の起動

本項では Windows 環境で Tera Term を使用したシリアルコンソールの操作方法について記載しています。(Linux や ATDE の場合は 「6.35. シリアル通信ソフトウェア(minicom)」 を参照してください。)

1. Tera Term（https://github.com/TeraTermProject/teraterm/releases）をダウンロード・インストールして起動します。

2. 下図のように、[新しい接続]ダイアログに Armadillo を接続している COM ポートを指定して[OK]をクリックします。ダイアログが表示されていない場合は、[ファイル]-[新しい接続]をクリックして表示します。

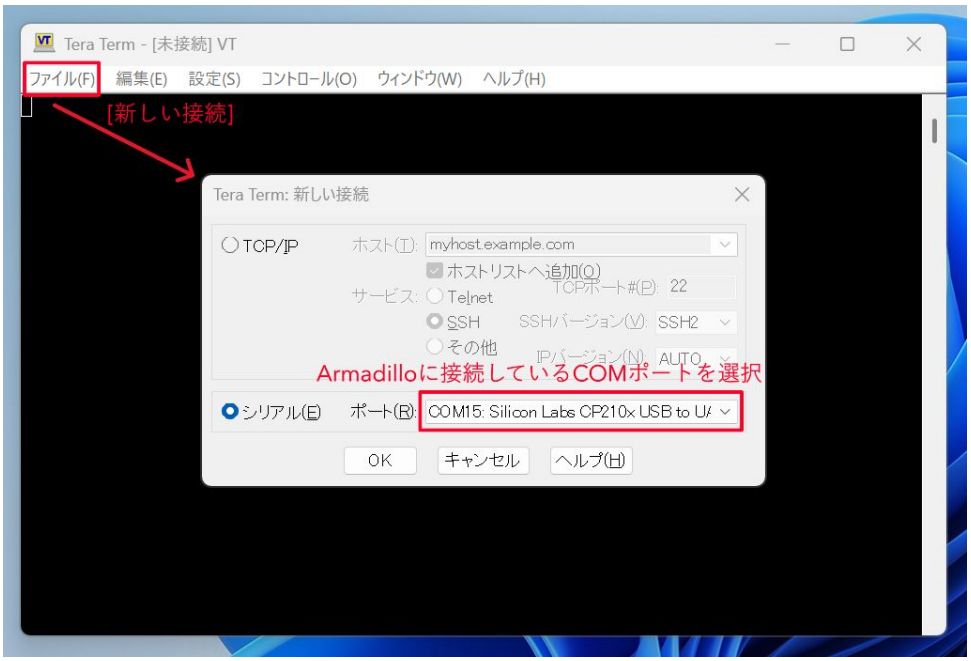


図 3.42 Armadillo を接続している COM ポートを指定

3. [設定]-[シリアルポート]をクリックして、Armadillo を接続しているシリアルポートの設定を下図のように行い、[OK]をクリックします。

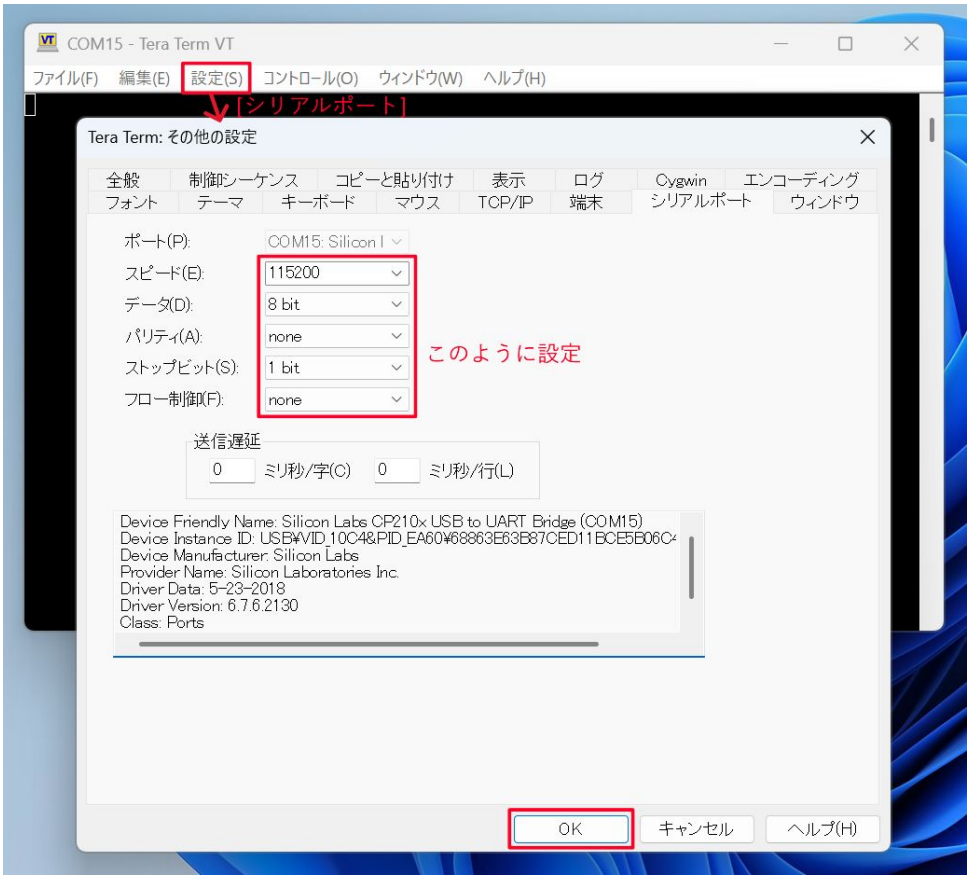


図 3.43 Armadillo を接続しているシリアルポートの設定

3.1.7.3. Armadillo の起動

電源を接続して Armadillo-IoT ゲートウェイ A6E を起動してください。シリアルコンソールに以下のような起動ログが出力されます。

```
U-Boot 2020.04-at17 (Jul 07 2023 - 06:28:15 +0000)

CPU:   i.MX6ULL rev1.1 at 396 MHz
Model: Atmark Techno Armadillo-IoT Gateway A6E Board

: (省略)

Starting kernel ...


   OpenRC 0.45.2 is starting up Linux 5.10.185-1-at (armv7l)

 * Mounting /proc ... [ ok ]
 * Mounting /run ... * /run/openrc: creating directory

: (省略)

Welcome to Alpine Linux 3.17
Kernel 5.10.185-1-at on an armv7l (/dev/ttymxc2)
```

```
armadillo login:
```

既に起動している場合は、Enter を 1 回押すことで armadillo login: が表示されます。

U-Boot プロンプト

ユーザースイッチ(SW1) を押しながら電源を投入すると、 U-Boot のプロンプトが表示されます。

```
U-Boot 2020.04-at10 (Oct 04 2022 - 11:22:32 +0900)

CPU:   i.MX6ULL rev1.1 at 396 MHz
Model: Atmark Techno Armadillo-IoT Gateway A6E
DRAM:  512 MiB
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
Saving Environment to MMC... Writing to MMC(1)... OK
switch to partitions #0, OK
mmc1 is current device
Net:   eth0: ethernet@2188000
Normal Boot
=>
```

3.1.7.4. ログイン

起動が完了するとログインプロンプトが表示されます。初期状態では「root」ユーザーと、一般ユーザーである「atmark」ユーザーが存在しますが、「atmark」ユーザーは初期状態ではロックされていますので、「root」ユーザーでログインしてください。initial_setup.swu をインストールしていない場合、「root」ユーザーは初回ログイン時にパスワードを入力せずに新しいパスワードを促されます。

「root」ユーザーでログインし、 passwd atmark コマンドで「atmark」ユーザーのパスワードを設定することで、「atmark」ユーザーのロックが解除されます。設定するパスワードには大文字のアルファベット、小文字のアルファベット、0 から 9 までの数字、その他(記号・句読点など)を含める事ができます。

1. root でログイン

   初期パスワードを変更します。

   ```
   armadillo login: root
   You are required to change your password immediately (administrator enforced).
   New password: ❶
   Retype new password: ❷
   Welcome to Alpine!
   ```

   ❶ 新しいパスワードを入力します

   ❷ 新しいパスワードを再入力します

2. atmark でログイン

「atmark」ユーザーは初期状態ではロックされています。そのため、「root」ユーザーでログイン後に「atmark」ユーザーのパスワードを設定してから「atmark」ユーザーでログインします。

```
armadillo:~# passwd atmark ❶
New password:
Retype new password:
passwd: password updated successfully
armadillo:~# persist_file /etc/shadow ❷
armadillo:~# exit

Welcome to Alpine Linux 3.17
Kernel 5.10.185-1-at on an armv7l (/dev/ttymxc2)

armadillo login: atmark
Password: ❸
Welcome to Alpine!
```

❶ atmark ユーザーのパスワード変更コマンドです。「5.4.1. SWU イメージの作成」 を使用した場合には不要です。

❷ パスワードファイルを永続化します。

❸ 設定したパスワードでログインすることができます。



Armadillo BaseOS ではルートファイルシステムに overlayfs を採用しており、そのままではシステムが OFF すると内容は消えてしまいます。そのため persist_file コマンドが用意されています。このコマンドを利用することでファイル単位で変更を反映することができます。パスワードを設定した後には以下のコマンドを実行してください。

```
[armadillo ~]# persist_file /etc/shadow
```

persist_file コマンドに関する詳細は「6.2. persist_file について」を参照してください。

3.1.7.5. Armadillo の終了方法

eMMC や USB メモリ等に書き込みを行っている時に電源を切断すると、データが破損する可能性が有ります。安全に終了させる場合は、次のように poweroff コマンドを実行し、「reboot: Power down」と表示されたのを確認してから電源を切断します。

```
armadillo:~# poweroff
armadillo:~# zramswap                | * Deactivating zram swap device ...podman-atmark                ⏎
| * Stopping all podman containers ...local                    | * Stopping                            ⏎
local ...modemmanager            | * Stopping modemmanager ...avahi-daemon            | * Stopping     ⏎
avahi-daemon ...atmark-power-utils       | * Stopping atmark-power-utils ... [ ok ]
wwan-led                | * Stopping wwan-led ... [ ok ]

:（省略）
```

```
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processes
Requesting system poweroff
[ 2923.728066] imx2-wdt 20bc000.watchdog: Device shutdown: Expect reboot!
[ 2923.735159] reboot: Power down
```



halt コマンドで終了させた場合、「reboot: System halted」と表示されてから約 128 秒後、Armadillo は自動的に再起動します。確実に終了させるためにも poweroff コマンドを利用してください。



電源を再投入する際は、コンデンサに蓄えられた電荷を抜くため、電源を切断後、一定時間以上待つ必要があります。開発セット付属の AC アダプタの場合に必要な時間は以下のとおりです。

- DC プラグ側で電源を切断した場合 : 約 5 秒
- AC プラグ側で電源を切断した場合 : 約 1 分

コンデンサに蓄えられた電荷が抜ける前に電源を再投入した場合、電源シーケンスが守られず、起動しない等の動作不具合の原因となります。

3.1.8. ユーザー登録

アットマークテクノ製品をご利用のユーザーに対して、購入者向けの限定公開データの提供や大切なお知らせをお届けするサービスなど、ユーザー登録すると様々なサービスを受けることができます。サービスを受けるためには、「アットマークテクノ Armadillo サイト」にユーザー登録をする必要があります。

ユーザー登録すると次のようなサービスを受けることができます。

- 製品仕様や部品などの変更通知の閲覧・配信
- 購入者向けの限定公開データのダウンロード
- 該当製品のバージョンアップに伴う優待販売のお知らせ配信
- 該当製品に関する開発セミナーやイベント等のお知らせ配信

詳しくは、「アットマークテクノ Armadillo サイト」をご覧ください。

アットマークテクノ Armadillo サイト

https://armadillo.atmark-techno.com/

3.1.8.1. 購入製品登録

ユーザー登録完了後に、購入製品登録することで、「購入者向けの限定公開データ」をダウンロードすることができるようになります。

購入製品登録の詳しい手順は以下の URL をご参照ください。

Armadillo-IoT ゲートウェイ A6E 購入製品登録

https://armadillo.atmark-techno.com/armadillo-iot-a6e/register

以上で開発環境のセットアップと動作確認の手順は終了です。

3.2. アプリケーション開発の流れ

基本的な Armadillo-IoT ゲートウェイ A6E でのアプリケーション開発の流れを「図 3.44. アプリケーション開発の流れ」に示します。

本章では、「図 3.44. アプリケーション開発の流れ」に示す開発時の流れに沿って手順を紹介していきます。

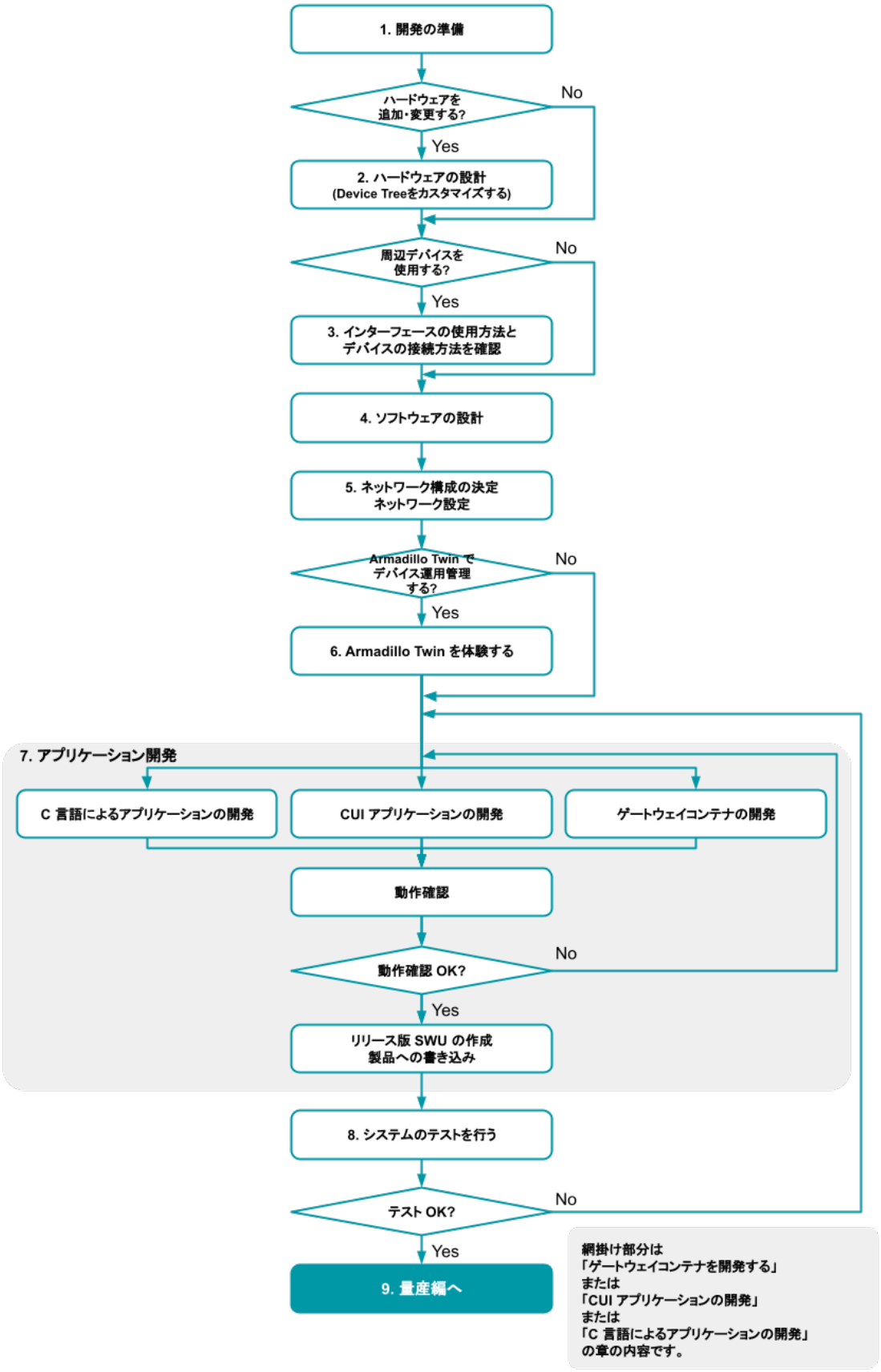


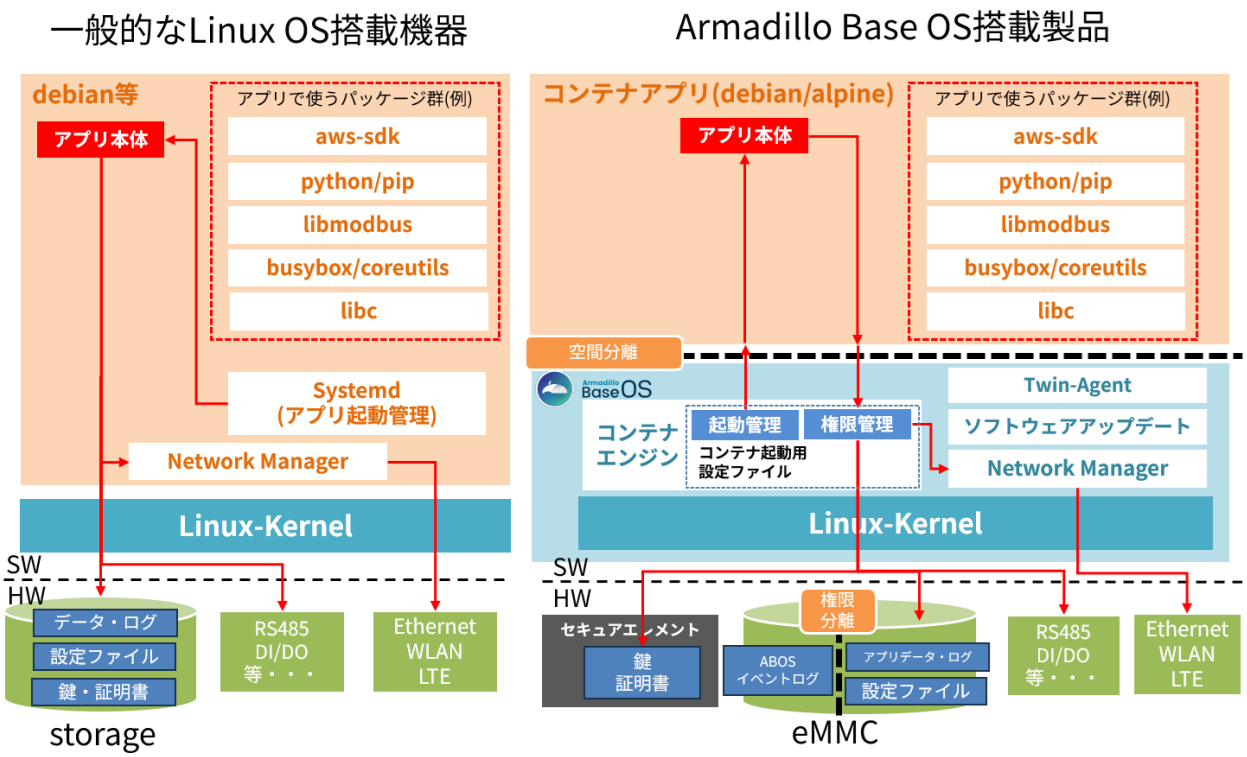
図 3.44 アプリケーション開発の流れ

1. 「3.1. 開発の準備」に従って開発環境の準備を行います。
2. 拡張基板を追加するなど、ハードウェアの追加・変更をする場合、「3.4. ハードウェアの設計」を行います。
    a. 拡張インターフェース(CON8)のピンを使用する場合「6.29. Device Tree をカスタマイズする」を参考にデバイスツリーのカスタマイズを行います。
3. Armadillo-IoT ゲートウェイ A6E に周辺デバイスを接続して使用する場合は、使用手順を「3.7. インターフェースの使用方法とデバイスの接続方法」で確認します。
4. 「3.8. ソフトウェアの設計」を行います。
5. 「3.9. ネットワーク設定」を行います。
6. Armadillo Twin を使用したデバイス運用管理を検討する場合、「3.14. Armadillo Twin を体験する」を行います。
7. アプリケーションの開発を行います。「図 3.44. アプリケーション開発の流れ」 の網掛け部分です。
    a. 「3.8. ソフトウェアの設計」 でゲートウェイコンテナを使用する場合は、「3.16. ゲートウェイコンテナアプリケーションの開発」を行います。
    b. 「3.8. ソフトウェアの設計」 でゲートウェイコンテナを使用せずに CUI アプリケーションを開発する場合は、シェスクリプトまたは Python で開発することを推奨します。その場合は「3.17. CUI アプリケーションの開発」を行います。
    c. C 言語で開発された既存のアプリケーションを Armadillo 上で動作させる必要がある、あるいは開発環境の制約によって C 言語でのアプリケーション開発が必要な場合、「3.18. C 言語によるアプリケーションの開発」を行います。
8. 開発したアプリケーションの動作確認が完了しましたら、「3.21. システムのテストを行う」を行います。
9. システムのテストが完了しましたら、「4. 量産編」へ進みます。

3.3. 開発前に知っておくべき Armadillo Base OS の機能・特徴

「2.1.3. Armadillo Base OS とは」にて Armadillo Base OS についての概要を紹介しましたが、開発に入るにあたってもう少し詳細な概要について紹介します。

3.3.1. 一般的な Linux OS 搭載組み込み機器との違い

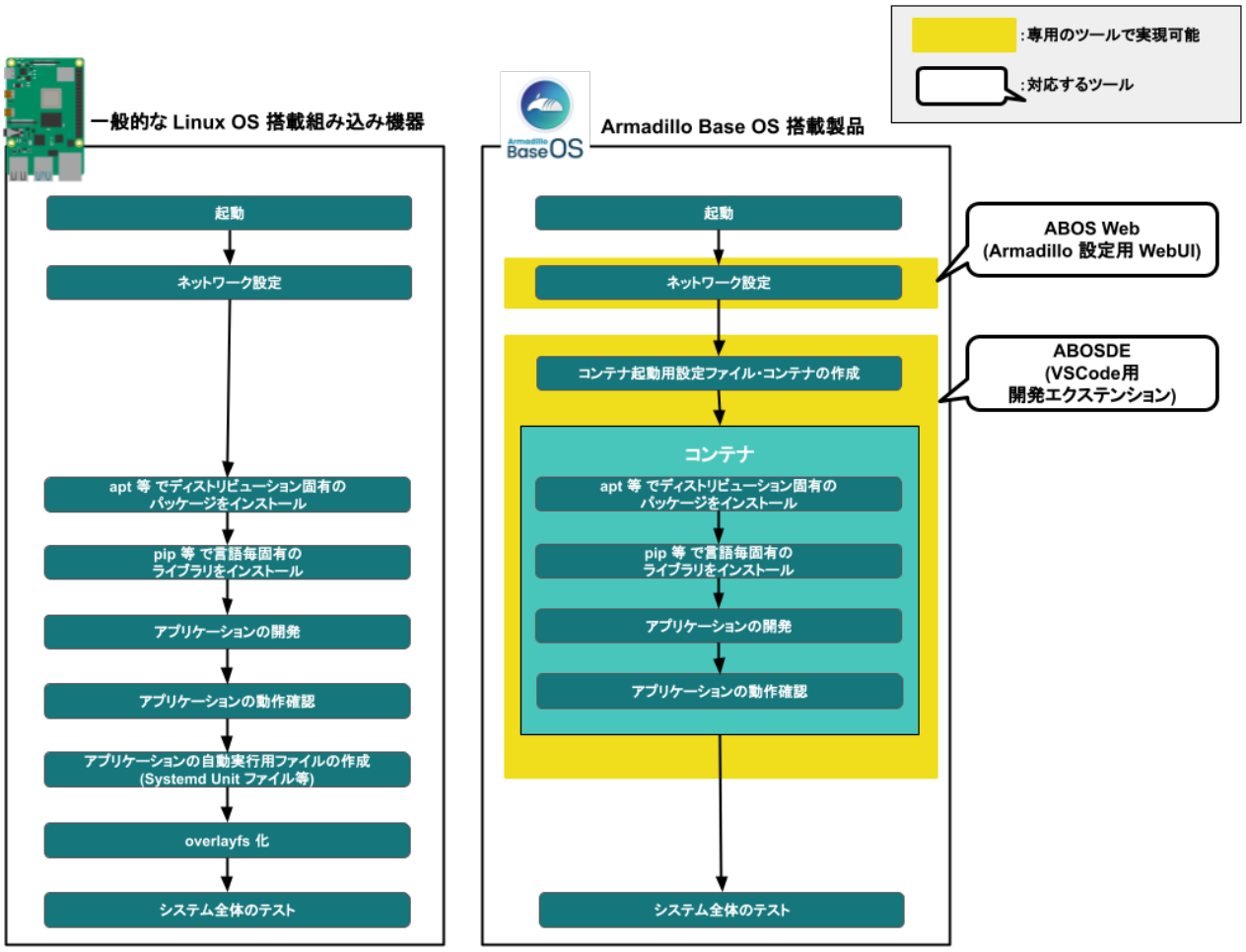


Linux OS 搭載組み込み機器ではアプリケーションの実行環境をユーザーランド上に直接用意し、Systemd などでアプリケーションを自動実行させるのが一般的です。Armadillo Base OS 搭載機器では、アプリケーションの実行環境をコンテナ内に用意して、コンテナ起動用設定ファイルを所定の場所に配置することでコンテナ(=アプリケーション)を自動実行させます。

また、Linux OS 搭載組み込み機器では、ストレージの保護のために overlayfs で運用するのが一般的です。そのため、アプリケーションが出力するログや画像などのデータは、 USB メモリなどの外部デバイスに保存する必要があります。Armadillo Base OS 搭載機器もルートファイルシステムが overlayfs 化されていますが、内部に USB メモリなどと同じように使用できるユーザーデータディレクトリを持っており、別途外部記録デバイスを用意しておく必要はありません。

Armadillo Base OS 搭載機器は、標準でセキュアエレメントを搭載しており、対応した暗号化方式の認証鍵や証明書を安全に保存・利用することが可能です。

3.3.2. Armadillo Base OS 搭載機器のソフトウェア開発手法



Armadillo Base OS 搭載機器上で動作するソフトウェアの開発は、基本的に作業用 PC 上で行います。

ネットワークの設定は ABOS Web という機能で、コマンドを直接打たずとも設定可能です。

開発環境として、ATDE(Atmark Techno Development Environment)という仮想マシンイメージを提供しています。その中で、ABOSDE(Armadillo Base OS Development Environment)という、Visual Studio Code にインストールできる開発用エクステンションを利用してソフトウェア開発を行います。

ABOSDE を使用することで、コンテナ及びコンテナ自動起動用設定ファイルの作成、コンテナ内におけるパッケージのインストール、コンテナ内で動作するアプリケーション本体の開発をすべて VS Code 内で行うことができます。

3.3.3. アップデート機能について

Armadillo-IoT ゲートウェイ A6E では、開発・製造・運用時にソフトウェアを書き込む際に、SWUpdate という仕組みを利用します。

3.3.4. SWUpdate とは

SWUpdate は、デバイス上で実行されるプログラムで、ネットワーク/ストレージ経由でデバイスのソフトウェアを更新することができます。Stefano Babic, DENX software engineering, Germany によってオープンソースで開発が進められています。

Armadillo-IoT ゲートウェイ A6E では、 SWUpdate を利用することで次の機能を実現しています。

- 機密性、完全性、真正性の担保
- A/B アップデート(アップデートの二面化)
- リカバリーモード
- ソフトウェアの圧縮、暗号化、署名付与
- Armadillo Twin でのリモートアップデート対応
- Web サーバーでのリモートアップデート対応
- ダウングレードの禁止



2024 年 2 月までは、hawkBit の WebUI を利用したアップデートも紹介していましたが、hawkBit は 2024 年 3 月 22 日 に行われたバージョン 0.5.0 へのアップデートで、これまで採用していた Web UI を廃止しました。これに伴い、今後 OTA によるアップデートを行いたい場合は、Armadillo Twin [https://armadillo.atmark-techno.com/guide/armadillo-twin/] の利用を推奨します。

なお、hawkBit 0.4.1 の配布は継続していますので、こちらを利用する場合は Armadillo-IoT ゲートウェイ A6E 開発用ツール [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/tools] から「Hawkbit docker-compose コンテナ」 をダウンロードして展開してください。

hawkBit に関する詳細な情報は hawkBit 公式サイト [https://eclipse.dev/hawkbit/] を参照してください。

3.3.4.1. SWU イメージとは

swu パッケージは、SWUpdate 独自のソフトウェアの配布フォーマットです。SWUpdate では、1 回のアップデートは 1 つの swu パッケージで行われます。

swu パッケージには、次のような様々なものを含めることができます。

- アップデート対象のイメージファイル
- アップデート対象のイメージファイルのチェックサム
- アップデート前後に実行するスクリプト
- 書き込み先ストレージの情報
- U-Boot 環境変数の書き換え情報
- ソフトウェアのバージョン情報
- etc…

SWU イメージは `swupdate` (https://sbabic.github.io/swupdate/swupdate.html) によって Armadillo Base OS 上で検証とインストールが実行されます。SWU イメージを Armadillo に転送するための方法は、用途や状況に合わせて様々な方法を用意しています。例えば、 USB メモリから読み取る、ウェブサーバーからダウンロードするなどです。

3.3.4.2. 機密性、完全性、真正性の担保

ユーザーは SWU イメージをネットワーク/ストレージ経由で Armadillo にインストールします。

インターネットを通じて Armadillo にインストールする場合、以下の脅威が存在することが考えられます。

- 攻撃者が正規のユーザーを偽りデータをインストールする（なりすまし）
- データの一部を悪意のあるコードに書き換えられる（改ざん）
- データを盗み見される（盗聴）

Armadillo Base OS では暗号化技術、SHA-256 によるハッシュ化、デジタル署名を駆使することで、インストールするデータに対する機密性、完全性、真正性を保証します。

それらの機能は SWUpdate によって実現しています。SWUpdate は以下の対策を提供します。

- SWU イメージ内の Armadillo にインストールするデータを暗号化する
- デジタル署名により正規の SWU イメージであることを保証する
- 復号したデータに対してもチェックサムの値を計算して、インストールするデータが正しいことを保証する

これらの対策により、たとえ攻撃者が不正な SWU イメージを Armadillo に送信したとしてもデジタル署名により正規の SWU イメージでないことが分かります。

攻撃者がインターネット上で SWU イメージ内のデータを書き換えたとしても、インストール前にそのデータに対してチェックサムが正しいかを確認します。そのため、不正なデータが Armadillo にインストールされることはありません。

また、攻撃者がネットワーク上で SWU イメージのデータを盗み見たとしても暗号化されているので、重要なデータが漏洩することもありません。

3.3.4.3. A/B アップデート(アップデートの二面化)

A/B アップデートは、Flash メモリにパーティションを二面確保し、アップデート時には交互に利用する仕組みです。

常に使用していない方のパーティションを書き換えるため次の特徴を持ちます。

- ○ アップデートによって動作中のソフトウェアは破壊されない
- ○ 書き込みが電源断などで中断しても、すぐに復帰出来る
- ○ 機器が動作中に書き込みが出来る
- × 使用 Flash メモリ量が増える

3.3.4.4. ロールバック(リカバリー)

アップデート直後に起動に失敗した場合、起動可能な状態へ復帰するためアップデート前の状態にロールバックします。

ロールバック状態の確認は 「6.25. ロールバック状態を確認する」 を参照してください。

自動ロールバックが動作する条件は以下の通りです：

- アップデート直後の再起動、または「 abos-ctrl rollback-clone 」コマンドを実行した後(アップデートが成功した後では古いバージョンに戻りません)
- 以下のどちらかに該当した場合：
  - rootfs にブートに必要なファイルが存在しない（/boot/uImage, /boot/armadillo.dtb)
  - 起動を 3 回試みて、Linux ユーザーランドの「reset_bootcount」サービスの起動まで至らなかった

また、ユーザースクリプト等で「abos-ctrl rollback」コマンドを実行した場合にもロールバック可能となります。このコマンドで「 --allow-downgrade 」オプションを設定すると古いバージョンに戻すことも可能です。

いずれの場合でもロールバックが実行されると /var/at-log/atlog にログが残ります。



Armadillo Base OS 3.19.1-at.4 以前のバージョンではアップデート直後の条件が存在しなかったため、古いバージョンに戻ることができる問題がありました。

最新の Armadillo Base OS へのアップデートを推奨しますが、上記バージョン以前の Armadillo Base OS をご利用でダウングレードを防ぎたい場合は、以下のコマンドを入力することで回避可能です：

```
[armadillo ~]# sed -i -e 's/fw_setenv bootcount/& ¥&¥& fw_setenv
upgrade_available/' /etc/init.d/reset_bootcount
[armadillo ~]# tail -n 3 /etc/init.d/reset_bootcount
        fw_setenv bootcount && fw_setenv upgrade_available
        eend $? "Could not set bootloader env"
}
[armadillo ~]# persist_file -v /etc/init.d/reset_bootcount
'/mnt/etc/init.d/reset_bootcount' -> '/target/etc/init.d/
reset_bootcount'
```

3.3.4.5. SWU イメージのインストール

イメージをインストールする方法として以下に示すような方法があります。

- 手元でイメージをインストールする方法
  - ABOS Web を使用した手動インストール
  - ABOSDE から ABOS Web を使用した手動インストール

- USB メモリまたは microSD カードからの自動インストール
- 外部記憶装置からイメージのインストール（手動）

- リモートでイメージをインストールする方法
  - Armadillo Twin を使用した自動インストール
  - ウェブサーバーからイメージのインストール（手動）
  - ウェブサーバーからの定期的な自動インストール

それぞれのインストール方法の詳細については、以下に記載しております。もし、作成した SWU イメージのインストールに失敗する場合は、「6.3.5. swupdate がエラーする場合の対処」をご覧ください。

- ABOS Web を使用した手動インストール

  Armadillo-IoT ゲートウェイ A6E で動作している Web アプリケーションの ABOS Web を使用してアップデートすることができます。「6.12.4. SWU インストール」を参考にしてください。

- ABOSDE から ABOS Web を使用した手動インストール

  VS Code 拡張機能の ABOSDE を使用することで、Armadillo-IoT ゲートウェイ A6E で動作している ABOS Web 経由でアップデートすることができます。「6.13.5. Armadillo に SWU をインストールする」を参考にしてください。

- USB メモリまたは microSD カードからの自動インストール

  Armadillo-IoT ゲートウェイ A6E に USB メモリを接続すると自動的にアップデートが始まります。アップデート終了後に Armadillo-IoT ゲートウェイ A6E は自動で再起動します。

  USB メモリや microSD カードを vfat もしくは ext4 形式でフォーマットし、作成した.swu のファイルをディレクトリを作らずに配置してください。



ATDE 上で USB メモリ/microSD カードのパーティションを作成・フォーマットする方法

https://armadillo.atmark-techno.com/howto/atde-partition-howto

```
[ATDE ~/mkswu]$ df -h
Filesystem           Size  Used Avail Use% Mounted on
:（省略）
/dev/sda1        15G  5.6G  9.1G  39% /media/USBDRIVE ❶
[ATDE ~/mkswu]$ cp initial_setup.swu /media/USBDRIVE/ ❷
[ATDE ~/mkswu]$ umount /media/USBDRIVE ❸
```

❶ USB メモリがマウントされている場所を確認します。

❷ ファイルをコピーします。

❸ /media/USBDRIVE をアンマウントします。コマンド終了後に USB メモリを取り外してください。

エラーの場合、/var/log/message に保存されます。例えば、コンソールで証明書が間違っているイメージのエラーは以下の様に表示されます。

```
[armadillo ~]# tail /var/log/messages
Nov 19 10:48:42 user.notice swupdate-auto-update: Mounting sda0 on /mnt
Nov 19 10:48:42 user.notice swupdate-auto-update: Trying update /mnt/initial_setup.swu
Nov 19 10:48:42 user.info swupdate: START Software Update started !
Nov 19 10:48:42 user.err swupdate: FAILURE ERROR : Signature verification failed ❶
Nov 19 10:48:42 user.err swupdate: FAILURE ERROR : Compatible SW not found
Nov 19 10:48:42 user.err swupdate: FATAL_FAILURE Image invalid or corrupted. Not installing ...
```

❶ 証明書エラーのメッセージ。

・外部記憶装置からイメージのインストール（手動）

USB メモリや microSD カード等の外部記憶装置のルートディレクトリ以外に SWU イメージを保存して、イメージのインストールを行います。ルートディレクトリに保存すると自動アップデートが行われますので、/var/log/messages を確認してください。

以下は外部記憶装置が/dev/mmcblk1p1(microSD カード）として認識された場合に、イメージのインストールを行う例です。

```
[armadillo ~]# mount /dev/mmcblk1p1 /mnt
[armadillo ~]# swupdate -i /mnt/swu/initial_setup.swu
SWUpdate v5f2d8be-dirty

Licensed under GPLv2. See source distribution for detailed copyright notices.

[INFO ] : SWUPDATE running :  [main] : Running on AGX4500 Revision at1
[INFO ] : SWUPDATE started :  Software Update started !
[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying current os over
[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers
[INFO ] : SWUPDATE running :  [read_lines_notify] : swupdate triggering reboot!
Killed
```

・Armadillo Twin を使用した自動インストール

Armadillo Twin で Armadillo-IoT ゲートウェイ A6E を複数台管理してアップデートすることができます。「5.5. Armadillo Twin から複数の Armadillo をアップデートする」を参考にしてください。

・ウェブサーバーからイメージのインストール（手動）

SWU イメージをウェブサーバーにアップロードして、イメージのインストールを行います。以下は、http://server/initial_setup.swu のイメージをインストールする例です。

```
[armadillo ~]# swupdate -d '-u http://server/initial_setup.swu'
SWUpdate v5f2d8be-dirty

Licensed under GPLv2. See source distribution for detailed copyright notices.
```

```
[INFO ] : SWUPDATE running :  [main] : Running on AGX4500 Revision at1
[INFO ] : SWUPDATE running :  [channel_get_file] : Total download size is 25 kB.
[INFO ] : SWUPDATE started :  Software Update started !
[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying current os over
[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers
[INFO ] : SWUPDATE running :  [read_lines_notify] : swupdate triggering reboot!
Killed
```

・ウェブサーバーからの定期的な自動インストール

swupdate-url を有効にしたら、定期的にチェックしてインストールします。以下はサービスの有効化とタイミングの設定の例です。

```
[armadillo ~]# rc-update add swupdate-url ❶
[armadillo ~]# persist_file /etc/runlevels/default/swupdate-url ❷
[armadillo ~]#
    echo https://download.atmark-techno.com/armadillo-iot-a6e/image/baseos-6e-latest.swu ¥
                        > /etc/swupdate.watch ❸
[armadillo ~]# echo 'schedule="0 tomorrow"' > /etc/conf.d/swupdate-url
[armadillo ~]# echo 'rdelay="21600"' >> /etc/conf.d/swupdate-url ❹
[armadillo ~]# persist_file /etc/swupdate.watch /etc/conf.d/swupdate-url ❺
```

❶ swupdate-url サービスを有効します。

❷ サービスの有効化を保存します。

❸ イメージの URL を登録します。一行ごとにイメージの URL を設定することができ、複数行にイメージの URL を設定することができます。

❹ チェックやインストールのスケジュールを設定します。

❺ 変更した設定ファイルを保存します。

USB メモリからのアップデートと同様に、ログは/var/log/messages に保存されます。



initial_setup のイメージを作成の際に /usr/share/mkswu/examples/enable_swupdate_url.desc を入れると有効にすることができます。

3.3.5. ファイルの取り扱いについて

Armadillo Base OS ではルートファイルシステムに overlayfs を採用しています。

その為、ファイルを変更した後 Armadillo の電源を切ると変更内容は保持されません。開発中などに rootfs の変更内容を保持するには、変更したファイルに対して persist_file コマンドを使用します。

```
[armadillo ~]# echo test > test
[armadillo ~]# persist_file -v test
'/root/test' -> '/mnt/root/test'
```

図 3.45 persist_file コマンド実行例

persist_file コマンドの詳細については、「6.2. persist_file について」を参照してください。

また、 SWUpdate によってルートファイルシステム上に配置されたファイルについては、 persist_file を実行しなくても保持されます。開発以外の時は安全のため、 persist_file コマンドではなく SWUpdate による更新を実行するようにしてください。

3.3.5.1. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)

「3.3.5. ファイルの取り扱いについて」 にて、 Armadillo Base OS 上のファイルは通常、 persist_file コマンドを実行せずに電源を切ると変更内容が保存されないと紹介しましたが、「表 3.2. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)」に示すディレクトリ内にあるファイルはこの限りでありません。

表 3.2 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)

ディレクトリ	備考
/var/app/volumes	SWUpdate の最中や後も保持され続けます。ロールバックが発生しても、アップデート前の状態には戻りません。ログやデータベースなど、アプリケーションが動作中に作成し続けるようなデータはこのディレクトリに保存してください。
/var/app/rollback/volumes	SWUpdate の最中や後も保持され続けます。ロールバックが発生すると、アップデート前の状態に戻ります。コンフィグファイルなど、アプリケーションのバージョンに追従してアップデートするようなデータはこのディレクトリに保存してください。



コンテナを前のバージョンに戻した場合(ロールバック)、/var/app/rollback/volumes/ のデータの前のバージョンに戻ります。

その為、アプリケーションのバージョンに依存するようなデータは /var/app/rollback/volumes/ に入れることを推奨します。

mkswu の swdesc_files (--extra-os 無し) と podman_start の add_volumes では、相対パスはそのディレクトリをベースにします。/var/app/rollback/volumes/myvolume は myvolume で簡潔に指定できます。



Copy-on-Write (CoW) について。

この二つの volumes ディレクトリは btrfs と呼ばれるファイルシステムに保存されています。btrfs ではデータは Copy on Write（CoW）を使ってデータ完全性を保証しますが、その保証にはコストがあります。

数百 MB のファイルに小さな変更を頻繁に行う場合 CoW を無効化することを推奨します。CoW を無効化されたファイルにチェックサムが入らなくなりますので、極端な場合以外に残してください。

```
[armadillo ~]# cd /var/app/volumes/
[armadillo /var/app/volumes]# mkdir database
[armadillo /var/app/volumes]# chattr +C database ❶
[armadillo /var/app/volumes]# echo example data > database/example
[armadillo /var/app/volumes]# lsattr database/ ❷
---------------C---- database/example
```

図 3.46 chattr によって copy-on-write を無効化する例

❶ chattr +C でディレクトリに NoCow を設定します。これから作成されるファイルが NoCow で作成されます。すでに存在していたファイルに影響ないのでご注意ください。

❷ lsattr 確認します。リストの C の字があればファイルが「no cow」です。

3.3.6. インストールディスクについて

インストールディスクは、 Armadillo の eMMC の中身をまとめて書き換えることのできる microSD カードを指します。インストールディスクは、インストールディスクイメージを microSD カードに書き込むことで作成できます。

インストールディスクには以下の 2 つの種類があります。

- 初期化インストールディスク

  Armadillo-IoT ゲートウェイ A6E インストールディスクイメージ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/disc-image] にある標準イメージです。Armadillo を初期化する際に使用します。

- 開発が完了した Armadillo-IoT ゲートウェイ A6E をクローンするためのインストールディスク。

  量産時など、特定の Armadillo を複製する際に使用されます。詳しくは、「4. 量産編」で説明します。

3.3.6.1. インストールディスクの作成

インストールディスクの作成方法は「3.1.4.1. 初期化インストールディスクの作成」を参照してください。

参照先では初期化インストールディスクの場合の手順を示していますが、「6.30. Armadillo のソフトウェアをビルドする」 でビルドしたイメージについても同じ手順になります。その際のインストールディスクイメージ（.img）は、以下のコマンドを実行して作成してください。

```
[ATDE ~/build-rootfs-[VERSION]]$ ./build_image.sh --board a6e
:（省略）
[ATDE ~/build-rootfs-[VERSION]]$ ls baseos-6e*img
baseos-6e-[VERSION].img
[ATDE ~/build-rootfs-[VERSION]]$ ./build_image.sh --board a6e ¥
        --boot ~/uboot-[VERSION]/u-boot-dtb.imx ¥
        --installer ./baseos-6e-[VERSION].img
```

コマンドの実行が完了すると、baseos-6e-[VERSION]-installer.img というファイルが作成されていますので、こちらを使用してください。

3.3.6.2. インストールディスクを使用する

インストールディスクを使用する方法については、「3.1.4.2. インストールディスクを使用する」を参照してください。

3.4. ハードウェアの設計

Armadillo-IoT ゲートウェイ A6E の機能拡張や信頼性向上のための設計情報について説明します。

3.4.1. 信頼性試験データについて

Armadillo-IoT ゲートウェイ A6E の各種信頼性試験データを、「アットマークテクノ Armadillo サイト」から「購入者向けの限定公開データ」としてダウンロード可能ですのでご確認ください。

3.4.2. 放射ノイズ

拡張インターフェースを使用して、Armadillo-IoT ゲートウェイ A6E と拡張基板を接続すると、放射ノイズが問題になる場合があります。放射ノイズを減らすために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6E の GND(固定穴等)と拡張基板の GND を太い導線や金属スペーサ等で接続する
- 未使用の拡張ピンは Low レベル出力とする
- 使用する拡張ピンはコンデンサ(1000pF 程度)を介して GND と接続する
- ハーネスケーブル等で拡張する場合は、最短で接続する。
- シールド付きのケーブルを使用する
  - 長さが余る場合は、ケーブルを折りたたむ
  - シールドは拡張基板の GND に接続する

3.4.3. ESD/雷サージ

Armadillo-IoT ゲートウェイ A6E の ESD 耐性を向上させるために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6E を金属筐体に組み込み、GND(固定穴等)を金属ねじ等で接続する
- 金属筐体を接地する

また、Armadillo-IoT ゲートウェイ A6E に接続されたケーブルが屋外に露出するような設置環境では、ケーブルに侵入した雷サージ等のストレスによりインターフェース回路が破壊される場合があります。ストレスへの耐性を向上させるために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6E と通信対向機の GND 接続を強化する
- シールド付きのケーブルを使用する

3.4.4. 拡張基板の設計

Armadillo-IoT ゲートウェイ A6E の拡張インターフェース(CON8)には、複数の機能をもった信号線が接続されており、様々な機能拡張が可能です。



拡張インターフェース(CON8)のピン配置マルチプレクス表は 「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex]からダウンロードしてください。

回路設計時の留意点と、拡張してケースに収める場合の制限事項について説明します。

ケースについては、垂直方向に拡張して製品付属のケースに収める場合と、水平方向に拡張してオプション品の Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M) (OP-CASEA6E-PLA-20)に収める場合の 2 通りの説明をします。

3.4.4.1. 回路設計時の留意点

拡張インターフェース(CON8)の信号配列を、「表 3.50. 拡張インターフェース(CON8) 信号配列」に記載しています。拡張インターフェース(CON8)のピンのうち、プルアップ/ダウン抵抗が基板上で接続されているピンは、電源投入時または再起動時、プル抵抗で決められた入力レベルに保持する必要があります。以下の方法を参考に拡張基板の回路設計をしてください。

- 3-State バッファなどを使用してピンと拡張基板の回路を一時的に切り離す
- ピンを出力ピンとして使用する

以下は 3-State バッファを使用して電源投入時または再起動時にピンの High/Low レベルを保持する回路例です。プルダウン抵抗が基板上で接続されているピンを 3-State バッファのイネーブル信号に使用することで、電源投入から Armadillo が起動するまでの間、ピンと拡張基板の回路を切り離します。

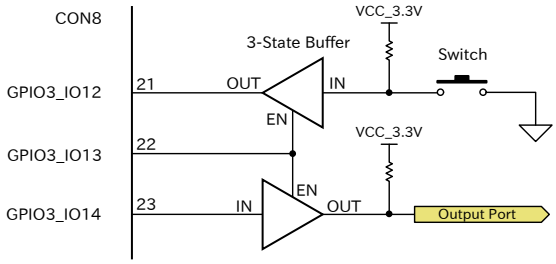


図 3.47 3-State バッファを使用した回路例

表 3.3 CON8 3-State バッファを使用した回路の IO ピン仕様

ピン番号	ピン名	基板上のプル抵抗	説明
21	GPIO3_IO12	10kΩ プルダウン	GPIO3_IO13 が Low レベルの場合は基板上のプル抵抗の状態が入力されます。GPIO3_IO13 が High レベルの場合は押しボタンの状態が入力されます。(High:ボタンが押されていない状態、Low:ボタンが押された状態)
22	GPIO3_IO13	10kΩ プルダウン	3-State バッファの EN ピンを制御する出力です。(Low:3-State バッファの出力は Hi-Z となり拡張基板側の回路と IO ピンは切り離された状態、High:3-State バッファの出力は IN 側の入力レベルに従い出力され拡張基板側の回路と IO ピンが接続された状態)

ピン番号	ピン名	基板上のプル抵抗	説明
23	GPIO3_IO14	10kΩ プルダウン	GPIO3_IO13 が Low レベルの場合は 3-State バッファの OUT は Hi-Z となります。GPIO3_IO13 が High レベルの場合は GPIO3_IO14 の出力レベルが 3-State バッファを通して Output Port 側に出力されます。

また、ピンを出力専用として使用することでも電源投入時または再起動時にプル抵抗で決められた入力レベルを保持することができます。以下の回路例では Armadillo-IoT ゲートウェイ A6E 基板上のプルダウン抵抗により電源投入時はリレーが OFF の状態を保持し、アプリケーションから GPIO3_IO12 を High レベル出力にすることで、リレーが ON になります。

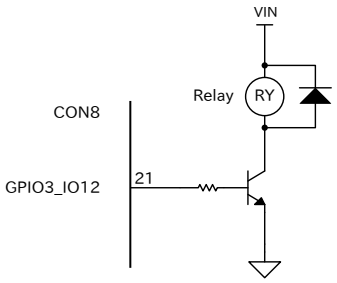


図 3.48 ピンを出力専用で使用した回路例

3.4.4.2. 垂直方向に拡張

垂直方向に拡張基板を配置することで、ケースのサイズを変えることなく機能拡張が可能です。

拡張インターフェース(CON8)と拡張基板はストレートアングルのピンヘッダおよびピンソケットで接続します。一般的なピンソケットを実装した場合、嵌合高さは約 11mm となります。

拡張ボード固定用に、φ2.3mm の穴を 2 箇所用意しており、M2 のスペーサーとねじで拡張基板を固定することが可能です。

拡張基板固定用穴の穴位置等基板の詳細寸法、ケースに収めるための高さ制限については「3.4.8. 基板形状図」、「3.4.9. ケース形状図」をご確認ください。

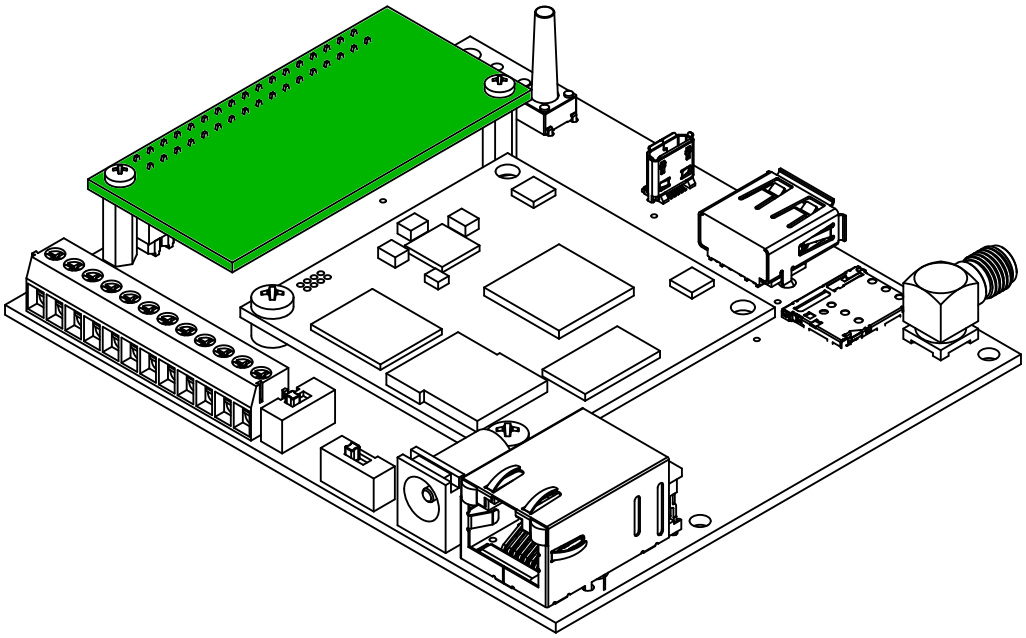


図 3.49 垂直方向に拡張基板を配置した場合の接続例

本製品と拡張基板を垂直方向に接続するための推奨コネクタは「表 3.4. 推奨コネクタ (垂直接続)」のとおりです。

表 3.4 推奨コネクタ (垂直接続)

搭載基板	メーカー	型番
本製品側(ピンヘッダ)	Wurth Elektronik	61303421121
拡張基板側(ピンソケット)	Wurth Elektronik	61303421821



ケース内側の側面は勾配になっており、本体基板からケース天面に行くにつれて内側が狭くなります。設計上、本体基板とケース内側のマージンは約 0.2mm です。11mm 離れた拡張基板の場合、マージンは約 0.1mm になります。お互いのピンコネクタ実装位置のずれなどにより、マージンがなくなる可能性もあるため、本体基板の寸法よりもさらに 0.2mm 程度のマージンを設けることをお勧めします。

3.4.4.3. 水平方向に拡張

オプション品の Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M) (OP-CASEA6E-PLA-20)を使用することで、水平方向に拡張基板を配置できます。

拡張インターフェース(CON8)と拡張基板はライトアングルのピンヘッダおよびピンソケットで接続します。

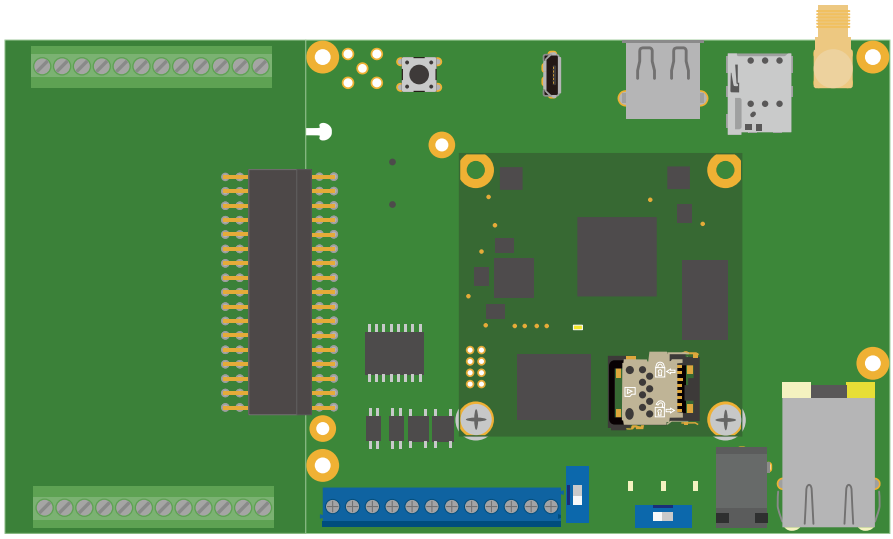


図 3.50 水平方向に拡張基板を配置した場合の接続例

本製品と拡張基板を接続するための推奨コネクタは「表 3.5. 推奨コネクタ (水平接続)」のとおりです。

表 3.5 推奨コネクタ (水平接続)

	メーカー	型番
本製品側推奨コネクタ	Sullins Connector Solutions	PRPC017DBAN-M71RC
拡張基板側推奨コネクタ	Sullins Connector Solutions	PPPC172LJBN-RC

搭載例は以下のとおりです。

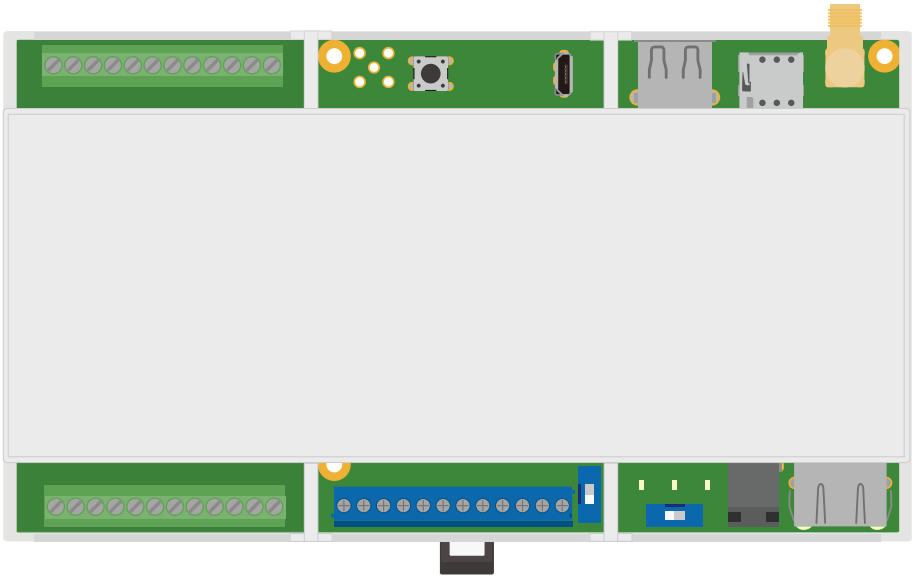


図 3.51 Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M) 搭載例

Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M) の詳細は「6.38.1. Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)」をご参照ください。

3.4.4.4. 水平方向に拡張する場合の基板形状図

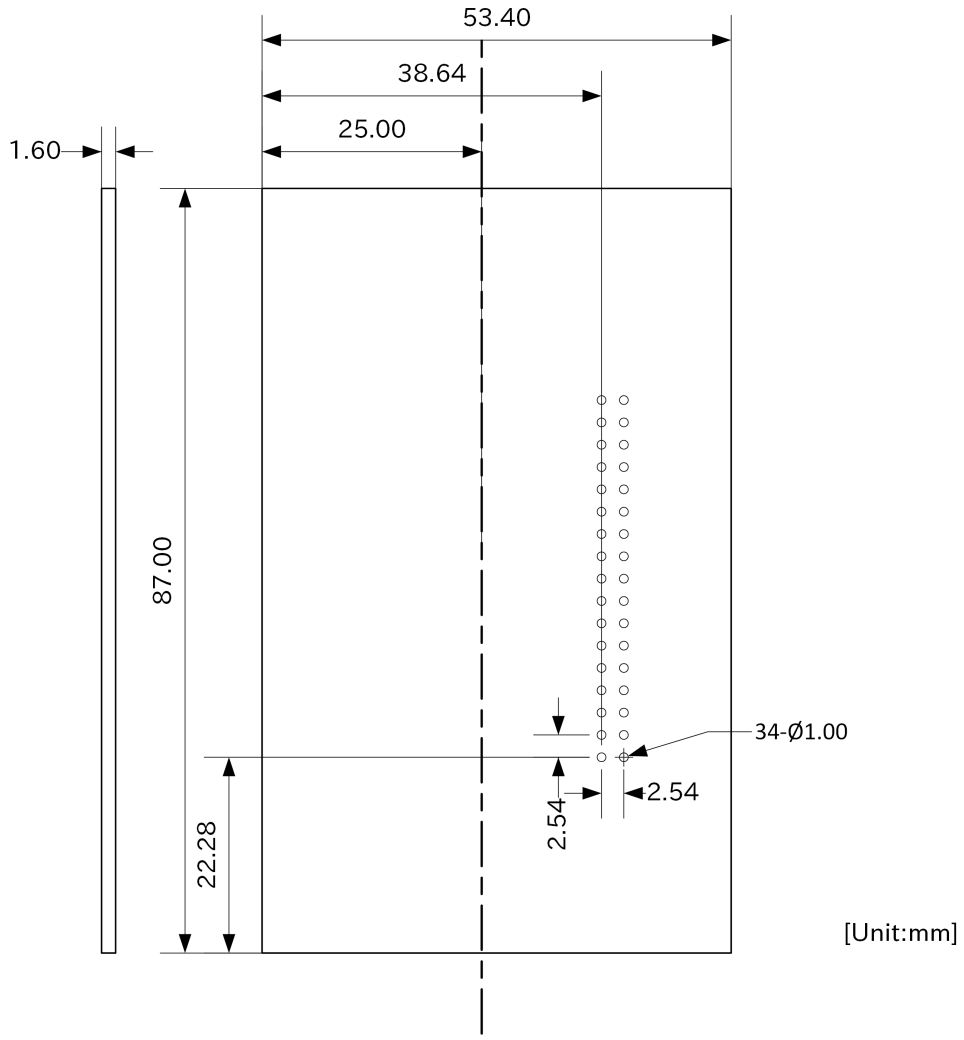


図 3.52 水平方向に拡張する場合の基板形状図

3.4.4.5. 水平方向に拡張する場合の部品搭載制限

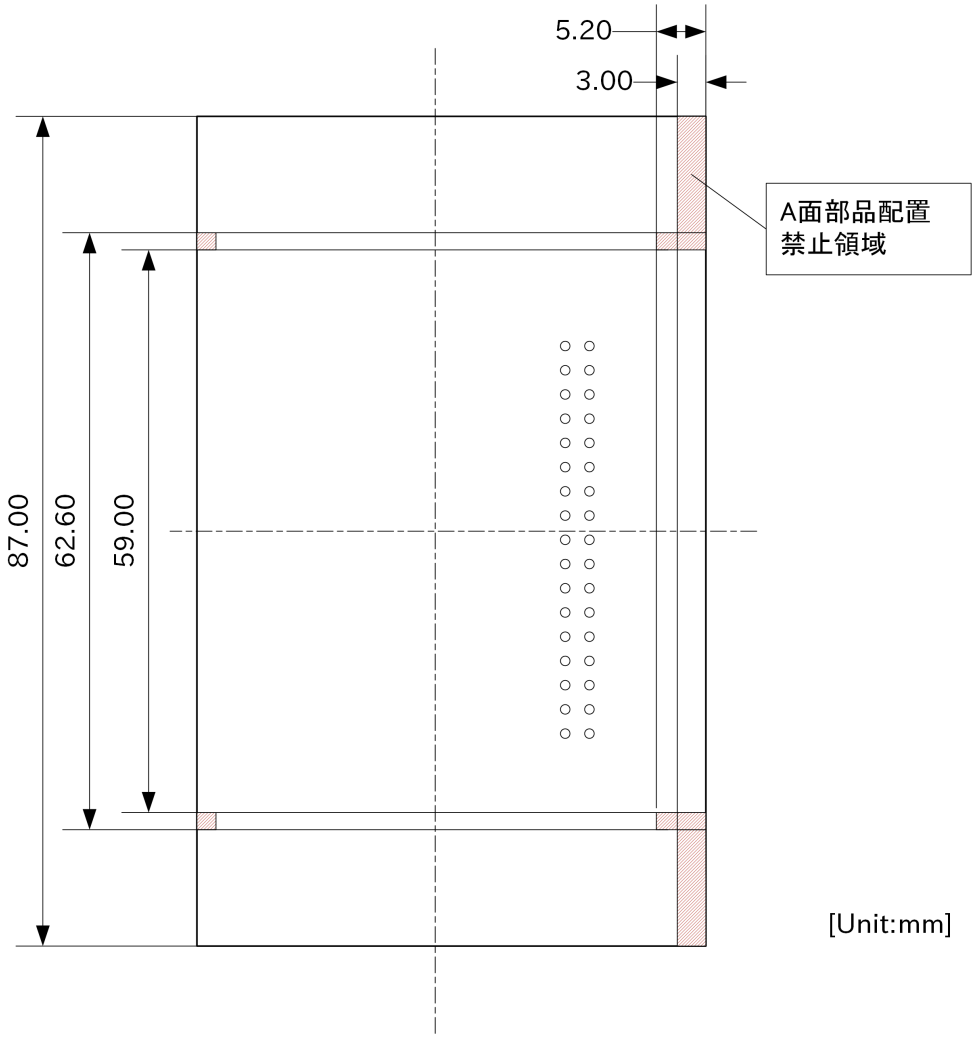


図 3.53 水平方向に拡張する場合の部品搭載制限(A 面側)

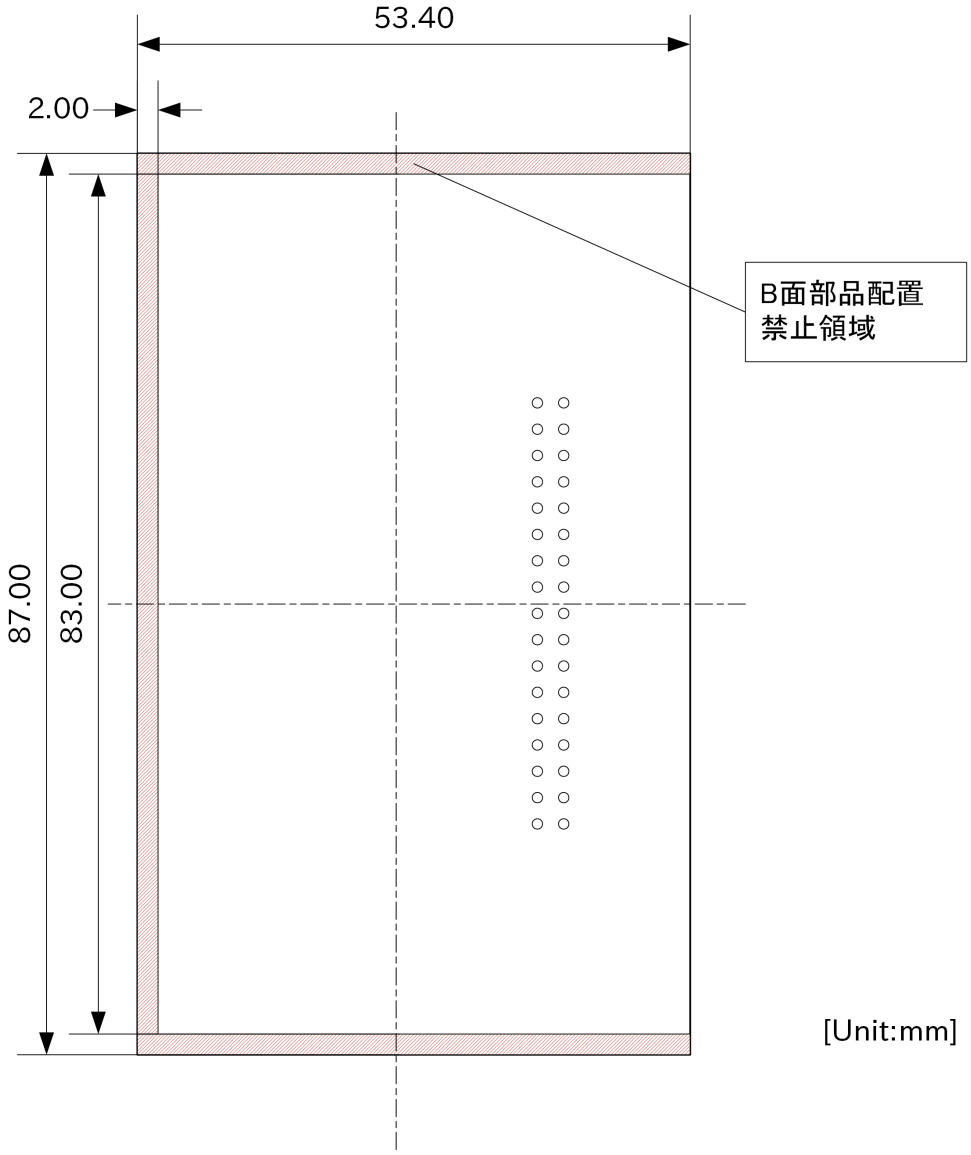


図 3.54 水平方向に拡張する場合の部品搭載制限(B 面側)

3.4.5. 電気的仕様

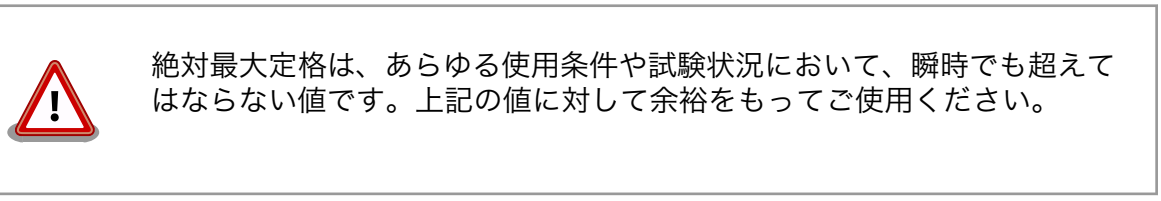
3.4.5.1. 絶対最大定格

表 3.6 絶対最大定格

項目	記号	Min.	Max.	単位	備考
電源電圧	VIN	-0.3	26.4	V	CON5,CON6
入出力電圧 (GPIO 信号)	VI,VO	-0.3	OVDD+0.3	V	CON8(OVDD=VCC_3.3V)
入出力電圧 (RS-485 信号)	VI_RS485 VO_RS485	-8.0	12.5	V	CON6(DATA+,DATA-)
入力電圧 (接点入力)	VI_DI	-26.4	26.4	V	CON6(DI1,DI2,COM), CON22(DI3～DI10,COM) [a]

項目	記号	Min.	Max.	単位	備考
入力電圧 (アナログ入力)	VI_AI	-0.5	5.7	V	CON21(AI1A,AI1B,AI2A,AI2B,AI3A,AI3B,AI4A,AI4B) [a]
入力電流 (アナログ入力)	CI_AI	-2	22.8	mA	CON21(AI1A,AI1B,AI2A,AI2B,AI3A,AI3B,AI4A,AI4B) [a]
出力耐圧 (接点出力)	Voff_DO	-60	60	V	CON6(DO1A,DO1B,DO2A,DO2B)
RTC バックアップ電源電圧	RTC_BAT	-0.3	5.5	V	CON10
動作温度範囲	Topr	-20	60	℃	結露なきこと

[a]+Di8+Ai4 タイプのみ



3.4.5.2. 推奨動作条件

表 3.7 推奨動作条件

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	VIN	8	12	26.4	V	CON5,CON6
RTC バックアップ電源電圧	RTC_BAT	2.4	3	3.6	V	CON10,対応電池: CR1220 等

3.4.5.3. 入出力仕様

・ 電源出力仕様

表 3.8 電源出力仕様

項目	記号	Min.	Typ.	Max.	単位	備考
5V 電源電圧	VCC_5V	4.75	5	5.25	V	CON8
3.3V 電源電圧	VCC_3.3V	3.102	3.3	3.498	V	CON8
USB VBUS 電圧	USB_OTG1_VBUS	4.75	5	5.25	V	CON9

・ 入出力インターフェース(CON6)の入出力仕様

表 3.9 入出力インターフェース(CON6)の入出力仕様

項目		内容
接点入力	入力インピーダンス	4.7 kΩ
	定格電圧	DC 8~26.4 V
	入力 ON 電流	1.0 mA 以上
	入力 OFF 電流	0.2 mA 以下
接点出力	定格電圧	最大 48 V
	定格電流	最大 500 mA
	応答時間	2ms 以内
	出力形式	無極性
絶縁耐圧		2kV

・ 拡張インターフェース(CON8)の入出力仕様 [2]

表 3.10 拡張インターフェース(CON8)の入出力仕様(OVDD = VCC_3.3V)

項目	記号	Min.	Max.	単位	備考
ハイレベル出力電圧	VOH	OVDD-0.15	OVDD	V	IOH = -0.1mA, -1mA
ローレベル出力電圧	VOL	0	0.15	V	IOL = 0.1mA, 1mA
ハイレベル入力電圧[a]	VIH	0.7xOVDD	OVDD	V	-
ローレベル入力電圧[a]	VIL	0	0.3xOVDD	V	-
入力リーク電流(no Pull-up/Pull-down)	IIN	-1	1	μA	-
Pull-up 抵抗(5kΩ)	-	4	6	kΩ	-
Pull-up 抵抗(47kΩ)	-	37.6	56.4	kΩ	-
Pull-up 抵抗(100kΩ)	-	80	120	kΩ	-
Pull-down 抵抗(100kΩ)	-	80	120	kΩ	-

[a]オーバーシュートとアンダーシュートは 0.6V 以下でかつ 4ns を超えないようにしてください。

・ アナログ入力インターフェース(CON21)の入力仕様 [3]

表 3.11 アナログ入力インターフェース(CON21)の入力仕様

項目		内容
入力方式		シングルエンド入力
入力レンジ	電圧	0.1～5.1V
	電流	0.5mA～20.4mA
入力インピーダンス	電圧入力時	1MΩ
	電流入力時	249Ω
ポート数		4ch
実効分解能		12bit
LSB		0.1875mV
非直線性誤差		最大 0.1875mV
ゲイン誤差		最大 8.6mV
オフセット誤差	電圧入力時	最大 5.1mV
	電流入力時	最大 10.2mV
絶縁仕様		バス絶縁
絶縁耐圧		0.5kV

・ 入出力インターフェース(CON22)の入出力仕様 [3]

表 3.12 入出力インターフェース(CON6)の入出力仕様

項目		内容
接点入力	入力インピーダンス	4.7 kΩ
	定格電圧	DC 8~26.4 V
	入力 ON 電流	1.0 mA 以上
	入力 OFF 電流	0.2 mA 以下
接点出力	定格電圧	最大 48 V
	定格電流	最大 500 mA
	応答時間	2ms 以内
	出力形式	無極性
絶縁耐圧		2kV

3.4.5.4. 電源回路の構成

Armadillo-IoT ゲートウェイ A6E の電源回路の構成は「図 3.55. 電源回路の構成」のとおりです。

[2]+Di8+Ai4 タイプの場合、+Di8+Ai4 拡張基板が接続されています。
[3]+Di8+Ai4 タイプのみ

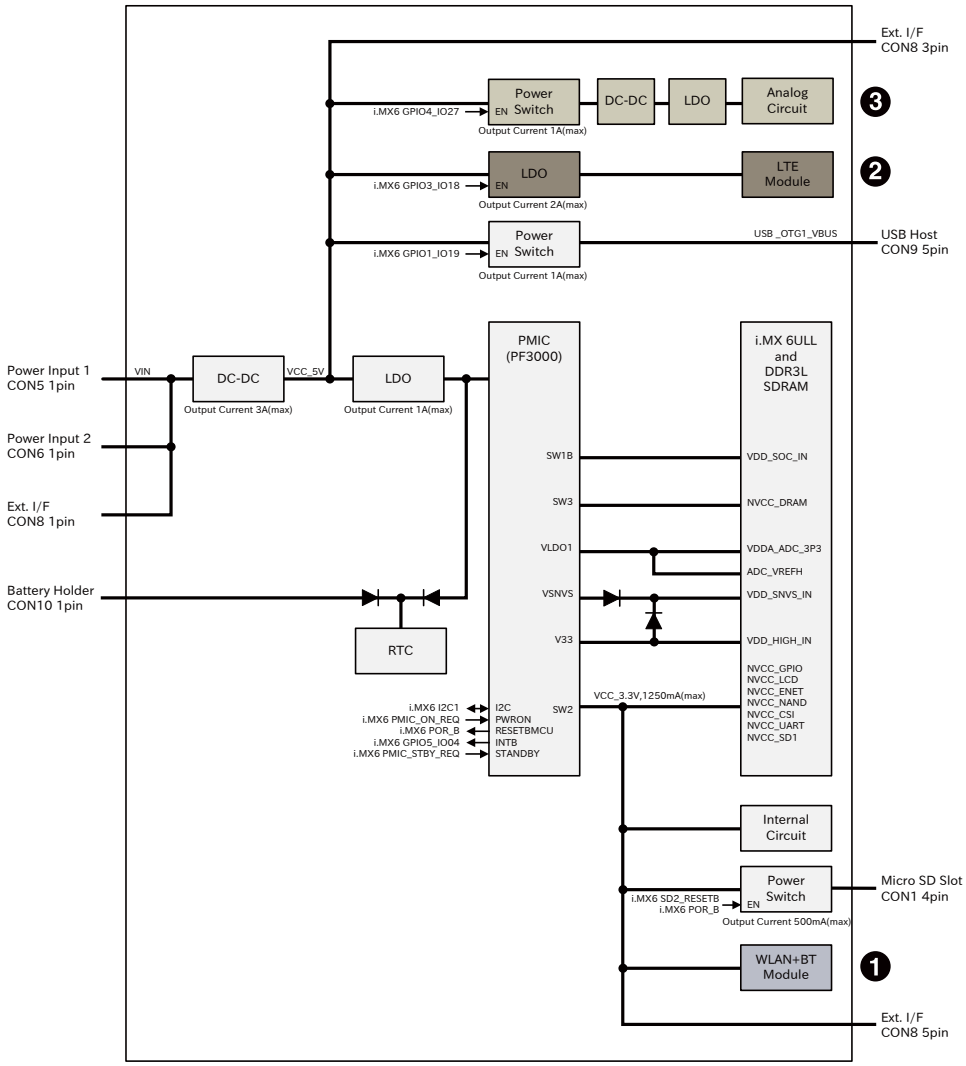


図 3.55 電源回路の構成

❶ Cat.1 bis+WLAN モデル、WLAN モデルの場合、WLAN モジュールが搭載されます。

❷ Cat.1 bis モデルの場合、LTE モジュールと直前の LDO が搭載されます。

❸ +Di8+Ai4 タイプの場合、CON8 に+Di8+Ai4 拡張基板が接続され、アナログ回路および直前のパワースイッチ、DC/DC コンバータ、LDO が搭載されます。

電源シーケンスは次のとおりです。

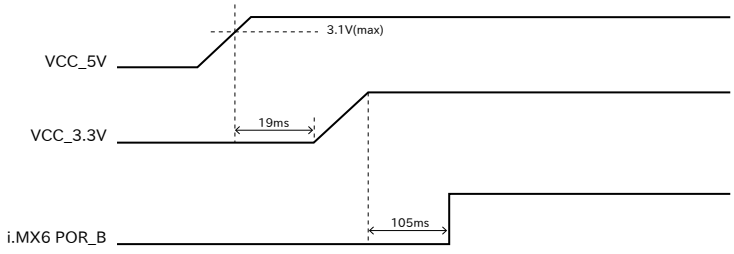


図 3.56 電源シーケンス

入力電圧(VIN)を電源 IC で各電圧に変換し、内部回路および各インターフェースに供給しています。各インターフェースやスイッチング・レギュレータの最大出力電流値を超えないように、外部機器の接続、供給電源の設計を行なってください。

3.4.6. 各動作モードにおける電源供給状況

各動作モードにおける電源供給状況は以下の通りです。

表 3.13 各動作モードにおける電源供給状況

動作モード	VCC_5V	VCC_3.3V
電源未接続	OFF	OFF
Shutdown	ON	OFF
Sleep(SMS)	ON	ON
Sleep	ON	ON
Active	ON	ON

3.4.7. reboot コマンドによる再起動時の電源供給について

reboot コマンドで再起動した場合の各電源供給状況は以下の通りです。

表 3.14 reboot コマンドで再起動した場合の各電源供給状況

電源	供給状況
VCC_5V	供給を保持します
VCC_3.3V	供給を保持します

3.4.8. 基板形状図

以下の 2 種類のタイプの基板形状を示します。

- スタンダードタイプ
- +Di8+Ai4 タイプ



DXF 形式の形状図を「アットマークテクノ Armadillo サイト」から「購入者向けの限定公開データ」としてダウンロード可能です。



基板改版や部品変更により、基板上の部品位置、高さは変更になることがあります。ケースや拡張基板を設計する場合、ある程度の余裕をもった寸法での設計をお願いいたします。



型番により部品の搭載/非搭載が異なります。詳細は納入仕様書をご確認ください。

本製品シリーズの納入仕様書は、アットマークテクノ Armadillo サイト (https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/spec)からご覧いただけます。(要ログイン)

3.4.8.1. 基板形状図(スタンダードタイプ)

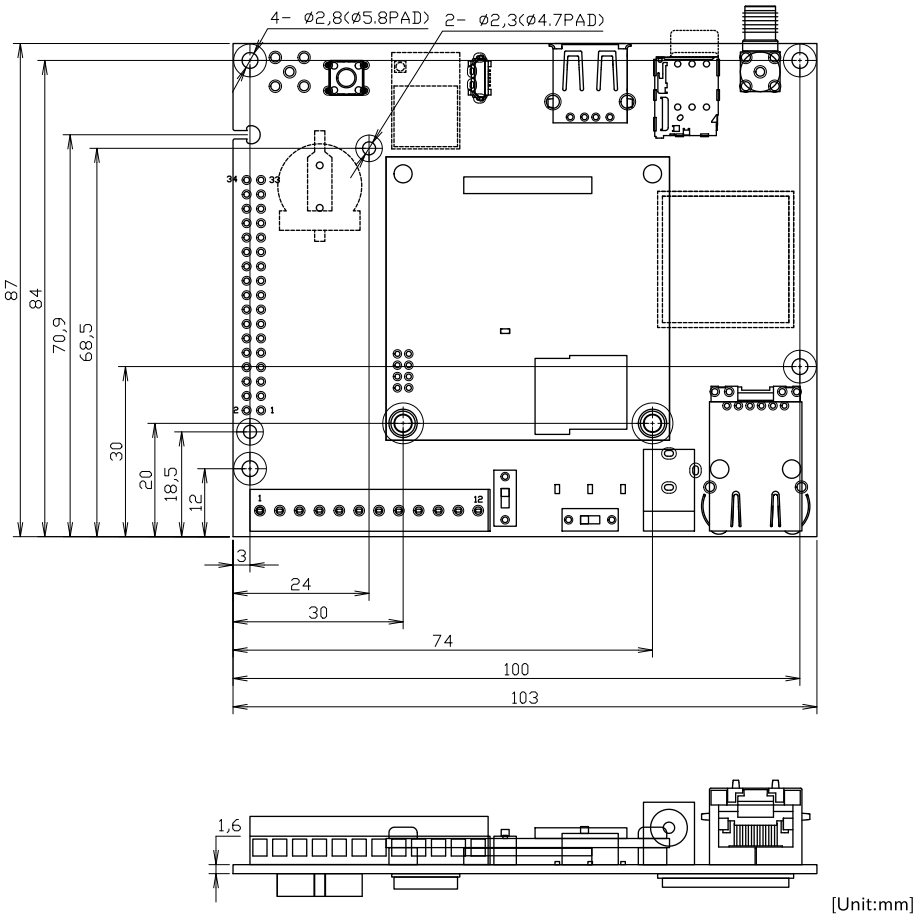


図 3.57 基板形状および固定穴寸法(スタンダードタイプ)

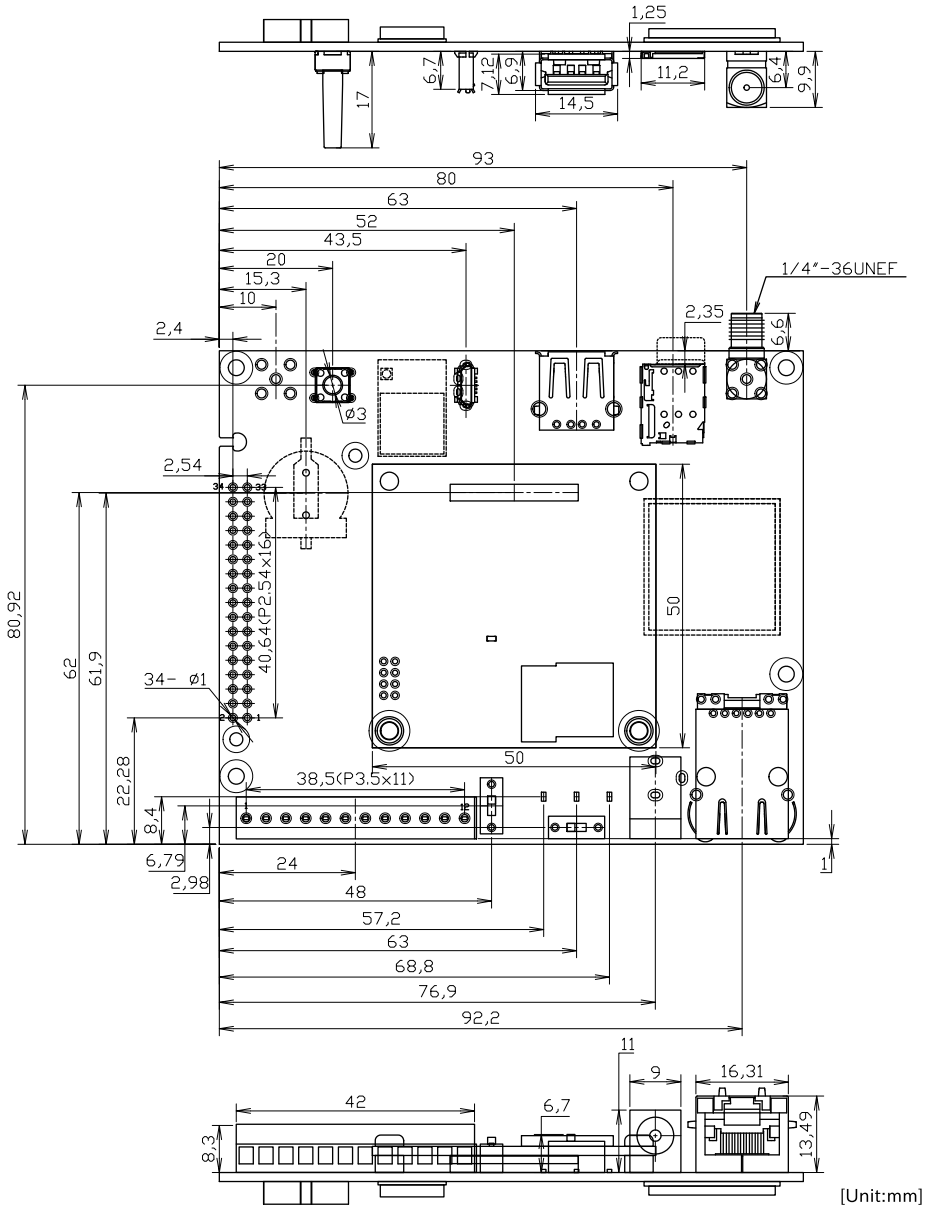


図 3.58 コネクタ、スイッチ、LED 位置(スタンダードタイプ)

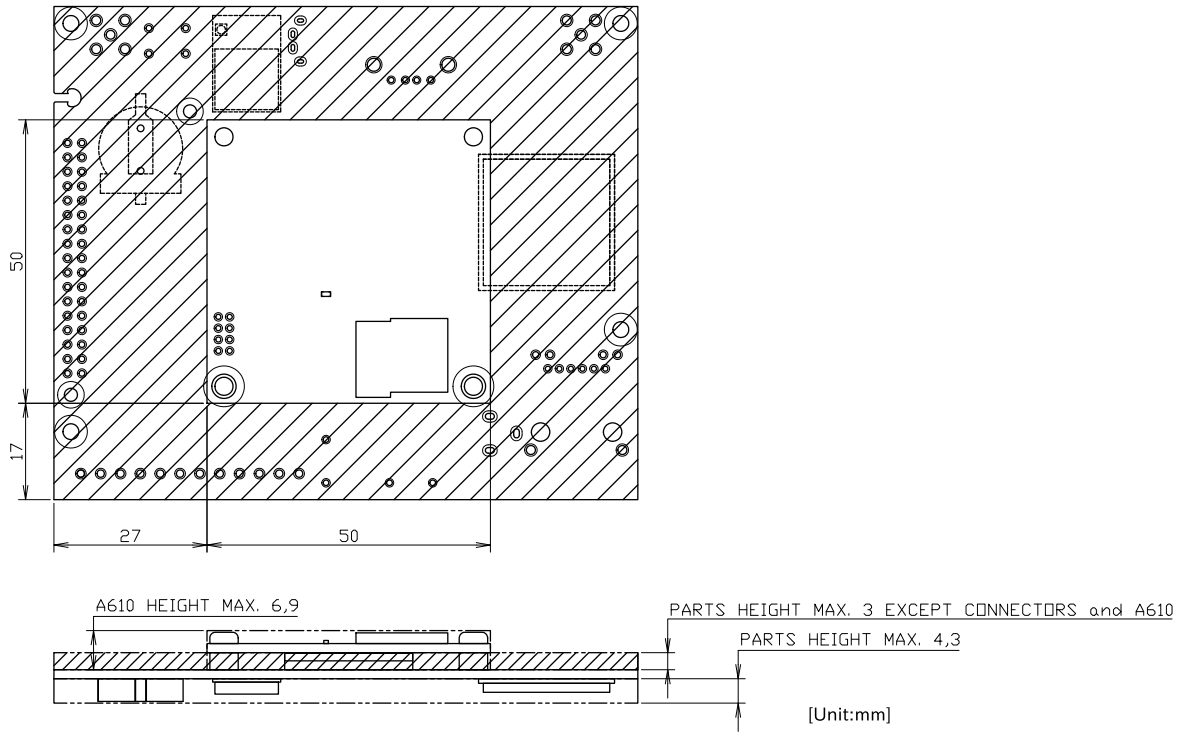


図 3.59 部品高さ(スタンダードタイプ)

3.4.8.2. 基板形状図(+Di8+Ai4 タイプ)

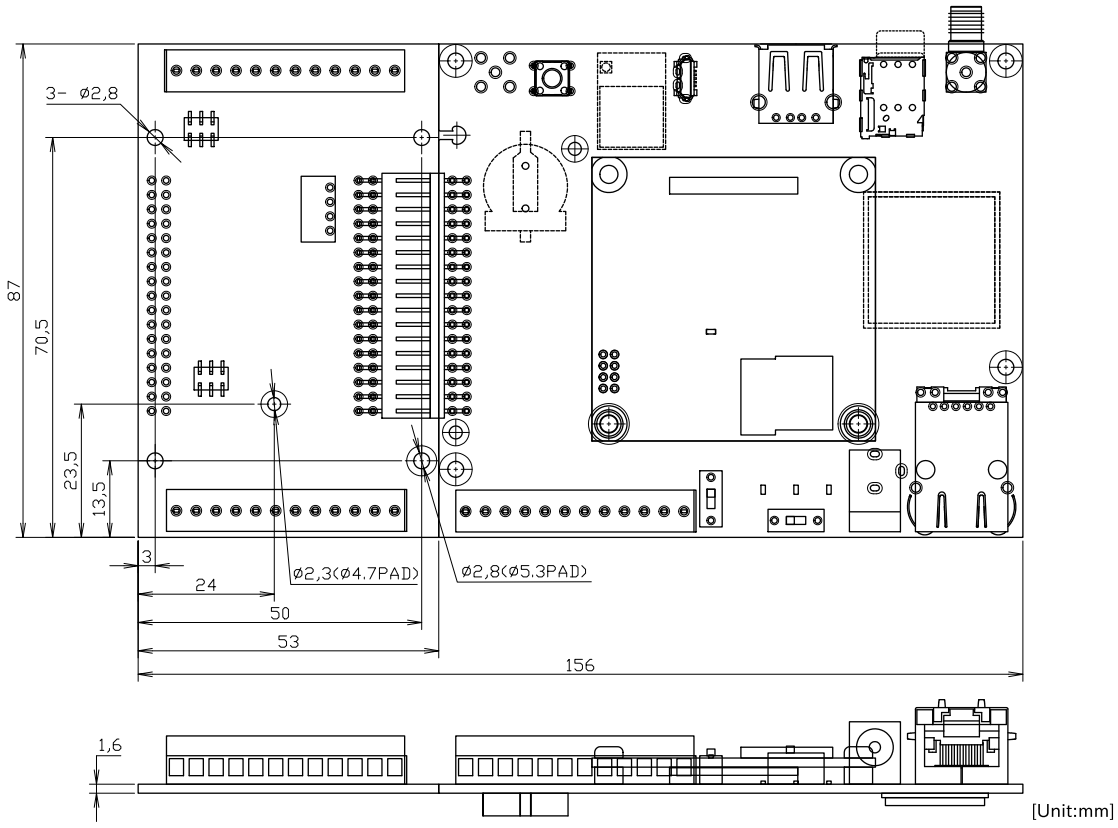


図 3.60 基板形状および固定穴寸法(+Di8+Ai4 タイプ)

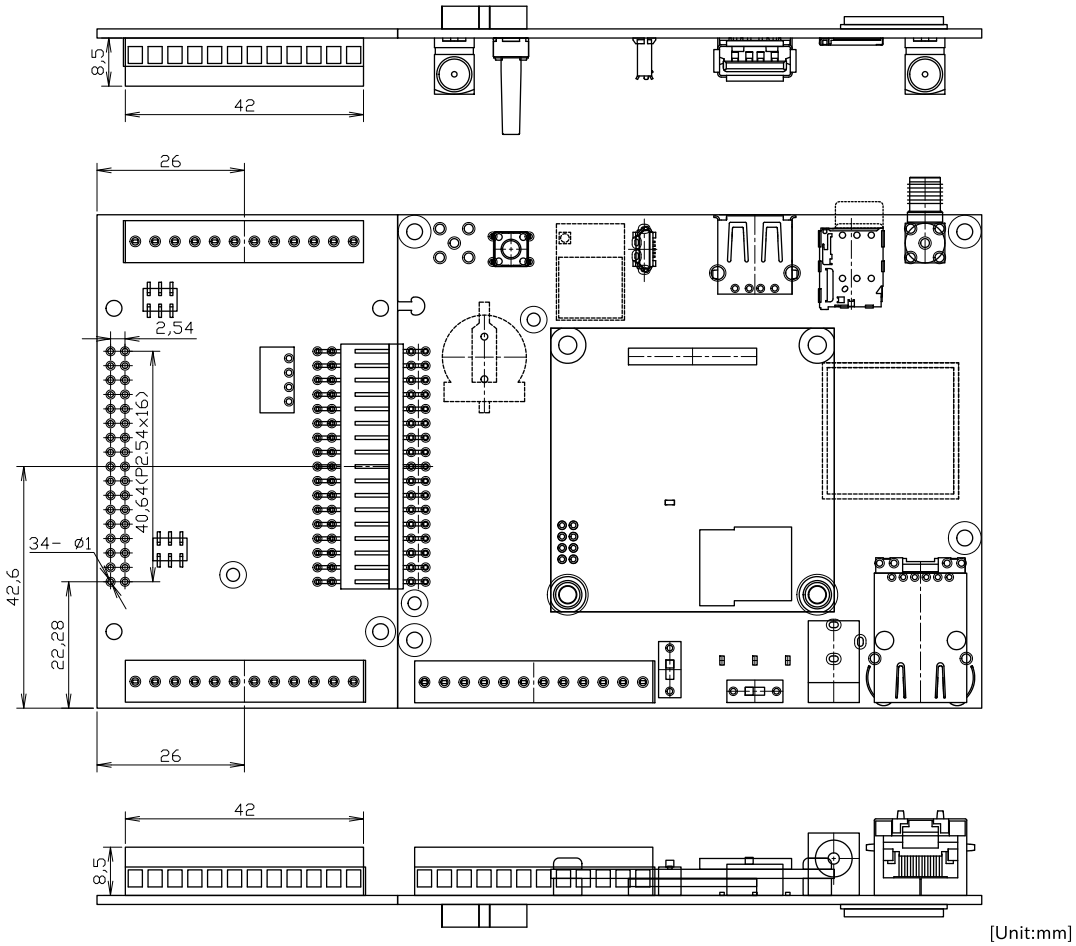


図 3.61 コネクタ、スイッチ、LED 位置(+Di8+Ai4 タイプ)

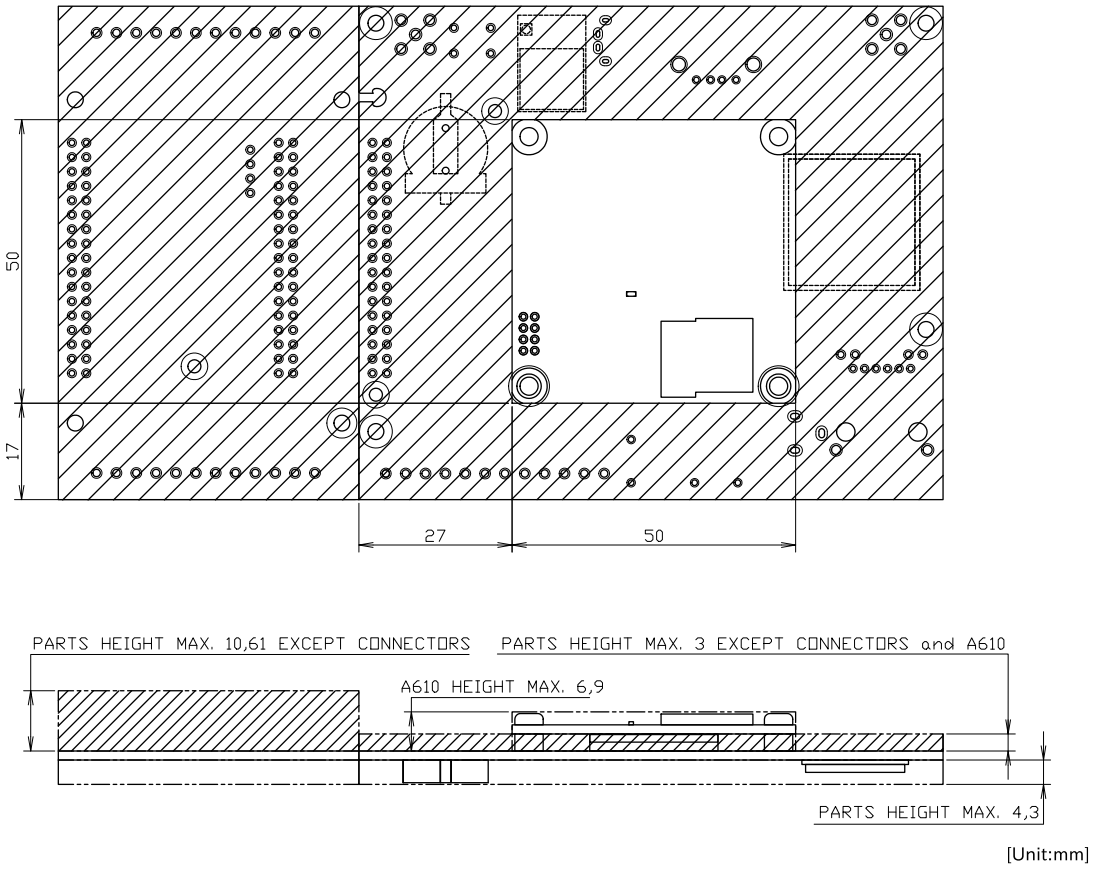


図 3.62 部品高さ(+Di8+Ai4 タイプ)

3.4.9. ケース形状図

以下の 2 種類のタイプのケース形状を示します。

- スタンダードタイプ
- +Di8+Ai4 タイプ



DXF 形式の形状図を「アットマークテクノ Armadillo サイト」から「購入者向けの限定公開データ」としてダウンロード可能です。

3.4.9.1. ケース形状図(スタンダードタイプ)

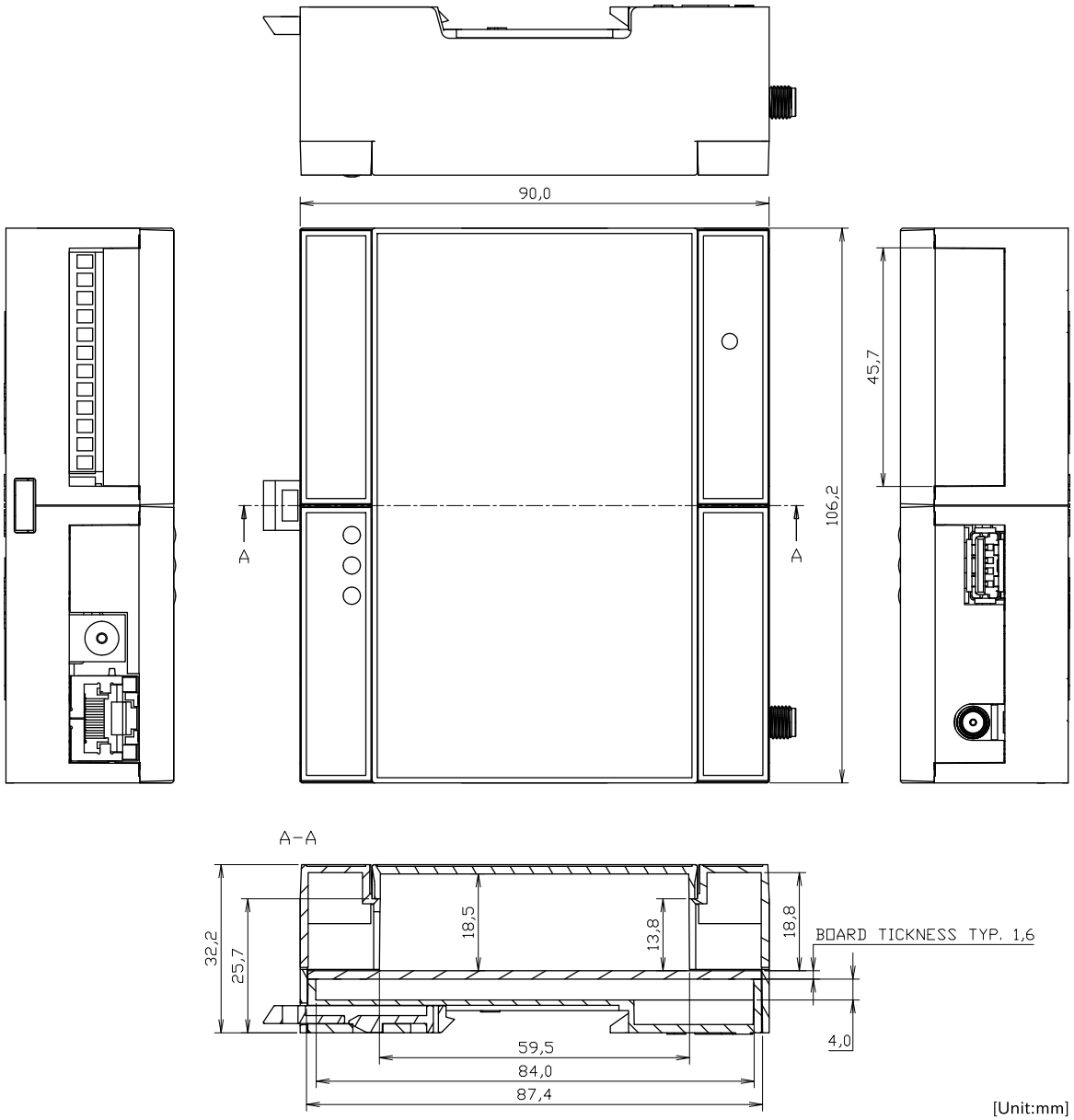


図 3.63 ケース形状図(スタンダードタイプ)

3.4.9.2. ケース形状図(+Di8+Ai4 タイプ)

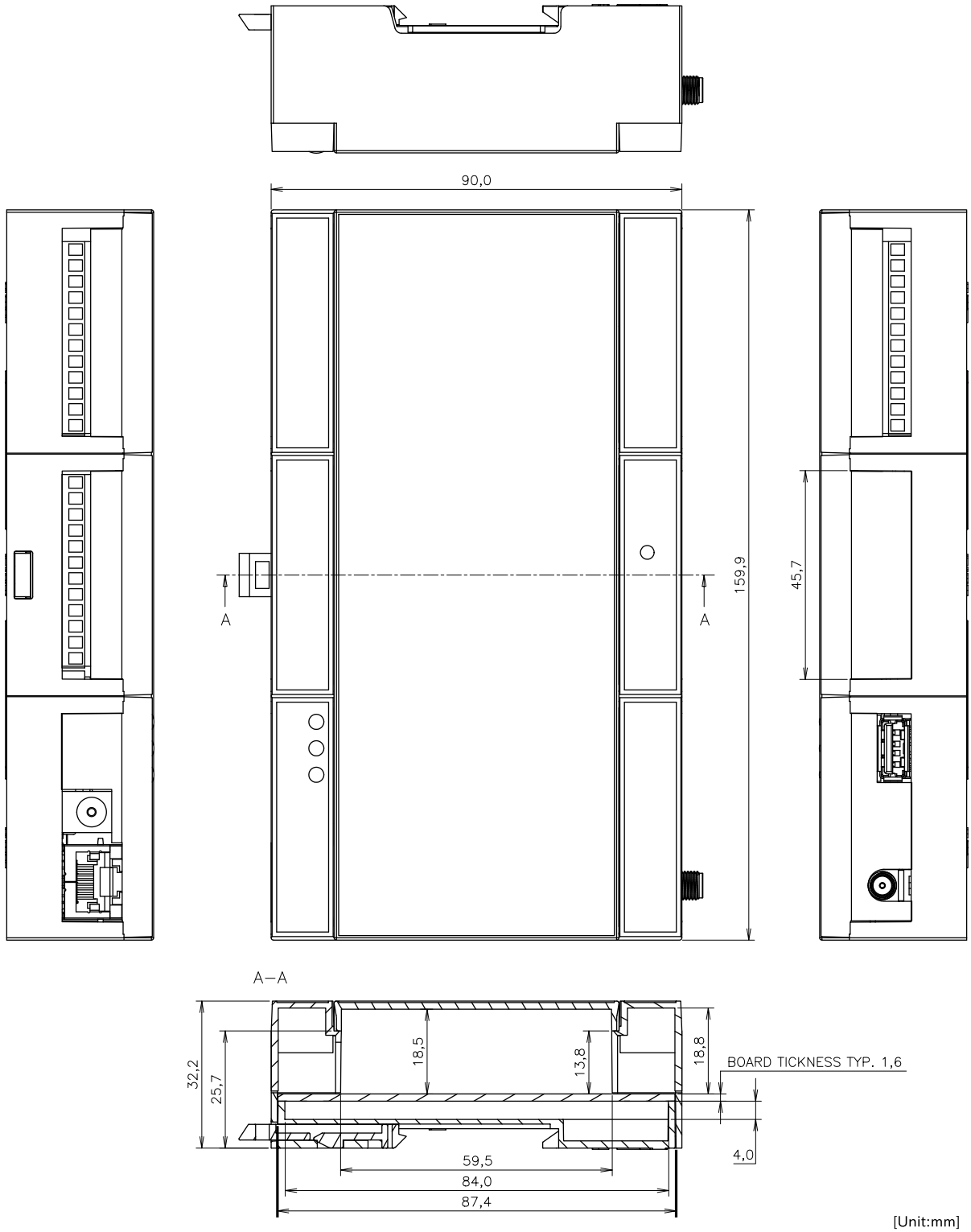


図 3.64 ケース形状図(+Di8+Ai4 タイプ)

3.4.10. アンテナ形状図

WLAN、LTE 用のアンテナの形状を示します。

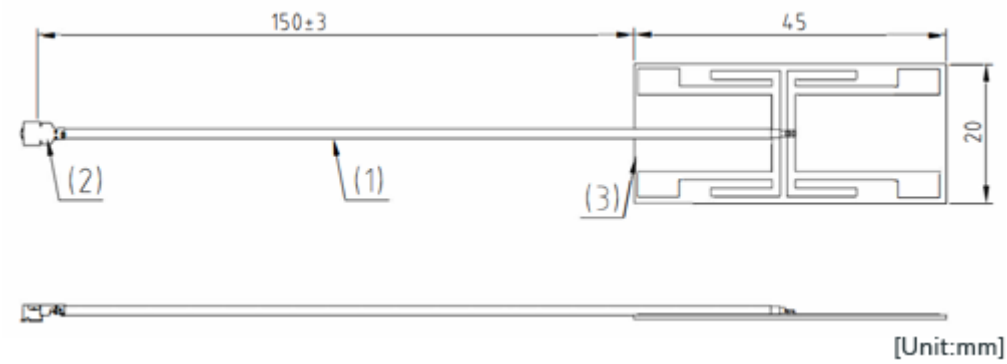


図 3.65 WLAN 基板アンテナ形状図(Cat.1 bis+WLAN モデル)

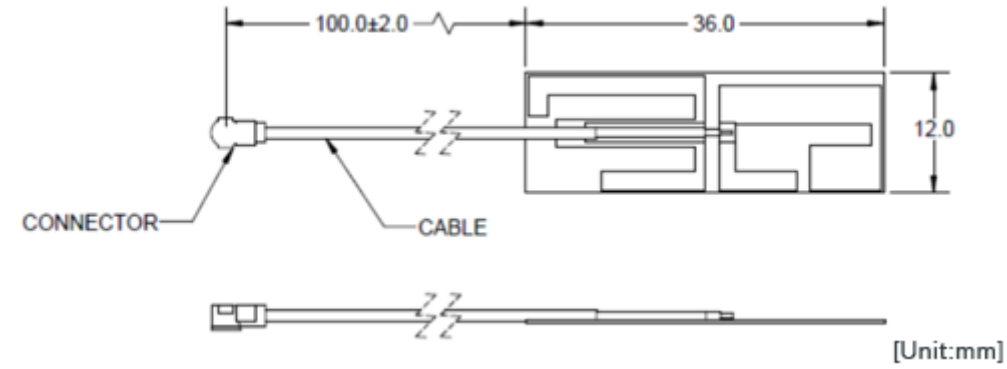


図 3.66 WLAN 基板アンテナ形状図(WLAN モデル)

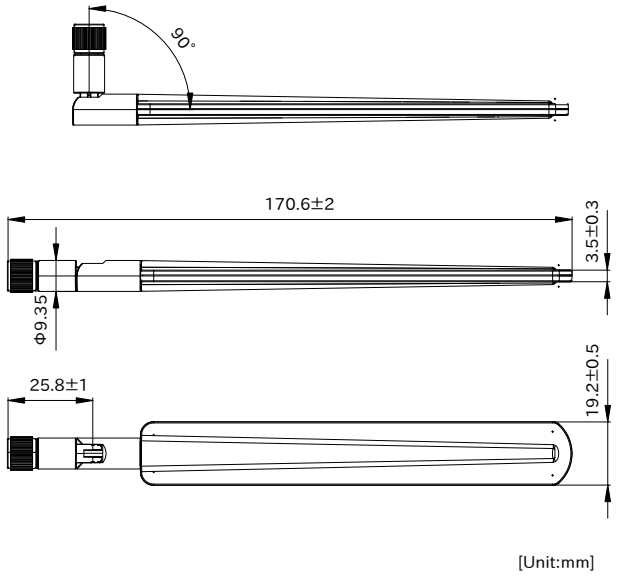


図 3.67 LTE アンテナ形状図

3.5. ケースの組み立てと分解方法

本製品はねじを使用しないスナップフィット方式を採用しており、容易に組み立てと分解が可能です。分解する際には手のけがやパーツの破損を防止するためマイナスドライバーなどの工具を使用してください。



組み立てや分解を行う際は以下の動画を参考にしてください。 (https://www.youtube.com/watch?v=MIADkKwxZmU)

3.5.1. ケースのパーツ構成

3.5.1.1. スタンダードタイプ

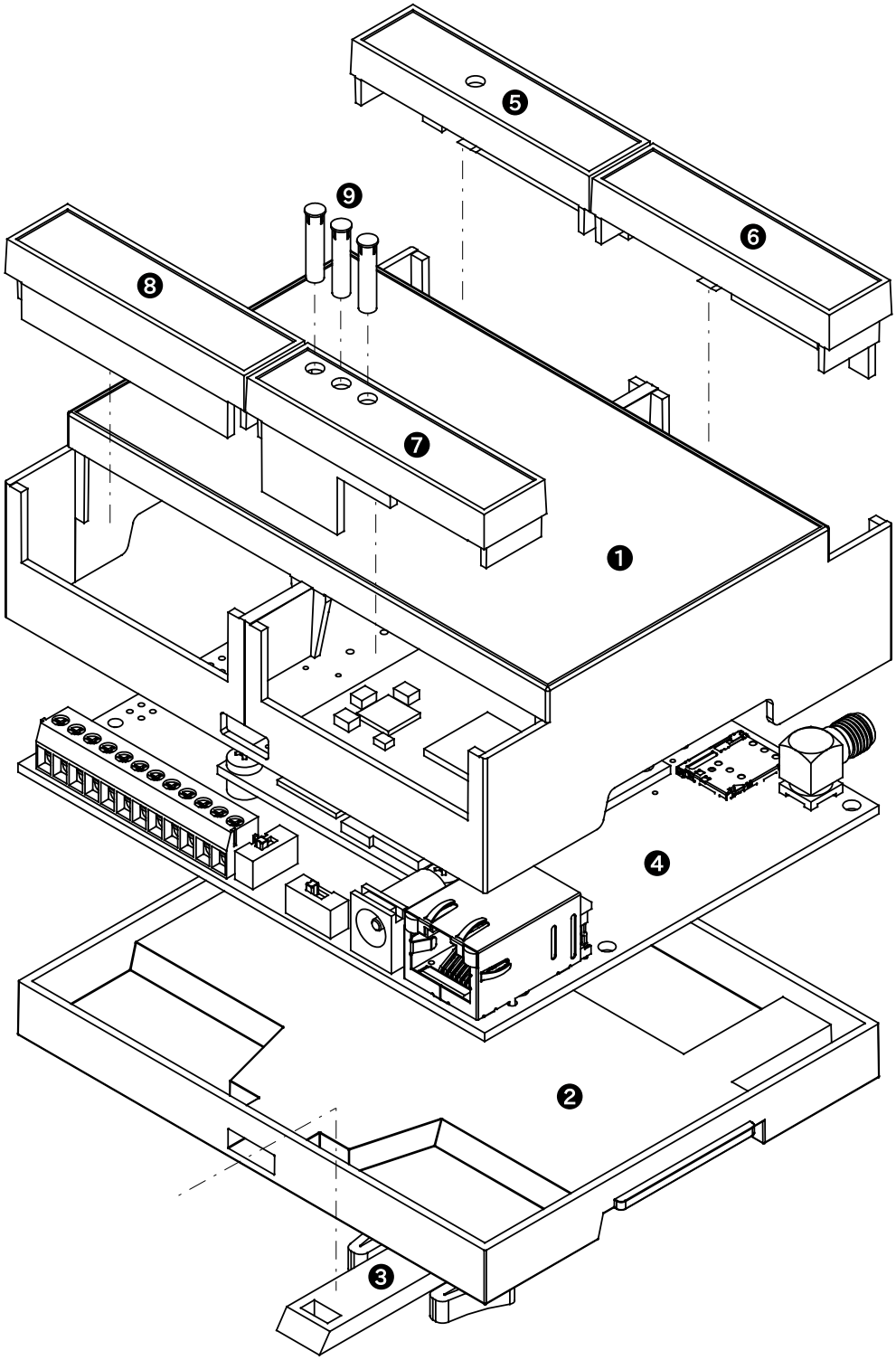


図 3.68 ケース展開図 (スタンダートタイプ)

表 3.15 ケース展開図パーツ一覧 (スタンダートタイプ)

番号	名称	説明
1	ケーストップ	ケース上側のパーツです。 ケースボトムとは 4 か所のツメで固定されます。 ケースを分解する際は、マイナスドライバーを使用してツメを破損させないよう慎重に取り外してください。
2	ケースボトム	ケース下側のパーツです。
3	フック	ケースを DIN レールに固定するためのパーツです。
4	基板	
5	カバーパーツ A	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
6	カバーパーツ B	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
7	カバーパーツ C	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
8	カバーパーツ D	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
9	LED ライトパイプ	カバーパーツ C に装着する LED のライトパイプです。強い衝撃を加えた場合、ライトパイプが外れる場合がありますので、「図 3.68. ケース展開図 (スタンダートタイプ)」を参考にカバーパーツ C の丸穴に差し込んでください。

フックは以下の図を参考に取り付けてください。

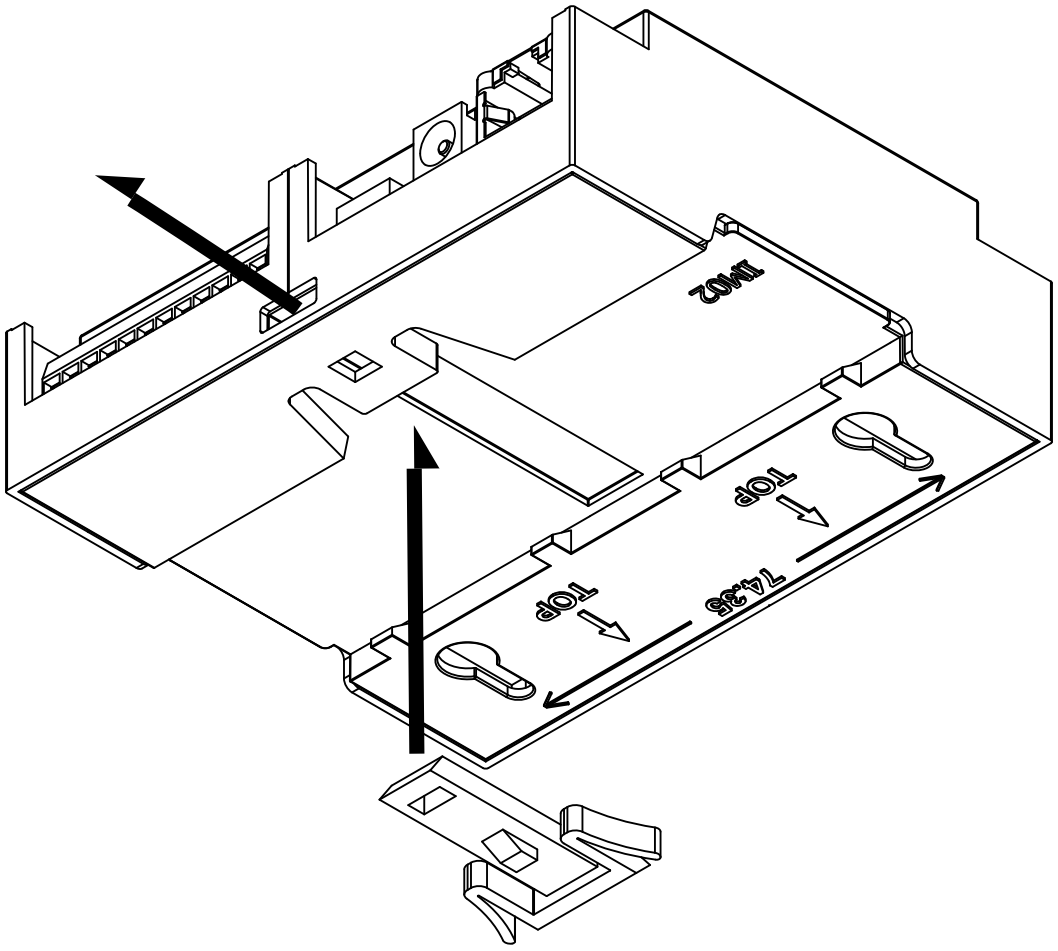


図 3.69 フック取り付け 1 (スタンダートタイプ)

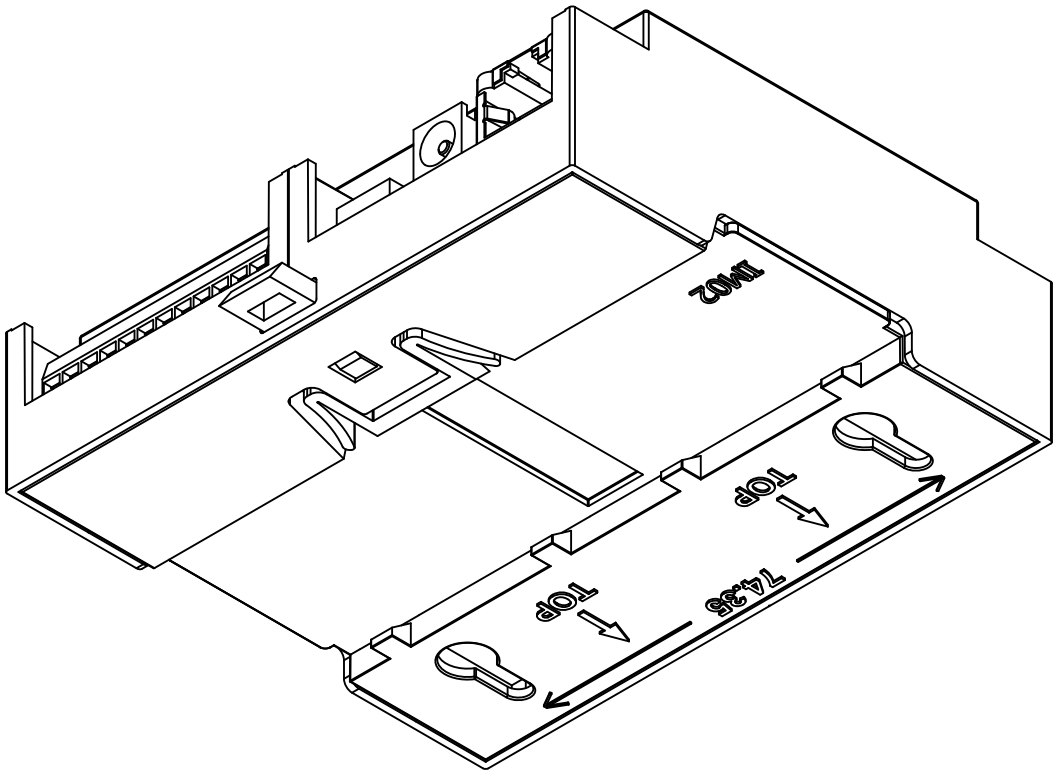


図 3.70 フック取り付け 2 (スタンダートタイプ)

3.5.1.2. +Di8+Ai4 タイプ

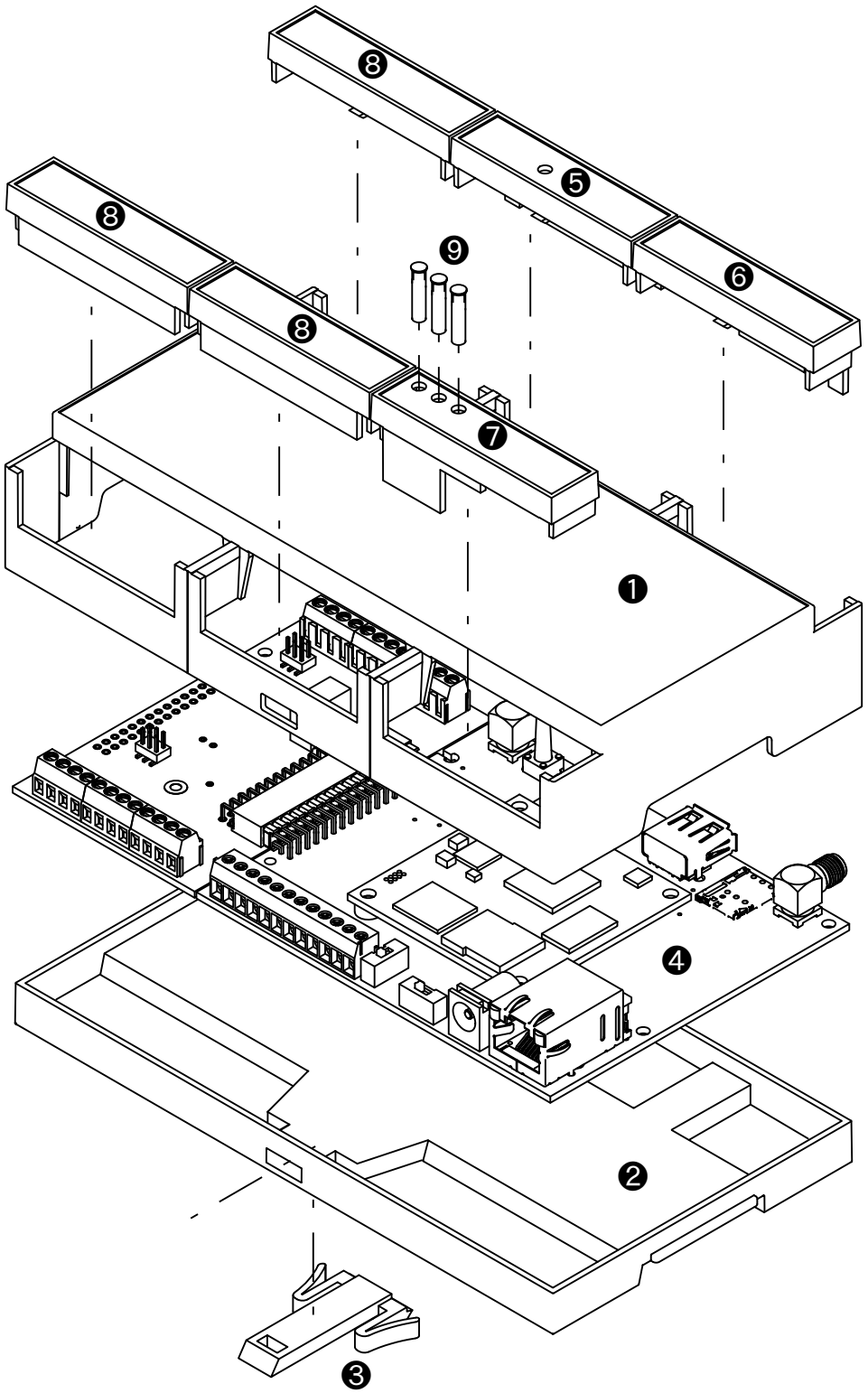


図 3.71 ケースモデル展開図 (+Di8+Ai4 タイプ)

表 3.16 ケースモデル展開図パーツ一覧 (+Di8+Ai4 タイプ)

番号	名称	説明
1	ケーストップ	ケース上側のパーツです。 ケースボトムとは 4 か所のツメで固定されます。ケースを分解する際は、マイナスドライバーを使用してツメを破損させないよう慎重に取り外してください。
2	ケースボトム	ケース下側のパーツです。
3	フック	ケースを DIN レールに固定するためのパーツです。
4	基板	
5	カバーパーツ A	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
6	カバーパーツ B	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
7	カバーパーツ C	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
8	カバーパーツ D	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
9	LED ライトパイプ	カバーパーツ C に装着する LED のライトパイプです。強い衝撃を加えた場合、ライトパイプが外れる場合がありますので、「図 3.68. ケース展開図 (スタンダートタイプ)」を参考にカバーパーツ C の丸穴に差し込んでください。

フックは以下の図を参考に取り付けてください。

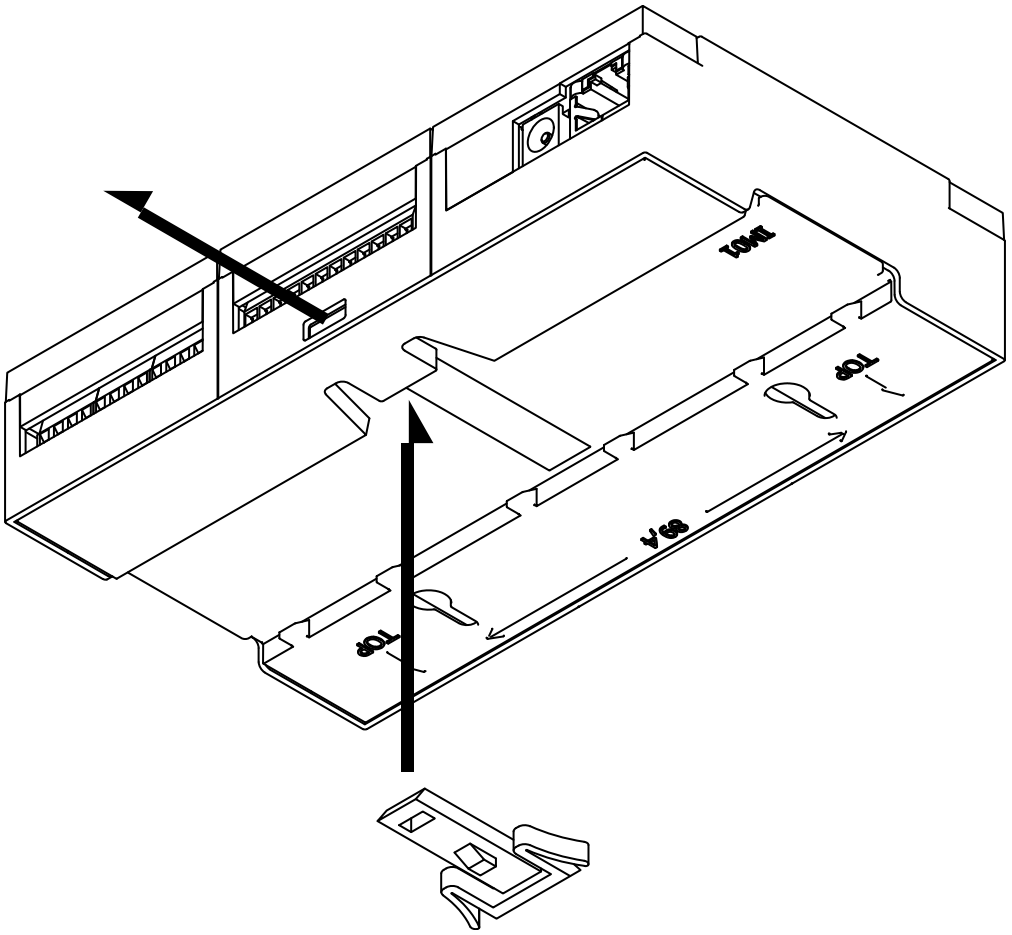


図 3.72 フック取り付け 1 (+Di8+Ai4 タイプ)

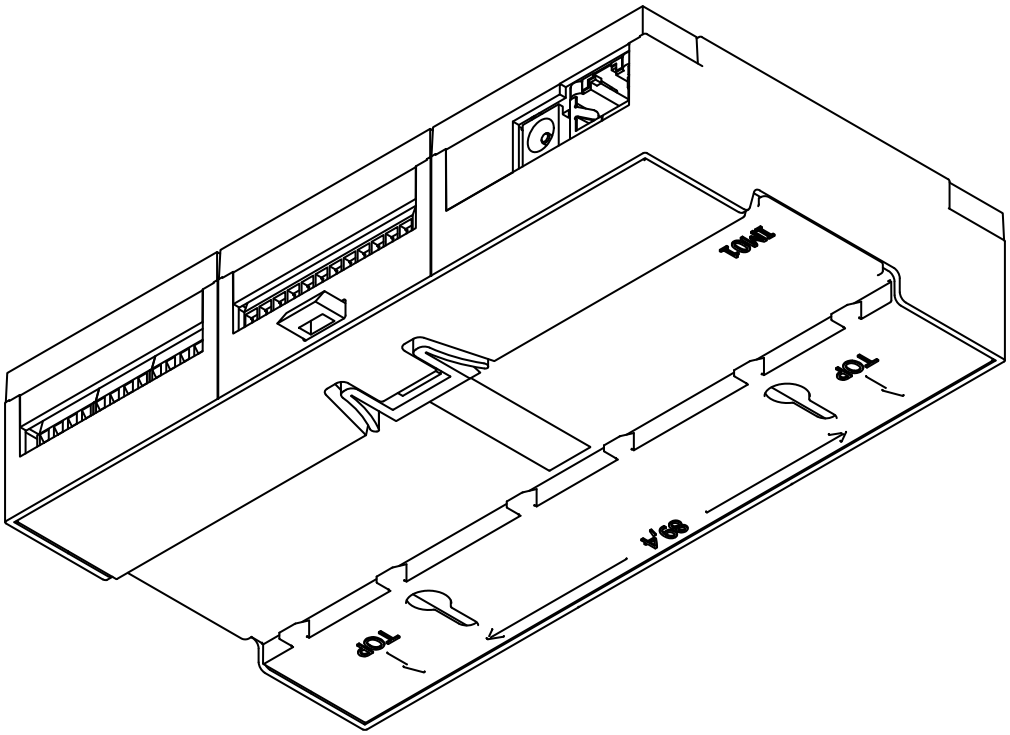


図 3.73 フック取り付け 2 (+Di8+Ai4 タイプ)

3.5.2. ケースの組み立て手順

> microSD カードの取り付けは、ケースの組み立て前に行う必要があります。取り付け手順については、「3.7.7.2. microSD カードの挿抜方法」を参照してください。

以下の手順に従い、ケースを組み立ててください。

1. 基板をケーストップに入れる
2. ケースボトムをケーストップにはめ込み、基板を固定する
3. フックをケースボトムにはめ込む
4. カバーパーツをケーストップにはめ込む

3.5.3. ケースの分解

> WLAN+BT コンボモジュールを搭載した製品では、ケーストップに貼り付けられている WLAN 基板アンテナのケーブルが、製品基板上のアンテナコネクタと接続されています。ケースを分解する際、無理な力をかけるとケーブルが破損する可能性があるため、慎重に作業してください。

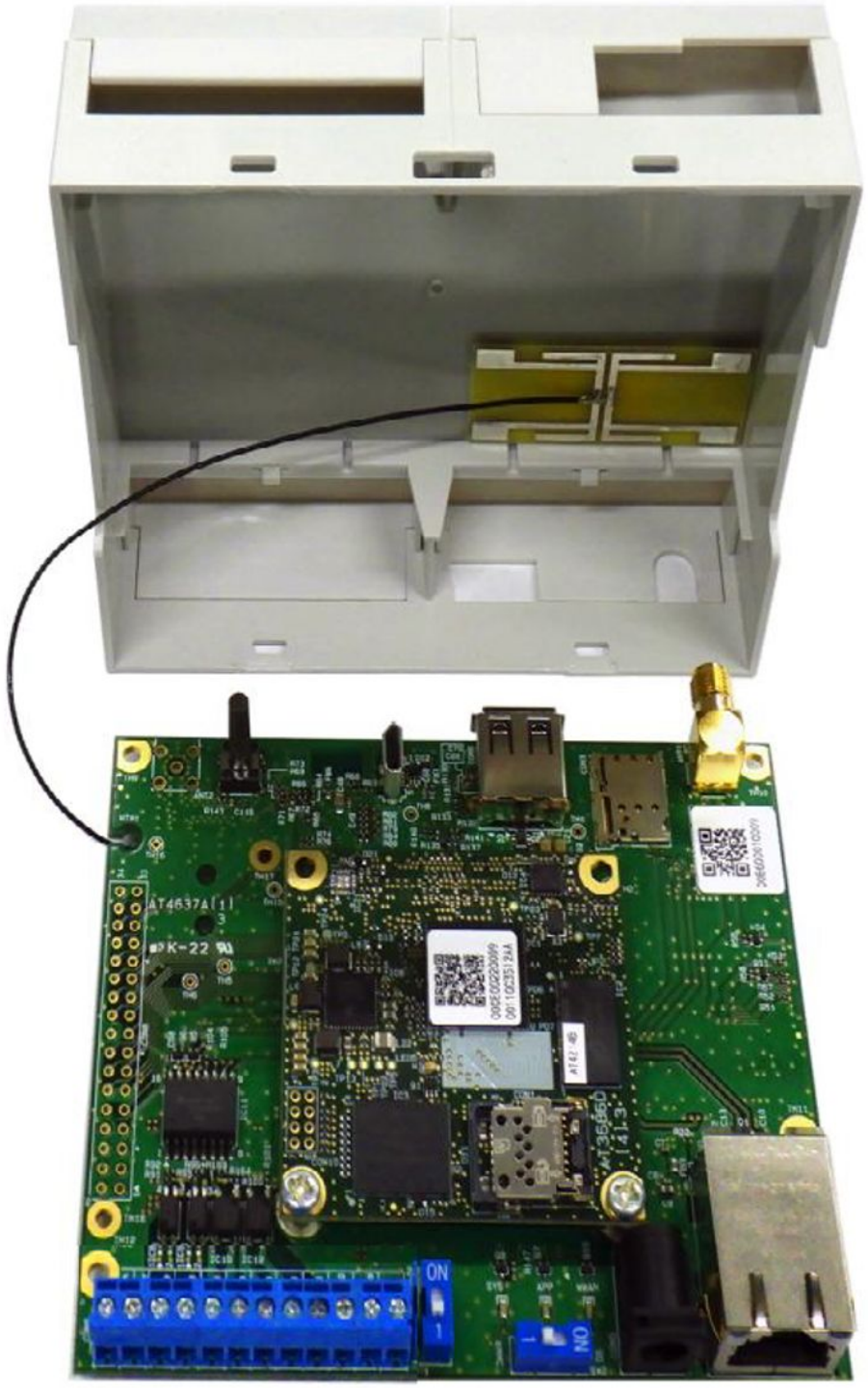


図 3.74 WLAN 基板アンテナの位置



ツメに強い力を加えますと破損する恐れがありますので、十分ご注意ください。

マイナスドライバーなどの工具を用意してください。以下の手順に従い、ケースを分解してください。

1. フックをケースボトムから取り外す
2. ケースボトムを取り外す
3. 基板を取り外す
4. カバーパーツを取り外す

フックはツメで固定されています。「図 3.75. フックのツメ」を参考にツメを押しながらフックを引き出してください。

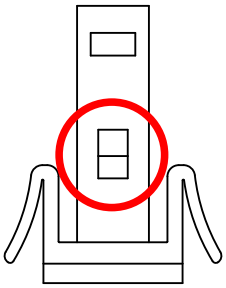


図 3.75 フックのツメ

ケースボトムはツメ 4 か所で固定されています。スタンダードタイプの場合は「図 3.76. ケースボトムのツメ(スタンダードタイプ)」、+Di8+Ai4 タイプの場合は「図 3.77. ケースボトムのツメ(+Di8+Ai4 タイプ)」の赤丸の箇所にあるケースの隙間にマイナスドライバーを差し込み、順に外してください。

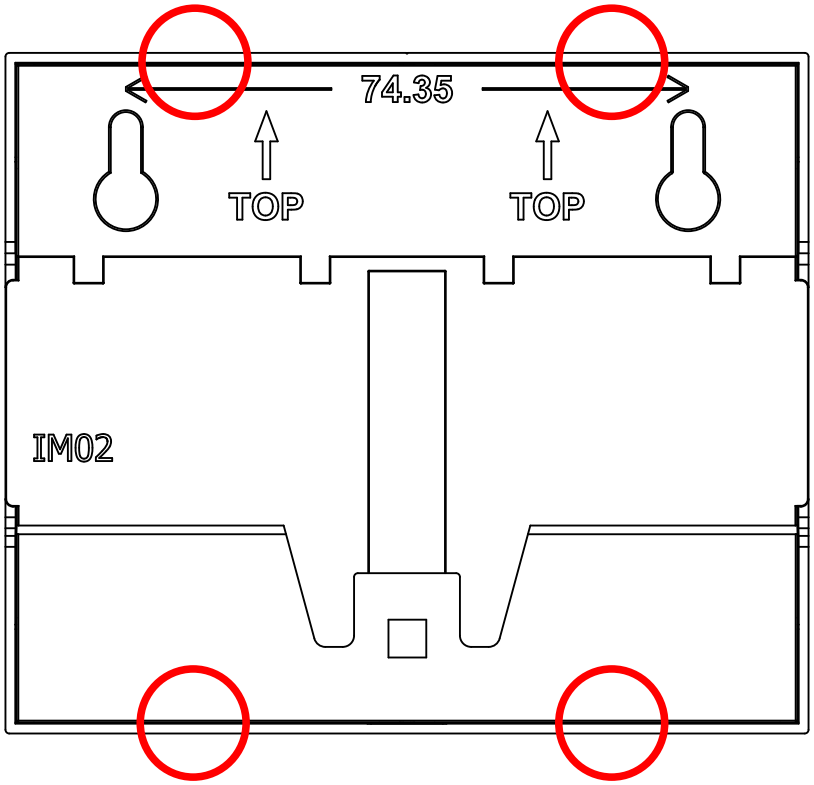


図 3.76 ケースボトムのツメ(スタンダードタイプ)

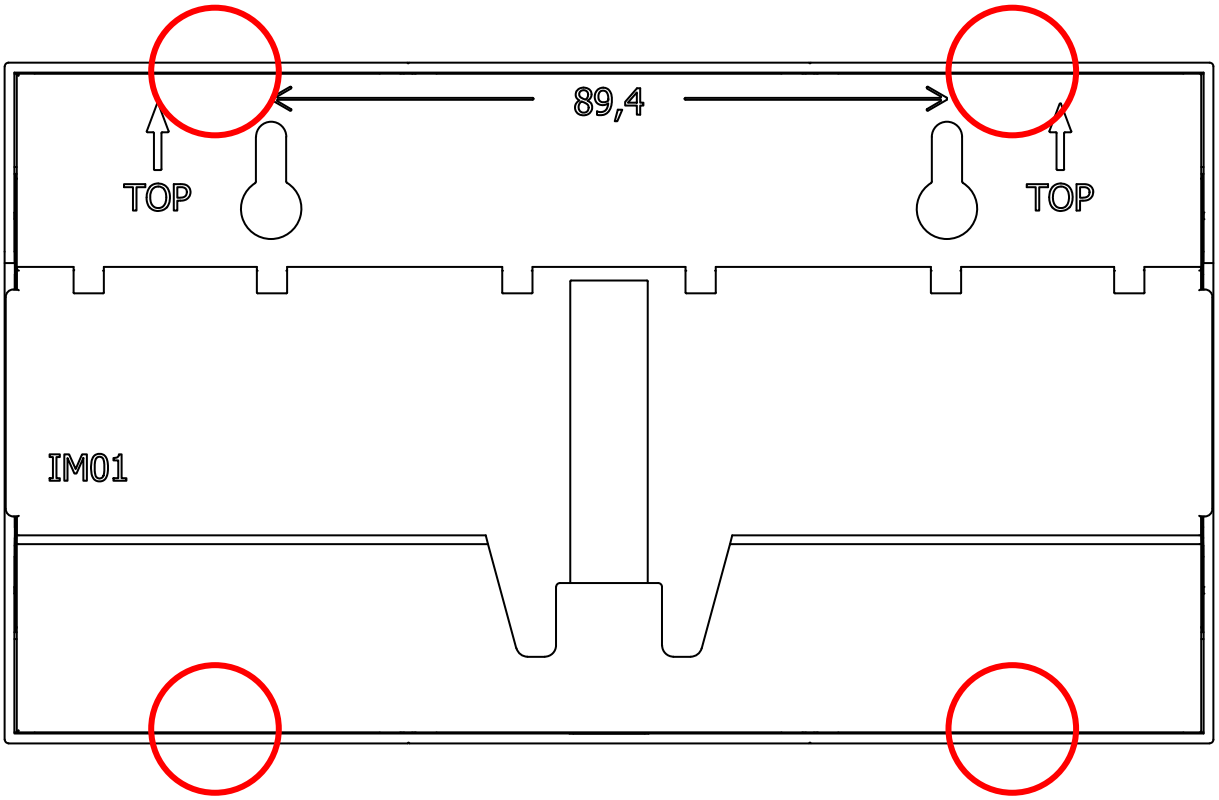


図 3.77 ケースボトムのツメ(+Di8+Ai4 タイプ)

アンテナコネクタを実装しているモデルでは、アンテナコネクタがケース開口部より飛び出しているため、反対側の LAN コネクタ側から先にケーストップから出すようにしてください。基板を取り外す際、LAN コネクタの突起部がケーストップに当たらないよう、ケースを広げながら基板を取り外すようにしてください。

カバーはツメ 1 か所でケーストップに固定されています。「図 3.78. カバーのツメ」を参考にマイナスドライバーをケースの隙間に差し込み外してください。

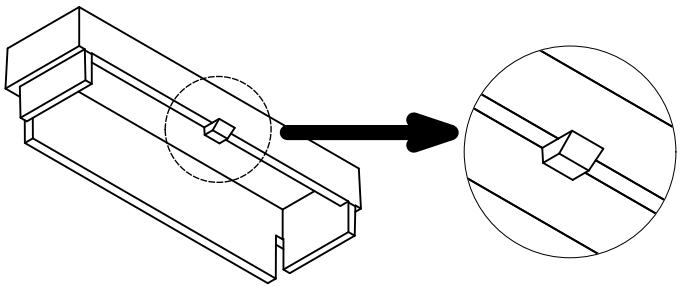


図 3.78 カバーのツメ

3.6. WLAN アンテナの取り付けと取り外し

WLAN+BT コンボモジュール用のアンテナを取り付ける方法および取り外す方法を、WLAN+BT コンボモジュールを搭載しているモデルに付属している WLAN 基板アンテナを例に説明します。

> WLAN 用アンテナの取り付けと取り外しは、しっかりと静電気対策した上で実施してください。静電気により製品が故障する可能性があります。

Cat.1 bis+WLAN モデルの場合、アンテナコネクタは「図 3.79. WLAN+BT コンボモジュールのアンテナコネクタの位置 (Cat.1 bis+WLAN モデル)」の位置にあります。

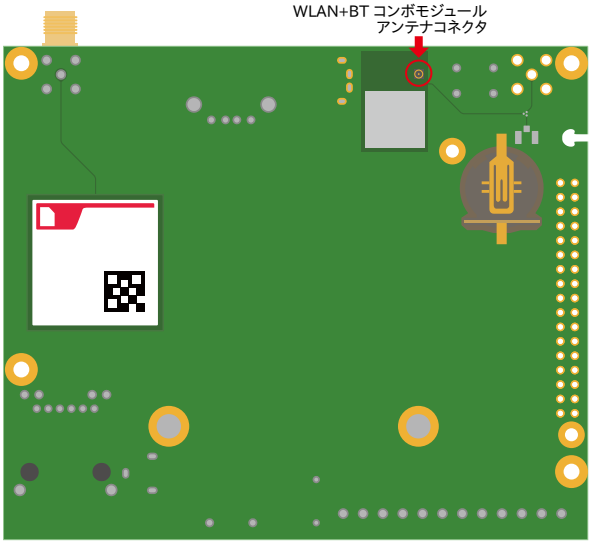


図 3.79 WLAN+BT コンボモジュールのアンテナコネクタの位置 (Cat.1 bis+WLAN モデル)

WLAN/LAN モデルの場合、アンテナコネクタは「図 3.80. WLAN+BT コンボモジュールのアンテナコネクタの位置 (WLAN モデル)」の位置にあります。

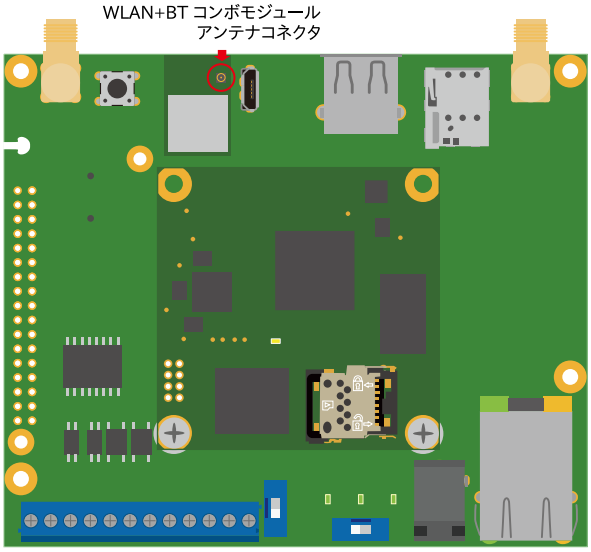


図 3.80 WLAN+BT コンボモジュールのアンテナコネクタの位置 (WLAN モデル)

WLAN+BT コンボモジュールを搭載しているモデルに付属している WLAN 基板アンテナは、MHF4L コネクタを採用しています。

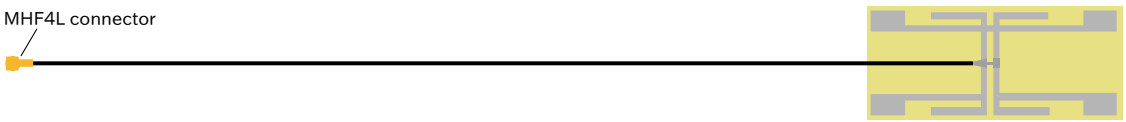


図 3.81 WLAN 基板アンテナ(Cat.1 bis+WLAN モデル)



図 3.82 WLAN 基板アンテナ(WLAN モデル)



WLAN+BT コンボモジュールのアンテナコネクタは非常に小さく破損しやすい部品ですので、取り扱いには十分ご注意ください。

3.6.1. WLAN 基板アンテナの取り付け

WLAN 基板アンテナは MHF4L コネクタ用の挿抜治具(90609-0001/IPEX)を使用して取り付けるか、手で直接取り付けします。



アンテナコネクタ実装面の反対側にスイッチ等の背の高い部品が実装されています。コネクタを嵌合する際に、背の高い部品を破損しないように注意してください。

3.6.1.1. 挿抜治具による取り付け

WLAN 基板アンテナのコネクタのストッパーに当たるまで挿抜治具をスライドさせ、コネクタ全体を抱えるようにします。WLAN 基板アンテナのコネクタが基板に対し平行になっていることを確認し、垂直に挿抜治具を押してコネクタを嵌合します。

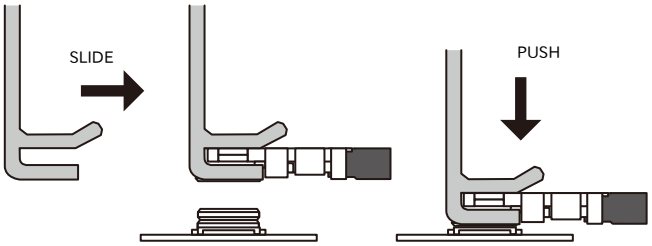


図 3.83 WLAN 基板アンテナの挿抜治具による取り付け



必ず WLAN 基板アンテナのコネクタを基板に対して平行にしてから嵌合してください。曲がったまま嵌合すると、コネクタ破損の原因となります。

3.6.1.2. 手で直接取り付け

WLAN 基板アンテナのケーブルを持ち、WLAN+BT コンボモジュールのアンテナコネクタに WLAN 基板アンテナのコネクタをセットします。セットしたら前後に軽く動かし、動かないことを確認します。

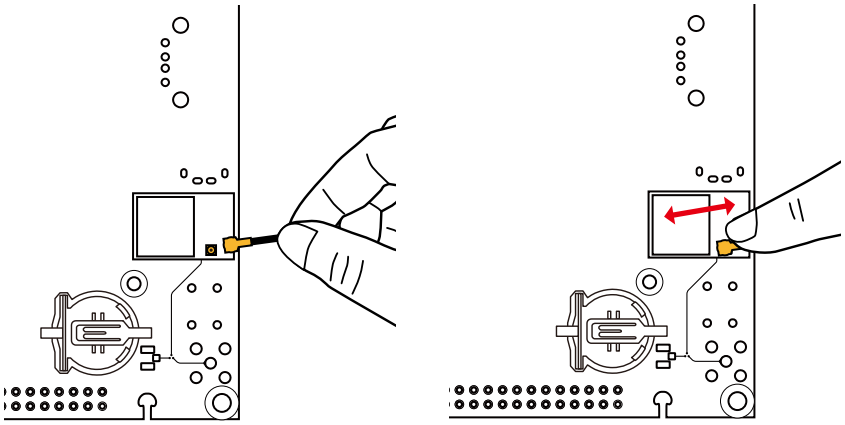


図 3.84 WLAN 基板アンテナの手による取り付け前の準備

WLAN 基板アンテナのケーブル側から、コネクタの中心を真上からカチっという音がするまで押します。

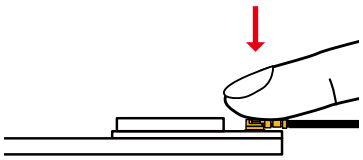


図 3.85 WLAN 基板アンテナの手による取り付け

3.6.2. WLAN 基板アンテナの取り外し

WLAN 基板アンテナのコネクタ首部へのストレスを避けるため、挿抜治具(90609-0001/IPEX)を使用して取り外します。WLAN 基板アンテナのコネクタのストッパーに当たるまで挿抜治具をスライドさせ、コネクタ全体を抱えるようにします。基板と垂直に挿抜治具を引き上げてコネクタを取り外します。

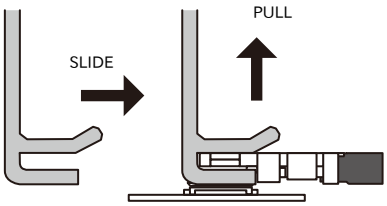


図 3.86 WLAN 基板アンテナの挿抜治具による取り外し



挿抜治具は必ず基板と垂直に引き上げてください。ひねったり、ななめに引き上げたりした場合、コネクタ破損の原因となります。

3.6.3. WLAN 基板アンテナのケースへの取り付け

WLAN 基板アンテナはケーストップに貼り付けます。ケーストップのフックパーツ挿し込み口の位置を確認し、両面テープで貼り付けます。

Cat.1 bis+WLAN モデルの場合、アンテナケーブルを基板の切り欠きに引っかけ、ケーストップを閉じる際に基板とケースの間に挟まれないようにします。

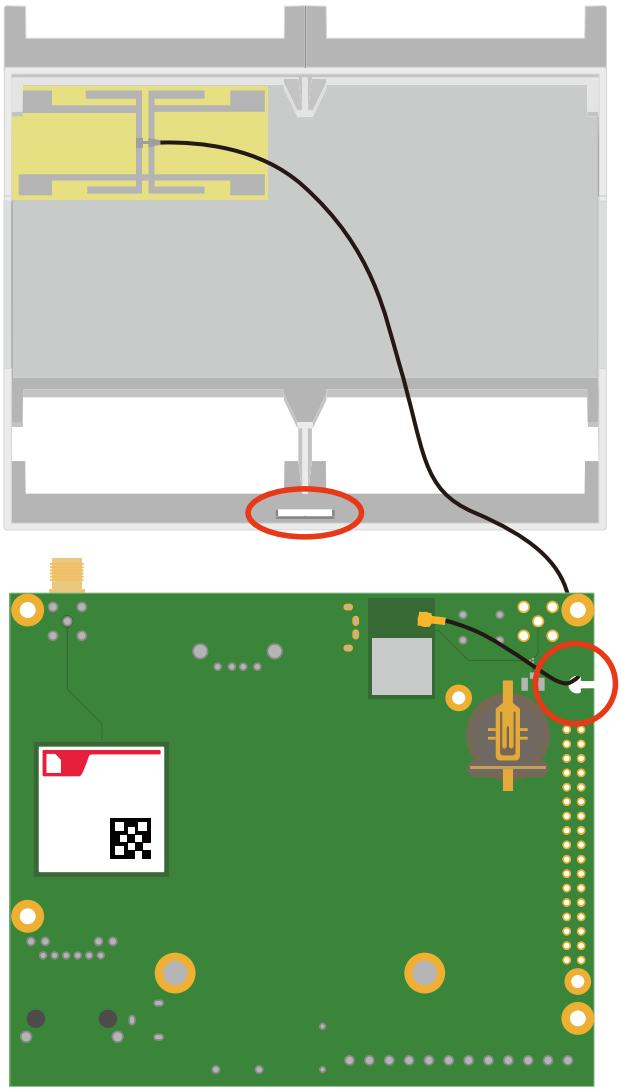


図 3.87 WLAN 基板アンテナの貼り付け位置 (スタンダードタイプ Cat.1 bis+WLAN モデル)

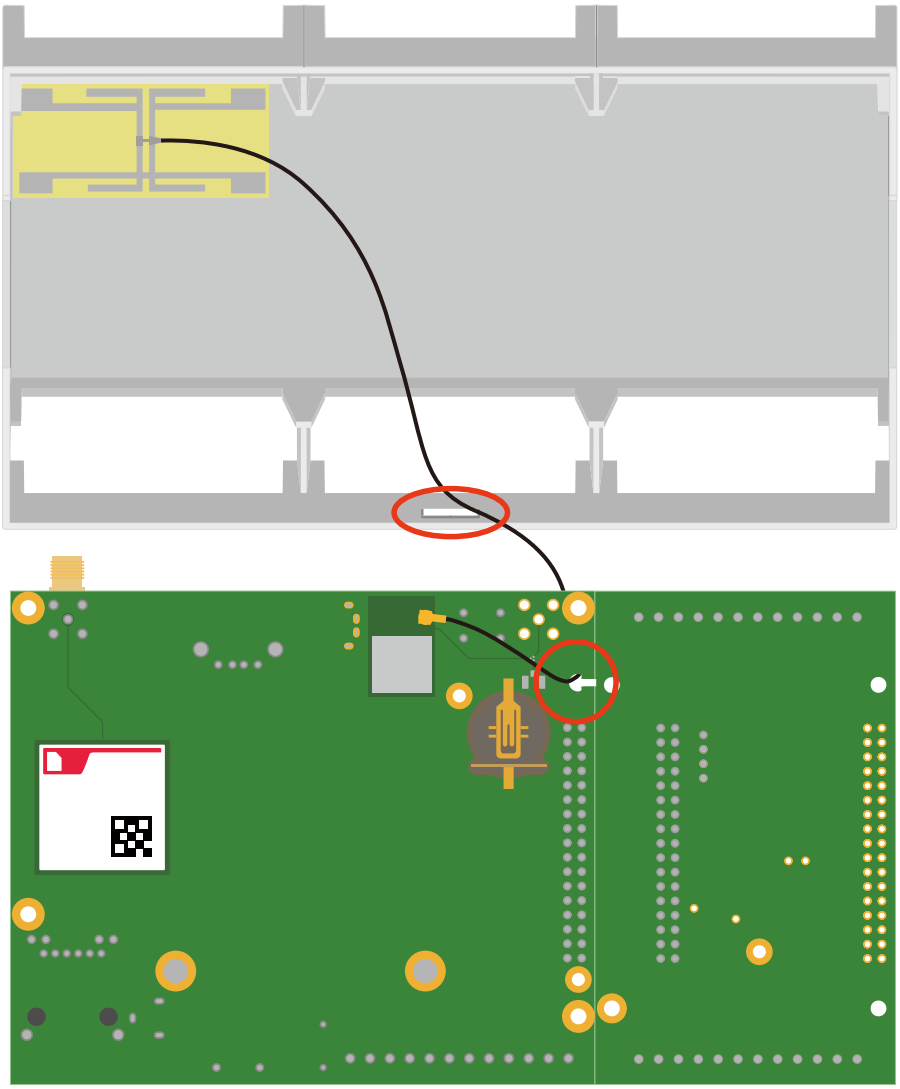


図 3.88 WLAN 基板アンテナの貼り付け位置 (+Di8+Ai4 タイプ Cat.1 bis+WLAN モデル)

WLAN/LAN モデルの場合、アンテナケーブルがタクトスイッチに干渉しないように引き回します。

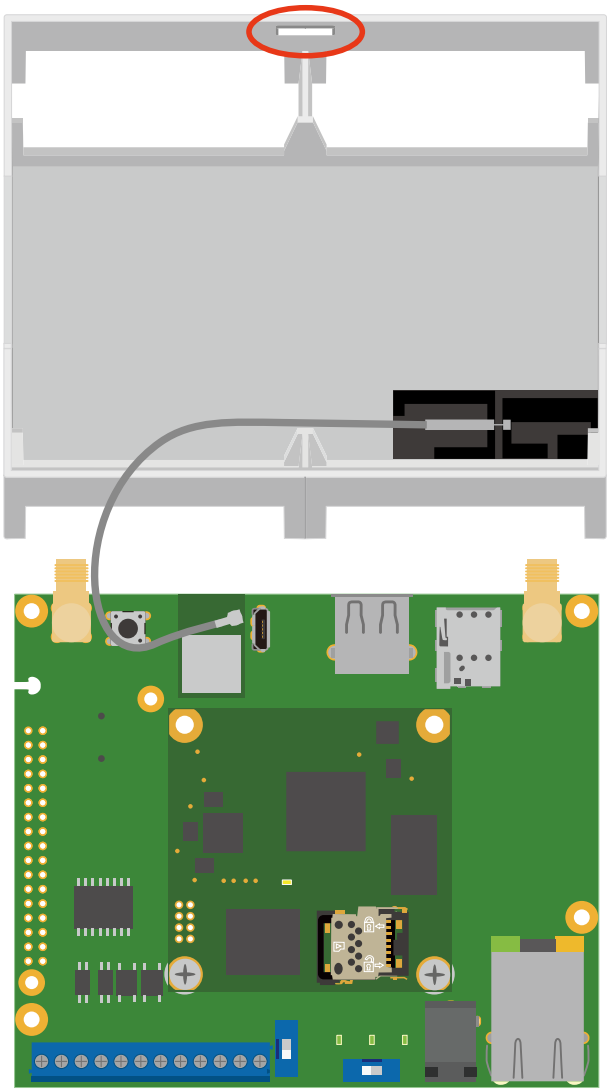


図 3.89 WLAN 基板アンテナの貼り付け位置 (スタンダードタイプ WLAN モデル)

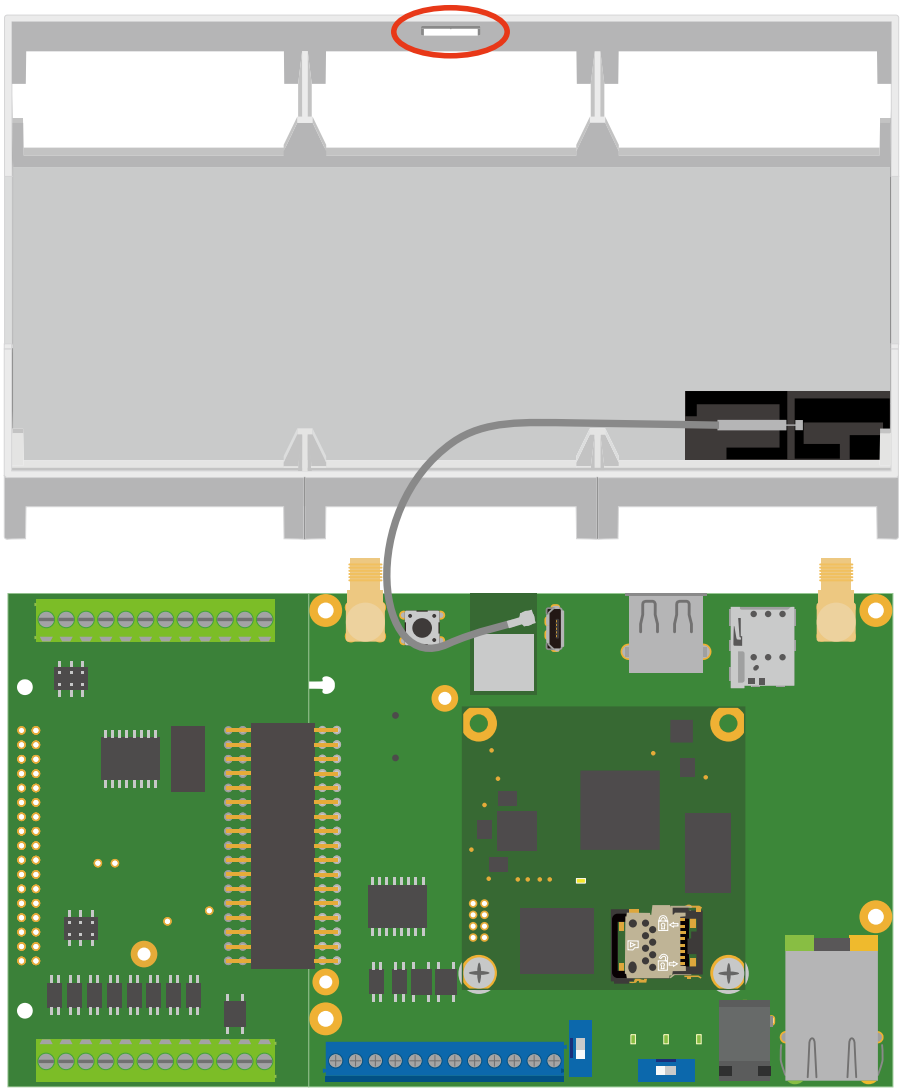


図 3.90 WLAN 基板アンテナの貼り付け位置 (+Di8+Ai4 タイプ WLAN モデル)

アンテナケーブルは「図 3.91. WLAN 基板アンテナのケーブル引き回し」の NG の方向に常に力が加わるような引き回しを行わないでください。コネクタ破損の恐れがあります。

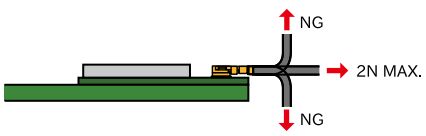


図 3.91 WLAN 基板アンテナのケーブル引き回し

3.7. インターフェースの使用方法とデバイスの接続方法

Armadillo-IoT ゲートウェイ A6E の各インターフェースの使用方法と、開発するシステムに必要な周辺デバイスの接続方法を紹介します。

各インターフェースを使用するために、コンテナを操作する場合があります。コンテナの操作方法は、「6.9. コンテナの概要と操作方法を知る」を参照してください。

3.7.1. インターフェース仕様

Armadillo-IoT ゲートウェイ A6E のインターフェースと使用部品を以下に示します。

製品型番により部品の搭載/非搭載が異なります。お手元の製品の搭載部品についての詳細は納入仕様書をご確認ください。また、「表 3.17. Armadillo-IoT ゲートウェイ A6E インターフェース一覧 (スタンダードタイプ)」に記載した型番の部品が必ずしも搭載されているとは限りません。形状確認の参考程度にご確認ください。



本製品シリーズの納入仕様書は、アットマークテクノ Armadillo サイト (https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/spec)からご覧いただけます。(要ログイン)

3.7.1.1. スタンダードタイプ

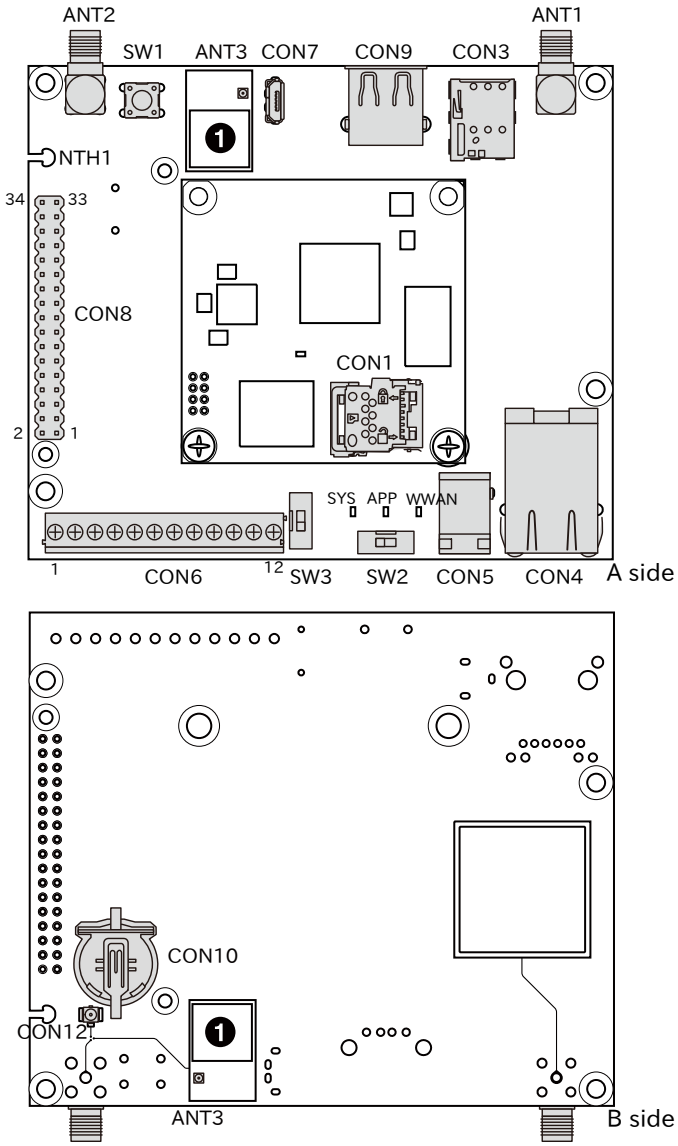


図 3.92 Armadillo-IoT ゲートウェイ A6E のインターフェース (スタンダードタイプ)

❶ Cat.1 bis モデルの場合は A 面側、WLAN モデルの場合は B 面側に ANT3 が搭載されます。

表 3.17 Armadillo-IoT ゲートウェイ A6E インターフェース一覧 (スタンダードタイプ)

部品番号	インターフェース名	型番	メーカー
CON1	SD インターフェース	SDHK-8BNS-K-303-TB(HF)	J.S.T.Mfg.
CON3	nanoSIM インターフェース	SF72S006VBDR2500	Japan Aviation Electronics Industry
CON4	LAN インターフェース	08B0-1X1T-36-F	Bel Fuse Inc.
CON5	電源入力インターフェース	PJ-102AH	Same Sky
CON6	入出力インターフェース	1-1776275-2	TE Connectivity
CON7	USB コンソールインターフェース	ZX80-B-5P(30)	HIROSE ELECTRIC
CON8	拡張インターフェース	A1-34PA-2.54DSA(71)	HIROSE ELECTRIC

部品番号	インターフェース名	型番	メーカー
CON9	USB インターフェース	SS-52100-001	Bel Fuse Inc.
CON10	RTC バックアップインターフェース	BH-44C-5	Adam Tech
CON12	アンテナ中継インターフェース	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
ANT1	LTE アンテナインターフェース	S-037-TGG	COSMTEC RESOURCES
ANT2	アンテナインターフェース	S-037-TGG	COSMTEC RESOURCES
ANT3	WLAN/BT アンテナインターフェース	20449-001E	IPEX
SYS	システム LED	SML-D12M1WT86	ROHM
APP	アプリケーション LED	SML-D12M1WT86	ROHM
WWAN	ワイヤレス WAN LED	SML-D12M1WT86	ROHM
SW1	ユーザースイッチ	SKHHDJA010	ALPS ELECTRIC
SW2	起動デバイス設定スイッチ	DS01-254-S-01BE	Same Sky
SW3	RS-485 終端抵抗設定スイッチ	DS01-254-S-01BE	Same Sky

3.7.1.2. +Di8+Ai4 タイプ

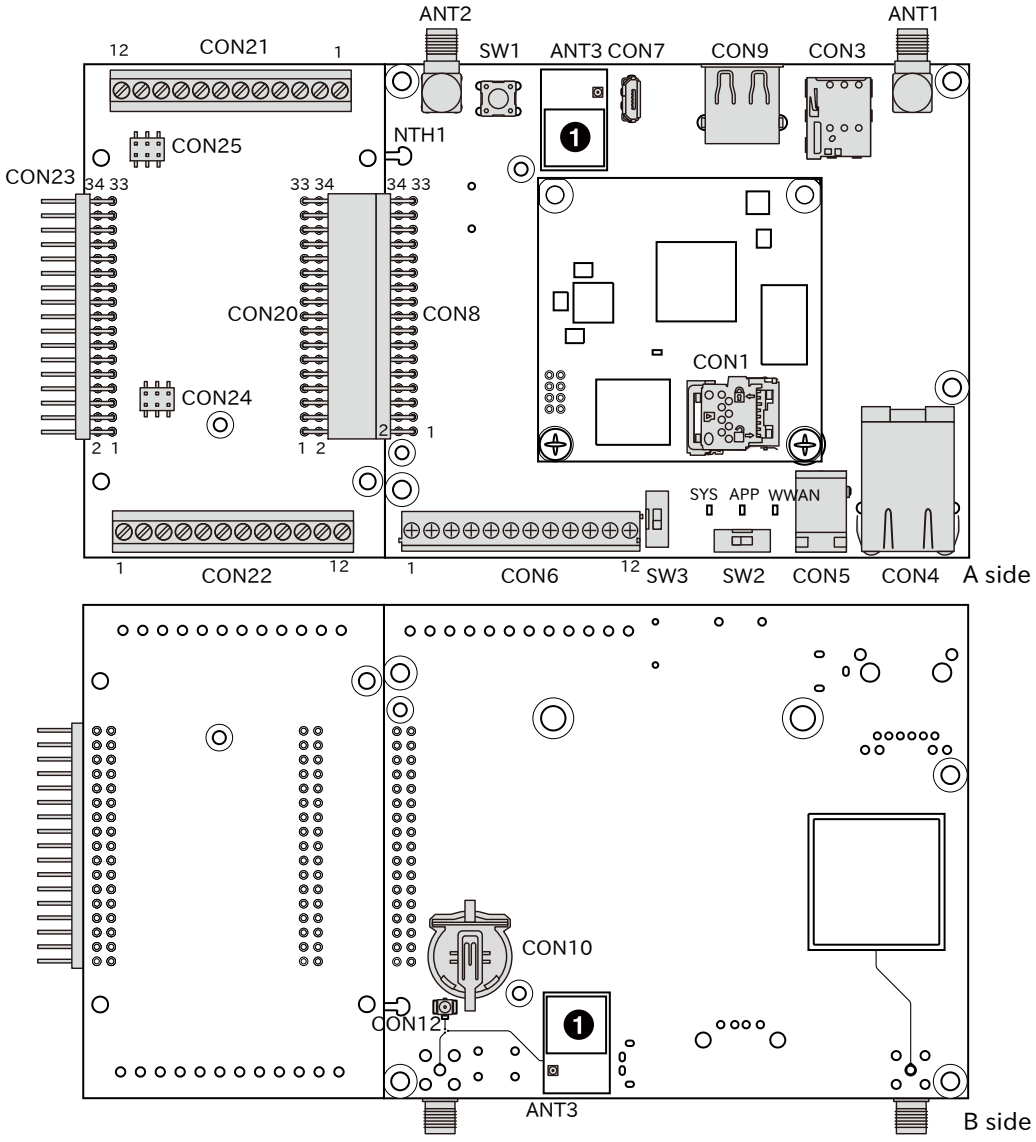


図 3.93 Armadillo-IoT ゲートウェイ A6E のインターフェース (+Di8+Ai4 タイプ)

❶ Cat.1 bis+WLAN モデルの場合は A 面側、WLAN モデルの場合は B 面側に ANT3 が搭載されます。

表 3.18 Armadillo-IoT ゲートウェイ A6E インターフェース一覧 (+Di8+Ai4 タイプ)

部品番号	インターフェース名	型番	メーカー
CON1	SD インターフェース	SDHK-8BNS-K-303-TB(HF)	J.S.T.Mfg.
CON3	nanoSIM インターフェース	SF72S006VBDR2500	Japan Aviation Electronics Industry
CON4	LAN インターフェース	08B0-1X1T-36-F	Bel Fuse Inc.
CON5	電源入力インターフェース	PJ-102AH	Same Sky
CON6	入出力インターフェース	1-1776275-2	TE Connectivity
CON7	USB コンソールインターフェース	ZX80-B-5P(30)	HIROSE ELECTRIC
CON8	拡張インターフェース	PSR-420256-17	Hirosugi-Keiki
CON9	USB インターフェース	SS-52100-001	Bel Fuse Inc.
CON10	RTC バックアップインターフェース	BH-44C-5	Adam Tech
CON12	アンテナ中継インターフェース	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
CON20	拡張インターフェース	AFSR-42085-17T	Hirosugi-Keiki
CON21	アナログ入力インターフェース	EBWA-12-A	Adam Tech
CON22	入出力インターフェース	EBWA-12-A	Adam Tech
CON23	拡張インターフェース	PSR-420256-17	Hirosugi-Keiki
CON24	設定ジャンパ	BF030-06A-B0-0360-0277-0600-LB	Global Connector Technology
CON25	設定ジャンパ	BF030-06A-B0-0360-0277-0600-LB	Global Connector Technology
ANT1	LTE アンテナインターフェース	S-037-TGG	COSMTEC RESOURCES
ANT2	アンテナインターフェース	S-037-TGG	COSMTEC RESOURCES
ANT3	WLAN/BT アンテナインターフェース	20449-001E	IPEX
SYS	システム LED	SML-D12M1WT86	ROHM
APP	アプリケーション LED	SML-D12M1WT86	ROHM
WWAN	ワイヤレス WAN LED	SML-D12M1WT86	ROHM
SW1	ユーザースイッチ	SKHHDJA010	ALPS ELECTRIC
SW2	起動デバイス設定スイッチ	DS01-254-S-01BE	Same Sky
SW3	RS-485 終端抵抗設定スイッチ	DS01-254-S-01BE	Same Sky

3.7.2. 周辺装置との接続

Armadillo-IoT ゲートウェイ A6E と周辺装置の接続例を「図 3.94. Armadillo-IoT ゲートウェイ A6E の接続例」に示します。

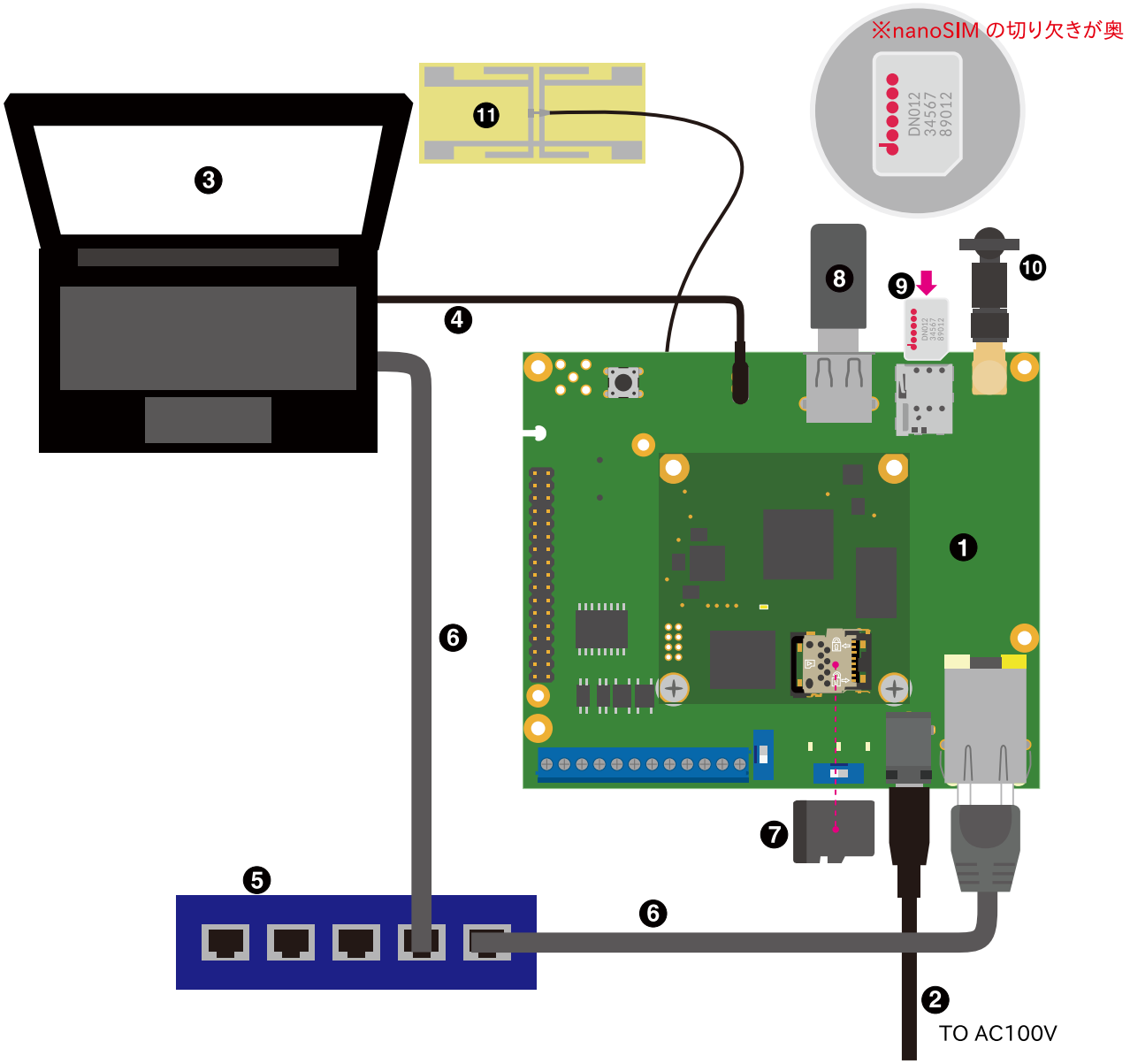


図 3.94 Armadillo-IoT ゲートウェイ A6E の接続例

1. Armadillo-IoT ゲートウェイ A6E
2. AC アダプタ(12V/2A)
3. 作業用 PC
4. シリアル通信用 USB ケーブル(A-microB)
5. LAN HUB
6. Ethernet ケーブル
7. SD カード

❽ USB メモリ

❾ nanoSIM カード

❿ LTE 用外付けアンテナ

⓫ WLAN 基板アンテナ

3.7.3. 電源を入力する

電源入力用に DC ジャック(CON5)と端子台(CON6)が搭載されています。

AC アダプタを使用する場合は DC ジャック(CON5)、電線から電源供給する場合は端子台(CON6)を使用してください。



CON5、CON6 の電源ライン(VIN)は接続されており、同時に電源を供給することはできません。



電源を再投入する際は、コンデンサに蓄えられた電荷を抜くため、電源を切断後、一定時間以上待つ必要があります。開発セット付属の AC アダプタの場合に必要な時間は以下のとおりです。

- DC プラグ側で電源を切断した場合 : 約 5 秒
- AC プラグ側で電源を切断した場合 : 約 1 分

コンデンサに蓄えられた電荷が抜ける前に電源を再投入した場合、電源シーケンスが守られず、起動しない等の動作不具合の原因となります。

3.7.3.1. 電源入力インターフェース(CON5)

AC アダプタから電源を供給する際に使用します。DC ジャックが実装されており、「図 3.95. AC アダプタの極性マーク」と同じ極性マークの AC アダプタが使用できます。対応プラグは内径 2.1mm、外形 5.5mm です。



図 3.95 AC アダプタの極性マーク



CON5 から電源供給する場合、AC アダプタの DC プラグを DC ジャックに接続してから、AC プラグをコンセントに挿してください。

3.7.3.2. 電源入力(CON6)

端子台を実装しています。電源電圧範囲は DC 8V～26.4V です。端子台に電線を接続する際は、ねじサイズに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのプラスねじです。

接続可能な電線については、「表 3.19. 接続可能な電線(電源)」をご確認ください。

表 3.19 接続可能な電線(電源)

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。

表 3.20 電源入力(CON6) 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源入力(VIN)
2	GND	Power	電源(GND)
12	GND	Power	電源(GND)



振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。



電線の先端に予備半田しないでください。正しい接続ができなくなります。



端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

3.7.4. Ethernet を使用する

10BASE-T/100BASE-TX に対応した LAN インターフェース(CON4)が搭載されています。カテゴリ 5 以上の Ethernet ケーブルを接続できます。AUTO-MDIX 機能を搭載しており、ストレートケーブルまたはクロスケーブルを自動認識して送受信端子を切り替えます。

3.7.4.1. LAN インターフェース(CON4)

LAN インターフェース(CON4)の信号線は Ethernet PHY(LAN8720AI-CP/Microchip Technology)を経由して i.MX6ULL の 10/100-Mbps Ethernet MAC(ENET1)に接続されています。

機能
- 通信速度: 100Mbps (100BASE-TX), 10Mbps (10BASE-T)

- 通信モード: Full-Duplex (全二重), Half-Duplex (半二重)
- Auto Negotiation サポート
- キャリア検知サポート
- リンク検出サポート

ネットワークデバイス
- eth0

表 3.21 LAN インターフェース(CON4) 信号配列

ピン番号	ピン名	I/O	説明
1	TX+	In/Out	送信データ(+)
2	TX-	In/Out	送信データ(-)
3	RX+	In/Out	受信データ(+)
4	-	-	5 ピンと接続後に 75Ω 終端
5	-	-	4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	受信データ(-)
7	-	-	8 ピンと接続後に 75Ω 終端
8	-	-	7 ピンと接続後に 75Ω 終端

表 3.22 CON4 LAN LED の動作

名称(色)	状態	説明
LAN スピード LED(緑)	消灯	10Mbps で接続されている、もしくは Ethernet ケーブル未接続
	点灯	100Mbps で接続されている
LAN リンクアクティビティ LED(黄)	消灯	リンクが確立されていない
	点灯	リンクが確立されている
	点滅	リンクが確立されており、データを送受信している

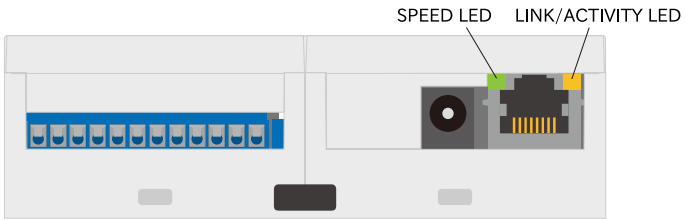


図 3.96 CON4 LAN LED

3.7.4.2. ネットワーク設定をする

有線 LAN の設定を ABOS Web で行う場合は「3.9. ネットワーク設定」、コマンドラインで行う場合は「6.16. コマンドラインからネットワーク設定を行う」をご確認ください。

3.7.5. 無線 LAN を使用する

Cat.1 bis+WLAN モデルおよび WLAN モデルには、WLAN+BT コンボモジュール(Sterling LWB5+/Ezurio)が搭載されています。

無線 LAN を使用する際は、WLAN/BT アンテナインターフェース(ANT3)にアンテナを接続してください。アンテナが接続されていない場合は、「3.6. WLAN アンテナの取り付けと取り外し」を参照し、アンテナを取り付けてください。

3.7.5.1. WLAN+BT コンボモジュール(WLAN)

WLAN+BT コンボモジュールの無線 LAN の信号線は i.MX6ULL の Ultra Secured Digital Host Controller(uSDHC2)に接続されています。

機能
- IEEE 802.11a/b/g/n/ac 準拠
- 最大通信速度: 49.5Mbps(理論値) [4]
- 動作モード: インフラストラクチャモード(STA/AP), アドホックモード
- チャンネル(2.4GHz): 1-14
- チャンネル(5GHz): 36-48, 52-64, 100-140

ネットワークデバイス
- `wlan0`



WLAN+BT コンボモジュールの技適認証取得済みのアンテナについて抜粋したリストを Armadillo サイト [https://armadillo.atmark-techno.com/]で公開しています。付属のアンテナ以外をご検討の際に、ご活用ください。

当社にて全てのアンテナの動作を確認したものではありませんので、通信性能の評価については、ユーザー様自身にて実施いただくようお願いいたします。



アンテナインターフェース(ANT2)に WLAN+BT コンボモジュール用のアンテナを接続するカスタマイズ品を製造することが可能です。詳細につきましては、アットマークテクノ営業部までお問い合わせください。

3.7.5.2. ネットワーク設定をする

無線 LAN の設定を ABOS Web で行う場合は「3.9.7. WWAN 設定」、コマンドラインで行う場合は「6.16.6. 無線 LAN」をご確認ください。



Sterling LWB5+ のファームウェアは、ATDE にインストールされている firmware-brcm80211 パッケージに含まれています。ファームウェアは Linux カーネルイメージ内に改変無く配置されます。firmware-ti-connectivity の著作権およびライセンス情報については、ATDE 上で `/usr/share/doc/firmware-brcm80211/copyright` を参照してください。

[4]Sterling LWB5+の最大通信速度は 433.3Mbps(802.11ac/1x1 SISO/HT80/MCS9/SGI)ですが、uSDHC2 を利用しているため、49.5Mbps に制限されます。

3.7.6. LTE を使用する

Armadillo-IoT ゲートウェイ A6E Cat.1 bis モデルには、SIMCom 製 SIM7672G または SIM7672JP が搭載されています。

LTE を使用する際は、LTE アンテナインターフェース(ANT1)にアンテナを取り付け、LTE データ通信で使用する nanoSIM カードを nanoSIM インターフェース(CON3)に挿入してください。nanoSIM カードの挿入向きについては、「図 3.97. nanoSIM カードの接続例」を参照してください。

3.7.6.1. LTE モジュール(Cat.1 bis)

LTE モジュールの信号線は、i.MX6ULL の USB 2.0 Controller(USB OTG2)に接続されています。

機能
- LTE データ通信
- リセットドライバによる LTE モジュール の電源制御

デバイスファイル
- `/dev/ttyACM0`

ネットワークデバイス
- `ppp0`



ModemManager が `/dev/ttyCommModem` のシンボリックリンクを作成して AT コマンド用ポートとして使用するため、他の USB デバイスを接続している場合、 `/dev/ttyACM0` から番号が変わる可能性があります。

3.7.6.2. nanoSIM インターフェース(CON3)

LTE データ通信で使用する nanoSIM カード用のインターフェースです。信号線は LTE モジュールに接続されています。

表 3.23 nanoSIM インターフェース(CON3) 信号配列

ピン番号	ピン名	I/O	説明
C1	SIM_VCC	Power	SIM 電源、LTE モジュールの CCVCC に接続
C2	SIM_RST	Out	SIM リセット、LTE モジュールの CCRST に接続
C3	SIM_CLK	Out	SIM クロック、LTE モジュールの CCCLK に接続
C5	GND	Power	電源(GND)
C6	SIM_VPP	-	未接続
C7	SIM_I/O	In	SIM データ、LTE モジュールの CCIO に接続

nanoSIM カードを挿入する際は「図 3.97. nanoSIM カードの接続例」のように、nanoSIM(UIM カード)の切り欠きを挿入方向に向け、刻印面を上にして挿入してください。

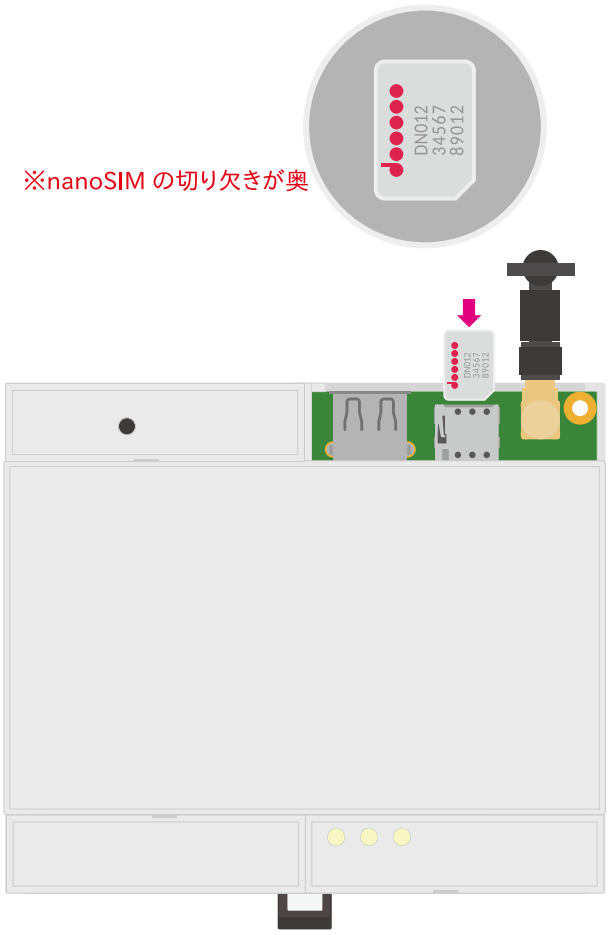


図 3.97 nanoSIM カードの接続例

3.7.6.3. LTE アンテナインターフェース(ANT1)

LTE データ通信時に利用するアンテナコネクタです。SMA オス端子のアンテナを接続することができます。アンテナコネクタの形状は「図 3.98. ANT1 接続可能なアンテナコネクタ形状」のとおりです。

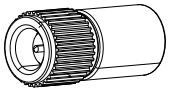


図 3.98 ANT1 接続可能なアンテナコネクタ形状

アンテナコネクタからアンテナまでの経路は 50Ω 同軸ケーブルでの延長が可能です。ただし、ケーブルロスが発生することにご注意ください。同軸ケーブルで延長する場合は、「図 3.99. ANT1 50Ω 同軸ケーブルでの延長例」を参考にケーブルを用意してください。



図 3.99 ANT1 50Ω 同軸ケーブルでの延長例



LTE モジュールメーカーにより、技適認証取得済みのアンテナについて抜粋したリストを Armadillo サイト [https://armadillo.atmark-techno.com/]で公開しています。付属のアンテナ以外をご検討の際に、ご活用ください。

当社にて全てのアンテナの動作を確認したものではありませんので、通信性能の評価については、ユーザー様自身にて実施いただくようお願いいたします。

3.7.6.4. ネットワーク設定をする

LTE のネットワーク設定を ABOS Web で行う場合は「3.9. ネットワーク設定」、コマンドラインで行う場合は「6.16.5. LTE」を参照してください。

3.7.6.5. LTE モデムの電源制御とリセット

LTE モデムの電源は、Armadillo の起動時に自動的にオンになり、Armadillo の終了時に自動的にオフになります。また、「6.16.5.10. LTE 再接続サービス」でも、通信状態に応じて LTE モデムのリセットなどを実施します。

LTE モデムのリセットや電源を制御するには、以下のコマンドを実行します。

処理が重複するため、以下のコマンドを実行する前に再接続サービスを停止する必要があります。再接続サービスを停止する場合は、「図 6.162. LTE 再接続サービスを停止する」を参考にしてください。

```
[armadillo:~#] wwan-force-restart
```

図 3.100 LTE モデムをリセットまたは 再起動する

```
[armadillo:~#] wwan-poweroff
```

図 3.101 LTE モデムの電源を切る

ネットワークの設定方法については「3.9. ネットワーク設定」を参照してください。

LTE 再接続サービスの設定、省電力設定に関しては「6.16.5. LTE」を参照してください。

3.7.7. SD カードを使用する

ハイスピード(最大クロック周波数: 49.5MHz)に対応した SD インターフェース(CON1)が搭載されています。

3.7.7.1. SD インターフェース(CON1)

microSD/microSDHC/microSDXC カード [5]を使用できます。SD インターフェース(CON1)の信号線は i.MX6ULL の Ultra Secured Digital Host Controller(uSDHC2)に接続されています。

[5]以降の説明では、共通の操作が可能な場合に、microSD/microSDHC/microSDXC カードを microSD カードと表記します。

SD インターフェース(CON1)に供給される電源は i.MX6ULL の NAND_ALE ピン(GPIO4_IO10)で制御が可能です。High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

機能
- カードタイプ: microSD/microSDHC/microSDXC/microSDIO
- バス幅: 1bit or 4bit
- スピードモード: Default Speed(24.75MHz), High Speed(49.5MHz)
- カードディテクトサポート

デバイスファイル

ディスクデバイス	先頭パーティション
/dev/mmcblk1	/dev/mmcblk1p1



Cat.1 bis+WLAN モデルおよび WLAN モデルの場合、SD インターフェース(CON1)は WLAN+BT コンボモジュールと排他使用となり、インストールディスク用途以外で使用できません。



SD インターフェース(CON1)は活線挿抜に対応していません。microSD カードの挿抜は、電源を切断してから行ってください。

表 3.24 SD インターフェース(CON1) 信号配列

ピン番号	ピン名	I/O	説明
1	DAT2	In/Out	SD データバス(bit2)、i.MX6ULL の NAND_DATA02 ピンに接続
2	CD/DAT3	In/Out	SD データバス(bit3)、i.MX6ULL の NAND_DATA03 ピンに接続
3	CMD	In/Out	SD コマンド/レスポンス、i.MX6ULL の NAND_WE_B ピンに接続
4	VDD	Power	電源(VCC_3.3V)
5	CLK	Out	SD クロック、i.MX6ULL の NAND_RE_B ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	SD データバス(bit0)、i.MX6ULL の NAND_DATA00 ピンに接続
8	DAT1	In/Out	SD データバス(bit1)、i.MX6ULL の NAND_DATA01 ピンに接続

3.7.7.2. microSD カードの挿抜方法

1. 上からカバーを軽く押し、約 1.2mm スライドさせて、ロックを解除します。

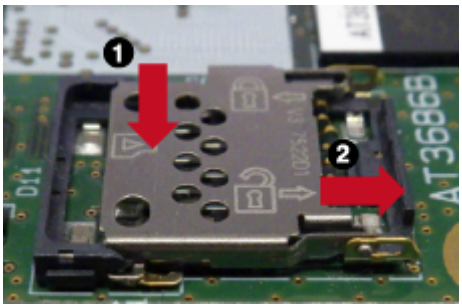


図 3.102 カバーのロックを解除する

2. カバーを開けます。

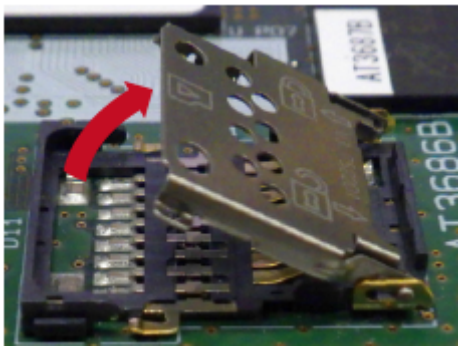


図 3.103 カバーを開ける



カバーは過度な力で回転させたり、回転方向以外の方向へ力を加えると、破損の原因となりますので、ご注意ください。

3. 任意の角度までトレイを開いた状態で、microSD カードを挿抜します。



図 3.104 microSD カードの挿抜



microSD カード挿入方向については、カバーに刻印されているカードマークを目安にしてください。

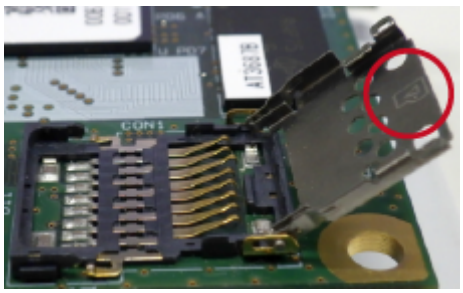


図 3.105 カードマークの確認

4. カバーを閉めます。

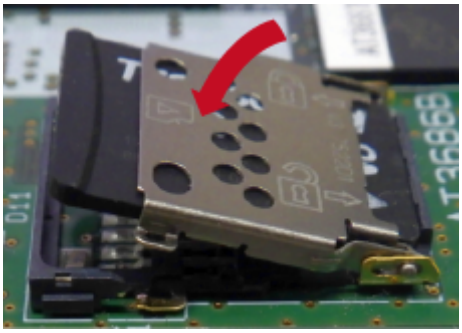


図 3.106 カバーを閉める

5. 上からカバーを軽く押し、約 1.2mm スライドさせて、ロックします。

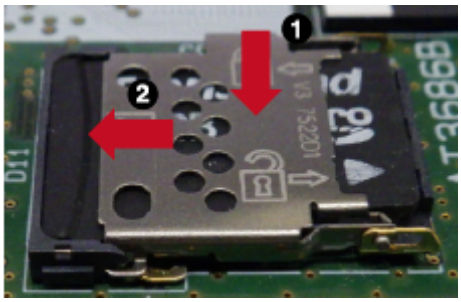


図 3.107 カバーをロックする



microSD カード装着後のカードの抜き取り手順は挿入時と同じです。

3.7.7.3. SD カードのマウント

microSD カードは、 /dev/mmcblk1 として認識されます。microSD カードを /mnt にマウントします。

```
[armadillo]# mount /dev/mmcblk1p1 /mnt
```

図 3.108 SD カードをマウントする例

SD カードのマウントの詳細については「6.17. コマンドラインからストレージを使用する」をご確認ください。

3.7.7.4. SD カードのフォーマット

通常、購入したばかりの SD カードは、ひとつのパーティションを持ち、FAT32 ファイルシステムでフォーマットされています。必要に応じてパーティション構成やファイルシステムを変更してください。

変更方法については、「6.17.1. ストレージのパーティション変更とフォーマット」をご確認ください。

3.7.7.5. コンテナで SD カードをマウントして使用する

microSD カードをコンテナ内からマウントして使用するには、Podman のイメージからコンテナを作成する際に、add_hotplugs に mmc を設定し、適切な権限を付与します。

alpine ベースのコンテナを SYS_ADMIN 権限を付与して作成し、microSD カードを使用する例を紹介します。

/etc/atmark/containers/sd_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo]#  vi /etc/atmark/containers/sd_example.conf
set_image docker.io/alpine
add_hotplugs mmc ❶
add_args --cap-add=SYS_ADMIN ❷
set_command sleep infinity
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo]# podman_start sd_example
Starting 'sd_example'
1d93ecff872276834e3c117861f610a9c6716c06eb95623fd56aa6681ae021d4
```

図 3.109 SD カードを使用するコンテナの作成例

❶ add_hotplugs に mmc を指定することで、 コンテナ内で microSD カードをホットプラグで認識します

❷ コンテナ内で microSD カードをマウントするための権限を付与します。

コンテナ内に入り、microSD カードを /mnt にマウントします。microSD カードは、 /dev/mmcblk1 として認識されます。

```
[armadillo]# podman exec -it sd_example sh
[container]# mount /dev/mmcblk1p1 /mnt
```

図 3.110 SD カードをマウントする例

3.7.8. USB デバイスを使用する

USB2.0 に対応した USB インターフェース(CON9)が搭載されています。

3.7.8.1. USB インターフェース(CON9)

信号線は i.MX6ULL の USB 2.0 Controller(USB OTG1)に接続されています。

USB デバイスに供給される電源 (USB_OTG1_VBUS) は i.MX6ULL の UART1_RTS_B ピン (GPIO1_IO19)で制御しており、High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

機能	・Universal Serial Bus Specification Revision 2.0 準拠
	・Enhanced Host Controller Interface (EHCI)準拠
	・転送レート: USB2.0 High-Speed (480Mbps), Full-Speed (12Mbps), Low-Speed (1.5Mbps)
デバイスファイル	・メモリデバイスの場合 : /dev/sdN (N : デバイスを認識した順に a から連番)
	・I/O デバイスの場合 : ファンクションに応じたデバイスファイル
sysfs ディレクトリ	・/sys/devices/platform/usb1-vbus/ CON9 の電源(USB_OTG1_VBUS)を制御するディレクトリです。

表 3.25 USB インターフェース(CON9) 信号配列

ピン番号	ピン名	I/O	説明
1	USB1_VBUS	Power	電源(USB_OTG1_VBUS)、i.MX6ULL の USB_OTG1_VBUS ピンに接続
2	USB1_DN	In/Out	USB1 のマイナス側信号、i.MX6ULL の USB_OTG1_DN ピンに接続
3	USB1_DP	In/Out	USB1 のプラス側信号、i.MX6ULL の USB_OTG1_DP ピンに接続
4	GND	Power	電源(GND)

3.7.8.2. USB メモリのマウント



USB デバイスが認識されない場合、USB デバイスの接続を拒否する設定が行われている可能性があります。

設定を変更するには「3.10. USB デバイスの接続を許可する」をご確認ください。

USB メモリを /mnt にマウントします。

```
[armadillo ~]# mount /dev/disk/by-label/[MYUSBMEMORY] /mnt ❶
```

図 3.111 USB メモリをマウントする例

❶ [MYUSBMEMORY] の部分は USB メモリに設定しているラベルに置き換えてください。



コンテナ内からマウントするデバイスを指定する際には /dev/sdN を使用することもできますが、他にもストレージデバイスを接続している場合などには N の値が変わることがあります。

USB メモリにラベルを設定している場合は、 /dev/disk/by-label/ 下にあるラベルと同名のファイルを指定することで、確実に目的のデバイスを使用できます。

USB メモリのマウントの詳細については「6.17. コマンドラインからストレージを使用する」をご確認ください。

3.7.8.3. USB メモリのフォーマット

通常、購入したばかりの USB メモリは、ひとつのパーティションを持ち、FAT32 ファイルシステムでフォーマットされています。必要に応じてパーティション構成やファイルシステムを変更してください。

変更方法については、「6.17.1. ストレージのパーティション変更とフォーマット」をご確認ください。

3.7.8.4. コンテナで USB メモリをマウントして使用する

USB メモリをコンテナ内からマウントして使用するには、Podman のイメージからコンテナを作成する際に、add_hotplugs に sd を設定し、適切な権限を付与します。この設定により、コンテナ起動後に USB メモリを接続しても正しく認識されます。

alpine ベースのコンテナを SYS_ADMIN 権限を付与して作成し、USB メモリを使用する例を紹介します。

/etc/atmark/containers/usbmem_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/usbmem_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_args --cap-add=SYS_ADMIN
add_hotplugs sd
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start usbmem_example
Starting 'usbmem_example'
387a2256530e9b35b5361ca681a99fba8f46d78b6a6cb8ecd60096246b9198a8
```

図 3.112 コンテナ内で USB メモリをマウントして使用するコンテナの作成例

コンテナ内に入り、USB メモリを /mnt にマウントします。

```
[armadillo ~]# podman exec -it usbmem_example sh
[container ~]# mount /dev/disk/by-label/[MYUSBMEMORY] /mnt ❶
```

図 3.113 コンテナ内で USB メモリをマウントする例

❶ [MYUSBMEMORY] の部分は USB メモリに設定しているラベルに置き換えてください。

3.7.8.5. ABOS 上でマウントした USB メモリをコンテナで使用する

ABOS 上でマウントした USB メモリをコンテナで使用する場合、ABOS 上で USB メモリをマウントし、Podman のイメージからコンテナを作成する際に、USB メモリをマウントしたディレクトリを渡します。

alpine ベースのコンテナを作成する例を紹介します。

USB メモリを /mnt にマウントします。

```
[armadillo ~]# mount -t vfat /dev/sda1 /mnt
```

図 3.114 ABOS 上で USB メモリをマウントする例

コンテナ内から USB メモリを使用するため、先ほど USB メモリをマウントしたディレクトリを渡します。/etc/atmark/containers/usbmem_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/usbmem_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /mnt
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start usbmem_example
Starting 'usbmem_example'
ef77d4bfd5b04f3b8b5ddcb5bfac321304fa64219a4b88c3130e45e5a14e1b3e
```

図 3.115 ABOS 上でマウントした USB メモリを使用するコンテナの作成例

コンテナ内に入り、USB メモリを確認します。

```
[armadillo ~]# podman exec -it usbmem_example sh
[container ~]# ls /mnt
```

図 3.116 USB メモリに保存されているデータの確認例

3.7.8.6. コンテナで USB シリアルデバイスを使用する

USB シリアルデバイスをコンテナ内から使用するには、Podman のイメージからコンテナを作成する際に、add_hotplugs に ttyUSB を指定します。この設定により、コンテナ起動後に USB シリアルデバイスを接続しても正しく認識されます。

alpine ベースのコンテナを作成し、その中で USB シリアルデバイスを使用する例を紹介します。

/etc/atmark/containers/usb_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/usb_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_hotplugs ttyUSB
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
```

```
a6215f271958c760a2975a67650160441l5dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start usb_example
Starting 'usb_example'
34cb0e60d6274ac1df87aed58a461bcf56d0c117c4d377af130605ea399e0950
```

図 3.117 USB シリアルデバイスを使用するコンテナの作成例

コンテナ内に入り、 setserial コマンドで現在の設定を確認します。

```
[armadillo ~]# podman exec -it usb_example sh
[container ~]# setserial -a /dev/serial/by-id/usb-067b_2303-if00-port0
/dev/serial/by-id/usb-067b_2303-if00-port0, Line 4, UART: 16654, Port: 0x0000, IRQ: 0
        Baud_base: 460800, close_delay: 0, divisor: 0
        closing_wait: infinite
        Flags: spd_normal
```

図 3.118 USB シリアルデバイスの設定の確認例



コンテナ内からのデバイスの指定には /dev/ttyUSBN を使用することもできますが、デバイスを接続するタイミングによっては N の値が変わる可能性があります。

/dev/serial/by-id/ 下にあるファイルを指定することで確実に目的のデバイスを使用することができます。

3.7.8.7. コンテナで USB カメラを使用する

USB カメラをコンテナ内から使用するには、Podman のイメージからコンテナを作成する際に、add_hotplugs に video4linux を設定します。この設定により、コンテナ起動後に USB カメラを接続しても正しく認識されます。

alpine ベースのコンテナを作成し、その中で USB カメラを使用する例を紹介します。

/etc/atmark/containers/usbcam_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/usbcam_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_hotplugs video4linux
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a67650160441l5dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start usbcam_example
```

```
Starting 'usbcam_example'
ffe06090b45826cc0b1c7710e9e850ba9521d36b70de4288d0dfe1fe91a35632
```

図 3.119 USB カメラを使用するコンテナの作成例

コンテナ内に入り、接続されている USB デバイスの情報を確認します。

```
[armadillo ~]# podman exec -it usbcam_example sh
[container ~]# ls /dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_78DA8CAF-video-index0
/dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_78DA8CAF-video-index0
```

図 3.120 USB カメラの情報を確認する例

GStreamer などのマルチメディアフレームワークと組み合わせることで、USB カメラからの映像のキャプチャが可能となります。



コンテナ内からデバイスを指定する際には /dev/videoN を使用することもできますが、デバイスを接続するタイミングによっては N の値が変わる可能性があります。

/dev/v4l/by-id/ 下にあるファイルを指定することで、確実に目的のデバイスを使用できます。

3.7.8.8. Type-A コネクタから供給される電源の ON/OFF を制御する

Type-A コネクタ(CON9)から供給される電源(USB_OTG1_VBUS)の ON/OFF を切り替えることができます。

コンテナ内で動作するアプリケーションからこの切り替えを行うには、コンテナイメージからコンテナを作成する際にホスト OS 側の /sys ディレクトリを渡します。デフォルト状態でもマウントされていますが、読み取り専用になっていて使用できませんのでご注意ください。以下は、 /sys を渡して alpine イメージからコンテナを作成する例です。ここで渡された /sys ディレクトリはコンテナ内の同じ /sys にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/usbvbus_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /sys
[armadillo ~]# podman_start usbvbus_example
Starting 'usbvbus_example'
ae7f9b0c4abe9f40531a1db706e255d7abb60e979e44363898c3bcefd06e9b22
```

図 3.121 電源の ON/OFF を切り替えるためのコンテナ作成例

コンテナ内に入り、/sys/devices/platform/usb1-vbus/ ディレクトリ内の state ファイルに、enabled か disabled を書き込むことで ON/OFF を切り替えます（デフォルトは enabled です）。

```
[armadillo ~]# podman exec -it usbvbus_example sh
[container ~]# echo disabled > /sys/devices/platform/usb1-vbus/state ❶
[container ~]# echo enabled > /sys/devices/platform/usb1-vbus/state  ❷
```

図 3.122 電源の制御例

❶ CON9 から供給される電源を OFF にします。

❷ CON9 から供給される電源を ON にします。

3.7.9. UART を使用する

USB コンソールインターフェース(CON7)、RS-485 インターフェース(CON6)が搭載されています。また、拡張インターフェース(CON8)で UART を最大 2 ポートまで拡張できます。[6]

3.7.9.1. USB コンソール用インターフェース(CON7)

USB コンソール用のインターフェースです。

信号線は USB シリアル変換 IC(CP2102N/Sillicon Labs)を経由して、i.MX6ULL の Universal Asynchronous Receiver/Transmitter(UART3)に接続されています。

機能
- フォーマット
  - データビット長: 8 ビット
  - ストップビット長: 1 ビット
  - パリティ: なし
  - フロー制御: なし
- 最大データ転送レート:3Mbps

デバイスファイル
- /dev/ttymxc2



i.MX6ULL の UART の最大データ転送レートは 4Mbps ですが、USB シリアル変換 IC（CP2102N／Silicon Labs）の上限が 3Mbps のため、USB コンソール用インターフェース（CON7）も 3Mbps が最大となります。

表 3.26 USB コンソール用インターフェース(CON7) 信号配列

ピン番号	ピン名	I/O	説明
1	VBUS_CNSL	Power	電源(VBUS_CNSL)
2	CNSL_USB_D-	In/Out	コンソール用 USB のマイナス側信号、USB シリアル変換 IC に接続
3	CNSL_USB_D+	In/Out	コンソール用 USB のプラス側信号、USB シリアル変換 IC に接続
4	CNSL_USB_ID	-	未接続

[6] +Di8+Ai4 タイプの場合、拡張インターフェース(CON8)に+Di8+Ai4 拡張ボードが接続されており、拡張できません。

ピン番号	ピン名	I/O	説明
5	GND	Power	電源(GND)

3.7.9.2. RS-485 インターフェース(CON6)

2 線式の RS-485 インターフェースです。RS-485 の信号線は RS-485 トランシーバを経由して、i.MX6ULL の Universal Asynchronous Receiver/Transmitter(UART5)に接続されています。

終端抵抗 120Ω の ON/OFF をスイッチで切り替えることができます。設定方法は「3.7.9.3. RS-485 終端抵抗設定スイッチ(SW3)」を参照してください。

機能
- 最大データ転送レート：4Mbps
- 半二重対応

デバイスファイル
- `/dev/ttymxc4`

端子台を実装しています。ねじに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのプラスねじです。

接続可能な電線については、「表 3.27. 接続可能な電線(RS-485)」をご確認ください。

表 3.27 接続可能な電線(RS-485)

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。



振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。



電線の先端に予備半田しないでください。正しい接続ができなくなります。



端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

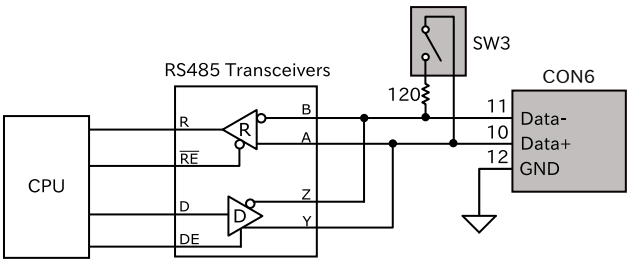


図 3.123 RS-485 内部回路

表 3.28 RS-485 インターフェース(CON6) 信号配列

ピン番号	ピン名
10	DATA+
11	DATA-
12	GND

3.7.9.3. RS-485 終端抵抗設定スイッチ(SW3)

RS-485 の終端抵抗設定スイッチです。終端抵抗 120Ω の ON/OFF を切り替えることができます。

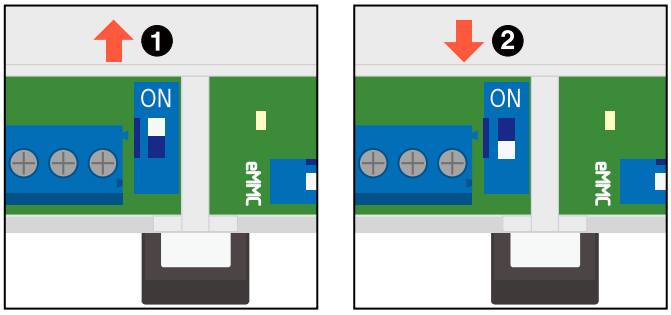


図 3.124 スイッチの状態と終端抵抗の ON/OFF

❶ 終端抵抗 120Ω が ON になります。

❷ 終端抵抗 120Ω が OFF になります。



終端は RS-485 の信号線の最遠端で行います。Armadillo-IoT ゲートウェイ A6E が最遠端になる場合は終端抵抗を ON にしてください。

3.7.9.4. 拡張インターフェース UART(CON8)

拡張インターフェース(CON8)で UART として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex] をご確認ください。i.MX6ULL の Universal Asynchronous Receiver/Transmitter(UART1、UART7)に接続されたピンを使用可能です。

機能
- フォーマット
  - データビット長: 7 or 8 ビット
  - ストップビット長: 1 or 2 ビット
  - パリティ: 偶数 or 奇数 or なし
  - フロー制御: CTS/RTS or XON/XOFF or なし
- 最大データ転送レート:4Mbps
- 信号レベル: VCC_3.3V

デバイスファイル

シリアルインターフェース	デバイスファイル
UART1	/dev/ttymxc0
UART7	/dev/ttymxc6

3.7.9.5. コンテナでシリアル通信する

コンテナ内で動作するアプリケーションから RS-232C や RS-485 などのシリアル通信を行うには、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/ttymxcN を渡します。

alpine ベースのコンテナを作成し、その中でシリアル通信を行う例を紹介します。

/etc/atmark/containers/serial_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/serial_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/ttymxc0
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start serial_example
Starting 'serial_example'
3999f09d51253371cacffd68967c90fdd5250770888a82f59d7810b54fcc873e
```

図 3.125 シリアルインターフェースを使用するコンテナの作成例

コンテナ内に入り、 setserial コマンドで現在の設定を確認します。

```
[armadillo ~]# podman exec -it serial_example sh
[container ~]# setserial -a /dev/ttymxc0
/dev/ttymxc0, Line 0, UART: undefined, Port: 0x0000, IRQ: 29
        Baud_base: 5000000, close_delay: 50, divisor: 0
        closing_wait: 3000
        Flags: spd_normal
```

図 3.126 シリアルインターフェイスの設定の確認例

3.7.10. GPIO を使用する

拡張インターフェース(CON8)で GPIO を最大 22 ポートまで拡張できます。[7]

3.7.10.1. 拡張インターフェース GPIO(CON8)

拡張インターフェース(CON8)で GPIO として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex] をご確認ください。i.MX6ULL の General Purpose Input/Output(GPIO)に接続されたピンを使用可能です。

機能
- 信号レベル: VCC_3.3V

デバイスファイル
- /dev/gpiochip0
- /dev/gpiochip2
- /dev/gpiochip3



sysfs GPIO クラス（/sys/class/gpio/）は使用しないでください。

拡張インターフェースの各ピンに対応する GPIO 名を次に示します。

表 3.29 GPIO に対応する CON8 ピン番号

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
7	CON8_7	GPIO1_IO01	gpiochip0	1
8	CON8_8	GPIO1_IO02	gpiochip0	2
9	CON8_9	GPIO1_IO03	gpiochip0	3
10	CON8_10	GPIO1_IO04	gpiochip0	4
14	CON8_14	GPIO3_IO05	gpiochip2	5
15	CON8_15	GPIO3_IO06	gpiochip2	6
16	CON8_16	GPIO3_IO07	gpiochip2	7
17	CON8_17	GPIO3_IO08	gpiochip2	8
19	CON8_19	GPIO3_IO10	gpiochip2	10
20	CON8_20	GPIO3_IO11	gpiochip2	11
21	CON8_21	GPIO3_IO12	gpiochip2	12
22	CON8_22	GPIO3_IO13	gpiochip2	13
23	CON8_23	GPIO3_IO14	gpiochip2	14
24	CON8_24	GPIO3_IO15	gpiochip2	15
25	CON8_25	GPIO3_IO16	gpiochip2	16
26	CON8_26	GPIO3_IO20	gpiochip2	20
27	CON8_27	GPIO3_IO21	gpiochip2	21
28	CON8_28	GPIO3_IO22	gpiochip2	22
29	CON8_29	GPIO4_IO25	gpiochip3	25
30	CON8_30	GPIO4_IO26	gpiochip3	26
31	CON8_31	GPIO4_IO27	gpiochip3	27

[7]+Di8+Ai4 タイプの場合、拡張インターフェース(CON8)に+Di8+Ai4 拡張ボードが接続されており、拡張できません。

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
32	CON8_32	GPIO4_IO28	gpiochip3	28



Linux 5.10.233-r0 以降では拡張インターフェースの GPIO に「CON8_14」の様な名前でアクセス可能となりました。gpioinfo コマンドでご確認ください。

3.7.10.2. GPIO をコンテナで使用する

コンテナ内で動作するアプリケーションから GPIO を使用する場合、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/gpiochipN を渡すことで、GPIO**N+1** を操作できます。

alpine ベースのコンテナを作成し、その中で GPIO を操作する例を紹介します。

/etc/atmark/containers/gpio_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 gpiochip2 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/gpio_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/gpiochip2
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start gpio_example
Starting 'gpio_example'
956a0fecc48d5ea1210069910f7bb48b9e90b2dadb12895064d9776dae0360b5
```

図 3.127 GPIO を扱うためのコンテナ作成例

コンテナ内に入り、必要なソフトウェアをインストールして、GPIO を操作します。ここでは、CON8 の 27 ピン(GPIO3_IO21)を操作します。

```
[armadillo ~]# podman exec -it gpio_example sh
[container ~]# apk upgrade
[container ~]# apk add libgpiod
[container ~]# gpioget --numeric CON8_27 ❶
0 ❷
[container ~]# gpioset -t0 CON8_27=1 ❸
```

図 3.128 コンテナ内からコマンドで GPIO を操作する例

❶ CON8_27 (GPIO3 番号 21) の値を取得します。

❷ 取得した値を表示します。

❸ CON8_27 (GPIO3 番号 21) に 1(High) を設定します。



libgpiod バージョンの注意

gpioset 、 gpioget および gpioinfo のコマンドは libgpiod のバージョンによって引数の仕様が変わります。

ABOS では、v3.20 までは libgpiod1、v3.21 以降は libgpiod2 がインストールされています。コンテナでは、Alpine 3.21 および Debian trixie 以降で libgpiod2 がインストールされています。

使い方の違いは https://armadillo.atmark-techno.com/howto/libgpiod2-update をご参照ください。

本書では libgpiod2 準拠で手順を紹介します。

gpiodetect コマンドで認識している gpiochip をリスト表示できます。

```
[container ~]# gpiodetect
gpiochip2 [30220000.gpio] (32 lines)
```

図 3.129 gpiochip をリスト表示

gpioinfo コマンドで、指定した gpiochip の詳細情報を表示できます。

```
[container ~]# gpioinfo -c gpiochip2
gpiochip2 - 32 lines:
        line   0:       unnamed                 input
        line   1:       unnamed                 input
        line   2:       unnamed                 input
        line   3:       unnamed                 input
        line   4:       unnamed                 input
        line   5:       "CON8_14"               input
        line   6:       "CON8_15"               input
        line   7:       "CON8_16"               input
        line   8:       "CON8_17"               input
        line   9:       unnamed                 input
        line  10:       "CON8_19"               input
        line  11:       "CON8_20"               input
        line  12:       "CON8_21"               input
        line  13:       "CON8_22"               input
        line  14:       "CON8_23"               input
        line  15:       "CON8_24"               input
        line  16:       "CON8_25"               input
        line  17:       unnamed                 input
        line  18:       unnamed                 output consumer=?
        line  19:       unnamed                 input
        line  20:       "CON8_26"               input
        line  21:       "CON8_27"               input
        line  22:       "CON8_28"               input
        line  23:       unnamed                 input
```

```
        line  24:       unnamed                 input
        line  25:       "CON8_29"               input
        line  26:       "CON8_30"               input
        line  27:       "CON8_31"               input
        line  28:       "CON8_32"               input
        line  29:       unnamed                 input
        line  30:       unnamed                 input
        line  31:       unnamed                 input
```

図 3.130 gpiochip の詳細情報を表示

C 言語プログラムから操作する場合は、GPIO 操作ライブラリである libgpiod を使用できます。

3.7.11. 接点入力を使用する

スタンダードタイプの場合、2 ポートの接点入力(CON6)、+Di8+Ai4 タイプの場合、2 ポートの接点入力(CON6)に加えて 8 ポートの接点入力(CON22)を使用できます。

3.7.11.1. 接点入力(CON6)

接点入力部はフォトカプラによる絶縁入力(電流シンク出力タイプに接続可能)となっています。入力部を駆動するために電源は、外部から供給する必要があります。

機能
- 接点入力 x 2
- 入力インピーダンス: 4.7kΩ
- 定格電圧: DC 8～26.4V
- 入力 ON 電流: 1.0mA 以上
- 入力 OFF 電流: 0.2mA 以下
- 絶縁耐圧: 2kV

端子台を実装しています。ねじサイズに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのプラスねじです。

接続可能な電線については、「表 3.30. 接続可能な電線(DI)」をご確認ください。

表 3.30 接続可能な電線(DI)

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。



振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。

電線の先端に予備半田しないでください。正しい接続ができなくなります。

端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

表 3.31 接点入力(CON6) 信号配列

ピン番号	ピン名	I/O	説明
3	COM	In	接点入力プラスコモン
4	DI1	In	接点入力 1
5	DI2	In	接点入力 2

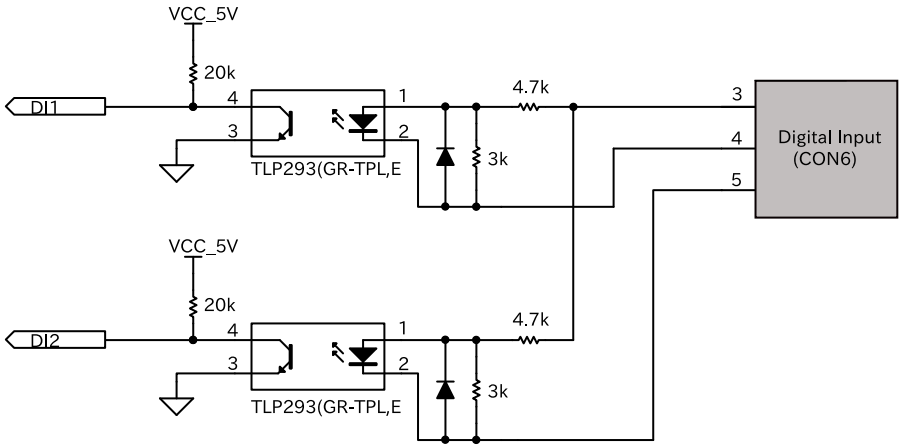


図 3.131 接点入力(CON6) 内部回路

ソフトウェアからは GPIO として制御可能です。接点入力に対応する GPIO 番号を以下に示します。

表 3.32 接点入力に対応する GPIO 番号

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
4	DI1	DI1	gpiochip5	0
5	DI2	DI2	gpiochip5	1

3.7.11.2. 接点入力(CON22)

接点入力部はフォトカプラによる絶縁入力(電流シンク出力タイプに接続可能)となっています。入力部を駆動するために電源は、外部から供給する必要があります。

機能
- 接点入力 x 8
- 入力インピーダンス: 4.7kΩ
- 定格電圧: DC 8～26.4V

- 入力 ON 電流: 1.0mA 以上
- 入力 OFF 電流: 0.2mA 以下
- 絶縁耐圧: 2kV

端子台を実装しています。ねじサイズに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのマイナスねじです。接続可能な電線については、「表 3.30. 接続可能な電線(DI)」をご確認ください。

表 3.33 接続可能な電線(DI)

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。



振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。



電線の先端に予備半田しないでください。正しい接続ができなくなります。



端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

表 3.34 接点入力(CON22) 信号配列

ピン番号	ピン名	I/O	説明
1	DI3	In	接点入力 3
2	DI4	In	接点入力 4
3	DI5	In	接点入力 5
4	DI6	In	接点入力 6
5	DI7	In	接点入力 7
6	DI8	In	接点入力 8
7	DI9	In	接点入力 9
8	DI10	In	接点入力 10
9	COM	In	接点入力プラスコモン

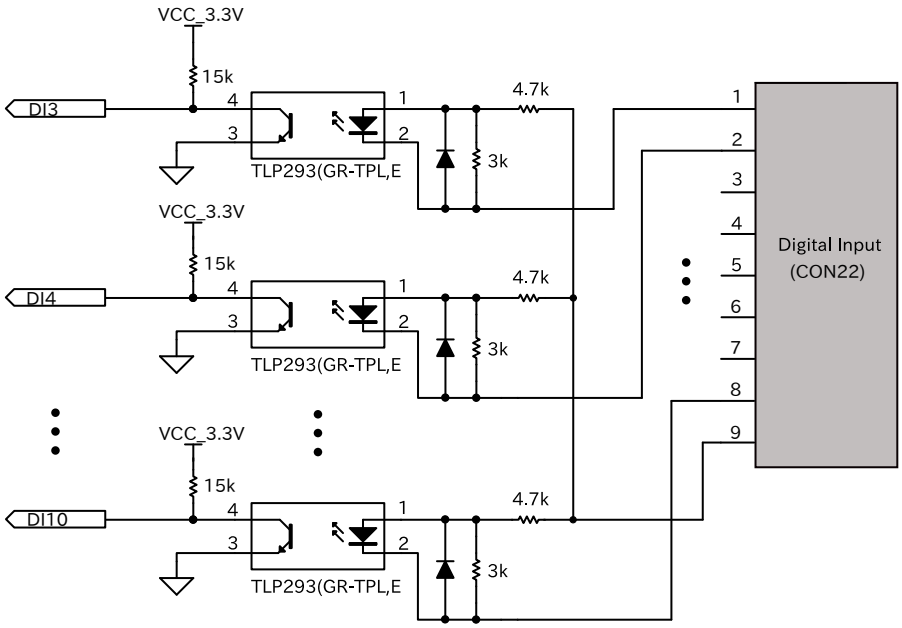


図 3.132 接点入力(CON22) 内部回路

ソフトウェアからは GPIO として制御可能です。接点入力に対応する GPIO 番号は以下のとおりです。

表 3.35 接点入力に対応する GPIO 番号

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
1	DI3	DI3	gpiochip6	0
2	DI4	DI4	gpiochip6	1
3	DI5	DI5	gpiochip6	2
4	DI6	DI6	gpiochip6	3
5	DI7	DI7	gpiochip6	4
6	DI8	DI8	gpiochip6	5
7	DI9	DI9	gpiochip6	6
8	DI10	DI10	gpiochip6	7



+Di8+Ai4 拡張基板をカスケード接続していた場合、接点入力の仕様は「表 3.34. 接点入力(CON22) 信号配列」と同様です。

N 枚目のピン番号 M の GPIO 名は DI に 2+M+8*(N-1)を付加した文字列となります。例えば、2 枚目のピン番号 1 番の GPIO 名は DI11 です。

また、それぞれの接点入力の GPIO チップは `gpiochip7` 、 `gpiochip8` 、 `gpiochip9` となります。

3.7.11.3. 入力レベルの確認

`gpioget` コマンドで入力レベルを確認できます。"0"は LOW レベル、"1"は HIGH レベルを表します。

```
[armadillo ~]# gpioget --numeric DI3 ❶
0 ❷
```

図 3.133 入力レベルの確認

❶ GPIO 名 DI3 の値を取得します。

❷ 取得した値を表示します。



接点入力に何も接続していない(開放状態)場合、取得できる入力レベルは "1" (HIGH レベル)となります。

3.7.11.4. 接点入力と接点出力をループバックして動作確認する

接点入力と接点出力をループバックして、動作確認ができます。VIN と COM、DI1 と DO1A、DO1B と GND をそれぞれ接続することで、DI1、DO1 をループバックできます。

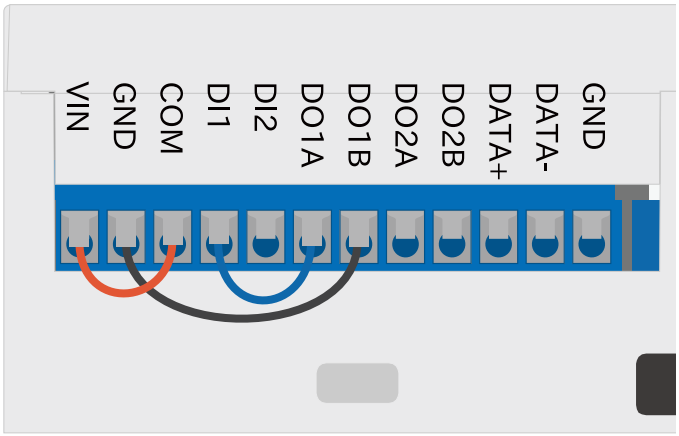


図 3.134 接点入力と接点出力をループバックする接続

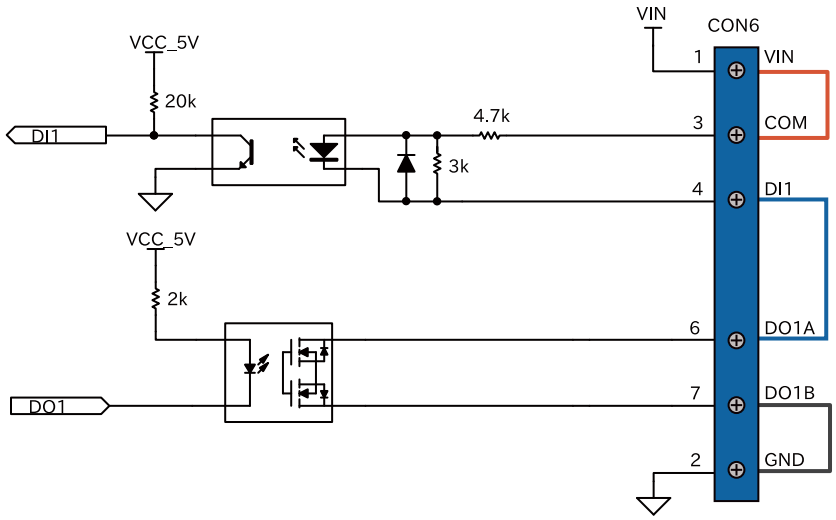


図 3.135 接点入力と接点出力をループバックする際の内部回路

```
[armadillo ~]# gpioget --numeric DI1
0
[armadillo ~]# gpioset -t0 DO1=1 ❶
[armadillo ~]# gpioget --numeric DI1
1 ❷
```

図 3.136 DI1、DO1 をループバックした場合のコマンド実行例

❶ DO1 の出力レベルを "1" に設定します。

❷ DI1 の入力レベルが "1" に変化します。

3.7.11.5. 接点入力をコンテナで使用する

コンテナ内で動作するアプリケーションから接点入力(GPIO)を使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/gpiochipN を渡すことで、GPIO**N+1** を操作できます。

alpine ベースのコンテナを作成し、その中で接点入力を操作する例を紹介します。

/etc/atmark/containers/di_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは gpiochip5 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/di_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/gpiochip5
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start di_example
```

```
Starting 'di_example'
956a0fecc48d5ea1210069910f7bb48b9e90b2dadb12895064d9776dae0360b5
```

図 3.137 接点入力を使用するコンテナの作成例

コンテナ内に入り、必要なソフトウェアをインストールして、接点入力を操作します。

```
[armadillo ~]# podman exec -it di_example sh
[container ~]# apk upgrade
[container ~]# apk add libgpiod
[container ~]# gpioget --numeric DI3
0
```

図 3.138 接点入力を操作する例

C 言語プログラムから操作する場合は、GPIO 操作ライブラリである libgpiod を使用することができます。

3.7.12. 接点出力を使用する

2 ポートの接点出力(CON6)を使用できます。

3.7.12.1. 接点出力(CON6)

接点出力部はフォトリレーによる絶縁出力(無極性)となっています。出力部を駆動するためには外部に電源が必要となります。出力 1 点につき最大電流 500mA(定格 48V)まで駆動可能です。

機能
- 絶縁出力 x 2
- 定格電圧: 最大 48V
- 定格電流: 最大 500mA
- 応答時間: 2ms 以内
- 出力形式: 無極性
- 絶縁耐圧: 2kV

端子台を実装しています。ねじサイズに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのプラスねじです。

接続可能な電線については、「表 3.36. 接続可能な電線(DO)」をご確認ください。

表 3.36 接続可能な電線(DO)

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。

振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。

電線の先端に予備半田しないでください。正しい接続ができなくなります。

端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

表 3.37 接点出力(CON6) 信号配列

ピン番号	ピン名	I/O	説明
6	DO1A	-	接点出力 1A
7	DO1B	-	接点出力 1B
8	DO2A	-	接点出力 2A
9	DO2B	-	接点出力 2B

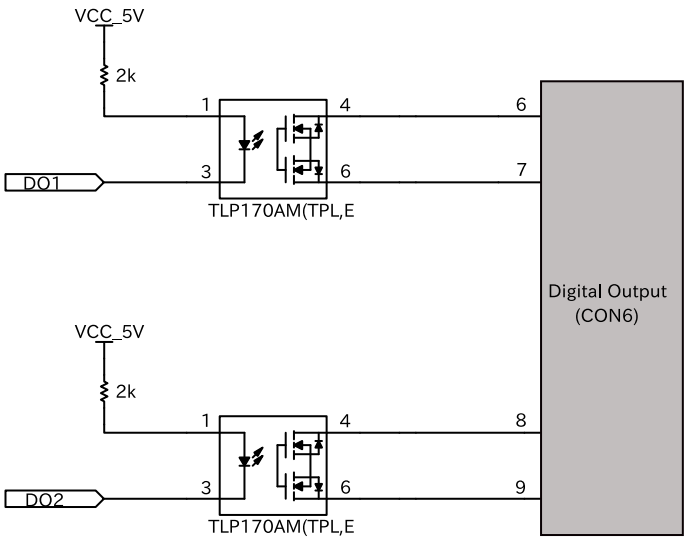


図 3.139 接点出力(CON6) 内部回路

ソフトウェアからは GPIO として制御可能です。対応する GPIO 名を次に示します。

表 3.38 接点出力に対応する CON6 ピン番号

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
6 - 7	DO1A - DO1B	DO1	gpiochip5	2

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
8 - 9	DO2A - DO2B	DO2	gpiochip5	3

3.7.12.2. 出力レベルの設定

gpioset コマンドで出力レベルを設定できます。出力レベルには "0" または "1" を設定します。"0"は LOW レベル、"1"は HIGH レベルを表します。

```
[armadillo ~]# gpioset -t0 DO1=0
```

図 3.140 出力レベルを "0" に設定する例

3.7.12.3. 接点入力と接点出力をループバックして動作確認する

接点入力と接点出力をループバックして、動作確認ができます。VIN と COM、DI1 と DO1A、DO1B と GND をそれぞれ接続することで、DI1、DO1 をループバックできます。

ループバックの方法については、「3.7.11.4. 接点入力と接点出力をループバックして動作確認する」を参照してください。

3.7.12.4. 接点出力をコンテナで使用する

コンテナ内で動作するアプリケーションから接点出力(GPIO)を使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上の /dev/gpiochipN を渡すことで、GPIO**N+1** を操作できます。

alpine ベースのコンテナを作成し、その中で接点出力を操作する例を紹介します。

/etc/atmark/containers/do_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは gpiochip5 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/do_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/gpiochip5
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start do_example
Starting 'gpio_example'
956a0fecc48d5ea1210069910f7bb48b9e90b2dadb12895064d9776dae0360b5
```

図 3.141 接点出力を使用するためのコンテナの作成例

コンテナ内に入り、必要なソフトウェアをインストールして、接点出力を操作します。

```
[armadillo ~]# podman exec -it gpio_example sh
[container ~]# apk upgrade
```

```
[container ~]# apk add libgpiod
[container ~]# gpioset -t0 DO1=0 ❶
```

図 3.142 コンテナ内からコマンドで接点出力を操作する例

❶ GPIO 名 DO1 の値を LOW に設定します。

C 言語プログラムから操作する場合は、GPIO 操作ライブラリである libgpiod を使用することができます。

3.7.13. アナログ入力を使用する

スタンダードタイプの場合、拡張インターフェース(CON8)でアナログ入力を最大 4 ポートまで拡張できます。[8]

+Di8+Ai4 タイプの場合、4 ポートのアナログ入力(CON22)を使用できます。

3.7.13.1. アナログ入力(CON8)

拡張インターフェース(CON8)でアナログ入力として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex] をご確認ください。i.MX6ULL の Analog-to-Digital Converter(ADC)に接続されたピンを使用可能です。

機能
- 分解能: 最大 12 ビット
- サンプルレート: 最大 1M サンプル/秒
- 測定電圧範囲: 0〜3.3V

3.7.13.2. アナログ入力(CON21)

アナログ入力(CON21)の信号線は A/D コンバーター(ADS1115/Texas Instruments)に接続されています。A/D コンバーターは i.MX6ULL の I2C4 バスに接続されており、I2C アドレスは 0x49 です。1〜5V および 4〜20mA(2 線式、4 線式)の信号に対応しています。接続方法については、「図 3.144. アナログ入力(CON21) 接続例」をご確認ください。

アナログ入力には端子台を実装しています。ねじサイズに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのマイナスねじです。

接続可能な電線については、「表 3.39. アナログ入力(CON21) 接続可能な電線」をご確認ください。

表 3.39 アナログ入力(CON21) 接続可能な電線

規格	UL	IEC
電線範囲	26〜18 AWG	0.12〜0.9mm^2
被覆剥き長さ	5〜6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。

[8]+Di8+Ai4 タイプの場合、拡張インターフェース(CON8)に+Di8+Ai4 拡張ボードが接続されており、拡張できません。



振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。



電線の先端に予備半田しないでください。正しい接続ができなくなります。



端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

機能
- 電圧入力(1-5V)
- 電流入力(4-20mA)
- 分解能: 12 ビット

設定値
- ゲイン: 6.144V
- サンプルレート: 250 サンプル/秒 (SPS)



A/D コンバーター自体は 16 ビットの分解能を持っていますが、ノイズなどの影響を考慮すると、実際に安定して使える精度は 12 ビット程度になります。

表 3.40 アナログ入力(CON21) 信号配列

ピン番号	ピン名	I/O	説明
1	AI1A	In	アナログ入力 1A
2	AI1B	In	アナログ入力 1B
3	AGND	Power	アナログ電源(GND)
4	AI2A	In	アナログ入力 2A
5	AI2B	In	アナログ入力 2B
6	AGND	Power	アナログ電源(GND)
7	AI3A	In	アナログ入力 3A
8	AI3B	In	アナログ入力 3B
9	AGND	Power	アナログ電源(GND)
10	AI4A	In	アナログ入力 4A
11	AI4B	In	アナログ入力 4B
12	AGND	Power	アナログ電源(GND)

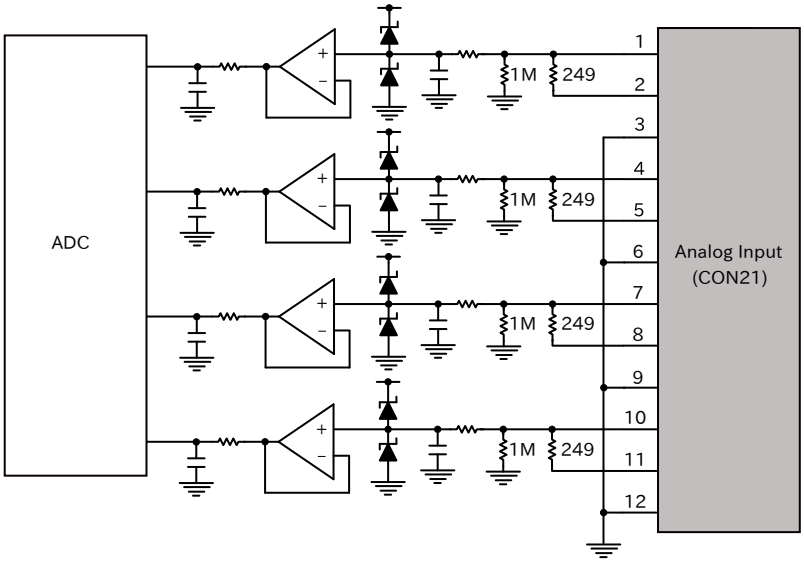


図 3.143 アナログ入力(CON21) 内部回路

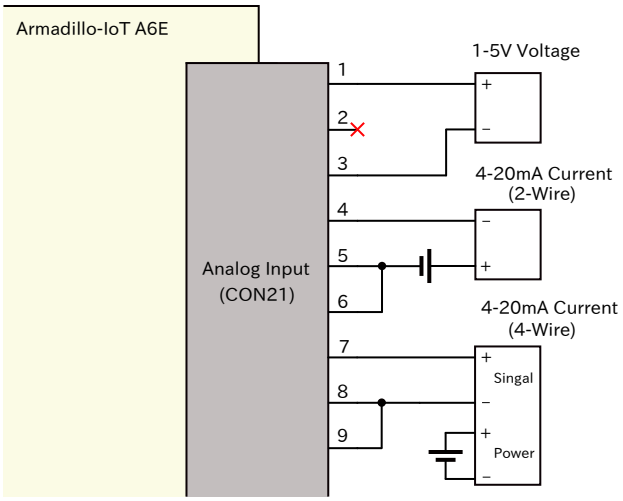


図 3.144 アナログ入力(CON21) 接続例



Armadillo-IoT ゲートウェイ A6E に電源が入っていない状態で、アナログ入力インターフェースに、電圧または電流を印加しないでください。内部回路が故障する可能性があります。

3.7.13.3. アナログ入力をコンテナで使用する

コンテナ内で動作するアプリケーションからアナログ入力を使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上の /sys ディレクトリを渡します。

alpine ベースのコンテナを作成し、その中でアナログ入力を使用する例を紹介します。

/etc/atmark/containers/ain_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 /sys を渡します。渡された /sys ディレクトリはコンテナ内の /sys にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/ain_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /sys
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start ain_example
Starting 'ain_example'
c770f76d7714f5cceb1229be2240382bded236c4a51bb69806bc0098c2cb987c
```

図 3.145 アナログ入力を扱うためのコンテナ作成例

コンテナ内に入り、該当するデバイスのパスを探します。

/sys/bus/iio/devices/iio:deviceX/label が di8ai4-X と設定されているデバイスがアナログ入力デバイスのパスです。X[X=0,1,2...] はデバイスの認識順序で変化することがありますので、Armadillo 起動後に label を確認して判別してください。

iio:device1 の label が di8ai4-1 の場合、以下のように確認できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device1/label
di8ai4-1
```

図 3.146 アナログ入力デバイスのラベル名の確認



Armadillo Base OS 3.19.1-at.2 以前のバージョンをご利用の場合は、label が存在しません。/sys/bus/iio/devices/iio:deviceX/name が ads1015 と設定されているデバイスがアナログ入力デバイスのパスです。X[X=0,1,2...] はデバイスの認識順序で変化することがありますので、Armadillo 起動後に name を確認して判別してください。

iio:device1 の name が ads1015 の場合、以下のように確認できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device1/name
ads1015
```

図 3.147 アナログ入力デバイス名の確認

ハードウェアは ADS1115 ですが、/sys/bus/iio/devices/iio:deviceX/name で表示される名称は ads1015 であることにご注意ください。

/sys/bus/iio/devices/iio:deviceX/in_voltageN_raw [N=0,1,2,3] が電圧値、/sys/bus/iio/devices/iio:deviceX/in_voltageN_scale [N=0,1,2,3] が分解能です。また、基板上の AI1 が N=0 、AI2 が N=1 、AI3 が N=2 、AI4 が N=3 です。例えば、基板上 AI2 の値は /sys/bus/iio/devices/iio:deviceX/in_voltage1_raw から取得できます。

raw を取得することで、現在の電圧値を取得できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device1/in_voltage0_raw
13293
```

図 3.148 アナログ入力 raw の取得例

scale を取得することで、分解能を取得できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device1/in_voltage0_scale
0.187500000
```

図 3.149 アナログ入力 scale の取得例

/sys/bus/iio/devices/iio:deviceX/in_voltageN_raw に /sys/bus/iio/devices/iio:deviceX/in_voltageN_scale を掛けると現在の入力電圧 (mV) が計測できます。上記の場合、13,293 * 0.1875 = 約 2,492 mV となります。

また、入力電圧を実装されている抵抗値 249 (Ω) で除算することで入力電流 (mA) を算出できます。上記の場合、 2,492 mV / 249 Ω = 約 10 mA となります。

3.7.14. I2C デバイスを使用する

Armadillo-IoT ゲートウェイ A6E 上には、複数の I2C デバイスを搭載しています。また、拡張インターフェース(CON8)で、i.MX6ULL の I2C Controller(I2C)を利用した I2C バスを 1 ポートまで拡張できます。[9]

3.7.14.1. Armadillo-IoT ゲートウェイ A6E の I2C デバイス

Armadillo-IoT ゲートウェイ A6E 上に搭載している I2C デバイスは以下のとおりです。

表 3.41 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
0(I2C1)	0x08	PF3000 (PMIC)
1(I2C2)	0x20	TCA9534 (GPIO エキスパンダー)
	0x32	RV8803 (RTC)
	0x48	SE050(セキュアエレメント)
	0x54	ADC101C021CIMK/NOPB (A/D コンバーター) [a]
3(I2C4) [b]	0x27	PI4IOE5V9554ZHEX (GPIO エキスパンダー)
	0x49	ADS1115 (A/D コンバーター)
	0x57	RM24C01-RDW6TP (EEPROM)

[a]Cat.1 bis モデルのみ
[b]+Di8+Ai4 タイプのみ

I2C デバイスは、I2C バスを介して接続されており、I2C バスのスレーブアドレスを指定することで制御できます。

[9]+Di8+Ai4 タイプの場合、拡張インターフェース(CON8)に+Di8+Ai4 拡張ボードが接続されており、拡張できません。

Armadillo-IoT ゲートウェイ A6E の標準状態では、CONFIG_I2C_CHARDEV が有効となっているためユーザードライバで I2C デバイスを制御できます。ユーザードライバを利用する場合は、Linux カーネルで I2C デバイスに対応するデバイスドライバを無効にしてください。

デバイスファイル
- /dev/i2c-0 (I2C1)
- /dev/i2c-1 (I2C2)
- /dev/i2c-3 (I2C4)

3.7.14.2. 拡張インターフェース I2C(CON8)

拡張インターフェース(CON8)で I2C として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex]をご確認ください。i.MX6ULL の I2C Controller(I2C4)に接続されたピンを使用可能です。

機能
- 最大転送レート: 384kbps
- 信号レベル: VCC_3.3V

デバイスファイル
- /dev/i2c-3 (I2C4)



I2C バスが足りない場合、 i2c-gpio を利用した GPIO を I2C バスとして拡張することができます。

3.7.14.3. コンテナで I2C デバイスを使用する

コンテナ内で動作するアプリケーションから I2C デバイスを使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/i2c-N を渡します。

alpine ベースのコンテナを作成し、その中で I2C デバイスを操作する例を紹介します。

/etc/atmark/containers/i2c_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 /dev/i2c-1 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/i2c_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/i2c-1
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start i2c_example
```

```
Starting 'i2c_example'
efa1eb129c1f036a709755f0d53b21a0f2a39307ecae32b24aac98c0b6567bf0
```

図 3.150 I2C を使用するコンテナの作成例

コンテナ内に入り、必要なソフトウェアをインストールします。i2c-tools に含まれる i2cdetect コマンドを使って I2C デバイスのスレーブアドレスを確認できます。

```
[armadillo ~]# podman exec -it i2c_example sh
[container ~]# apk upgrade
[container ~]# apk add i2c-tools
[container ~]# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- UU -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- UU -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- 72 -- -- -- -- --
```

図 3.151 I2C デバイスのスレーブアドレスの確認例

3.7.15. リアルタイムクロックを使用する

温度保証タイプのリアルタイムクロック(RV-8803-C7/Micro Crystal)を搭載しています。i.MX6ULL のリアルタイムクロック機能もありますが、精度が高くバックアップ可能な RV-8803-C7 側の使用をおすすめします。

本書では i.MX6ULL のリアルタイムクロック説明はしません。詳細につきましては、NXP Semiconductors のホームページからダウンロード可能な『i.MX 6ULL Applications Processor Reference Manual』をご参照ください。

3.7.15.1. リアルタイムクロック(RV-8803-C7)

温度保証タイプのリアルタイムクロックです。信号線は i.MX6ULL の I2C Controller(I2C2)に接続されています。

RTC バックアップインターフェース(CON10)により、電源が切断されても時刻データを保持できます。時刻データを保持したい場合は、電池を入れて使用してください。

リアルタイムクロックの時刻保持時の平均消費電流は、240nA(Typ.)のため、電池寿命までの時刻保持が期待できます。

最大月差は周囲温度-20℃～60℃で 8 秒です。(経年変化を除く)

機能	・アラーム割り込みサポート
デバイスファイル	・/dev/rtc (/dev/rtc0 へのシンボリックリンク) ・/dev/rtc0 (RV-8803-C7) ・/dev/rtc1 (i.MX6ULL SNVS_HP Real Time Counter)



RV-8803-C7 が /dev/rtc0 となるよう、Device Tree でエイリアスを設定しているため、i.MX6ULL の RTC 機能は /dev/rtc1 となります。エイリアスの設定は、 arch/arm/boot/dts/armadillo-iotg-a6e.dts で行っています。



アラーム割り込みは、デバイスファイル経由で利用することができます。

詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/admin-guide/rtc.rst)やサンプルプログラム(tools/testing/selftests/rtc/rtctest.c)を参照してください。

3.7.15.2. RTC バックアップインターフェース(CON10)

リアルタイムクロック(RV-8803-C7)のバックアップ用のインターフェースです。電源が切断されても時刻データを保持したい場合は、電池を入れてご使用ください。CR1220、BR1220 等の電池を接続できます。

リアルタイムクロックの時刻保持時の平均消費電流は、240nA(Typ.)のため、電池寿命までの時刻保持が期待できます。

表 3.42 RTC バックアップインターフェース(CON10) 信号配列

ピン番号	ピン名	I/O	説明
1	RTC_BAT	Power	リアルタイムクロックのバックアップ用電源入力(RTC_BAT)
2	GND	Power	電源(GND)



電池をホルダーへ装着する際は、異物の挟み込みや不完全な装着がないように、目視での異物確認や装着状態の確認を行ってください。

3.7.15.3. リアルタイムクロックに時刻を設定する

Linux の時刻には、Linux カーネルが管理するシステムクロックと、リアルタイムクロックが管理するハードウェアクロックの 2 種類があります。リアルタイムクロックに時刻を設定するには、まずシステムクロックを設定します。その後、ハードウェアクロックをシステムクロックと一致させます。

システムクロックは、 date コマンドを使用して設定します。date コマンドの引数には、設定する時刻を[MMDDhhmmCCYY.ss] というフォーマットで指定します。時刻フォーマットの各フィールドの意味を以下に示します。

表 3.43 時刻フォーマットのフィールド

フィールド	意味
MM	月
DD	日(月内通算)
hh	時
mm	分

フィールド	意味
CC	年の最初の 2 桁(省略可)
YY	年の最後の 2 桁(省略可)
ss	秒(省略可)

2023 年 3 月 2 日 12 時 34 分 56 秒に設定する場合は、次のように指定します。

```
[armadillo ~]# date
Sat Jan  1 09:00:00 JST 2000
[armadillo ~]# date 030212342023.56
Fri Mar  2 12:34:56 JST 2023
[armadillo ~]# date
Fri Mar  2 12:34:57 JST 2023
```

図 3.152 システムクロックを設定

システムクロックを設定した後、ハードウェアクロックを hwclock コマンドで設定します。

```
[armadillo ~]# hwclock ❶
2000-01-01 00:00:00.000000+09:00
[armadillo ~]# hwclock --utc --systohc ❷
[armadillo ~]# hwclock --utc ❸
2023-03-02 12:57:20.534140+09:00
```

図 3.153 ハードウェアクロックを設定

❶ 現在のハードウェアクロックを表示します。

❷ ハードウェアクロックを協定世界時(UTC)で設定します。

❸ ハードウェアクロックが UTC で正しく設定されていることを確認します。



インターネットに接続している場合は、chronyd により自動的に日時設定が行われるため、手動で設定する必要はありません。

3.7.15.4. リアルタイムクロックをコンテナで使用する

コンテナ内からリアルタイムクロックを使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/rtc0 を渡し、リアルタイムクロックの時刻の設定を行う権限を付与します。

alpine ベースのコンテナを作成し、その中でリアルタイムクロックを操作する例を紹介します。

/etc/atmark/containers/rtc_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 /dev/rtc0 を渡し、権限として SYS_TIME を付与します。

```
[armadillo ~]# vi /etc/atmark/containers/rtc_example.conf
set_image docker.io/alpine
```

```
set_command sleep infinity
add_args --cap-add=SYS_TIME
add_devices /dev/rtc0
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start rtc_example
Starting 'rtc_example'
025209e0d96f43c2911239a8397b7002c3eaab057e031d8abb765df5707d75bd
```

図 3.154 リアルタイムクロックを使用するコンテナの作成例

コンテナ内に入り、 hwclock コマンドでリアルタイムクロックの時刻表示と設定を行います。

```
[armadillo ~]# podman exec -it rtc_example sh
[container ~]# hwclock ❶
Thu Feb 18 05:14:37 2021  0.000000 seconds
[container ~]# date --set "2021-04-01 09:00:00" ❷
Thu Apr  1 09:00:00 UTC 2021
[container ~]# hwclock --systohc ❸
[container ~]# hwclock ❹
Thu Apr  1 09:00:28 2021  0.000000 seconds
```

図 3.155 リアルタイムクロックの時刻表示と設定例

❶ リアルタイムクロックに設定されている現在時刻を表示します。

❷ システム時刻を 2021 年 4 月 1 日 9 時 0 分 0 秒に設定します。

❸ システム時刻をリアルタイムクロックに反映させます。

❹ リアルタイムクロックに設定されている時刻が変更されていることを確認します。

3.7.16. ユーザースイッチを使用する

ユーザーが自由に利用できる押しボタンスイッチを 1 つ搭載しています。

3.7.16.1. ユーザースイッチ(SW1)

ユーザースイッチ(SW1)の信号線は i.MX6ULL の General Purpose Input/Output(GPIO)に接続されています。

表 3.44 ユーザースイッチ(SW1) 信号配列

部品番号	名称	説明
SW1	ユーザースイッチ	i.MX6ULL の JTAG_MOD ピンに接続 (Low: 押されていない状態、High: 押された状態)

Linux ではユーザー空間でイベント(Press/Release)を検出することができ、GPIO 接続用キーボードドライバ(gpio-keys)で制御することができます。

ユーザースイッチと信号には、次に示すキーコードが割り当てられています。

表 3.45 インプットデバイスファイルとイベントコード

ユーザースイッチ	インプットデバイスファイル	イベントコード
SW1	/dev/input/by-path/platform-gpio-keys-event	148 (KEY_PROG1)



インプットデバイスは検出された順番にインデックスが割り振られます。USB デバイスなどを接続してインプットデバイスを追加している場合は、デバイスファイルのインデックスが異なる可能性があります。

3.7.16.2. ユーザースイッチをコンテナで使用する

スイッチのプッシュ/リリースイベントを取得するには、Podman のイメージからコンテナを作成する際に、ABOS 上の /dev/input ディレクトリを渡します。

alpine ベースのコンテナを作成し、その中でスイッチのイベントを取得する例を紹介します。

/etc/atmark/containers/sw_example.conf というファイルを以下の内容で作成し、コンテナを起動します。/dev/input ディレクトリを渡すと、渡された /dev/input ディレクトリはコンテナ内の /dev/input にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/sw_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/input
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start sw_example
Starting 'sw_example'
c0cd8b801883266197a3c20552b0e8b6c7dd473bb0b24e05bf3ecdb581c822b9
```

図 3.156 ユーザースイッチのイベントを取得するコンテナの作成例

コンテナ内に入り、 evtest コマンドでイベントを確認します。

```
[armadillo ~]# podman exec -it sw_example sh
[container ~]# apk upgrade
[container ~]# apk add evtest
[container ~]# evtest /dev/input/event1
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 28 (KEY_ENTER)
Properties:
Testing ... (interrupt to exit)
Event: time 1612849227.554456, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1 ❶
```

```
Event: time 1612849227.554456, ---------- SYN_REPORT ------------
Event: time 1612849229.894444, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0 ❷
Event: time 1612849229.894444, ---------- SYN_REPORT ------------
```

図 3.157 evtest コマンドによる確認例

❶ SW1 のボタン **プッシュ** イベントを検出したときの表示

❷ SW1 のボタン **リリース** イベントを検出したときの表示



ABOS では、スイッチの制御を簡単に実装できる **buttond** デーモンを用意しています。詳細は「6.21. ボタンやキーを扱う」 をご確認ください。

3.7.17. LED を使用する

緑色 LED が 3 つ搭載されています。

3.7.17.1. LED(SYS、APP、WWAN)

SYS、APP、WWAN の 3 つの LED は、i.MX6ULL の General Purpose Input/Output(GPIO)に接続されています。それぞれ、電源の入力状態やシステム状態、アプリケーション状態の表示に使用しています。

GPIO 接続用 LED ドライバ(leds-gpio)で制御することができます。

表 3.46 LED の状態表示

LED 状態\LED 名称	SYS	APP	WWAN
OFF	電源 OFF	アプリ起動不可	SIM 未検出または認識中、または LTE モデム未検出
ON	電源 ON	アプリ起動可能	LTE 接続済み
Blink Slow	シャットダウン中	アプリ起動完了 [a]	SIM 検出、LTE 未接続 [b]
Blink Fast	アップデート中	アプリエラー [a]	SIM 検出、LTE 未接続 [b]、電波品質が低い

[a]APP LED の「起動完了」と「エラー」の点滅動作は、アプリ自身が行います。
[b]LTE コネクションが未作成、設定間違いの場合もこの状態となります。

sysfs LED クラスディレクトリ
- /sys/class/leds/app
- /sys/class/leds/sys
- /sys/class/leds/wwan

表 3.47 LED の接続

部品番号	名称(色)	説明
SYS	システム LED(緑)	電源(VCC_3.3V)の入力状態を表示、i.MX6ULL の UART2_CTS_B ピン(GPIO1_IO22)に接続 (Low: 消灯、High: 点灯)

部品番号	名称(色)	説明
APP	アプリケーション LED(緑)	アプリケーションの状態を表示、i.MX6ULL の UART2_RTS_B ピン(GPIO1_IO23)に接続 (Low: 消灯、High: 点灯)
WWAN	ワイヤレス WAN LED(緑)	LTE 通信の状態を表示、i.MX6ULL の UART1_RX_DATA ピン(GPIO1_IO17)に接続 (Low: 消灯、High: 点灯)



WLAN/LAN モデルでは WWAN LED を自由に使用することができます。

3.7.17.2. LED を点灯/消灯する

LED クラスディレクトリ以下の brightness ファイルへ値を書き込むことによって、LED の点灯/消灯を行うことができます。brightness ファイルに 0 を書き込むと消灯、1~255 の値を書き込むと点灯します。

ここでは、任意の LED を示す LED クラスディレクトリを /sys/class/leds/[LED]/ のように表記します。



Armadillo-IoT ゲートウェイ A6E の LED には輝度制御の機能がないため、0(消灯)、1~255(点灯)の 2 つの状態のみ指定することができます。

brightness ファイルに 0 を書き込むと消灯します。

```
[armadillo ~]# echo 0 > /sys/class/leds/[LED]/brightness
```

図 3.158 LED を消灯させる

brightness ファイルに 1 を書き込むと点灯します。

```
[armadillo ~]# echo 1 > /sys/class/leds/[LED]/brightness
```

図 3.159 LED を点灯させる

brightness を読み出すと LED の状態が取得できます。

```
[armadillo ~]# cat /sys/class/leds/[LED]/brightness
```

図 3.160 LED の状態を表示する

3.7.17.3. LED のトリガを使用する

Linux では、LED をある特定のタイミングで光らせることができます。これを「トリガ」と呼びます。LED クラスディレクトリ以下の trigger ファイルへ値を書き込むことで、LED の点灯/消灯にトリガを設定できます。 trigger でサポートされている値は以下の通りです。

表 3.48 LED トリガの種類

設定	説明
none	トリガを設定しません。
mmc0	eMMC のアクセスランプにします。
mmc1	microSD スロットのアクセスランプにします。
timer	任意のタイミングで点灯/消灯を行います。この設定にすると、LED クラスディレクトリ以下に delay_on 、 delay_off ファイルが出現し、それぞれ点灯時間, 消灯時間をミリ秒単位で指定します。
heartbeat	心拍のように点灯/消灯を行います
default-on	主に Linux カーネルから使用します。LED が点灯します。

trigger ファイルを読み出すとサポートしているトリガと、現在有効のトリガが表示されます。 [] が付いているものが現在のトリガです。ここでは、任意の LED を示す LED クラスディレクトリを /sys/class/leds/[LED]/ のように表記します。

```
[armadillo ~]# cat /sys/class/leds/[LED]/trigger
[none] rc-feedback bluetooth-power rfkill-any rfkill-none kbd-scrolllock kbd-numlock kbd-capslock ⏎
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock ⏎
kbd-ctrllock kbd-ctrlrlock rfkill0 rfkill1 timer oneshot heartbeat backlight gpio default-on mmc0 ⏎
mmc1
```

図 3.161 対応している LED トリガを表示

以下のコマンドを実行すると、LED が 2 秒点灯、1 秒消灯を繰り返します。

```
[armadillo ~]# echo timer > /sys/class/leds/[LED]/trigger
[armadillo ~]# echo 2000 > /sys/class/leds/[LED]/delay_on
[armadillo ~]# echo 1000 > /sys/class/leds/[LED]/delay_off
```

図 3.162 LED のトリガに timer を指定する

3.7.17.4. LED をコンテナで使用する

LED をコンテナ内で使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上の /sys ディレクトリを渡す必要があります。

alpine ベースのコンテナを作成し、その中で LED を使用します。

/etc/atmark/containers/led_example.conf というファイルを以下の内容で作成し、コンテナを起動します。/sys ディレクトリはコンテナ内の /sys にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/led_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /sys
```

```
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start led_example
Starting 'led_example'
c770f76d7714f4cceb1229be2240382bded236c2c51bb6b866bc0098c2cb987a
```

図 3.163 LED を使用するコンテナの作成例

コンテナ内に入り、LED の点灯/消灯を行います。ここでは、 /sys/class/leds/app ディレクトリを使用します。brightness ファイルに 0 を書き込むと消灯、1~255 の値を書き込むと点灯します。

```
[armadillo ~]# podman exec -it led_example sh
[container ~]# echo 0 > /sys/class/leds/app/brightness
[container ~]# echo 1 > /sys/class/leds/app/brightness
```

図 3.164 LED の点灯/消灯の例

3.7.18. Bluetooth® を使用する

Cat.1 bis+WLAN モデルおよび WLAN モデルには、WLAN+BT コンボモジュール(Sterling LWB5+/Ezurio)が搭載されています。

Bluetooth® を使用する際は、WLAN/BT アンテナインターフェース(ANT3)にアンテナを接続してください。アンテナが接続されていない場合は、「3.6. WLAN アンテナの取り付けと取り外し」を参照し、アンテナを取り付けてください。

3.7.18.1. WLAN+BT コンボモジュール(Bluetooth®)

WLAN+BT コンボモジュールの Bluetooth® の信号線は i.MX6ULL の Universal Asynchronous Receiver/Transmitter(UART2)に接続されています。

デバイスファイル	・ hci0



WLAN+BT コンボモジュールの技適認証取得済みのアンテナについて抜粋したリストを Armadillo サイト [https://armadillo.atmark-techno.com/]で公開しています。付属のアンテナ以外をご検討の際に、ご活用ください。

当社にて全てのアンテナの動作を確認したものではありませんので、通信性能の評価については、ユーザー様自身にて実施いただくようお願いいたします。



アンテナインターフェース(ANT2)に WLAN+BT コンボモジュール用のアンテナを接続するカスタマイズ品を製造することが可能です。詳細につきましては、アットマークテクノ営業部までお問い合わせください。

3.7.18.2. Bluetooth® をコンテナで使用する



標準出荷状態の Armadillo Base OS には BlueZ をはじめとした Bluetooth® のプロトコルスタックがインストールされていません。動作確認時は以下の手順に従って、コンテナ内にプロトコルスタックをインストールして行ってください。

コンテナ内から Bluetooth® を使用するには、Podman のイメージからコンテナを作成する際に、ホストネットワークを使用するため、 NET_ADMIN 権限を付与します。

alpine ベースのコンテナを作成し、その中で Bluetooth® を使用する例を紹介します。

/etc/atmark/containers/bt_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/bt_example.conf
set_image docker.io/alpine
set_command sleep infinity
set_network host
add_args --cap-add=NET_ADMIN
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start bt_example
Starting 'bt_example'
45fe1eb6b25529f0c84cd4b97ca1aef8451785fc9a87a67d54873c1ed45b70a4
```

図 3.165 Bluetooth® を使用するコンテナの作成例

コンテナ内に入り、必要なソフトウェアをインストールして、Bluetooth® 機能を有効にします。

```
[armadillo ~]# podman exec -it bt_example sh
[container ~]# apk upgrade
[container ~]# apk add bluez
[container ~]# mkdir /run/dbus
[container ~]# dbus-daemon --system
[container ~]# /usr/lib/bluetooth/bluetoothd &
```

図 3.166 Bluetooth® 機能を有効にする例

bluetoothctl コマンドで Bluetooth® 機器のスキャンやペアリングなどを行えます。周辺機器をスキャンしてペアリングを行う例を示します。

```
[container ~]# bluetoothctl
Agent registered
[..CHG..] Controller XX:XX:XX:XX:XX:XX Pairable: yes
[bluetooth]# power on ❶
Changing power on succeeded
[..CHG..] Controller XX:XX:XX:XX:XX:XX Powered: yes
[bluetooth]# scan on ❷
Discovery started
[..CHG..] Controller XX:XX:XX:XX:XX:XX Discovering: yes
[..NEW..] Device AA:AA:AA:AA:AA:AA AA-AA-AA-AA-AA-AA
[..NEW..] Device BB:BB:BB:BB:BB:BB BB-BB-BB-BB-BB-BB
[..NEW..] Device CC:CC:CC:CC:CC:CC CC-CC-CC-CC-CC-CC
[..NEW..] Device DD:DD:DD:DD:DD:DD DD-DD-DD-DD-DD-DD
[..NEW..] Device EE:EE:EE:EE:EE:EE EE-EE-EE-EE-EE-EE
[bluetooth]# pair AA:AA:AA:AA:AA:AA ❸
[bluetooth]# exit ❹
[container ~]#
```

図 3.167 周辺機器のスキャンとペアリングの例

❶ コントローラを起動します。

❷ 周辺機器をスキャンします。

❸ ペアリングしたい機器の MAC アドレスを指定してペアリングします。

❹ exit で bluetoothctl のプロンプトを終了します。

3.7.19. Wi-SUN デバイスを使用する

拡張インターフェース(CON8)から UART 接続の Wi-SUN デバイスを使用できます。UART として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex] をご確認ください。

3.7.19.1. Wi-SUN デバイスをコンテナで使用する

コンテナ内で動作するアプリケーションから Wi-SUN デバイスで通信するには、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/ttymxcN のうち、Wi-SUN に対応するものを渡します。

alpine ベースのコンテナを作成し、その中で Wi-SUN デバイスで通信する例を紹介します。

/etc/atmark/containers/wisun_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 /dev/ttymxc0 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/wisun_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/ttymxc0
[armadillo ~]# podman pull docker.io/alpine
```

```
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start wisun_example
Starting 'wisun_example'
ef9a5a2f7eee4236cb28c1fbf5090a6f0db9d6dfe7f3a34573867e0833dd3122
```

図 3.168 Wi-SUN デバイスで通信するコンテナの作成例

コンテナ内に入り、 /dev/ttymxc0 を使ってデータの送受信を行います。

```
[armadillo ~]# podman exec -it wisun_example sh
[container ~]# ls /dev/ttymxc0
/dev/ttymxc0
```

図 3.169 Wi-SUN デバイスの確認例

3.7.20. EnOcean デバイスを使用する

拡張インターフェース(CON8)から UART 接続の EnOcean デバイスを使用できます。UART として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex] を ご確認ください。

3.7.20.1. EnOcean デバイスをコンテナで使用する

コンテナ内で動作するアプリケーションから EnOcean デバイスで通信するには、Podman のイメージからコンテナを作成する際に、ABOS 上のデバイスファイル /dev/ttymxcN のうち、EnOcean に対応するものを渡します。

alpine ベースのコンテナを作成し、その中で EnOcean デバイスで通信する例を紹介します。

/etc/atmark/containers/enocean_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 /dev/ttymxc0 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/enocean_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/ttymxc0
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start enocean_example
Starting 'enocean_example'
a808b491a100f9078d8c72a7b36966d9182614f3657fe054fb8d7f87b0d4b31c
```

図 3.170 EnOcean デバイスで通信するコンテナの作成例

コンテナ内に入り、 /dev/ttymxc0 を使ってデータの送受信を行います。

```
[armadillo ~]# podman exec -it enocean_example sh
[container ~]# ls /dev/ttymxc0
/dev/ttymxc0
```

図 3.171 EnOcean デバイスの確認例

3.7.21. CAN デバイスを使用する

拡張インターフェース(CON8)で CAN を最大 2 ボートまで拡張できます。CAN として使用できるピンについては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex]をご確認ください。

3.7.21.1. CAN デバイスをコンテナで使用する

コンテナ内で動作するアプリケーションから CAN 通信するには、Podman のイメージからコンテナを作成する際に、コンテナを実行するネットワークとして host を、権限として NET_ADMIN を指定します。

alpine ベースのコンテナを作成し、その中で CAN 通信する例を紹介します。

/etc/atmark/containers/can_example.conf というファイルを以下の内容で作成し、コンテナを起動します。

```
[armadillo ~]# vi /etc/atmark/containers/can_example.conf
set_image docker.io/alpine
set_command sleep infinity
set_network host
add_args --cap-add=NET_ADMIN
[armadillo:~]# podman pull docker.io/alpine:latest
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
8d591b0b7dea080ea3be9e12ae563eebf9869168ffced1cb25b2470a3d9fe15e
[armadillo ~]# podman_start can_example
Starting 'can_example'
73e7dbce86e84eef337bbc5c580a747948b94b87015bb34143da341b8301c16a
```

図 3.172 CAN 通信するコンテナの作成例

コンテナ内に入り、必要なソフトウェアをインストールします。ip コマンドで CAN の設定を行い、CAN を有効にします。

```
[armadillo ~]# podman exec -it can_example sh
[container ~]# apk upgrade
[container ~]# apk add iproute2 ❶
[container ~]# ip link set can0 type can bitrate 125000 ❷
[container ~]# ip link set can0 up ❸
[container ~]# ip -s link show can0 ❹
4: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 10
    link/can
```

```
    RX: bytes  packets  errors  dropped missed  mcast
    0          0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    0          0        0       0       0       0
```

図 3.173 CAN の設定例

❶ CAN の設定のために必要な iproute2 をインストールします。すでにインストール済みの場合は不要です。

❷ CAN の通信速度を 125kbps に設定します。

❸ can0 インターフェースを起動します。

❹ can0 インターフェースの現在の使用状況を表示します。

3.7.22. 入力電圧を監視する



スタンダードタイプ WLAN/LAN モデルの場合、A/D コンバータが搭載されておらず、入力電圧を監視できません。

Armadillo に入力されている電圧を監視することができます。バッテリー駆動時などは、入力電圧を監視することで、電源の残量や電圧低下を監視し、システムの安全な動作や電源管理に役立てることができます。

3.7.22.1. 入力電圧をコンテナで監視する

コンテナ内で動作するアプリケーションから入力電圧を監視するには、Podman のイメージからコンテナを作成する際に、ABOS 上の /sys ディレクトリを渡します。

alpine ベースのコンテナを作成し、その中で入力電圧を計測する例を紹介します。

/etc/atmark/containers/vin_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは、 /sys ディレクトリを渡します。 /sys ディレクトリは、コンテナ内の /sys にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/vin_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /sys
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start vin_example
Starting 'vin_example'
c780f76d7714f4cdeb1229be2240382bded236c2c51bd6b8469c10d822cb987e
```

図 3.174 入力電圧を監視するコンテナの作成例

コンテナ内に入り、該当するデバイスのパスを探します。

/sys/bus/iio/devices/iio:deviceX/label が vin と設定されているデバイスが A/D コンバータのパスになります。X[X=0,1,2...] はデバイスの認識順序で変化することがありますので、Armadillo 起動後に label を確認して判別してください。

iio:device5 の label が vin の場合、以下のように確認できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device5/label
vin
```

図 3.175 入力電圧監視デバイス名の確認

/sys/bus/iio/devices/iio:deviceX/in_voltage_raw が電圧値、 /sys/bus/iio/devices/iio:deviceX/in_voltage_scale が分解能です。

raw を取得することで、現在の電圧値を取得できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device5/in_voltage_raw
405
```

図 3.176 入力電圧 raw の取得例

scale を取得することで、分解能を取得できます。

```
[container ~]# cat /sys/bus/iio/devices/iio:device5/in_voltage_scale
3.222656250
```

図 3.177 入力電圧 scale の取得例

入力電圧の値は「図 3.178. 入力電圧監視計算式」の式で計算できます。raw と scale の値が上記の場合、約 12,103 mV となります。

```
計測電圧(mV) = raw * scale * (910 + 110) / 110
```

図 3.178 入力電圧監視計算式



バッテリー駆動などで入力電圧が変化し、閾値以下または以上になった際に何らかのアクションを起こしたい場合は、「6.15. 入力電圧監視サービス (power-alertd) を使用する」のご利用もご検討ください。

3.7.23. 拡張インターフェースを使用する

機能拡張用のインターフェース(CON8)が搭載されています。[10]

3.7.23.1. 拡張インターフェース(CON8)

機能拡張用のインターフェースです。複数の機能(マルチプレクス)を持つ、i.MX6ULL の信号線が接続されており、GPIO、UART、I2C、SPI、CAN、PWM 等の機能を拡張することができます。

拡張基板の設計に関しては、「3.4.4. 拡張基板の設計」も合わせてご確認ください。



Armadillo-IoT ゲートウェイ A6E が起動できなくなる恐れがありますので、CON8 のプルアップ/ダウン抵抗が基板上で接続されているピンは、電源投入時または再起動時、プル抵抗で決められた入力レベル以外にしないでください。ピンの使い方については、「3.4.4.1. 回路設計時の留意点」をご参照ください。



拡張できる機能の詳細につきましては、「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」 [https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-a6e/manual-multiplex]をご参照ください。

拡張インターフェースに搭載可能なコネクタと対向コネクタの例を以下に示します。

表 3.49 CON8 搭載コネクタと対向コネクタ例

名称	型番	メーカー	備考
搭載コネクタ	6130xx21121 [a]	Wurth Elektronik	許容電流 3A(端子 1 本あたり)
対向コネクタ	6130xx21821 [a]	Wurth Elektronik	-

[a]xx にはピン数が入ります。

表 3.50 拡張インターフェース(CON8) 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源出力(VIN)
2	GND	Power	電源(GND)
3	VCC_5V	Power	電源出力(VCC_5V)
4	GND	Power	電源(GND)
5	VCC_3.3V	Power	電源出力(VCC_3.3V)
6	GND	Power	電源(GND)
7	GPIO1_IO01	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO01 ピンに接続
8	GPIO1_IO02	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO02 ピンに接続
9	GPIO1_IO03	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO03 ピンに接続
10	GPIO1_IO04	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続
11	GND	Power	電源(GND)
12	I2C4_SCL	In/Out	拡張入出力、i.MX6ULL の I2C4_SCL ピンに接続
13	I2C4_SDA	In/Out	拡張入出力、i.MX6ULL の I2C4_SDA ピンに接続

[10]+Di8+Ai4 タイプの場合、拡張インターフェース(CON8)に+Di8+Ai4 拡張ボードが接続されています。

ピン番号	ピン名	I/O	説明
14	GPIO3_IO05	In/Out	拡張入出力、i.MX6ULL の LCD_DATA00 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
15	GPIO3_IO06	In/Out	拡張入出力、i.MX6ULL の LCD_DATA01 ピンに接続、基板上で 10kΩ プルアップ(VCC_3.3V)されています。[a]
16	GPIO3_IO07	In/Out	拡張入出力、i.MX6ULL の LCD_DATA02 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
17	GPIO3_IO08	In/Out	拡張入出力、i.MX6ULL の LCD_DATA03 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
18	GND	Power	電源(GND)
19	GPIO3_IO10	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続、起動デバイス設定スイッチ(SW2)が eMMC 側に設定されている時 10kΩ プルアップ(VCC_3.3V)、SD 側に設定されている時 10kΩ プルダウンされます。[a]
20	GPIO3_IO11	In/Out	拡張入出力、i.MX6ULL の LCD_DATA06 ピンに接続、基板上で 10kΩ プルアップ(VCC_3.3V)されています。[a]
21	GPIO3_IO12	In/Out	拡張入出力、i.MX6ULL の LCD_DATA07 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
22	GPIO3_IO13	In/Out	拡張入出力、i.MX6ULL の LCD_DATA08 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
23	GPIO3_IO14	In/Out	拡張入出力、i.MX6ULL の LCD_DATA09 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
24	GPIO3_IO15	In/Out	拡張入出力、i.MX6ULL の LCD_DATA10 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
25	GPIO3_IO16	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続、起動デバイス設定スイッチ(SW2)が eMMC 側に設定されている時 10kΩ プルダウン、SD 側に設定されている時 10kΩ プルアップ(VCC_3.3V)されます。[a]
26	GPIO3_IO20	In/Out	拡張入出力、i.MX6ULL の LCD_DATA15 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
27	GPIO3_IO21	In/Out	拡張入出力、i.MX6ULL の LCD_DATA16 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
28	GPIO3_IO22	In/Out	拡張入出力、i.MX6ULL の LCD_DATA17 ピンに接続、基板上で 10kΩ プルダウンされています。[a]
29	GPIO4_IO25	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
30	GPIO4_IO26	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
31	GPIO4_IO27	In/Out	拡張入出力、i.MX6ULL の CSI_DATA06 ピンに接続
32	GPIO4_IO28	In/Out	拡張入出力、i.MX6ULL の CSI_DATA07 ピンに接続
33	GND	Power	電源(GND)
34	GND	Power	電源(GND)

[a]本ピンを使用する場合は「3.4.4.1. 回路設計時の留意点」を参照ください。

3.7.23.2. 拡張基板で使用する各ピンの設定

拡張基板で使用する各ピンはデバイスドライバが利用する場合があるため、拡張基板の回路とピンの接続はデバイスドライバが動作を始めるよりも前に行う必要があります。

「図 3.47. 3-State バッファを使用した回路例」を使う場合の設定例を紹介します。

GPIO3_IO13 を High 出力に設定するノードを 「6.29.5. 独自の DTS overlay を追加する」 の手順で追加します。「図 3.179. GPIO3_IO13 を High 出力にするノードの例」では、以下 2 つのノードを追加しています。

- regulators に regulator-con8ioen を追加
- iomuxc に con8ioengrp を追加

```
[ATDE ~/linux-[version]]$ vi arch/arm/boot/dts/armadillo-600-customize.dts

/dts-v1/;
```

```
/plugin/;

#include <dt-bindings/gpio/gpio.h>
#include "imx6ull-pinfunc.h"

&{/} {
        regulators {
                reg_con8_io_en: regulator-con8ioen {
                        pinctrl-names = "default";
                        pinctrl-0 = <&pinctrl_con8_io_en>;
                        compatible = "regulator-fixed";
                        regulator-name = "CON8_IO_EN";
                        regulator-min-microvolt = <3300000>;
                        regulator-max-microvolt = <3300000>;
                        gpio = <&gpio3 13 GPIO_ACTIVE_HIGH>;
                        enable-active-high;
                        regulator-always-on;
                };
        };
};

&iomuxc {
        pinctrl_con8_io_en: con8ioengrp {
                fsl,pins = <
                        MX6UL_PAD_LCD_DATA08__GPIO3_IO13     0x00008
                >;
        };
};
```

図 3.179 GPIO3_IO13 を High 出力にするノードの例

「図 3.47. 3-State バッファを使用した回路例」では、プルダウン抵抗が基板上で接続されているピンを 3-State バッファの制御ピンとしています。もしプルアップ抵抗が基板上で接続されているピンを制御ピンにしたい場合は、3-State バッファ IC を制御ピンの入力が負論理のものに置き換え、ノード regulator-con8ioen から enable-active-high の行を削除してください。

3.7.24. 起動デバイスを設定する

起動デバイス設定スイッチ(SW2)で起動デバイスを設定できます。起動デバイスは電源投入前に設定します。

SD カードから起動する場合、ブートディスクが必要となります。「6.28. SD ブートの活用」を参考に、ブートディスクの作成を行ってください。また、SD カードの取り扱い方法について、「3.7.7. SD カードを使用する」も合わせてご確認ください。

3.7.24.1. 起動デバイス設定スイッチ(SW2)

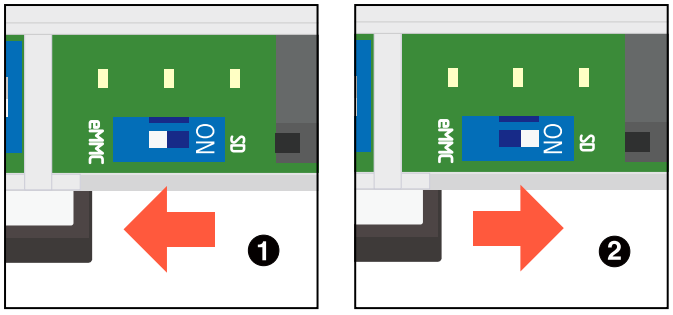


図 3.180 スイッチの状態と起動デバイス

表 3.51 スイッチの状態と起動デバイス

番号	起動デバイス
❶	eMMC
❷	microSD

3.7.25. 外部電源制御出力を使用する

+Di8+Ai4 タイプの場合、外部電源制御出力(CON22)が搭載されています。

3.7.25.1. 外部電源制御出力(CON22)

フォトリレーによる絶縁出力(無極性)となっています。出力部を駆動するためには外部に電源が必要となります。最大電流 500mA(定格 48V)まで駆動可能です。

デフォルトでは、Armadillo-IoT ゲートウェイ A6E がシャットダウンモード時は自動的に OFF(HIGH)になり、スリープモード時、アクティブモード時は ON(LOW)になります。Armadillo-IoT ゲートウェイ A6E が間欠動作する際に、Armadillo-IoT ゲートウェイ A6E の動作に連動して接続しているセンサーなど外部デバイス電源を ON/OFF することを想定しています。

外部電源制御出力には、端子台を実装しています。ねじサイズに合ったドライバービットを使用してください。端子台のねじは、M2 サイズのマイナスねじです。

接続可能な電線については、「表 3.52. CON22 接続可能な電線」をご確認ください。

表 3.52 CON22 接続可能な電線

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.20Nm [a]	

[a]お客様が使用される電線でご確認の上、ねじ締めトルクを設定いただきますようお願いいたします。



振動や衝撃のある場所に設置された場合、端子ねじが緩む事がありますので、定期的な増し締めを行ってください。

電線の先端に予備半田しないでください。正しい接続ができなくなります。

端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

機能 ・ 絶縁出力(無極性) x 1

表 3.53 外部電源制御出力(CON22) 信号配列

ピン番号	ピン名	I/O	説明
10	VOUT	-	外部電源制御出力
11	VCOM	-	外部電源制御出力コモン

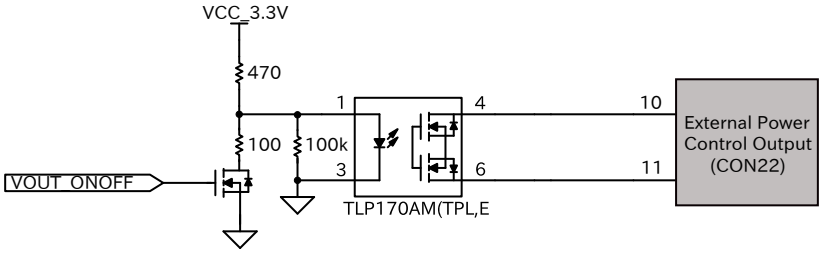


図 3.181 CON22 外部電源制御出力内部回路

ソフトウェアからは GPIO として制御可能です。対応する GPIO 名などを「表 3.54. 外部電源制御出力に対応する CON22 ピン番号」に示します。

表 3.54 外部電源制御出力に対応する CON22 ピン番号

ピン番号	ピン名	GPIO 名	GPIO チップ	GPIO 番号
10 - 11	VOUT - VCOM	CON8_9	gpiochip0	3

3.7.25.2. 外部電源制御出力の出力レベルの設定

gpioset コマンドを用いて、出力レベルを設定することができます。出力レベルには "0" または "1" を設定します。"0"は LOW レベル、"1"は HIGH レベルを表します。

```
[armadillo ~]# gpioset -t0 CON8_9=1
```

図 3.182 出力レベルを "1" に設定する場合

3.7.25.3. 外部電源制御出力をコンテナで使用する

コンテナ内で動作するアプリケーションから外部電源制御出力を使用するには、Podman のイメージからコンテナを作成する際に、ABOS 上の /dev/gpiochipN を渡すと、GPIO**N+1** を操作することができます。

alpine ベースのコンテナを作成し、その中で外部電源制御出力を使用する例を紹介します。

/etc/atmark/containers/vout_example.conf というファイルを以下の内容で作成し、コンテナを起動します。ここでは外部電源制御出力で使用する gpiochip0 を渡します。

```
[armadillo ~]# vi /etc/atmark/containers/vout_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/gpiochip0
[armadillo ~]# podman pull docker.io/alpine
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start vout_example
Starting 'vout_example'
956a0fecc48d5ea1210069910f7bb48b9e90b2dadb12895064d9776dae0360b5
```

図 3.183 外部電源制御出力を扱うためのコンテナ作成例

コンテナ内に入り、必要なソフトウェアをインストールして、外部電源制御出力を操作します。

```
[armadillo ~]# podman exec -it gpio_example sh
[container ~]# apk upgrade
[container ~]# apk add libgpiod
[container ~]# gpioset -t0 CON8_9=1 ❶
```

図 3.184 外部電源制御出力を操作する例

❶ GPIO 名 CON8_9 の値を high に設定します。

C 言語プログラムから操作する場合は、GPIO 操作ライブラリである libgpiod を使用できます。

3.8. ソフトウェアの設計

Armadillo-IoT ゲートウェイ A6E を用いた製品のソフトウェア設計は、一般的な組み込み開発と基本的には変わりません。しかし、 Armadillo Base OS という独自 OS を搭載しているため、ソフトウェアの設計には特有のポイントがいくつかあります。本章では、それらの設計時に考慮すべき Armadillo Base OS 特有のポイントについて紹介していきます。

3.8.1. 開発者が開発するもの、開発しなくていいもの

Armadillo Base OS では、組み込み機器において必要になる様々な機能を標準で搭載しています。

「図 3.185. 開発者が開発するもの、開発しなくていいもの」と「図 3.186. ゲートウェイコンテナ使用時、開発者が開発するもの、開発しなくていいもの」は、 Armadillo Base OS 搭載製品において、開発者が開発するものと開発しなくていいものをまとめた図です。

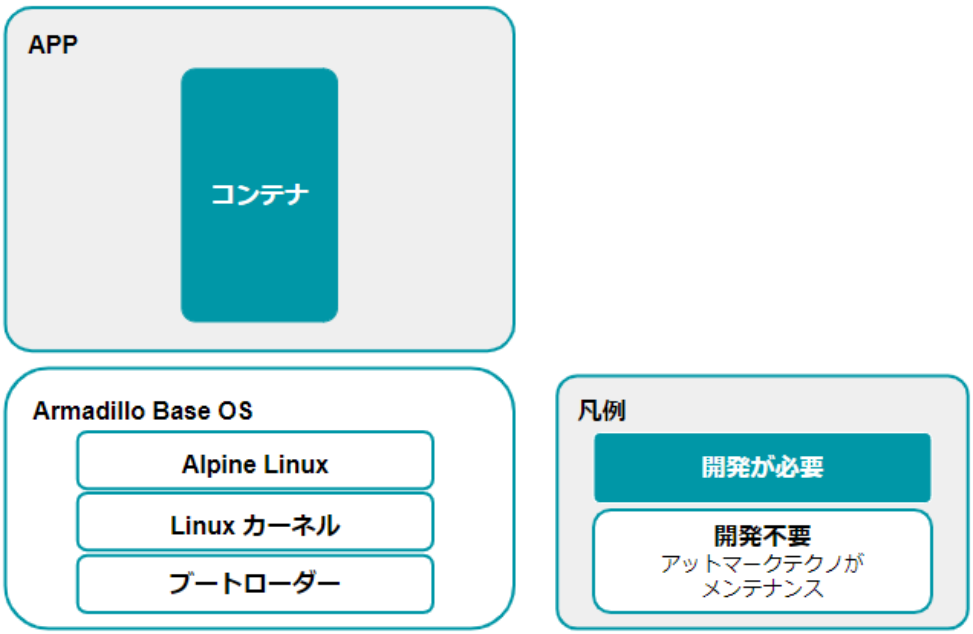


図 3.185 開発者が開発するもの、開発しなくていいもの

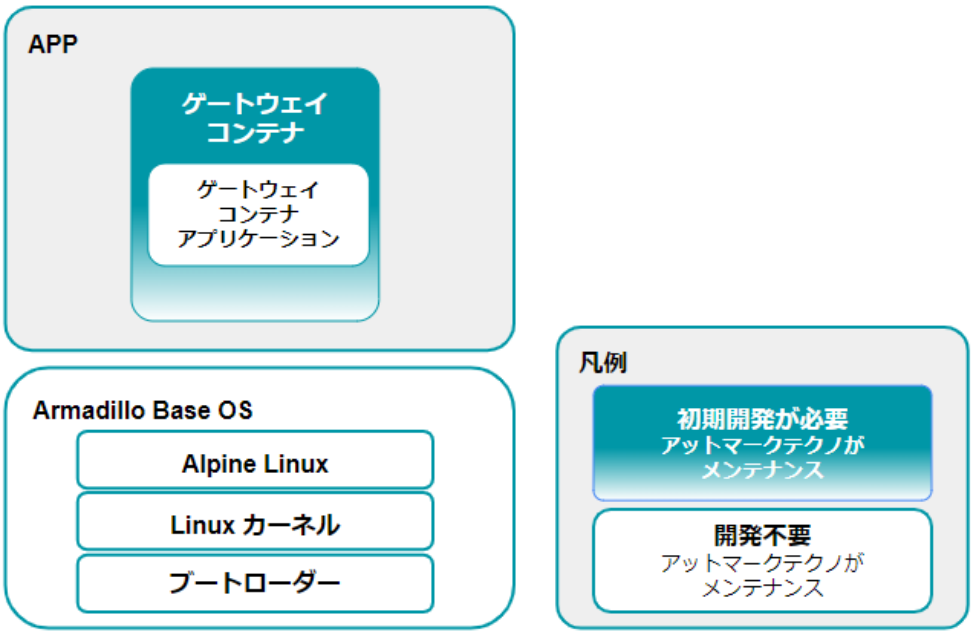


図 3.186 ゲートウェイコンテナ使用時、開発者が開発するもの、開発しなくていいもの

開発しなくていいものについては設計を考える必要はありません。開発するものに絞って設計を進めることができます。



拡張ボードを追加するためにデバイスツリーのカスタマイズが必要となる場合は、デバイスツリー(dtbo)の追加が必要となります。

使用するデバイスによっては、Linux カーネルドライバの追加が必要となり、Linux カーネルのカスタマイズが必要となります。

3.8.2. ユーザーアプリケーションの設計

Armadillo Base OS では基本的にユーザーアプリケーションを Podman コンテナ上で実行します。そのため、実行環境として Armadillo Base OS を意識する必要はありません。

Podman は、同じくコンテナを扱えるソフトウェアである Docker [https://www.docker.com/] と基本的に互換性があります。

アットマークテクノでは、アットマークテクノが提供する Debian GNU/Linux ベースのコンテナイメージ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/debian-container]を提供しておりますが、それ以外の Docker Hub [https://hub.docker.com/search?type=image&image_filter=official] などから使い慣れたディストリビューションのコンテナイメージを取得して開発することができます。

また Armadillo-IoT ゲートウェイ A6E では、ゲートウェイコンテナというコンテナイメージを用意しています。必要な機能がゲートウェイコンテナに全て含まれているのであれば、ゲートウェイコンテナのインストールと、VS Code にて設定を実施して Armadillo にインストールするだけでクラウドへの計測データーの送信や Armadillo の簡易な制御が可能となります。ゲートウェイコンテナの概要については「6.10.1. ゲートウェイコンテナの概要」をご参照ください。

3.8.2.1. ユーザーデータの保存場所

アプリケーションが出力するユーザーデータで保存が必要なものは、「3.3.5.1. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)」にも記載があるとおり、 /var/app/volumes/ 以下に配置してください。

色々な場所にデータが保存されていますと Armadillo-IoT ゲートウェイ A6E の初期化を行う際に削除の処理が煩雑になりますので、 /var/app/volumes/ 以下に集約してください。

3.8.2.2. アプリケーション設定情報の保存場所

開発したアプリケーションやコンテナがバージョンアップした際にも必要となる設定情報は、 「3.3.5.1. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)」 にも記載があるとおり、 /var/app/rollback/volumes/ 以下に保存してください。

3.8.2.3. LTE 通信を使用する場合に考慮すべきこと

LTE 通信は、周辺の状況や工事などによって長時間通信ができなくなる可能性があります。そのため、クラウドやサーバーへ送信すべきデーターを即時に送信できない可能性があります。

データーの再送処理や動作しているコンテナ内にキャッシュする処理を実装して、上記状況に備えてください。

3.8.3. 省電力・間欠動作の設計

Armadillo-IoT ゲートウェイ A6E は、バッテリー駆動などで必要となる、省電力・間欠動作での動作を行う為の制御を用意しております。必要があれば、どのタイミングでスリープ・シャットダウンモードへ遷移するか、なにをトリガーとして起床するかを設計します。次の章「3.8.3.1. 間欠動作モード・起床条件と状態遷移図」にて、省電力・間欠動作の起床条件・状態遷移を説明します。詳細な使用方法は「6.1. 省電力・間欠動作機能」に記載しております。

3.8.3.1. 間欠動作モード・起床条件と状態遷移図

Armadillo-IoT ゲートウェイ A6E の動作モード・起床条件と状態遷移を「図 3.187. 状態遷移図」に示します。また、動作モード毎のデバイス状態を「表 3.55. 動作モード別デバイス状態」に示します。

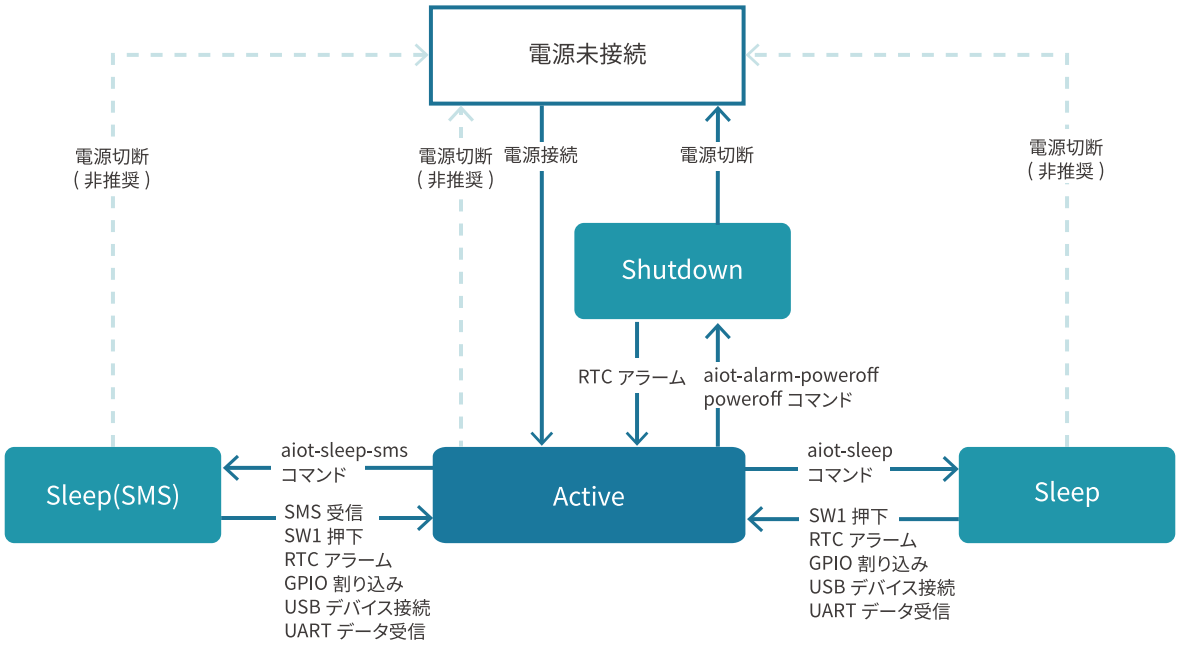


図 3.187 状態遷移図

表 3.55 動作モード別デバイス状態

動作モード	CPU	LTE モジュール	LED	有線 LAN	USB, RS-485 など
アクティブ	動作	通信	動作	動作	通電
シャットダウン	停止	停止	消灯	停止	停止
スリープ	suspend-to-RAM	動作 [a]	消灯	停止	通電
スリープ(SMS 起床可能)	suspend-to-RAM	動作 [a]	消灯	停止	通電

[a]LTE 通信は停止し、LTE モジュールは動作している状態です。「6.16.5.2. 省電力などの設定」の設定に応じた省電力動作になります。

3.8.3.2. 間欠動作モード・起床条件

次に、各動作モードと利用することのできる起床条件について説明します。

3.8.3.3. アクティブモード

「CPU:動作」、「LTE-M:動作」 状態のモードです。

Armadillo-IoT ゲートウェイ A6E の電源投入後 Linux カーネルが起動し、まずはアクティブモードに遷移します。

任意のアプリケーションの実行や、外部センサー・デバイスの制御、LTE-M や Ethernet での通信が可能ですが、最も電力を消費するモードです。アクティブモードの時間をより短くすることで、消費電力を押さえることができます。

3.8.3.4. シャットダウンモード

「CPU:停止」、「LTE-M:停止」の状態であり最も消費電力を抑えることのできるモードです。

その反面、CPU を停止させ、Linux カーネルをシャットダウンしている状態であるため、アクティブモードに遷移する場合は Linux カーネルの起動分の時間がかかります。

シャットダウンモードからアクティブモードに遷移するには、RTC のアラーム割り込みを使用するか、一度電源を切断・再接続を行う必要があります。

3.8.3.5. スリープモード

「CPU:待機」、「LTE-M:停止」 状態のモードです。

CPU(i.MX6ULL)はパワーマネジメントの Suspend-to-RAM 状態になり、Linux カーネルは Pause の状態になります。シャットダウンモードと比較すると消費電力は高いですが、Linux カーネルの起動は不要であるため数秒程度でアクティブモードに遷移が可能です。ユーザスイッチの投下、RTC アラーム割り込み、GPIO 割り込み、USB デバイスの接続、UART によるデータ受信、によってアクティブモードへの遷移ができます。



Cat.1 bis モデルで LTE 接続中にスリープモードをご利用になる場合、スリープモードからアクティブモードへ遷移するタイミングで ping による LTE 通信の導通確認を実施します。

ping 導通確認先の IP アドレスは以下の順序・ルールで決定します。「6.16.5.10. LTE 再接続サービス」 で使用している設定ファイルを参照しています。

1. /etc/atmark/connection-recover/gsm-ttyMux0_connection-recover.conf が存在してファイル内に PING_DEST_IP があれば、この値を使用します。
2. /etc/atmark/connection-recover.conf が存在してファイル内に PING_DEST_IP があれば、この値を使用します。
3. 両方とも存在しない場合は、8.8.8.8 を導通先として使用します。

3.8.3.6. スリープ(SMS 起床可能)モード

「CPU:待機」、「LTE-M:待機」 状態のモードです。

スリープモードとの違いは、SMS の受信によって、アクティブモードへの遷移も可能である点です。LTE-M:待機(PSM)の状態であるため、スリープモードよりも電力を消費します。

3.8.4. ログの設計

ログは、不具合発生時の原因究明の一助になるため必ず残しておくことを推奨します。

3.8.4.1. Armadillo Base OS のログ

- Ethernet、LTE、WLAN などのネットワーク系のログ

  /var/log/messages に保存されます。

- ソフトウェアのバージョンおよびロールバック状況のログ

  /var/at-log/atlog に保存されます。

3.8.4.2. コンテナアプリケーションからのログの保存方法の例

開発したアプリケーションはコンテナ上で動作します。コンテナアプリケーションからのログをホスト OS 側に保存する方法として、例えば以下のように Armadillo Base OS 側の syslogd を使用する方法があります。

- コンテナにホスト OS 側の UNIX ドメインソケットである /dev/log をマウントして、コンテナ内で logger コマンドなどを使用することで /dev/log 経由でホスト OS 側の /var/log/messages にログを送信する

syslogd を使用することで、ログのローテートや古いログの削除などを自動的に行うことができます。

Armadillo Base OS では、podman_start コマンドを使用することでコンテナを起動することができます。podman_start コマンドはコンテナの起動に必要な情報を記述されたコンテナ自動起動設定ファイル(/etc/atmark/containers/<コンテナ名>.conf)を読み取ります。

```
[armadillo ~]# podman pull alpine:latest ❶
Resolved "alpine" as an alias (/etc/containers/registries.conf.d/00-shortnames.conf)
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Writing manifest to image destination
e8f9ca9f1870bc194d961e259fd1340c641bf188e0d02e58b86b86445a4bc128
[armadillo ~]# cat /etc/atmark/containers/example.conf ❷
set_image docker.io/library/alpine:latest

add_volumes /dev/log:/dev/log

set_command sleep infinity
[armadillo ~]# podman_start example ❸
Starting 'example'
ab78aa68aa22fce4574e796fa32604674cba30975d102a1eb5a2969e5e993abc
[armadillo ~]# podman exec -it example /bin/sh ❹
[container ~]#
```

図 3.188 podman_start コマンドでコンテナを起動する例

❶ コンテナイメージを取得します。

❷ コンテナ自動起動設定ファイルの例です。add_volumes コマンドでホスト OS 側の /dev/log をコンテナ内にマウントします。

❸ podman_start コマンドでコンテナを起動します。

❹ podman exec コマンドでコンテナ内に入ります。

以下にシェルスクリプトを使用した場合と Python を使用した場合で、コンテナ内で出力したログを /dev/log 経由でホスト OS 側の /var/log/messages に書き込む例を示します。

```
[container ~]# / # cat ./example.sh ❶
#!/bin/sh

logger -t example "Test log message from container by shellscript"
[container ~]# chmod +x ./example.sh ❷
[container ~]# ./example.sh ❸
```

```
[container ~]# exit ❹
[armadillo ~]# tail -n 1 /var/log/messages ❺
Jan  7 13:08:29 armadillo user.notice example: Test log message from container by shellscript
```

図 3.189 コンテナ内で syslog を使用してログを出力する例（シェルスクリプト）

❶ logger コマンドで syslog にログを出力します。

❷ chmod コマンドでスクリプトに実行権限を付与します。

❸ スクリプトを実行します。

❹ exit コマンドでコンテナから抜けます。

❺ ホスト OS 側の /var/log/messages にログが保存されていることが確認できます。

```
[container /]# apk add python3 ❶
::(省略)
[container /]# cat example.py ❷
import logging
from logging.handlers import SysLogHandler

logger = logging.getLogger('example')
logger.setLevel(logging.INFO)
handler = SysLogHandler(address='/dev/log')
fmt = logging.Formatter('%(name)s: %(levelname)s: %(message)s')
handler.setFormatter(fmt)
logger.addHandler(handler)

logger.info('Test log message from container by Python')
[container /]# python3 example.py ❸
[container /]# exit ❹
[armadillo ~]# tail -n 1 /var/log/messages ❺
Jan  7 12:46:18 armadillo user.info example: INFO: Test log message from container by Python
```

図 3.190 コンテナ内で syslog を使用してログを出力する例（Python）

❶ apk add コマンドで Python3 をインストールします。

❷ Python スクリプトの例です。 logging ライブラリを使用してログを出力します。

❸ python3 コマンドでスクリプトを実行します。

❹ exit コマンドでコンテナから抜けます。

❺ ホスト OS 側の /var/log/messages にログが保存されていることが確認できます。

ログのバッファサイズや出力先の変更を行いたい場合、syslogd の挙動を変更する必要があります。

```
[armadillo ~]# mkdir /var/app/volumes/log
[armadillo ~]# cat /etc/conf.d/syslog ❶
SYSLOGD_OPTS="-t -s 8192 -O /var/app/volumes/log/messages"
[armadillo ~]# persist_file -P /etc/conf.d/syslog ❷
[armadillo ~]# rc-service syslog restart ❸
```

```
::(省略)
[armadillo ~]# cat /var/app/volumes/log/messages
Jan  8 18:38:07 armadillo syslog.info syslogd started: BusyBox v1.37.0
Jan  8 18:38:07 armadillo kern.notice kernel: klogd started: BusyBox v1.37.0 (2025-12-16 14:19:28
UTC)
```

図 3.191 syslogd に渡す引数を変更

❶ syslogd に対して -s オプションでバッファサイズを 8192 バイトに、 -O オプションでログの出力先を /var/app/volumes/log/messages に変更しています。

❷ persist_file コマンドで変更内容を永続化しています。

❸ rc-service コマンドで syslogd を再起動しています。

3.8.4.3. syslogd 以外のログ保存方法

syslogd を使用せず、アプリケーション側でログを作成、保存する処理を追加する場合、/var/app/volumes 以下にログを保存するようにしてください。

その際は、パーティションの容量を考慮したログのローテートや古いログの削除などの仕組みをアプリケーション側で実装する必要があります。:leveloffset: 1

3.8.4.4. ウォッチドッグタイマー

Armadillo-IoT ゲートウェイ A6E のウォッチドッグタイマーは、i.MX6ULL の WDOG(Watchdog Timer)を利用しています。

ウォッチドッグタイマーは、U-Boot によって有効化されます。標準状態でタイムアウト時間は 10 秒に設定されます。

何らかの要因でウォッチドッグタイマーのキックができなくなりタイムアウトすると、システムリセットが発生します。

ウォッチドッグタイマーの設定変更は、ioctl システムコール経由で行うことができます。詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/watchdog/watchdog-api.rst)を参照してください。



ウォッチドッグタイマーを停止することはできません。

3.8.4.5. コンテナに Armadillo の情報を渡す方法

Armadillo Base OS からコンテナに環境変数として情報を渡すためにコンテナ起動設定ファイルを使用します。

コンテナ起動設定ファイル（conf ファイル）に関しては「6.9.4. コンテナ起動設定ファイルを作成する」を参照してください。

・ アットマークテクノが提供する情報を環境変数として渡す

コンテナ起動設定ファイルに add_armadillo_env を使用してください。

アットマークテクノが設定した LAN1（eth0）の MAC アドレス、個体番号などの Armadillo の情報を環境変数としてコンテナに渡します。

add_armadillo_env については「6.9.4.6. 個体識別情報の環境変数の追加」を参照してください。

・任意の情報を環境変数として渡す

コンテナ起動設定ファイルに add_args を使用してください。

add_args については「6.9.4.20. podman run に引数を渡す設定」を参照してください。

add_args を下記のように使用することでコンテナに環境変数として情報を渡すことができます。

```
add_args --env=<環境変数名>=<値> ❶
```

図 3.192 add_args を用いてコンテナに情報を渡すための書き方

❶ シェルコマンドの出力を環境変数に代入する場合は <値> として $(シェルコマンド) を使用してください。

add_args --env の例を示します。

```
add_args --env=MY_ENV=my_value
```

図 3.193 add_args を用いてコンテナに情報を渡す例

これにより、コンテナ内の環境変数 MY_ENV に文字列 my_value が設定されます。

3.8.4.6. Armadillo Base OS のデフォルトで開放しているポート

Armadillo Base OS では「表 3.56. Armadillo Base OS のデフォルトで開放しているポート」に示すポートをデフォルトで開放しています。

表 3.56 Armadillo Base OS のデフォルトで開放しているポート

ポート番号	プロトコル	使用目的
58080	TCP	ABOS Web
5353	UDP	avahi(mDNS)

使用していないポートを開放することは攻撃の標的になります。使用しないサービスを停止しポートを閉じてください。

ABOS Web のサービスを停止する方法は「6.12.9. ABOS Web を停止する」を、起動する方法は「6.12.10. ABOS Web を起動する」を参照してください。

「図 3.194. avahi-daemon を停止する」に avahi のサービスを停止する方法を示します。

```
[armadillo ~]# rc-update | grep avahi-daemon ❶
         avahi-daemon |      default
[armadillo ~]# rc-service avahi-daemon status ❷
```

```
 * status: started
[armadillo ~]# rc-service avahi-daemon stop ❸
avahi-daemon             | * Stopping avahi-daemon ... [ ok ]
[armadillo ~]# rc-update del avahi-daemon ❹
 * service avahi-daemon deleted from runlevel default
[armadillo ~]# persist_file -d /etc/runlevels/default/avahi-daemon ❺
```

図 3.194 avahi-daemon を停止する

❶ OpenRC に avahi のサービスが登録されていることを確認します。

❷ avahi のサービスが起動していることを確認します。

❸ avahi のサービスを停止します。

❹ サービスを管理している OpenRC から avahi のサービスの登録を解除します。

❺ サービス設定ファイルの削除を永続化します。

「図 3.195. avahi-daemon を起動する」に avahi サービスを起動する方法を示します。

```
[armadillo ~]# rc-update | grep avahi-daemon ❶
[armadillo ~]# rc-update add avahi-daemon ❷
 * service avahi-daemon added to runlevel default
[armadillo ~]# rc-service avahi-daemon start ❸
avahi-daemon             | * Starting avahi-daemon ... [ ok ]
[armadillo ~]# persist_file /etc/runlevels/default/avahi-daemon ❹
```

図 3.195 avahi-daemon を起動する

❶ OpenRC に avahi のサービスが登録されていないことを確認します。

❷ サービスを管理している OpenRC に avahi のサービスを登録します。

❸ avahi のサービスを起動します。

❹ サービス設定ファイルを永続化します。

3.9. ネットワーク設定

必要であれば、 Armadillo のネットワークの設定を行います。

3.9.1. ABOS Web とは

Armadillo Base OS(以降、 ABOS) には、 Armadillo と作業用 PC が同一 LAN 内に存在していれば Web ブラウザからネットワークの設定や、 SWU イメージのインストールなどを行うことが可能となる、 ABOS Web という機能があります。この機能は、バージョン v3.17.4-at.7 以降の ABOS に標準で組み込まれています。

現在、ネットワークに関して ABOS Web で設定できるのは以下のものです。

- WWAN 設定
- WLAN 設定

- 各接続設定（各ネットワークインターフェースの設定）
- DHCP サーバー設定
- NAT 設定
- VPN 設定



ABOS Web で設定できる項目はネットワーク関連以外にもありますが、それらについては「6.12. Web UI から Armadillo をセットアップする (ABOS Web)」で紹介します。



バージョン v3.17.4-at.7 以前から ABOS をアップデートした場合の注意

バージョン v3.17.4-at.7 以前からこのバージョン以降へ ABOS をアップデートすると、avahi サービスが新しく追加されます。ABOS Web にアクセスできるようにするためには、この avahi サービスが自動起動するように設定を変更する必要があります。そのため、以下の手順にしたがって設定を変更してください。(新しく追加されたサービスが自動起動することによる悪影響を防ぐため、アップデート直後では avahi サービスは自動起動しない設定になっています。)

```
[armadillo ~]# rc-update add avahi-daemon
[armadillo ~]# rc-service avahi-daemon start
[armadillo ~]# persist_file /etc/runlevels/default/avahi-daemon
```



バージョン 4.13 以前の mkswu を使用した場合の注意

バージョン v3.17.4-at.7 以降の ABOS に、バージョン 4.13 以前の mkswu の mkswu --init で作成した initial_setup.swu をインストールした場合、 ABOS Web にパスワードが設定されていないため自動起動しません。そのため、以下の手順にしたがって ABOS Web のパスワードを設定してください。

```
[armadillo ~]# passwd abos-web-admin
[armadillo ~]# persist_file /etc/shadow
[armadillo ~]# rc-service abos-web restart
```

LTE モジュールを搭載した Armadillo をお使いで、LTE モジュールによる WWAN 接続でインターネットにアクセスする場合に、Armadillo に LAN で接続した機器から Armadillo をルーターとして利

用したい場合には、NAT 設定機能が役に立つでしょう。LTE モジュールによる WWAN 通信でクラウドサービスに接続し、WLAN や LAN で接続した機器から集めたデータをクラウドサービスに転送したり、それらの機器を、クラウドサービスから Armadillo 経由で遠隔制御するようなシステムを構成する場合にご利用ください。

以下では、 ABOS Web を利用した各種ネットワーク設定の方法について紹介します。

3.9.2. ABOS Web へのアクセス

Armadillo と PC を有線 LAN で接続し、Armadillo の電源を入れて PC で Web ブラウザを起動した後、Web ブラウザのアドレスバーに次の URL を入力してください： https://armadillo.local:58080

ABOS Web は、初期状態では同一サブネットのネットワークのみアクセス可能です。
サブネット外からのアクセスを許可したい場合は、 `/etc/atmark/abos_web/init.conf` を作成し、ABOS Web のサービスを再起動してください。

以下の例ではコンテナとループバックからのアクセスのみを許可します：

```
[armadillo ~]# vi /etc/atmark/abos_web/init.conf
command_args="--allowed-subnets '10.88.0.0/16 127.0.0.0/8 ::1/128'"
[armadillo ~]# persist_file -v /etc/atmark/abos_web/init.conf
'/mnt/etc/atmark/abos_web/init.conf' -> '/target/etc/atmark/abos_web/init.conf'
[armadillo ~]# rc-service abos-web restart
```



ABOS Web が動作する Armadillo が、同じ LAN 上に複数あると、ABOS Web に接続する URL のホスト名部分（armadillo.local）は、2 台めでは armadillo-2.local、3 台めでは armadillo-3.local のように、違うものが自動的に割り当てられます。どのホスト名が、どの Armadillo のものなのかを判別するのが難しいので、複数台の Armadillo で同時に ABOS Web を動かすときは、LAN に固定 IP アドレスを設定して、IP アドレスで指定できるようにするのがよいでしょう。

また、VS Code 上で ABOSDE(Armadillo Base OS Development Environment) から、ABOS Web が動作している Armadillo の一覧を確認し、指定した Armadillo の ABOS Web を Web ブラウザ で開くことが出来ます。ただし、ATDE のネットワークを NAT に設定している場合は Armadillo がリストに表示されません。

「図 3.196. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲われているボタンをクリックすることで、ローカルネットワーク上で ABOS Web が実行されている Armadillo をスキャンすることができます。

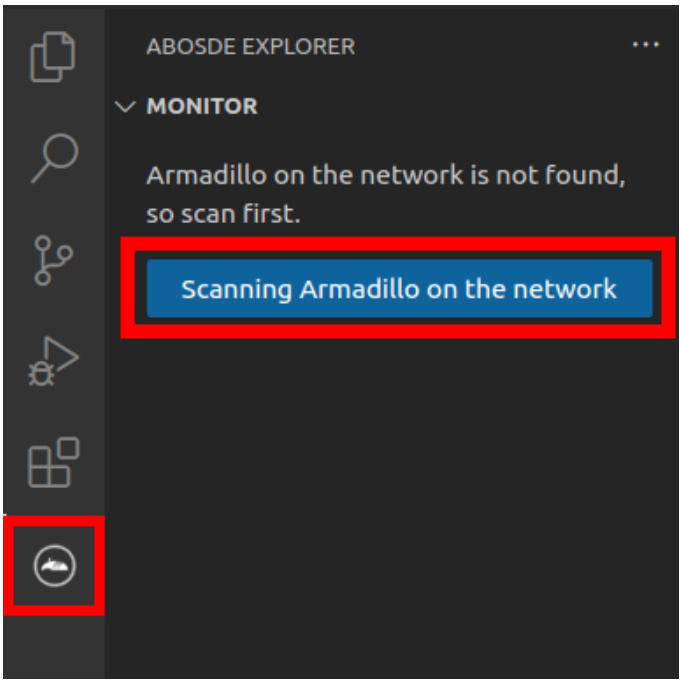


図 3.196 ABOSDE で ローカルネットワーク上の Armadillo をスキャンする

「図 3.197. ABOSDE を使用して ABOS Web を開く」 の赤枠で囲われているマークをクリックすることで、ABOS Web を Web ブラウザで開くことができます。

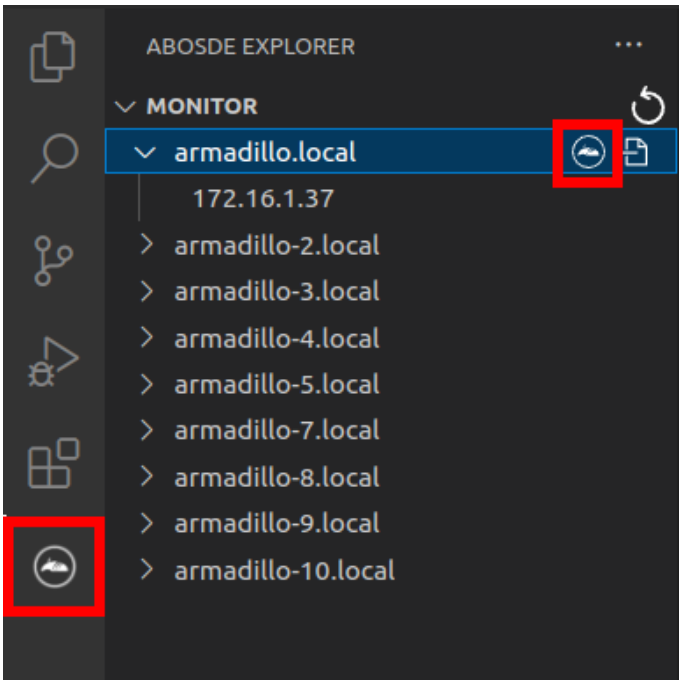


図 3.197 ABOSDE を使用して ABOS Web を開く

「図 3.198. ABOSDE に表示されている Armadillo を更新する」 の赤枠で囲われているマークをクリックすることで、ABOSDE に表示されている Armadillo を更新することができます。

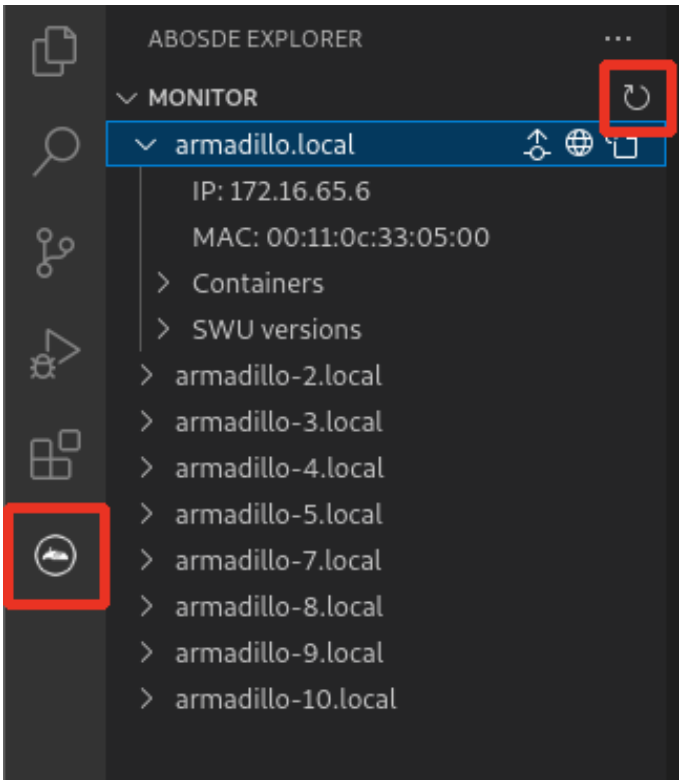


図 3.198 ABOSDE に表示されている Armadillo を更新する

3.9.3. ABOS Web のパスワード登録

「3.1.5.1. `initial_setup.swu` の作成」で ABOS Web のログイン用パスワードを設定していない場合、 ABOS Web 初回ログイン時に、 "初回ログイン"のパスワード登録画面が表示されますので、パスワードを設定してください。



図 3.199 パスワード登録画面

"初回ログイン"のパスワード登録画面で、"パスワード" フィールドと "パスワード(確認)" フィールドに、登録したいパスワードを入力してから、"登録" ボタンをクリックしてください。パスワード登録完了画面が表示されたら、パスワード登録の完了です。



図 3.200 パスワード登録完了画面

パスワード登録完了画面にある "トップページ" というリンクをクリックすると、「図 3.202. トップページ」が表示されます。

以降の接続の際は以下のログイン画面が表示されます。



図 3.201 ログイン画面

ログイン画面で ABOS Web のパスワードを入力して認証されれば、ABOS Web の設定画面に表示が変わり、設定操作を行うことができます。



図 3.202 トップページ

3.9.4. ABOS Web の設定操作

トップページには左上のロゴをクリックするとアクセスできます。

各設定ページには、左端のサイドメニューからアクセスできます。「詳細」がついているカードをクリックすることでも、その設定の詳細ページに移動します。

右上には WWAN、WLAN、LAN の各接続状態が表示されます。WWAN と WLAN は、それらの通信モジュールが Armadillo に搭載されていなければ、表示されません。

3.9.5. ログアウト

ABOS Web の左メニューのログアウトをクリックしてください。ログアウトすると、ログイン画面が再び表示されます。

3.9.6. ABOS Web のパスワード変更

登録した ABOS Web のログイン用パスワードは「設定管理」画面から変更することができます。トップページから「設定管理」をクリックすると、移動した先に「パスワード変更」画面が表示されますので、現在のパスワードと変更後のパスワードを入力して登録ボタンをクリックしてください。

「パスワードリセット」画面では、ABOS Web のパスワードをリセットすることができます。リセットを行うと、次回、ABOS Web にログインする際にパスワードの再設定を求められます。

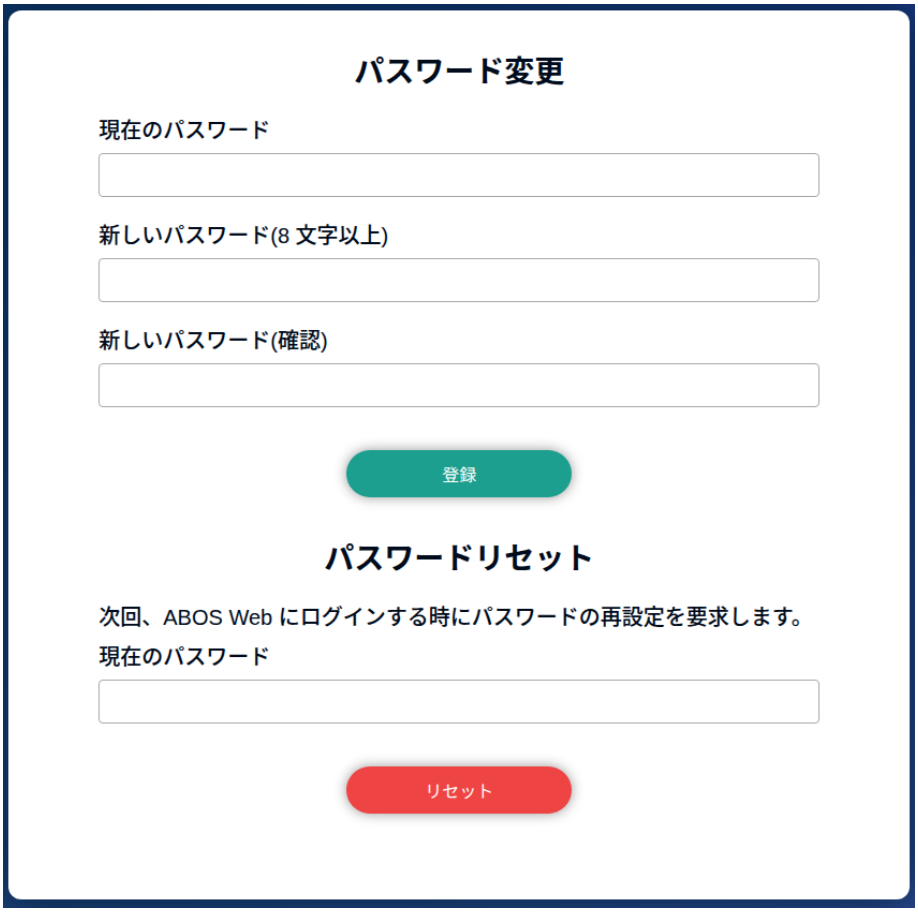


図 3.203 パスワード変更画面

3.9.7. WWAN 設定

LTE をはじめとする WWAN 通信モジュールを搭載した Armadillo の、WWAN 設定を行います。この設定画面では、WWAN 接続設定の登録と、WWAN 接続の状態（現在のアドレス情報）の表示、登録済み WWAN 接続設定の編集と削除を行うことができます。



図 3.204 WWAN 設定入力画面

設定項目のうち、"MCC/MNC" は、通常は空欄にしてください。MCC/MNC 以外の項目を正しく設定しても WWAN 通信が動作しない場合、特に SIM カードがマルチキャリア SIM の場合は、ご契約の通信事業者に MCC/MNM を問い合わせ、通信事業者から提示された MCC/MNC の値を設定してください。それぞれの入力フィールドに設定値を入力して "接続して保存" ボタンをクリックすると、WWAN 接続の設定を登録して、WWAN 接続動作を実行します。このとき nanoSIM カードが挿入されていない場合や、設定した値に誤りがある場合はエラーを表示します。"保存" ボタンをクリックした場合は、

nanoSIM カードが挿入されていない場合や、設定した値に誤りがある場合でもエラーは表示されません。WWAN 通信設定が行われ、ネットワーク接続が確立した状態では、割当たっている IP アドレスなどを "現在の WWAN 接続情報" に表示します。「図 3.205. WWAN 設定画面」に、WWAN 設定を行った状態を示します。



図 3.205 WWAN 設定画面



ABOS Web のバージョン 1.3.3 以降では「IPv6 設定」を選択することができます。使用する SIM によっては IPv6 が有効だと接続できず、無効にすると接続できることがあります。その場合は、この設定を「使用しない」に設定して接続してください。



閉域 LTE 網を使用する料金プランをご契約で本サービスをご利用になられる際の注意点。

「6.16.5.10. LTE 再接続サービス」をご利用になられる場合、コネクション状態確認時 PING 送付先の初期値は 8.8.8.8 ですが、この IP アドレスに対して ping 導通ができない場合、ping 導通が可能となる IP アドレスを指定する必要があります。設定方法は、「6.16.5.10. LTE 再接続サービス」を参照ください。

3.9.7.1. WWAN 再接続機能設定

WWAN 接続が切断された場合に自動的に再接続を試みる機能の設定を行います。

「図 3.206. 「WWAN 再接続設定」画面」に、WWAN 再接続機能の設定画面を示します。

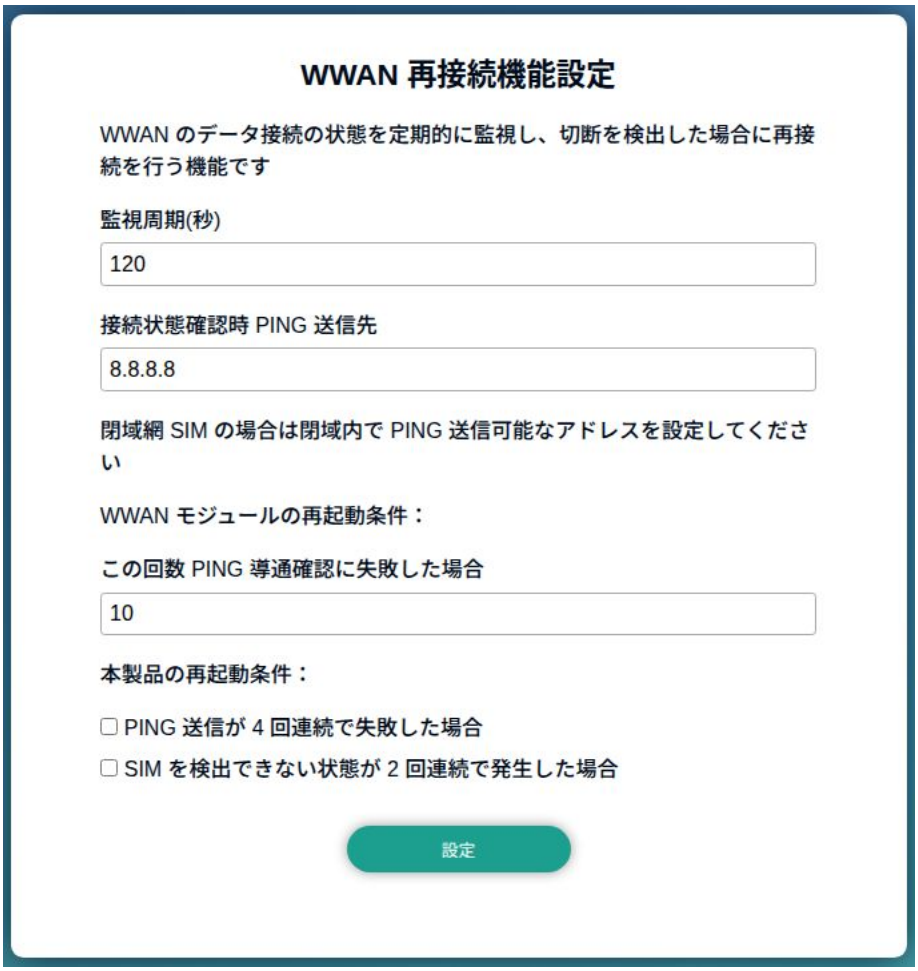


図 3.206 「WWAN 再接続設定」画面

表 3.57 WWAN 再接続設定項目

項目	設定値	説明
監視周期（秒）	120（デフォルト）	WWAN 接続を確認する周期を設定します。
接続状態確認時 PING 送信先	8.8.8.8（デフォルト）	接続状態を確認するために PING を送信する先の IP アドレスを設定します。 閉域網 SIM の場合は閉域内で PING 送信可能なアドレスを設定してください。

項目	設定値	説明
WWAN モジュールの再起動条件	10（デフォルト）	PING 送信が失敗した回数をトリガーに WWAN モジュールを再起動できます。
本製品の再起動条件	無効（デフォルト） 有効	本製品を再起動する条件として以下の 2 つを設定できます。 ・PING 送信が 4 回連続で失敗した場合 ・SIM を検出できない状態が 2 回連続で発生した場合

3.9.8. WLAN 設定

無線 LAN モジュールを搭載した Armadillo の、WLAN（無線 LAN）設定を行います。この設定画面では、WLAN クライアント（子機）としての設定または、WLAN アクセスポイントとしての設定を行うことができます。クライアントとアクセスポイントのどちらか一方について、接続設定の登録と接続の状態の表示、登録済み設定の削除を行なえます。クライアントとアクセスポイントのどちらに設定するかは、"動作モード選択"欄で指定します。

クライアント設定とアクセスポイント設定の、それぞれについて、以下に説明します。

3.9.8.1. WLAN 設定（クライアントとしての設定）

"動作モード選択"欄で"クライアントとして使用する"を選択すると、クライアント設定画面が表示されます。もしアクセスポイントに設定済みの場合は、アクセスポイントの設定を削除してください。そうしないと、動作モードをクライアントに切り替えることができません。設定項目のうち、ネットワーク名(SSID) は、リストから選択してください。WLAN アクセスポイントを Armadillo が何も検出できない場合は、このリストが空になります。非表示に設定しているアクセスポイントに接続する場合は、「非表示の SSID に接続する」のチェックボックスをチェックしてください。セキュリティ方式も、リストから選択してください。DHCP と 固定 は、DHCP を選択すると DHCP サーバーから IP アドレスを取得します。固定 を選択すると、固定 IP アドレス設定用の入力フィールドを表示します。

それぞれの入力フィールドに設定値を入力して "接続して保存" ボタンをクリックすると、WLAN クライアント設定を登録して、WLAN 接続動作を実行します。

ABOS-WEB 上では複数のネットワーク設定を保存することが可能です。設定項目のうちにネットワーク情報を入力した後、 "保存" ボタンをクリックすると、入力した内容の登録のみを行い、接続は行いません。



図 3.207 WLAN クライアント設定入力画面

WLAN 接続設定が行われ、WLAN 接続が確立した状態では、割当たっている IP アドレスなどを "現在の WLAN 接続情報" に表示します。「図 3.208. WLAN クライアント設定画面」に、WLAN クライアント設定を行った状態を示します。



図 3.208 WLAN クライアント設定画面

登録した設定の一覧は WLAN ページの中央にあるリストに表示されます。このリストでは WLAN 設定の接続／編集／削除を行うことができます。保存した設定に接続先を変更したい場合はリストから選択して、"接続" ボタンをクリックしてください。保存した設定を編集したい場合はリストから選択して、"設定を編集" ボタンをクリックしてください。保存した設定を削除したい場合はリストから選択して、"設定を削除" ボタンをクリックしてください。

登録済み WLAN クライアント設定を削除して、WLAN アクセスポイントとの接続を切断するには、"設定を削除" ボタンをクリックしてください。

3.9.8.2. WLAN 設定（アクセスポイントとしての設定）

"動作モード選択"欄で"アクセスポイントとして使用する"を選択すると、アクセスポイント設定画面が表示されます。もしクライアントに設定済みの場合は、クライアントの設定を削除してください。そうしないと、動作モードをアクセスポイントに切り替えることができません。設定項目のうち、ブリッジアドレス は、WLAN アクセスポイントに割り当てる IP アドレスを入力してください。サブネットマスクは、アクセスポイントのサブネットのものを入力してください。使用周波数は、5GHz と 2.4GHz のうち使用するものを選択してください。両方の周波数を同時に使用することはできません。使用チャンネルは、リストから選択してください。SSID と パスワード に入力した値は、アクセスポイントに設定した Armadillo に WLAN 子機を接続する際に使用します。

それぞれの入力フィールドに設定値を入力して "設定" ボタンをクリックすると、WLAN アクセスポイント設定を登録して、WLAN アクセスポイント動作を開始します。



図 3.209 WLAN アクセスポイント設定入力画面

WLAN アクセスポイント設定が行われ、アクセスポイント動作中の状態では、"現在のアクセスポイント情報" に設定内容を表示します。「図 3.210. WLAN アクセスポイント設定画面」に、WLAN アクセスポイント設定を行った状態を示します。



図 3.210 WLAN アクセスポイント設定画面



アクセスポイントモードのセキュリティ方式は、WPA2 を使用します。

3.9.9. 接続詳細設定

設定されたネットワーク接続の一覧を表示します。

「詳細を表示」ボタンをクリックすると、現在設定されているネットワーク接続の一覧を表示します。

表示したそれぞれの接続について、接続の有効化（「接続」）や無効化（「切断」）、および接続設定内容の編集や削除を行うことができます。接続の操作を行う時は、操作したい接続をラジオボタンで選択してください。



図 3.211 現在の接続情報画面

ここで、「ネットワーク接続」は、Linux のネットワーク接続管理機能（NetworkManager）が管理するコネクションです。ネットワーク接続に対する設定項目の詳細は、NetworkManager のリファレンス（ https://networkmanager.dev/docs/api/latest/nmcli.html ）をご覧ください。接続設定内容を編集したい接続を選択して "設定を編集" ボタンをクリックすると、設定内容の編集画面を表示します。LAN の接続以外、つまり、WWAN と WLAN の接続に対する設定は、"WWAN 設定" や "WLAN 設定" の設定画面をお使いいただくのが簡単です。

それぞれの接続設定画面では、IPv4 と IPv6 のそれぞれについて、IP アドレスを自動割り当てするかまたは固定 IP アドレスにするかを選択して設定できます。IP アドレスの割り当ては、デフォルトでは自動割り当てです。Armadillo を接続した LAN や WLAN で、Armadillo を DHCP サーバーとして運用する場合は、それらのネットワーク接続を固定 IP アドレスに設定してください。

3.9.9.1. LAN 接続設定

LAN 接続の接続名は、デフォルトでは "Wired connection 1" です。LAN ポートを二つ搭載した Armadillo では、二つめの LAN ポートに対応する "Wired connection 2" も有効です。Armadillo を LAN と WWAN との間で IPv4 ルーターとして運用する場合は、LAN 接続の設定で IPv4 アドレスを固定 IP アドレスに設定して下さい。「図 3.212. LAN 接続設定で固定 IP アドレスに設定した画面」に、LAN 接続の設定編集画面で固定 IP アドレスに設定した状態を示します。



図 3.212 LAN 接続設定で固定 IP アドレスに設定した画面

3.9.9.2. WWAN 接続設定

WWAN 接続の接続名は、デフォルトでは "gsm-ttyCommModem" です。

3.9.9.3. WLAN 接続設定

WLAN 接続の接続名は、デフォルトでは、クライアントモードが "abos_web_wlan"、アクセスポイントモードが "abos_web_br_ap" です。

3.9.10. DHCP サーバー設定

ネットワークインターフェースごとに、接続したネットワーク上で Armadillo を DHCP サーバーとして動作させる設定を行うことができます。



図 3.213 DHCP サーバー設定入力

接続済みの DHCP サーバー情報を、画面上部の"現在の DHCP 情報"に表示します。DHCP サーバーの設定を登録する場合は、"DHCP 情報入力"欄に設定内容を入力して"有効"ボタンをクリックしてください。「図 3.214. eth1 に対する DHCP サーバー設定」に、LAN ポート（eth1）に対する設定を行った状態を示します。



図 3.214 eth1 に対する DHCP サーバー設定

DHCP サーバー設定を削除するには、"現在の DHCP 情報"の一覧で削除したい設定を選択して、"無効"ボタンをクリックしてください。

3.9.11. NAT 設定

この設定画面では、ルーター機能での宛先インターフェース設定と、Armadillo を接続した LAN 上の機器用のポートフォワーディング設定を行うことができます。Armadillo を LAN や WLAN と WWAN との間でルーターとして運用する場合は、NAT 設定の宛先インターフェースを WWAN のインターフェースに設定してください。そして、LAN や WLAN 上の機器を、WWAN 接続したインターネットにサーバーとして公開したい場合は、ポートフォワーディング設定を使ってください。さらに、WAN や LAN インターフェースを指定した上で、受信と転送における IP アドレスフィルタリングを設定することもできます。

3.9.11.1. NAT 設定

ルーター機能での宛先インターフェース設定を行なえます。「図 3.215. LTE を宛先インターフェースに指定した設定」に、宛先インターフェースに ppp0 を指定した場合の画面を示します。



図 3.215 LTE を宛先インターフェースに指定した設定

3.9.11.2. ポートフォワーディング設定

受信インターフェースごとに、ポートフォワーディング設定を登録できます。受信インターフェースとして * を指定すると、すべてのインターフェースで受信したパケットを対象にします。



図 3.216 ポートフォワーディング設定入力

「図 3.217. 全ての受信パケットに対するポートフォワーディング設定」 に、全ての受信インターフェースについて、ポート 8080 番宛の tcp パケットをポートフォワーディングする設定を行った状態を示します。



図 3.217 全ての受信パケットに対するポートフォワーディング設定

3.9.11.3. IP アドレスフィルタリング設定

この設定画面では、IP アドレスフィルタリング設定を行うことができます。

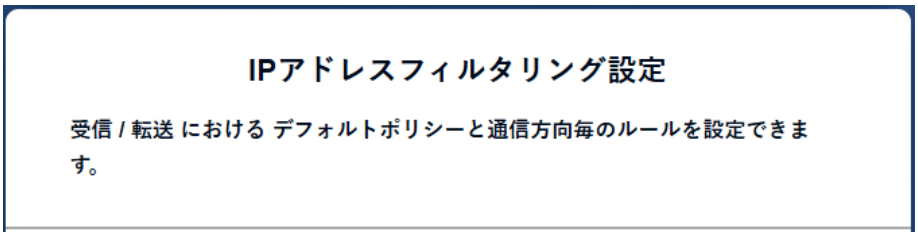


図 3.218 「IP アドレスフィルタリング設定」画面

・ **デフォルトポリシー設定**

デフォルトポリシー設定では、受信、転送において個別のルールに該当しなかったパケットを、以下のどちらにするかを設定します。

・ ACCEPT：全て通す

・ DROP：全て破棄する



図 3.219 「デフォルトポリシー」画面



デフォルトポリシーを DROP にすると、個別ルールに設定されてないパケットは破棄されることになります。
本製品で通信を行う設定を個別ルールに追加してください。
また、受信のデフォルトポリシーを DROP にした場合、ABOS Web、avahi(mDNS)、SSH、ICMP は自動的に個別ルールとして ACCEPT に設定されます。

・ **WAN/LAN インターフェースの設定**

WAN LAN に割り当てインターフェースの設定、削除ができます。インターフェースが設定されていないとフィルタルールは追加できません。



図 3.220 「WAN/LAN インターフェースの設定」画面

・ フィルタルールの設定

フィルタルールの設定では、現在のルールの確認、選択したルールの削除と移動、ルールの追加をすることができます。ルールは上にある方が優先され、移動は同じ通信方向内でのみ可能です。



図 3.221 「現在のルール」画面

ルールの追加では、各項目を設定して、設定した通信方向の先頭か末尾にルールが追加できます。



図 3.222 「ルールの追加」画面

表 3.58 「ルールの追加」画面の項目

項目	設定値	説明
通信方向	WAN からの受信 LAN からの受信 WAN から LAN への転送 WAN から LAN への転送	ルールを設定する通信方向を選択します。 WAN インターフェースと LAN インターフェースを一つ以上設定してないと全ての選択肢が出てきません。
IPver	auto ipv4 ipv6 all	ルールの IP バージョンを選択します。auto を指定した場合、アドレス値から IP バージョンを自動判定します。all を指定した場合、アドレスは設定不可になります。
プロトコル	tcp udp icmp all	ルールを設定するプロトコルを選択します。icmp,all を指定した場合、ポート番号は設定不可になります。
送信元アドレス	任意	送信元の IP アドレスを指定します。IPv4/CIDR または IPv6/CIDR 形式が指定できます。

項目	設定値	説明
送信元ポート番号/範囲	任意	送信元のポート番号を指定します。:を使用して範囲指定することもできます。
宛先アドレス	任意	宛先 IP アドレスを指定します。IPv4/CIDR または IPv6/CIDR 形式が指定できます。
宛先ポート番号/範囲	任意	宛先のポート番号を指定します。:を使用して範囲指定することもできます。
アクション	ACCEPT(許可) DROP(破棄) REJECT(拒否)	このルールにマッチしたパケットをどうするか設定します。

また、WAN インターフェース LAN インターフェース を設定していないと、以下が表示されルールの追加はできません。

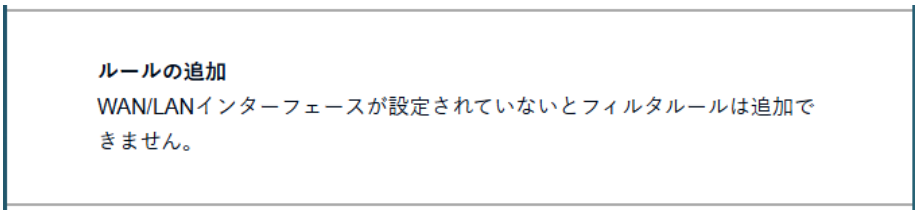


図 3.223 「ルールの追加」不可画面

・ **設定初期化**

IP アドレスフィルタリング設定の初期化ができます。

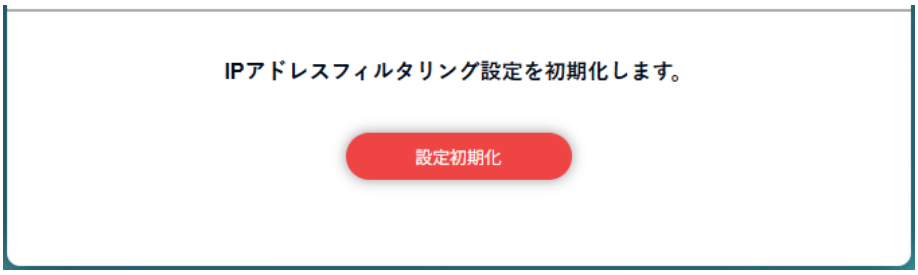


図 3.224 「IP アドレスフィルタリング設定の初期化」画面

3.9.12. VPN 設定

Armadillo の VPN 接続設定を行います。

この設定画面では、認証方式や VPN クライアントの設定ファイル、認証用の証明書と鍵の設定を行うことができます。

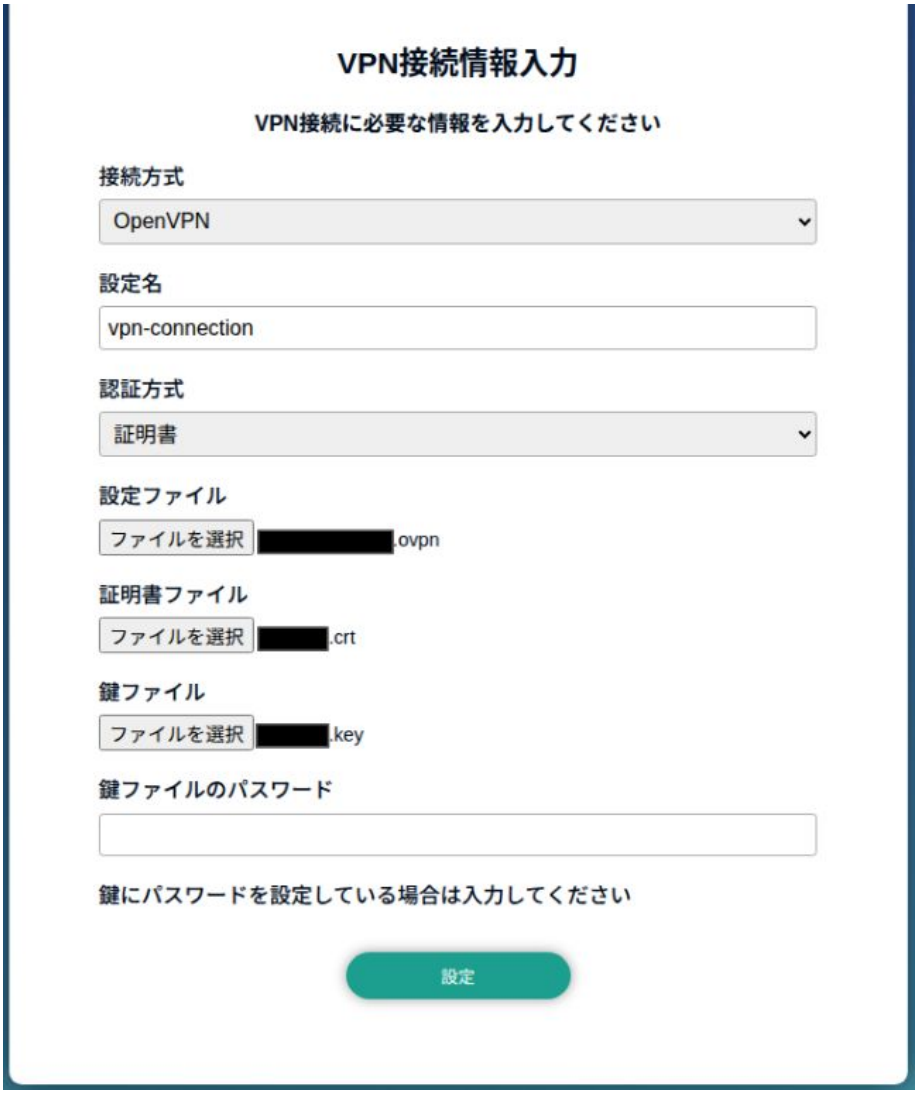


図 3.225 VPN 接続情報入力画面

表 3.59 OpenVPN 設定項目

項目	説明
設定名	設定につける名前を入力してください。
認証方式	「ユーザー名とパスワード」または「証明書」を選択してください。
設定ファイル	OpenVPN の設定ファイルを選択してください。

認証方式によって、以下の設定項目が表示されます。

表 3.60 認証方式として「ユーザー名とパスワード」を選択した場合の設定項目

項目	説明
ユーザー名	認証用のユーザー名を入力してください。
パスワード	認証用のパスワードを入力してください。

表 3.61 認証方式として「証明書」を選択した場合の設定項目

項目	説明
証明書ファイル	認証用の証明書ファイルを選択してください。設定ファイルに証明書が埋め込まれていれば選択する必要はありません。
鍵ファイル	認証用の鍵ファイルを選択してください。設定ファイルに鍵が埋め込まれていれば選択する必要はありません。
鍵ファイルのパスワード	鍵ファイルにパスワードを設定している場合は入力してください。

”設定" ボタンをクリックすると、ABOS Web は abos_web_openvpn という名前のコンテナを自動的に作成し、このコンテナで VPN クライアントを実行します。VPN クライアントは、現在 OpenVPN [https://openvpn.net/community/] をサポートしています。

設定の進行状況は "設定" ボタンをクリックした時に表示されるログで確認できます。

「図 3.226. VPN 設定確認画面」に、VPN 接続設定を行った状態を示します。



図 3.226 VPN 設定確認画面

現在の接続状態表示欄にある "接続" ボタンと "切断" ボタンで、VPN 接続の接続と切断を行なえます。登録済み VPN 設定を削除して、VPN 接続を切断するには、"設定を削除" ボタンをクリックしてください。

3.9.13. 状態一覧

各設定画面で行った設定の現在状態を、設定ごとに区切って一覧表示します。

3.10. USB デバイスの接続を許可する

IoT 機器において、悪意のある第三者が容易に悪用できる USB コネクタなどのインターフェースを外部に露出したままでの運用は、セキュリティ的な脆弱性に繋がります。しかし、保守運用のために USB インターフェースを露出しておかなければならない場合や、機器として USB で接続する周辺デバイスがある場合など、全ての USB インターフェースを物理的に閉じておくことが難しいことも考えられます。

Armadillo Base OS は、 USB デバイスの許可リストを作成・管理し、許可リストにないデバイスが接続されても認識しないように設定できる USB 接続制御機能を持っています。この機能を用いることで、悪意のある第三者が不正な USB デバイスを接続しても、システムに影響を与えません。



ABOS Web による USB 接続制御機能は、ABOS バージョン 3.21.3-at.12 以降で対応しています。

この機能について、CUI で実行したい場合は「6.37.3. 不正な USB デバイスの接続を拒否する」をご参照ください。



工場出荷状態における USB 接続制御機能

製品のセキュリティ向上のため、 ABOS バージョン 3.22.1-at.1 以降では、USB 接続制御機能がデフォルトで有効化されています。デフォルトで接続が許可されている USB デバイスは次の通りです。

表 3.62 デフォルトで許可されている USB デバイス

デバイス名/デバイスクラス名	利用目的
MassStorage クラス	SWUpdate で使用するため
Hub クラス	Armadillo 内部で使用しているため
LTE モジュール(SIM7672G/SIM7672JP) (搭載している場合)	製品として標準で搭載しているため

開発中に追加で上記以外の USB デバイスを接続する場合は、以下の手順にしたがってデバイスの接続を許可してください。

ABOS Web では、Armadillo の USB 接続制御の設定を行うことができます。

ここでは、USB デバイスの許可リストを作成・管理し、許可リストにないデバイスが接続されても認識しないように設定できます。この機能を用いることで、悪意のある第三者が不正な USB デバイスを接続しても、システムに影響を与えません。

「図 3.227. USB 接続制御の設定画面」に設定画面を示します。



図 3.227 USB 接続制御の設定画面

一番上の「機能の状態」に現在の USB 接続制御機能が有効であるか（enabled）無効であるか（disabled）表示します。

有効である場合、「接続済みの USB デバイス」、「許可済みの USB デバイス」および「許可済みの USB デバイスクラス」の欄が表示されます。

「図 3.228. 接続済みの USB デバイス」に示すように、「接続済みの USB デバイス」では、現在接続されている USB デバイスが一覧として表示されます。



図 3.228 接続済みの USB デバイス

"詳細表示" ボタンを押すと、「図 3.229. 接続済み USB デバイスの詳細情報」に示す画面が表示されて、その USB デバイスに関するより詳細な情報が表示されます。



図 3.229 接続済み USB デバイスの詳細情報

「可否」が allow である場合、その USB デバイスは接続が許可されています。

block の場合、「図 3.230. USB デバイスを許可する」に表示されている "許可" ボタンを押すことで allow に変更できます。



図 3.230 USB デバイスを許可する

"許可" ボタンを押すと、「図 3.231. 許可済み USB デバイス」に示すように「許可済みの USB デバイス」の欄に許可ルールが追加されます。

「許可済みの USB デバイス」に表示されている USB デバイスは永続的に接続が許可されます。



図 3.231 許可済み USB デバイス

「図 3.232. 接続済みの個体と同じメーカー・製品の全ての個体を許可する」に示すように、「同じメーカー・製品の全ての個体を許可」のチェックボックスを選択した状態で "許可" ボタンを押すと、選択した USB デバイスと同じメーカー・製品であれば、どの個体であっても接続が許可されます。



図 3.232 接続済みの個体と同じメーカー・製品の全ての個体を許可する

その場合、「図 3.233. 全ての個体を許可する場合の表示」に示すように、「許可済みの USB デバイス」の「全ての個体を許可」の欄に「✓」が表示されます。



図 3.233 全ての個体を許可する場合の表示

「図 3.233. 全ての個体を許可する場合の表示」に示すように、USB デバイスの許可ルールを選択した状態で "削除" ボタンを押すと、「図 3.234. 許可ルールを削除する」に示す画面に遷移します。



図 3.234 許可ルールを削除する

確認画面が表示されて、問題なければ改めて "削除" のボタンを押すことでその許可ルールを削除できます。

その USB デバイスに関する許可ルールが削除されると永続的にその USB デバイスの接続は拒否されます。「接続済みの USB デバイス」では、「可否」が block に変更されます。

「図 3.235. USB デバイスクラス」に表示されている「許可済みの USB デバイスクラス」では、許可する USB デバイスをデバイスクラス単位で設定することができます。



図 3.235 USB デバイスクラス

一覧に表示されている USB デバイスクラスは現在許可されているデバイスクラスです。

デバイスクラスを選択した後、"削除" ボタンを押すことでそのデバイスクラスの許可ルールを削除できます。削除の手順は「図 3.234. 許可ルールを削除する」と同様です。

"追加" ボタンを押すと、「図 3.236. USB デバイスクラスの追加」に示す画面が表示されます。追加可能な USB デバイスクラスを選択し、"追加" ボタンを押してください。



図 3.236 USB デバイスクラスの追加

3.11. ABOS Web をカスタマイズする

ABOS Web では以下の要素についてお客様自身で用意したものを使用してカスタマイズすることができます。

- ロゴ画像
- ヘッダロゴアイコン画像
- ヘッダタイトル
- 製品型番
- favicon 画像
- 背景色
- メニューの表示名

ABOS Web をお客様の最終製品に組み込む場合、自社のロゴ画像に変更するといったような使い方ができます。

カスタマイズは、「設定管理」で行うことができます。



カスタマイズは ABOS Web のバージョン 1.3.0 以降で対応しています。製品型番のカスタマイズに関してはバージョン 1.10.0 以降で対応しています。



図 3.237 ABOS Web のカスタマイズ設定

- **ロゴ画像**

  ログインページや新規パスワード設定画面で表示される画像です。「ファイルを選択」をクリックしてアップロードしたい画像ファイルを選択してください。フォーマットは PNG のみで、ファイルサイズは 3MB のものまでアップロードできます。

- **ヘッダロゴアイコン画像**

  画面左上に常に表示されている画像です。「ファイルを選択」をクリックしてアップロードしたい画像ファイルを選択してください。フォーマットは PNG のみで、ファイルサイズは 3MB のものまでアップロードできます。

- **ヘッダタイトル**

  画面左上に常に表示されている文字列です。24 文字まで入力できます。

- **製品型番**

  設定するとヘッダタイトルの下に表示されます。お客様の最終製品の型番を 24 文字まで入力できます。

- **favicon 画像**

  Web ブラウザのタブなどに小さく表示される画像です。favicon 画像は以下の種類を `favicon` ディレクトリに保存して、`favicon` ディレクトリごと zip 圧縮したものをアップロードしてください。

表 3.63 用意する favicon 画像

ファイル名	縦横サイズ	説明
android-chrome-192x192.png	192x192	スマートフォンのホームに Web ページを追加した時に使用されます。
android-chrome-512x512.png	512x512	Web ページを開いた時のスプラッシュ画面に使用されます。
apple-touch-icon.png	180x180	スマートフォンのホームに Web ページを追加した時に使用されます。
favicon-16x16.png	16x16	Web ブラウザで使用されます。
favicon-32x32.png	32x32	Web ブラウザで使用されます。
mstile-150x150.png	150x150	Windows でスタート画面にピン止めしたときに使用されます。

- **背景色**

  5 種類の中から選択できます。

- **メニューの表示名**

  画面左にあるメニューの表示名を変更する、または非表示にすることができます。「メニュー項目を変更する」をクリックし、変更用ページへ行ってください。



図 3.238 メニュー変更画面 (一部)

各メニュー項目名と説明を変更することができます。項目名を空欄にするとそのメニューは非表示になります。入力し終わったらページ下部の「メニューを設定」をクリックしてください。

画像やメニューの変更後、すぐに Web ブラウザ画面に反映されない場合は、お使いの Web ブラウザの設定でキャッシュの削除を行ってください。

変更完了後は、「カスタマイズ機能を無効にする」をクリックするとカスタマイズ項目が非表示になりそれ以上カスタマイズできなくなります。お客様の最終製品に ABOS Web を組み込む場合に実行してください。



Armadillo 内の `/etc/atmark/abos_web/customize_disable` ファイルを削除すると、再びカスタマイズ項目が表示されるようになります。

3.12. ABOS Web から Armadillo の電源を操作する

開発中及び運用中に Armadillo の再起動または電源を切るには、ターミナルでコマンドを実行することでも可能ですが、ABOS Web からも実行可能です。

左サイドメニューの「電源制御」をクリックしてください。



図 3.239 「電源制御」の画面

操作可能な機能は以下です。

表 3.64 制御できる機能

機能名	Armadillo 内で実行されるコマンド	説明
再起動	reboot	Armadillo を再起動します。
電源オフ	poweroff	Armadillo の電源を切ります。

これらの機能を ABOS Web で選択すると、確認画面が表示されます。よければ「OK」を選択してください。

電源オフを実行した場合、Armadillo の LED が消えるまで電源を抜かないでください。

3.13. Network Time Protocol (NTP, ネットワーク・タイム・プロトコル) の設定

Armadillo Base OS では chronyd を使用しています。

デフォルトの設定（使用するサーバーなど）は /lib/chrony.conf.d/ にあり、設定変更用に /etc/chrony/conf.d/ のファイルも読み込みます。/etc/chrony/conf.d/ ディレクトリに /lib/chrony.conf.d/ と同名の設定ファイルを配置することで、デフォルトのファイルを読み込まないようになります。

時刻取得に関する設定は 2 つのファイルに分かれています：

- initstepslew.conf： chronyd 起動時「 initstepslew 」コマンドでサーバーと通信し時刻を取得します。
- servers.conf： chronyd 起動後周期的に「 pool 」または「 server 」コマンドでサーバーと通信し時刻を補正します。

例えば、 NTP サーバーを変更する際は 「図 3.240. chronyd のコンフィグの変更例」 に示す通り/etc/chrony/conf.d/initstepslew.conf と /etc/chrony/conf.d/servers.conf に記載します：

```
[armadillo ~]# vi /etc/chrony/conf.d/initstepslew.conf ❶
initstepslew 10 192.0.2.1
[armadillo ~]# vi /etc/chrony/conf.d/servers.conf ❷
server 192.0.2.1 iburst
[armadillo ~]# persist_file -rv /etc/chrony/conf.d ❸
'/mnt/etc/chrony/conf.d/initstepslew.conf' -> '/target/etc/chrony/conf.d/initstepslew.conf'
'/mnt/etc/chrony/conf.d/servers.conf' -> '/target/etc/chrony/conf.d/servers.conf'
[armadillo ~]# rc-service chronyd restart ❹
```

```
chronyd          | * Stopping chronyd ... [ ok ]
chronyd          | * Starting chronyd ... [ ok ]
armadillo:~# chronyc -n sources ❺
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* 192.0.2.1                     2   6    17    24    +11us[  +34us] +/-   53ms
```

図 3.240 chronyd のコンフィグの変更例

❶ 起動時のサーバー設定です。不要な場合は空のファイルを生成してください。

❷ 運用時のサーバー設定です。複数の行または「pool」の設定も可能です。

❸ ファイルを保存します。

❹ chronyd サービスを再起動します。

❺ chronyc で新しいサーバーが使用されていることを確認します。

NTP の設定は ABOS Web や Rest API を使って行うこともできます。詳細は、「6.12.5. 時刻設定」および 「6.12.6.13. Rest API : 時刻の設定」 を参照してください。

3.14. Armadillo Twin を体験する

Armadillo Twin を利用したデバイス運用管理を検討する場合は、一度 Armadillo Twin をお試しいただくことをおすすめします。Armadillo Twin は、無償トライアルでご登録いただくことで、3 ヶ月間無償で全ての機能をご利用いただくことができます。また、トライアル中の設定内容は、有料の月額プランに申込後も引き継いで利用することができます。

詳細は Armadillo Twin ユーザーマニュアル 「アカウント・ユーザーを作成する」 [https://manual.armadillo-twin.com/create-account-and-user/] をご確認ください。

3.15. ABOSDE によるアプリケーションの開発

ここでは、ABOSDE(Armadillo Base OS Development Environment) によるアプリケーション開発の概要と ABOSDE で作成される各プロジェクトの違いについて説明します。

ABOSDE は Visual Studio Code にインストールできる開発用エクステンションです。ABOSDE を使用することで、コンテナ及びコンテナ自動起動用設定ファイルの作成、コンテナ内におけるパッケージのインストール、コンテナ内で動作するアプリケーション本体の開発をすべて VS Code 内で行うことができます。

ABOSDE では、以下のようなアプリケーションを開発できます。

- ゲートウェイコンテナアプリケーション
- CUI アプリケーション
- C 言語アプリケーション

3.15.1. ABOSDE の対応言語

「表 3.65. ABOSDE の対応言語」に示すように、アプリケーション毎に対応している言語が異なります。

表 3.65 ABOSDE の対応言語

アプリケーションの種類	使用言語（フレームワーク）
ゲートウェイアプリケーション	Python
CUI アプリケーション	シェルスクリプト
	Python
C 言語アプリケーション	C 言語

3.15.2. 参照する開発手順の章の選択

どのようなアプリケーションを開発するかによって ABOSDE による開発手順が異なります。「図 3.241. 参照する開発手順の章を選択する流れ」を参考に、ご自身が開発するアプリケーションに適した章を参照してください。

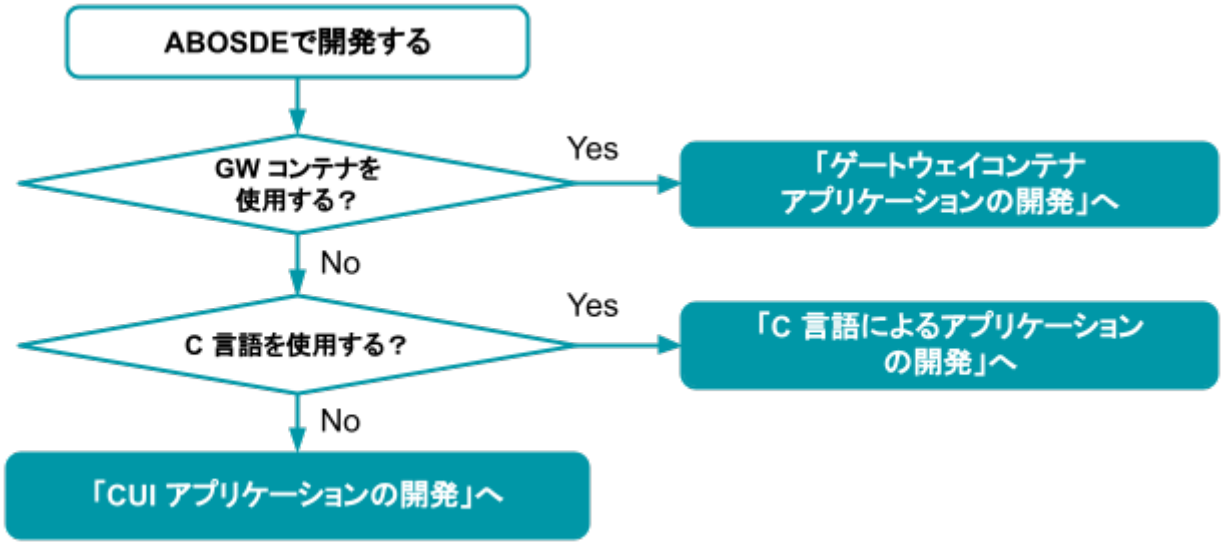


図 3.241 参照する開発手順の章を選択する流れ

ゲートウェイコンテナアプリケーション

- 対象ユーザー
  - 既存のゲートウェイコンテナを拡張したい
- マニュアルの参照先
  - 「3.16. ゲートウェイコンテナアプリケーションの開発」を参照

CUI アプリケーション

- 対象ユーザー
  - 画面を使用しないアプリケーションを開発したい
- マニュアルの参照先
  - 「3.17. CUI アプリケーションの開発」を参照

C 言語アプリケーション

- 対象ユーザー
  - C 言語でないと実現できないアプリケーションを開発したい

- 既存の C 言語によって開発されたアプリケーションを Armadillo で動作させたい
- 開発環境に制約がある

- マニュアルの参照先
  - 「3.18. C 言語によるアプリケーションの開発」を参照

3.16. ゲートウェイコンテナアプリケーションの開発

ATDE 上の VS Code でゲートウェイコンテナ内で動作するゲートウェイコンテナアプリケーションを開発する手順を示します。ゲートウェイコンテナについては、「6.10. ゲートウェイコンテナを動かす」を参照してください。また、以降の手順を行う前に、ゲートウェイコンテナを Armadillo にインストールしてください。

ゲートウェイコンテナのインストールは ABOS Web から可能です。「6.12.4. SWU インストール」をご参照ください。以下のゲートウェイコンテナの SWU イメージの URL を ABOS Web 上の「SWU URL」に入力して「インストール」ボタンを押してください。

https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-a6e/image/a6e-gw-container-latest.swu

ゲートウェイコンテナが Armadillo にインストールされます。

3.16.1. ゲートウェイコンテナアプリケーション開発の流れ

ゲートウェイコンテナアプリケーションを開発する流れは以下のようになります。

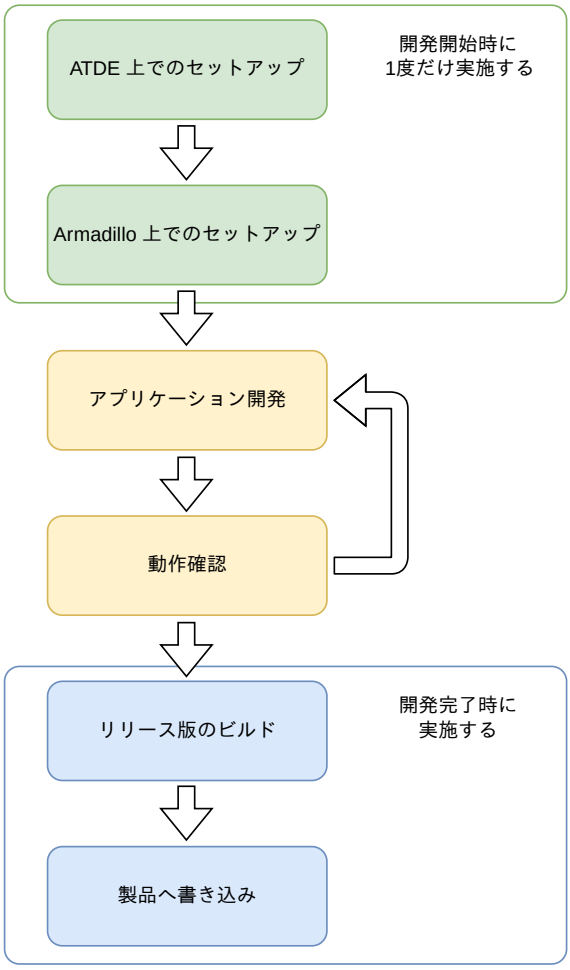


図 3.242 ゲートウェイコンテナアプリケーション開発の流れ

3.16.2. ATDE 上でのセットアップ

ここでは、開発開始時の ATDE 上でのセットアップ手順について説明します。本章は ATDE と VS Code のセットアップが完了していることを前提としております。セットアップがまだの方は、「3.1. 開発の準備」を参照してセットアップを完了してください。

3.16.2.1. プロジェクトの作成

VS Code の左ペインの [A6E] から [GW New Project] を実行し、表示されるディレクトリ選択画面からプロジェクトを保存するディレクトリを選択してください。実行するためには右に表示されている三角形ボタンを押してください。保存先を選択すると、プロジェクト名を入力するダイアログが表示されるので、任意のプロジェクト名を入力してエンターキーを押してください。この操作により、選択した保存先に、入力したプロジェクト名と同名のディレクトリが作成されます。

また、ここでは次のように設定しています。

- **保存先**：ホームディレクトリ
- **プロジェクト名**：`my_project`

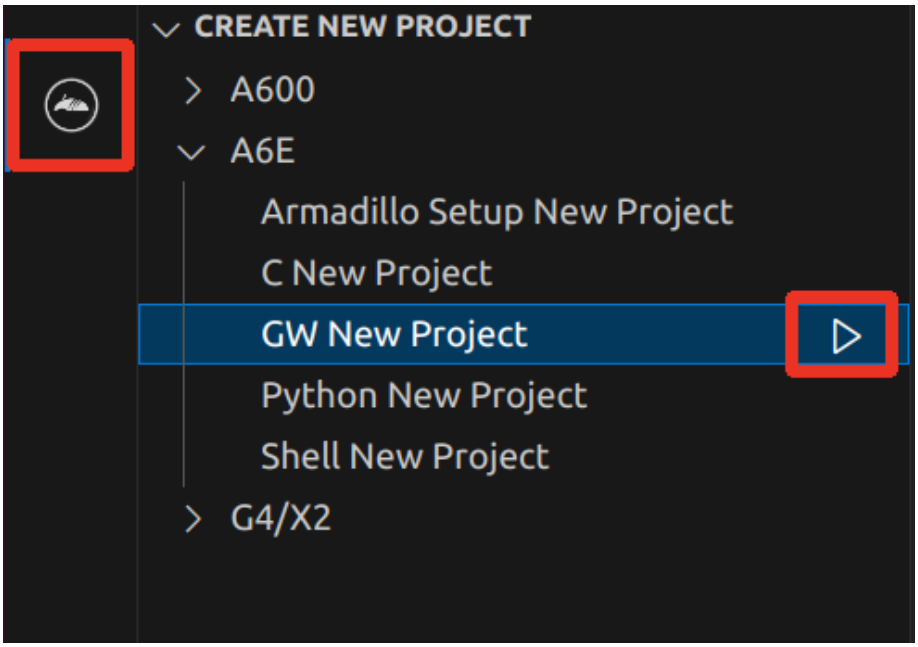


図 3.243 プロジェクトを作成する

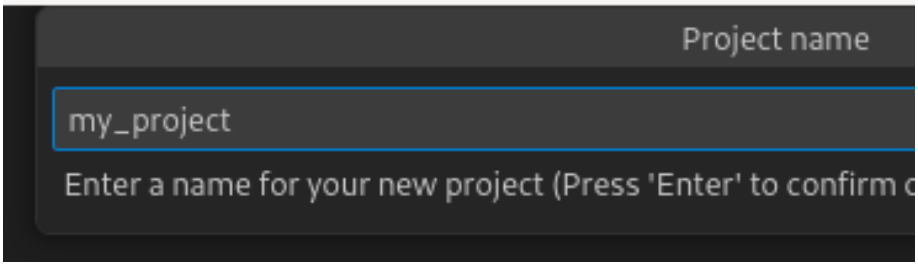


図 3.244 プロジェクト名を入力する

3.16.3. アプリケーション開発

3.16.3.1. VS Code の起動

ここでは、実際に Armadillo 上でサンプルアプリケーションを起動する場合の手順を説明します。プロジェクトディレクトリへ移動し VS Code を起動します。

```
[ATDE ~]$ code ./my_project
```

図 3.245 VS Code で `my_project` を起動する

3.16.3.2. ディレクトリ構成

プロジェクトには下記のディレクトリがあります。

- **app**：アプリケーションに直接関わるファイルが含まれるディレクトリです。
  - **config**：クラウド情報の設定ファイルとインターフェースの設定ファイルが配置されます。

- **example** : ゲートウェイコンテナアプリケーションの拡張例のサンプルファイルがあります。詳細は「6.11. ゲートウェイコンテナアプリケーションを改造する」を参照してください。
- **src** : ゲートウェイコンテナアプリケーションのソースファイルが配置されます。

- **config** : 設定に関わるファイルが含まれるディレクトリです。
  - **ssh_config** : Armadillo への ssh 接続に使用します。「3.16.7.2. ssh 接続に使用する IP アドレスの設定」 を参照してください。

3.16.3.3. 初期設定

初期設定では Armadillo と SSH で接続するための秘密鍵と公開鍵の生成を行います。

作成したプロジェクトディレクトリへ移動して VS Code を起動してください。

```
[ATDE ~]$ cd my_project
[ATDE ~/my_project]$ code ./
```

図 3.246 初期設定を行う

VS Code の左ペインの [my_project] から [Setup environment] を実行します。

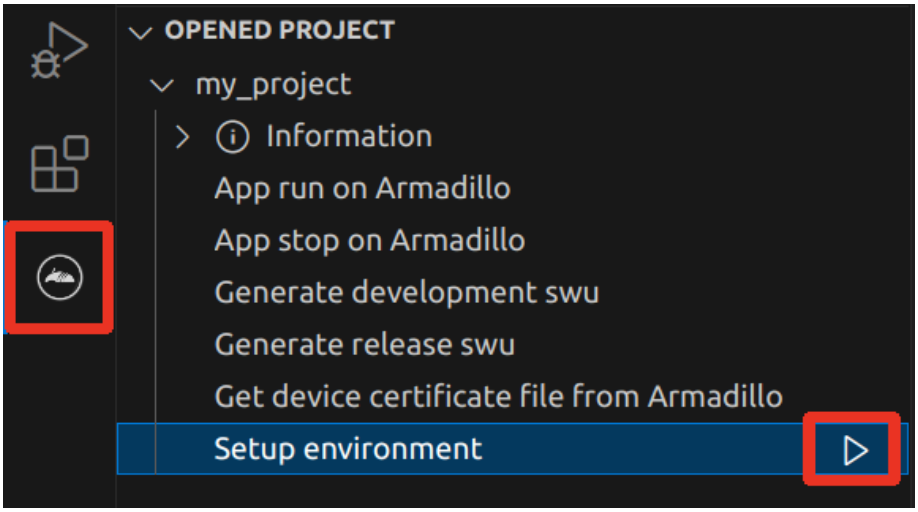


図 3.247 VS Code で初期設定を行う

選択すると、 VS Code の下部に以下のようなターミナルが表示されます。

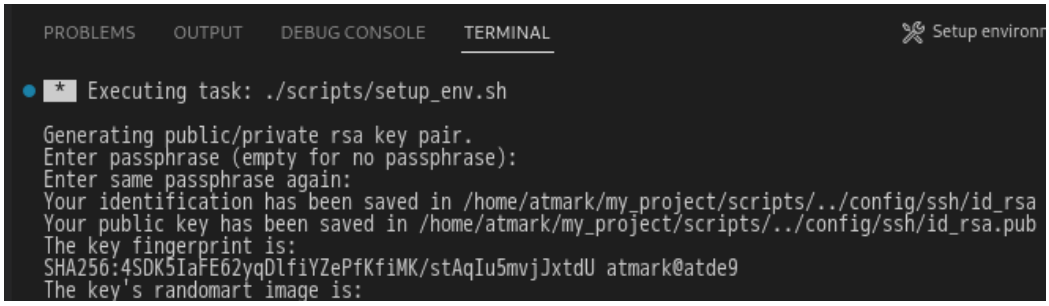


図 3.248 VS Code のターミナル

このターミナル上で以下のように入力してください。

```
 *  Executing task: ./scripts/setup_env.sh

Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase): ❶
Enter same passphrase again: ❷
Your identification has been saved in /home/atmark/.ssh/id_ed25519_vscode
:(省略)

 *  Terminal will be reused by tasks, press any key to close it.  ❸
```

図 3.249 SSH 用の鍵を生成する

❶ パスフレーズを設定します。設定しない場合は何も入力せず Enter を押します。

❷ 1 でパスフレーズを設定した場合は、確認のため再度入力してください。

❸ ここで何か任意のキーを押すとターミナルが閉じます。

パスフレーズを設定した場合は、アプリケーションを Armadillo へ転送する時にパスフレーズの入力を求められることがあります。



SSH の鍵は $HOME/.ssh/id_ed25519_vscode (と id_ed25519_vscode.pub) に保存されていますので、プロジェクトをバックアップする時は $HOME/.ssh も保存してください。

3.16.4. ゲートウェイコンテナアプリケーションの設定

ゲートウェイコンテナアプリケーションは、ゲートウェイコンテナ上で動作します。ゲートウェイコンテナについての詳細は 「6.10. ゲートウェイコンテナを動かす」 をご参照ください。

3.16.4.1. ゲートウェイコンテナの設定ファイルの編集

ゲートウェイコンテナの設定ファイルは app/config ディレクトリに配置されています。

- cloud_agent.conf: クラウド情報の設定
- sensing_mgr.conf: インターフェース設定

3.16.4.2. 接続先クラウド情報の設定

クラウドと連携する場合、接続先クラウドの情報を入力する必要があります。設定ファイルは Armadillo Base OS では /var/app/rollback/volumes/gw_container/config/cloud_agent.conf に存在し、VS Code では app/config/cloud_agent.conf に存在します。

```
[CLOUD]
SERVICE = ;AWS or AZURE or SORACOM
```

```
[LOG]
FILE_LOG = true
STREAM_LOG = true

[AWS]
AWS_IOT_HOST =
AWS_IOT_REGION =
AWS_IOT_ACCOUNTID =
AWS_IOT_ENDPOINT =
AWS_IOT_CERT_FILE = /cert/device/device_cert.pem
AWS_IOT_POLICY_FILE = /config/aws_iot_policy.json
AWS_IOT_SHADOW_ENDPOINT =
AWS_IOT_CA_FILE = /cert/ca/AmazonRootCA1.pem
AWS_IOT_PKCS11_PATH = /usr/lib/plug-and-trust/libsss_pkcs11.so
AWS_IOT_KEY_LABEL = sss:100100F0
AWS_ACCESS_KEY =
AWS_SECRET_KEY =
AWS_IOT_PORT = 443
AWS_IOT_PIN =

[AZURE]
AZURE_IOT_DEVICE_DPS_ENDPOINT = global.azure-devices-provisioning.net
AZURE_IOT_DEVICE_DPS_ID_SCOPE =
AZURE_IOT_KEY_FILE = /cert/device/key.pem
AZURE_IOT_CERT_FILE = /cert/device/device_cert.pem

[SORACOM]
SORACOM_HARVEST_ENDPOINT=http://harvest.soracom.io
```

図 3.250 /var/app/rollback/volumes/gw_container/config/cloud_agent.conf のフォーマット

・接続先の クラウドサービス 種別

ゲートウェイコンテナが接続するクラウドサービスの種別を指定します。設定ファイル中の以下の箇所が該当します。

```
[CLOUD]
SERVICE = ;AWS or AZURE or SORACOM
```

表 3.66 [CLOUD] 設定可能パラメータ

項目	概要	設定値	内容
SERVICE	接続先クラウドサービスを指定	AWS	AWS IoT Core に接続
		Azure	Azure IoT に接続
		SORACOM	SORACOM Harvest に接続

・ログ出力

クラウド との接続状態や送受信したデータのログを ログファイルに保存したり、コンソールに出力することが可能です。設定ファイル中の以下の箇所が該当します。

```
[LOG]
FILE_LOG = true
STREAM_LOG = true
```

表 3.67 [CLOUD] 設定可能パラメータ

項目	概要	設定値	内容
FILE_LOG	ログファイルに出力するか	(デフォルト)true	出力する
		false	出力しない
STREAM_LOG	コンソールに出力するか	(デフォルト)true	出力する
		false	出力しない

- AWS

ここでは、 AWS に接続する場合の設定内容を記載します。設定ファイル中の以下の箇所が該当します。

```
[AWS]
AWS_IOT_HOST =
AWS_IOT_REGION =
AWS_IOT_ACCOUNTID =
AWS_IOT_ENDPOINT =
AWS_IOT_CERT_FILE = /cert/device/device_cert.pem
AWS_IOT_POLICY_FILE = /config/aws_iot_policy.json
AWS_IOT_SHADOW_ENDPOINT =
AWS_IOT_CA_FILE = /cert/ca/AmazonRootCA1.pem
AWS_IOT_PKCS11_PATH = /usr/lib/plug-and-trust/libsss_pkcs11.so
AWS_IOT_KEY_LABEL = sss:100100F0
AWS_ACCESS_KEY =
AWS_SECRET_KEY =
AWS_IOT_PORT = 443
AWS_IOT_PIN =
```

表 3.68 [AWS] 設定可能パラメータ

項目	概要	設定値・設定例	取得方法
AWS_IOT_HOST	IoT Core REST API エンドポイント(リージョンに準ずる)	(例) iot.ap-northeast-1.amazonaws.com	AWS IoT Core - コントロールプレーンエンドポイント [https://docs.aws.amazon.com/ja_jp/general/latest/gr/iot-core.html] から取得
AWS_IOT_REGION	リージョン	(例) ap-northeast-1	AWS リージョンエンドポイント[https://docs.aws.amazon.com/ja_jp/general/latest/gr/rande.html] から取得
AWS_IOT_ACCOUNTID	アカウント ID	(例) 111111111111	AWS マネジメントコンソール上から取得(参考:「6.10.4.6. 設定に必要となるパラメータを取得する」)
AWS_IOT_ENDPOINT	AWS IoT Core エンドポイント(リージョンに準ずる)	(例) https://iot.ap-northeast-1.amazonaws.com	AWS IoT Core - コントロールプレーンエンドポイント [https://docs.aws.amazon.com/ja_jp/general/latest/gr/iot-core.html] から取得

項目	概要	設定値・設定例	取得方法
AWS_IOT_CERT_FILE	デバイス証明書ファイルパス	(デフォルト)/cert/device/device_cert.pem [a]	変更不要
AWS_IOT_POLICY_FILE	AWS IoT Core ポリシーテンプレートファイルパス	(デフォルト)/config/aws_iot_policy.json	変更不要
AWS_IOT_SHADOW_ENDPOINT	AWS IoT Core エンドポイント	(例)xxxxxxxxx-ats.iot.ap-northeast-1.amazonaws.com	AWS IoT Core [設定] - [デバイスデータエンドポイント] から取得 (参考:「6.10.4.6. 設定に必要となるパラメータを取得する」)
AWS_IOT_CA_FILE	AWS IoT Core ルート CA ファイルパス	(デフォルト)/cert/ca/AmazonRootCA1.pem [a]	変更不要
AWS_IOT_PKCS11_PATH	PKCS#11 ライブラリパス	(デフォルト)/usr/lib/plug-and-trust/libsss_pkcs11.so	変更不要
AWS_IOT_KEY_LABEL	利用する秘密鍵のラベル	(デフォルト)sss:100100F0	変更不要
AWS_ACCESS_KEY	アクセスキー	(例)AAAAAAAAAXXXXXXX	「6.10.4.3. アクセスキーを作成する」 でダウンロードした IAM ユーザー アクセスキー情報
AWS_SECRET_KEY	シークレットキー	(例)sssssssddddddttttttttt	「6.10.4.3. アクセスキーを作成する」 でダウンロードした IAM ユーザー アクセスキー情報
AWS_IOT_PORT	MQTT 接続ポート	(デフォルト)443	変更不要
AWS_IOT_PIN	PIN	-	指定不要

[a]ゲートウェイコンテナバージョン 2.1.1 でパスを変更しました



上記パラメータのうち、以下のパラメータは AWS IoT Core へのデバイス登録完了後クリアされます。デバイスを AWS IoT Core から削除した場合など再度デバイス登録を行いたい場合は、再度設定してください。

- AWS_IOT_ACCOUNTID
- AWS_ACCESS_KEY
- AWS_SECRET_KEY

- Azure

ここでは、 Azure に接続する場合の設定内容を記載します。設定ファイル中の以下の箇所が該当します。

```
[AZURE]
AZURE_IOT_DEVICE_DPS_ENDPOINT = global.azure-devices-provisioning.net
AZURE_IOT_DEVICE_DPS_ID_SCOPE =
AZURE_IOT_KEY_FILE = /cert/device/key.pem
AZURE_IOT_CERT_FILE = /cert/device/device_cert.pem
```

表 3.69 [AZURE] 設定可能パラメータ

項目	概要	設定値・設定例	取得方法
AZURE_IOT_DEVICE_DPS_ENDPOINT	DPS エンドポイント	(デフォルト)global.azure-devices-provisioning.net	変更不要
AZURE_IOT_DEVICE_DPS_ID_SCOPE	Azure IoT Central ID スコープ	(例)0ne12345678	「図 6.98. Azure IoT Hub と DPS の設定を実行する」 で表示された内容を使用
AZURE_IOT_KEY_FILE	デバイスリファレンスキーファイルパス	(デフォルト)/cert/device/key.pem [a]	変更不要
AZURE_IOT_CERT_FILE	デバイス証明書ファイルパス	(デフォルト)/cert/device/device_cert.pem [a]	変更不要

[a]ゲートウェイコンテナバージョン 2.1.1 でパスを変更しました。

・ SORACOM

ここでは、 SORACOM に接続する場合の設定内容を記載します。設定ファイル中の以下の箇所が該当します。

```
[SORACOM]
SORACOM_HARVEST_ENDPOINT=http://harvest.soracom.io
```

表 3.70 [SORACOM] 設定可能パラメータ

項目	概要	設定値・設定例	取得方法
SORACOM_HARVEST_ENDPOINT	SORACOM Harvest エンドポイント	(デフォルト)http://harvest.soracom.io	変更不要

3.16.4.3. インターフェース設定

インターフェースの動作設定を行います。設定ファイルは Armadillo Base OS では、`/var/app/rollback/volumes/gw_container/config/sensing_mgr.conf`に存在し、VS Code では `app/config/sensing_mgr.conf`に存在します。

```
[DEFAULT]
; cloud_config=true or false
cloud_config=false
; send_cloud=true or false
send_cloud=false
; cache=true or false
cache=false
; send_interval[sec]
send_interval=10
; data_send_oneshot=true or false
data_send_oneshot=false
; wait_container_stop[sec]
wait_container_stop=0

[LOG]
file=true
stream=true

[CPU_temp]
; type=polling or none
```

```
type=polling
; polling_interval[sec]
polling_interval=1

[DI1]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=

[DI2]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=

[DO1]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=

[DO2]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=

[RS485_Data1]
;[RS485_Data1] ~ [RS485_Data4]
method=none
baudrate=
data_size=
; parity=none or odd or even
parity=
; stop=1 or 2
stop=
device_id=
func_code=
register_addr=
register_count=
; endian=little or big
endian=
; interval[sec]
interval=
; data_offset is option
data_offset=
; data_multiply is option
data_multiply=
```

```
; data_divider is option
data_divider=
```

図 3.251 /var/app/rollback/volumes/gw_container/config/sensing_mgr.conf のフォーマット

・全体動作設定

全体的な動作設定を行います。設定ファイル中の以下の箇所が該当します。

```
[DEFAULT]
; cloud_config=true or false
cloud_config=false
; send_cloud=true or false
send_cloud=false
; cache=true or false
cache=false
; send_interval[sec]
send_interval=10
; data_send_oneshot=true or false
data_send_oneshot=false
; wait_container_stop[sec]
wait_container_stop=0
```

表 3.71 [DEFAULT] 設定可能パラメータ

項目	概要	設定値	内容
cloud_config	クラウドからの設定変更を許容するか	true	許容する
		(デフォルト)false	無視する
send_cloud	クラウドにデータを送信するか	true	送信する
		(デフォルト)false	送信しない
cache	キャッシュ実施可否	true	キャッシュを実施する。ネットワーク状態の異常などによりクラウドへデーターを送信できない場合、キャッシュに計測データ―を一時保存し、ネットワーク復旧後にクラウドへ送信します。
		(デフォルト)false	キャッシュを実施しない
send_interval	データ送信間隔[sec]	1～10	この値に従って、クラウドへデータを送信する
data_send_oneshot	データ取得後コンテナを終了させるか	true	1 回データを取得し、コンテナを終了します。コンテナ終了通知をトリガに間欠動作を行う(「6.1.4. 状態遷移トリガにコンテナ終了通知を利用する」)場合は、この設定にする必要があります。
		(デフォルト)false	コンテナの実行を継続する (設定したインターバルでデータを取得する)
wait_container_stop	コンテナ終了までの待ち時間[sec]	0～60	data_send_oneshot が true の場合、クラウドへのデータ送信後、設定した時間 wait してからコンテナを終了する [a]

[a]現時点では 0 を設定してください



クラウドへのデータ送信は send_interval で指定した間隔毎に行います。値の取得間隔は、後述の通り各項目毎に指定することができます。値を取得するタイミングとクラウドへのデータ送信のタイミングを近くするためには、値の取得間隔より send_interval を短くするか、同じにすることを推奨します。

・ログ出力

取得したデータのログを ログファイルに保存したり、コンソールに出力することが可能です。設定ファイル中の以下の箇所が該当します。

```
[LOG]
file=true
stream=true
```

表 3.72 [LOG] 設定可能パラメータ

項目	概要	設定値	内容
FILE_LOG	ログファイルに出力するか	(デフォルト)true	出力する
		false	出力しない
STREAM_LOG	コンソールに出力するか	(デフォルト)true	出力する
		false	出力しない

・CPU_temp

CPU 温度読み出しに関する設定を行います。設定ファイル中の以下の箇所が該当します。

```
[CPU_temp]
; type=polling or none
type=polling
; polling_interval[sec]
polling_interval=1
```

表 3.73 [CPU_temp] 設定可能パラメータ

項目	概要	設定値	内容
type	動作種別	(空欄) or none	CPU 温度取得を行わない
		polling	ポーリング
polling_interval	データ取得間隔[sec]	1～3600	この値に従って、CPU 温度を読み出します

・接点入力

接点入力に関する設定を行います。設定ファイル中の以下の箇所が該当します。

```
[DI1]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
```

```
edge_type=

[DI2]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=
```

表 3.74 [DI1,DI2] 設定可能パラメータ

項目	概要	設定値	内容
type	動作種別	(空欄) or none	接点入力状態取得を行わない
		polling	ポーリング
		edge	エッジ検出。データ取得間隔に設定した周期で値を取得し、前回取得時から指定方向に値が変化した場合、クラウドへデータ―を送信します。
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します
edge_type	エッジ検出設定	falling	立ち下がりエッジ
		rising	立ち上がりエッジ
		both	両方

・接点出力

接点出力に関する設定を行います。設定ファイル中の以下の箇所が該当します。「表 3.71. [DEFAULT] 設定可能パラメータ」 において、クラウドと通信しない場合はゲートウェイコンテナ起動後に設定した内容を出力します。クラウドと通信する場合は、「6.10.9. クラウドからの操作」がトリガとなり、出力を開始します。

```
[DO1]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=

[DO2]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=
```

表 3.75 [DO1,DO2] 設定可能パラメータ

項目	概要	設定値	内容
output_state	出力状態	high	High 出力。クラウドからの設定内容更新が有効の場合に、クラウドからの設定変更が可能です。
		low	Low 出力。クラウドからの設定内容更新が有効の場合に、クラウドからの設定変更が可能です。
		disable	「6.23. 電源を安全に切るタイミングを通知する」 で接点出力を使用する場合など、ゲートウェイコンテナで接点出力を使用しないときに設定します。また、この値に設定すると、クラウドからの設定変更・動作指示は無視されます。
		指定なし	ゲートウェイコンテナで接点出力の初期状態を設定しないときに使用します。接点出力を設定しないため、ゲートウェイコンテナ起動時の出力状態になります。クラウドからの設定内容更新が有効の場合に、クラウドからの設定変更が可能です。
output_time	出力時間[sec]	1～3600	出力コマンド実行後に output_state で指定したレベルを出力する時間。 0 を指定すると出力値を固定します。
output_delay_time	出力遅延時間[sec]	0～3600	出力コマンド実行後、指定した時間遅延して出力します。

設定と DO の出力タイミングの関連を 「図 3.252. DO の出力タイミング」 に示します。

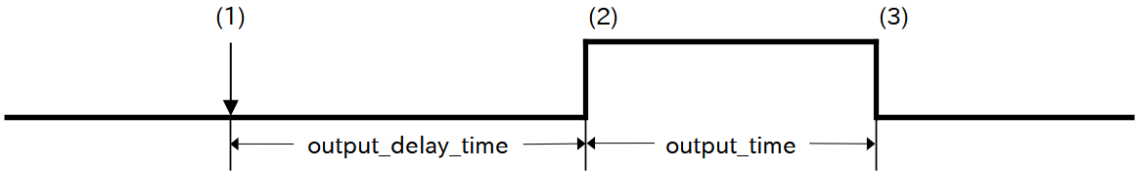


図 3.252 DO の出力タイミング

(1) ゲートウェイコンテナはクラウドからの要求を取り込みます

(2) クラウドからの要求を取り込んでから output_delay_time 経過後、出力を切り替えます

(3) output_time 経過後出力を戻します

- RS-485

RS-485 に関する設定を行います。設定ファイル中の以下の箇所が該当します。なお、RS485_Data1 から RS485_Data4 まで、4 個のデータについて設定することができます。デフォルトでは RS485_Data1 のみファイルに記載されているため、RS485_Data2, RS485_Data3, RS485_Data4 については適宜コピーして記載してください。

```
[RS485_Data1]
;[RS485_Data1] ~ [RS485_Data4]
method=none
baudrate=
data_size=
; parity=none or odd or even
parity=
; stop=1 or 2
stop=
device_id=
func_code=
register_addr=
register_count=
; endian=little or big
endian=
; interval[sec]
interval=
; data_offset is option
data_offset=
; data_multiply is option
data_multiply=
; data_divider is option
data_divider=
```

表 3.76 [RS485_Data1, RS485_Data2, RS485_Data3, RS485_Data4] 設定可能パラメータ

項目	概要	設定値	内容
method	通信種別	none	RS-485 を利用しない
		rtu	Modbus-RTU
data_size	データサイズ	7 or 8	
baudrate	ボーレート	300～921600[bps]	通信速度を指定します
parity	パリティビット	none	None
		odd	Odd
		even	Even
stop	ストップビット	1	1
		2	2
device_id	Modbus スレーブ機器のデバイス ID	0x01 ～ 0xF7	
func_code	ファンクションコード	0x01 ～ 0x04	
register_addr	レジスタアドレス	機器依存	値を読み出すレジスタのアドレスを指定
register_count	読み出しレジスタ数	1 or 2	一度に読み出すレジスタ数を指定
endian	エンディアン設定	little	リトルエンディアン
		big	ビッグエンディアン
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します
data_offset	読み出し値に加算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値に加算する値を指定します
data_multiply	読み出し値と乗算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と乗算する値を指定します
data_divider	読み出し値と除算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と除算する値を指定します

3.16.4.4. 開発用の SWU イメージの作成

Armadillo 上でゲートウェイコンテナアプリケーションを実行するために、ゲートウェイコンテナアプリケーションのソースファイルと設定ファイル、SSH の公開鍵を含む SWU イメージを作成します。SWU イメージを使用して Armadillo へインストールするため、事前に 「5.4.1. SWU イメージの作成」を参照して SWU の初期設定を行ってください。

SWU イメージの作成も VS Code で行います。VS Code の左ペインの [my_project] から [Generate development swu] を実行します。

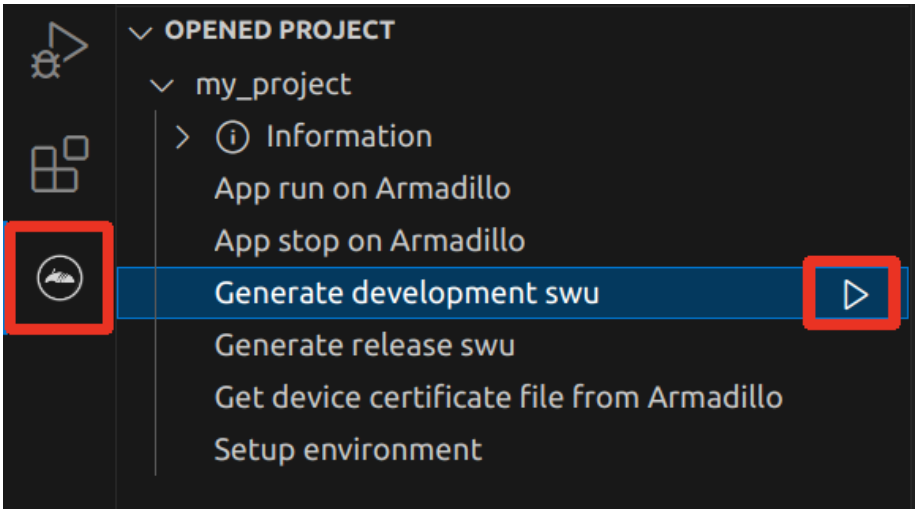


図 3.253 VS Code で開発用の SWU の作成を行う

SWU イメージの作成にはしばらく時間がかかります。VS Code のターミナルに以下のように表示されるとコンテナイメージの作成は完了です。

```
./swu/app.desc のバージョンを 0 から 1 に変更しました。
./development.swu を作成しました。
次は Armadillo に ./development.swu をインストールしてください。
 *  Terminal will be reused by tasks, press any key to close it.
```

図 3.254 開発用の SWU の作成完了

作成した SWU イメージは `my_project` ディレクトリ下に `development.swu` というファイル名で保存されています。

3.16.5. ゲートウェイコンテナのディストリビューション

ゲートウェイコンテナのディストリビューションは以下のとおりです。

ディストリビューション	・ alpine

3.16.6. Armadillo に転送するディレクトリ及びファイル

以下に示すディレクトリやファイルを Armadillo に転送します。ここでは、プロジェクト名は `my_project` としています。

Armadillo に転送するディレクトリ及びファイル	・ my_project/app/config/sensing_mgr.conf ・ my_project/app/config/cloud_agent.conf ・ my_project/app/src

3.16.6.1. ゲートウェイコンテナアプリケーションが使用するデバイス証明書の取得

「図 3.255. Armadillo 上でゲートウェイコンテナアプリケーションを実行する」に示すように、VS Code の左ペインの [my_project] から [Get device certificate file from Armadillo] を実行すると、ゲートウェイコンテナアプリケーションが使用するデバイス証明書を取得することができます。取得したデバイス証明書は `app/device/cert` ディレクトリに保存されます。

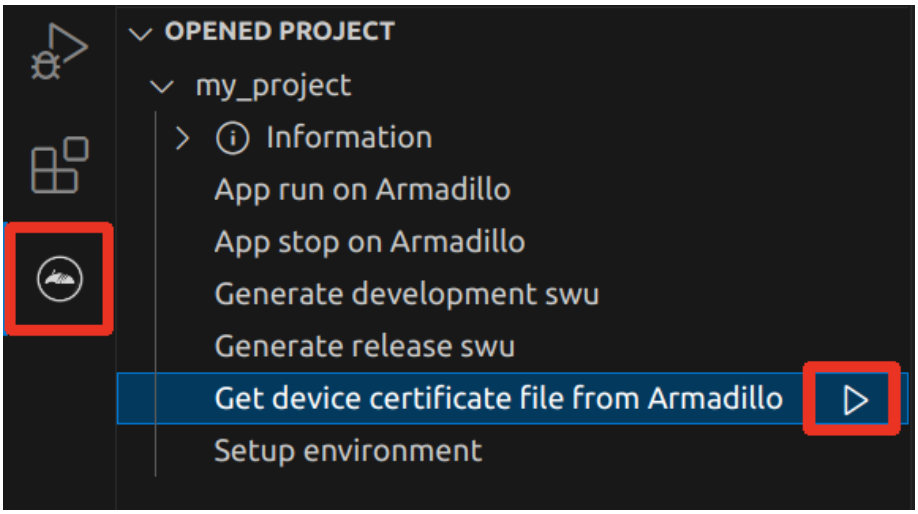


図 3.255 Armadillo 上でゲートウェイコンテナアプリケーションを実行する



このタスクは、「6.10.5.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 のデバイス証明書を取得する箇所で使用します。

3.16.7. Armadillo 上でのセットアップ

3.16.7.1. ゲートウェイコンテナアプリケーションのインストール

「3.16.4.4. 開発用の SWU イメージの作成」 で作成した `development.swu` を 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールしてください。

インストール後に自動で Armadillo が再起動します。

3.16.7.2. ssh 接続に使用する IP アドレスの設定

VS Code 上で ABOSDE(Armadillo Base OS Development Environment) から、ABOS Web が動作している Armadillo の一覧を確認し、指定した Armadillo の IP アドレスを ssh 接続に使用することができます。ただし、ATDE のネットワークを NAT に設定している場合は Armadillo がリストに表示されません。

「図 3.256. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲われているボタンをクリックすることで、ローカルネットワーク上で ABOS Web が実行されている Armadillo をスキャンすることができます。

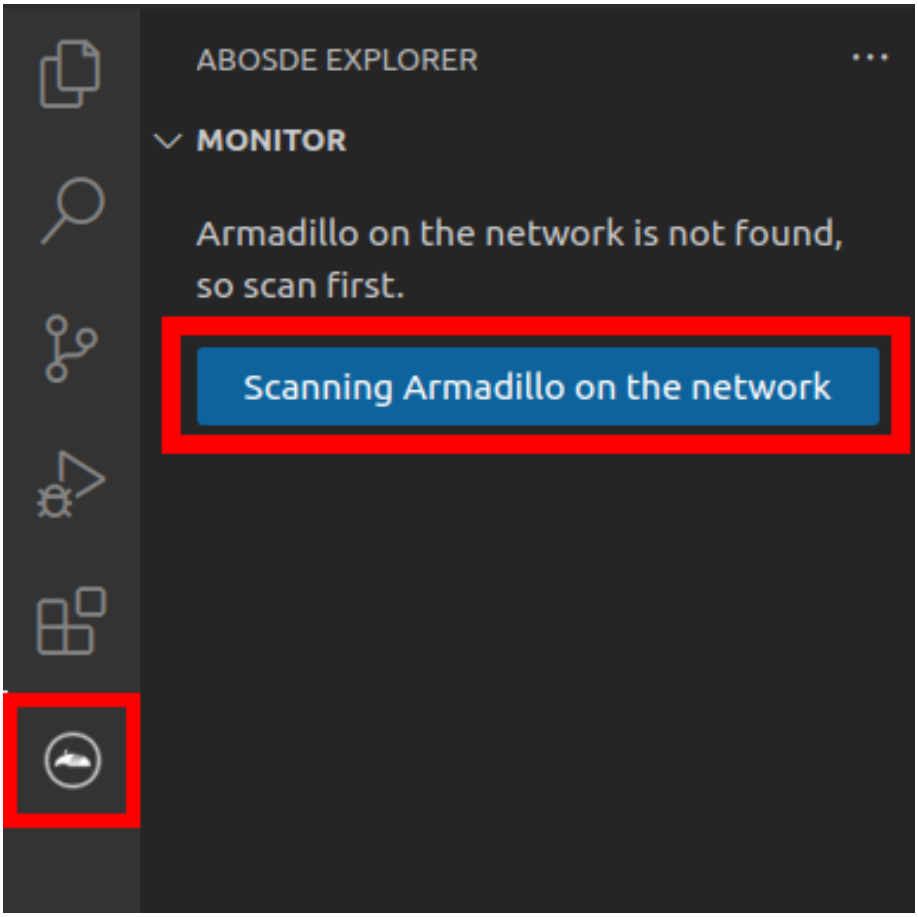


図 3.256 ABOSDE で ローカルネットワーク上の Armadillo をスキャンする

「図 3.257. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する」 の赤枠で囲われているマークをクリックすることで、指定した Armadillo の IP アドレスを ssh 接続に使用する IP アドレスに設定することができます。

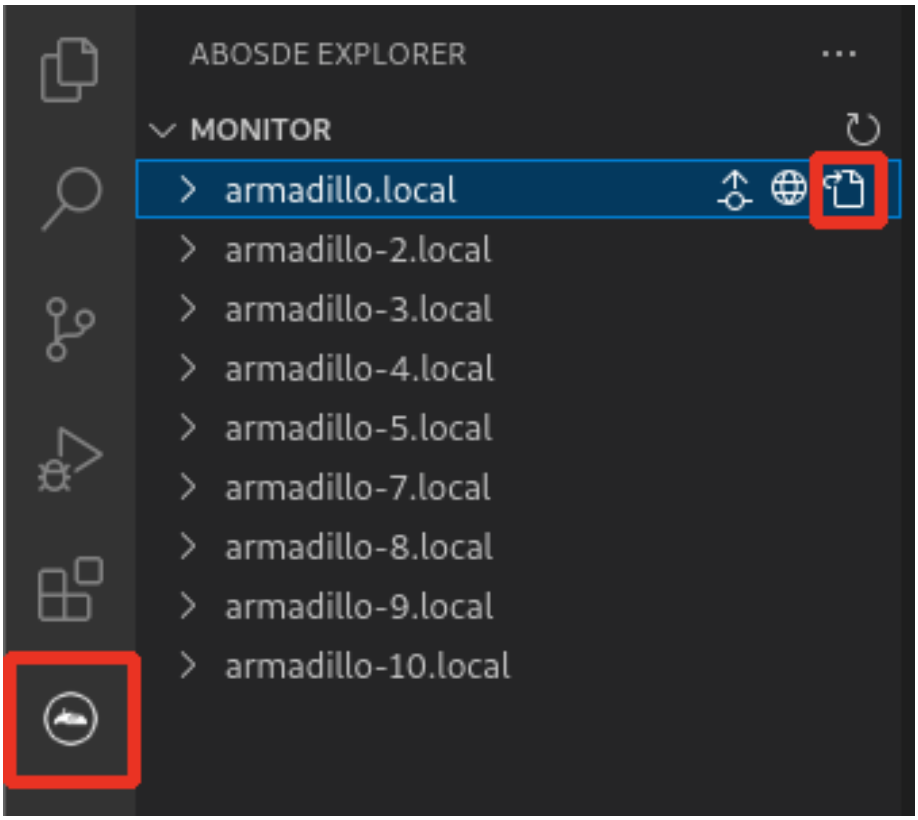


図 3.257 ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する

「図 3.258. ABOSDE に表示されている Armadillo を更新する」 の赤枠で囲われているマークをクリックすることで、ABOSDE に表示されている Armadillo を更新することができます。

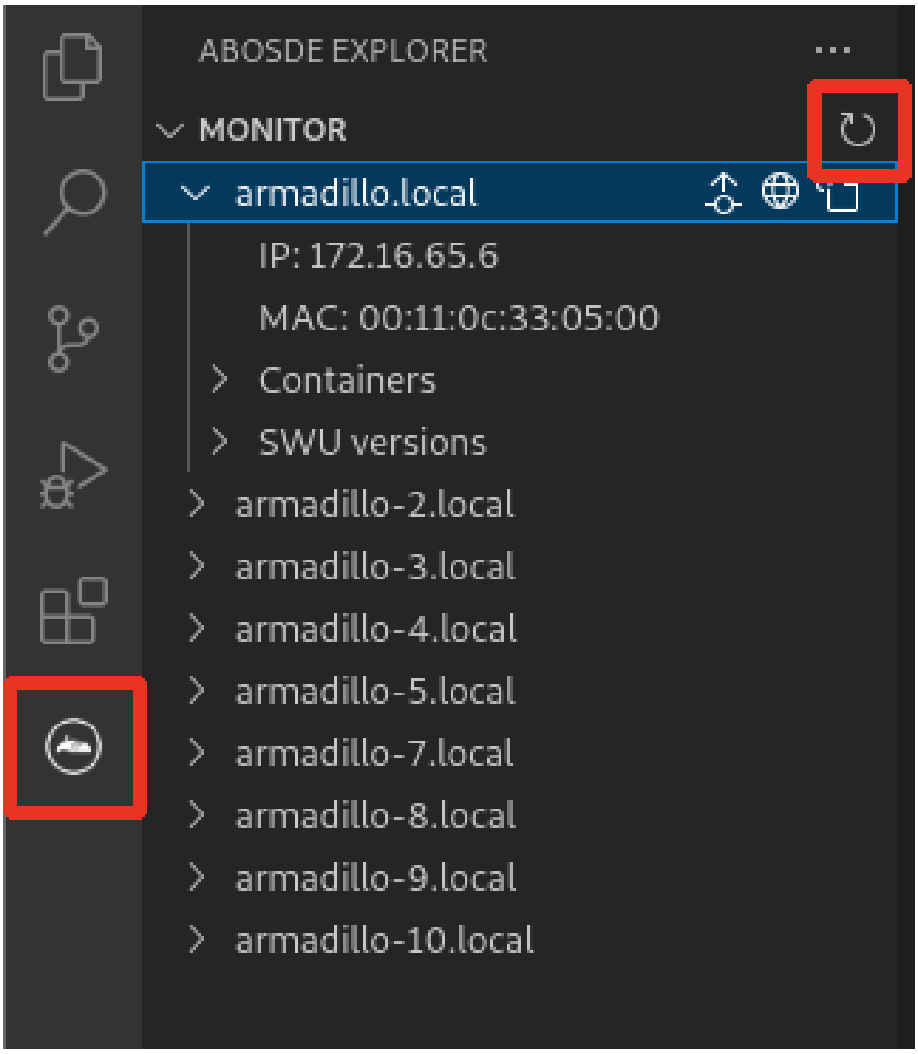


図 3.258 ABOSDE に表示されている Armadillo を更新する

ATDE のネットワークを NAT に設定している場合や、ABOS Web を起動していない場合等、ABOSDE のリストに Armadillo が表示されない場合は、プロジェクトディレクトリに入っている config/ssh_config ファイルを編集して IP アドレスを書き換えてください。

```
[ATDE ~/my_project]$ code config/ssh_config
Host Armadillo
    Hostname x.x.x.x ❶
    User root
    IdentityFile ${HOME}/.ssh/id_ed25519_vscode
    UserKnownHostsFile config/ssh_known_hosts
    StrictHostKeyChecking accept-new
```

図 3.259 ssh_config を編集する

❶ Armadillo の IP アドレスに置き換えてください。



Armadillo を初期化した場合や、プロジェクトを実行する Armadillo を変更した場合は、プロジェクトの config/ssh_known_hosts に保存されてい

る公開鍵で Armadillo を認識できなくなります。その場合はファイルを削除するか、「Setup environment」タスクを再実行してください。

3.16.7.3. ゲートウェイコンテナアプリケーションの実行

VS Code の左ペインの [my_project] から [App run on Armadillo] を実行すると、ゲートウェイコンテナアプリケーションが Armadillo へ転送されて起動します。

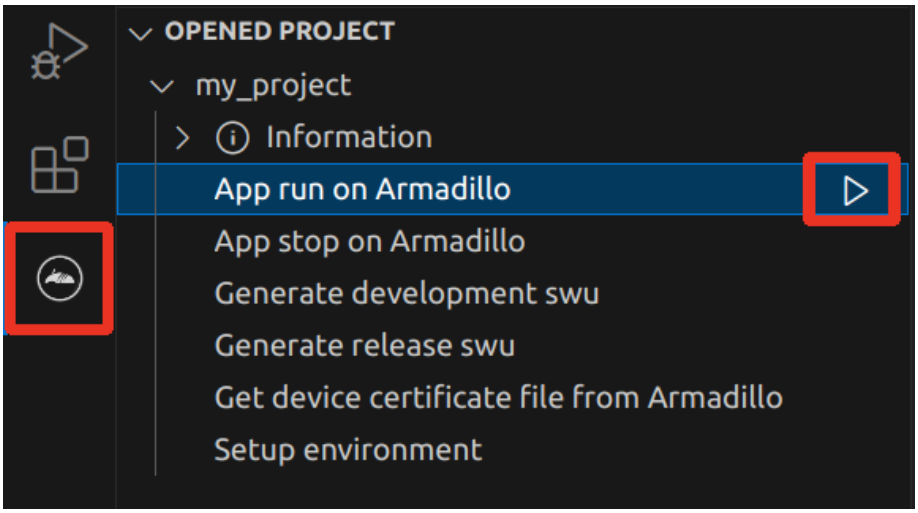


図 3.260 Armadillo 上でゲートウェイコンテナアプリケーションを実行する

VS Code のターミナルに以下のメッセージが表示されることがあります。これが表示された場合は yes と入力して下さい。

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

図 3.261 実行時に表示されるメッセージ

ゲートウェイコンテナアプリケーションを終了するには VS Code の左ペインの [my_project] から [App stop on Armadillo] を実行してください。

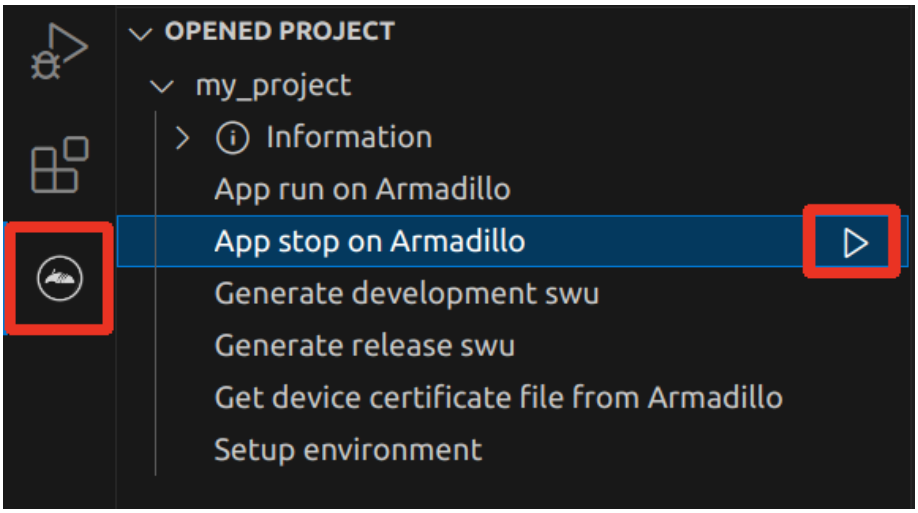


図 3.262 ゲートウェイコンテナアプリケーションを終了する

3.16.8. SBOM 生成に関する設定

SWU イメージ作成時に、同時に SBOM を生成することができます。詳細は 「3.19. SBOM 生成に関わる設定を行う」 を参照してください。ゲートウェイコンテナについての SBOM は **Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ** https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container をご確認ください。

3.16.9. リリース版のビルド

ここでは完成したゲートウェイコンテナアプリケーションをリリース版としてビルドする場合の手順について説明します。

VS Code の左ペインの [my_project] から [Generate release swu] を実行すると、リリース版のゲートウェイコンテナアプリケーションを含んだ SWU イメージが作成されます。事前に 「5.4.1. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。

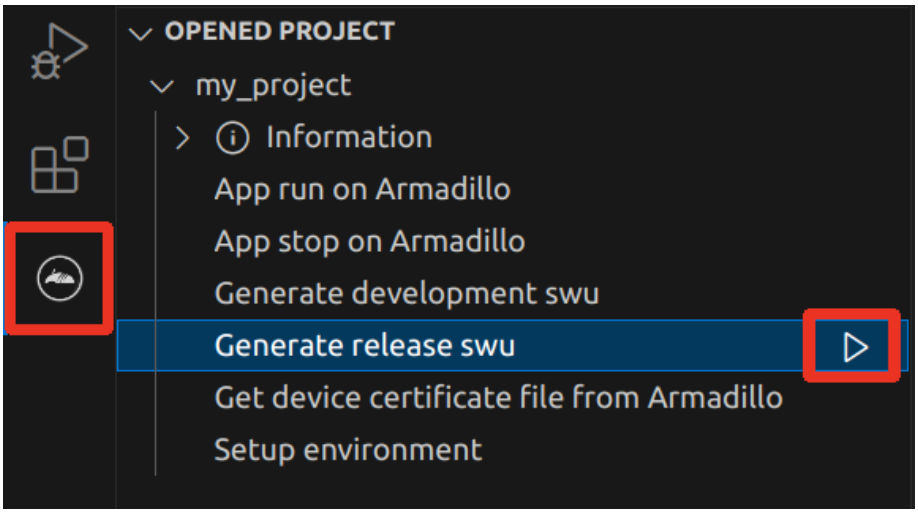


図 3.263 リリース版をビルドする

3.16.10. 製品への書き込み

作成した SWU イメージは `my_project` ディレクトリ下に `release.swu` というファイル名で保存されています。

この SWU イメージを 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールすると、Armadillo 起動時にゲートウェイコンテナアプリケーションも自動起動します。

3.16.11. Armadillo 上のゲートウェイコンテナイメージの削除

Armadillo 上のゲートウェイコンテナイメージを削除する方法は、 「6.9.3.1. VS Code から実行する」を参照してください。

ゲートウェイコンテナイメージを再インストールする場合は **Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ** https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container からゲートウェイコンテナイメージの SWU イメージファイルをダウンロードした後、「3.3.4.5. SWU イメージのインストール」 を参照してください。

3.16.12. クラウドを含めた動作確認

クラウドを含めた動作確認方法は「6.10. ゲートウェイコンテナを動かす」を参照ください。

3.17. CUI アプリケーションの開発

ここではシェルスクリプトおよび Python を使った CUI アプリケーションの開発方法を紹介します。開発手順としてはシェルスクリプトと Python で同じであるため、シェルスクリプトの場合の例で説明します。

3.17.1. CUI アプリケーション開発の流れ

Armadillo 向けに CUI アプリケーションを開発する場合の流れは以下のようになります。

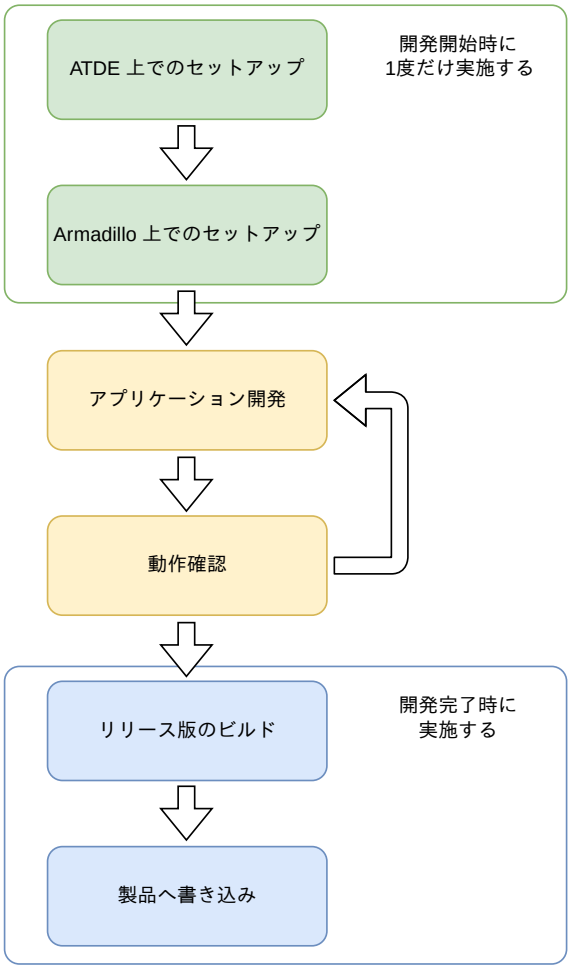


図 3.264 CUI アプリケーション開発の流れ

3.17.2. ATDE 上でのセットアップ

ここでは、開発開始時の ATDE 上でのセットアップ手順について説明します。ATDE をお使いでない場合は、先に 「3.1. 開発の準備」 を参照して ATDE 及び、 VS Code のセットアップを完了してください。

3.17.2.1. プロジェクトの作成

VS Code の左ペインの [A6E] から [Shell New Project] を実行し、表示されるディレクトリ選択画面からプロジェクトを保存するディレクトリを選択してください。実行するためには右に表示されている三角形ボタンを押してください。Python の場合は [Python New Project] を実行してください。保存先を選択すると、プロジェクト名を入力するダイアログが表示されるので、任意のプロジェクト名を入力してエンターキーを押してください。この操作により、選択した保存先に、入力したプロジェクト名と同名のディレクトリが作成されます。

また、ここでは次のように設定しています。

- **保存先** : ホームディレクトリ
- **プロジェクト名** : `my_project`

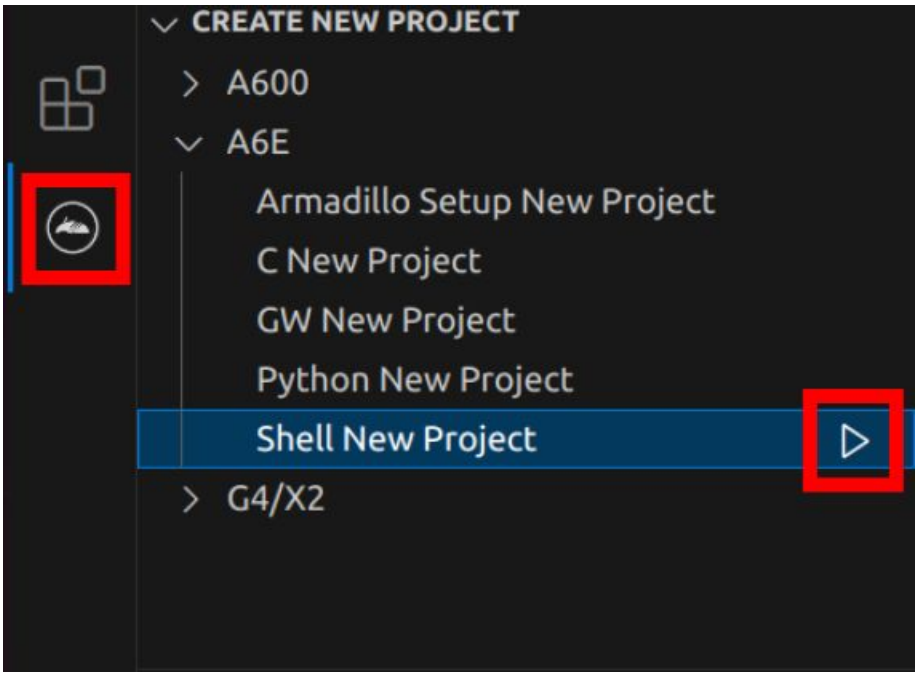


図 3.265 プロジェクトを作成する

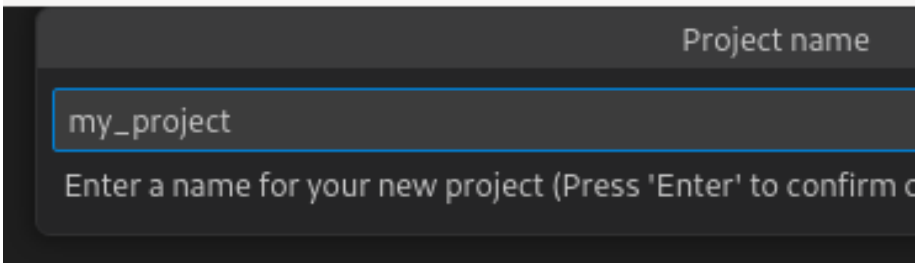


図 3.266 プロジェクト名を入力する

3.17.3. アプリケーション開発

3.17.3.1. VS Code の起動

ここでは、実際に Armadillo 上でサンプルアプリケーションを起動する場合の手順を説明します。プロジェクトディレクトリへ移動し VS Code を起動します。

```
[ATDE ~]$ code ./my_project
```

図 3.267 VS Code で `my_project` を起動する

3.17.3.2. ディレクトリ構成

プロジェクトには下記のディレクトリがあります。

- **app**：アプリケーションのソースです。Armadillo ではビルドしたアプリケーションが `/var/app/rollback/volumes/my_project` にコピーされます。

    - **requirements.txt** : Python プロジェクトにのみ存在しており、このファイルに記載したパッケージは pip を使用してインストールされます。

- **config** : 設定に関わるファイルが含まれるディレクトリです。

    - **app.conf** : コンテナのコンフィグです。記載内容については 「6.9.4. コンテナ起動設定ファイルを作成する」 を参照してください。

    - **app.desc** : SWU イメージを生成するための .desc ファイルです。記載内容については 「6.4. mkswu の .desc ファイルを編集する」 を参照してください。

    - **ssh_config** : Armadillo への ssh 接続に使用します。「3.17.6.2. ssh 接続に使用する IP アドレスの設定」 を参照してください。

- **container** : スクリプトを実行するコンテナの設定ファイルが含まれるディレクトリです。

    - **packages.txt** : このファイルに記載されているパッケージがインストールされます。

    - **Dockerfile** : 直接編集することも可能です。

    - **resources**: Armadillo の ルートディレクトリ（ / ）配下に同様のディレクトリ構造でコピーされます。

    - **resources_python**: resources と同様ですが、Python プロジェクト固有のファイルが配置されています。

        - **bin/python_launch**: Python プロジェクトの場合に、コンテナ起動時に実行するスクリプトです。

デフォルトのコンテナコンフィグ（ app.conf ）ではシェルスクリプトの場合は app の src/main.sh または Python の場合 src/main.py を実行しますので、リネームが必要な場合にコンテナのコンフィグも修正してください。

このサンプルアプリケーションは、CPU と SOC の温度を /vol_data/log/temp.txt に出力し、アプリケーション LED を点滅させます。

3.17.3.3. 初期設定

初期設定では主に Armadillo と SSH で接続するための秘密鍵と公開鍵の生成を行います。

作成したプロジェクトディレクトリへ移動して VS Code を起動してください。

```
[ATDE ~]$ cd my_project
[ATDE ~/my_project]$ code ./
```

図 3.268 初期設定を行う

VS Code の左ペインの [my_project] から [Setup environment] を実行します。

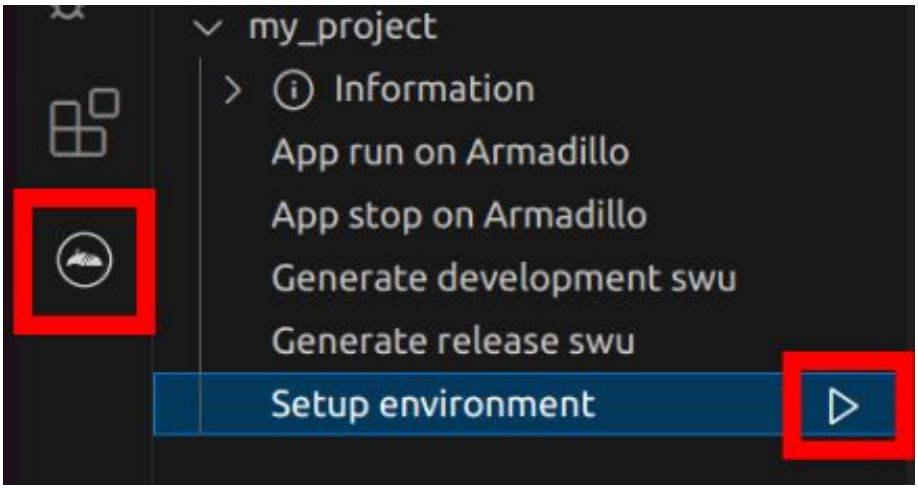


図 3.269 VS Code で初期設定を行う

選択すると、 VS Code の下部に以下のようなターミナルが表示されます。

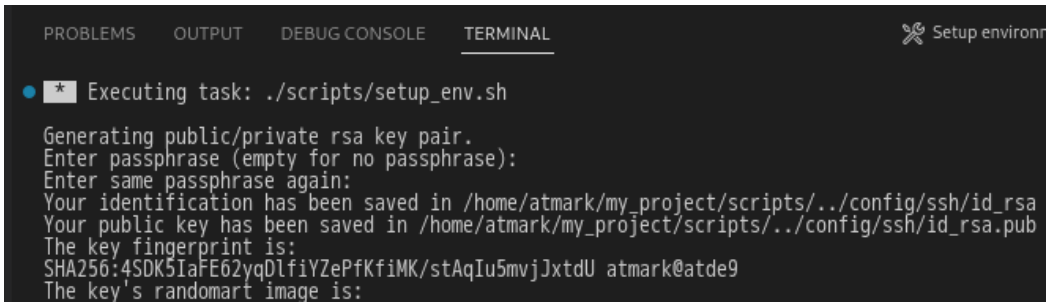


図 3.270 VS Code のターミナル

このターミナル上で以下のように入力してください。

```
 *  Executing task: ./scripts/setup_env.sh

Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase): ❶
Enter same passphrase again: ❷
Your identification has been saved in /home/atmark/.ssh/id_ed25519_vscode
:(省略)

 *  Terminal will be reused by tasks, press any key to close it.  ❸
```

図 3.271 SSH 用の鍵を生成する

❶ パスフレーズを設定します。設定しない場合は何も入力せず Enter を押します。

❷ 1 でパスフレーズを設定した場合は、確認のため再度入力してください。

❸ ここで何か任意のキーを押すとターミナルが閉じます。

パスフレーズを設定した場合は、アプリケーションを Armadillo へ転送する時にパスフレーズの入力を求められることがあります。



ssh の鍵は $HOME/.ssh/id_ed25519_vscode (と id_ed25519_vscode.pub) に保存されていますので、プロジェクトをバックアップする時は $HOME/.ssh も保存してください。

3.17.3.4. ABOSDE におけるコンテナ OS の選択方法



コンテナ OS を Debian から Alpine に変更する ABOSDE の機能は ABOSDE のバージョン 1.9.3 以降で、かつ 2025 年 6 月 25 日以降に「3.17.2.1. プロジェクトの作成」 の手順で新たに作成したプロジェクトで使用できるようになります。

ABOSDE では、コンテナ OS として Debian と Alpine を提供しています。デフォルトでは、Alpine よりも開発が容易である Debian が使用されます。

Alpine を使用する場合は、VS Code の左ペインの [OPENED PROJECT] の [Generate development swu] または [Generate release swu] を開き、[Use alpine container] を選択してください。

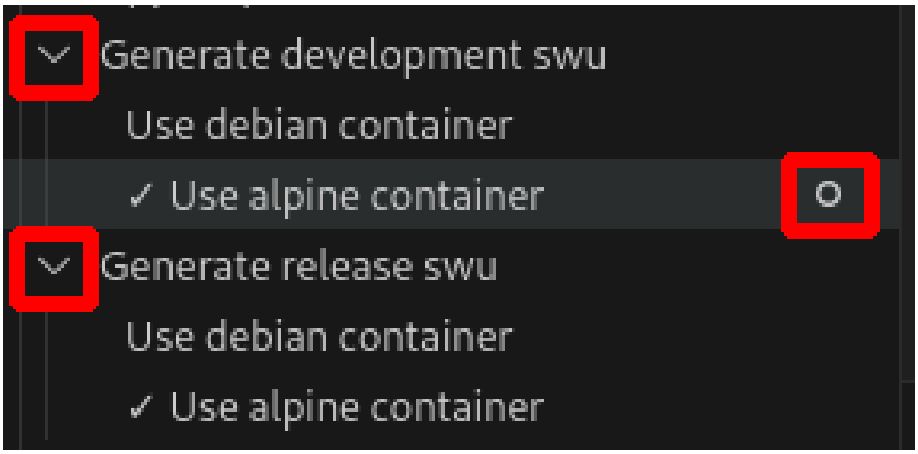


図 3.272 ABOSDE でコンテナ OS を選択する

SWU イメージを生成する時に以下のファイルが使用されます。使用するコンテナ OS にあわせて、対応するファイルを編集してください。

[Use debian container] を選択した場合：

- container/Dockerfile
- container/packages.txt

[Use alpine container] を選択した場合：

- container/Dockerfile_alpine
- container/packages_alpine.txt（シェルスクリプトプロジェクトの場合は packages.txt）

以下の説明では、Dockerfile および packages.txt を使用する場合を前提としていますが、Dockerfile_alpine および packages_alpine.txt を使用する場合も同様の手順で進めることができます。

3.17.3.5. Debian コンテナのディストリビューション

Debian コンテナを使用する場合、ディストリビューションは以下のとおりです。

ディストリビューション	・debian:bullseye-slim

3.17.3.6. アプリケーション実行用コンテナイメージの作成

Armadillo 上でアプリケーションを実行するためのコンテナイメージを作成します。ここで作成したコンテナイメージは SWU イメージを使用して Armadillo へインストールするため、事前に 「5.4.1. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。

コンテナイメージの作成および SWU イメージの作成も VS Code で行います。VS Code の左ペインの [my_project] から [Generate development swu] を実行します。

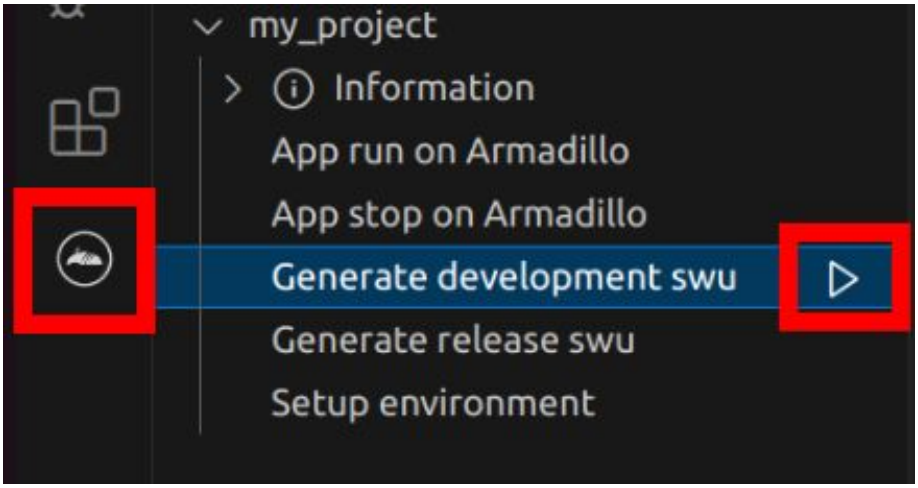


図 3.273 VS Code でコンテナイメージの作成を行う

コンテナイメージの作成にはしばらく時間がかかります。VS Code のターミナルに以下のように表示されるとコンテナイメージの作成は完了です。

```
コンテナイメージを ./swu/my_project.tar に保存しました。
./swu/app.desc のバージョンを 1 から 2 に変更しました。
./development.swu を作成しました。
次は Armadillo に ./development.swu をインストールしてください。
 *  Terminal will be reused by tasks, press any key to close it.
```

図 3.274 コンテナイメージの作成完了

作成した SWU イメージは my_project ディレクトリ下に development.swu というファイル名で保存されています。

3.17.3.7. Python アプリケーションに Bluetooth® Low Energy パッケージをインストールする

Python アプリケーションの場合は、アプリケーションから Bluetooth® Low Energy を使用するために必要なパッケージを VS Code からインストールすることができます。

左ペインの [my_project] から [external packages] を開き [bleak] の右にある+ をクリックするとインストールされます。

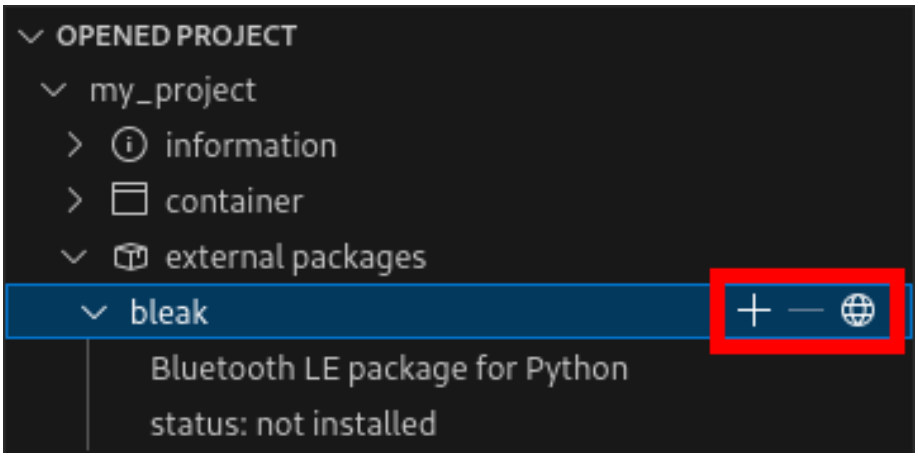


図 3.275 Bluetooth® Low Energy パッケージをインストールする

すでにインストール済みの状態で - をクリックするとアインストールされます。一番右にある丸アイコンをクリックすると Web ブラウザで `bleak` パッケージの API リファレンスページを開きます。



Bluetooth® Low Energy パッケージのインストールは ABOSDE のバージョン 1.8.4 以降で、かつ 2024 年 7 月 24 日以降に「3.17.2.1. プロジェクトの作成」 の手順で新たに作成したプロジェクトで使用できるようになります。

3.17.4. Armadillo に転送するディレクトリ及びファイル

コンテナイメージ以外に、以下に示すディレクトリやファイルを Armadillo に転送します。ここでは、プロジェクト名は `my_project` としています。

Armadillo に転送するディレクトリ及びファイル	・ my_project/app/src

3.17.5. コンテナ内のファイル一覧表示

「図 3.276. コンテナ内のファイル一覧を表示するタブ」 の赤枠で囲われているタブをクリックすることで、`development.swu` または「3.17.8. リリース版のビルド」 で作成される `release.swu` に含まれるコンテナ内のファイルおよびディレクトリを表示します。

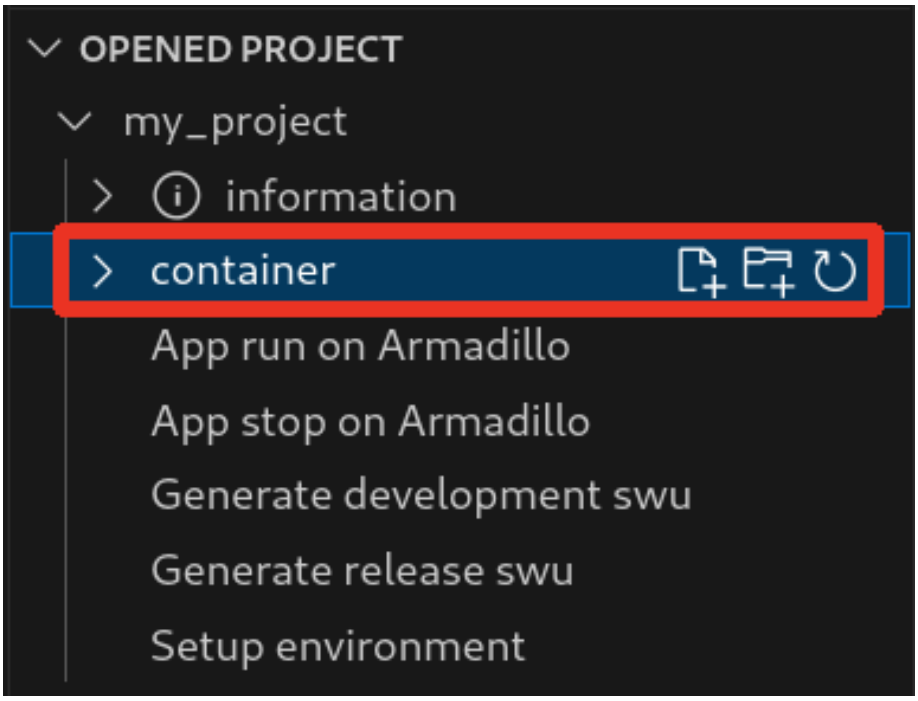


図 3.276 コンテナ内のファイル一覧を表示するタブ

クリック後の表示例を 「図 3.277. コンテナ内のファイル一覧の例」 に示します。

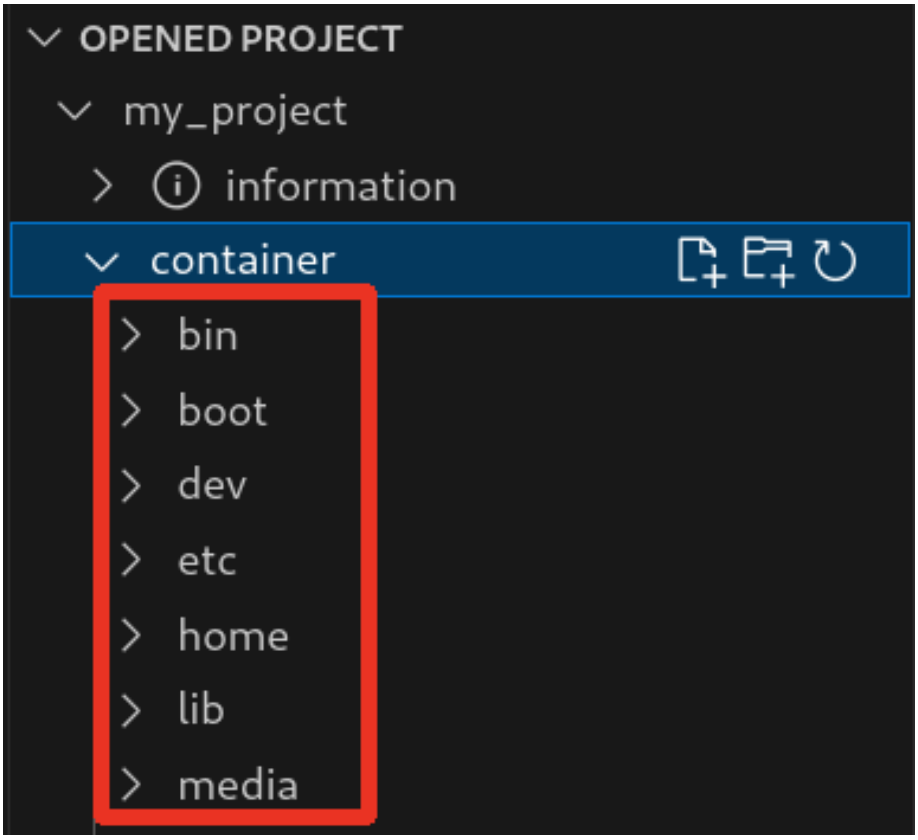


図 3.277 コンテナ内のファイル一覧の例

コンテナ内のファイル一覧は [Generate development swu] または [Generate release swu] を実行することで ATDE 上に作成されるコンテナイメージから取得しています。

そのため、[Generate development swu] または [Generate release swu] を実行していない場合はコンテナ内のファイル一覧は表示されません。その場合は [Generate development swu] または [Generate release swu] を先に実行してください。



この機能を使用するにあたり、ATDE 上でプロジェクトのコンテナイメージからコンテナを作成します。

コンテナ名は「プロジェクト名-abosde」を使用します。例えば、プロジェクト名が my_project の場合、コンテナ名は「my_project-abosde」になります。

ユーザー自身で同名のコンテナを既に作成していた場合、そのコンテナはこの機能を使用時に削除されます。



コンテナ内のファイル一覧には、ファイルおよびディレクトリのみを表示しています。シンボリックリンク、特殊デバイスファイルなどは表示していません。

3.17.5.1. resources ディレクトリについて

「図 3.278. resources ディレクトリ」に示すように ATDE 上のプロジェクトディレクトリには container/resources ディレクトリがあります。

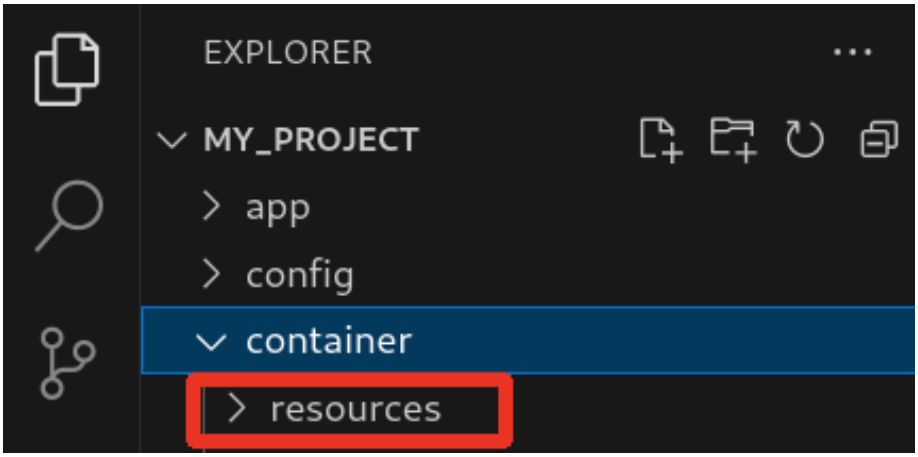


図 3.278 resources ディレクトリ

container/resources ディレクトリ下に、コンテナ内と同じパスでファイルまたはディレクトリを配置することで、それらは [Generate development swu] または [Generate release swu] を実行時にコンテナ内にコピーされます。

例えば、コンテナ内にある /etc/adduser.conf を上書きする場合は、編集した adduser.conf ファイルをプロジェクトディレクトリにある container/resources/etc/adduser.conf に配置してください。

プロジェクトディレクトリにある container/resources 下のファイルおよびディレクトリを操作する方法は以下の 2 通りがあります。

- エクスプローラーを使用する
- ABOSDE のコンテナ内のファイル一覧表示機能を使用する

ABOSDE のコンテナ内のファイル一覧表示機能を使用することで、視覚的にファイル構成や、差分があるファイルを把握しながら操作可能です。以降に詳細を説明します。

3.17.5.2. コンテナ内のファイル一覧の再表示

「図 3.276. コンテナ内のファイル一覧を表示するタブ」 の赤枠で囲われているボタンをクリックすることで、コンテナ内のファイル一覧を再表示します。

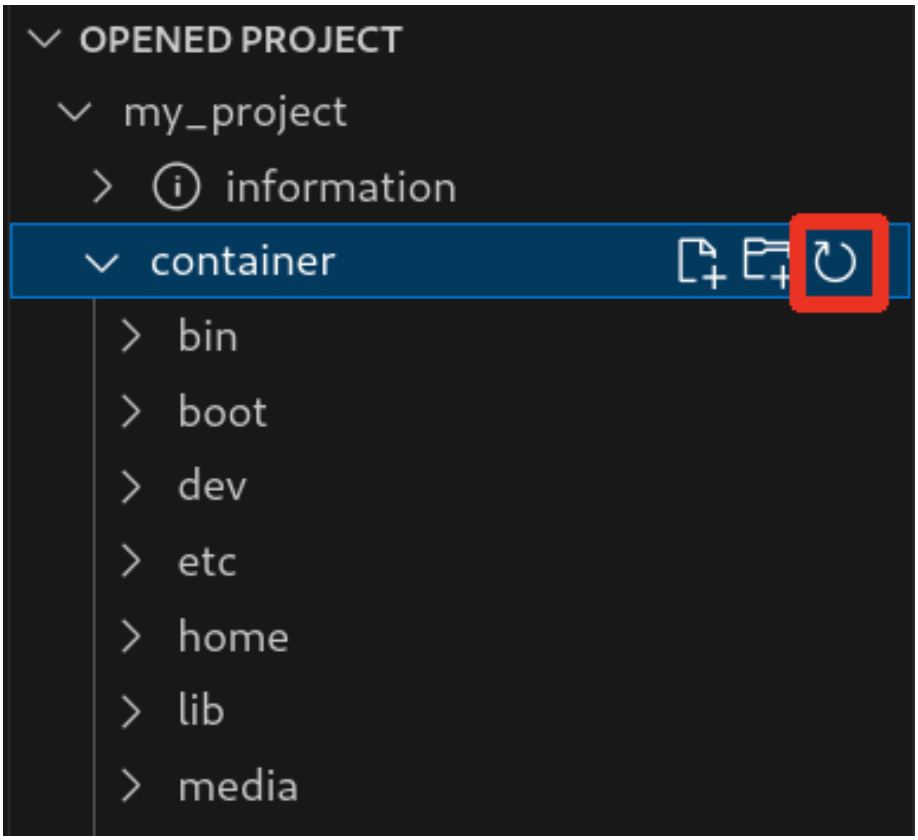


図 3.279 コンテナ内のファイル一覧を再表示するボタン

3.17.5.3. container/resources 下にファイルおよびフォルダーを作成

「図 3.280. container/resources 下にファイルを追加するボタン」 の赤枠で囲われている表記のボタンをクリックすることで、プロジェクトディレクトリにある container/resources 下にファイルを追加することが可能です。

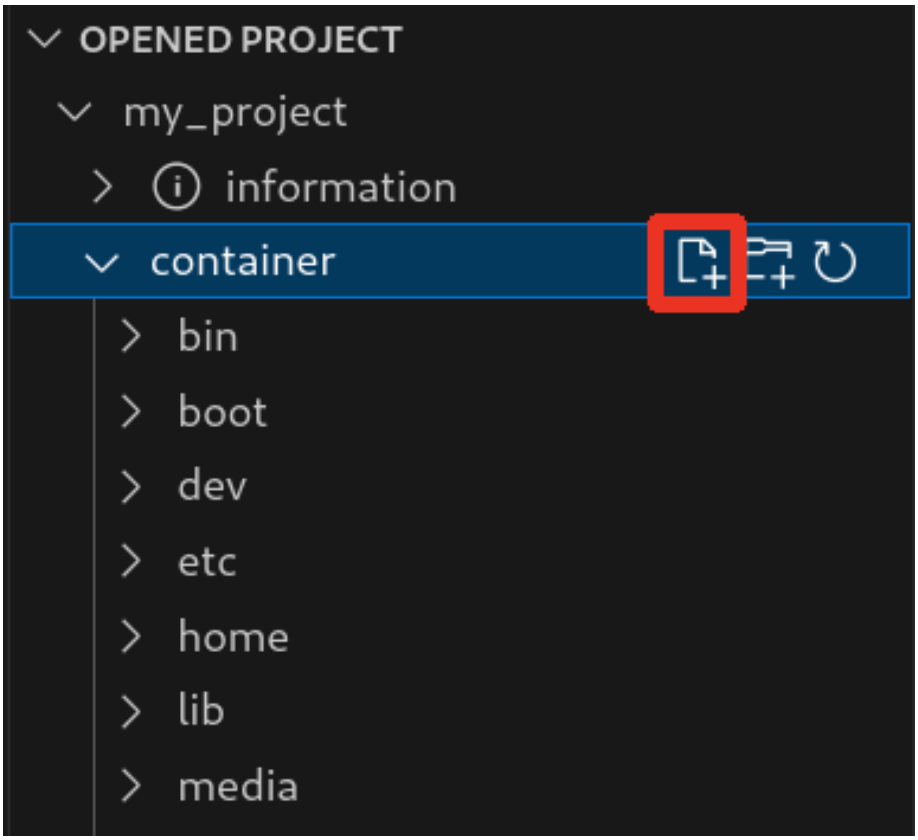


図 3.280 container/resources 下にファイルを追加するボタン

「図 3.281. ファイル名を入力」 に示すように、コマンドパレットが表示されますのでファイル名を入力してください。

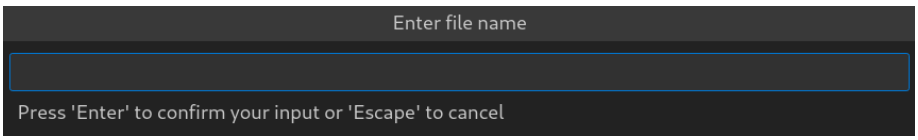


図 3.281 ファイル名を入力

例として、「add_file」というファイル名を入力したとします。

「図 3.282. 追加されたファイルの表示」 に示すように、追加したファイルには「A」というマークが表示されます。

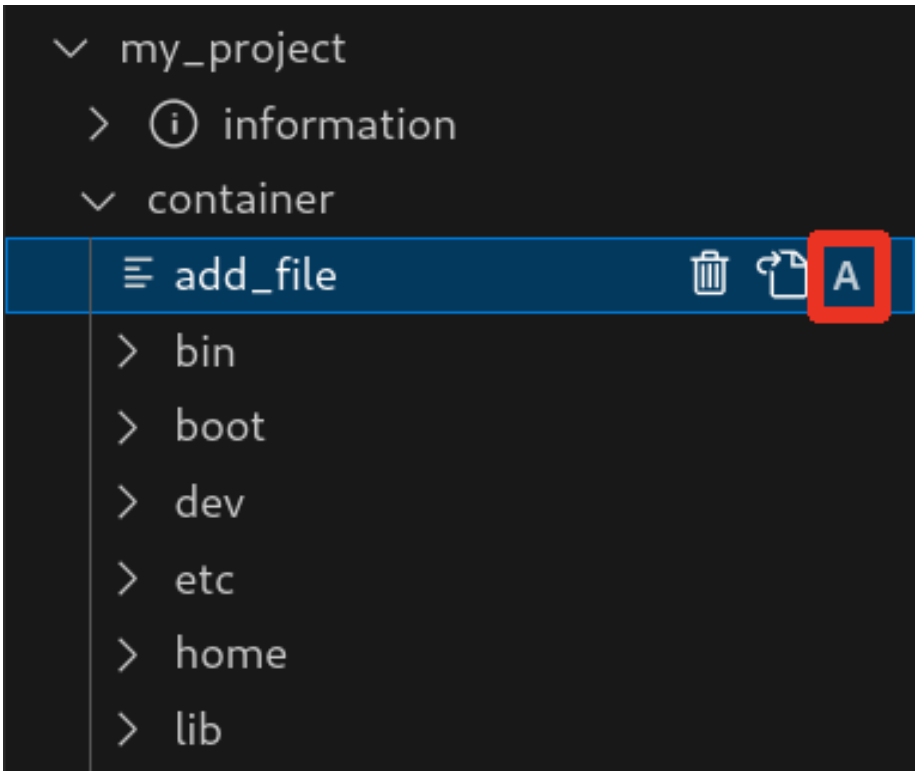


図 3.282 追加されたファイルの表示

また、「図 3.283. container/resources 下にフォルダーを追加するボタン」 の赤枠で囲われている表記のボタンをクリックすることで、ファイルの追加と同様の操作でディレクトリを追加することが可能です。

追加したディレクトリも同様に "A" というマークが表示されます。

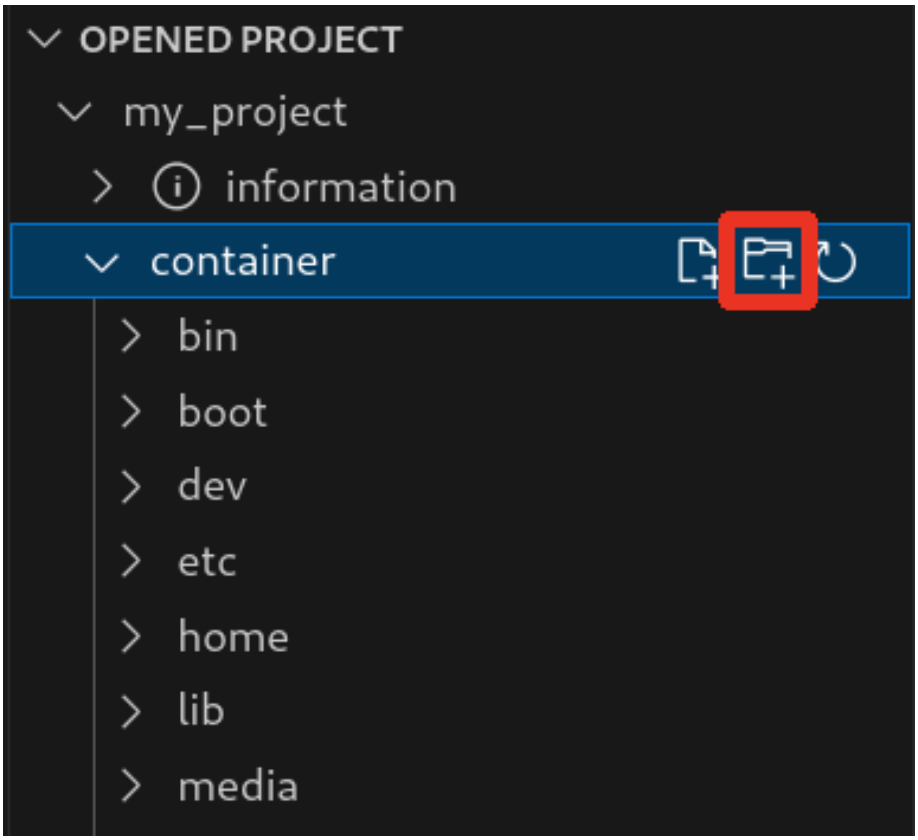


図 3.283 container/resources 下にフォルダーを追加するボタン

3.17.5.4. container/resources 下にあるファイルを開く

「図 3.284. container/resources 下にあるファイルを開くボタン」 の赤枠で囲われている表記のボタンをクリックすることで、プロジェクトディレクトリにある container/resources 下のファイルをエディタに表示することができます。

この例では、プロジェクトディレクトリにある container/resources 下の add_file をエディタに表示します。

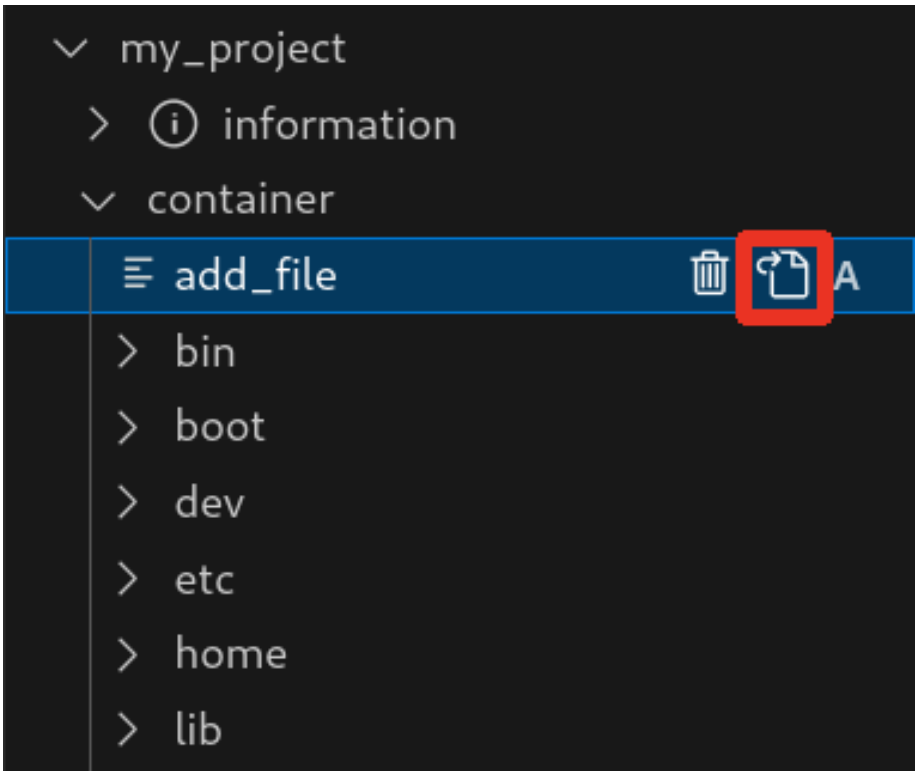


図 3.284 container/resources 下にあるファイルを開くボタン

3.17.5.5. container/resources 下にあるファイルおよびフォルダーの削除

「図 3.284. container/resources 下にあるファイルを開くボタン」 の赤枠で囲われている表記のボタンをクリックすることで、container/resources 下にあるファイルを削除することができます。

この例では、プロジェクトディレクトリにある container/resources 下の add_file を削除します。

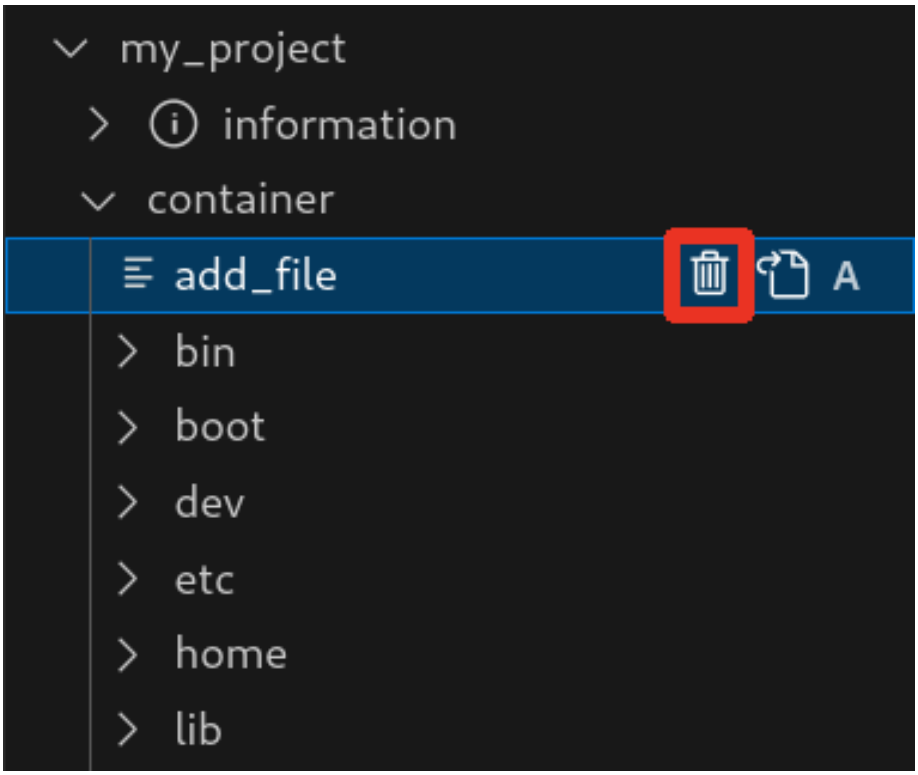


図 3.285 container/resources 下にあるファイルを削除するボタン

ディレクトリも同様に「図 3.284. container/resources 下にあるファイルを開くボタン」 の赤枠で囲われている表記のボタンをクリックすることで削除することができます。

3.17.5.6. コンテナ内のファイルを container/resources 下に保存

「図 3.286. コンテナ内のファイルを container/resources 下に保存するボタン」 の赤枠で囲われている表記のボタンをクリックすることで、コンテナ内にあるファイルをプロジェクトディレクトリにある container/resources 下に保存します。

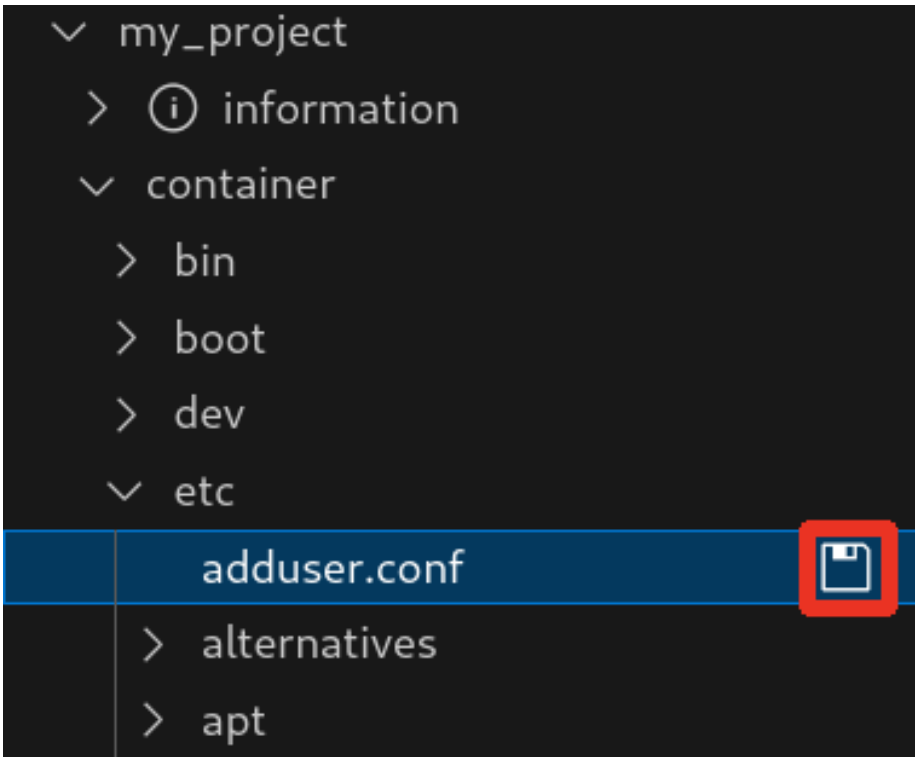


図 3.286 コンテナ内のファイルを container/resources 下に保存するボタン

ファイルが container/resources 下に保存されると、「図 3.287. 編集前のファイルを示すマーク」 に示すように、ファイル名の右側に "U" のマークが表示されます。

"U" のマークはプロジェクトディレクトリにある container/resources 下のファイルとコンテナ内にあるファイルの内容が同一であることを示します。

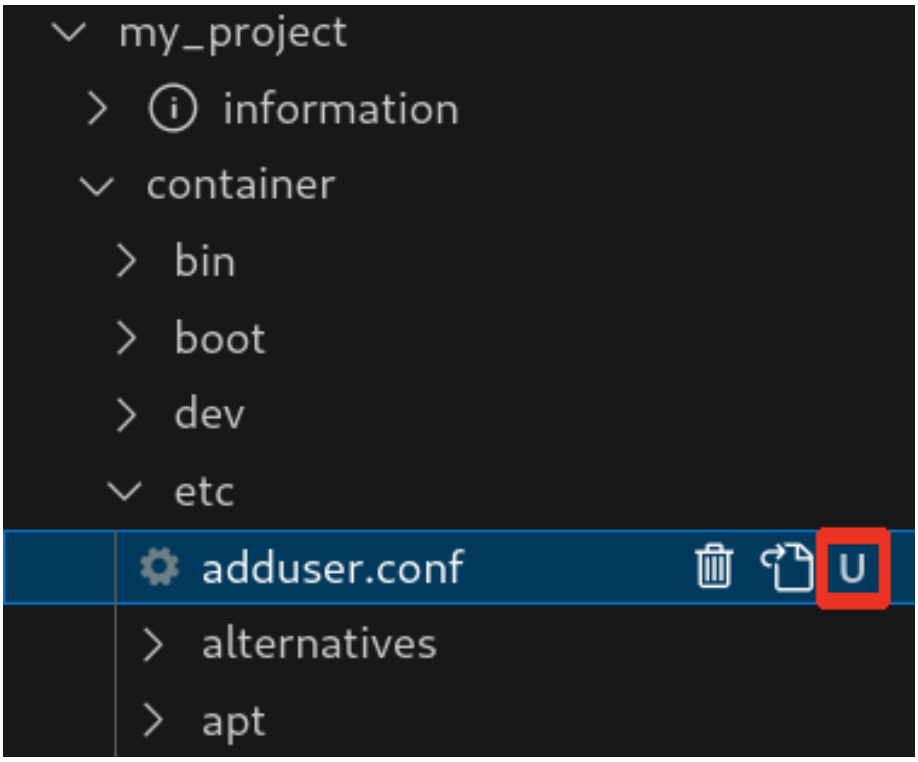


図 3.287 編集前のファイルを示すマーク

container/resources 下にあるファイルを編集して再表示すると、「図 3.288. 編集後のファイルを示すマーク」 に示すように、ファイル名の右側に "M" のマークが表示されます。

"M" のマークはプロジェクトディレクトリにある container/resources 下のファイルとコンテナ内にあるファイルの内容に差があることを示します。

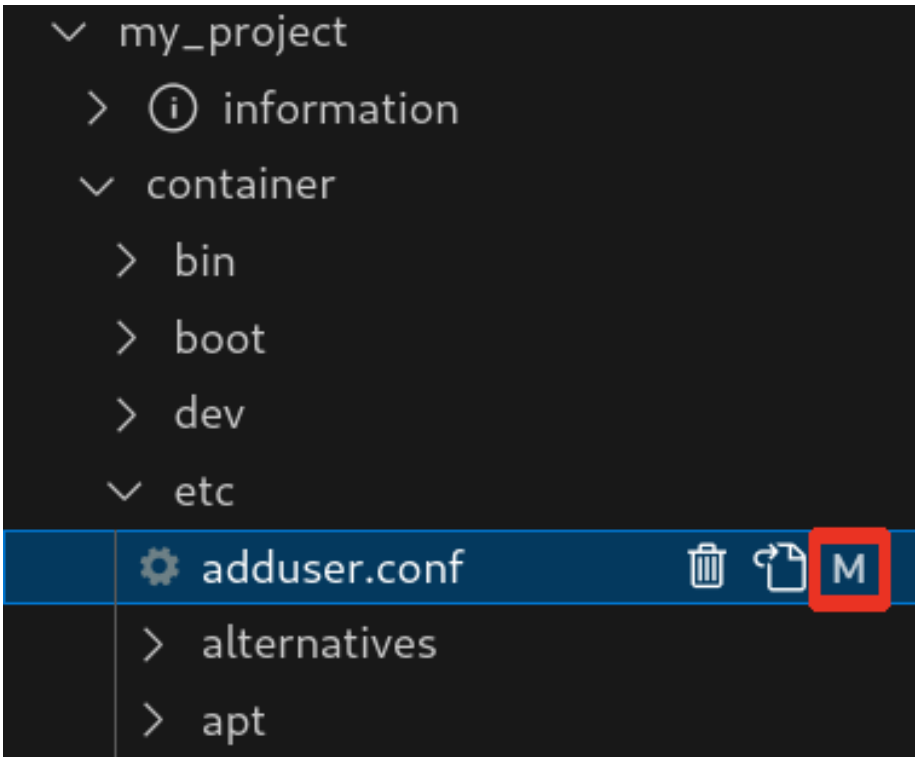


図 3.288 編集後のファイルを示すマーク

3.17.5.7. エラー表示

container/resources 下とコンテナ内にあるファイルまたはディレクトリを比較して、同名でかつファイルの種類が異なる場合、「図 3.289. コンテナ内にコピーされないことを示すマーク」 に示すように、ファイル名の右側に "E" のマークが表示されます。

"E" のマークが表示された場合、そのファイルまたはディレクトリは [Generate development swu] または [Generate release swu] を実行してもコンテナにコピーされません。

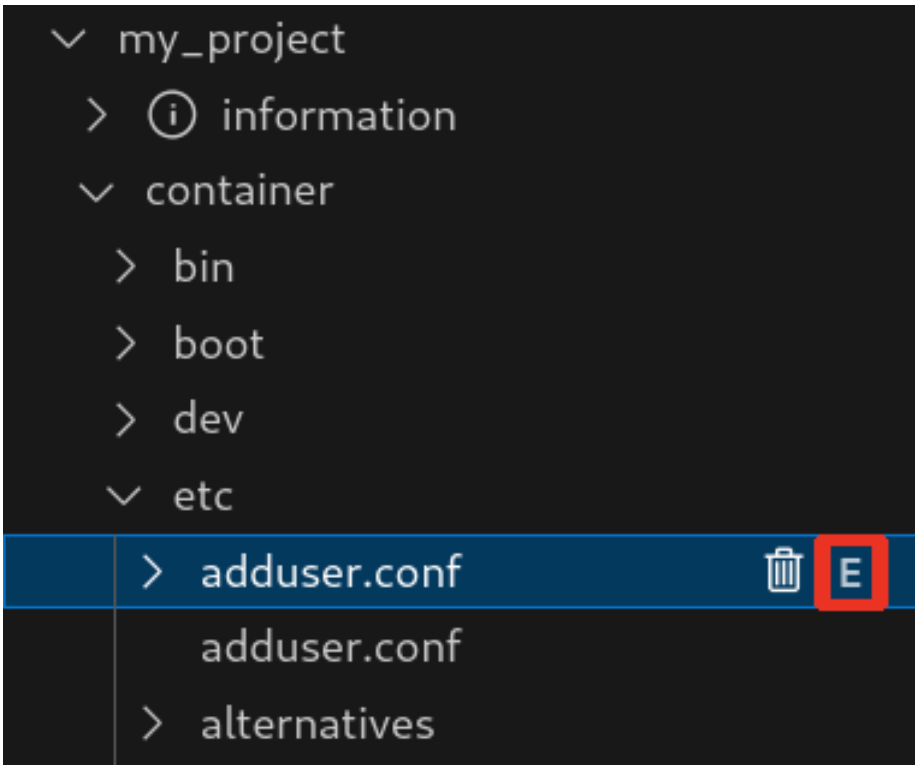


図 3.289 コンテナ内にコピーされないことを示すマーク

3.17.6. Armadillo 上でのセットアップ

3.17.6.1. アプリケーション実行用コンテナイメージのインストール

「3.17.3.6. アプリケーション実行用コンテナイメージの作成」 で作成した `development.swu` を「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールしてください。

インストール後に自動で Armadillo が再起動します。

3.17.6.2. ssh 接続に使用する IP アドレスの設定

VS Code 上で ABOSDE(Armadillo Base OS Development Environment) から、ABOS Web が動作している Armadillo の一覧を確認し、指定した Armadillo の IP アドレスを ssh 接続に使用することができます。ただし、ATDE のネットワークを NAT に設定している場合は Armadillo がリストに表示されません。

「図 3.290. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲われているボタンをクリックすることで、ローカルネットワーク上で ABOS Web が実行されている Armadillo をスキャンすることができます。

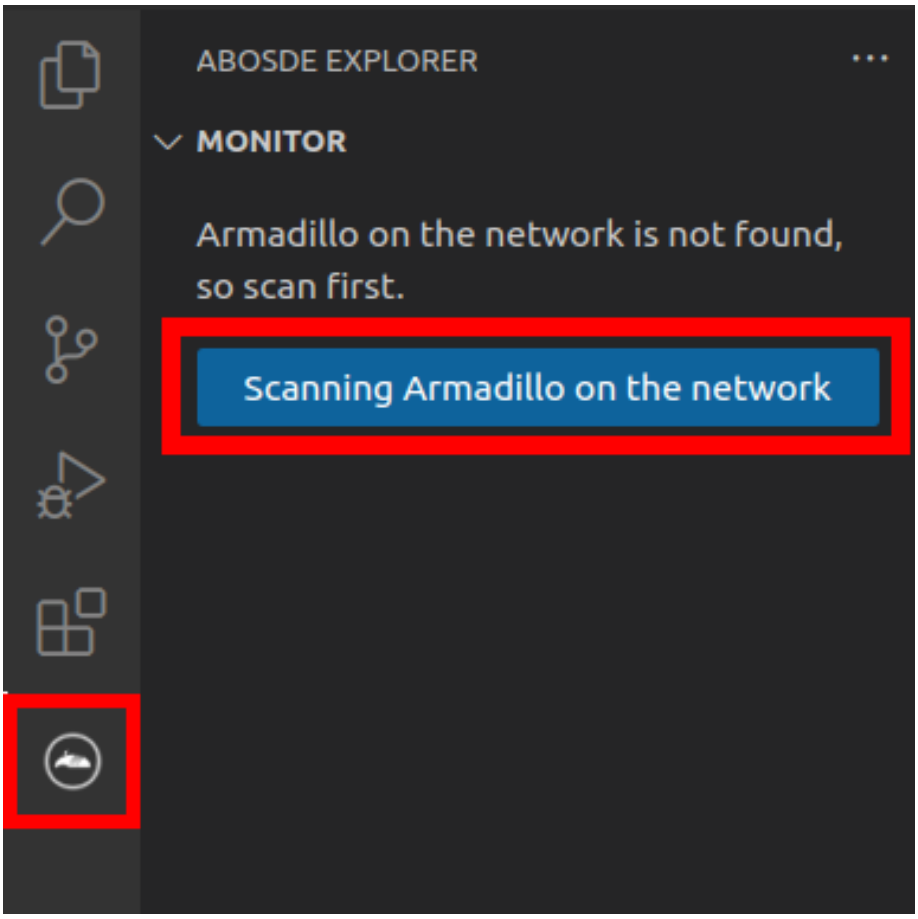


図 3.290 ABOSDE で ローカルネットワーク上の Armadillo をスキャンする

「図 3.291. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する」 の赤枠で囲われているマークをクリックすることで、指定した Armadillo の IP アドレスを ssh 接続に使用する IP アドレスに設定することができます。

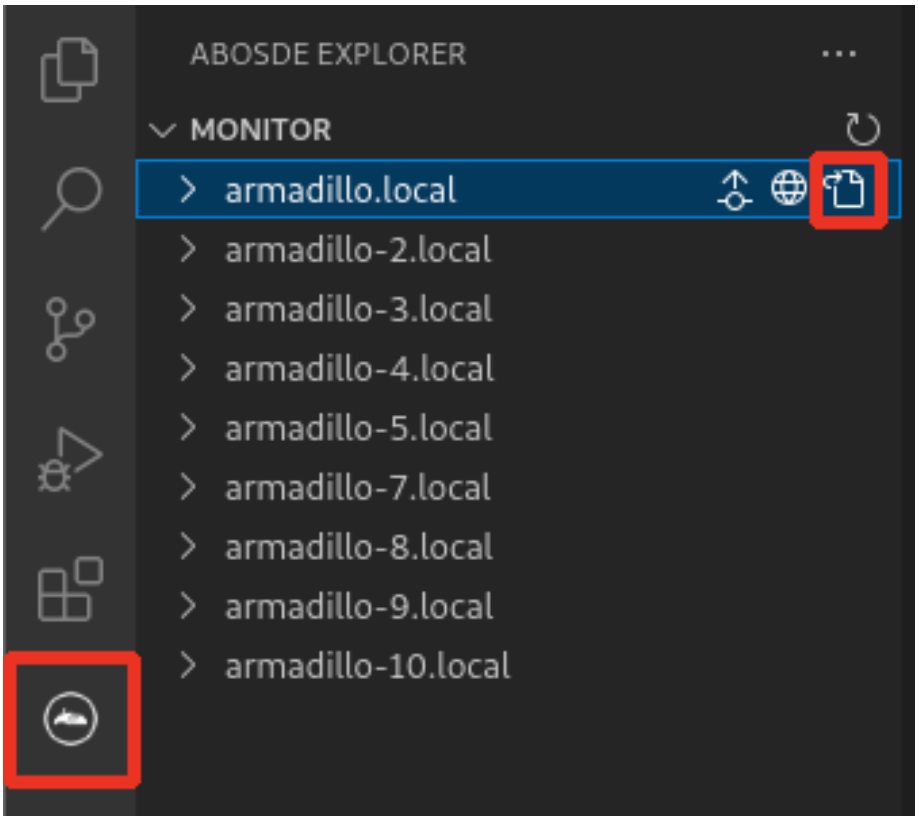


図 3.291 ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する

「図 3.292. ABOSDE に表示されている Armadillo を更新する」 の赤枠で囲われているマークをクリックすることで、ABOSDE に表示されている Armadillo を更新することができます。

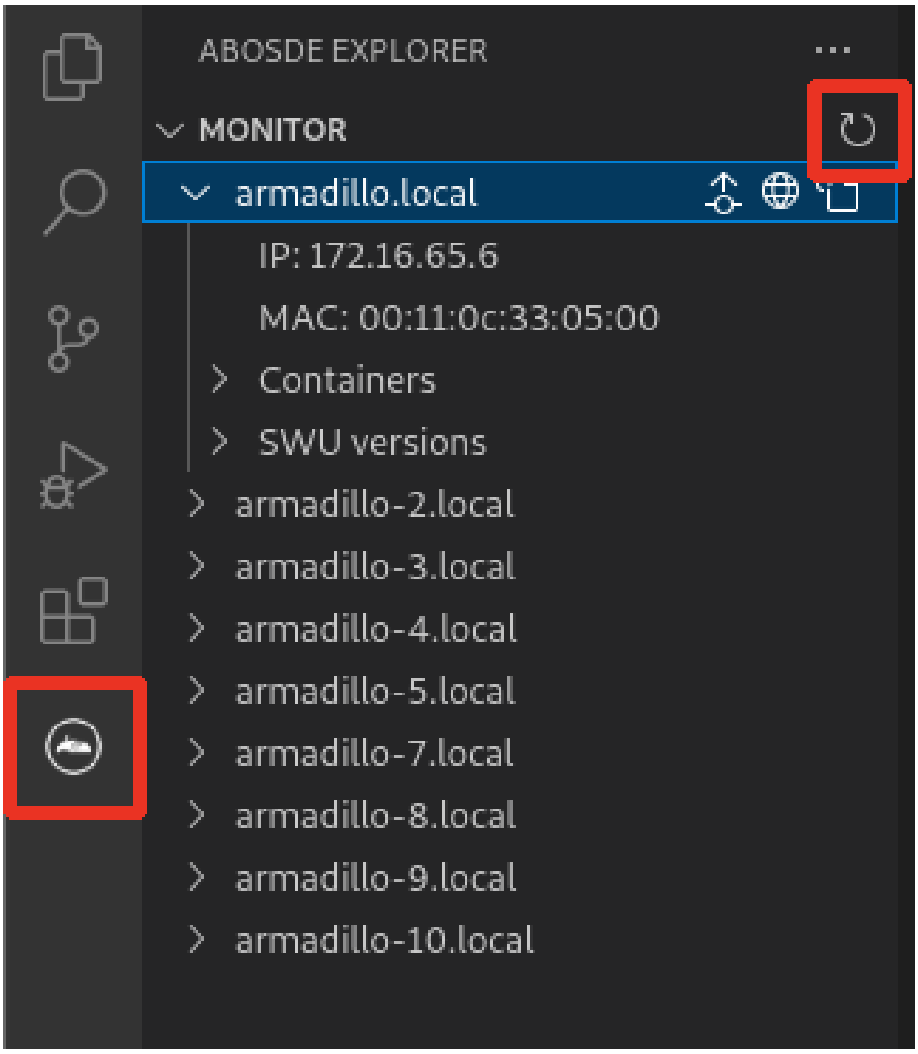


図 3.292 ABOSDE に表示されている Armadillo を更新する

ATDE のネットワークを NAT に設定している場合や、ABOS Web を起動していない場合等、ABOSDE のリストに Armadillo が表示されない場合は、プロジェクトディレクトリに入っている config/ssh_config ファイルを編集して IP アドレスを書き換えてください。

```
[ATDE ~/my_project]$ code config/ssh_config
Host Armadillo
    Hostname x.x.x.x ❶
    User root
    IdentityFile ${HOME}/.ssh/id_ed25519_vscode
    UserKnownHostsFile config/ssh_known_hosts
    StrictHostKeyChecking accept-new
```

図 3.293 ssh_config を編集する

❶ Armadillo の IP アドレスに置き換えてください。



Armadillo を初期化した場合や、プロジェクトを実行する Armadillo を変えた場合は,プロジェクトの config/ssh_known_hosts に保存されている公

開鍵で Armadillo を認識できなくなります。その場合はファイルを削除するか、「Setup environment」タスクを再実行してください。

3.17.6.3. アプリケーションの実行

VS Code の左ペインの [my_project] から [App run on Armadillo] を実行すると、アプリケーションが Armadillo へ転送されて起動します。

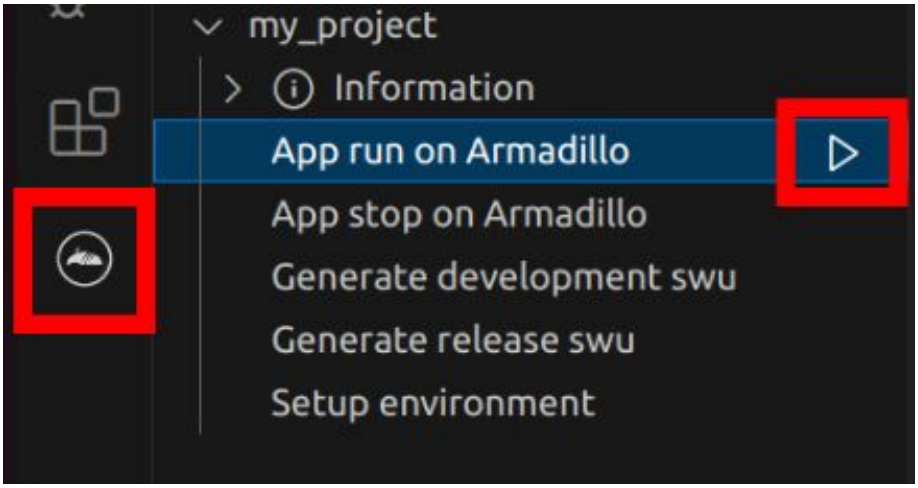


図 3.294 Armadillo 上でアプリケーションを実行する

VS Code のターミナルに以下のメッセージが表示されることがあります。これが表示された場合は yes と入力して下さい。

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

図 3.295 実行時に表示されるメッセージ

アプリケーションを終了するには VS Code の左ペインの [my_project] から [App stop on Armadillo] を実行してください。

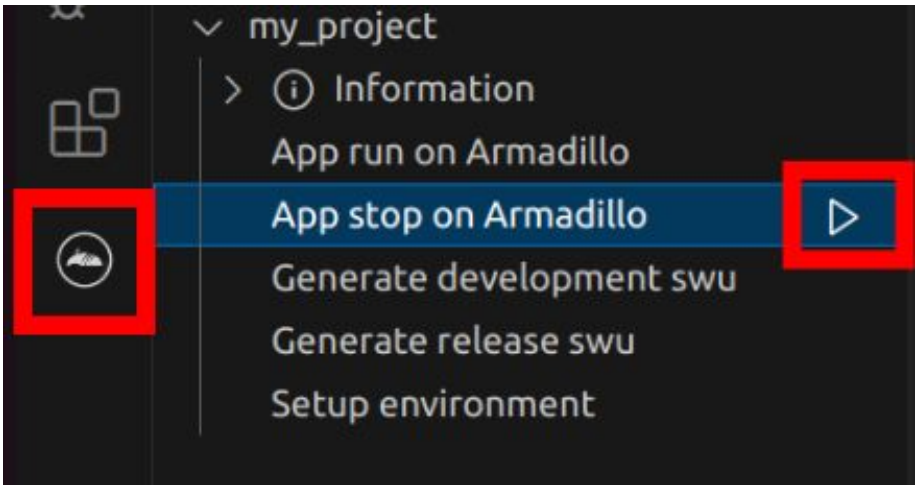


図 3.296 アプリケーションを終了する

3.17.7. SBOM 生成に関する設定

SWU イメージ作成時に、同時に SBOM を生成することができます。詳細は 「3.19. SBOM 生成に関わる設定を行う」 を参照してください。

3.17.8. リリース版のビルド

ここでは完成したアプリケーションをリリース版としてビルドする場合の手順について説明します。

VS Code の左ペインの [my_project] から [Generate release swu] を実行すると、リリース版のアプリケーションを含んだ SWU イメージが作成されます。事前に 「5.4.1. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。

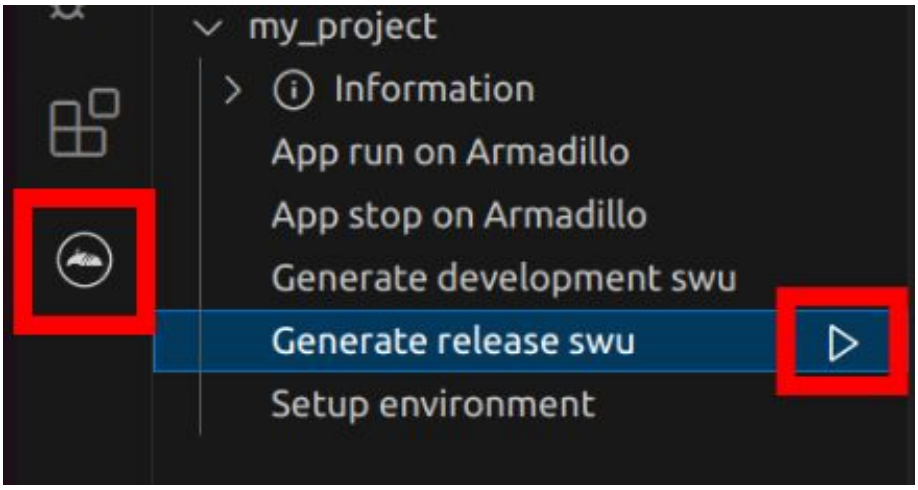


図 3.297 リリース版をビルドする



リリース版の SWU イメージには、開発用の機能は含まれていません。このため、リリース版の SWU イメージをインストールした Armadillo では、[App run on Armadillo] を使用したリモート実行は使用できません。

3.17.9. 製品への書き込み

作成した SWU イメージは `my_project` ディレクトリ下に `release.swu` というファイル名で保存されています。

この SWU イメージを 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールすると、Armadillo 起動時にアプリケーションも自動起動します。

3.17.10. Armadillo 上のコンテナイメージの削除

「6.9.3. コンテナとコンテナに関連するデータを削除する」 を参照してください。

3.18. C 言語によるアプリケーションの開発

ここでは C 言語によるアプリケーション開発の方法を紹介します。

C 言語によるアプリケーション開発は下記に当てはまるユーザーを対象としています。

・既存の C 言語によって開発されたアプリケーションを Armadillo で動作させたい

・C 言語でないと実現できないアプリケーションを開発したい

上記に当てはまらず、開発するアプリケーションがシェルスクリプトまたは Python で実現可能であるならば、「3.17. CUI アプリケーションの開発」を参照してください。

3.18.1. C 言語によるアプリケーション開発の流れ

Armadillo 向けに C 言語によるアプリケーションを開発する場合の流れは以下のようになります。

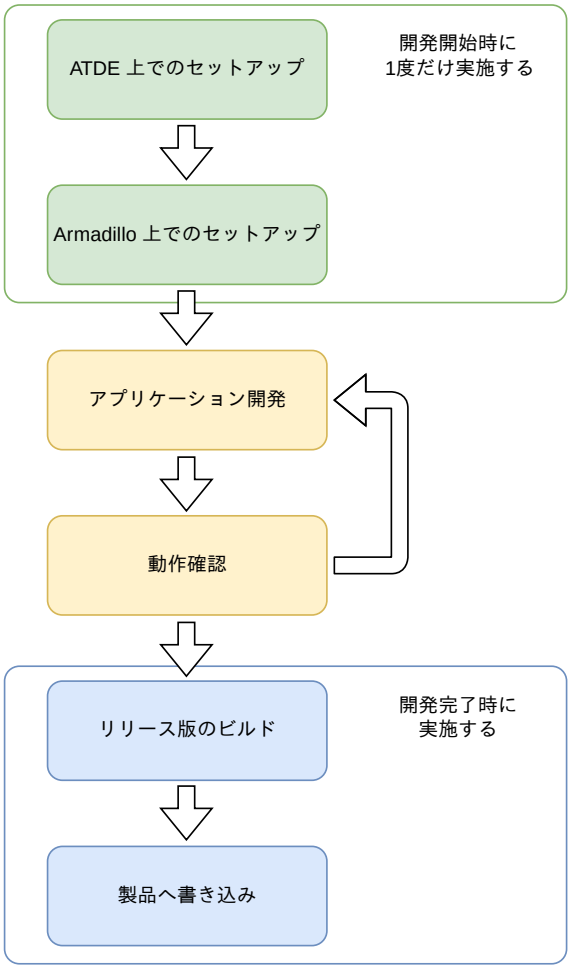


図 3.298 C 言語によるアプリケーション開発の流れ

3.18.2. ATDE 上でのセットアップ

ここでは、開発開始時の ATDE 上でのセットアップ手順について説明します。ATDE をお使いでない場合は、先に 「3.1. 開発の準備」 を参照して ATDE 及び、 VS Code のセットアップを完了してください。

3.18.2.1. プロジェクトの作成

VS Code の左ペインの [A6E] から [C New Project] を実行し、表示されるディレクトリ選択画面からプロジェクトを保存するディレクトリを選択してください。実行するためには右に表示されている三角形ボタンを押してください。保存先を選択すると、プロジェクト名を入力するダイアログが表示され

るので、任意のプロジェクト名を入力してエンターキーを押してください。この操作により、選択した保存先に、入力したプロジェクト名と同名のディレクトリが作成されます。

また、ここでは次のように設定しています。

- **保存先**：ホームディレクトリ
- **プロジェクト名**：`my_project`

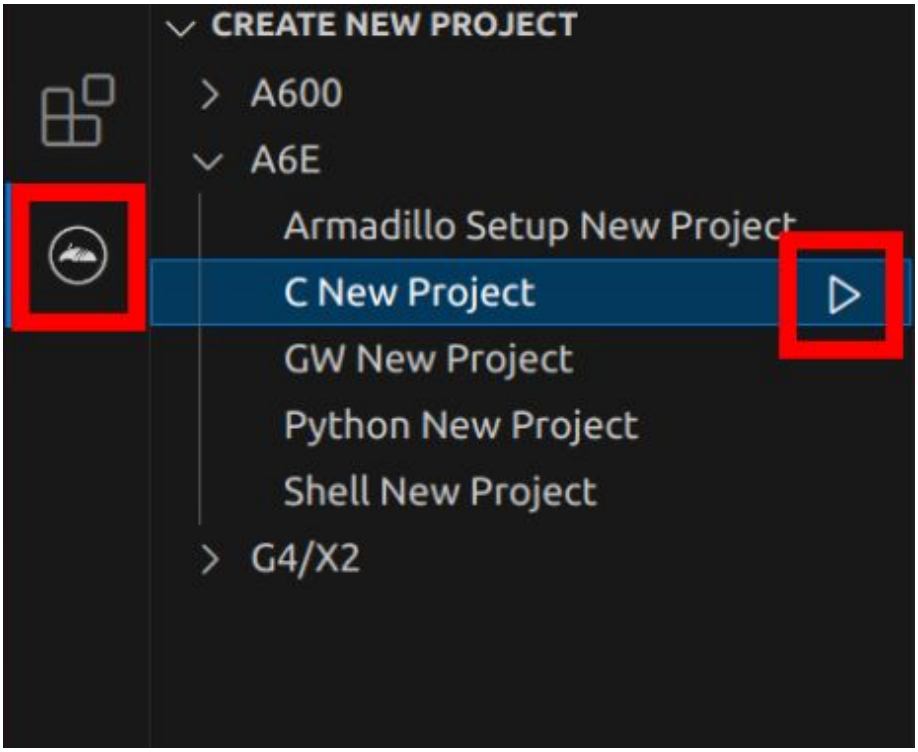


図 3.299 プロジェクトを作成する

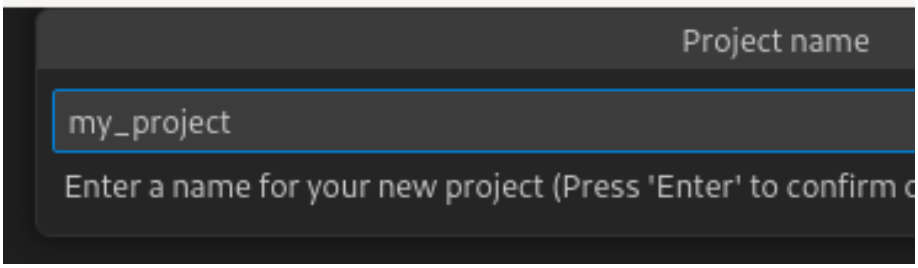


図 3.300 プロジェクト名を入力する

3.18.3. アプリケーション開発

3.18.3.1. VS Code の起動

ここでは、実際に Armadillo 上でサンプルアプリケーションを起動する場合の手順を説明します。プロジェクトディレクトリへ移動し VS Code を起動します。

```
[ATDE ~]$ code ./my_project
```

図 3.301 VS Code で my_project を起動する

3.18.3.2. ディレクトリ構成

プロジェクトには下記のディレクトリがあります。

- **app** : 各ディレクトリの説明は以下の通りです。
  - **src** : アプリケーションのソースファイル（拡張子が .c ）と Makefile を配置してください。
  - **build** : ここに配置した実行ファイルが Armadillo 上で実行されます。
  - **lib** : 共有ライブラリの検索パスとしてこのディレクトリを指定しているので、ここに共有ライブラリ（拡張子が .so ）を配置することができます。
- **config** : 設定に関わるファイルが含まれるディレクトリです。
  - **app.conf** : コンテナのコンフィグです。記載内容については 「6.9.4. コンテナ起動設定ファイルを作成する」 を参照してください。
  - **app.desc** : SWU イメージを生成するための .desc ファイルです。記載内容については 「6.4. mkswu の .desc ファイルを編集する」 を参照してください。
  - **ssh_config** : Armadillo への ssh 接続に使用します。「3.18.6.2. ssh 接続に使用する IP アドレスの設定」 を参照してください。
- **container** : スクリプトを実行するコンテナの設定ファイルが含まれるディレクトリです。
  - **packages.txt**: このファイルに記載されているパッケージがインストールされます。
  - **Dockerfile**: 直接編集することも可能です。
  - **resources**: Armadillo の ルートディレクトリ（ / ）配下に同様のディレクトリ構造でコピーされます。
  - **resources_c**: resources と同様ですが、C 言語プロジェクト固有のファイルが配置されています。
    - **build.sh**: 実行ファイルをビルドするためのスクリプトです。必要に応じて内容を変更してください。
    - **bin/c_launch**: コンテナ起動時に実行するスクリプトです。

デフォルトのコンテナコンフィグ（ app.conf ）では C 言語の場合は build/main を実行しますので、リネームが必要な場合にコンテナのコンフィグも修正してください。

このサンプルアプリケーションは、CPU と SOC の温度を /vol_data/log/temp.txt に出力し、アプリケーション LED を点滅させます。

3.18.3.3. 初期設定

初期設定では主に Armadillo と SSH で接続するための秘密鍵と公開鍵の生成を行います。

作成したプロジェクトディレクトリへ移動して VS Code を起動してください。

```
[ATDE ~]$ cd my_project
[ATDE ~/my_project]$ code ./
```

図 3.302 初期設定を行う

VS Code の左ペインの [my_project] から [Setup environment] を実行します。

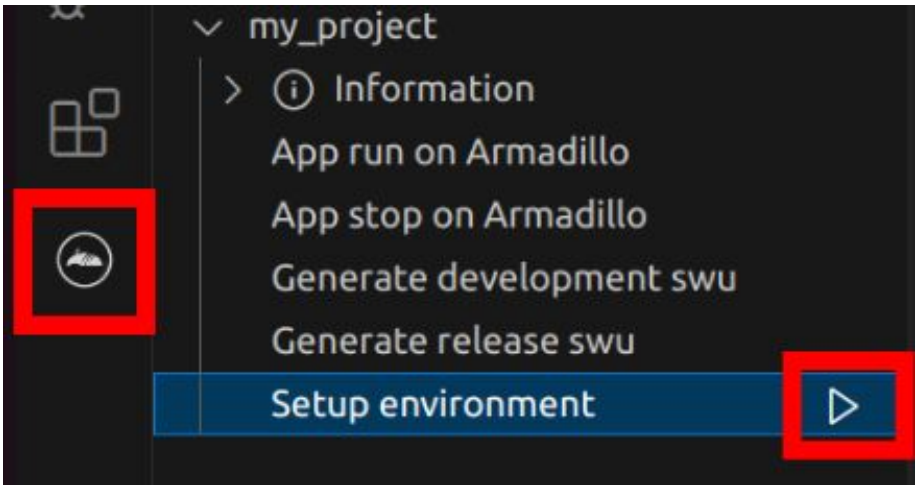


図 3.303 VS Code で初期設定を行う

選択すると、 VS Code の下部に以下のようなターミナルが表示されます。

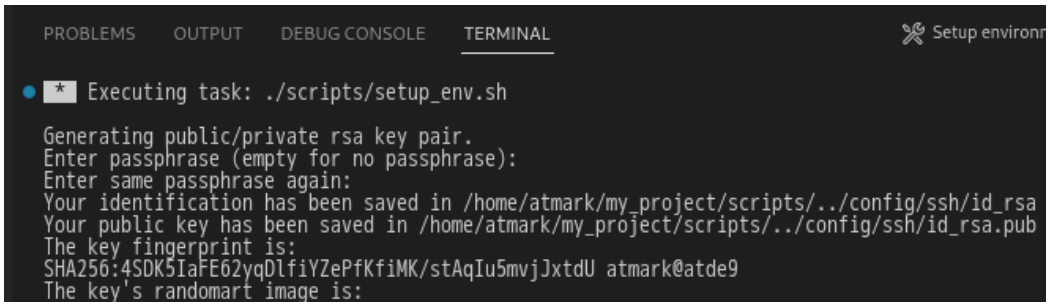


図 3.304 VS Code のターミナル

このターミナル上で以下のように入力してください。

```
 *  Executing task: ./scripts/setup_env.sh

Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase): ❶
Enter same passphrase again: ❷
Your identification has been saved in /home/atmark/.ssh/id_ed25519_vscode
:(省略)

 *  Terminal will be reused by tasks, press any key to close it.  ❸
```

図 3.305 SSH 用の鍵を生成する

❶ パスフレーズを設定します。設定しない場合は何も入力せず Enter を押します。

❷ 1 でパスフレーズを設定した場合は、確認のため再度入力してください。

❸ ここで何か任意のキーを押すとターミナルが閉じます。

パスフレーズを設定した場合は、アプリケーションを Armadillo へ転送する時にパスフレーズの入力を求められることがあります。



ssh の鍵は $HOME/.ssh/id_ed25519_vscode (と id_ed25519_vscode.pub) に保存されていますので、プロジェクトをバックアップする時は $HOME/.ssh も保存してください。

3.18.3.4. ABOSDE におけるコンテナ OS の選択方法



コンテナ OS を Debian から Alpine に変更する ABOSDE の機能は ABOSDE のバージョン 1.9.3 以降で、かつ 2025 年 6 月 25 日以降に「3.18.2.1. プロジェクトの作成」 の手順で新たに作成したプロジェクトで使用できるようになります。

ABOSDE では、コンテナ OS として Debian と Alpine を提供しています。デフォルトでは、Alpine よりも開発が容易である Debian が使用されます。

Alpine を使用する場合は、VS Code の左ペインの [OPENED PROJECT] の [Generate development swu] または [Generate release swu] を開き、[Use alpine container] を選択してください。

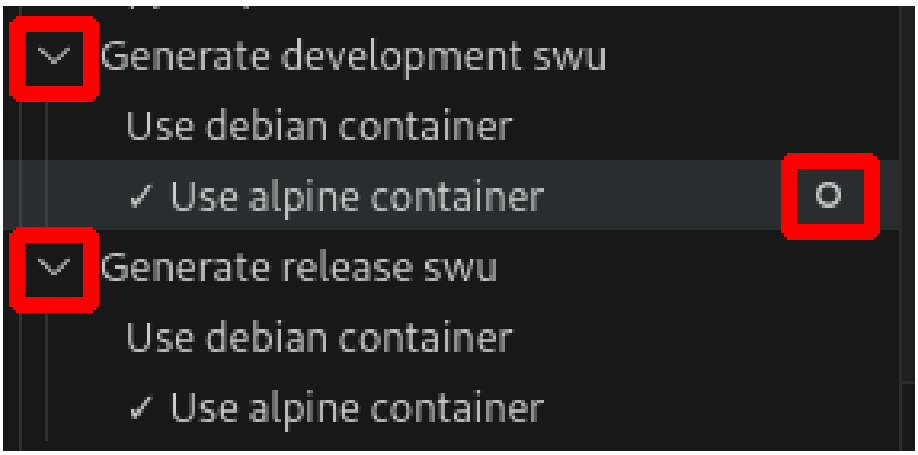


図 3.306 ABOSDE でコンテナ OS を選択する

SWU イメージを生成する時に以下のファイルが使用されます。使用するコンテナ OS にあわせて、対応するファイルを編集してください。

[Use debian container] を選択した場合：

- container/Dockerfile

・ container/packages.txt

[Use alpine container] を選択した場合：

・ container/Dockerfile_alpine

・ container/packages_alpine.txt

以下の説明では、Dockerfile および packages.txt を使用する場合を前提としていますが、Dockerfile_alpine および packages_alpine.txt を使用する場合も同様の手順で進めることができます。

3.18.3.5. Debian コンテナのディストリビューション

Debian コンテナを使用する場合、ディストリビューションは以下のとおりです。

ディストリビューション	・ debian:bullseye-slim

3.18.3.6. packages.txt の書き方

ABOSDE ではコンテナイメージにパッケージをインストールするために container ディレクトリにある packages.txt を使用します。packages.txt に記載されているパッケージは "apt install" コマンドによってコンテナイメージにインストールされます。

C 言語による開発の場合、packages.txt に [build] というラベルを記載することで、ビルド時のみに使用するパッケージを指定することが出来ます。

「図 3.307. C 言語による開発における packages.txt の書き方」に C 言語による開発の場合における packages.txt の書き方の例を示します。ここでは、パッケージ名を package_A 、package_B 、package_C としています。

```
package_A
package_B

[build]   ❶
package_C
```

図 3.307 C 言語による開発における packages.txt の書き方

❶ このラベル以降のパッケージはビルド時のみに使用されます。

上記の例の場合、Armadillo 上で実行される環境では package_A 、package_B のみがインストールされ、package_C はインストールされません。

"[build] package_C" のように [build] の後に改行せずに、一行でパッケージ名を書くことは出来ませんのでご注意ください。

3.18.3.7. 実行ファイルのビルド

以下のいずれかのボタンを実行すると、container/resources_c 直下にある build.sh をビルド用コンテナ内で実行して、実行ファイルが app/build に生成されます。

・ [Generate development swu]

・ [Generate release swu]

・ [App run on Armadillo]

なお [App run on Armadillo] は [Generate development swu] または [Generate release swu] を先に実行して、Armadillo に SWU イメージをインストールしておく必要があります。

build.sh のデフォルトの中身は以下のようになっています。

container/resources_c/build.sh の中身.

```
  #!/bin/sh

make -C /vol_app/src all
```

デフォルトでは、ATDE 上のプロジェクトディレクトリ内の app/src はコンテナ内の /vol_app/src にコピーされます。

Makefile は app/src 直下に配置しているため、C オプションでパスを指定した後、make all を実行しています。

build.sh の内容を変更することで任意のビルドコマンドを使用することが可能です。

3.18.3.8. ABOSDE での開発における制約

app/build と app/lib 内のファイルが Armadillo に転送されますので、実行ファイルは app/build 、共有ライブラリ（ 拡張子が .so ファイル ）は app/lib に配置してください。

3.18.3.9. アプリケーション実行用コンテナイメージの作成

Armadillo 上でアプリケーションを実行するためのコンテナイメージを作成します。ここで作成したコンテナイメージは SWU イメージを使用して Armadillo へインストールするため、事前に 「5.4.1. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。

コンテナイメージの作成、 実行ファイルや共有ライブラリの作成および SWU イメージの作成も VS Code で行います。VS Code の左ペインの [my_project] から [Generate development swu] を実行します。

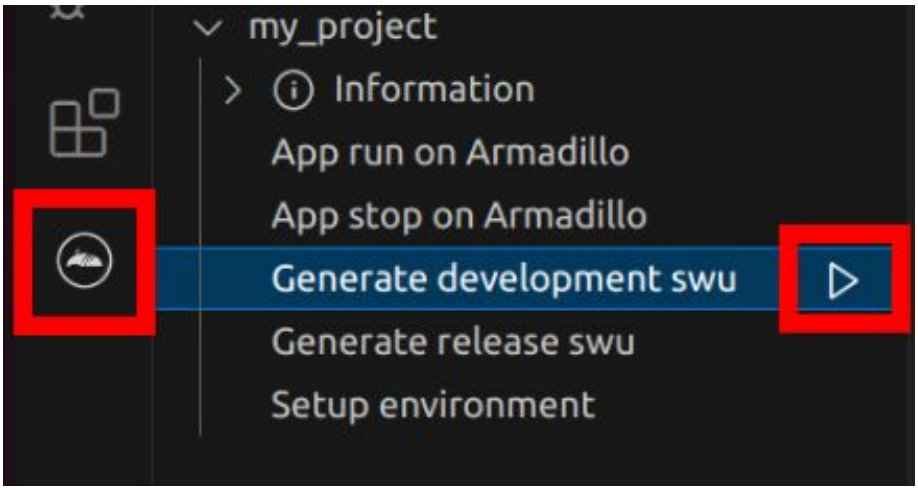


図 3.308 VS Code でコンテナイメージの作成を行う

コンテナイメージの作成にはしばらく時間がかかります。VS Code のターミナルに以下のように表示されるとコンテナイメージの作成は完了です。

```
コンテナイメージを ./swu/my_project.tar に保存しました。
./swu/app.desc のバージョンを 1 から 2 に変更しました。
./development.swu を作成しました。
次は Armadillo に ./development.swu をインストールしてください。
 *  Terminal will be reused by tasks, press any key to close it.
```

図 3.309 コンテナイメージの作成完了

作成した SWU イメージは my_project ディレクトリ下に development.swu というファイル名で保存されています。

3.18.4. コンテナ内のファイル一覧表示

「図 3.310. コンテナ内のファイル一覧を表示するタブ」 の赤枠で囲われているタブをクリックすることで、development.swu または「3.18.8. リリース版のビルド」で作成される release.swu に含まれるコンテナ内のファイルおよびディレクトリを表示します。

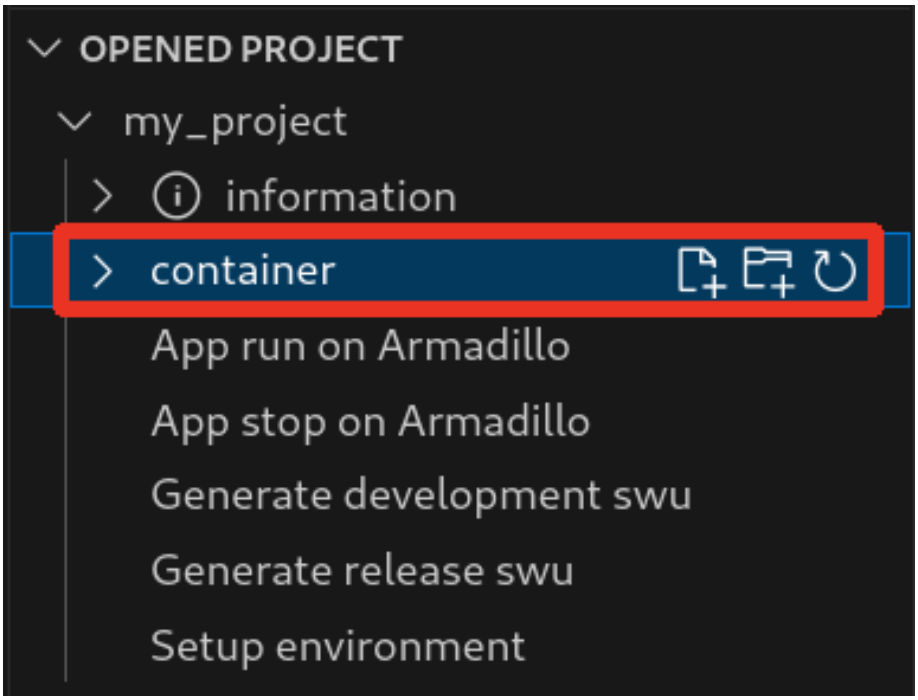


図 3.310 コンテナ内のファイル一覧を表示するタブ

クリック後の表示例を 「図 3.311. コンテナ内のファイル一覧の例」 に示します。

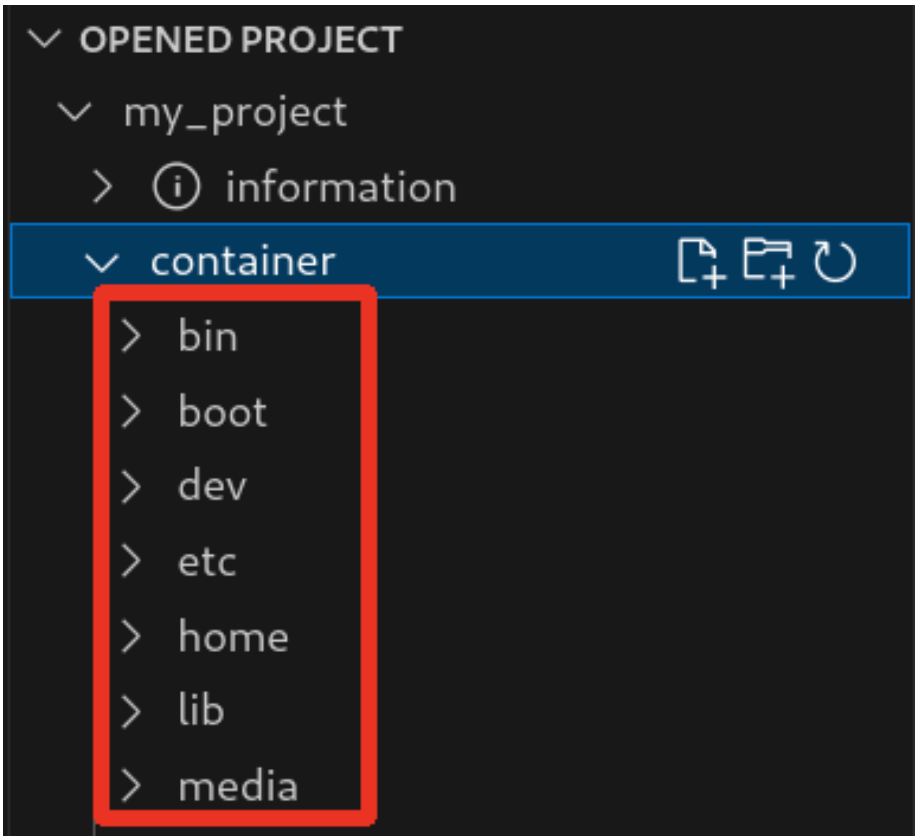


図 3.311 コンテナ内のファイル一覧の例

コンテナ内のファイル一覧は [Generate development swu] または [Generate release swu] を実行することで ATDE 上に作成されるコンテナイメージから取得しています。

そのため、[Generate development swu] または [Generate release swu] を実行していない場合はコンテナ内のファイル一覧は表示されません。その場合は [Generate development swu] または [Generate release swu] を先に実行してください。

この機能を使用するにあたり、ATDE 上でプロジェクトのコンテナイメージからコンテナを作成します。

コンテナ名は「プロジェクト名-abosde」を使用します。例えば、プロジェクト名が my_project の場合、コンテナ名は「my_project-abosde」になります。

ユーザー自身で同名のコンテナを既に作成していた場合、そのコンテナはこの機能を使用時に削除されます。

コンテナ内のファイル一覧には、ファイルおよびディレクトリのみを表示しています。シンボリックリンク、特殊デバイスファイルなどは表示していません。

3.18.4.1. resources ディレクトリについて

「図 3.312. resources ディレクトリ」に示すように ATDE 上のプロジェクトディレクトリには container/resources ディレクトリがあります。

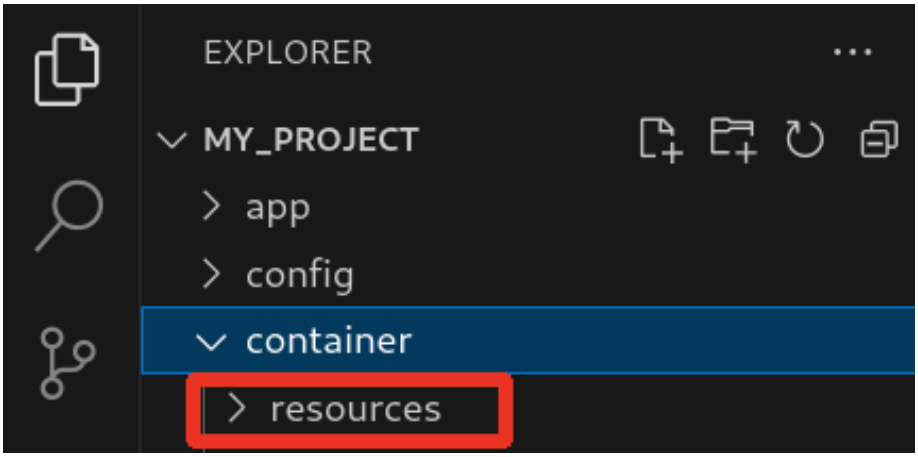


図 3.312 resources ディレクトリ

container/resources ディレクトリ下に、コンテナ内と同じパスでファイルまたはディレクトリを配置することで、それらは [Generate development swu] または [Generate release swu] を実行時にコンテナ内にコピーされます。

例えば、コンテナ内にある /etc/adduser.conf を上書きする場合は、編集した adduser.conf ファイルをプロジェクトディレクトリにある container/resources/etc/adduser.conf に配置してください。

プロジェクトディレクトリにある container/resources 下のファイルおよびディレクトリを操作する方法は以下の 2 通りがあります。

- エクスプローラーを使用する
- ABOSDE のコンテナ内のファイル一覧表示機能を使用する

ABOSDE のコンテナ内のファイル一覧表示機能を使用することで、視覚的にファイル構成や、差分があるファイルを把握しながら操作可能です。以降に詳細を説明します。

3.18.4.2. コンテナ内のファイル一覧の再表示

「図 3.310. コンテナ内のファイル一覧を表示するタブ」 の赤枠で囲われているボタンをクリックすることで、コンテナ内のファイル一覧を再表示します。

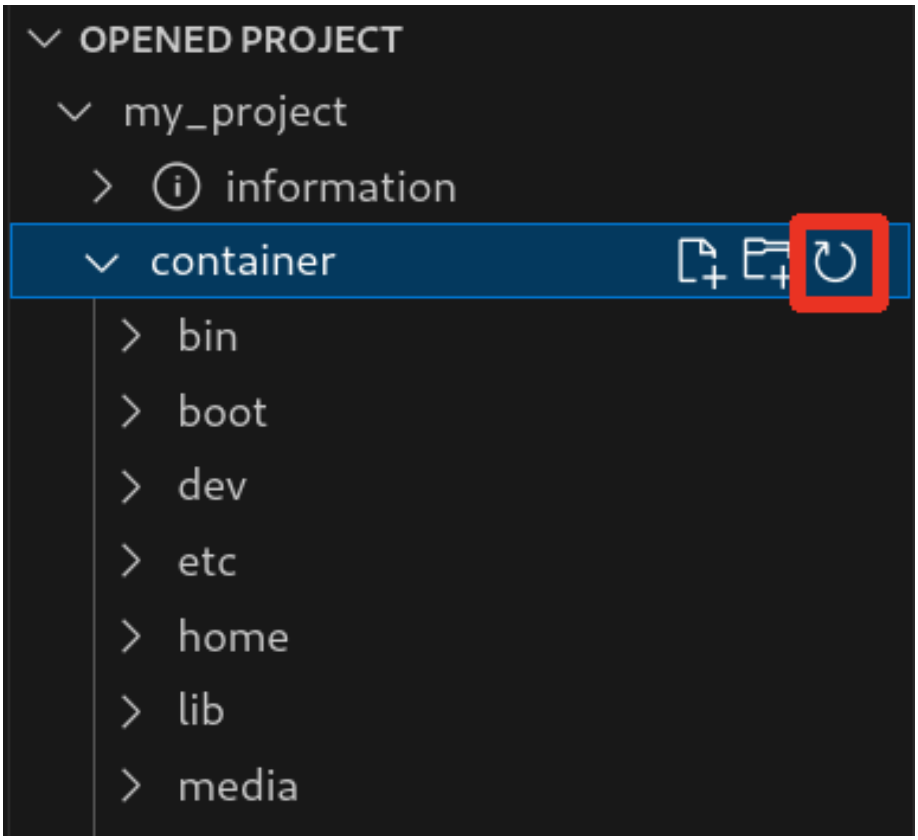


図 3.313 コンテナ内のファイル一覧を再表示するボタン

3.18.4.3. container/resources 下にファイルおよびフォルダーを作成

「図 3.314. container/resources 下にファイルを追加するボタン」 の赤枠で囲われている表記のボタンをクリックすることで、プロジェクトディレクトリにある container/resources 下にファイルを追加することが可能です。

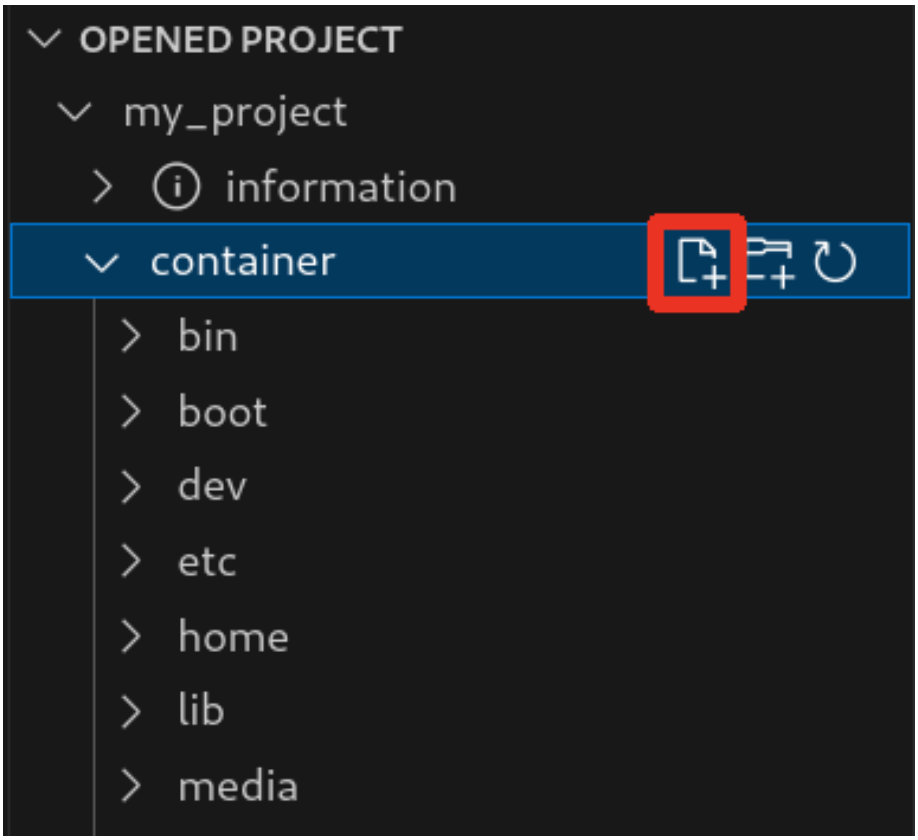


図 3.314 container/resources 下にファイルを追加するボタン

「図 3.315. ファイル名を入力」 に示すように、コマンドパレットが表示されますのでファイル名を入力してください。

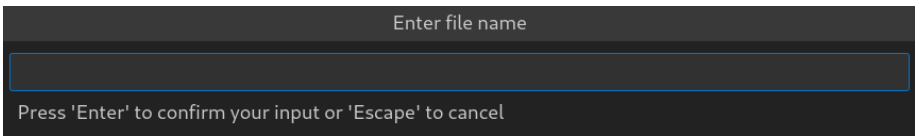


図 3.315 ファイル名を入力

例として、「add_file」というファイル名を入力したとします。「図 3.316. 追加されたファイルの表示」に示すように、追加したファイルには「A」というマークが表示されます。

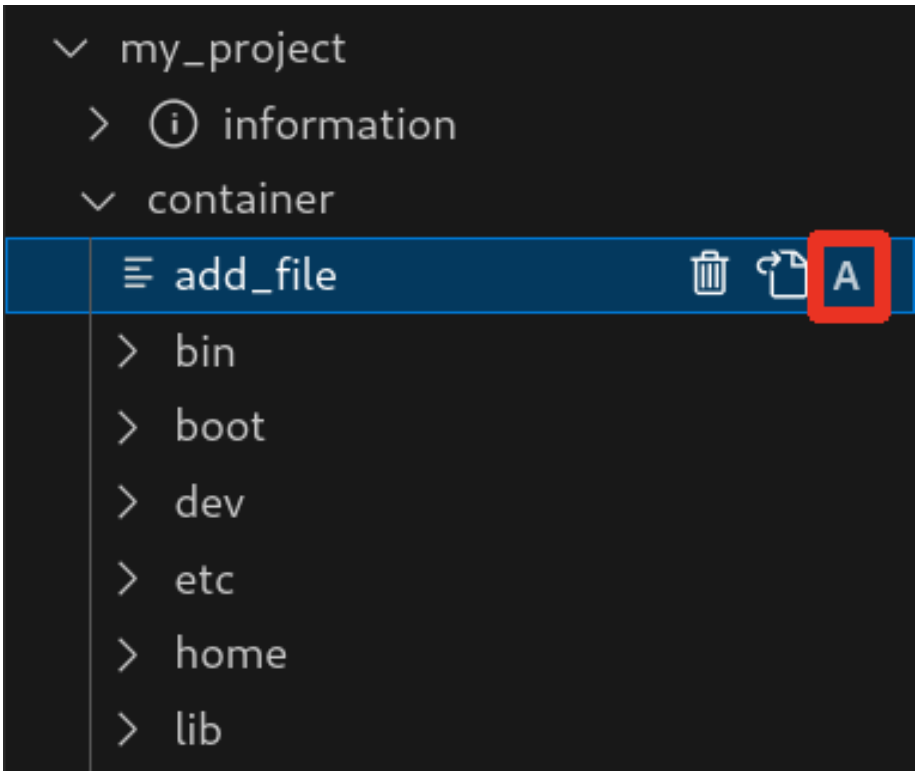


図 3.316 追加されたファイルの表示

また、「図 3.317. container/resources 下にフォルダーを追加するボタン」 の赤枠で囲われている表記のボタンをクリックすることで、ファイルの追加と同様の操作でディレクトリを追加することが可能です。

追加したディレクトリも同様に "A" というマークが表示されます。

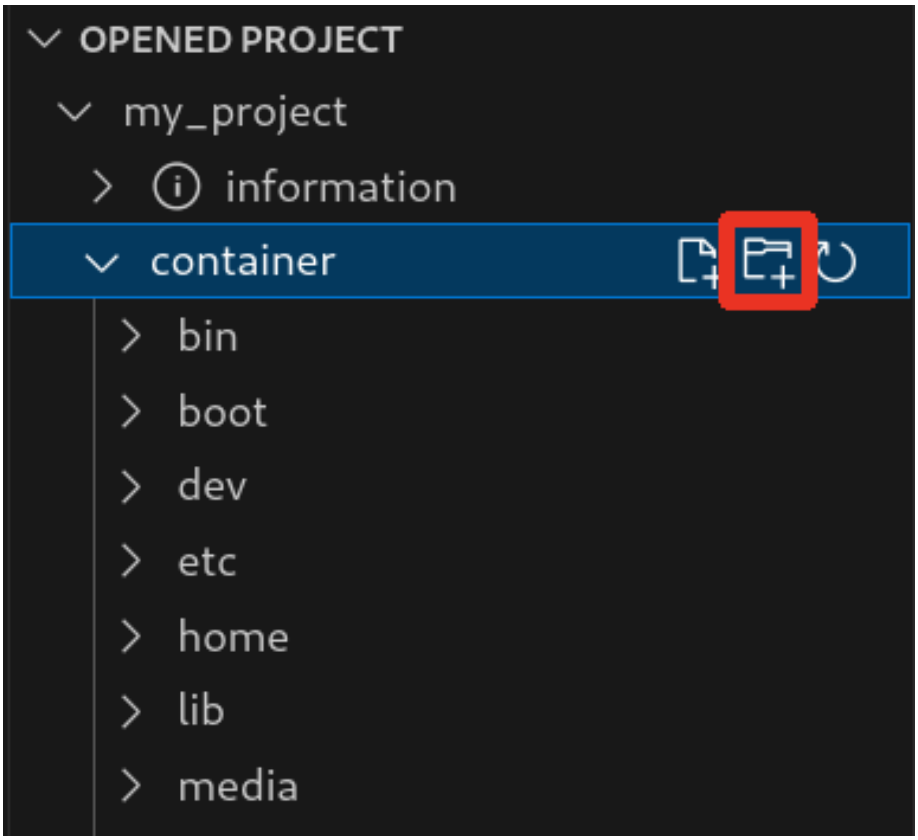


図 3.317 container/resources 下にフォルダーを追加するボタン

3.18.4.4. container/resources 下にあるファイルを開く

「図 3.318. container/resources 下にあるファイルを開くボタン」 の赤枠で囲われている表記のボタンをクリックすることで、プロジェクトディレクトリにある container/resources 下のファイルをエディタに表示することができます。

この例では、プロジェクトディレクトリにある container/resources 下の add_file をエディタに表示します。

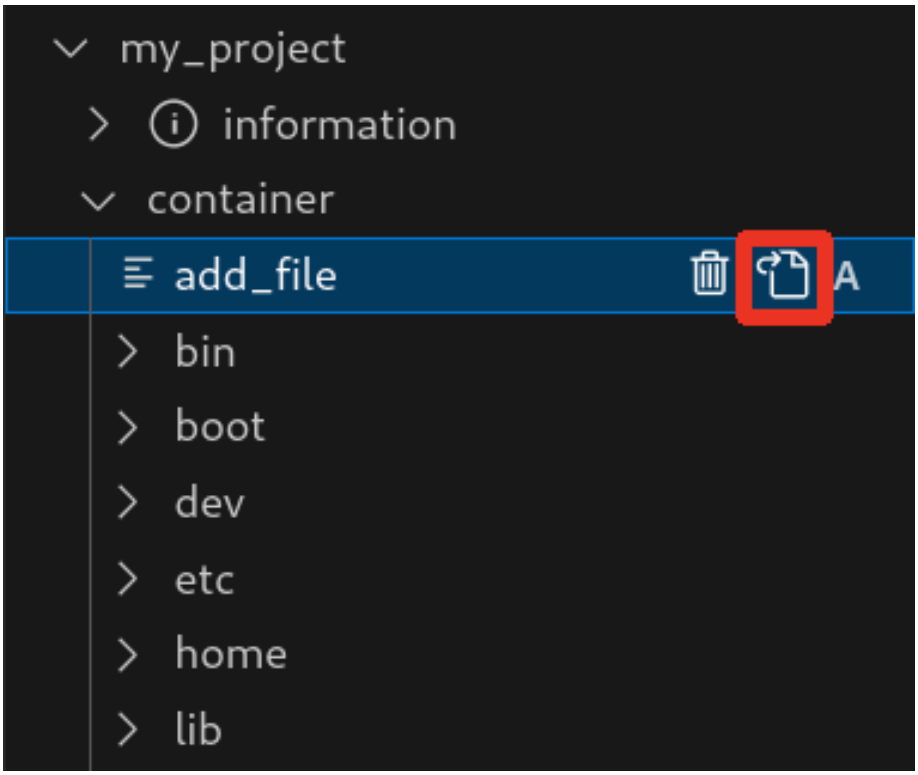


図 3.318 container/resources 下にあるファイルを開くボタン

3.18.4.5. container/resources 下にあるファイルおよびフォルダーの削除

「図 3.318. container/resources 下にあるファイルを開くボタン」 の赤枠で囲われている表記のボタンをクリックすることで、container/resources 下にあるファイルを削除することができます。

この例では、プロジェクトディレクトリにある container/resources 下の add_file を削除します。

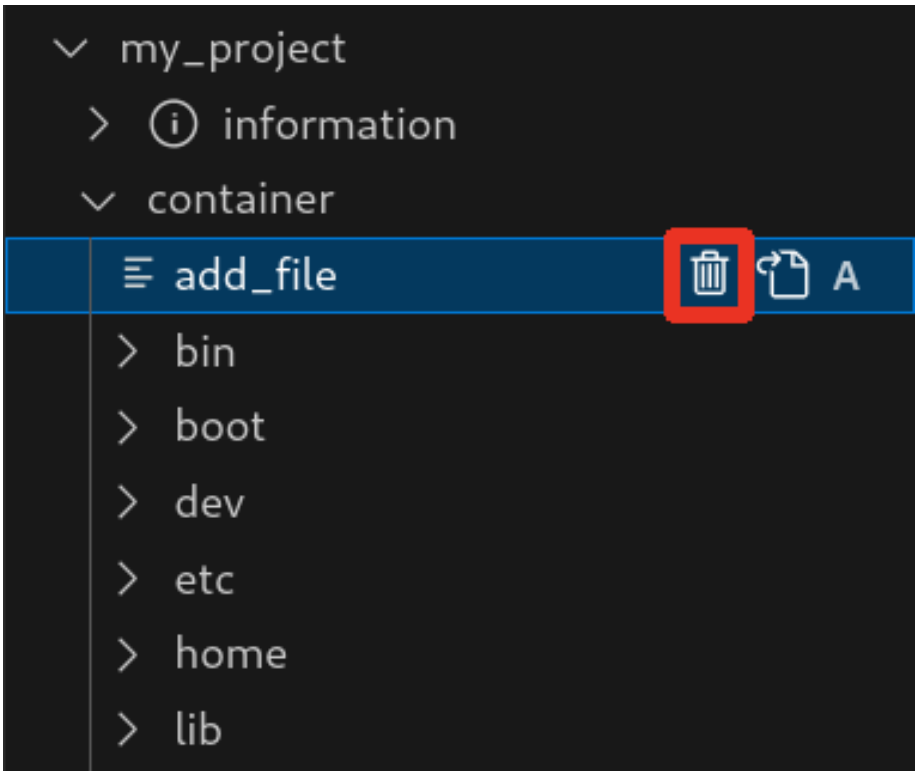


図 3.319 container/resources 下にあるファイルを削除するボタン

ディレクトリも同様に「図 3.318. container/resources 下にあるファイルを開くボタン」 の赤枠で囲われている表記のボタンをクリックすることで削除することができます。

3.18.4.6. コンテナ内のファイルを container/resources 下に保存

「図 3.320. コンテナ内のファイルを container/resources 下に保存するボタン」 の赤枠で囲われている表記のボタンをクリックすることで、コンテナ内にあるファイルをプロジェクトディレクトリにある container/resources 下に保存します。

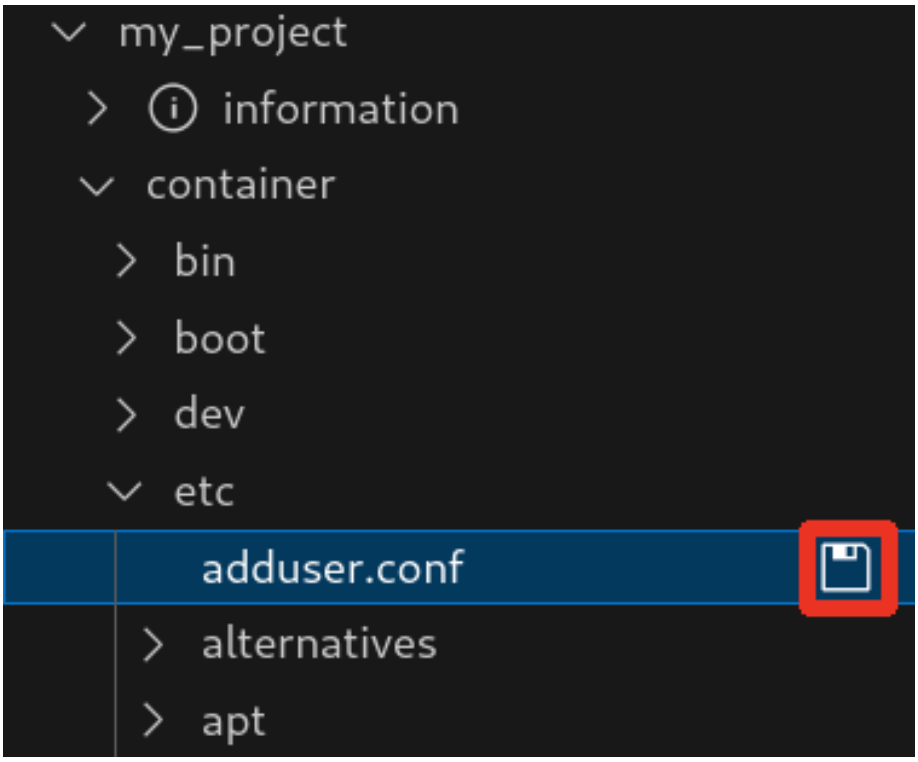


図 3.320 コンテナ内のファイルを container/resources 下に保存するボタン

ファイルが container/resources 下に保存されると、「図 3.321. 編集前のファイルを示すマーク」 に示すように、ファイル名の右側に "U" のマークが表示されます。

"U" のマークはプロジェクトディレクトリにある container/resources 下のファイルとコンテナ内にあるファイルの内容が同一であることを示します。

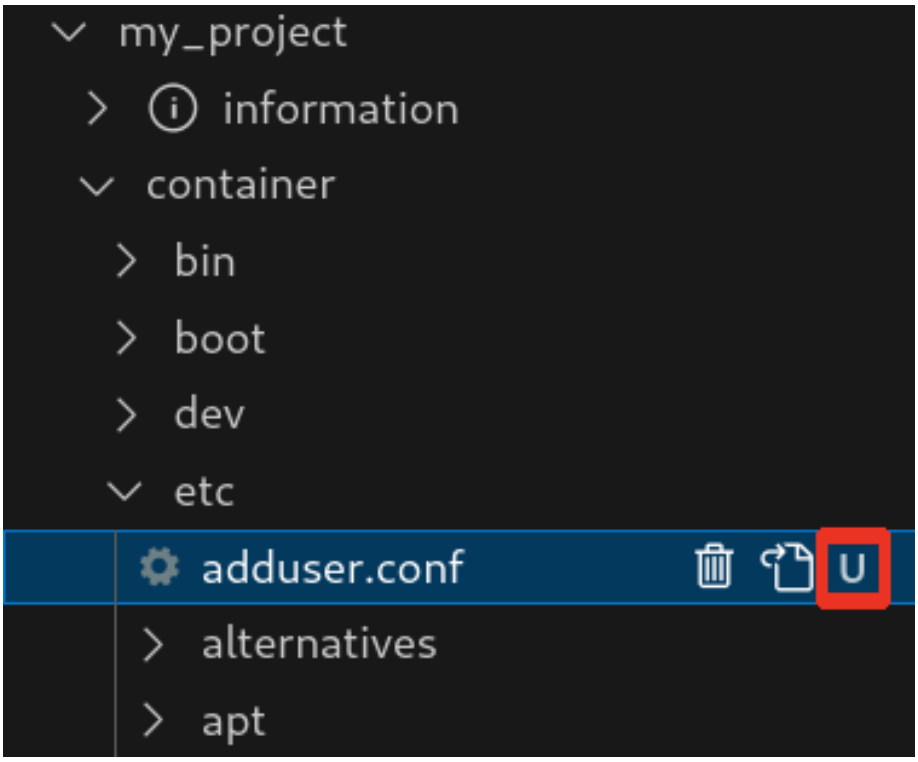


図 3.321 編集前のファイルを示すマーク

container/resources 下にあるファイルを編集して再表示すると、「図 3.322. 編集後のファイルを示すマーク」 に示すように、ファイル名の右側に "M" のマークが表示されます。

"M" のマークはプロジェクトディレクトリにある container/resources 下のファイルとコンテナ内にあるファイルの内容に差があることを示します。

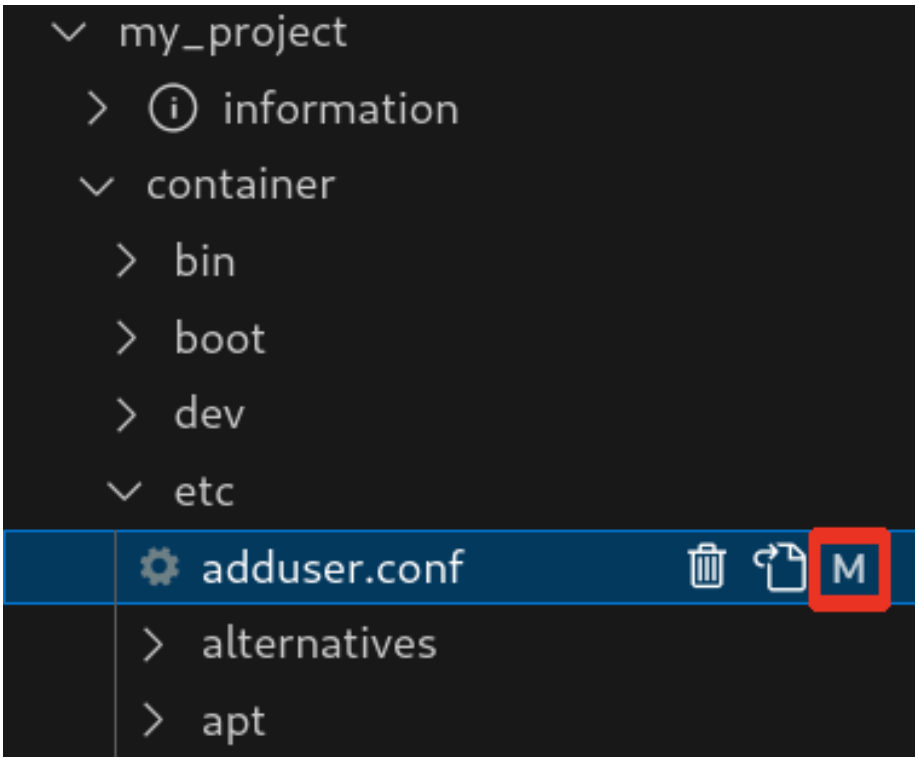


図 3.322 編集後のファイルを示すマーク

3.18.4.7. エラー表示

container/resources 下とコンテナ内にあるファイルまたはディレクトリを比較して、同名でかつファイルの種類が異なる場合、「図 3.323. コンテナ内にコピーされないことを示すマーク」 に示すように、ファイル名の右側に "E" のマークが表示されます。

"E" のマークが表示された場合、そのファイルまたはディレクトリは [Generate development swu] または [Generate release swu] を実行してもコンテナにコピーされません。

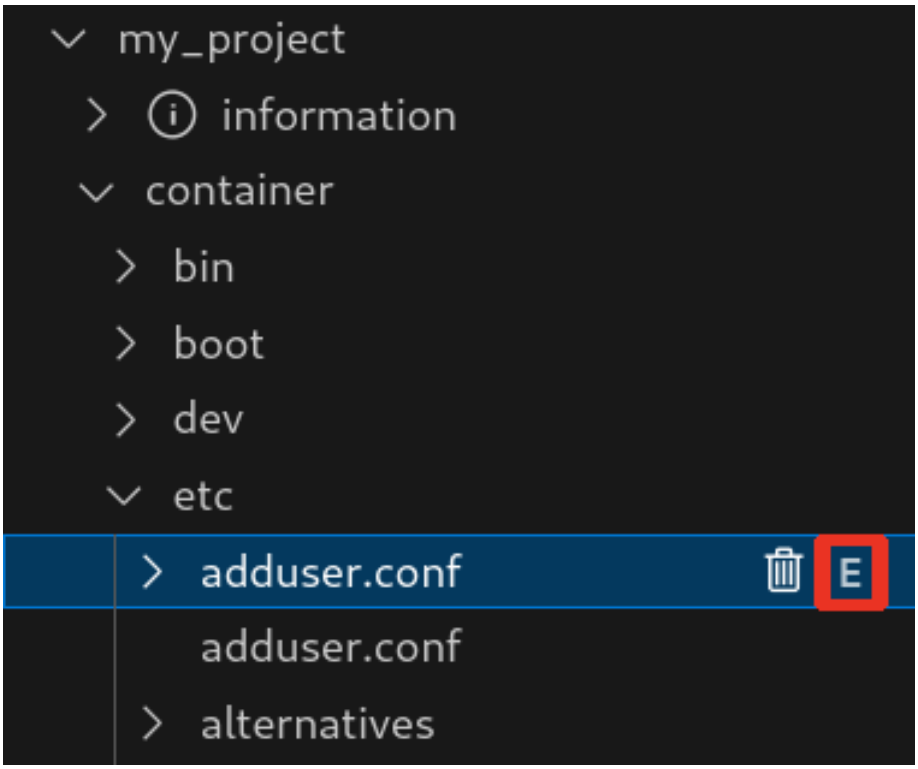


図 3.323 コンテナ内にコピーされないことを示すマーク

3.18.5. Armadillo に転送するディレクトリ及びファイル

コンテナイメージ以外に、以下に示すディレクトリやファイルを Armadillo に転送します。ここでは、プロジェクト名は `my_project` としています。

Armadillo に転送するディレクトリ及びファイル

- my_project/app/build
- my_project/app/lib

3.18.6. Armadillo 上でのセットアップ

3.18.6.1. アプリケーション実行用コンテナイメージのインストール

「3.18.3.9. アプリケーション実行用コンテナイメージの作成」 で作成した `development.swu` を「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールしてください。

インストール後に自動で Armadillo が再起動します。

3.18.6.2. ssh 接続に使用する IP アドレスの設定

VS Code 上で ABOSDE(Armadillo Base OS Development Environment) から、ABOS Web が動作している Armadillo の一覧を確認し、指定した Armadillo の IP アドレスを ssh 接続に使用することができます。ただし、ATDE のネットワークを NAT に設定している場合は Armadillo がリストに表示されません。

「図 3.324. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲われているボタンをクリックすることで、ローカルネットワーク上で ABOS Web が実行されている Armadillo をスキャンすることができます。

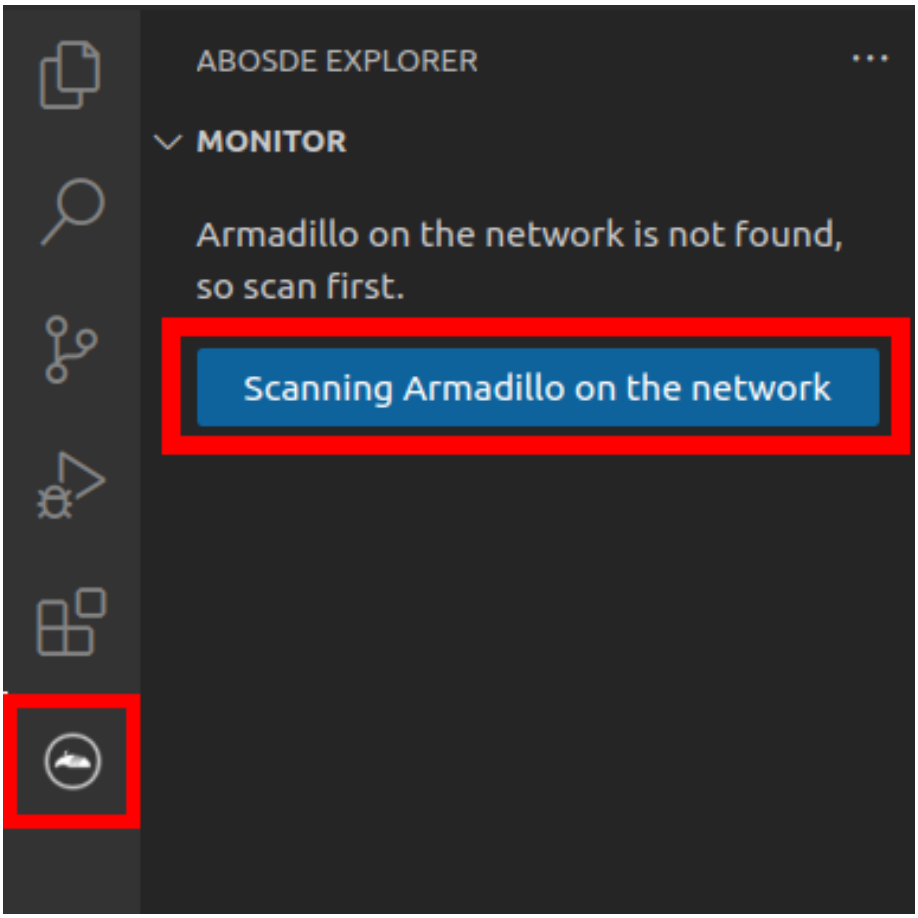


図 3.324 ABOSDE で ローカルネットワーク上の Armadillo をスキャンする

「図 3.325. ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する」 の赤枠で囲われているマークをクリックすることで、指定した Armadillo の IP アドレスを ssh 接続に使用する IP アドレスに設定することができます。

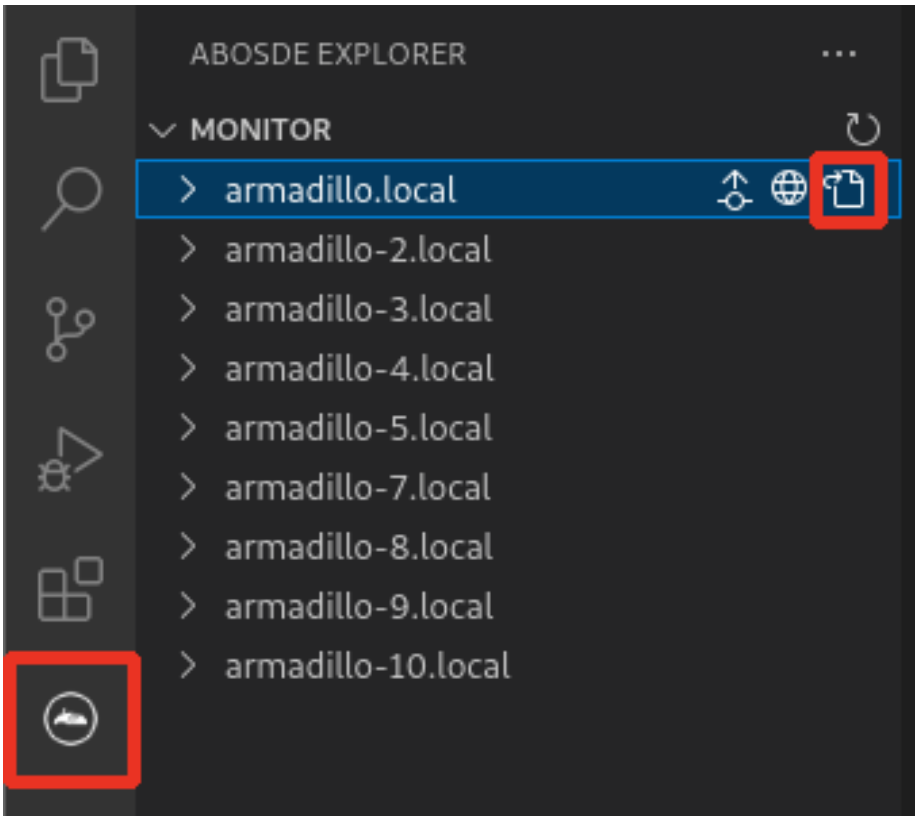


図 3.325 ABOSDE を使用して ssh 接続に使用する IP アドレスを設定する

「図 3.326. ABOSDE に表示されている Armadillo を更新する」 の赤枠で囲われているマークをクリックすることで、ABOSDE に表示されている Armadillo を更新することができます。

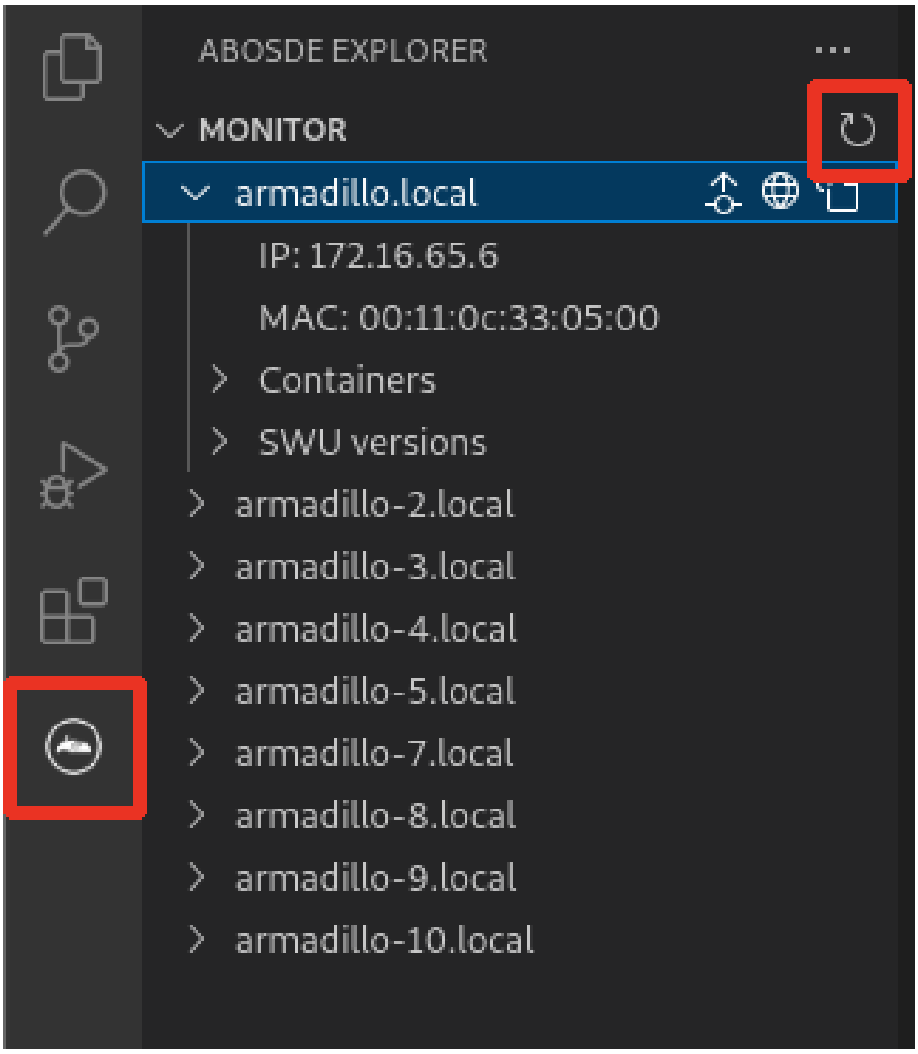


図 3.326 ABOSDE に表示されている Armadillo を更新する

ATDE のネットワークを NAT に設定している場合や、ABOS Web を起動していない場合等、ABOSDE のリストに Armadillo が表示されない場合は、プロジェクトディレクトリに入っている config/ssh_config ファイルを編集して IP アドレスを書き換えてください。

```
[ATDE ~/my_project]$ code config/ssh_config
Host Armadillo
    Hostname x.x.x.x ❶
    User root
    IdentityFile ${HOME}/.ssh/id_ed25519_vscode
    UserKnownHostsFile config/ssh_known_hosts
    StrictHostKeyChecking accept-new
```

図 3.327 ssh_config を編集する

❶ Armadillo の IP アドレスに置き換えてください。



Armadillo を初期化した場合や、プロジェクトを実行する Armadillo を変えた場合は,プロジェクトの config/ssh_known_hosts に保存されている公

開鍵で Armadillo を認識できなくなります。その場合はファイルを削除するか、「Setup environment」タスクを再実行してください。

3.18.6.3. アプリケーションの実行

VS Code の左ペインの [my_project] から [App run on Armadillo] を実行すると、実行ファイルや共有ライブラリを作成した後、アプリケーションが Armadillo へ転送されて起動します。

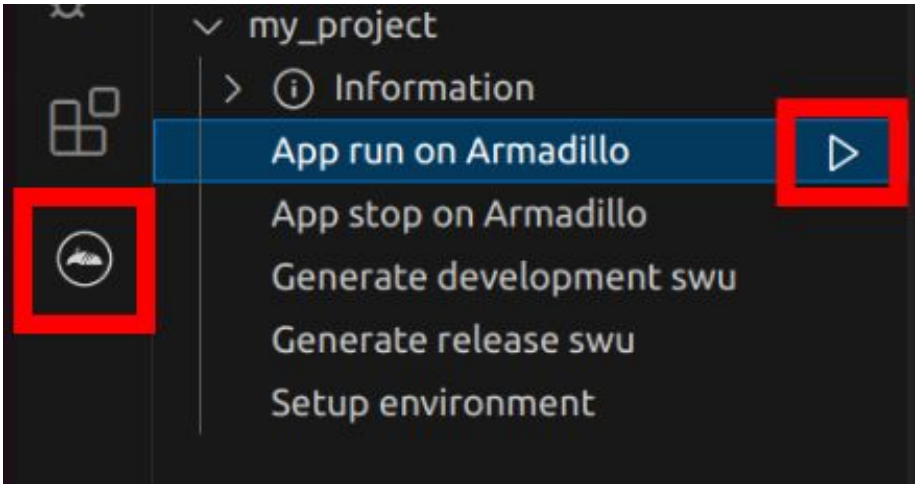


図 3.328 Armadillo 上でアプリケーションを実行する

VS Code のターミナルに以下のメッセージが表示されることがあります。これが表示された場合は yes と入力して下さい。

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

図 3.329 実行時に表示されるメッセージ

アプリケーションを終了するには VS Code の左ペインの [my_project] から [App stop on Armadillo] を実行してください。

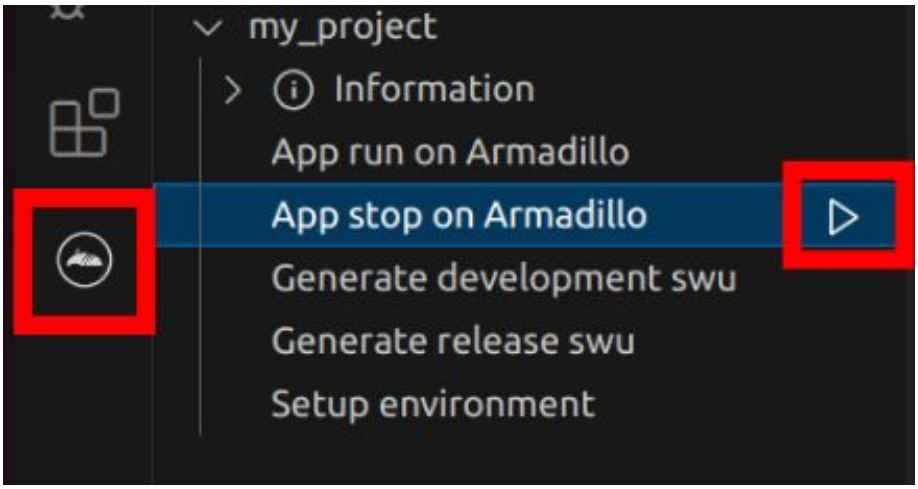


図 3.330 アプリケーションを終了する

3.18.7. SBOM 生成に関する設定

SWU イメージ作成時に、同時に SBOM を生成することができます。詳細は 「3.19. SBOM 生成に関わる設定を行う」 を参照してください。

3.18.8. リリース版のビルド

ここでは完成したアプリケーションをリリース版としてビルドする場合の手順について説明します。

VS Code の左ペインの [my_project] から [Generate release swu] を実行すると、リリース版のアプリケーションを含んだ SWU イメージが作成されます。事前に 「5.4.1. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。

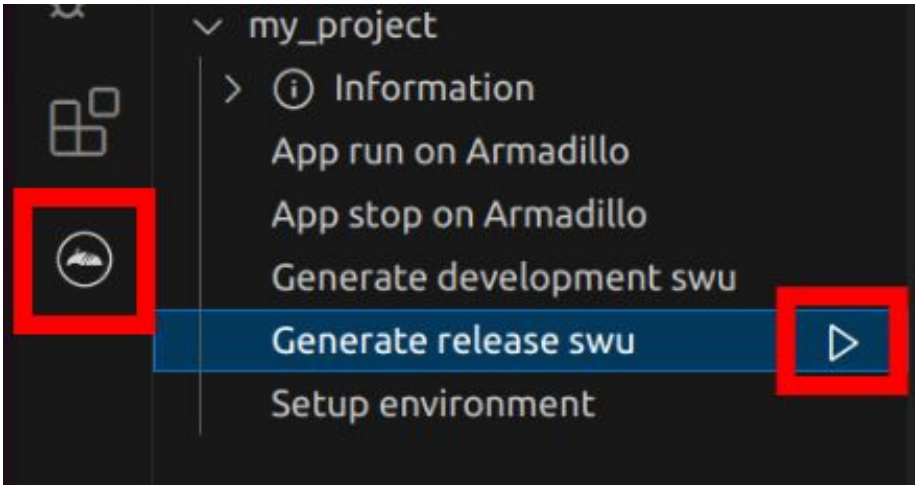


図 3.331 リリース版をビルドする



リリース版の SWU イメージには、開発用の機能は含まれていません。このため、リリース版の SWU イメージをインストールした Armadillo では、[App run on Armadillo] を使用したリモート実行は使用できません。

3.18.9. 製品への書き込み

作成した SWU イメージは `my_project` ディレクトリ下に `release.swu` というファイル名で保存されています。

この SWU イメージを 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールすると、Armadillo 起動時にアプリケーションも自動起動します。

3.18.10. Armadillo 上のコンテナイメージの削除

「6.9.3. コンテナとコンテナに関連するデータを削除する」 を参照してください。

3.19. SBOM 生成に関わる設定を行う

ABOSDE では SWU イメージの生成と同時に SBOM が生成されます。生成される SBOM 名は `SWU イメージ.spdx.json` になります。json 形式で ISO/IEC5962 で国際標準となっている SPDX2.2 のフォーマットに準拠しています。SBOM についての詳細は 「6.37.2. SBOM の提供」 をご参照ください。



SBOM の生成には以下が必要です。

- mkswu （6.4 以上）
- python3-make-sbom パッケージ

以下のコマンドでインストール状況を確認してください。

```
[ATDE ~]$ mkswu --version
mkswu バージョン 6.4
[ATDE ~]$ make_sbom.sh -h
（ヘルプが表示される）
```

図 3.332 必要パッケージの確認

mkswu バージョンが 6.4 未満 またはヘルプが表示されない場合は、「図 3.3. ソフトウェアをアップデートする」 を実行してアップデートしてください。

3.19.1. SBOM 生成に必要なファイルを確認する

SBOM の生成には以下の二つのファイルが必要です。

- コンフィグファイル
- desc ファイル

SBOM の生成にはライセンス情報を示したコンフィグファイルを使用します。コンフィグファイルは `config/sbom_config.yaml.tmpl` になります。SWU イメージ作成時にこのコンフィグファイルからバージョン番号をアップデートした `swu/sbom_config.yaml` が生成されます。リリース時にはコンフィグファイルの内容を確認し、正しい内容に変更してください。各項目の詳細な説明については SPDX specification v2.2.2 (https://spdx.github.io/spdx-spec/v2.2.2/) をご覧ください。

SBOM に含めるコンテナイメージ等の情報については desc ファイルに記載されています。各項目の説明については 「6.37.2.4. SWU イメージと同時に SBOM を作成する」 をご覧ください。

3.20. 生成した SBOM をスキャンする

SBOM の利点のひとつに、スキャンツールに入力することでソフトウェアに含まれる脆弱性を検出することができる点が挙げられます。ここでは、 Google が提供しているオープンソース SBOM スキャンツール OSV-Scanner[11] を用いて、開発したソフトウェアに既知の脆弱性が含まれているかを確認する方法を紹介します。



OSV-Scanner は脆弱性を確認するため、検査対象のソフトウェアの情報を api.osv.dev に送信しています。送信されるのはソフトウェアに関する以下の情報です。

[11]OSV-Scanner: https://github.com/google/osv-scanner

- パッケージ名
- パッケージエコシステム(npm, crates.io など)
- バージョン

3.20.1. OSV-Scanner のインストール

以下の手順はすべて ATDE 上で行います。

OSV-Scanner は GitHub にてビルド済みの実行ファイルが配布されているのでそちらを使用します。OSV-Scanner のリリースページ [https://github.com/google/osv-scanner/releases]から、最新の実行ファイル(osv-scanner_linux_amd64)をクリックしてダウンロードしてください。

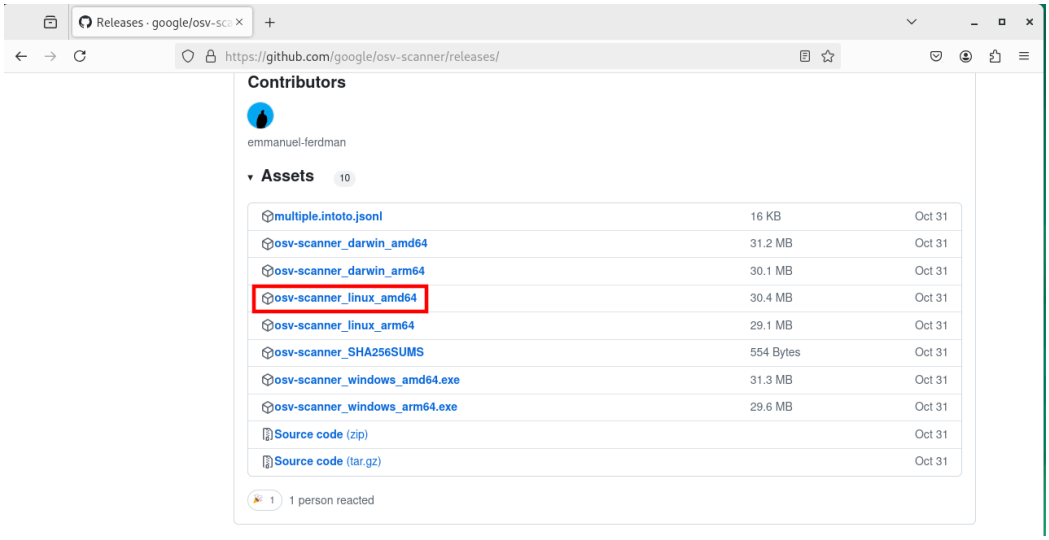


図 3.333 OSV-Scanner の実行ファイルをダウンロード

次に、「図 3.334. OSV-Scanner をインストールする」のコマンドを実行することで、 OSV-Scanner がインストールされます。

```
[ATDE ~]$ sudo install ~/ダウンロード/osv-scanner_linux_arm64 /usr/local/bin/osv-scanner
```

図 3.334 OSV-Scanner をインストールする

osv-scanner --help コマンドを実行して、正しくインストールされていることを確認してください。

```
[ATDE ~]$ osv-scanner --help
NAME:
   osv-scanner - scans various mediums for dependencies and checks them against the OSV database

USAGE:
   osv-scanner [global options] command [command options]

VERSION:
   1.9.1
```

```
COMMANDS:
   scan     scans various mediums for dependencies and matches it against the OSV database
   fix      [EXPERIMENTAL] scans a manifest and/or lockfile for vulnerabilities and suggests changes for remediating them
   help, h  Shows a list of commands or help for one command

GLOBAL OPTIONS:
   --help, -h     show help
   --version, -v  print the version
```

図 3.335 OSV-Scanner がインストールされたことを確認する

これで OSV-Scanner のインストールが完了しました。

3.20.2. OSV-Scanner でソフトウェアの脆弱性を検査する

「図 3.336. OSV-Scanner を用いて SBOM をスキャンする」に示すコマンドを実行することで、ソフトウェアに含まれる既知の脆弱性を検出します。ここでは例として ABOSDE で開発したアプリケーションの SBOM をスキャンします。SBOM 生成については 「6.37.2.4. SWU イメージと同時に SBOM を作成する」 を参照してください。

```
[ATDE ~]$ osv-scanner scan --sbom ~/my_project/development.swu.spdx.json --format markdown ❶
Scanned /home/atmark/my_project/development.swu.spdx.json as SPDX SBOM and found 97 packages
5 unimportant vulnerabilities have been filtered out.
Filtered 5 vulnerabilities from output
| OSV URL | CVSS | Ecosystem | Package | Version | Source |
| --- | --- | --- | --- | --- | --- |
| https://osv.dev/CVE-2022-3715 | 7.8 | Debian | bash | 5.1-2+deb11u1 | development.swu.spdx.json |
| https://osv.dev/CVE-2016-2781 | 6.5 | Debian | coreutils | 8.32-4 | development.swu.spdx.json |
| https://osv.dev/CVE-2021-33560 | 7.5 | Debian | libgcrypt20 | 1.8.7-6 | development.swu.spdx.json |
| https://osv.dev/CVE-2024-2236 |  | Debian | libgcrypt20 | 1.8.7-6 | development.swu.spdx.json |
```

図 3.336 OSV-Scanner を用いて SBOM をスキャンする

❶ 今回は見やすさのために format を markdown に設定しています

上記の例では、 development.swu に含まれるソフトウェアから既知の重要な脆弱性が 5 件検出されました。OSV URL 列の URL にアクセスすることで各脆弱性の詳細を確認することができます。

3.21. システムのテストを行う

Armadillo 上で動作するシステムの開発が完了したら、製造・量産に入る前に開発したシステムのテストを行ってください。

テストケースは開発したシステムに依ると思いますが、 Armadillo で開発したシステムであれば基本的にテストすべき項目について紹介します。

3.21.1. ランニングテスト

長期間のランニングテストは実施すべきです。

ランニングテストで発見できる現象としては、以下のようなものが挙げられます。

- 長期間稼働することでソフトウェアの動作が停止してしまう

  開発段階でシステムを短い時間でしか稼働させていなかった場合、長期間ランニングした際になんらかの不具合で停止してしまう可能性が考えられます。

  開発が完了したら必ず、長時間のランニングテストでシステムが異常停止しないことを確認するようにしてください。

  コンテナの稼働情報は `podman stats` コマンドで確認することができます。

- メモリリークが発生する

  アプリケーションのなんらかの不具合によってメモリリークが起こる場合があります。

  また、運用時の Armadillo は基本的に overlayfs で動作しています。そのため、外部ストレージやボリュームマウントに保存している場合などの例外を除いて、動作中に保存したデータは tmpfs (メモリ)上に保存されます。よくあるケースとして、動作中のログなどのファイルの保存先を誤り、 tmpfs 上に延々と保存し続けてしまうことで、メモリが足りなくなってしまうことがあります。

  長時間のランニングテストで、システムがメモリを食いつぶさないかを確認してください。

  メモリの空き容量は「図 3.337. メモリの空き容量の確認方法」に示すように free コマンドで確認できます。

```
[armadillo ~]# free -h
              total        used        free      shared  buff/cache   available
Mem:           1.9G      327.9M        1.5G        8.8M       97.4M        1.5G
Swap:       1024.0M           0     1024.0M
```

図 3.337 メモリの空き容量の確認方法

3.21.2. 異常系における挙動のテスト

開発したシステムが、想定した条件下で正しく動作することは開発時点で確認できていると思います。しかし、そのような正常系のテストだけでなく、正しく動作しない環境下でどのような挙動をするのかも含めてテストすべきです。

よくあるケースとしては、動作中に電源やネットワークが切断されてしまった場合です。

電源の切断時には、 Armadillo に接続しているハードウェアに問題はないか、電源が復旧した際に問題なくシステムが復帰するかなどをよくテストすると良いです。

ネットワークの切断時には、再接続を試みるなどの処理が正しく実装されているか、 Armadillo とサーバ側でデータなどの整合性が取れるかなどをよくテストすると良いです。

この他にもシステムによっては多くの異常系テストケースが考えられるはずですので、様々な可能性を考慮しテストを実施してから製造・量産ステップに進んでください。

3.22. ユーザー設定とユーザーデータを一括削除する

ユーザー設定とユーザーデータを一括削除することができます。ユーザー設定の削除では ABOS Web から設定できる以下の項目を削除します。

- ネットワーク設定
  - LAN、WLAN、WWAN の設定を全て削除します。WLAN はクライアント設定とアクセスポイント設定の両方を削除します。
- DHCP 設定
- NAT 設定
- VPN 設定
- NTP 設定

ABOS Web から設定できるものであっても以下は削除されません。

- Rest API トークン
- UI カスタマイズの内容

ユーザーデータの削除では以下のデータを削除します。

- /var/app/volumes ディレクトリ下のファイルを全て
- /var/log ディレクトリ下のファイルを全て

ユーザー設定とユーザーデータを削除するには Armadillo 上で abos-ctrl reset-default コマンドを使用します。

```
[armadillo ~]# abos-ctrl reset-default ❶
Run with dry-run mode.
rm -f /etc/NetworkManager/system-connections/*
persist_file -r /etc/NetworkManager/system-connections
persist_file -r /etc/dnsmasq.d
rc-service dnsmasq restart
/etc/init.d/iptables save
sed -i -e '/NETAVARK/d' /etc/iptables/rules-save
persist_file /etc/iptables/rules-save
podman stop -a
find /var/app/volumes /var/log -mindepth 1 -delete
If you want to actually run the above commands,
add the -f/--force option.
```

図 3.338 削除されるユーザー設定とユーザーデータを確認

❶ 何もオプションを付けない場合、 DRY-RUN モードとなり実際に削除は行われません。実際に削除を行う時に実行されるコマンドが表示されるのみです。

表示されたコマンドを確認し実際に削除されてもよい場合は、以下のように -f オプションを付けて実行してください。

```
[armadillo ~]# abos-ctrl reset-default -f
rm -f /etc/NetworkManager/system-connections/*
persist_file -r /etc/NetworkManager/system-connections
persist_file -r /etc/dnsmasq.d
rc-service dnsmasq restart
```

```
/etc/init.d/iptables save
sed -i -e '/NETAVARK/d' /etc/iptables/rules-save
persist_file /etc/iptables/rules-save
podman stop -a
find /var/app/volumes /var/log -mindepth 1 -delete
Starting clone to /dev/mmcblk2p1
Cloning rootfs
Updating appfs snapshots
Reusing up-to-date bootloader
Rollback clone successful
WARNING: Rebooting!
```

図 3.339 実際にユーザー設定とユーザーデータを削除する

コマンド実行後は自動的に Armadillo が再起動します。

ABOS Web または Rest API から実行することもできます。ABOS Web から実行する場合は 「6.12.8. ユーザー設定とユーザーデータの削除」 を参照してください。Rest API から実行する場合は 「6.12.6.16. Rest API : ユーザー設定とユーザーデータの管理」 を参照してください。



再起動後、再び設定が必要な場合は ABOS Web や REST API を使用して行ってください。特に Armadillo Twin を利用している場合は、必ずネットワークの再設定を行ってください。

4. 量産編

本章では Armadillo を組み込んだ最終製品をお客様が製造・量産するうえで、必要となる情報や作業について記載します。

- 「4.1. 概略」では、量産の進め方に関する概略を記載します。
- 「4.2. BTO サービスを使わない場合と使う場合の違い」では、BTO(Build To Order) サービスに関する説明をします。
- 「4.3. 量産時のイメージ書き込み手法」では、開発を完了したソフトウェアの量産用イメージ作成・書き込み方法を説明します。
  - 「4.4. インストールディスクを用いてイメージ書き込みする」は、インストールディスクを使用する方法を説明します。
  - 「4.5. SWUpdate を用いてイメージ書き込みする」は、SWUpdate を使用する方法を説明します。

4.1. 概略

量産の進め方の概略図を「図 4.1. Armadillo 量産時の概略図」に示します。お客様の製品仕様や製造工程の要件によってはこの例とは違った工程順となる場合や、工程の追加・削除がある可能性があります。

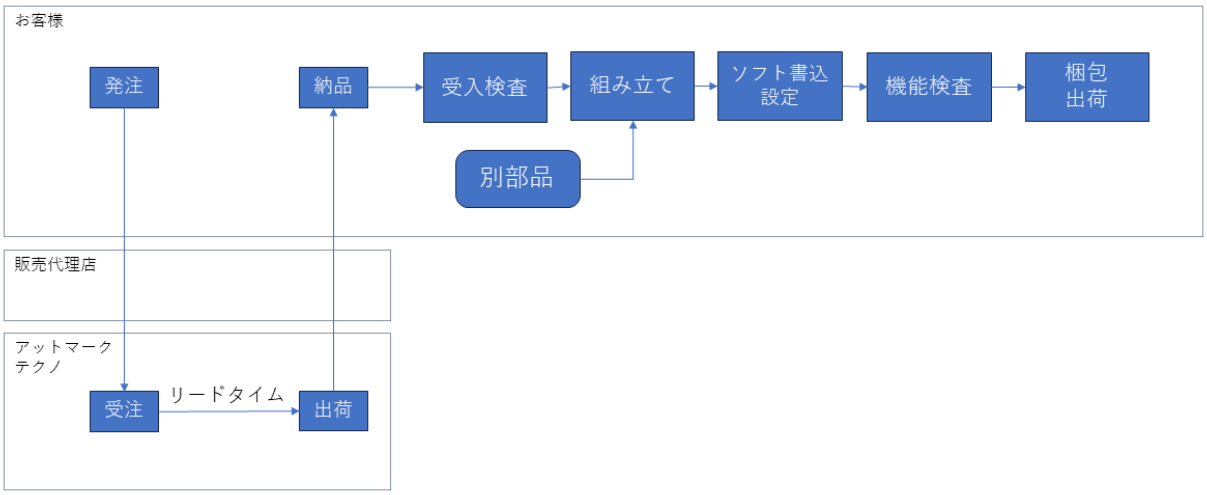


図 4.1 Armadillo 量産時の概略図

4.1.1. Armadillo Twin を契約する

Armadillo Twin を使用したデバイス運用管理を行う場合は、量産モデルの発注とは別に Armadillo Twin の契約が必要となります。Armadillo Twin の契約の詳細については、弊社営業、ご利用の販売代理店にお問い合わせください。

4.1.2. リードタイムと在庫

量産モデルを発注後、お客様に納品されるまでにリードタイムが発生します。開発セットや少量の量産モデル購入の場合、アットマークテクノや代理店在庫によって、短期間で納品できることもあります。しかし、まとまった数量の量産モデルの場合、納品までにお時間をいただくことがあります。新規に製品を量産・出荷する場合はリードタイムを考慮したスケジューリングをお願いします。また、リピート製造をする場合でも、欠品を起こさないよう適切な在庫の確保をお願いいたします。

リードタイムは状況・タイミングによって異なりますので、都度、弊社営業、ご利用の販売代理店にお問い合わせください。

4.1.3. Armadillo 納品後の製造・量産作業

お客様が Armadillo を納品後に次に示すようなキッティング作業、組み立て、検査を実施し出荷を行います。

- ソフトウェア書き込み
  - Armadillo Base OS やアプリケーションコンテナイメージの書き込み
  - 設定ファイルの書き込み
- 別部品の組み立て
  - SD カード/ SIM カード/ RTC バックアップ電池等の接続
  - 拡張基板接続やセンサー・外部機器の接続
  - お客様専用筐体への組み込み
- 検査
  - Armadillo の受け入れ検査
  - 組み立て後の通電電検・機能検査
  - 目視検査
- 梱包作業
- 出荷作業

有償の BTO サービスを利用することで、これらの作業の一部をアットマークテクノへ委託・実施済みの状態で Armadillo を納品することも可能です。費用はいただきますがお客様による工程立ち上げ、場所の確保、作業者の教育、品質管理等のトータルコストを考えると委託した方が安く済むケースが多いです。

また、 BTO サービスではお受けできないようなキッティング、検査、作業については、実施可能な業者をご紹介する等、個別の対応をすることで解決できる場合もございます。詳しくは弊社担当の営業、またはご利用の販売代理店にご相談ください。

4.2. BTO サービスを使わない場合と使う場合の違い



図 4.2 BTO サービスで対応する範囲

4.2.1. BTO サービスを利用しない(標準ラインアップ品)

有償の量産サービスを利用しない場合、標準ラインアップ仕様での納品となります。大きく分けて試作開発用途で使う「開発セット」と量産向けの「量産モデル」の 2 種類があります。量産用途では「量産モデル」をご利用ください。

「量産モデル」には AC アダプタ等のオプション品が付属されておりませんので、内容物を確認の上、発注をお願いいたします。ラインアップ一覧については「2.2. 製品ラインアップ」をご確認ください。

4.2.1.1. 標準ラインアップ品に書き込まれているソフトウェア

標準ラインアップ品に書き込まれるソフトウェアイメージ(Armadillo Base OS)は、アットマークテクノで公開している標準イメージとなります。また、ソフトウェアバージョンは指定することができず、ランニングチェンジで随時最新版を適用していきます。このため、納品後の Armadillo 個体では、開発段階で評価した Armadillo Base OS と異なるバージョンが書き込まれている可能性があります。

また、アプリケーションコンテナについては何も書き込まれていない状態となります。

納品後、お客様の量産工程でソフトウェアの書き込み作業が必要となります。詳しくは「4.3. 量産時のイメージ書き込み手法」をご確認ください。

4.2.2. BTO サービスを利用する

BTO サービスは、セミオーダー式メニューから選択して Armadillo の量産品を一括手配いただける有償サービスです。標準ラインアップ品の仕様をベースとして、搭載するモジュールの種類やケース、 AC アダプタの有無、お客様支給品の SD カードや SIM カードの接続、お客様ご指定のソフトウェアイメージ書き込みなど、メニュー内から指定可能なキッティング項目を選択・指定することが可能です。

販売代理店またはアットマークテクノの窓口からお申し込みいただけます。

製品ごとに、対応できる作業とできない作業がございます。また、販売直後の製品の場合など BTO サービスに未対応である場合もあります。詳しくは Armadillo サイトの BTO サービス [https://armadillo.atmark-techno.com/services/customize/bto] をご確認ください。

4.3. 量産時のイメージ書き込み手法

量産時に必要な手順は最終製品によって異なりますが、開発したソフトウェアを Armadillo に書き込む手順は必ず実施することになります。Armadillo Base OS 搭載製品において、量産時に任意のソフトウェアを書き込む際には、以下の 2 つの手法のどちらかを用いると実現できます。

- インストールディスクを用いてソフトウェアを書き込む

・ SWUpdate を用いてソフトウェアを書き込む

ただし、SWUpdate は運用中の Armadillo のアップデート機能であり、量産時のイメージ書き込みは本来の用途でないため、基本的にはイメージ書き込みに特化しているインストールディスクを用いた方法を選択してください。

それぞれの手法の特徴を「表 4.1. インストールディスクと SWUpdate によるソフトウェア書き込みの比較」にまとめます。ソフトウェア書き込み工程を決定する際の参考にしてください。

表 4.1 インストールディスクと SWUpdate によるソフトウェア書き込みの比較

手段	メリット	デメリット
インストールディスク	・ インストールの前後処理を行なうシェルスクリプトのテンプレートが用意されている ・ インストールの前後処理は、microSD カード内にシェルスクリプトを配置するだけなので製造担当者にも編集しやすい	・ 動いているシステムをそのままインストールディスクにするため、出荷時の標準イメージから手動で同じ環境を構築する手順が残らない
SWUpdate	・ 必ず必要となる初回アップデートを別途実行する必要がない	・ SWU イメージの作成には、mkswu を使用できる環境と desc ファイルの記述方法を知る必要があるため、開発担当者以外に SWU イメージを更新させるハードルが少し高い ・ ログの取得など、インストール前後の処理が必要な場合は自分で記述する必要がある

量産時のイメージ書き込みにインストールディスクを使用する場合は、「4.4. インストールディスクを用いてイメージ書き込みする」に進んでください。

量産時のイメージ書き込みに SWUpdate を使用する場合は、「4.5. SWUpdate を用いてイメージ書き込みする」に進んでください。

4.4. インストールディスクを用いてイメージ書き込みする

「3.3.6. インストールディスクについて」でも紹介したとおり、 Armadillo Base OS 搭載製品では、開発が完了した Armadillo のクローン用インストールディスクを作成することができます。

以下では、クローン用インストールディスクを作成する手順を準備段階から紹介します。

4.4.1. /etc/swupdate_preserve_file への追記

Armadillo Base OS のバージョンを最新版にしておくことを推奨しています。最新版でない場合は、バージョンが古いゆえに以下の作業を実施出来ない場合もありますので、ここで Armadillo Base OS のバージョンをアップデートしてください。

ここでは SWUpdate を使用して Armadillo Base OS のアップデートを行ないますが、このアップデートを行なうと、/etc/swupdate_preserve_files に記載の無いファイルは消えてしまいます。Armadillo Base OS のルートファイルシステム上に消えてほしくないファイルを開発中に配置していた場合は、「図 4.3. 任意のファイルパスを/etc/swupdate_preserve_files に追記する」に示すコマンドを実行することで /etc/swupdate_preserve_files にそのファイルが追記され、アップデート後も保持し続けるようになります。

一部のファイルやディレクトリは初めから /etc/swupdate_preserve_files に記載されている他、`podman commit` したコンテナイメージについてもアップデート後に引き継がれるので、本ドキュメントのサンプルアプリケーションの場合は実行する必要はありません。

```
[armadillo /]# persist_file -p <ファイルのパス>
```

図 4.3 任意のファイルパスを/etc/swupdate_preserve_files に追記する

4.4.2. Armadillo Base OS の更新

「 abos-ctrl update 」で Armadillo Base OS を更新できます。

/etc/swupdate.watch に記載されている URL の SWU イメージでアップデートを行いますので、デフォルトでは 最新の Armadillo Base OS へアップデートします。

また、GW コンテナを使用する場合はコンテナも 2 段階で更新されますので、Base OS の更新で再起動した後にもう一度コマンドを実行してください。



Armadillo Base OS 3.19.1-at.3 以前のバージョンで abos-ctrl update を利用できないか、 /etc/swupdate.watch ファイルに記載されていない場合は任意の URL にある SWU もインストール可能です。

「図 4.4. Armadillo Base OS を最新にアップデートする」に示すコマンドを実行することで Armadillo Base OS を最新版にアップデートできます。

```
[armadillo /]# swupdate -d '-u https://armadillo.atmark-techno.com/
files/downloads/armadillo-iot-a6e/image/baseos-6e-latest.swu'
```

図 4.4 Armadillo Base OS を最新にアップデートする

正常に実行された場合は自動的に再起動します。

4.4.3. パスワードの確認と変更

「3.1.5.1. initial_setup.swu の作成」 で SWUpdate の初回アップデートを行った際に、各ユーザーのパスワード設定をしました。開発中はログインしやすいような単純なパスワードにしていることがよくあるので、製品に適用しないようにこのタイミングで強固なパスワードに変更しておきましょう。

```
[armadillo /]# passwd ❶
Changing password for root
New password: ❷
Retype password: ❸
passwd: password for root changed by root

[armadillo /]# passwd atmark ❹
Changing password for atmark
New password: ❺
Retype password: ❻
passwd: password for atmark changed by root
```

```
[armadillo /]# persist_file /etc/shadow ❼
```

図 4.5 パスワードを変更する

❶ root ユーザのパスワードを変更します。

❷ 新しい root ユーザ用パスワードを入力します。

❸ 再度新しい root ユーザ用パスワードを入力します。

❹ atmark ユーザのパスワードを変更します。

❺ 新しい atmark ユーザ用パスワードを入力します。

❻ 再度新しい atmark ユーザ用パスワードを入力します。

❼ パスワードの変更を永続化させます。

4.4.4. 開発したシステムをインストールディスクにする

Armadillo Base OS では、現在起動しているルートファイルシステム及びブートローダーをそのままインストールディスクイメージとして生成することができます。インストールディスクイメージの生成方法は二種類あります。それぞれの特徴をまとめます。

・ VS Code を使用して生成

ATDE と VS Code を使用して、開発したシステムのインストールディスクイメージを USB メモリ上に生成します。USB メモリは vfat もしくは ext4 形式でフォーマットし、空き容量が 10GB 以上のものを使用してください。VS Code に開発用エクステンションである ABOSDE をインストールする必要があります。

・ コマンドラインから生成

`abos-ctrl make-installer` コマンドを実行すると microSD カードにインストールディスクイメージを生成することができます。コマンド実行前に、Armadillo がインターネットに接続されており、かつ 10GB 以上の空き容量がある microSD カードが挿入されていることを確認してください。microSD カード内のデータはインストールディスク作成時に上書きされて消えてしまうので、必要なデータは予めバックアップを取っておいてください。microSD カード上にインストールディスクイメージを生成した場合、インストール時に任意のシェルスクリプトを実行することが可能です。この機能が必要な場合はコマンドラインからの生成を推奨します。

コマンドラインから生成する場合は、Armadillo の JTAG と SD ブート、U-Boot のコマンドプロンプトを無効化するインストールディスクイメージを生成することができます。

4.4.5. VS Code を使用して生成する

ATDE と VS Code を使用して、開発したシステムのインストールディスクイメージを生成します。「3.1.3. VS Code のセットアップ」 を参考に、 ATDE に VS Code 開発用エクステンションをインストールしてください。VS Code を使用してインストールディスクを生成する場合は以下の手順になります。

・ VS Code を使用したインストールディスク作成用 SWU の生成

・ Armadillo に USB メモリを挿入

・ インストールディスク作成用 SWU を ABOS Web からインストール

・USB メモリ上にインストールディスクイメージを生成



この機能を使用するには、以下に示すバージョンのソフトウェアが必要です。

- ABOSDE 1.6.0 以上
- mkswu 5.3 以上
- abos-base 2.3 以上

4.4.5.1. VS Code を使用したインストールディスク作成用 SWU の生成

VS Code の左ペインの [COMMON PROJECT COMMAND] から [Generate Installer On USB Swu] を実行します。

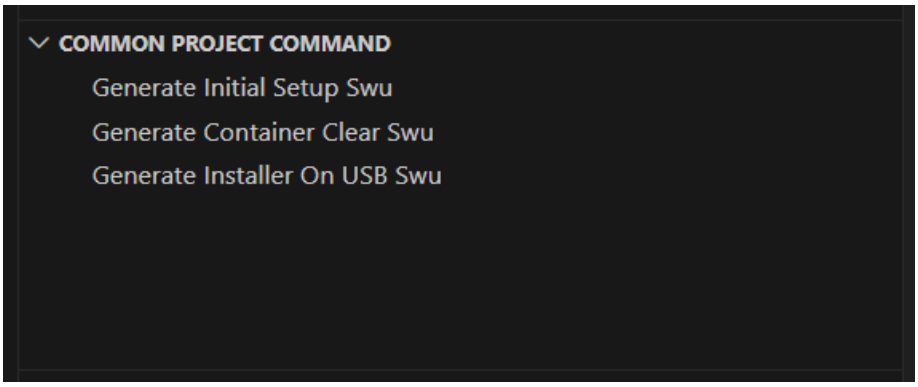


図 4.6 make-installer.swu を作成する

次に、対象製品を選択します。

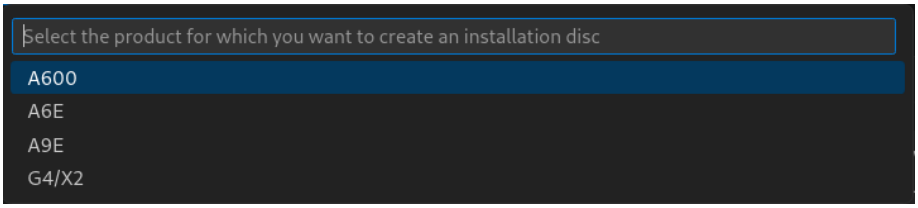


図 4.7 対象製品を選択する

無事に生成された場合、コンソールに以下のログが出力されます。

```
/home/atmark/.vscode/extensions/atmark-techno.armadillo-base-os-development-environment-1.6.0/
shell/desc/make_installer_usb.desc のバージョンを 1 から 2 に変更しました。
Enter pass phrase for /home/atmark/mkswu/swupdate.key: ❶
/home/atmark/mkswu/make_installer_usb.swu を作成しました。
To create Armadillo installer on USB memory install /home/atmark/mkswu/make_installer_usb.swu in
```

```
Armadillo
 *  Terminal will be reused by tasks, press any key to close it.
```

図 4.8 make-installer.swu 生成時のログ

❶ パスワードの入力を求められますので、初期化用 swu を生成したときと同じパスワードを入力します

/home/atmark/mkswu ディレクトリ内に make-installer.swu が作成されます。

4.4.5.2. Armadillo に USB メモリを挿入

Armadillo に電源を投入し、インストールディスクを保存するための USB メモリを挿入してください。



USB メモリは vfat もしくは ext4 形式でフォーマットし、空き容量が 10GB 以上のものを使用してください。Armadillo-IoT ゲートウェイ A6E への USB メモリのマウントは不要です。

インストールディスクイメージは installer.img という名前で保存します。すでに同名のファイルが存在する場合は上書きされます。

4.4.5.3. インストールディスク作成用 SWU を ABOS Web からインストール

ABOS Web を使用して、生成した make-installer.swu をインストールします。「6.12.4. SWU インストール」を参考に make-installer.swu を Armadillo へインストールしてください。実行時は ABOS Web 上に「図 4.9. make-installer.swu インストール時のログ」ようなログが表示されます。

```
make_installer_usb.swu をインストールします。
SWU アップロード完了

SWUpdate v2023.05_git20231025-r0

Licensed under GPLv2. See source distribution for detailed copyright notices.

[INFO ] : SWUPDATE running : [main] : Running on iot-a6e Revision at1

[INFO ] : SWUPDATE started : Software Update started !

[INFO ] : SWUPDATE running : [install_single_image] : Installing pre_script

[INFO ] : SWUPDATE running : [read_lines_notify] : No base os update: copying current os over

[INFO ] : SWUPDATE running : [read_lines_notify] : Waiting for btrfs to flush deleted subvolumes

[INFO ] : SWUPDATE running : [install_single_image] : Installing Copying installer to USB device

[INFO ] : SWUPDATE running : [install_single_image] : Installing swdesc_command_nochroot 'podman kill -a'

[INFO ] : SWUPDATE running : [install_single_image] : Installing swdesc_command_nochroot --stdout-info 'abos-ctrl make-installer --noprompt --output /target/mnt/installer.img'
```

```
[INFO ] : SWUPDATE running : [read_lines_notify] : Using installer image on image file.

[INFO ] : SWUPDATE running : [read_lines_notify] : Would you like to create a windows partition?

[INFO ] : SWUPDATE running : [read_lines_notify] : That partition would only be used for customization script at the end of

[INFO ] : SWUPDATE running : [read_lines_notify] : install, leave at 0 to skip creating it.

[INFO ] : SWUPDATE running : [read_lines_notify] : Custom partition size (MB, [0] or 16 - 364): 0

[INFO ] : SWUPDATE running : [read_lines_notify] : Checking and growing installer main partition

[INFO ] : SWUPDATE running : [read_lines_notify] : Resize device id 1 (/dev/loop0p1) from 513.00MiB to max

[INFO ] : SWUPDATE running : [read_lines_notify] : Copying boot image

[INFO ] : SWUPDATE running : [read_lines_notify] : Copying rootfs

[INFO ] : SWUPDATE running : [read_lines_notify] : Copying appfs

[INFO ] : SWUPDATE running : [read_lines_notify] : At subvol app/snapshots/volumes

[INFO ] : SWUPDATE running : [read_lines_notify] : At subvol app/snapshots/boot_volumes

[INFO ] : SWUPDATE running : [read_lines_notify] : At subvol app/snapshots/boot_containers_storage

[INFO ] : SWUPDATE running : [read_lines_notify] : Trying to shrink the installer partition...

[INFO ] : SWUPDATE running : [read_lines_notify] : Shrinking the installer partition...

[INFO ] : SWUPDATE running : [read_lines_notify] : Cleaning up and syncing changes to disk...

[INFO ] : SWUPDATE running : [read_lines_notify] : Installer updated successfully!

[INFO ] : SWUPDATE running : [read_lines_notify] : -rwxr-xr-x 1 root root 687.0M Jan 23 15:12 /target/mnt/installer.img

[INFO ] : SWUPDATE running : [install_single_image] : Installing post_script

[INFO ] : SWUPDATE running : [read_lines_notify] : Removing unused containers

[INFO ] : SWUPDATE running : [read_lines_notify] : Command 'command podman rm -a -f' output:

[INFO ] : SWUPDATE running : [read_lines_notify] : 9f4f64ec1926d17e75de4060dac4a448e66ca3d9535c408f632e4e2de4bafa4f

[INFO ] : SWUPDATE running : Installation in progress

[INFO ] : SWUPDATE successful ! SWUPDATE successful !

[INFO ] : No SWUPDATE running : Waiting for requests...

swupdate exited
```

```
インストールが成功しました。
```

図 4.9 make-installer.swu インストール時のログ

完了後、USB メモリを抜いてください。もし、エラーが出た場合は Armadillo-IoT ゲートウェイ A6E の電源を再投入してやり直してください。

4.4.5.4. USB メモリ上にインストールディスクイメージを生成

無事に生成が完了した場合、USB メモリ上に installer.img が保存されています。この installer.img を microSD カードに書き込むことでインストールディスクを作成することができます。動作確認については「4.4.6. インストールディスクの動作確認を行う」をご参照ください。これで、VS Code を使用してインストールディスクを生成する方法については終了です。

4.4.6. インストールディスクの動作確認を行う

作成したインストールディスクの動作確認を実施してください。開発に使用した Armadillo 以外の個体が必要になります。また、インストール先の Armadillo の eMMC 内のデータは上書きされて消えるため、必要なデータは予めバックアップを取っておいてください。

「4.4.4. 開発したシステムをインストールディスクにする」 の手順で使用した USB メモリの中に installer.img が存在しますので、ATDE 上でこのイメージをもとに microSD カードにインストールディスクを作成してください。ATDE 上に installer.img をコピーした場合、コマンドは以下のようになります。/dev/sd[X] の [X] は microSD を示す文字を指定してください。

```
[ATDE ~] sudo dd if=installer.img of=/dev/sd[X] bs=1M oflag=direct status=progress
```

上記コマンドで作成した microSD のインストールディスクを、インストール先の Armadillo に挿入してください。その後、SW2 (起動デバイス設定スイッチ)を ON にしてから電源を入れます。 Armadillo がインストールディスクから起動し、自動的にインストールスクリプトが動作します。

しばらくすると「reboot: Power down」と表示されるので、Armadillo の電源を切ります。 その後 Armadillo から microSD カードを抜き、SW2 (起動デバイス設定スイッチ)を OFF にします。 再度電源を投入することで、インストールは完了です。

実際にクローンした Armadillo が想定した通りの動作をすることを確認してください。

4.4.7. コマンドラインから生成する

4.4.7.1. JTAG と SD ブートを無効化する

コマンドラインから生成する場合は、 Armadillo の JTAG と SD ブートを無効にするインストールディスクイメージを生成することができます。

Armadillo の出荷時は、JTAG ポートは有効なままで出荷されます。JTAG が有効なままですと攻撃者が悪意のあるコードを実行する、メモリをダンプして鍵などセキュアな情報を取り出すなどの行為が可能となります。

SD ブートは SD メディアを挿すだけで起動する便利な機能ですが、その反面、SD メディアの盗難や流出によってシステムへの侵入、SD ブートを利用したセキュアブート鍵に対する攻撃が考えられます。

JTAG と SD ブートの無効化はこれらのセキュリティリスクに対して有効です。

JTAG と SD ブートを無効にするには abos-ctrl installer-setting コマンドを実行します。



SD ブートはシステムの復旧の役割も担っています。SD ブートを無効化することによって、eMMC ブートでは起動できない状態に陥った場合、合わせて JTAG の無効化が設定されていると、二度と復旧することができないデバイスになる (廃棄するしかない) 可能性があることに留意してください。



生成したインストールディスクを使用して初期化した Armadillo の JTAG と SD ブートを無効にする設定であり、開発用の Armadillo の JTAG と SD ブートが無効になることはありません。

```
[armadillo /]# abos-ctrl installer-setting
Would you like to disable JTAG in the installer ? [y/N] ❶
y
JTAG disabled setting for production Armadillo has been configured.
Would you like to disable SD boot in the installer ? [y/N] ❷
y
SD boot disabled setting for production Armadillo has been configured.
```

図 4.10 JTAG と SD ブートを無効化する

❶ JTAG を無効化する場合は y を入力します。無効化しない場合は何も入力せず Enter キーを押してください。

❷ SD ブートを無効化する場合は y を入力します。無効化しない場合は何も入力せず Enter キーを押してください。

現在の設定値を確認するには abos-ctrl check-secure コマンドを実行します。disabled の場合は無効化する設定になっています。

```
[armadillo /]# abos-ctrl check-secure
- JTAG access disabled for mass production.
- SD boot access disabled for mass production.
```

図 4.11 JTAG と SD ブートの設定値を確認する

設定をリセットするには --reset オプションを付けて abos-ctrl installer-setting コマンドを実行します。

```
[armadillo /]# abos-ctrl installer-setting --reset
cleaned up all settings.
```

図 4.12 JTAG と SD ブートの設定値をリセットする



この手順で作成したインストールディスクの使用は慎重に行ってください。JTAG および SD ブートを無効化したインストールディスクで初期化した Armadillo は、再びこれらを有効に戻すことはできなくなります。そのため不具合発生時にこれらのインターフェースを使用した不具合解析ができなくなる点に留意してください。

4.4.7.2. U-Boot のコマンドプロンプトを無効化する

コマンドラインから生成する場合は、 Armadillo の U-Boot のコマンドプロンプトを無効にするインストールディスクイメージを生成することができます。U-Boot のコマンドプロンプトを無効にするには abos-ctrl installer-setting コマンドを実行します。このコマンドにより、SD ブート時の U-Boot のコマンドプロンプトも同時に無効化されます。



生成したインストールディスクを使用して初期化した Armadillo の U-Boot のコマンドプロンプトを無効にする設定であり、開発用の Armadillo の U-Boot のコマンドプロンプトが無効になることはありません。

```
[armadillo /]# abos-ctrl installer-setting
:(省略)
Would you like to disable boot prompt in the installer ? [y/N] ❶
y
Boot prompt disabled setting for production Armadillo has been configured.
```

図 4.13 U-Boot のコマンドプロンプトを無効化する

❶ U-Boot のコマンドプロンプトを無効化する場合は y を入力します。無効化しない場合は何も入力せず Enter キーを押してください。

現在の設定値を確認するには abos-ctrl check-secure コマンドを実行します。disabled の場合は無効化する設定になっています。

```
[armadillo /]# abos-ctrl check-secure
:(省略)
- boot prompt disabled for mass production.
```

図 4.14 U-Boot のコマンドプロンプトの設定値を確認する

設定をリセットするには「図 4.12. JTAG と SD ブートの設定値をリセットする」と同様に、--reset オプションを付けて abos-ctrl installer-setting コマンドを実行してください。

4.4.7.3. インストールディスクイメージを生成する

abos-ctrl make-installer コマンドを実行して、microSD カードにインストールディスクイメージを生成します。

```
[armadillo /]# abos-ctrl make-installer
Checking if /dev/mmcblk1 can be used safely...
```

```
It looks like your SD card does not contain an installer image
Download baseos-6e-installer-latest.zip image from armadillo.atmark-techno.com (~170M) ? [y/N] ❶
WARNING: it will overwrite your SD card!!
y
Downloading and extracting image to SD card...
Finished writing baseos-6e-installer-[VERSION].img, verifying written content...

Would you like to create a windows partition?
That partition would only be used for customization script at the end of
install, leave at 0 to skip creating it.
Custom partition size (MB, [0] or 16 - 29014): 500 ❷
Checking and growing installer main partition
Trying to install mkfs.exfat (exfatprogs) in memory from internet
fetch https://download.atmark-techno.com/alpine/v3.19/atmark/armv7/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/main/armv7/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/community/armv7/APKINDEX.tar.gz
(1/1) Installing exfatprogs (1.2.2-r0)
Executing busybox-1.36.1-r15.trigger
OK: 148 MiB in 197 packages
exfatprogs version : 1.2.2
Creating exFAT filesystem(/dev/mmcblk1p2, cluster size=131072)

Writing volume boot record: done
Writing backup volume boot record: done
Fat table creation: done
Allocation bitmap creation: done
Upcase table creation: done
Writing root directory entry: done
Synchronizing...

exFAT format complete!
Resize device id 1 (/dev/mmcblk1p1) from 520.00MiB to max
Installer will disable JTAG access
Installer will disable SD boot after installation
Installer will disable uboot prompt
Environment OK, copy 1
Copying boot image
Copying rootfs
Copying appfs
At subvol app/snapshots/volumes
At subvol app/snapshots/boot_volumes
At subvol app/snapshots/boot_containers_storage
Cleaning up and syncing changes to disk...
Installer updated successfully!
```

図 4.15 開発完了後のシステムをインストールディスクイメージにする

❶ y を入力し Enter を押下します。

❷ インストールディスク内にインストールログを保存したい場合など、自由に使用できる第 2 パーティションを指定したサイズで作成します。サイズを入力し Enter を押下します。詳細は「4.4.7.4. インストール時に任意のシェルスクリプトを実行する」を参照してください。

「Installer updated successfully!」と表示されれば、正常に microSD カードにインストールディスクイメージを書き込むことができています。Armadillo から microSD カードを抜去してください。



セキュリティーの観点から、インストールディスクによるインストール実行時に以下のファイルを再生成しております：

- /etc/machine-id (ランダムな個体識別番号)
- /etc/abos_web/tls (ABOS-Web の https 鍵)
- /etc/ssh/ssh_host_*key* (ssh サーバーの鍵。なければ生成しません）ABOS 3.19.1-at.3 以降

他のファイルを個体毎に変更したい場合は 「4.4.7.4. インストール時に任意のシェルスクリプトを実行する」 で対応してください。

4.4.7.4. インストール時に任意のシェルスクリプトを実行する

作成したインストールディスクの所定の場所に、 installer_overrides.sh というファイル名でシェルスクリプトを配置することで、インストール処理の前後で任意の処理を行なうことができます。

installer_overrides.sh に記載された「表 4.2. インストール中に実行される関数」に示す 3 つの名前の関数のみが、それぞれ特定のタイミングで実行されます。

表 4.2 インストール中に実行される関数

関数名	備考
preinstall	インストール中、eMMC のパーティションが分割される前に実行されます。
postinstall	send_log 関数を除く全てのインストール処理の後に実行されます。
send_log	全てのインストール処理が完了した後に実行されます。指定した場所にインストールログを保存できます。

installer_overrides.sh を書くためのサンプルとして、 インストールディスクイメージの第 1 パーティション及び、「4.4.4. 開発したシステムをインストールディスクにする」で作成したのであれば第 2 パーティション直下に installer_overrides.sh.sample を用意してあります。このサンプルをコピーして編集するなどして、行ないたい処理を記述してください。

作成した installer_overrides.sh は、インストールディスクの第 1 パーティション(ラベル名は "rootfs_0")か、「4.4.4. 開発したシステムをインストールディスクにする」で作成したのであれば第 2 パーティション(ラベル名は"INST_DATA")の直下に配置することで実行されます。両方に配置した場合は、第 2 パーティションに配置した記述が適用されます。



インストールディスクの第 1 パーティションは btrfs、第 2 パーティションは exfat でフォーマットされているため、第 2 パーティションのみ Windows PC でもマウントして読み書きすることができます。

製造担当者が installer_overrides.sh を記述する場合に、仮に Windows PC しか作業環境がない場合でも、第 2 パーティションを作成しておくことで作業を行なうことができるというメリットもあります。

これを利用することで、複数台の Armadillo に対してそれぞれに異なる固定 IP アドレスを設定したり、各種クラウドへの接続鍵などを個体ごとに配置したりしたいなど、個体ごとに異なる設定を行なうなど柔軟な製造を行なうことも可能です。以下ではこの機能を利用して、個体ごとに異なる固定 IP アドレスを設定する方法と、インストール実行時のログを保存する方法を紹介します。

これらを必要としない場合は「4.4.8. インストールの実行」に進んでください。

4.4.7.5. 個体ごとに異なる固定 IP アドレスを設定する

インストール時に任意のシェルスクリプトを実行できる機能を利用して、複数の Armadillo に対して異なる固定 IP アドレスを割り当てる例を紹介します。

INST_DATA 内の installer_overrides.sh.sample と ip_config.txt.sample は個体ごとに異なる IP アドレスを割り振る処理を行なうサンプルファイルです。それぞれ installer_overrides.sh と ip_config.txt にリネームすることで、 ip_config.txt に記載されている条件の通りに個体ごとに異なる固定 IP アドレスを設定することができます。全てをここでは説明しませんので、詳細はそれぞれのファイル内の記述も参照してください。

今回はそれぞれのファイルの内容は変更せず使用します。サンプルそのままですが、 ip_config.txt の内容を「図 4.16. ip_config.txt の内容」に示します。

```
# mandatory first IP to allocate, inclusive
START_IP=10.3.4.2 ❶

# mandatory last IP to allocate, inclusive
END_IP=10.3.4.249 ❷

# netmask to use for the IP, default to 24
#NETMASK=24 ❸

# Gateway to configure
# not set if absent
GATEWAY=10.3.4.1 ❹

# DNS servers to configure if present, semi-colon separated list
# not set if absent
DNS="1.1.1.1;8.8.8.8" ❺

# interface to configure, default to eth0
#IFACE=eth0 ❻
```

図 4.16 ip_config.txt の内容

❶ このインストールディスクで割り振る IP アドレスの範囲の始まりを指定します。

❷ このインストールディスクで割り振る IP アドレスの範囲の終わりを指定します。

❸ ネットマスクを指定します。指定しない場合は 24 になります。デフォルトでコメントアウトされています。

❹ ゲートウェイアドレスを指定します。

❺ DNS アドレスを指定します。セミコロンで区切ることでセカンダリアドレスも指定できます。

❻ IP アドレスの設定を行なうインターフェースを指定します。指定しない場合は eth0 になります。デフォルトでコメントアウトされています。



インストール作業の並列化の為に、複数枚のインストールディスクで固定 IP アドレスを割り振る場合は、それぞれのインストールディスクが割り振る IP アドレスの範囲が被らないように ip_config.txt を設定してください。

これらのファイルを配置したインストールディスクでインストールを実行した Armadillo が、正しく設定できていることを確認します。

```
[armadillo /]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:11:22:33:44:55 brd ff:ff:ff:ff:ff:ff
    inet 10.3.4.2/24 brd 10.3.4.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet6 ffff::ffff:ffff:ffff:ffff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

図 4.17 IP アドレスの確認

また、サンプルスクリプトをそのまま使用すると、インストールディスクの第 2 パーティションに allocated_ips.csv というファイルが生成されます。このファイルには、このインストールディスクを使用して IP アドレスの設定を行なった個体のシリアル番号、MAC アドレス、設定した IP アドレスが追記されていきます。

```
SN,MAC,IP
00C700010009,00:11:22:33:44:55,10.3.4.2
```

図 4.18 allocated_ips.csv の内容



2 台目以降の Armadillo にこのインストールディスクで IP アドレスの設定を行なう際に、 allocated_ips.csv を参照して次に割り振る IP アドレスを決めますので、誤って削除しないように注意してください。

4.4.7.6. インストール実行時のログを保存する

installer_overrides.sh 内の send_log 関数は、インストール処理の最後に実行されます。インストールしたルートファイルシステムやファームウェアのチェックサムなどの情報が記録されたログファイルのパスが LOG_FILE に入るため、この関数内でインストールディスクの第 2 パーティションに保存したり、外部のログサーバにアップロードしたりすることが可能です。

「図 4.19. インストールログを保存する」は、インストールディスクの第 2 パーティションにインストールログを保存する場合の send_log 実装例です。

```
send_log() {
        : "This function is called after aggregating logs for archival"
        local LOG_FILE="$1"

        if [ -n "$USER_MOUNT" ]; then
```

```
                        mount /dev/mmcblk1p2 "$USER_MOUNT" ❶
                        cp $LOG_FILE $USER_MOUNT/${SN}_install.log ❷
                        umount "$USER_MOUNT" ❸
        fi
}
```

図 4.19 インストールログを保存する

❶ send_log 関数中では、SD カードの第 2 パーティション(/dev/mmcblk1p2)はマウントされていないのでマウントします。

❷ ログファイルを <シリアル番号>_install.log というファイル名で第 2 パーティションにコピーします。

❸ 第 2 パーティションをアンマウントします。

これらの変更を行なったインストールディスクでインストールを実行した後に、インストールディスクを PC などに接続して正しくログを保存できていることを確認してください。保存したログファイルの中身の例を「図 4.20. インストールログの中身」に示します。

```
RESULT:OK
abos-ctrl make-rootfs on Tue Jun 21 17:57:07 JST 2022 4194304 6b8250df711de66b
abos-ctrl make-rootfs on Tue Jun 21 17:57:24 JST 2022 314572800 58a9b6664158943e
firm 8e9d83d1ba4db65d
appfs 5108 1fa2cbaff09c2dbf
```

図 4.20 インストールログの中身

4.4.8. インストールの実行

前章までの手順で作成したインストールディスクを、開発に使用した Armadillo 以外の Armadillo に対して適用します。

クローン先の Armadillo の eMMC 内のデータは上書きされて消えるため、必要なデータは予めバックアップを取っておいてください。

「3.1.4.2. インストールディスクを使用する」を参照して、クローン先の Armadillo にインストールディスクを適用してください。

「4.4.7.1. JTAG と SD ブートを無効化する」 で JTAG と SD ブートを無効化した場合は、インストールを行った Armadillo の JTAG と SD ブートが無効化されています。

ここまで完了したら、「4.6. イメージ書き込み後の動作確認」に進んでください。

4.5. SWUpdate を用いてイメージ書き込みする

4.5.1. SWU イメージの準備

ここでは、sample-container という名称のコンテナの開発を終了したとします。コンテナアーカイブの作成方法は 「6.9.2.7. コンテナの自動作成やアップデート」 を参照ください。

1. sample-container-v1.0.0.tar (動かしたいアプリケーションを含むコンテナイメージアーカイブ)
2. sample-container.conf (コンテナ自動実行用設定ファイル)

これらのファイルを /home/atmark/mkswu/sample-container ディレクトリを作成して配置した例を記載します。

```
[ATDE ~/mkswu/sample-container]$ ls
sample-container-v1.0.0.tar  sample-container.conf
```

図 4.21 Armadillo に書き込みたいソフトウェアを ATDE に配置

4.5.2. desc ファイルの記述

SWUpdate 実行時に、「4.5.1. SWU イメージの準備」で挙げたファイルを Armadillo に書き込むような SWU イメージを生成します。

SWU イメージを作成するためには、SWUpdate 時に実行する処理等を示した desc ファイルを記述し、その desc ファイルを入力として mkswu コマンドを実行することで、SWU イメージが出来上がります。

```
[ATDE ~/mkswu/sample-container]$ cat sample-container.desc
swdesc_option component=sample-container
swdesc_option version=1
swdesc_option POST_ACTION=poweroff ❶

swdesc_embed_container "sample-container-v1.0.0.tar" ❷
swdesc_files --extra-os --dest /etc/atmark/containers/ "sample-container.conf" ❸
```

図 4.22 desc ファイルの記述例

❶ SWUpdate 完了後に電源を切るように設定します。

❷ コンテナイメージファイルを SWU イメージに組み込み、Armadillo に転送します。

❸ コンテナ起動設定ファイルを Armadillo に転送します。

ここまで完了したら、「4.6. イメージ書き込み後の動作確認」に進んでください。desc ファイルの詳細な書式については、「6.4. mkswu の .desc ファイルを編集する」を参照してください。また、作成された SWU イメージの内容を確認したい場合は、「6.6. SWU イメージの内容の確認」を参照してください。

4.6. イメージ書き込み後の動作確認

「4.4. インストールディスクを用いてイメージ書き込みする」で作成したインストールディスクを使用してインストール、または「4.5. SWUpdate を用いてイメージ書き込みする」にて SWUpdate によってイメージ書き込みを行った後には、イメージが書き込まれた Armadillo が正しく動作するか、実際に動かして確認してみます。

再度電源を投入して、期待したアプリケーションが動作することを確認してください。

ここまで完了したならば、量産時のイメージ書き込みは完了です。

5. 運用編

5.1. Armadillo Twin に Armadillo を登録する

5.1.1. Armadillo の設置前に登録する場合

Armadillo を Armadillo Twin に登録する場合、ケース裏や基板本体に貼付されているシール上の QR コードを使用します。登録方法についての詳細は Armadillo Twin ユーザーマニュアル 「Armadillo Twin にデバイスを登録する」 [https://manual.armadillo-twin.com/register-device/] をご確認ください。

5.1.2. Armadillo の設置後に登録する場合

Armadillo 設置後の登録については、弊社営業までお問い合わせください。

5.2. Armadillo を設置する

Armadillo を組み込んだ製品を設置する際の注意点や参考情報を紹介します。

5.2.1. 設置場所

開発時と同様に、水・湿気・ほこり・油煙等の多い場所に設置しないでください。火災、故障、感電などの原因になる場合があります。

本製品に搭載されている部品の一部は、発熱により高温になる場合があります。周囲温度や取扱いによってはやけどの原因となる恐れがあります。本体の電源が入っている間、または電源切断後本体の温度が下がるまでの間は、基板上の電子部品、及びその周辺部分には触れないでください。

無線 LAN 機能を搭載した製品は、心臓ペースメーカーや補聴器などの医療機器、火災報知器や自動ドアなどの自動制御器、電子レンジ、高度な電子機器やテレビ・ラジオに近接する場所、移動体識別用の構内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

5.2.2. ケーブルの取り回し

一般的に以下の点を注意して設置してください。また、「3.4. ハードウェアの設計」や「3.7. インターフェースの使用方法とデバイスの接続方法」の各章ハードウェア仕様に記載していることにも従ってください。

- 設置時にケーブルを強く引っ張らないでください。
- ケーブルはゆるやかに曲げてください。 ケーブルを結線する場合、きつくせず緩く束ねてください。

5.2.3. WLAN+BT コンボモジュール用アンテナの指向性

WLAN+BT コンボモジュール用アンテナはケースに貼り付けられており、「図 5.1. Armadillo-IoT ゲートウェイ A6E WLAN+BT アンテナの指向性」に示す指向性があります。

Armadillo-IoT ゲートウェイ A6E 内部に搭載されているアンテナの指向性イメージです。実際のアンテナの特性を正確に表しているものではありません。

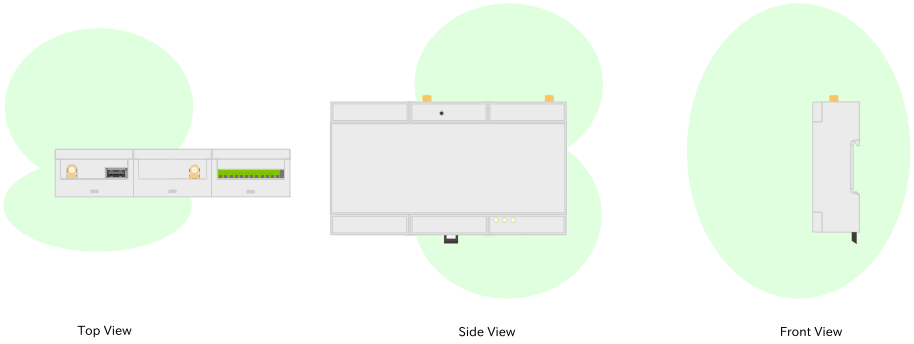


図 5.1 Armadillo-IoT ゲートウェイ A6E WLAN+BT アンテナの指向性

5.2.4. LTE 外付け用アンテナの指向性

LTE アンテナは「図 5.2. LTE 外付け用アンテナの指向性」に示す指向性があります。

一般的な λ/2 ダイポールアンテナ、λ/4 モノポールアンテナの指向性イメージです。実際のアンテナの特性を正確に表しているものではありません

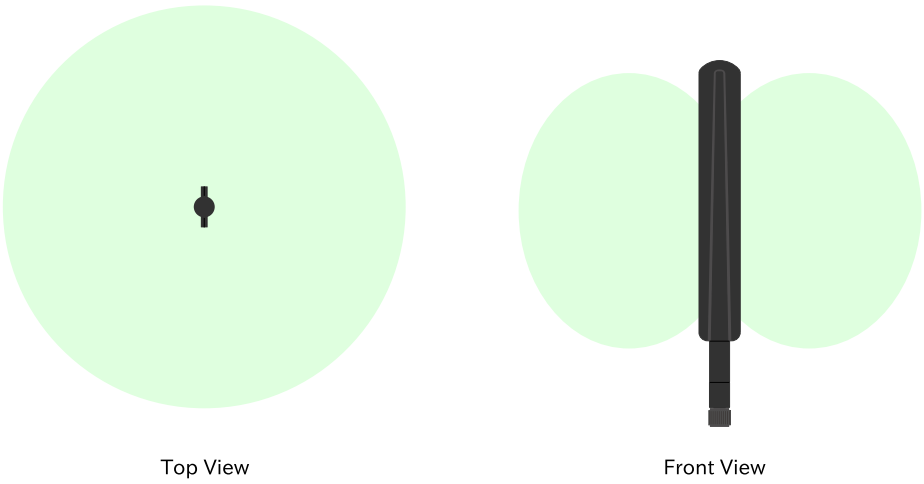


図 5.2 LTE 外付け用アンテナの指向性

5.2.5. LTE の電波品質に影響する事項

一般的には、周辺に障害物、特に鉄板や鉄筋コンクリート、電波シールドフィルムの貼られたガラスが存在する場合電波を大きく妨げます。また、周辺機器の電波出力、人通り(周辺のスマートフォン=機器が増える)、基地局との距離・方向など多くの要素によって変化します。

WWAN LED にて電波状況をチェックできますので、設置時に電波状況を確認の上運用ください。WWAN LED の状態と意味は「表 3.46. LED の状態表示」を参照ください。

5.2.6. サージ対策

サージ対策については、「3.4.3. ESD/雷サージ」を参照してください。

5.2.7. Armadillo の状態を表すインジケータ

LED にて状態を表示しています。

有線 LAN の状態は「表 3.22. CON4 LAN LED の動作」を、Armadillo の状態を表示する LED に関しては「表 3.46. LED の状態表示」を参照ください。

5.2.8. 個体識別情報の取得

設置時に Armadillo を個体ごとに識別したい場合、以下の情報を個体識別情報として利用できます。

- 個体番号
- MAC アドレス



Armadillo の設置前に個体識別情報を記録しておき、設置後の Armadillo を識別できるようにしておくことを推奨します。

これらの情報を取得する方法は以下のとおりです。状況に合わせて手段を選択してください。

- 本体シールから取得する
- コマンドによって取得する

5.2.8.1. 本体シールから取得

Armadillo の個体番号、各種 MAC アドレスなどの個体識別情報は、ケース裏や基板本体に貼付されているシールに記載されています。製品モデル毎に記載されている内容やシールの位置が異なるので、詳細は各種納入仕様書を参照してください。

5.2.8.2. コマンドによる取得

シールだけでなくコマンドを実行することによっても個体識別情報を取得することができます。以下に個体番号と MAC アドレスを取得する方法を説明します。

個体番号を取得する場合、「図 5.3. 個体番号の取得方法（device-info）」に示すコマンドを実行してください。device-info はバージョン v3.18.4-at.7 以降の ABOS に標準で組み込まれています。

```
[armadillo ~]# device-info -s
00C900010001 ❶
```

図 5.3 個体番号の取得方法（device-info）

❶ 使用している Armadillo の個体番号が表示されます。

device-info がインストールされていない場合は ABOS 3.18-at.7 以降にアップデートしてください。非推奨ですが、ABOS をアップデートせずに個体番号を取得したい場合は「図 5.4. 個体番号の取得方法（get-board-info）」に示すように get-board-info を実行することでも取得できます。

```
[armadillo ~]# apk add get-board-info
[armadillo ~]# get-board-info -s
00C900010001 ❶
```

図 5.4 個体番号の取得方法（get-board-info）

❶ 使用している Armadillo の個体番号が表示されます。



コンテナ上で個体番号を表示する場合は、個体番号を環境変数として設定することで可能となります。「図 5.5. 個体番号の環境変数を conf ファイルに追記」に示す内容を/etc/atmark/containers の下の conf ファイルに記入します。

```
add_args --env=SERIALNUM=$(device-info -s) ❶
```

図 5.5 個体番号の環境変数を conf ファイルに追記

❶ コンテナ起動毎に環境変数 SERIALNUM に値がセットされます。

「図 5.6. コンテナ上で個体番号を確認する方法」に示すコマンドを実行することでコンテナ上で個体番号を確認することができます。

```
[container ~]# echo $SERIALNUM
00C900010001
```

図 5.6 コンテナ上で個体番号を確認する方法

次に MAC アドレスを取得する方法を説明します。「図 5.7. MAC アドレスの確認方法」に示すコマンドを実行することで、各インターフェースの MAC アドレスを取得できます。

```
[armadillo ~]# ip addr
: (省略)
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
```

```
    link/ether 00:11:0c:12:34:56 brd ff:ff:ff:ff:ff:ff ❶
: (省略)
```

図 5.7 MAC アドレスの確認方法

❶ link/ether に続くアドレスが MAC アドレスです。

また、出荷時にアットマークテクノが書き込んだ Ethernet MAC アドレスは「図 5.8. 出荷時の Ethernet MAC アドレスの確認方法」に示すコマンドを実行することで取得することができます。

```
[armadillo ~]# device-info -m
eth0: 00:11:0C:12:34:56 ❶
```

図 5.8 出荷時の Ethernet MAC アドレスの確認方法

❶ 出荷時にアットマークテクノが書き込んだ Ethernet MAC アドレスが表示されます。

ただし、「図 5.8. 出荷時の Ethernet MAC アドレスの確認方法」で示すコマンドでは、お客様自身で設定した Ethernet MAC アドレスを取得することはできないのでご注意ください。お客様自身で設定した Ethernet MAC アドレスを取得したい場合は「図 5.7. MAC アドレスの確認方法」に示すコマンドを実行してください。

5.2.9. 電源を切る

Armadillo の電源を切る場合は、 poweroff コマンドを実行してから電源を切るのが理想的です。しかし、設置後はコマンドを実行できる環境にない場合が多いです。この場合、条件が整えば poweroff コマンドを実行せずに電源を切断しても安全に終了できる場合があります。

詳細は、「3.1.7.5. Armadillo の終了方法」を参照してください。

5.3. ABOSDE で開発したアプリケーションをアップデートする

ABOSDE で開発したアプリケーションのアップデートは、開発時と同様に ABOSDE を用いて行うことが出来ます。

「3.15. ABOSDE によるアプリケーションの開発」で示したように、開発時にはリリース版のアプリケーションを Armadillo にインストールするために、 VS Code の左ペインの [Generate release swu] を実行して release.swu を作成しました。

アップデート時にも、アップデートに必要なアプリケーションの編集をした後に [Generate release swu] を実行して、アップデート版のアプリケーションを含む release.swu を作成します。

具体的な ABOSDE を用いたアプリケーションのアップデートの流れは「5.3.1. アプリケーションのアップデート手順」に示します。

5.3.1. アプリケーションのアップデート手順

ここでは、プロジェクト名を my_project としています。

5.3.1.1. アップデートするアプリケーションのプロジェクトを VS Code で開く

「図 5.9. VS Code を起動」で示すように、アップデートするアプリケーションのプロジェクトを指定して VS Code を起動してください。

```
[ATDE ~]$ code my_project
```

図 5.9 VS Code を起動

5.3.1.2. アップデート前のバージョンのプロジェクトを管理する

ABOSDE では、プロジェクトのバージョン管理は行っていません。必要な場合はユーザー自身でアップデート前のプロジェクトを管理してください。



アップデート前のプロジェクトの release.swu のバージョンを知りたい場合は「6.6. SWU イメージの内容の確認」を参照してください。

5.3.1.3. アプリケーションのソースコードを編集しテストする

既存のアプリケーションのソースコードを編集した後、「3.15. ABOSDE によるアプリケーションの開発」を参考に、アプリケーションが Armadillo 上で問題なく動作するかテストを行ってください。

5.3.1.4. アップデート用の swu を作成する

VS Code の左ペインの [Generate release swu] を実行してください。my_project ディレクトリ下に release.swu というファイル名で SWU ファイルが作成されます。

5.3.1.5. 運用中の Armadillo のアプリケーションをアップデートする

アプリケーションをアップデートするために、作成した release.swu を運用中の Armadillo にインストールしてください。SWU イメージファイルをインストールする方法は「3.3.4.5. SWU イメージのインストール」を参照してください。

5.4. Armadillo のソフトウェアをアップデートする

設置後の Armadillo のソフトウェアアップデートは SWUpdate を使用することで実現できます。

ここでは、ソフトウェアのアップデートとして以下のような処理を行うことを例として説明します。

- すでに Armadillo に sample_container_image というコンテナイメージがインストールされている
- sample_container_image のバージョンを 1.0.0 から 1.0.1 にアップデートする
- sample_container_image からコンテナを自動起動するための設定ファイル(sample_container.conf)もアップデートする

5.4.1. SWU イメージの作成

アップデートのために SWU イメージを作成します。SWU イメージの作成には、 mkswu というツールを使います。「3.1. 開発の準備」で作成した環境で作業してください。

5.4.1.1. SBOM の生成

SWU イメージ作成時に、同時に SBOM を生成することができます。詳細は 「6.37.2. SBOM の提供」を参照してください。

5.4.2. mkswu の desc ファイルを作成する

SWU イメージを生成するには、 desc ファイルを作成する必要があります。

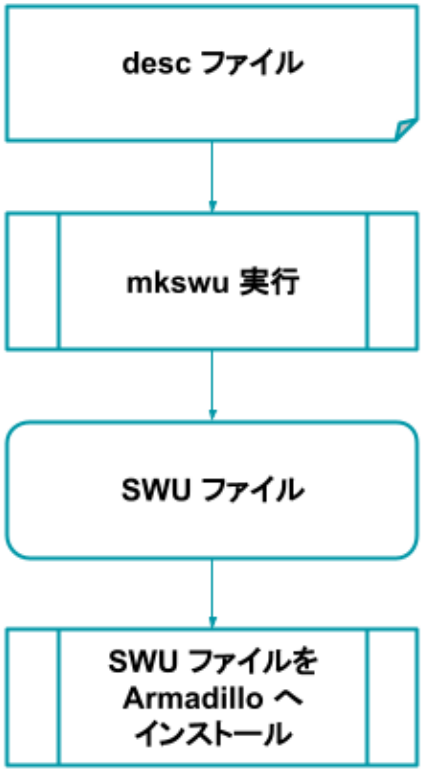


図 5.10 desc ファイルから Armadillo へ SWU イメージをインストールする流れ

desc ファイルとは、 SWU イメージを Armadillo にインストールする際に行われる命令を記述したものです。/usr/share/mkswu/examples/ ディレクトリ以下にサンプルを用意していますので、やりたいことに合わせて編集してお使いください。なお、 desc ファイルの詳細な書式については、「6.4. mkswu の .desc ファイルを編集する」を参照してください。

まず、以下のようなディレクトリ構成で、 sample_container.conf を作成しておきます。設定ファイルの内容については割愛します。

```
[ATDE ~/mkswu]$ tree container_start
container_start
└── etc
    └── atmark
```

```
        └──── containers
            └──── sample_container.conf
```

このような階層構造にしているのは、インストール先の Armadillo 上で sample_container.conf を /etc/atmark/containers/ の下に配置したいためです。

次に、アップデート先のコンテナイメージファイルである sample_container_image.tar を用意します。コンテナイメージを tar ファイルとして出力する方法を「図 5.11. コンテナイメージアーカイブ作成例」に示します。

```
[armadillo ~]# podman save sample_container:[VERSION] -o sample_container_image.tar
```

図 5.11 コンテナイメージアーカイブ作成例

次に、sample_container_update.desc という名前で desc ファイルを作成します。「図 5.12. sample_container_update.desc の内容」に、今回の例で使用する sample_container_update.desc ファイルの内容を示します。sample_container_image.tar と、コンテナ起動設定ファイルを Armadillo にインストールする処理が記述されています。

```
[ATDE ~/mkswu]$ cat sample_container_update.desc
swdesc_option version=1.0.1

swdesc_usb_container "sample_container_image.tar" ❶
swdesc_files --extra-os "container_start" ❷
```

図 5.12 sample_container_update.desc の内容

❶ sample_container_image.tar ファイルに保存されたコンテナをインストールします。

❷ container_start ディレクトリの中身を転送します。

コマンドは書かれた順番でインストールされます。

5.4.3. desc ファイルから SWU イメージを生成する

mkswu コマンドを実行することで、 desc ファイルから SWU イメージを生成できます。

```
[ATDE ~/mkswu]$ mkswu -o sample_container_update.swu sample_container_update.desc ❶
[ATDE ~/mkswu]$ ls sample_container_update.swu ❷
sample_container_update.swu
```

図 5.13 sample_container_update.desc の内容

❶ mkswu コマンドで desc ファイルから SWU イメージを生成

❷ sample_container_update.swu が生成されていることを確認

作成された SWU イメージの内容を確認したい場合は、「6.6. SWU イメージの内容の確認」を参照してください。

5.4.4. イメージのインストール

インストールの手順については、「3.3.4.5. SWU イメージのインストール」を参照してください。

5.5. Armadillo Twin から複数の Armadillo をアップデートする

Armadillo Twin を使用することで、自身でサーバー構築を行うことなくネットワーク経由で SWU イメージを配信し、デバイスのソフトウェアを更新することができます。Armadillo Twin を使用したソフトウェアアップデートを行うためには、Armadillo Twin へのデバイスの登録が完了している必要があります。Armadillo Twin へのデバイスの登録方法については、「5.1. Armadillo Twin に Armadillo を登録する」 をご確認ください。また、Armadillo Twin を使用したソフトウェアアップデートの実施方法については、 Armadillo Twin ユーザーマニュアル 「デバイスのソフトウェアをアップデートする」 [https://manual.armadillo-twin.com/update-software/] をご確認ください。

5.6. Armadillo Twin を利用してソフトウェアの脆弱性チェックを行う

ソフトウェア開発時には発見されなかった脆弱性が運用時に発見されることがあるため、運用時でもソフトウェアの脆弱性を定期的に確認することが重要です。Armadillo Twin の脆弱性チェック機能では、登録した SBOM を定期的にスキャンし、新たに脆弱性が発見された際に管理画面に表示します。機能の詳細は Armadillo Twin ユーザーマニュアル 「SWU イメージを管理する」 [https://manual.armadillo-twin.com/management-swu-image/] をご確認ください。

5.7. eMMC の寿命を確認する

5.7.1. eMMC について

eMMC とは embedded Multi Media Card の頭文字を取った略称で NAND 型のフラッシュメモリを利用した内蔵ストレージです。当社で使用しているものは長期間運用を前提としている為、使用する容量を半分以下にして SLC モードで使用しています。(例えば 32GB 製品を 10GB で使用、残り 22GB は予備領域とする)。

eMMC は耐性に問題が発生した個所を内部コントローラがマスクし、予備領域を割り当てて調整しています。絶対ではありませんが、この予備領域がなくなると書き込みが出来なくなる可能性があります。

5.7.2. eMMC 予備領域の確認方法

Armadillo Base OS には emmc-utils というパッケージがインストールされています。

「図 5.14. eMMC の予備領域使用率を確認する」に示すコマンドを実行し、 EXT_CSD_PRE_EOL_INFO の内容を確認することで eMMC の予備領域の使用率がわかります。 EXT_CSD_PRE_EOL_INFO の値と意味の対応を「表 5.1. EXT_CSD_PRE_EOL_INFO の値の意味」に示します。

```
[armadillo ~]# mmc extcsd read /dev/mmcblk0 | grep EXT_CSD_PRE_EOL_INFO
eMMC Pre EOL information [EXT_CSD_PRE_EOL_INFO]: 0x01
```

図 5.14 eMMC の予備領域使用率を確認する

表 5.1 EXT_CSD_PRE_EOL_INFO の値の意味

値	意味
0x01	定常状態(問題無し)
0x02	予備領域 80% 以上使用
0x03	予備領域 90% 以上使用

また、Armadillo Twin からも eMMC の予備領域使用率を確認することができます。詳細は Armadillo Twin ユーザーマニュアル 「デバイス監視アラートを管理する」 [https://manual.armadillo-twin.com/management-device-monitoring-alert/] をご確認ください。

5.8. 本製品の部品変更情報を知る

本製品に搭載されている部品が変更された場合や、本製品が EOL となった場合には以下のページから確認できます。

Armadillo サイト - 変更通知(PCN)/EOL 通知

https://armadillo.atmark-techno.com/change_notification

また、Armadillo サイトにユーザー登録していただくと、お知らせをメールで受信することが可能です。変更通知についても、メールで受け取ることが可能ですので、ユーザー登録をお願いいたします。

ユーザー登録については「3.1.8. ユーザー登録」を参照してください。

5.9. Armadillo を廃棄する

運用を終了し Armadillo を廃棄する際、セキュリティーの観点から以下のようなことを実施する必要があります。

- 設置場所に Armadillo を放置せず回収する
- Armadillo をネットワークから遮断する
  - SIM カードが挿入されているのであれば抜き、プロバイダーとの契約を終了する
  - 無線 LAN の設定を削除する
  - 接続しているクラウドのデバイス証明書を削除・無効にすることでクラウドに接続できなくする
- 「3.1.4. Armadillo の初期化と ABOS のアップデート」の手順にしたがって初期化を行う
  - インストールディスクは、 blkdiscard コマンドを用いて eMMC の GPP を含む全てのパーティション内のデータを消去しています
- 物理的に起動できなくする
- Armadillo Twin をご利用の場合は、Armadillo Twin 上で当該デバイスの廃棄手続きを行うか、Armadillo Twin を解約する際に廃棄する旨を申告する
  - 実際の手続きなど詳細は Armadillo Twin お問い合わせフォーム [https://apps.armadillo-twin.com/ja/inquiry]からお問い合わせください
- Armadillo を物理的に破壊する

- SD ブートを無効化していてインストールディスクが適用できない場合や、情報吸い出しのリスクをより下げたい場合は、 eMMC やセキュアエレメント等の記憶媒体を物理的に破壊してください
- 詳細な破壊箇所や破壊方法については、弊社までお問い合わせください

6. 応用編

本章では、ここまでの内容で紹介しきれなかった、より細かな Armadillo の設定方法や、開発に役立つヒントなどを紹介します。

各トピックを羅列していますので、目次の節タイトルからやりたいことを探して辞書的にご使用ください。

6.1. 省電力・間欠動作機能

本章では、Armadillo-IoT ゲートウェイ A6E の 省電力・間欠動作機能や動作モード、状態遷移について説明します。

6.1.1. シャットダウンモードへの遷移と起床

シャットダウンモードへ遷移するには、poweroff コマンド、または aiot-alarm-poweroff コマンドを実行します。

6.1.1.1. poweroff コマンド

poweroff コマンドを実行してシャットダウンモードに遷移した場合、電源の切断・接続のみでアクティブモードに遷移が可能です。poweroff コマンドの実行例を次に示します。

```
[armadillo ~]# poweroff
[  OK  ] Stopped target Timers.
[  OK  ] Stopped Daily man-db regeneration.
[  OK  ] Stopped Daily rotation of log files.

※省略

[39578.876586] usb usb1: USB disconnect, device number 1
[39578.882754] ci_hdrc ci_hdrc.0: USB bus 1 deregistered
[39578.888133] reboot: Power down
```

6.1.1.2. aiot-alarm-poweroff コマンド

aiot-alarm-poweroff コマンドを実行することで、シャットダウンモードに遷移後、RTC のアラーム割り込みをトリガで起床（アクティブモードに遷移）することができます。なお、RTC を起床要因に使って間欠動作させる場合は、 「3.7.15. リアルタイムクロックを使用する」 を参考に、必ず RTC の日時設定を行ってください。aiot-alarm-poweroff コマンドによる間欠動作を行っている時に、シャットダウンモードへの遷移を一時的に禁止したい場合は、 「6.1.6. スリープ動作の禁止と禁止解除」 で説明しているコマンドをご利用ください。



RTC 未設定によるエラーが発生した場合、シャットダウンモードへの遷移は行われません。

```
[armadillo ~]# aiot-alarm-poweroff +[現在時刻からの経過秒数]
```

図 6.1 aiot-alarm-poweroff コマンド書式

シャットダウンモードに遷移し、300 秒後にアラーム割り込みを発生させるには、次のようにコマンドを実行します。

```
[armadillo ~]# aiot-alarm-poweroff +300
aiot-alarm-poweroff: alarm_timer +300 second
```

現在時刻からの経過秒数は 180 秒以上を指定する必要があります。

6.1.2. スリープモードへの遷移と起床

aiot-sleep コマンドを実行することで、スリープモードに遷移することができます。スリープモードからの起床（アクティブモードに遷移する）条件は、aiot-sleep コマンドを実行する前に aiot-set-wake-trigger コマンドで事前指定します。ユーザースイッチによる起床は標準で有効になっています。また、起床条件は OR 条件での設定が可能です。

6.1.2.1. RTC アラーム割り込み以外での起床

aiot-set-wake-trigger コマンドの書式と設定可能なパラメータを以下に示します。

ain を指定する際は、別途閾値の設定が必要となります。設定方法は「6.1.2.3. アナログ入力電圧閾値設定」を参照ください。

```
[armadillo ~]# aiot-set-wake-trigger [TRIGGER] [enabled|disabled]
```

図 6.2 aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込み以外での起床のとき）

表 6.1 aiot-set-wake-trigger TRIGGER 一覧

TRIGGER	説明
usb	CON9(USB ホストインターフェース)に USB デバイスを挿抜したとき
uart3	CON3(シリアルインターフェース /dev/ttymxc2)にデータ受信があったとき
rs485	UART5(シリアルインターフェース /dev/ttymxc4)にデータ受信があったとき
gpio	GPIO 割り込みが発生したとき
sw1	SW1 が押下されたとき
ain	アナログ入力電圧閾値割り込み発生時

コンソール(/dev/ttymxc2)から入力があった場合にスリープモードから起床するには、次に示すコマンドを実行します。

```
[armadillo ~]# aiot-set-wake-trigger uart3 enabled
aiot-set-wake-trigger: uart3 enabled

[armadillo ~]# aiot-sleep
aiot-sleep: Power Management suspend-to-ram
[ 1767.050404] PM: suspend entry (deep)
[ 1767.054019] PM: Syncing filesystems ...
[ 1767.236546] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full -
```

```
flow control rx/tx
[ 1767.428714] done.
[ 1767.431262] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 1767.439582] OOM killer disabled.
[ 1767.442834] Freezing remaining freezable tasks ... (elapsed 0.001
seconds) done.
[ 1767.451485] Suspending console(s) (use no_console_suspend to debug)

※ コンソールに入力

[ 1767.567686] OOM killer enabled.
[ 1767.570875] Restarting tasks ... done.
[ 1767.606048] PM: suspend exit
aiot-sleep: change mode CPU Idle
```

6.1.2.2. RTC アラーム割り込みでの起床

RTC アラーム割り込みでの起床を行う場合、パラメーター設定が異なります。なお、RTC を起床要因に使って間欠動作させる場合は、 「3.7.15. リアルタイムクロックを使用する」 を参考に、必ず RTC の日時設定を行ってください。

RTC アラーム割り込みでの起床は、毎分 00 秒で起床する分指定 (Armadillo-IoT ゲートウェイ A6E 搭載の RTC アラーム割り込みを用いた起床) と秒指定 (SoC 内蔵の RTC アラーム割り込みを用いた起床) の 2 種類があります。

分指定のコマンド書式を 「図 6.3. aiot-set-wake-trigger コマンド書式 (RTC アラーム割り込みでの起床の場合: 分指定)」 に示します。

```
[armadillo ~]# aiot-set-wake-trigger rtc [enabled|disabled] <現在時刻からの経過秒数>
```

図 6.3 aiot-set-wake-trigger コマンド書式 (RTC アラーム割り込みでの起床の場合: 分指定)

現在時刻からの経過秒数 は 60 秒以上を指定する必要があります。

300 秒後に RTC アラーム割り込みを発生させ、スリープモードから起床させるコマンド実行例を以下に示します。

```
[armadillo ~]# aiot-set-wake-trigger rtc enabled +300
aiot-set-wake-trigger: rtc enabled
aiot-set-wake-trigger: alarm_timer +300 second

[armadillo ~]# aiot-sleep
aiot-sleep: Power Management suspend-to-ram
[ 1767.050404] PM: suspend entry (deep)
[ 1767.054019] PM: Syncing filesystems ...
[ 1767.236546] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full -
flow control rx/tx
[ 1767.428714] done.
[ 1767.431262] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 1767.439582] OOM killer disabled.
[ 1767.442834] Freezing remaining freezable tasks ... (elapsed 0.001
seconds) done.
[ 1767.451485] Suspending console(s) (use no_console_suspend to debug)
```

```
※ 約 300 秒待つ

[ 1767.567686] OOM killer enabled.
[ 1767.570875] Restarting tasks ... done.
[ 1767.606048] PM: suspend exit
aiot-sleep: change mode CPU Idle
```

秒指定のコマンド書式を 「図 6.4. aiot-set-wake-trigger コマンド書式 (RTC アラーム割り込みでの起床の場合: 秒指定)」 に示します。

```
[armadillo ~]# aiot-set-wake-trigger rtc_sec [enabled|disabled] <現在時刻からの経過秒数>
```

図 6.4 aiot-set-wake-trigger コマンド書式 (RTC アラーム割り込みでの起床の場合: 秒指定)

現在時刻からの経過秒数 は 5 秒以上を指定する必要があります。

45 秒後に RTC アラーム割り込みを発生させ、スリープモードから起床させるコマンド実行例を以下に示します。

```
[14:46:35.650] armadillo:~# aiot-set-wake-trigger rtc_sec enabled +45
[14:46:38.671] aiot-set-wake-trigger: rtc_sec alarm enabled
[14:46:38.673] aiot-set-wake-trigger: alarm_timer_snvs +45 second
[14:46:38.673] armadillo:~# aiot-sleep
[14:46:43.023] aiot-sleep: Power Management suspend-to-ram

※ 約 45 秒待つ

[14:47:23.128] aiot-sleep: change mode CPU Idle
[14:47:23.128] armadillo:~#
```

6.1.2.3. アナログ入力電圧閾値設定

指定ポートのアナログ入力電圧が設定値以下・以上になった際に、スリープ状態から起床するための設定手順を記載します。

1. 設定ファイル /etc/atmark/ain-set-wake-triggers.conf の作成

   /etc/atmark/ain-set-wake-triggers.conf.sample を参考に /etc/atmark/ain-set-wake-triggers.conf を作成します。記載例を「図 6.5. /etc/atmark/ain-set-wake-triggers.conf の記載例」に示します。

   複数ポートを指定する場合は改行を追加してください。

   使用するオプションは以下のとおりです。

   - -e: 有効指定
   - -p: ポート番号 (1〜4)
   - -u: 指定以下の電圧 (mV) 時に割り込み発生
   - -o: 指定以上の電圧 (mV) 時に割り込み発生

```
AIN_WAKE_TRIGGERS="-e -p 1 -o 5000
                   -e -p 2 -u 1000 -o 2000"
```

図 6.5 /etc/atmark/ain-set-wake-triggers.conf の記載例

2. 「6.1.2.1. RTC アラーム割り込み以外での起床」 の手順で起床要因 ain を有効にします。

本設定を解除したい場合、 /etc/atmark/ain-set-wake-triggers.conf を削除し、「6.1.2.1. RTC アラーム割り込み以外での起床」 の手順で起床要因 ain を無効にしてください。

6.1.2.4. アナログ入力電圧閾値割り込みの無効化

ブートローダーのバージョン at21 以降でカスケード接続した際にカスケード非対応版の拡張基板が混在している場合、アナログ入力電圧閾値割り込みが無効化されます。カスケード非対応版の拡張基板の確認方法については、「6.38.3. +Di8+Ai4 拡張基板のカスケード接続」を参照ください。

アナログ入力電圧閾値割り込みが無効化されている場合、Armadillo 起動時に以下のようなログが表示されます。

```
Disabled additional board interrupts.
```

アナログ入力電圧閾値割り込みが無効化されており /etc/atmark/ain-set-wake-triggers.conf が存在する状態で aiot-sleep を実行すると、以下のエラーが発生しスリープモードに遷移せず動作し続けます。

```
error: wakeup by ain is disabled.
```

これらのログは、アプリケーション上で間欠動作に移行可能かの判断基準として使用できます。

アナログ入力電圧閾値割り込みが無効化されている状態で他の起床要因でスリープモードに遷移したい場合は、 「6.1.2.3. アナログ入力電圧閾値設定」 を無効化してください。

6.1.2.5. 起床要因のクリア

すべての起床要因をクリアするには次に示すコマンドを実行します。ユーザースイッチによる起床設定は無効化できません。

```
[armadillo ~]# aiot-set-wake-trigger all disabled
aiot-set-wake-trigger: clear_all disabled
```

aiot-set-wake-trigger コマンドで起床条件に RTC アラームを設定して、aiot-sleep コマンドと組み合わせれば、一定時間おきに動作する間欠動作を実現できます。aiot-sleep コマンドによるスリープモードへの遷移と起床で間欠動作を行っている時に、スリープモードへの遷移を一時的に禁止したい場合は、「6.1.6. スリープ動作の禁止と禁止解除」 で説明しているコマンドをご利用ください。

6.1.3. スリープ(SMS 起床可能)モードへの遷移と起床

aiot-sleep-sms コマンドを実行することで、スリープ(SMS 起床可能)モードに遷移することができます。スリープモードからの起床（アクティブモードに遷移する）条件は、aiot-sleep-sms コマンドを実

行する前に aiot-set-wake-trigger コマンドで事前指定します。ユーザースイッチによる起床は標準で有効になっています。aiot-sleep-sms コマンドを実行した場合 SMS 受信による起床は強制的に有効になります。また、起床条件は OR 条件での設定が可能です。

aiot-sleep-sms コマンドの実行例を次に示します。

```
[armadillo ~]# aiot-sleep-sms
aiot-sleep-sms: Power Management suspend-to-ram
AT+CMGF=1
OK

AT^SIND=”message”,0
^SIND: message,0,0

OK

AT+CMGD=1,4
OK

AT+CMGL=”ALL”
OK

[ 3508.609638] PM: suspend entry (deep)

^SIND: message,1,0

OK

[ 3508.613982] PM: Syncing filesystems ... done.
[ 3508.637946] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 3508.646276] OOM killer disabled.
[ 3508.649527] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[ 3508.658161] Suspending console(s) (use no_console_suspend to debug)

※ SMS 受信

[ 1767.567686] OOM killer enabled.
[ 1767.570875] Restarting tasks ... done.
[ 1767.606048] PM: suspend exit
aiot-sleep: change mode CPU Idle
```



ご利用の SMS 送信サービスの SMS 送信制限により SMS の送信ができないことがあります。また、ネットワーク状態によって SMS の受信を検知できなかったり、検知が遅れることがあります。

起床要因として SMS のみを設定されるシステムを想定されている場合は、上記検知できない可能性を考慮して RTC など別な起床要因で周期的に起床することを推奨します。

また「6.16.5.2. 省電力などの設定」の初期値では、SMS 受信を検知して起床するまでに最長で 3 分かかります。より短時間で起床する必要がある場合は psm と edrx を disable に設定する対応をご検討ください。



aiot-sleep-sms でスリープモードへ遷移する際、LTE モジュールの SMS 保存用ストレージに空きがない場合 SMS 受信での起床ができなくなるため、LTE モジュールのストレージから 1 件 SMS を削除してからスリープモードへ遷移します。

SMS で受信した内容が必要な場合は、 SMS の内容を別なファイルなどに保存してから aiot-sleep-sms を実施してください。



aiot-sleep-sms でスリープモードへ遷移する際、 SMS 受信以外にも LTE 網でのデータ受信をトリガーとして起床することがあります。グローバル IP を使用している場合に多いのですが、プライベート IP を使用している場合でも網の状況によっては起床することがあります。

その際は、 「6.12.6.9. Rest API : WWAN の設定」 や 「6.20. SMS を利用する」 の手順で SMS を受信しての起床かを確認し、SMS を受信しての起床では無い場合は再度 aiot-sleep-sms を実施してください。

6.1.4. 状態遷移トリガにコンテナ終了通知を利用する

動作中のコンテナの終了をトリガに、省電力状態のモードへの遷移を行うことができます。



コンテナの終了契機は「/etc/atmark/containers/*.conf ファイルの set_command で指定したコンテナ起動時に実行するコマンド」のプロセスが終了した時」となります。ファイルの詳細は 「6.9.2.1. イメージからコンテナを作成する」 を参照してください。

遷移先の動作モードと起床条件は設定ファイルで指定し、コンテナが終了すると指定した動作モードへ遷移、指定した起床条件が発生すると省電力モードから復帰します。また、その際自動的にコンテナも開始します。

コンテナ終了時に遷移する動作モードと起床条件については、設定ファイル(/etc/atmark/power-utils.conf)で指定します。設定ファイルは下記の通り、 TARGET , MODE , WAKEUP を指定します。

```
[armadillo ~]# cat /etc/atmark/power-utils.conf
TARGET='a6e-gw-container'
MODE='NONE'
WAKEUP='SW1', 'USB', 'UART', 'GPIO', 'SMS', 'RTC:60', 'AIN'
```

設定ファイルの概要を以下に示します。

表 6.2 設定パラメーター

パラメーター名	意味
TARGET	状態遷移トリガの対象となるコンテナ名
MODE	遷移先の動作モード

パラメーター名	意味
WAKEUP	起床条件

表 6.3 遷移先の動作モード

モード名	設定値
省電力・間欠動作 OFF	NONE (初期値)
シャットダウンモード	SHUTDOWN
スリープモード	SLEEP

表 6.4 起床条件

起床条件	設定値
RTC	RTC:[コンテナ終了からの経過秒数[a]]
SW1 押下	SW1
GPIO 割り込み	GPIO
USB デバイス接続	USB
UART データ受信	UART
SMS 受信	SMS
AIN [b]	アナログ入力電圧閾値割り込み発生時

[a]現在時刻からの経過秒数は MODE が SHUTDOWN の場合は 180 秒以上、SLEEP の場合は 60 秒以上を指定する必要があります。
[b]Armadillo-IoT ゲートウェイ +Di8+Ai4 のみ使用可能です



Cat.1 モデルで SMS 受信を起床条件に指定すると、間欠動作が正常に動作しません。SMS はデフォルトで起床条件に含まれているため、 Cat.1 モデルで間欠動作を実施する際は WAKEUP から削除してください。

以下は遷移する動作モードがシャットダウンモード、起床条件が RTC(300 秒後起床) のパターンです。起床条件に RTC を設定した場合、「6.1.2.2. RTC アラーム割り込みでの起床」 で説明している、SoC 外付け RTC による分単位のアラーム割り込みで起床します。そのため、RTC:300 で"300 秒後起床"を指定した場合、実際に起床するまでの時間は、コンテナ終了から 300 秒～359 秒の間となります。なお、デフォルトでは省電力・間欠動作は OFF (MODE=NONE) となっています。

```
[armadillo ~]# vi /etc/atmark/power-utils.conf
TARGET='a6e-gw-container'
MODE='SHUTDOWN'
WAKEUP='RTC:300'
```

設定ファイル(/etc/atmark/power-utils.conf)変更後、変更内容を永続化するには「図 6.6. 状態遷移トリガにコンテナ終了通知を利用する場合の設定値を永続化する」に示すコマンドを実行してください。

```
[armadillo ~]# persist_file /etc/atmark/power-utils.conf
```

図 6.6 状態遷移トリガにコンテナ終了通知を利用する場合の設定値を永続化する

状態遷移トリガの対象はデフォルトでゲートウェイコンテナが指定されていますが、任意のコンテナを指定することも可能です。ここでは、 "my_container" というコンテナを状態遷移トリガの対象にする場合の設定を記載します。

```
[armadillo ~]# vi /etc/atmark/power-utils.conf ❶
TARGET='my_container' ❷
MODE='SHUTDOWN'
WAKEUP='RTC:300'
[armadillo ~]# persist_file /etc/atmark/power-utils.conf ❸
[armadillo ~]# vi /etc/atmark/containers/my_container.conf ❹
set_image docker.io/alpine
set_command ls /
add_hook --stage poststop -- /usr/bin/pwu_notify container_stop ❺
[armadillo ~]# persist_file /etc/atmark/containers/my_container.conf ❻
```

図 6.7 状態遷移トリガの対象コンテナを設定する

❶ 設定ファイル(/etc/atmark/power-utils.conf)を編集します。

❷ コンテナ名 my_container を指定します。

❸ 設定内容を永続化します。

❹ コンテナの設定ファイル(/etc/atmark/containers/my_container.conf)を編集します。記載内容の詳細は「6.9.2.1. イメージからコンテナを作成する」を参照してください。

❺ コンテナの終了を検知するため、フックを設定します。

❻ コンテナの設定内容を永続化します。

6.1.5. コンテナ終了後、指定した秒数だけスリープしてコンテナを再始動する

設定ファイル（/etc/atmark/power-utils.conf）で起床条件に RTC を指定して間欠動作する場合、コンテナが終了してから起床するまでの時間は、指定した秒数よりも最大 59 秒長くなります。これは、RTC アラーム割り込みでの起床に使用する、SoC 外付けの RTC による制限です。以下に述べる方法を使うと、コンテナ終了後、終了前にコンテナアプリケーションが指定した秒数だけスリープして、起床時にコンテナを再始動することができます。この方法を使う場合、設定ファイル（/etc/atmark/power-utils.conf）の MODE と WAKEUP の設定内容は無視されます。

以下に、"my_container"というコンテナをスリープモードへの状態遷移トリガの対象にして、コンテナアプリケーションの終了後 50 秒後に起床する手順を述べます。

1. 設定ファイル（/etc/atmark/power-utils.conf）を編集します。
2. コンテナ名 my_container を TARGET に指定します。
3. 設定内容を永続化します。
4. コンテナの設定ファイル（/etc/atmark/containers/my_container.conf）に、「図 6.8. コンテナ終了後に指定した秒数だけスリープして再始動する場合のコンテナ設定」 に示す行を追加します。
5. コンテナの設定内容を永続化します。
6. コンテナアプリケーション自身が、終了する前に /tmp/power-utils/sleep_secs というパスのファイルを作り、そのファイルに、スリープしたい秒数の文字列（つまり 50）を書き込みます。指定可能な秒数は、5 から 3600 です。

7. コンテナアプリケーションが自発終了します。コンテナアプリケーションが終了するとスリープモードに遷移して、sleep_secs に書き込んだ秒数が経過すると起床してコンテナが始動します。

```
[ -e /etc/atmark/power-utils ] || mkdir /etc/atmark/power-utils
add_volumes /etc/atmark/power-utils:/tmp/power-utils
```

図 6.8 コンテナ終了後に指定した秒数だけスリープして再始動する場合のコンテナ設定

コンテナアプリケーションがスリープしたい秒数を書き込んだ sleep_secs ファイルは、起床時に削除されます。このファイルが存在しない場合は、コンテナ終了時の状態遷移と起床条件は、設定ファイル（/etc/atmark/power-utils.conf）の MODE と WAKEUP で設定した内容になります。

6.1.6. スリープ動作の禁止と禁止解除

aiot-sleep および aiot-alaram-poweroff コマンドによるスリープや シャットダウンの動作を、一時的に禁止することができます。禁止しているときに aiot-sleep や aiot-alarm-poweroff コマンドを実行すると、実行途中でブロックされて、禁止解除するまでブロックされたまま停止します。禁止解除すると停止状態から復帰して、スリープやシャットダウンが行われます。aiot-sleep によるスリープや aiot-alarm-poweroff によるシャットダウンの動作を、一時的に禁止するコマンド、および禁止解除するコマンドを、以下に示します：

```
[armadillo ~]# power-utils inhibit_sleep
```

図 6.9 スリープやシャットダウンの動作を一時的に禁止する

```
[armadillo ~]# power-utils allow_sleep
```

図 6.10 スリープやシャットダウンの禁止を解除する

power-utils inhibit_sleep コマンドを実行すると、power-utils allow_sleep コマンドを実行するか、または Armadillo を再起動するまでの間、aiot-sleep と aiot-alarm-poweroff によるスリープやシャットダウンの動作を禁止します。



この機能は、主に armadillo-twin-agent のために導入したものです。

aiot-set-wake-trigger と aiot-sleep や aiot-alarm-poweroff を組み合わせて使い、間欠動作するプログラムを開発する時に、スリープ動作の禁止と解除機能を利用すると便利なこともあるでしょう。開発中のプログラムが間欠動作用に aiot-sleep を実行して間欠動作する場合、開発中のデバッグ作業を行うときに、プログラムを変更することなくスリープ動作を一時的に禁止できるからです。

コンテナ終了通知をトリガに間欠動作を行う(「6.1.4. 状態遷移トリガにコンテナ終了通知を利用する」)設定にしている場合に、間欠動作のトリガにしているコンテナアプリケーションをデバッグしなければいけない時は、デバッグしている間、コンテナが終了しても何も起きないようにしたくなるかも知れません。aiot-sleep コマンドや aiot-alarm-poweroff コマンドの動作を無効にすれば、コンテナ終了通知が無視されます。間欠動作用の atmark-power-utils サービスが、aiot-sleep と aiot-alarm-

poweroff を使っているからです。aiot-sleep によるスリープや aiot-alarm-poweroff によるシャットダウンの動作を、無効にするコマンド、および有効に戻すコマンドを、以下に示します。

```
[armadillo ~]# power-utils disable_sleep
```

図 6.11 スリープやシャットダウンの動作を無効にする

```
[armadillo ~]# power-utils enable_sleep
```

図 6.12 スリープやシャットダウンの動作を有効に戻す

power-utils disable_sleep コマンドを実行すると、power-utils inhibit_sleep コマンドと違い、aiot-sleep や aiot-alarm-poweroff コマンドを実行するとエラー終了します。power-utils enable_sleep コマンドを実行するか、または Armadillo を再起動するまでの間、aiot-sleep と aiot-alarm-poweroff によるスリープやシャットダウンの動作を無効化します。

6.2. persist_file について

Armadillo Base OS ではルートファイルシステムに OverlayFS を採用しています。

「図 6.13. OverlayFS の構成」のように、Armadillo Base OS では「下位層」である eMMC 上に存在するファイルシステムに対して「上位層」である RAM 上のファイルシステムを統合しています。

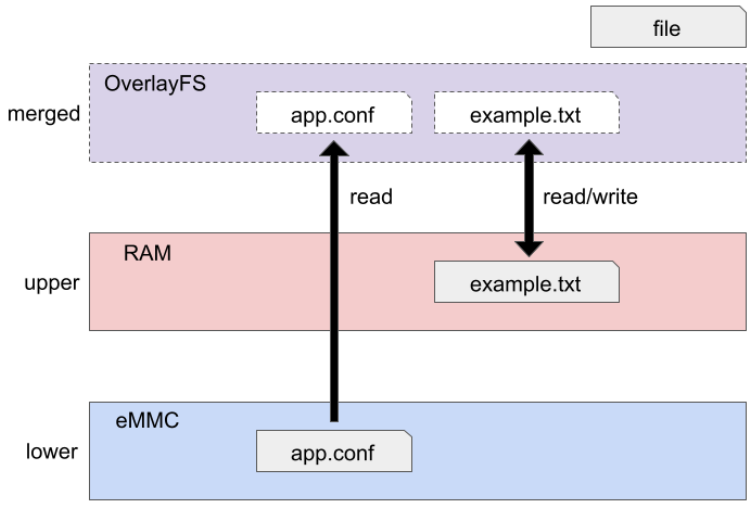


図 6.13 OverlayFS の構成

このシステムを OverlayFS と呼び、ユーザーランド上では OverlayFS がファイルシステムとして見えています。

Armadillo Base OS でファイルを作成すると RAM 上でファイル（またはディレクトリ）が作成されます。これにより、ファイル内容を変更した後 Armadillo の電源を切ると変更内容は保持されません。

`persist_file` コマンドは RAM 上のファイル（またはディレクトリ）を eMMC 上に反映するためのコマンドです。

開発中などに rootfs の変更内容を保持するには、変更したファイルに対して persist_file コマンドを使用する必要があります。

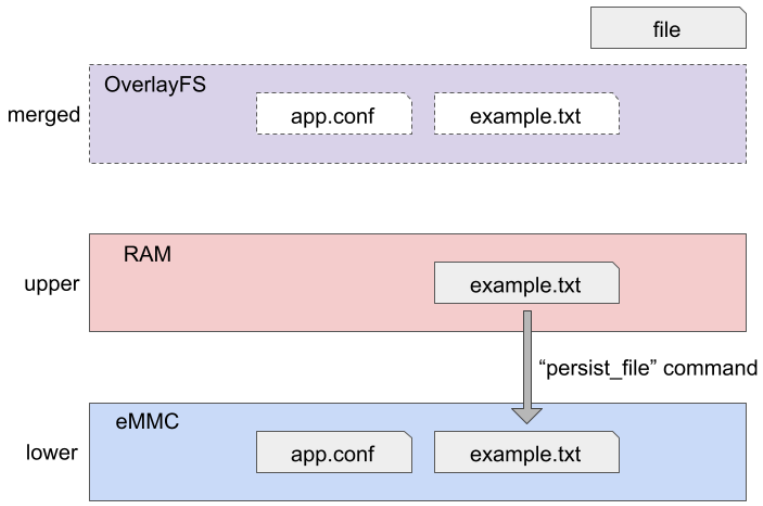


図 6.14 persist_file コマンドの説明

OverlayFS からファイルを書き込むと RAM 上のファイルに書き込みが行われます。eMMC 上にのみファイルが存在する場合、「図 6.15. 書き込み時の OverlayFS の挙動」に示すように、そのファイルは eMMC から読み込まれて編集した内容は RAM 上に書き込まれます。

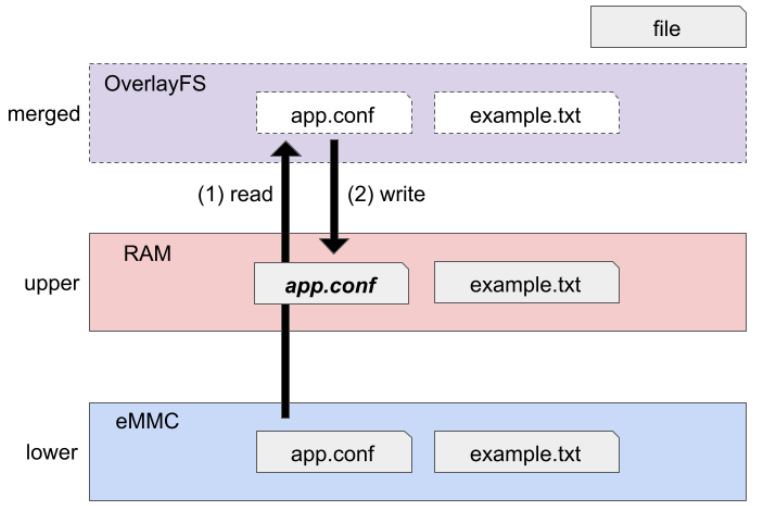


図 6.15 書き込み時の OverlayFS の挙動

編集内容を eMMC 上のファイルに反映させるためには、「図 6.16. 書き込み後の persist_file コマンドの実行」に示すように、persist_file コマンドを実行する必要があります。

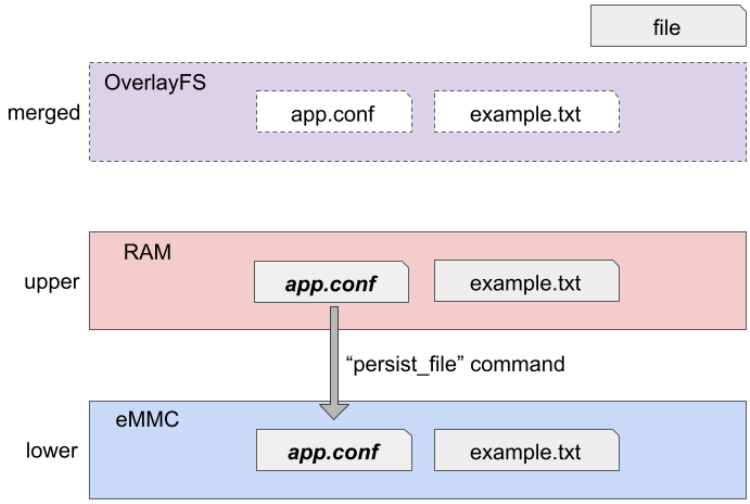


図 6.16 書き込み後の persist_file コマンドの実行

開発以外の時は安全のため、ソフトウェアアップデート（SWUpdate）による更新を実行してください。SWUpdate に関しては 「3.3.3. アップデート機能について」 を参照してください。

rootfs の内容を変更しても、ソフトウェアアップデートを実施した際に変更した内容が保持されない可能性があります。ソフトウェアアップデート実施後も変更内容を保持する手順に関しては 「6.5. swupdate_preserve_files について」 を参照してください。

以下、persist_file コマンドの使い方について説明します。

persist_file コマンドの概要を 「図 6.17. persist_file のヘルプ」 に示します。

```
[armadillo ~]# persist_file -h
Usage: /usr/bin/persist_file [options] file [more files...]

Mode selection:
  (none) single entry copy
  -d, --delete   delete file
  -l, --list     list content of overlay
  -a, --apk      apk mode: pass any argument after that to apk on rootfs
  -R, --revert   revert change: only delete from overlay, making it
                 look like the file was reverted back to original state

Copy options:
  -r, --recurse  recursive copy (note this also removes files!)
  -p, --preserve make the copy persist through baseos upgrade
                 by adding entries to /etc/swupdate_preserve_files
  -P, --preserve-post   same, but copy after upgrade (POST)

Delete options:
  -r, --recurse  recursively delete files

Common options:
  -v, --verbose  verbose mode for all underlying commands
```

```
Note this directly manipulates overlayfs lower directories
so might need a reboot to take effect
```

図 6.17 persist_file のヘルプ

1. ファイルの保存・削除手順例

```
[armadillo ~]# echo test > test
[armadillo ~]# persist_file -rv /root
'/root/test' -> '/mnt/root/test' ❶
'/root/.ash_history' -> '/mnt/root/.ash_history'
[armadillo ~]# rm -f test
[armadillo ~]# persist_file -rv /root
removed '/mnt/root/test' ❷
removed '/mnt/root/.ash_history' ❸
'/root/.ash_history' -> '/mnt/root/.ash_history'
```

図 6.18 persist_file 保存・削除手順例

❶ 追加・変更したファイルを rootfs へコピーします。

❷ -r を指定すると、ひとつ前の rm -f コマンドで削除したファイルが rootfs からも削除されますのでご注意ください。

❸ すでに rootfs に存在するファイルも一度削除してからコピーするため、このようなメッセージが表示されます。

2. ソフトウェアアップデート後も変更を維持する手順例

```
[armadillo ~]# vi /etc/conf.d/podman-atmark ❶
[armadillo ~]# persist_file -P /etc/conf.d/podman-atmark ❷
[armadillo ~]# tail -n 2 /etc/swupdate_preserve_files ❸
# persist_file 20211216
POST /etc/conf.d/podman-atmark
```

図 6.19 persist_file ソフトウェアアップデート後も変更を維持する手順例

❶ 何らかのファイルの内容を変更します。

❷ -P オプションを付与して persist_file を実行します。

❸ swupdate_preserve_files に追加されたことを確認します。

3. 変更ファイルの一覧表示例

```
[armadillo ~]# mkdir dir
[armadillo ~]# persist_file -l
directory          /
directory          /root
opaque directory   /root/dir ❶
whiteout           /root/test ❷
regular file       /root/.ash_history
```

```
directory           /etc
regular file        /etc/resolv.conf
directory           /var
symbolic link       /var/lock
: (省略)
```

図 6.20 persist_file 変更ファイルの一覧表示例

❶ rootfs のファイルを見せないディレクトリは opaque directory と表示されます。

❷ 削除したファイルは whiteout と表示されます。

4. パッケージをインストールする時は apk コマンドを使用してメモリ上にインストールできますが、persist_file コマンドで rootfs に直接インストールすることも可能です。

```
[armadillo ~]# persist_file -a add strace
(1/3) Installing fts (1.2.7-r1)
(2/3) Installing libelf (0.185-r0)
(3/3) Installing strace (5.14-r0)
Executing busybox-1.34.1-r3.trigger
OK: 251 MiB in 188 packages
Install succeeded, but might not work in the running system
Please reboot if installed program does not work ❶
[armadillo ~]# strace ls
: (省略)
exit_group(0)                           = ?
+++ exited with 0 +++
```

図 6.21 persist_file でのパッケージインストール手順例

❶ この例では Armadillo を再起動せずにインストールしたコマンドを使用できましたが、Armadillo の再起動が必要となるパッケージもありますので、その場合は Armadillo を再起動してください。



ABOS を更新する際に追加したパッケージのない状態に戻りますので、こちらのコマンドは開発・解析用であって、運用に利用できません。運用に必要なソフトウェアは基本的にすべてコンテナにしていただくように推奨していますが、パッケージの追加がどうしても必要な場合は 「6.30.3. Alpine Linux ルートファイルシステムをビルドする」 でパッケージを追加したイメージをビルドして運用してください。

6.3. swupdate を使用してアップデートする

6.3.1. swupdate で可能なアップデート

swupdate を実行する目的としては以下が考えられます。

a. コンテナをアップデートしたい

開発したコンテナのアップデートが可能です。

b. ユーザーデータディレクトリや Armadillo Base OS のファイルを差分アップデートしたい

ユーザーデータをアップデートする場合は、以下のディレクトリを更新します。

・ /var/app/rollback/volumes

ユーザーディレクトリについては「3.3.5.1. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)」を参照してください。



SWUpdate による /var/app/volumes の更新は推奨しません。

/var/app/volumes が二面化されていないため、書込みの途中で問題が発生した場合、不完全な書込みになる恐れがあります。

また、アップデート中にアプリケーションがそのデータにアクセスすると、書込み中のデータにアクセスすることになります。

/var/app/volumes のデータに対して更新が必要な場合、SWUpdate では /var/app/rollback/volumes に更新するデータを書き込んでください。その後、次回起動時にアプリケーション側から /var/app/rollback/volumes に書き込んだデータを /var/app/volumes に移動するようにしてください。

c. Armadillo Base OS を一括アップデートしたい

アットマークテクノがリリースする Armadillo Base OS の機能追加、更新、セキュリティパッチの追加が可能です。

d. ブートローダーをアップデートしたい

アットマークテクノがリリースするブートローダーのアップデートが可能です。

「2.1.3. Armadillo Base OS とは」で示すように、Armadillo Base OS は OS・ブートローダー・コンテナ部分の安全性を担保するため二面化しています。

それにより、万が一アップデートに失敗した場合でも起動中のシステムに影響ありません。

以降、それぞれの目的ごとに swupdate によるアップデートの流れを示します。

・ a, b の場合

「6.3.2. コンテナのアップデート、ユーザーデータディレクトリや Armadillo Base OS の差分アップデート」を参照してください。

・ c の場合

「6.3.3. Armadillo Base OS の一括アップデート」を参照してください。

・ d の場合

「6.3.4. ブートローダーのアップデート」を参照してください。

6.3.2. コンテナのアップデート、ユーザーデータディレクトリや Armadillo Base OS の差分アップデート

以下にアップデートの流れを示します。

ここでは、boot して起動中の面を A 面、アップデート先の面を B 面とします。

1. Armadillo Base OS を B 面へコピー

   Armadillo Base OS を B 面にコピーする流れを「図 6.22. Armadillo Base OS を B 面にコピー」に示します。

   A 面と B 面の Armadillo Base OS が同期しているか確認します。

   同期していない場合、 A 面の Armadillo Base OS を B 面にコピーします。

   同期している場合はコピーしません。

   `swdesc_option version` で指定するバージョンの書き方については「6.4.1. インストールバージョンを指定する」を参照してください。

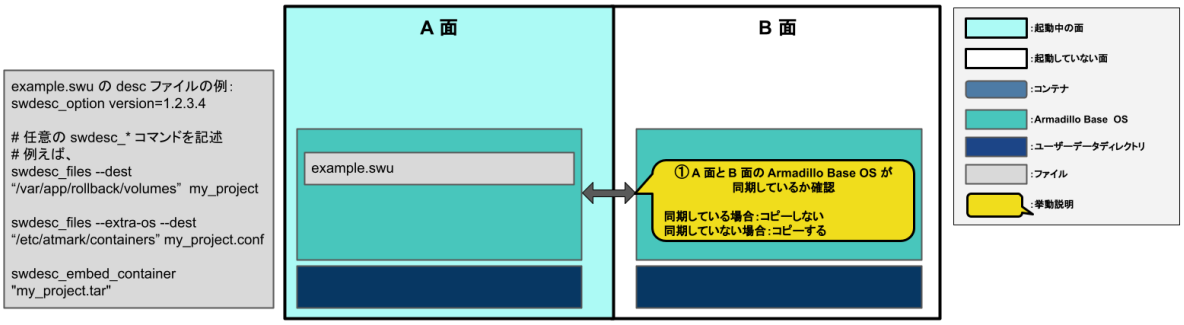


図 6.22 Armadillo Base OS を B 面にコピー

2. コマンドを順番に実行

   「図 6.23. desc ファイルに記述した swudesc_* コマンドを実行」に示すように、desc ファイルに記述した順番に従って swudesc_* コマンドを実行します。

   「6.4.1. インストールバージョンを指定する」に示すように、swdesc_* コマンドによって Armadillo Base OS に対して書き込みをする場合は --extra-os オプションをつけてください。

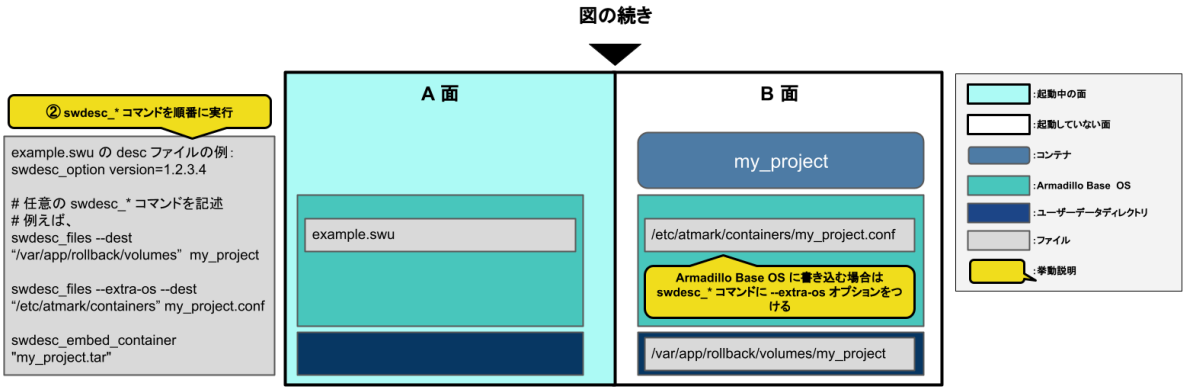


図 6.23 desc ファイルに記述した swudesc_* コマンドを実行

swudesc_* コマンドの種類を「表 6.5. swudesc_* コマンドの種類」に示します。

表 6.5 swudesc_* コマンドの種類

おおまかな機能	コマンド名	説明
ファイル転送 参照先：「6.4.2. Armadillo へファイルを転送する」	swdesc_files	指定したファイルをアップデート先の環境にコピー
	swdesc_tar	指定した tar アーカイブをアップデート先の環境に展開してコピー
コマンド実行 参照先：「6.4.3. Armadillo 上で任意のコマンドを実行する」	swdesc_command	指定したコマンドをアップデート先の環境で実行
	swdesc_script	指定したスクリプトをアップデート先の環境で実行
ファイル転送 + コマンド実行 参照先：「6.4.4. Armadillo にファイルを転送し、そのファイルをコマンド内で使用する」	swdesc_exec	指定したファイルをアップデート先の環境にコピーした後、そのファイル名を"$1"としてコマンドを実行
起動中の面に対してコマンド実行（非推奨） 参照先：「6.4.5. 動作中の環境でのコマンドの実行」	swdesc_command_nochroot	指定したコマンドを起動中の環境で実行
	swdesc_script_nochroot	指定したスクリプトを起動中の環境で実行
起動中の面に対してファイル転送 + コマンド実行（非推奨） 参照先：「6.4.5. 動作中の環境でのコマンドの実行」	swdesc_exec_nochroot	指定したファイルを起動中の環境にコピーした後、そのファイル名を"$1"としてコマンドを実行
コンテナイメージの転送 参照先：「6.4.6. Armadillo にコンテナイメージを転送する」	swdesc_embed_container	SWU ファイルに含まれるコンテナイメージの tar アーカイブをアップデート先の環境に展開
	swdesc_pull_container	アップデート先の環境でコンテナイメージをダウンロード
	swdesc_usb_container	SWU ファイルに含めない形で用意したコンテナイメージの tar アーカイブをアップデート先の環境に展開

3. アップデート完了後の挙動

   デフォルトではアップデート後に再起動（ POST_ACTION=reboot ）が行われます。

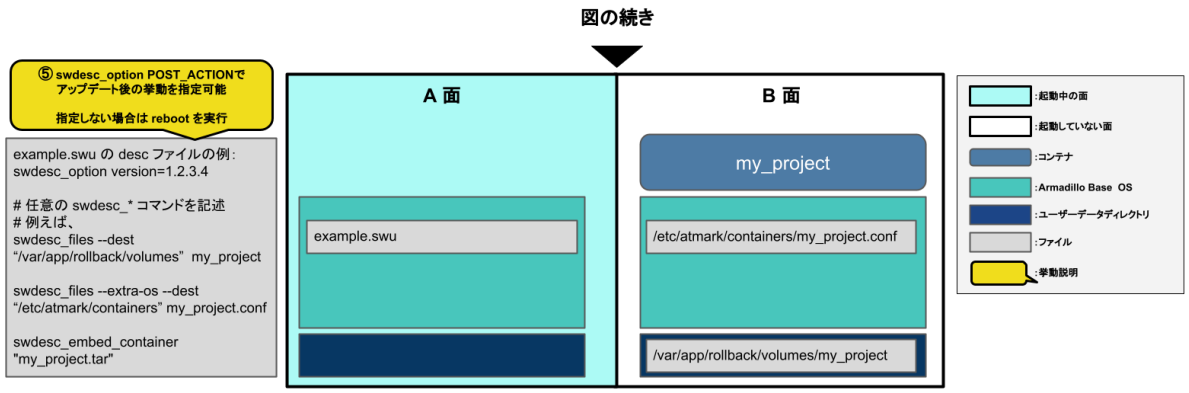


図 6.24 アップデート完了後の挙動

アップデート後の挙動を変更するには desc ファイルに `swdesc_option POST_ACTION` を追加してください。

`swdesc_option POST_ACTION` に指定できる挙動の種類を「表 6.6. アップデート完了後の挙動の種類」に示します。

表 6.6 アップデート完了後の挙動の種類

オプション名	説明
container	アップデート後にコンテナのみを再起動 (ただし、アップデート時に --extra_os オプションを指定したコマンドが実行された場合は reboot になる)
poweroff	アップデート後にシャットダウン
reboot	アップデートの後に再起動
wait	アップデート後に再起動は行われず、次回起動時に B 面に切り替わる

`swdesc_option POST_ACTION` の詳細は「6.4.10. SWUpdate 実行中/完了後の挙動を指定する」を参照してください。

4. B 面への切り替え

「図 6.25. B 面への切り替え」に示すように、正常にアップデートが行われると、次回起動時に B 面に切り替わります。

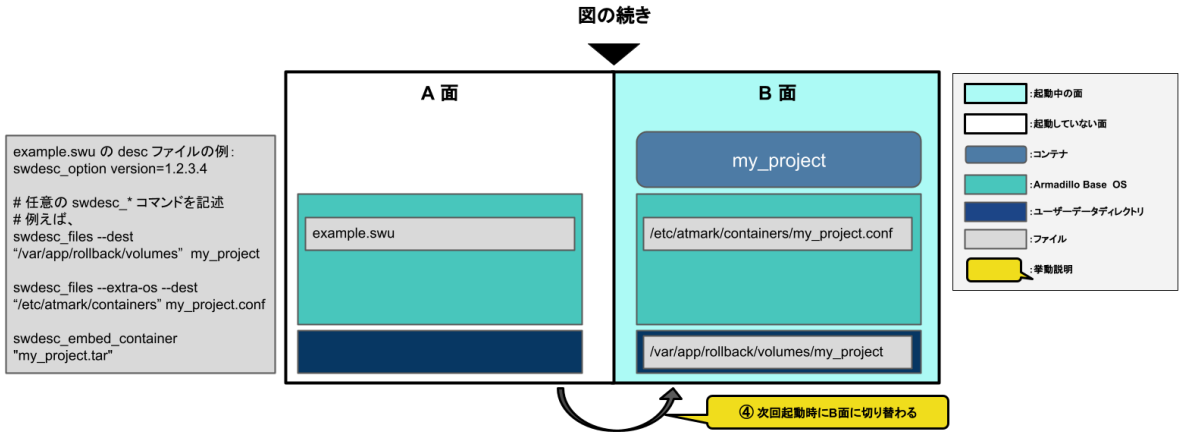


図 6.25 B 面への切り替え

5. desc ファイルの書き方の例

   下記に SWUpdate を用いたアップデートの例を示します。

   - コンテナイメージとコンテナ自動設定ファイルのアップデート：「6.9.2.17. イメージを SWUpdate で転送する」
   - sshd の設定：「6.4.11.1. 例: sshd を有効にする」

6.3.3. Armadillo Base OS の一括アップデート

アップデートの流れを示します。

ここでは、boot して起動中の面を A 面、アップデート先の面を B 面とします。

1. Armadillo Base OS とアップデート後に保持するファイルを B 面へコピー

   Armadillo Base OS とアップデート後に保持するファイルを B 面にコピーする流れを「図 6.26. Armadillo Base OS とファイルを B 面にコピー」に示します。

   「6.4.1. インストールバージョンを指定する」に示すように、Armadillo Base OS の tar アーカイブを展開する swdesc_tar コマンドに --base-os オプションをつけてください。

   `swdesc_option version` で指定するバージョンの書き方については「6.4.1. インストールバージョンを指定する」を参照してください。

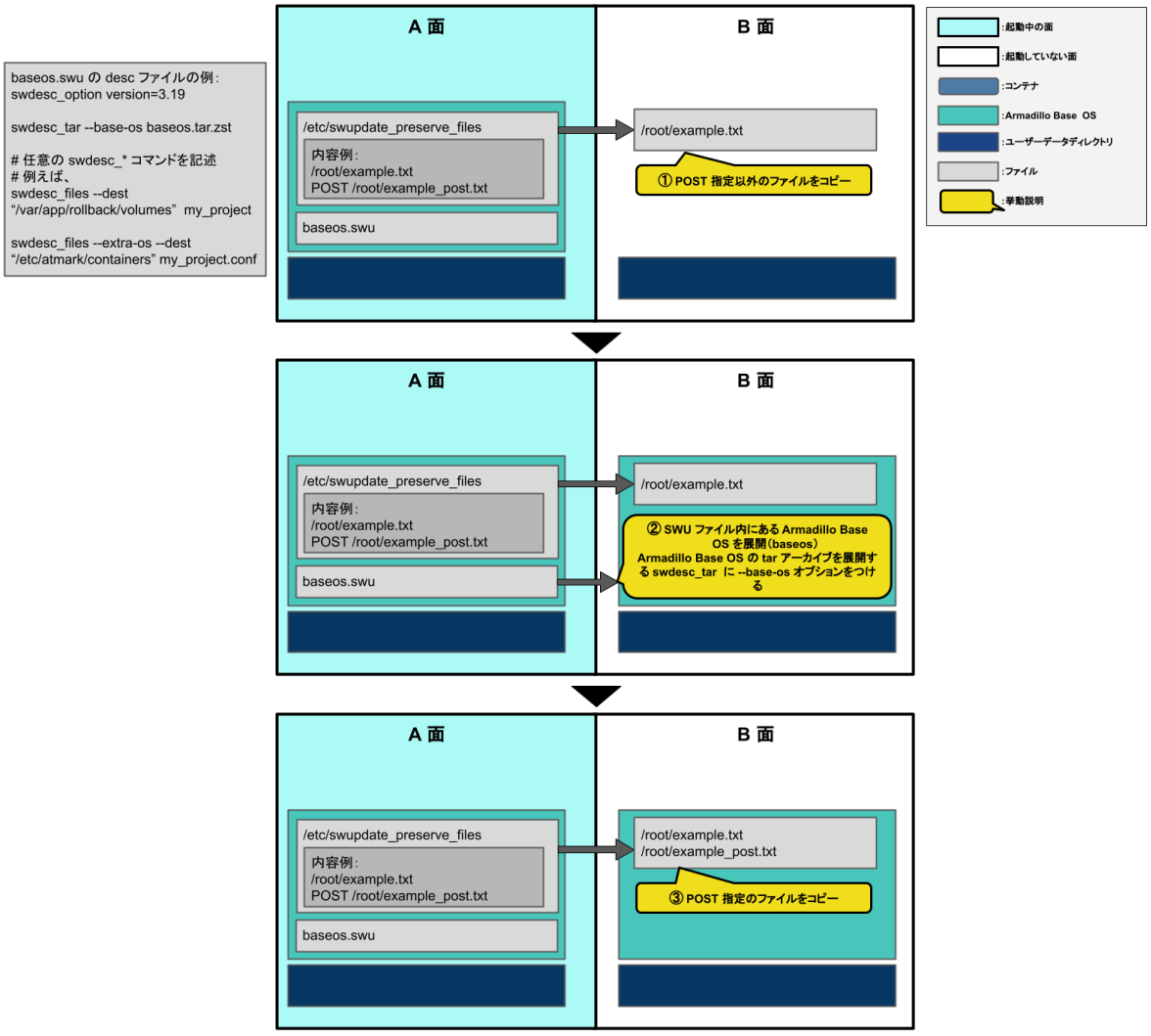


図 6.26 Armadillo Base OS とファイルを B 面にコピー

1. `/etc/swupdate_preserve` に記載された POST 指定以外のファイルを B 面にコピーします。

2. SWU ファイル内にある Armadillo Base OS を B 面に展開します。

3. `/etc/swupdate_preserve` に記載された POST 指定のファイルを B 面にコピーします。

`/etc/swupdate_preserve` への追記方法については「6.5. swupdate_preserve_files について」と「4.4.1. /etc/swupdate_preserve_file への追記」を参照してください。

2. コマンドを順番に実行

「図 6.27. desc ファイルに記述した swudesc_* コマンドを実行」に示すように、desc ファイルに記述した順番に従って swudesc_* コマンドを実行します。

「6.4.1. インストールバージョンを指定する」に示すように、swdesc_* コマンドによって Armadillo Base OS に対して書き込みをする場合は --extra-os オプションをつけてください。

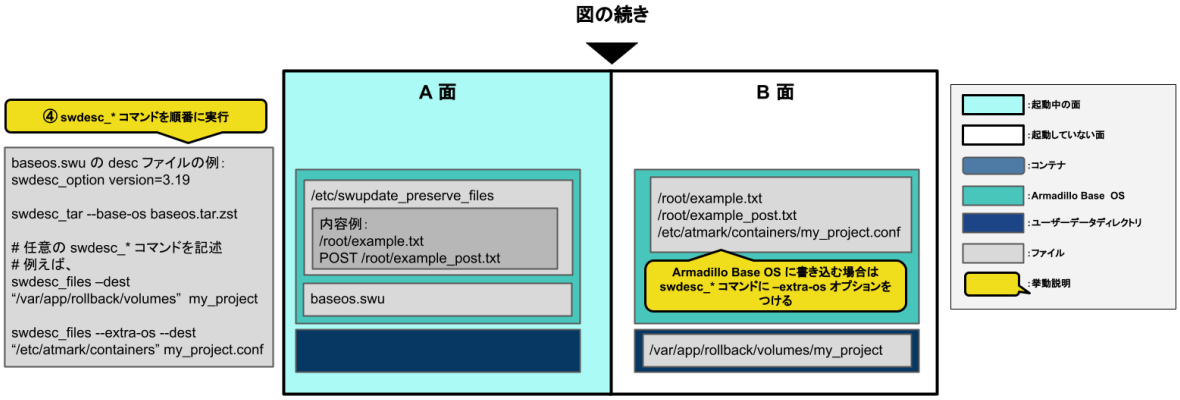


図 6.27 desc ファイルに記述した swudesc_* コマンドを実行

swudesc_* コマンドの種類を「表 6.7. swudesc_* コマンドの種類」に示します。

表 6.7 swudesc_* コマンドの種類

おおまかな機能	コマンド名	説明
ファイル転送 参照先：「6.4.2. Armadillo へファイルを転送する」	swdesc_files	指定したファイルをアップデート先の環境にコピー
	swdesc_tar	指定した tar アーカイブをアップデート先の環境に展開してコピー
コマンド実行 参照先：「6.4.3. Armadillo 上で任意のコマンドを実行する」	swdesc_command	指定したコマンドをアップデート先の環境で実行
	swdesc_script	指定したスクリプトをアップデート先の環境で実行
ファイル転送 + コマンド実行 参照先：「6.4.4. Armadillo にファイルを転送し、そのファイルをコマンド内で使用する」	swdesc_exec	指定したファイルをアップデート先の環境にコピーした後、そのファイル名を"$1"としてコマンドを実行
起動中の面に対してコマンド実行（非推奨） 参照先：「6.4.5. 動作中の環境でのコマンドの実行」	swdesc_command_nochroot	指定したコマンドを起動中の環境で実行
	swdesc_script_nochroot	指定したスクリプトを起動中の環境で実行
起動中の面に対してファイル転送 + コマンド実行（非推奨） 参照先：「6.4.5. 動作中の環境でのコマンドの実行」	swdesc_exec_nochroot	指定したファイルを起動中の環境にコピーした後、そのファイル名を"$1"としてコマンドを実行
コンテナイメージの転送 参照先：「6.4.6. Armadillo にコンテナイメージを転送する」	swdesc_embed_container	SWU ファイルに含まれるコンテナイメージの tar アーカイブをアップデート先の環境に展開
	swdesc_pull_container	アップデート先の環境でコンテナイメージをダウンロード
	swdesc_usb_container	SWU ファイルに含めない形で用意したコンテナイメージの tar アーカイブをアップデート先の環境に展開
ブートローダーの更新 参照先：「6.4.7. Armadillo のブートローダーを更新する」	swdesc_boot	SWU ファイルに含まれるブートローダーをアップデート先の環境に展開

3. アップデート完了後の挙動

   デフォルトではアップデート後に再起動（ POST_ACTION=reboot ）が行われます。

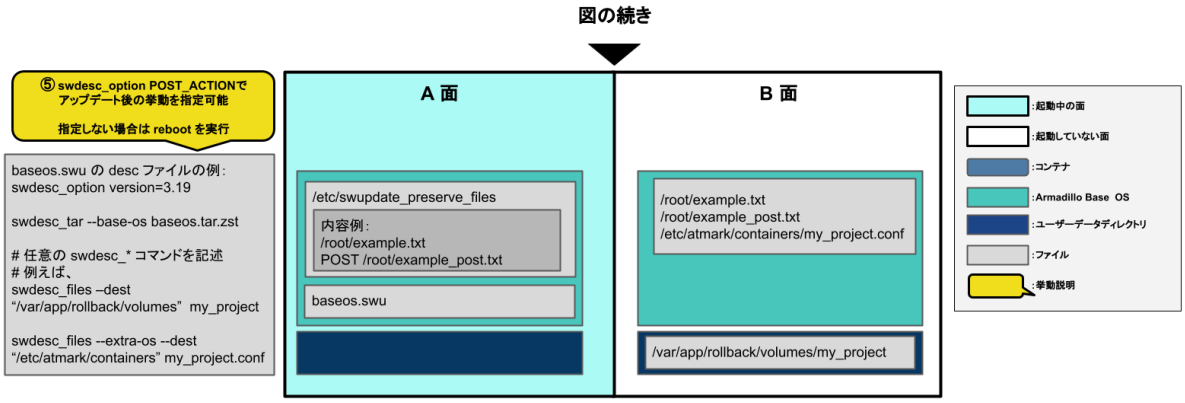


図 6.28 アップデート完了後の挙動

アップデート後の挙動を変更するには desc ファイルに `swdesc_option POST_ACTION` を追加してください。

`swdesc_option POST_ACTION` に指定できる挙動の種類を「表 6.8. アップデート完了後の挙動の種類」に示します。

表 6.8 アップデート完了後の挙動の種類

オプション名	説明
poweroff	アップデート後にシャットダウン
reboot	アップデートの後に再起動
wait	アップデート後に再起動は行われず、次回起動時に B 面に切り替わる

`swdesc_option POST_ACTION` の詳細は「6.4.10. SWUpdate 実行中/完了後の挙動を指定する」を参照してください。

4. B 面への切り替え

「図 6.29. B 面への切り替え（component=base_os)」に示すように、正常にアップデートが行われると、次回起動時に B 面に切り替わります。

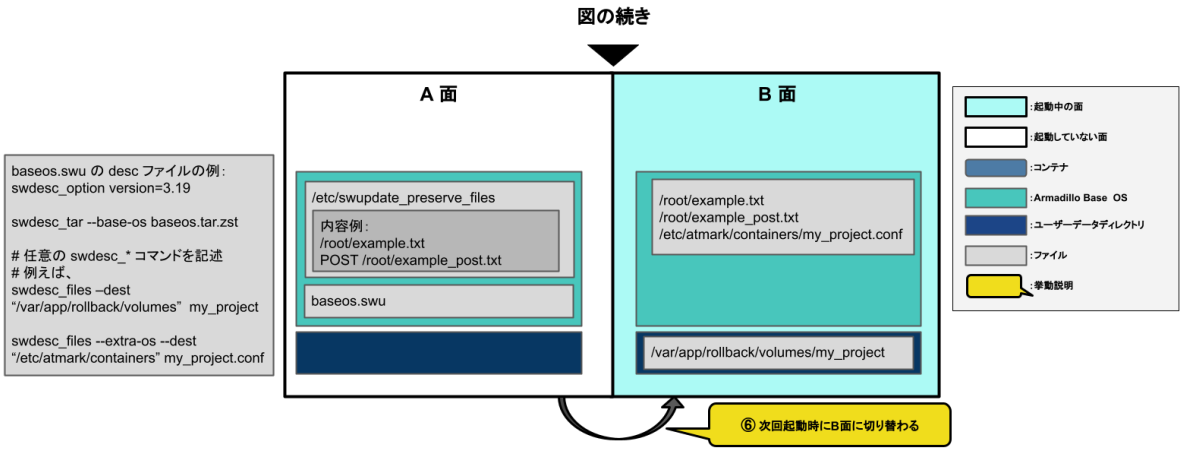


図 6.29 B 面への切り替え（component=base_os)

5. desc ファイルの書き方の例

下記に SWUpdate を用いたアップデートの例を示します。

- Armadillo Base OS のアップデート：「6.4.11.2. 例: Armadillo Base OS アップデート」
- Alpine Linux ルートファイルシステムのアップデート：「6.30.3. Alpine Linux ルートファイルシステムをビルドする」

6.3.4. ブートローダーのアップデート

swdesc_* コマンドには、swdesc_boot を指定してください。

swdesc_boot については「6.4.7. Armadillo のブートローダーを更新する」を参照してください。

ブートローダーのアップデートの流れは以下の通りです。

- desc ファイルに swdesc_boot がある場合

  SWU ファイルに含まれるブートローダーを B 面に書き込む

- desc ファイルに swdesc_boot がない場合

  A 面のブートローダーを B 面にコピーする

下記に SWUpdate を用いたアップデートの例を示します。

- ブートローダーのアップデート：「6.30.1. ブートローダーをビルドする」

6.3.5. swupdate がエラーする場合の対処

SWU イメージのインストール動作は、「3.3.4.1. SWU イメージとは」で述べたように swupdate が実行します。mkswu で作成した SWU イメージの内容が適切でなかったり、あるいは、ストレージの空き容量が不足していたりするなど、いくつかの理由で swupdate のインストール動作が失敗することがあります。インストールに失敗すると、swupdate は /var/log/messages にエラーメッセージのログを残しますので、エラーメッセージを見ると、エラーの内容・原因が分かります。

エラーの原因ごとに、エラーメッセージとエラーの内容および対処方法を記した FAQ ページ (https://armadillo.atmark-techno.com/faq/swupdate-troubleshooting-abos) を公開しています。SWU イメージのインストールに失敗して対処法が分からないときは、この FAQ ページをご覧ください。

6.4. mkswu の .desc ファイルを編集する

mkswu で SWU イメージを生成するためには、 desc ファイルを正しく作成する必要があります。以下では、 desc ファイルの記法について紹介します。

6.4.1. インストールバージョンを指定する

```
swdesc_option component=<component>
swdesc_option version=<version>
か
swdesc_xxx --version <component> <version> [options]
```

- <component>は以下のどれかにしてください (デフォルトでは .desc ファイルのファイル名を使います)

1. base_os: rootfs (Armadillo Base OS)を最初から書き込む時に使います。現在のファイルシステムは保存されないです。

   この場合、/etc/swupdate_preserve_files に載ってるファイルのみをコピーして新しい base OS を展開します。

   この component がないと現在の rootfs のすべてがコピーされます。

   swdesc_tar コマンドで rootfs (Armadillo Base OS) の tar アーカイブを展開する時に、--base-os オプションをつけることで component に base_os を指定したときと同じ動作となります。

2. extra_os.<文字列>: rootfs の変更を行う時に使います。<文字列> には任意の文字列を指定します。

   rootfs を変更を行う時に使います。 swdesc_* コマンドに --extra-os オプションを追加すると、 component に自動的に extra_os. を足します。

3. <文字列> (コンテナの名前などの任意の文字列）: rootfs の変更がないときに使います。

   この component を使うと rootfs の変更ができませんのでご注意ください。

・ アップデートを行う際にこのバージョンと現在のバージョンを比べてアップデートの判断を行います。

<component> がまだインストールされてなかった時や <version> が上がる時にインストールします。

デフォルトではダウングレードはできませんが、 --install-if=different オプションを追加することで <version> が変わる際にインストール可能になります。

アップデートの一部をインストールすることもありますので、複数の component で管理し、いくつかの古いバージョンに対応するアップデートも作成可能です。

・ バージョンの指定方法

swdesc_option version で指定可能なバージョンのフォーマットは以下の 2 種類があります。

・ x[.y[.z[-t]]]

  x, y, z にはそれぞれ 0 〜 2147483647 の整数を適用してください。t には任意のアルファベットまたは 0 〜 147483647 の整数を適用してください。

  成功例は以下です：

  ・ 1

  ・ 1.2.3

  ・ 1.2.3-4

  ・ 1.2.3-abc.4

  ・ 1.2.3-a.b.c.4

  失敗例は以下です：

  ・ 2147483648

x には 0 ～ 2147483647 の整数を適用してください。

- 1.2.3-a.2147483648

  t には 0 ～ 2147483647 の整数を適用してください。

- 1.2.3-abc123

  t には 数字とアルファベットを混在しないでください。1.2.3-abc.123 ならば可能です。

- a.2.3

  x にはアルファベットではなく 0 ～ 2147483647 の整数を適用してください。

- 1.1.1.1-a

  x[.y[.z[-t]]]の形式で書いてください。

- x.y.z.t

  x, y, z, t にはそれぞれ 0 ～ 65535 の整数を適用してください。

  成功例は以下です：

  - 1.2.3.4
  - 65535.65535.65535.65535
  - 65535.2.3.4



アットマークテクノがリリースするファームウェアはバージョンのサフィックスとして"-at.[数字]"を含めています。オリジナルのサフィックスをつける場合は、"-at.[数字]"を使用しないことを強く推奨します。

  失敗例は以下です：

  - 65536.2.3.4

    x には 0 ～ 65535 の整数を適用してください。

  - 1.2.3.a

    t にはアルファベットではなく 0 ～ 65535 の整数を適用してください。

  - 1.2.3.4.5

    x.y.z.t の形式で書いてください。

6.4.2. Armadillo へファイルを転送する

- `swdesc_tar` と `swdesc_files` でファイルを転送します。

```
swdesc_tar [--dest <dest>] [--preserve-attributes] <tar_file>
swdesc_files [--dest <dest>] [--basedir <basedir>] [--preserve-attributes] ¥
             <file> [<more files>]
```

swdesc_tar の場合、予め用意されてある tar アーカイブをこのままデバイスで展開します。swdesc_files は、機能としては swdesc_tar と同じですが、tar アーカイブを作る手間を省けます。

--dest <dest> で展開先を選ぶことができます。component や指定するオプションによってデフォルトのコピー先が違います。

- --extra-os オプションを指定する場合
  - コピー先：/
  - 例： swdesc_files --extra-os "resources"
    - resources ディレクトリの中身を / に展開します。
- component が 'base_os` または extra_os.* の場合
  - コピー先：/
  - 例： swdesc_files --version extra_os.my_component 1 "resources"
    - resources ディレクトリの中身を / に展開します。
- それ以外の component の場合
  - コピー先：/var/app/rollback/volume/
  - 例：swdesc_files --version my_component 1 --dest "my_project" "resources"
    - resources ディレクトリの中身を /var/app/rollback/volume/my_project に展開します。
    - この場合は /var/app/volumes と /var/app/rollback/volumes 以外にコピーできませんので、--extra-os をご使用ください。

--preserve-attributes を指定しない場合はファイルのオーナー、モード、タイムスタンプ等が保存されませんので、必要な場合は設定してください。バージョンが base_os の場合は自動的に設定されます。

swdesc_files では、--basedir <basedir> でアーカイブ内のパスをどこで切るかを決めます。(このオプションは swdesc_tar にはありません。)

- 例えば、swdesc_files --extra-os --basedir /dir /dir/subdir/file ではデバイスに /subdir/file を作成します。
- デフォルトは <file> から設定されます。ディレクトリであればそのまま basedir として使います。それ以外であれば親ディレクトリを使います。

6.4.3. Armadillo 上で任意のコマンドを実行する

- swdesc_command や swdesc_script でコマンドを実行します。

```
swdesc_command <command> [<more commands>]
swdesc_script <script>
```

アップデート先となるパーティションをマウントして、そのマウント先でコマンドやスクリプトファイルを実行します。そのマウント先は OverlayFS を使用していないため、persist_file コマンドを swdesc_command や swdesc_script 内で実行する必要はありません。また、内部実装で安全に同期処理を行っているため、 sync コマンドも実行する必要はありません。

バージョンの component は base_os と extra_os 以外の場合、 /var/app/volumes と /var/app/rollback/volumes 以外は変更できないのでご注意ください。

コマンドの実行が失敗した場合、アップデートも失敗します。

6.4.4. Armadillo にファイルを転送し、そのファイルをコマンド内で使用する

- swdesc_exec でファイルを配り、コマンド内でそのファイルを使用します。

```
swdesc_exec <file> <command>
```

swdesc_command と同じくコマンドを実行しますが、<file> を先に転送してコマンド内で転送したファイル名を"$1"として使えます。

アップデート先となるパーティションをマウントして、そのマウント先でコマンドやスクリプトファイルを実行します。そのマウント先は OverlayFS を使用していないため、persist_file コマンドを swdesc_command や swdesc_script 内で実行する必要はありません。また、内部実装で安全に同期処理を行っているため、 sync コマンドも実行する必要はありません。

6.4.5. 動作中の環境でのコマンドの実行



本節で紹介する swdesc_command_nochroot、swdesc_script_nochroot、swdesc_exec_nochroot は基本的に使用することはありません。

swdesc_command、swdesc_script、swdesc_exec をご使用ください。

- swdesc_command_nochroot, swdesc_script_nochroot, swdesc_exec_nochroot は アップデート先の環境ではなく動作中の環境でコマンドを実行します。

  使い方は「6.4.2. Armadillo へファイルを転送する」と「6.4.4. Armadillo にファイルを転送し、そのファイルをコマンド内で使用する」に示した nochroot なしのバージョンと同じです。

  アップデート先の環境は /target にマウントされるので、nochroot のコマンドを用いてアップデート先の環境に対してアクセスすることは可能です。

  しかし、その方法によるアップデート先の環境に対するコマンドの実行は nochroot なしのコマンドでは実現できない特殊な場合にのみ行ってください。



nochroot コマンドは確認を一切しないため、Armadillo が起動できない状態になる可能性もあります。充分にご注意ください。

例が必要な場合は /usr/share/mkswu/examples/firmware_update.desc を参考にしてください。

6.4.6. Armadillo にコンテナイメージを転送する

- swdesc_embed_container, swdesc_usb_container, swdesc_pull_container で予め作成したコンテナを転送します。

```
swdesc_embed_container <container_archive>
swdesc_usb_container <container_archive>
swdesc_pull_container <container_url>
```

例は「6.9.2.15. リモートリポジトリにコンテナを送信する」、「6.9.2.17. イメージを SWUpdate で転送する」を参考にしてください。

6.4.7. Armadillo のブートローダーを更新する

- swdesc_boot でブートローダーを更新します。

```
swdesc_boot <boot image>
```

このコマンドだけはバージョンは自動的に設定されます。

6.4.8. SWU イメージの設定関連

コマンドの他には、設定変数もあります。以下の設定は /home/atmark/mkswu/mkswu.conf に設定できます。

- DESCRIPTION="<text>": イメージの説明、ログに残ります。
- PRIVKEY=<path>, PUBKEY=<path>: 署名鍵と証明書
- PRIVKEY_PASS=<val>: 鍵のパスワード（自動用）

  openssl の Pass Phrase をそのまま使いますので、pass:password, env:var や file:pathname のどれかを使えます。pass や env の場合他のプロセスに見られる恐れがありますので file をおすすめします。
- ENCRYPT_KEYFILE=<path>: 暗号化の鍵

6.4.9. Armadillo 上のコンテナイメージと自動起動用 conf ファイルを削除する

以下のオプションも mkswu.conf に設定できますが、.desc ファイルにも設定可能です。swdesc_option で指定することで、誤った使い方をした場合 mkswu の段階でエラーを出力しますので、必要な場合は使用してください。

- swdesc_option CONTAINER_CLEAR: インストールされているコンテナと /etc/atmark/containers/*.conf をすべて削除します。

  このオプションは簡単な初期化と考えてください。通常の運用では、不要になったイメージは自動的に削除されますのでこのオプションを設定する必要はありません。

6.4.10. SWUpdate 実行中/完了後の挙動を指定する

以下のオプションは Armadillo 上の /etc/atmark/baseos.conf に、例えば MKSWU_POST_ACTION=xxx として設定することができます。

その場合に swu に設定されなければ /etc の設定で実行されますので、アットマークテクノが用意している Base OS のアップデートでも動作の変更は可能です。swu に特定のオプションが設定された場合は設定されたオプションが優先されますので、一時的な変更も可能です。

- swdesc_option POST_ACTION=container: コンテナのみのアップデート後に再起動を行いません。コンテナの中身だけをアップデートする場合、Armadillo-IoT ゲートウェイ A6E を再起動せずにコンテナだけを再起動させます。
- swdesc_option POST_ACTION=poweroff: アップデート後にシャットダウンを行います。
- swdesc_option POST_ACTION=wait: アップデート後に自動的に再起動は行われず、次回起動時にアップデートが適用されます。
- swdesc_option POST_ACTION=reboot: デフォルトの状態に戻します。アップデートの後に再起動します。
- swdesc_option NOTIFY_STARTING_CMD="command", swdesc_option NOTIFY_SUCCESS_CMD="command", swdesc_option NOTIFY_FAIL_CMD="command": アップデートをインストール中、成功した場合と失敗した場合に実行されるコマンドです。

  コマンドを実行する事で、アプリケーションやユーザーにアップデートを知らせることができます。

  LED で知らせる例を /usr/share/mkswu/examples/enable_notify_led.desc に用意してあります。

6.4.11. desc ファイル設定例

6.4.11.1. 例: sshd を有効にする

/usr/share/mkswu/examples/enable_sshd.desc を参考にします。

desc ファイルを編集する必要がありませんが自分の公開鍵を指定された場所に配置してください。

```
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/enable_sshd* .
[ATDE ~/mkswu]$ cat enable_sshd.desc
swdesc_option component=extra_os.sshd version=1

# add your public key in enable_sshd/root/.ssh/authorized_keys
if [ -z "$SWDESC_TEST" ]; then
    grep -qE '^ssh-' enable_sshd/root/.ssh/authorized_keys ¥
        || error "Add your keys in enable_sshd/root/.ssh/authorized_keys"
fi
swdesc_files --dest /root enable_sshd/root ❶

swdesc_command "ssh-keygen -A" ¥ ❷
        "rc-update add sshd" ❸
[ATDE ~/mkswu]$ cp ~/.ssh/id_rsa.pub ¥
                        enable_sshd/root/.ssh/authorized_keys ❹
[ATDE ~/mkswu]$ mkswu enable_sshd.desc ❺
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
enable_sshd.swu を作成しました。
```

❶ 自分の公開鍵を転送します。デフォルトのオプションなので enable_sshd/root ディレクトリの中身をこのまま /root に転送されます。

❷ 再起動する度に新しいサーバーの鍵が変わらないように、アップデートの時に一回作成します。

❸ サービスを有効にします。

❹ 自分の公開鍵を指定された場所に配置します。

❺ イメージを作成します。パスワードは証明鍵のパスワードです。

6.4.11.2. 例: Armadillo Base OS アップデート

ここでは、「6.30. Armadillo のソフトウェアをビルドする」でメインシステム向けのビルドで作成したファイルを使用します。

/usr/share/mkswu/examples/OS_update.desc を参考にします。

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/OS_update.desc update-[VERSION].desc
[ATDE ~/mkswu]$ vi update-[VERSION].desc
# uboot image can be generated with atmark imx-boot script
swdesc_uboot imx-boot_armadillo_x2 ❶

# base OS is a tar that will be extracted on a blank filesystem,
# after copying just a few key config files.
#
# OS updates are only installed if version is greater than previous update
# so if you install your own updates atmark-techno provided Armadillo Base OS
# updates might not get installed
swdesc_tar "baseos-x2-[VERSION].tar.zst" ¥ ❷
           --version base_os [VERSION] ❸
[ATDE ~/mkswu]$ mkswu update-[VERSION].desc ❹
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
update-[VERSION].swu を作成しました。
```

❶ imx-boot でビルドしたイメージを使います。

❷ build-rootfs でビルドしたイメージを使います。

❸ バージョンが上がるときにしかインストールされませんので、現在の/etc/sw-versions を確認して適切に設定してください。

❹ イメージを作成します。パスワードは証明鍵の時のパスワードです。

6.4.11.3. 例: swupdate_preserve_files で Linux カーネル以外の Armadillo-IoT ゲートウェイ A6E 向けのイメージをインストールする方法

Armadillo-IoT ゲートウェイ A6E 向けのアップデートイメージに Linux カーネルが含まれています。

swupdate_preserve_files を使って、以下のコマンドでインストール後に現在のカーネルをコピーして更新させないようにします。

```
[armadillo ~]# echo 'POST /boot' >> /etc/swupdate_preserve_files
[armadillo ~]# echo 'POST /lib/modules' >> /etc/swupdate_preserve_files ❶
[armadillo ~]# persist_file /etc/swupdate_preserve_files ❷
```

❶ swupdate_preserve_files に /boot と /lib/modules を保存するように追加します。

❷ 変更した設定ファイルを保存します



/usr/share/mkswu/examples/kernel_update*.desc のように update_preserve_files.sh のヘルパーで、パスを自動的に /etc/swupdate_preserve_files に追加することができます。

```
[ATDE ~/mkswu]$ cat example.desc
swdesc_script "$SCRIPT_DIR/examples/update_preserve_files.sh" -- ¥ ❶
        "POST /boot" ¥
        "POST /lib/modules"
```

❶ スクリプトの内容確認する場合は /usr/share/mkswu/examples/update_preserve_files.sh を参照してください。



Armadillo Base OS のカーネルを再び使用したい場合は同じスクリプトの --del オプションで行を削除することができます。

```
[ATDE ~/mkswu]$ cat example.desc
swdesc_script "$SCRIPT_DIR/examples/update_preserve_files.sh" -- ¥
        --del "POST /boot" "POST /lib/modules"
```

6.5. swupdate_preserve_files について

extra_os のアップデートで rootfs にファイルを配置することができますが、次の OS アップデートの際に削除される可能性があります。

デフォルトでは、 /etc/atmark と、 swupdate 、 sshd やネットワークの設定を保存しますがそれ以外はコピーされてません。

そうでないファイルを更新する際には /etc/swupdate_preserve_files に記載します。「6.4.11.3. 例: swupdate_preserve_files で Linux カーネル以外の Armadillo-IoT ゲートウェイ A6E 向けのイメージをインストールする方法」 を参考にしてください。

コピーのタイミングによって、以下のどれかを使用してください：

1. 単にファイルを記載する

   この場合、アップデートする前にファイルをコピーします。 baseos のイメージと同じ swu にアップデートしたいファイルを記載していても、このファイルが Armadillo Base OS に含まれないのであれば問題なくアップデートできます。

   例: echo "/root/.profile" >> /etc/swupdate_preserve_files

2. POST のキーワードの後に記載する

この場合、アップデートの最後でコピーします。 Armadillo Base OS に含まれてるファイルであれば、インストール前にコピーしても保存されないのでコピーのタイミングをずらします。

そのコピーが最後に行われるので、同じアップデートでファイルの変更ができません。アップデートを別けて、 baseos のイメージをインストールしてからこのファイルを更新することができます。

例: echo "POST /etc/conf.d/podman-atmark" >> /etc/swupdate_preserve_files

6.6. SWU イメージの内容の確認

mkswu --show [file.swu] で SWU イメージの内容を確認することができます。

出力は desc ファイルに似ていますが、そのまま desc ファイルとして利用できませんので確認用としてお使いください。

```
[ATDE ~/mkswu]$ mkswu --show enable_sshd.swu
enable_sshd.swu

# built with mkswu 4.1

swdesc_files --dest /root enable_sshd/root
  --version extra_os.sshd 1
  (encrypted)

swdesc_command ssh-keygen -A && rc-update add sshd default
  --version extra_os.sshd 1
```

6.7. SWUpdate と暗号化について

mkswu --init の時に暗号化を有効にする場合は AES でファイルを暗号化します。

現在使われてる SWUpdate の暗号化はコマンドやメタデータを含む sw-description ファイルは暗号化されてません。そのため、通信の暗号化（HTTPS で送信するなど）を使うことを推奨します。

6.8. SWUpdate の署名鍵と証明書の更新

mkswu で SWU イメージを生成する際に SWU イメージ内の sw-description という命令ファイルを ATDE にある署名鍵と証明書を用いて署名します。swupdate を実行する際には、署名に使用した証明書が /etc/swupdate.pem に含まれているかを確認します。

その署名を確認できなかった場合、SWU イメージをインストールできないので、Armadillo にある証明書を管理しなければなりません。

また、暗号鍵管理のガイドラインとしては定期的に鍵を交換することが強く推奨されています。mkswu --init を実行した際に 1 つだけ署名鍵と証明書を生成しましたが、ここでは他の署名鍵および証明書の生成と Armadillo 側の管理の方法について説明します。

6.8.1. 署名鍵と証明書の追加

署名鍵と証明書の生成は以下の様に、 mkswu --genkey で行います。

```
[ATDE ~]$ mkswu --genkey
/home/atmark/mkswu/swupdate.key はすでに存在します。新しい鍵を作成しますか？ [Y/n] ❶
```

```
証明書のコモンネーム(一般名)を入力してください: [COMMON_NAME] ❷
署名鍵 /home/atmark/mkswu/swupdate-2.key と証明書 swupdate-2.pem を作成します。
Generating an EC private key
writing new private key to '/home/atmark/mkswu/swupdate-2.key.tmp'
Enter PEM pass phrase: ❸
Verifying - Enter PEM pass phrase:
 -----
/home/atmark/mkswu/swupdate-2.pem をコンフィグファイルに追加します。
/home/atmark/mkswu/swupdate-2.pem が次のアップデートをインストールするときに転送されます。
インストールされてから現在の鍵を /home/atmark/mkswu/mkswu.conf から外してください。
```

図 6.30 mkswu --genkey で署名鍵と証明書を追加する

❶ Enter キーを押下します。

❷ [COMMON_NAME] には会社や製品が分かる任意の名称を入力してください。任意ではありますが、一つ目の鍵と違う名前にすることを推奨します。

❸ 署名鍵を保護するパスフレーズを 2 回入力します。

このコマンドを実行すると以下の文字列が ~/mkswu/mkswu.conf に追加されます。

```
# extra swupdate certificate. Remove the old one and use new
# PRIVKEY after having installed an update with this first
PUBKEY="$PUBKEY,$CONFIG_DIR/swupdate-2.pem" ❶
# remove "NEW_" to use
NEW_PRIVKEY="$CONFIG_DIR/swupdate-2.key" ❷
# This controls if we should update certificates on device, and can be
# removed once all devices have been updated to only allow new certificate
UPDATE_CERTS=yes
```

図 6.31 mkswu --genkey により mkswu.conf に追加された内容

❶ PUBKEY の値に生成した証明書のパスが追加されます。UPDATE_CERTS=yes を設定して生成した SWU イメージは、PUBKEY の値にリストされている証明書を全て Armadillo にインストールします。PUBKEY にリストされてない証明書、またはアットマークテクノ側で管理している証明書以外は全て削除されます。

❷ 新しい署名鍵のパスです。この段階では未使用になります。

この状態で任意 (POST_ACTION=container 以外) の SWU イメージを生成し、インストールすると Armadillo の /etc/swupdate.pem に PUBKEY にリストされている両方の証明書がインストールされます。Armadillo にインストールされている鍵は、abos-ctrl certificates list で確認できます：

```
[armadillo ~]# abos-ctrl certificates list
- atmark-2
- atmark-3
- swupdate-2.pem: [mkswu --genkey で入力した COMMON NAME] ❶
- swupdate.pem: [mkswu --init で入力した COMMON NAME] ❷
```

図 6.32 新しい証明書が Armadillo に追加されていることを確認する

❶ 追加した証明書のコモンネーム

❷ mkswu --init 時に生成した証明証のコモンネーム。当時の mkswu のバージョンによっては swupdate.pem が記載されてない可能性があります。

この状態で新しい鍵を使えるようになります。

6.8.2. 署名鍵と証明書の削除

上記のように Armadillo に複数の署名鍵を使用できる状態になった場合は証明に使う署名鍵と証明書を切り替えることができます。

mkswu.conf の PUBKEY の値から一つ目の値を削除し、 PRIVKEY に新しい値を設定し、必要であれば UPDATE_CERTS=yes を記述します。

```
[ATDE ~] tail -n 3 ~/mkswu/mkswu.conf
PUBKEY="$CONFIG_DIR/swupdate-2.pem"
PRIVKEY="$CONFIG_DIR/swupdate-2.key"
UPDATE_CERTS=yes
```

図 6.33 署名鍵と証明書を削除する設定

署名鍵の追加と同じく、この状態で SWU イメージを生成し Armadillo にインストールすると前の証明書が削除されます。

こちらも abos-ctrl certificates list コマンドで確認可能です。

```
[armadillo ~]# abos-ctrl certificates list
- atmark-2
- atmark-3
- swupdate-2.pem: [mkswu --genkey で入力した COMMON NAME]
```

図 6.34 証明書がインストールされていることを確認する

6.9. コンテナの概要と操作方法を知る

Armadillo Base OS において、ユーザーアプリケーションは基本的にコンテナ内で実行されます。「3. 開発編」で紹介した開発手順では、基本的に SWUpdate を使用してコンテナを生成・実行していました。

以下では、より自由度の高いコンテナの操作のためにコマンドラインからの操作方法について紹介します。

6.9.1. Podman - コンテナ仮想化ソフトウェアとは

コンテナとはホスト OS 上に展開される仮想的なユーザ空間のことです。コンテナを使用することで複数の Armadillo-IoT ゲートウェイ A6E でも同一の環境がすぐに再現できます。ゲスト OS を必要としない仮想化であるため、アプリケーションの起動が素早いという特徴があります。

Podman とはこのようなコンテナを管理するためのソフトウェアであり、使用方法はコンテナ管理ソフトウェアの 1 つである Docker と互換性があります。

6.9.2. コンテナの基本的な操作

この章では、コンテナ仮想化ソフトウェアの 1 つである Podman の基本的な使い方について説明します。Armadillo-IoT ゲートウェイ A6E で実行させたいアプリケーションとその実行環境自体を 1 つ

の Podman イメージとして扱うことで、複数の Armadillo-IoT ゲートウェイ A6E がある場合でも、全てのボード上で同一の環境を再現させることが可能となります。

この章全体を通して、イメージの公開・共有サービスである Docker Hub [https://hub.docker.com] から取得した、Alpine Linux のイメージを使って説明します。

6.9.2.1. イメージからコンテナを作成する

イメージからコンテナを作成するためには、`podman_start` コマンドを実行します。podman や docker にすでに詳しいかたは `podman run` コマンドでも実行できますが、ここでは 「6.9.4. コンテナ起動設定ファイルを作成する」 で紹介するコンテナの自動起動の準備も重ねて `podman_start` を使います。イメージは Docker Hub [https://hub.docker.com] から自動的に取得されます。ここでは、簡単な例として "ls /" コマンドを実行するコンテナを作成します。

```
[armadillo ~]# vi /etc/atmark/containers/my_container.conf ❶
set_image docker.io/alpine
set_command ls /
[armadillo ~]# podman pull docker.io/alpine ❷
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
:（省略）
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start my_container ❸
Starting 'my_container'
b141e899b5ef7c9ec5434bda8f6a83d3e6bfc94f74bfb5dcef2a22041c71fdbf
[armadillo ~]# podman logs my_container ❹
bin
dev
:（省略）
usr
var
[armadillo ~]#
```

図 6.35 コンテナを作成する実行例

❶ コンテナのコンフィグを作成します。このファイルでは、コンテナのイメージやコマンド、デバイスへのアクセス権限を設定します。詳しい設定の説明には 「6.9.4. コンテナ起動設定ファイルを作成する」 を参照ください。

❷ コンテナのイメージを取得します。イメージが Armadillo に置いてない場合は「Error: docker.io/alpine: image not known」の様なエラーで失敗します。

❸ コンテナを起動します。これは Armadillo 起動時に自動的に起動されるコンテナと同じものになります。自動起動が不要な場合には `set_autostart no` で無効化できます。

❹ `podman logs` コマンドで出力を確認します。

"ls /" を実行するだけの "my_container" という名前のコンテナが作成されました。コンテナが作成されると同時に "ls /" が実行され、その結果がログに残ります。ここで表示されているのは、コンテナ内部の "/" ディレクトリのフォルダの一覧です。



コンフィグファイルの直接な変更と podman pull によるコンテナの取得はデフォルト状態ではメモリ上でしか保存されません。

ファイルは persist_file で必ず保存し、コンテナイメージは abos-ctrl podman-storage --disk で podman のストレージを eMMC に切り替えるか abos-ctrl podman-rw で一時的に eMMC に保存してください。

運用中の Armadillo には直接に変更をせず、 SWUpdate でアップデートしてください。

コンフィグファイルを保存して、 set_autostart no を設定しない場合は自動起動します。



podman_start でコンテナが正しく起動できない場合は podman_start -v <my_container> で podman run のコマンドを確認し、 podman logs <my_container> で出力を確認してください。

6.9.2.2. イメージ一覧を表示する

コンテナを作成するためのイメージは、イメージ一覧を表示する podman images コマンドで確認できます。

```
[armadillo ~]# podman images
REPOSITORY                TAG     IMAGE ID      CREATED      SIZE
docker.io/library/alpine  latest  9c74a18b2325  2 weeks ago  4.09 MB
```

図 6.36 イメージ一覧の表示実行例

podman images コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman images --help
```

図 6.37 podman images --help の実行例

6.9.2.3. コンテナー覧を表示する

作成済みコンテナー覧を表示するためには podman ps コマンドを実行します。

```
[armadillo ~]# podman ps -a
CONTAINER ID  IMAGE                           COMMAND     CREATED        STATUS                    ⏎
PORTS     NAMES
```

```
d6de5881b5fb  docker.io/library/alpine:latest  ls /       12 minutes ago  Exited (0) 11 minutes ⏎
ago          my_container
```

図 6.38 コンテナ一覧の表示実行例

一覧表示により、コンテナ名やコンテナ ID を確認することができます。-a オプションを付けない場合は、動作中のコンテナのみ表示されます。podman ps コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman ps --help
```

図 6.39 podman ps --help の実行例

6.9.2.4. コンテナを起動する

作成済みのコンテナを起動するためには podman start コマンドを実行します。

```
[armadillo ~]# podman start my_container
podman start my_container
[ 3119.081068] IPv6: ADDRCONF(NETDEV_CHANGE): vethe172e161: link becomes ready
[ 3119.088214] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 3119.094812] cni-podman0: port 1(vethe172e161) entered blocking state
[ 3119.101231] cni-podman0: port 1(vethe172e161) entered disabled state
[ 3119.107745] device vethe172e161 entered promiscuous mode
[ 3119.113185] cni-podman0: port 1(vethe172e161) entered blocking state
[ 3119.119546] cni-podman0: port 1(vethe172e161) entered forwarding state
my_container
[ 3119.620731] cni-podman0: port 1(vethe172e161) entered disabled state
[ 3119.627696] device vethe172e161 left promiscuous mode
[ 3119.632762] cni-podman0: port 1(vethe172e161) entered disabled state
```

図 6.40 コンテナを起動する実行例

-a オプションを与えると、コンテナ内で実行されたアプリケーションの出力を確認できます。

```
[armadillo ~]# podman start -a my_container
[ 3150.303962] IPv6: ADDRCONF(NETDEV_CHANGE): vetha9ef8f8e: link becomes ready
[ 3150.311106] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 3150.317703] cni-podman0: port 1(vetha9ef8f8e) entered blocking state
[ 3150.324139] cni-podman0: port 1(vetha9ef8f8e) entered disabled state
[ 3150.330687] device vetha9ef8f8e entered promiscuous mode
[ 3150.336085] cni-podman0: port 1(vetha9ef8f8e) entered blocking state
[ 3150.342443] cni-podman0: port 1(vetha9ef8f8e) entered forwarding state
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
[ 3150.804164] cni-podman0: port 1(vetha9ef8f8e) entered disabled state
[ 3150.811249] device vetha9ef8f8e left promiscuous mode
[ 3150.816349] cni-podman0: port 1(vetha9ef8f8e) entered disabled state
```

図 6.41 コンテナを起動する実行例(a オプション付与)

ここで起動している my_container は、起動時に "ls /" を実行するようになっているので、その結果が出力されます。podman start コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman start --help
```

図 6.42 podman start --help 実行例

6.9.2.5. コンテナを停止する

動作中のコンテナを停止するためには podman stop コマンドを実行します。

```
[armadillo ~]# podman stop my_container
my_container
```

図 6.43 コンテナを停止する実行例

podman stop コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman stop --help
```

図 6.44 podman stop --help 実行例

6.9.2.6. コンテナの変更を保存する

コンテナに対して変更が行われた状態で、そのままコンテナを停止してしまうと変更が失なわれてしまいます。

変更を保存するには二つの方法があります。

1. podman commit コマンドで保存する。

   ```
   [armadillo ~]# podman commit my_container image_name:latest
   Getting image source signatures
   Copying blob f4ff586c6680 skipped: already exists
   Copying blob 3ae0874b0177 skipped: already exists
   Copying blob ea59ffe27343 done
   Copying config 9ca3c55246 done
   Writing manifest to image destination
   Storing signatures
   9ca3c55246eaac267a71731bad6bfe4b0124afcdd2b80c4f730c46aae17a88f3
   ```

   図 6.45 my_container を保存する例

   podman commit で保存する度に、変更が行なわれた差分が保存されます。繰り返し差分を保存すると、イメージサイズが大きくなってしまいます。ストレージ容量が不足する場合は、ベースとなる OS のイメージから作り直してください。

2. 「3.3.5.1. 電源を切っても保持されるディレクトリ(ユーザーデータディレクトリ)」を使用する。

   podman_start の add_volumes コマンドでコンテナに Armadillo Base OS のディレクトリをコンテナで使うことができます。

保存するデータの性質によって、保存先を選択してください。

1. /var/app/volumes/myvolume: アップデートした場合はコピーされません。ログやデータベースなど、アプリケーションが作成し続けるようなデータの保存に向いています。

2. myvolume か /var/app/rollback/volumes/myvolume: アップデートの際にコピーしてアップデートを行うので、アップデート中でも安全に使いつづけます。アプリケーションと一緒にアップデートするようなデータの保存に向いています。

6.9.2.7. コンテナの自動作成やアップデート

podman run, podman commit でコンテナを作成できますが、定期的にアップデートをする際にはコンテナの作成やアップデートを自動化できると便利です。

これを実現するために、Dockerfile と podman build を使います。この手順は Armadillo で実行可能です。

1. イメージを docker.io のイメージから作りなおします

```
[armadillo ~/podman-build]# cat Dockerfile
FROM docker.io/arm64v8/alpine:latest

# update & install dependencies (example: usbutils)
RUN apk upgrade && apk add usbutils && rm -f /var/cache/apk/*

# copy our application and set it to run on start
COPY my_application /my_application
ENTRYPOINT /my_application

[armadillo ~/podman-build]# podman build -t my_image:1 -t my_image:latest .
STEP 1: FROM docker.io/arm64v8/alpine:latest
STEP 2: RUN apk upgrade && apk add usbutils && rm -f /var/cache/apk/*
--> 234bf79175e
STEP 3: COPY my_application /my_application
--> 05ab31bb278
STEP 4: ENTRYPOINT /my_application
STEP 5: COMMIT my_image:latest
--> 590e3ba6d55
Successfully tagged localhost/my_image:1
Successfully tagged localhost/my_image:latest
590e3ba6d55f3e29bdef158d7283e9c4f7515567b2d3f978cfab2510dc02376b

[armadillo ~/podman-build]# podman save my_image:latest -o my_image_1.tar
```

図 6.46 podman build の実行例

2. イメージを前のバージョンからアップデートします

```
[armadillo ~/podman-build-update]# cat Dockerfile
FROM localhost/my_image:latest

# update OS packages
RUN apk upgrade --no-cache

# update application
```

```
COPY my_application /my_application
[armadillo ~/podman-build-update]# podman build -t my_image:2 -t my_image:latest .
STEP 1: FROM localhost/my_image:latest
STEP 2: RUN apk upgrade --no-cache
--> cf1dc0d7296
STEP 3: COPY my_application /my_application
STEP 4: COMMIT my_image:latest
--> 9e9d9366072
Successfully tagged localhost/my_image:2
Successfully tagged localhost/my_image:latest
9e9d9366072751007b2e70544d76c46b95a7a5a02df658ef0fa3f7dcccf8850a

[armadillo ~/podman-build-update]# podman save -o my_image_2.tar my_image:2
```

図 6.47 podman build でのアップデートの実行例

この場合、 podman_partial_image コマンドを使って、差分だけをインストールすることもできます。

```
[armadillo ~/podman-build-update]# podman_partial_image -b my_image:1 ¥
        -o my_image_2_partial.tar my_image:2

[armadillo ~/podman-build-update]# ls -lh
-rw-r--r-- 1 root root   88 Dec 21 15:24 Dockerfile
-rw-r--r-- 1 root root 9.4M Dec 21 15:26 my_image_1.tar
-rw-r--r-- 1 root root 9.4M Dec 21 15:26 my_image_2.tar
-rw-r--r-- 1 root root  51K Dec 21 15:26 my_image_2_partial.tar
```

作成した .tar アーカイブは 「6.4. mkswu の .desc ファイルを編集する」 の swdesc_embed_container と swdesc_usb_container で使えます。

6.9.2.8. コンテナを削除する

作成済みコンテナを削除する場合は podman rm コマンドを実行します。

```
[armadillo ~]# podman rm my_container
d6de5881b5fb973227b84d1d74abf269ac3183aad7e18b7a9d85208632641d94
[armadillo ~]# podman ps -a
CONTAINER ID  IMAGE                            COMMAND    CREATED         STATUS          ⏎
PORTS    NAMES
```

図 6.48 コンテナを削除する実行例

podman ps コマンドの出力結果より、コンテナが削除されていることが確認できます。podman rm コマンドの詳細は --help オプションで確認できます。

1. podman rm --help 実行例

```
[armadillo ~]# podman rm --help
```

6.9.2.9. イメージを削除する

podman のイメージを削除するには podman rmi コマンドを実行します。イメージを削除するためには、そのイメージから作成したコンテナを先に削除しておく必要があります。podman rmi コマンドにはイメージ ID を指定する必要があるため、podman images コマンドで確認します。

```
[armadillo ~]# podman rm my_container
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE
docker.io/library/alpine  latest      02480aeb44d7  2 weeks ago  5.62 MB
[armadillo ~]# podman rmi 02480aeb44d7
Untagged: docker.io/library/alpine:latest
Deleted: 02480aeb44d78f1a44b8791af7edf7d6e1b18707397a1dfb3ff4f21c5ce4a44f
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE
```

図 6.49 イメージを削除する実行例

podman images コマンドの出力結果より、コンテナが削除されていることが確認できます。podman rmi コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman rmi --help
```

図 6.50 podman rmi --help 実行例



SWU で転送されたイメージは podman images で Read-Only として表示されますので、podman rmi を実行するとエラーとなります。その場合は abos-ctrl podman-rw rmi をご使用ください。 abos-ctrl podman-rw については 「6.9.2.16. イメージを eMMC に保存する」 を参照してください。

```
[armadillo ~]# podman images
REPOSITORY              TAG       IMAGE ID     CREATED     SIZE      R/O
docker.io/library/alpine  latest     02480aeb44d7  2 weeks ago  5.62
MB    true
[armadillo ~]# podman rmi docker.io/alpine
Error: cannot remove read-only image
"02480aeb44d78f1a44b8791af7edf7d6e1b18707397a1dfb3ff4f21c5ce4a44f"
[armadillo ~]# abos-ctrl podman-rw rmi docker.io/alpine
Untagged: docker.io/library/alpine:latest
Deleted:
02480aeb44d78f1a44b8791af7edf7d6e1b18707397a1dfb3ff4f21c5ce4a44f
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE
```

図 6.51 Read-Only のイメージを削除する実行例

6.9.2.10. 実行中のコンテナに接続する

実行中のコンテナに接続し、コンテナ内で指定したコマンドを実行するには podman exec コマンドを実行します。podman exec コマンドでコンテナ内部のシェルを起動すると、コンテナ内部を操作できるよ

うになります。ここでは、sleep infinity コマンドを実行して待ち続けるだけのコンテナを作成し、そのコンテナに対して podman exec コマンドでシェルを起動する例を示します。

```
[armadillo ~]# vi /etc/atmark/containers/sleep_container.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# podman_start sleep_container
Starting 'test'
f62e7a666d7156d261905c8406c72fc271534fa29e69771c76f4f6660a2da41a
[armadillo ~]# podman exec -it sleep_container sh
[container ~]# ps
PID   USER     TIME  COMMAND
    1 root      0:00 /run/podman-init -- sleep infinity
    2 root      0:00 sleep infinity
    3 root      0:00 sh
    4 root      0:00 ps
```

図 6.52 コンテナ内部のシェルを起動する実行例

podman_start コマンドでコンテナを作成し、その後作成したコンテナ内で sh を実行しています。sh を実行すると、コンテナ内のプロンプトが表示されコンテナ内部を操作できるようになります。上記ではコンテナ内で、ps コマンドを実行しています。コンテナ作成時に実行した sleep と podman exec で実行した sh がプロセスとして存在していることが確認できます。

コンテナ内のシェルから抜ける時は exit コマンドを実行します。

```
[container ~]# exit
```

図 6.53 コンテナ内部のシェルから抜ける実行例

podman exec コマンドから抜けても、コンテナがまだ実行中です。コンテナを停止したい場合は podman stop sleep_container か podman kill sleep_container で停止して podman rm sleep_container でそのコンテナを削除してください。

podman exec コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman exec --help
```

図 6.54 podman exec --help 実行例

6.9.2.11. コンテナ間で通信をする

複数のコンテナを実行している環境で、それらのコンテナ間で通信を行う方法を示します。これにより、例えば SQL サーバを実行しているコンテナに対し別のコンテナから接続するといった使い方ができます。

コンテナには作成した時点でローカル IP アドレスが割り当てられるので、コンテナの名前かその IP アドレスで通信を行うことができます。

準備として、2 つのコンテナを作成します。

```
[armadillo ~]# vi /etc/atmark/containers/my_container_1.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# vi /etc/atmark/containers/my_container_2.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# podman_start my_container_1 my_container_2
Starting 'my_container_1'
cbe0802f4e2d2fec88f4e300dabeba3b48865359dc02cbd99375b1b38c2c28eb
Starting 'my_container_2'
5e645f5e40fc096ad0bea323a00bebebbda4bd825a5e8d12103f752d8868692e
```

図 6.55 コンテナを作成する実行例

コンテナに割り当てられた IP アドレスを確認するには podman inspect コマンドを実行します。

```
[armadillo ~]# podman inspect --format='{{.NetworkSettings.IPAddress}}' my_container_1
10.88.0.108
[armadillo ~]# podman inspect --format='{{.NetworkSettings.IPAddress}}' my_container_2
10.88.0.109
```

図 6.56 コンテナの IP アドレスを確認する実行例

これらの IP アドレスを使って、一方のコンテナからもう一方のコンテナへ対し ping コマンドで疎通確認を行うことができます。

```
[armadillo ~]# podman exec -it my_container_1 sh
[container ~]# ping -c 2 my_container_2
PING my_container_2 (10.88.0.109): 56 data bytes
64 bytes from 10.88.0.109: seq=0 ttl=42 time=0.144 ms
64 bytes from 10.88.0.109: seq=1 ttl=42 time=0.210 ms

--- my_container_2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.144/0.177/0.210 ms
[container ~]# ping -c 2 10.88.0.109
PING 10.88.0.109 (10.88.0.109): 56 data bytes
64 bytes from 10.88.0.109: seq=0 ttl=42 time=0.140 ms
64 bytes from 10.88.0.109: seq=1 ttl=42 time=0.138 ms

--- 10.88.0.109 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.138/0.139/0.140 ms
```

図 6.57 ping コマンドによるコンテナ間の疎通確認実行例

このように、my_container_1(10.88.0.108) から my_container_2(10.88.0.109) への通信が確認できます。

6.9.2.12. pod でコンテナのネットワークネームスペースを共有する

podman_start で pod 機能を使うことができます。

pod を使うことで、複数のコンテナが同じネットワークネームスペースを共有することができます。同じ pod の中のコンテナが IP の場合 localhost で、 unix socket の場合 abstract path で相互に接続することができます。

```
[armadillo ~]# cat /etc/atmark/containers/mypod.conf
set_type pod
add_ports 80:80

[armadillo ~]# cat /etc/atmark/containers/nginx.conf
set_image docker.io/library/nginx:alpine
set_readonly no
set_pod mypod

[armadillo ~]# podman ps
CONTAINER ID  IMAGE                           COMMAND               CREATED      STATUS                  ⏎
PORTS               NAMES
0cdb0597b610  localhost/podman-pause:4.3.1-1683096588              2 hours ago  Up 2 hours ago          ⏎
0.0.0.0:80->80/tcp  5ba7d996f673-infra
3292e5e714a2  docker.io/library/nginx:alpine  nginx -g daemon o...  2 hours ago  Up 2 hours ago          ⏎
0.0.0.0:80->80/tcp  nginx
```

図 6.58 pod を使うコンテナを自動起動するための設定例

コンテナと同じく、 /etc/atmark/containers/[NAME].conf ファイルを作って、 set_type pod を設定することで pod を作成します。

pod を使う時にコンテナの設定ファイルに set_pod [NAME] の設定を追加します。

ネットワークネームスペースは pod を作成するときに必要なため、 ports, network と ip の設定は pod のコンフィグファイルに入れなければなりません。

必要であれば、他の podman pod create のオプションを add_args で設定することができます。

.conf ファイルで使用できる各種パラメータについては、「6.9.4. コンテナ起動設定ファイルを作成する」を参照してください。

6.9.2.13. network の作成

podman_start で podman の network も作成できます。

デフォルトの 10.88.0.0/16 が使えない場合、あるいはコンテナ同士で接続できないようにしたい場合は使ってください。

```
[armadillo ~]# cat /etc/atmark/containers/mynetwork.conf
set_type network
set_subnet 198.51.100.0/24

[armadillo ~]# cat /etc/atmark/containers/nginx.conf
set_image docker.io/library/nginx:alpine
add_ports 80:80
set_ip 198.51.100.10
set_network mynetwork

[armadillo ~]# podman ps
CONTAINER ID  IMAGE                           COMMAND               CREATED      STATUS                  ⏎
```

```
PORTS                NAMES
3292e5e714a2  docker.io/library/nginx:alpine  nginx -g daemon o...  2 hours ago  Up 2 hours ago
0.0.0.0:80->80/tcp  nginx
```

図 6.59 network を使うコンテナを自動起動するための設定例

コンテナと同じく、 /etc/atmark/containers/[NAME].conf ファイルを作って、 set_type network を設定することで network を作成します。

そのネットワークを使う時にコンテナの設定ファイルに set_network [NAME] の設定をいれます。

ネットワークのサブネットは set_subnet [SUBNET] で設定します。この設定は set_type network の後しか使えませんので、set_type はファイルの最初のところに使ってください

他の podman network create のオプションが必要であれば、 add_args で設定することができます。

.conf ファイルで使用できる各種パラメータについては、「6.9.4. コンテナ起動設定ファイルを作成する」を参照してください。

6.9.2.14. コンテナからのコンテナ管理

podman では REST API による管理アクセスも可能です。

自分のコンテナから他のコンテナの管理が必要な場合に、ホストの podman サービスを有効にして、コンテナに /run/podman をボリュームマウントすれば podman --remote で管理できます。



コンテナの設定によって podman の socket へのパスが自動設定されない場合もあります。podman --remote でエラーが発生した場合に CONTAINER_HOST=unix:/path/to/podman.sock で socket へのパスを設定してください。

Armadillo のホスト側の udev rules からコンテナを起動する場合は podman_start 等を直接実行すると udev の子プロセス管理によってコンテナが停止されますので、その場合はサービスを有効にし、podman_start --create <container> コマンドまたは set_autostart create の設定でコンテナを生成した上 podman --remote start <container> で起動してください。

6.9.2.15. リモートリポジトリにコンテナを送信する

1. イメージをリモートリポジトリに送信する：

```
[armadillo ~]$ podman image push <localimage> docker://<registry>/<remoteimage>:<tag>
```

2. set_pull always を設定しないかぎり、SWUpdate でダウンロードの命令を送らないとアップデートを行いません。

   (mkswu については「5.4. Armadillo のソフトウェアをアップデートする」を参考にしてください)

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/pull_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
```

```
[ATDE ~/mkswu]$ cat pull_container_nginx.desc
swdesc_option version=1

swdesc_pull_container "docker.io/nginx:alpine"
swdesc_files --extra-os nginx_start
[ATDE ~/mkswu]$ mkswu pull_container_nginx.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
pull_container_nginx.swu を作成しました。
```

6.9.2.16. イメージを eMMC に保存する

Armadillo Base OS のデフォルトでは、Podman のデータは tmpfs に保存されます。

起動時にコンテナを起動するにはイメージを eMMC に書き込む必要があります。開発が終わって運用の場合は 「6.9.2.17. イメージを SWUpdate で転送する」 でコンテナのイメージを転送します。この場合は読み取り専用の app パーティションのサブボリュームに展開します。

開発の時に以下の abos-ctrl podman-rw か abos-ctrl podman-storage --disk のコマンドを使って直接にイメージを編集することができます。



ここで紹介する内容はコンテナのイメージの管理の説明です。データベース等のコンテナから書き込みが必要な場合には 「6.9.2.6. コンテナの変更を保存する」 にあるボリュームの説明を参照してください。

- abos-ctrl podman-rw

`abos-ctrl podman-rw` を使えば、read-only になっているイメージを扱う事ができます。

```
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED           SIZE        R/O
[armadillo ~]# mount /dev/sda1 /mnt
[armadillo ~]# abos-ctrl podman-rw load -i /mnt/at-debian-image.tar
Getting image source signatures
Copying blob 63c098a71e7b done
Copying blob 837e73dd4d20 done
Copying blob a25086e65f63 done
Copying config b5a30f8581 done
Writing manifest to image destination
Storing signatures
Loaded image(s): localhost/at-debian-image:latest
[armadillo ~]# podman image list
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE        R/O
localhost/at-debian-image  latest      b5a30f8581cc  2 hours ago  233 MB      true
```

図 6.60 abos-ctrl podman-rw の実行例

- abos-ctrl podman-storage

`abos-ctrl podman-storage` はメモリとディスクの切り替えの他に、読み書きストレージから読み取り専用ストレージへのコピーもできます。

```
[armadillo ~]# podman pull docker.io/alpine ❶
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Copying blob f97344484467 done
Copying config 3d81c46cd8 done
Writing manifest to image destination
Storing signatures
3d81c46cd8756ddb6db9ec36fa06a6fb71c287fb265232ba516739dc67a5f07d
[armadillo ~]# abos-ctrl podman-storage ❷
List of images configured on development storage:
REPOSITORY                TAG         IMAGE ID      CREATED     SIZE
docker.io/library/alpine  latest      3d81c46cd875  3 days ago  5.56 MB

What should we do? ([C]opy (default), [N]othing, [D]elete)
copy ❸
Create a snapshot of '/mnt/boot_1/containers_storage' in '/mnt/new_storage'
Getting image source signatures
Copying blob 8ec3165d6e61 done
Copying config 4a49b68e7c done
Writing manifest to image destination
Storing signatures
Delete subvolume (no-commit): '/mnt/new_storage'
Merging development images to readonly storage succeeded
Feel free to adjust the result with abos-ctrl podman-rw commands

Now freeing up original data...
Podman is in tmpfs mode ❹
[armadillo ~]# podman image list ❺
REPOSITORY                TAG         IMAGE ID      CREATED     SIZE        R/O
docker.io/library/alpine  latest      3d81c46cd875  3 days ago  5.56 MB     true
```

図 6.61 abos-ctrl podman-storage のイメージコピー例

❶ イメージを書き込み可能ストレージに取得します。

❷ abos-ctrl podman-storage をオプション無しで実行します。

❸ 書き込み可能ストレージにイメージがある場合に対応を聞かれます。今回はコピー（copy）します。

❹ abos-ctrl podman-storage にオプションを指定しなかったので、ストレージが tmpfs のままになります。すでに --disk で切り替えた場合にディスクのままでも可能です。

❺ コピーされたイメージを確認します。イメージが読み取り専用（R/O, Read only）になりました。



podman が壊れやすいので、デフォルトの「abos-ctrl podman-storage --tmpfs」で運用することを推奨しますが、tmpfs の容量が小さくてイメージの操作には向いてません。

開発時には「abos-ctrl podman-storage --disk」の状態で作業を行い、運用時には「abos-ctrl podman-storage --tmpfs」に戻してください。戻る際に「copy」を選択する場合は一時的なストレージをそのまま使いつづけますので、すべての変更が残ります。



SWUpdate でアップデートをインストールする際には、/var/lib/containers/storage_readonly ディレクトリの不要になったイメージを自動的に削除します。

自動起動させる予定がなくても、「6.9.4. コンテナ起動設定ファイルを作成する」 を参考にして、 /etc/atmark/containers/*.conf を使ってください。 set_autostart no を設定することで自動実行されません。

6.9.2.17. イメージを SWUpdate で転送する

1. イメージをファイルに保存する：

```
[armadillo ~]$ podman image save -o <myimage>.tar <localimage>
```

2. ファイルを SWUpdate のイメージに入れる。

   二つのやり方があります：

   a. SWU イメージ内に組み込む

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/embed_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
[ATDE ~/mkswu]$ cat embed_container_nginx.desc
swdesc_option version=1

swdesc_embed_container "nginx_alpine.tar"
swdesc_files --extra-os nginx_start
[ATDE ~/mkswu]$ podman pull --arch arm64 docker.io/nginx:alpine
[ATDE ~/mkswu]$ podman run --rm docker.io/nginx:alpine uname -m
aarch64
[ATDE ~/mkswu]$ podman save docker.io/nginx:alpine > nginx_alpine.tar
[ATDE ~/mkswu]$ mkswu embed_container_nginx.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
embed_container_nginx.swu を作成しました
```

   b. USB ドライブに保存する

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/usb_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
[ATDE ~/mkswu]$ cat usb_container_nginx.desc
swdesc_option version=1

swdesc_usb_container "nginx_alpine.tar"
swdesc_files --extra-os nginx_start
[ATDE ~/mkswu]$ podman pull --arch arm64 docker.io/nginx:alpine
[ATDE ~/mkswu]$ podman run --rm docker.io/nginx:alpine uname -m
aarch64
[ATDE ~/mkswu]$ podman save docker.io/nginx:alpine > nginx_alpine.tar
[ATDE ~/mkswu]$ mkswu -o usb_container_nginx.swu usb_container_nginx.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
```

```
以下のファイルを USB メモリにコピーしてください：
'/home/atmark/mkswu/usb_container_nginx.swu'
'/home/atmark/mkswu/nginx_alpine.tar'
'/home/atmark/mkswu/.usb_container_nginx/nginx_alpine.tar.sig'

usb_container_nginx.swu を作成しました。
```

6.9.2.18. 開発時に有用な—privileged オプション

コンテナに、全権限と全てのデバイスへのアクセスを許可するオプション `--privileged` があります。このオプションを利用すると、コンテナに与えるべき最小の権限を洗い出す必要が無いため、開発時に有用です。

実運用の際、このオプションを利用することはセキュリティー上問題がある為、開発時にのみご利用ください。コンテナに必要な最低限の権限を与えることをおすすめします。

6.9.3. コンテナとコンテナに関連するデータを削除する



全てのコンテナとコンテナイメージ、コンテナに関するデータが削除されるため、充分に注意して使用してください。

6.9.3.1. VS Code から実行する

VS Code 上で ABOSDE(Armadillo Base OS Development Environment) から、Armadillo のコンテナイメージを全て削除する SWU イメージを作成することができます。

VS Code の左ペインの [COMMON PROJECT COMMAND] から [Generate Container Clear Swu] を実行すると、SWU イメージが作成されます。SWU イメージは `~/mkswu/container_clear.swu` に保存されます。

この SWU イメージを 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールしてください。

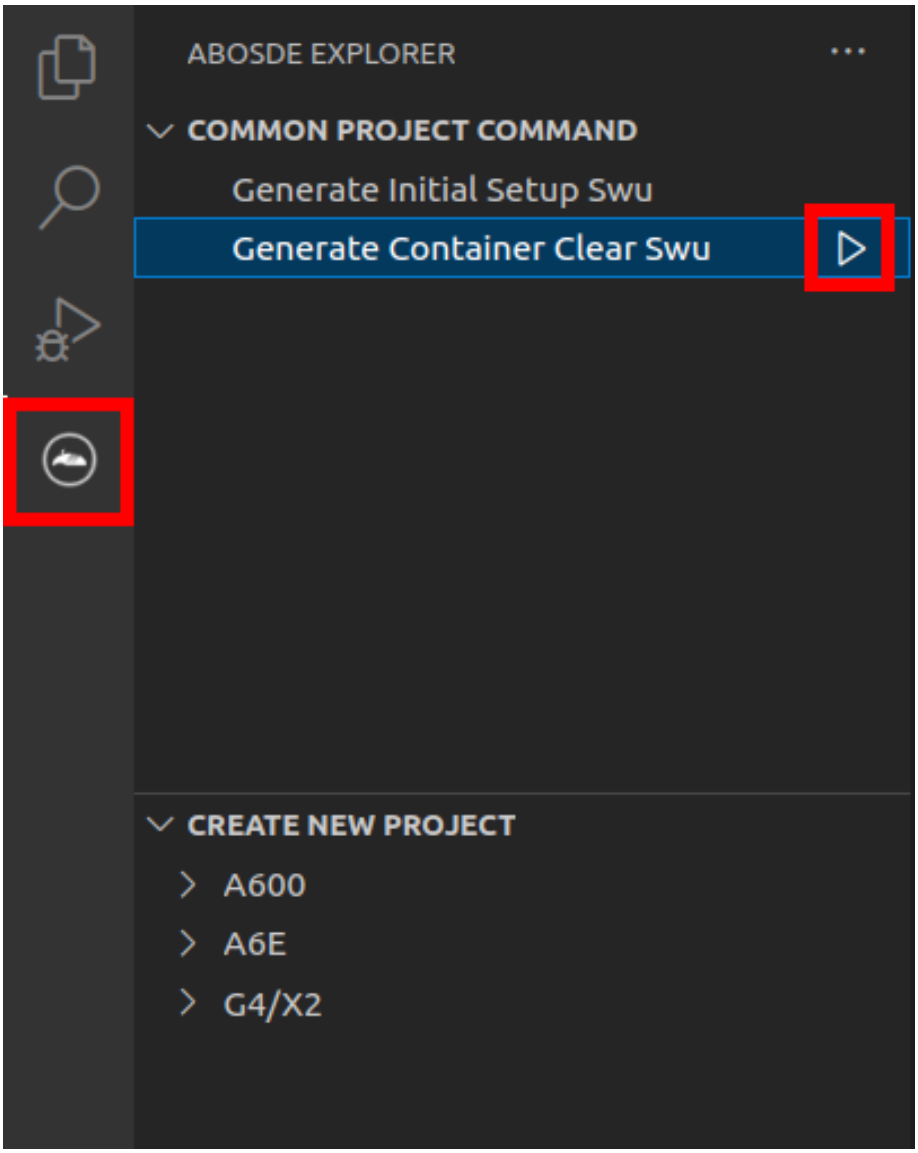


図 6.62 Armadillo 上のコンテナイメージを削除する

6.9.3.2. コマンドラインから実行する

`abos-ctrl container-clear` を使用すると、コンテナ、コンテナイメージ、コンテナに関するデータを削除することができます。

`abos-ctrl container-clear` は以下の通り動作します。

- 以下のファイル、ディレクトリ配下のファイルを削除
  - /var/app/rollback/volumes/
  - /var/app/volumes/
  - /etc/atmark/containers/*.conf
- 以下のファイルで `container` を含む行を削除
  - /etc/sw-versions

・ /etc/swupdate.watch

```
[armadillo ~]# abos-ctrl container-clear
This command will remove all containers and related data.
 - The following file and directories will be removed:
   - /var/app/rollback/volumes/
   - /var/app/volumes/
   - /etc/atmark/containers/*.conf
 - Lines containing the word "container" will be deleted from the following files:
   - /etc/sw-versions
   - /etc/swupdate.watch
Continue? [y/N]
y
Remove all container data succeeded
```

図 6.63 abos-ctrl container-clear 実行例

6.9.4. コンテナ起動設定ファイルを作成する

Armadillo Base OS では、/etc/atmark/containers/*.conf ファイルに指定されているコンテナがブート時に自動的に起動します。nginx.conf の記載例を以下に示します。

```
[armadillo ~]# cat /etc/atmark/containers/nginx.conf
set_image docker.io/library/nginx:alpine
set_readonly no
add_ports 80:80
```

図 6.64 コンテナを自動起動するための設定例

.conf ファイルに設定可能なパラメーターの説明を以下に記載します。podman_start --long-help コマンドでも詳細を確認できます。

6.9.4.1. コンテナイメージの選択

set_image [イメージ名]

イメージの名前を設定できます。

例: set_image docker.io/debian:latest, set_image localhost/myimage

イメージを rootfs として扱う場合に --rootfs オプションで指定できます。

例: set_image --rootfs /var/app/volumes/debian

6.9.4.2. ポート転送

add_ports [ホストポート]:[コンテナポート]

設定したポートで外部からコンテナへのアクセスが可能となります。

デフォルトは TCP で、UDP も /udp を付けて使えます。スペースで分けて複数のポートを設定することができます。

以下の例では、ポート 80、443(web)、UDP の 69(tftp)にアクセスすることができ、コンテナのポート 22(ssh)にはポート 2222 からアクセスすることができます。

例: add_ports 80:80 443:443 2222:22 69:69/udp



pod を使う場合、このオプションは pod の設定にしないと有効になりませんのでご注意ください。

6.9.4.3. デバイスファイル作成

add_devices [ホストパス]:[コンテナパス]

コンテナでデバイスを作成して、使用可能となります。

コンテナパスを設定しない場合はホストと同じパスを使います。

複数のデバイスを作成したい場合はスペースで分けて設定してください。

例： add_devices /dev/galcore /dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_78DA8CAF-video-index0:/dev/video3

ホストパスに「:」を含む場合は add_device ”[ホストパス]” ”[コンテナパス]” で追加できます。

例: add_device ”/dev/v4l/by-path/platform-xhci-hcd.1.auto-usb-0:1.1:1.0-video-index1” ”/dev/video3”

コンテナパスに「:」を含むようなパスは設定できません。

6.9.4.4. ボリュームマウント

add_volumes [ホストパス]:[コンテナパス]:[オプション]

指定するパスをコンテナ内でマウントして、データの保存や共有ができます。

ホストパスは以下のどれかを指定してください。

- /var/app/rollback/volumes/<folder> か <folder>:

  アップデートの際に新しくコピー（snapshot）した場合、コピー先のみ変更しますので、アップデート中でもこのデータを使うことができます。途中で電源が落ちた場合でも、このデータに影響はありません。

  SWUpdate でアップデートするデータに向いています。

- /var/app/volumes/<folder>: app パーティションに書きます。

  アップデートの際にコピーされませんので、アップデート中の新たな変更は更新されたコンテナ内のアプリケーションで見れます。

  ログやデータベースに向いています。

- /tmp/<folder>: 複数のコンテナでメモリファイルシステムを共有したい場合に使ってください。

・ /opt/firmware: 学習能力に必要なファムウェアライブラリーのパス。

コンテナパスを設定しない場合はホストパスと同じパスを使います。

オプションは podman run の --volume のオプションになりますので、 ro (read-only), nodev, nosuid, noexec, shared, slave 等を設定できます。

例：add_volumes /var/app/volumes/database:/database: ロールバックされないデータを/database で保存します。

例: add_volumes assets:/assets:ro,nodev,nosuid /opt/firmware: アプリケーションのデータを/assets で読み取り、/opt/firmware のファームウェアを使えます。

「:」はホスト側のパスとコンテナの側のパスを分割する意味があるため、ファイル名に「:」を使用することはできません。ホスト側のパスにのみ「:」が含まれてる場合は「 add_volumes "[ホストパス]" "[コンテナパス]" "[オプション]" 」と指定することで設定できます。



複数のコンテナでマウントコマンドを実行することがあれば、shared のフラグで起動後のマウントを共有することができます。

```
[armadillo ~]# cat /etc/atmark/containers/mounter.conf
set_image docker.io/alpine
add_args -ti
add_volumes /tmp/mnt:/mnt:shared ❶
add_args --cap-add SYS_ADMIN
add_devices /dev/sda1
[armadillo ~]# cat /etc/atmark/containers/client.conf
set_image docker.io/alpine
add_volumes /tmp/mnt:/mnt:slave ❷
add_args -ti
[armadillo ~]# podman exec mounter mount /dev/sda1 /mnt ❸
[armadillo ~]# podman exec client ls /mnt ❹
file_on_usb
```

図 6.65 ボリュームを shared でサブマウントを共有する例

❶ マウントを行うコンテナに shared の設定とマウント権限 (SYS_ADMIN) を与えます。

❷ マウントを使うコンテナに slave だけを設定すれば一方にしか共有されません。

❸ USB デバイスをマウントします。

❹ マウントされたことを確認します。

6.9.4.5. ホットプラグデバイスの追加

add_hotplugs [デバイスタイプ]

コンテナ起動後に挿抜を行なっても認識される(ホットプラグ)デバイスを設定できます。

通常、コンテナ内からデバイスを扱うためには、あらかじめ Armadillo 本体に当該のデバイスを接続した状態で、コンテナを起動する必要がありますが、 add_hotplugs を使用することでホットプラグに対応できます。

例: add_hotplugs input

add_hotplugs に指定できる主要な文字列とデバイスファイルの対応について、「表 6.9. add_hotplugs オプションに指定できる主要な文字列」に示します。

表 6.9 add_hotplugs オプションに指定できる主要な文字列

文字列	引数の説明	対象のデバイスファイル
input	マウスやキーボードなどの入力デバイス	/dev/input/mouse0, /dev/input/event0 など
video4linux	USB カメラなどの video4linux デバイスファイル	/dev/video0 など
sd	USB メモリなどの SCSI ディスクデバイスファイル	/dev/sda1 など

「表 6.9. add_hotplugs オプションに指定できる主要な文字列」に示した文字列の他にも、/proc/devices の数字から始まる行に記載されている文字列を指定することができます。「図 6.66. /proc/devices の内容例」に示す状態の場合、デバイスタイプを示す文字列としては、各行の先頭の数字を除いた mem や pty などを指定できることがわかります。

```
[armadillo ~]# cat /proc/devices
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
 10 misc
 13 input
 29 fb
 81 video4linux
: (省略)
```

図 6.66 /proc/devices の内容例

デバイスタイプと実際のデバイスファイルの対応については、 カーネルドキュメント: devices.txt(Github) [https://github.com/torvalds/linux/blob/master/Documentation/admin-guide/devices.txt] を参照してください。

複数のデバイスタイプを指定したい場合はスペースで分けて設定してください。

例: add_hotplugs input video4linux sd

6.9.4.6. 個体識別情報の環境変数の追加

add_armadillo_env

アットマークテクノが設定した個体識別情報をコンテナの環境変数として追加することができます。

例: add_armadillo_env

add_armadillo_env を設定することで追加されるコンテナの環境変数について、「表 6.10. add_armadillo_env で追加される環境変数」に示します。

表 6.10 add_armadillo_env で追加される環境変数

環境変数	環境変数の説明	表示例
AT_ABOS_VERSION	ABOS のバージョン	3.18.4-at.5
AT_LAN_MAC1	アットマークテクノが設定した LAN1（eth0）の MAC アドレス	00:11:0C:12:34:56
AT_PRODUCT_NAME	製品名	Armadillo-IoT A6E
AT_MODEL_NAME	モデル名	WLAN
AT_SERIAL_NUMBER	個体番号	00C900010001
AT_SE_PARAM	セキュアエレメントのデバイスファイルのパス	/dev/i2c-1:0x48

「表 6.10. add_armadillo_env で追加される環境変数」に示した環境変数をコンテナ上で確認する場合、「図 6.67. add_armadillo_env で設定した環境変数の確認方法」に示すコマンドを実行してください。ここでは、個体番号の環境変数を例に示します。

```
[container ~]# echo $AT_SERIAL_NUMBER
00C900010001
```

図 6.67 add_armadillo_env で設定した環境変数の確認方法

お客様が独自の環境変数をコンテナに追加する場合は「図 5.5. 個体番号の環境変数を conf ファイルに追記」を参考に conf ファイルを編集してください。

6.9.4.7. pod の選択

set_pod [ポッド名]

「6.9.2.12. pod でコンテナのネットワークネームスペースを共有する」で作成した pod の名前を入れてコンテナを pod 内で起動します。

例: set_pod mypod

6.9.4.8. ネットワークの選択

set_network [ネットワーク名]

この設定に「6.9.2.13. network の作成」で作成したネットワーク以外に none と host の特殊な設定も選べます。

none の場合、コンテナに localhost しかないネームスペースに入ります。

host の場合は OS のネームスペースをそのまま使います。

例: set_network mynetwork

6.9.4.9. IP アドレスの設定

set_ip [アドレス]

コンテナの IP アドレスを設定することができます。

例: set_ip 10.88.0.100



コンテナ間の接続が目的であれば、pod を使って localhost か pod の名前でアクセスすることができます。

6.9.4.10. 読み取り専用設定

set_readonly yes

コンテナ内からのファイルシステムへの書き込み許可を設定します。

デフォルトで書き込み可能となっています。

コンテナ内からのファイルシステムへの書き込みを禁止することで、tmpfs として使うメモリの消費を明示的に抑えることができますが、アプリケーションによっては読み込み専用のファイルシステムでは動作しない可能性もあります。

6.9.4.11. イメージの自動ダウンロード設定

set_pull [設定]

この設定を missing にすると、イメージが見つからない場合にイメージを自動的にダウンロードします。

always にすると、イメージがすでにダウンロード済みでも起動前に必ず更新の確認を取ります。

デフォルトでは never で、イメージが見つからない場合にエラーを表示します。

例：set_pull missing か set_pull always

6.9.4.12. コンテナのリスタート設定

set_restart [設定]

コンテナが停止した時にリスタートさせます。

podman kill か podman stop で停止する場合、この設定と関係なくリスタートしません。

デフォルトで on-failure になっています。

例: set_restart always か set_restart no

6.9.4.13. 信号を受信するサービスの無効化

set_init no

コンテナのメインプロセスが PID 1 で起動していますが、その場合のデフォルトの信号の扱いが変わります: SIGTERM などのデフォルトハンドラが無効です。

そのため、init 以外のコマンドを set_command で設定する場合は podman-init のプロセスを PID 1 として立ち上げて、設定したコマンドをその子プロセスとして起動します。

例: set_init no

6.9.4.14. podman logs 用のログサイズ設定

set_log_max_size <サイズ>

podman logs でログを表示するために /run にログファイルを保存しています。そのログのサイズが設定したサイズを越えるとクリアされます。デフォルトは「1MB」です。

6.9.4.15. podman のフックの仕組み

add_hook --stage <ステージ> [--] コマンド [コマンド引数]

コンテナが起動されるなど、動作ステージの変化をフックとしてコマンドを実行します。複数のステージで実行したい場合は --stage オプションを複数設定してください。

指定可能なステージは precreate, prestart, createRuntime, createContainer, startContainer, poststart, と poststop です。ステージの意味や使用方法の詳細は podman のドキュメンテーションを参照してください。



Armadillo Base OS 3.19.1-at.4 現在では set_restart によるコンテナの再起動でも 1 度目の停止時のみ poststop フックが実行されます。2 度目以降の停止では実行されませんのでご注意ください。

6.9.4.16. ヘルスチェック機能の設定

set_healthcheck [引数] [--] コマンド [コマンド引数]

定期的にコマンドを実行して、コンテナの正常性を確認します。指定可能な引数は以下のとおりです：

- **--retries <リトライ数>**: エラーを検知するまでのリトライ回数。(デフォルト: 3)
- **--action <none|restart|kill|stop|reboot|rollback>**: 指定したリトライ回数分連続でチェックが失敗したときのアクション (デフォルト: restart)：
  - none: set_healthcheck_fail_command に指定した処理を実行する以外何もしません。
  - restart: コンテナを再起動します。 set_restart オプションと異なり、コンテナを起動しなおし初期状態で再起動します。
  - kill/stop: コンテナを停止します。
  - reboot: Armadillo を再起動します。
  - rollback: ロールバック可能の場合はロールバックして Armadillo を再起動します。ロールバック不可能な場合はそのまま Armadillo を再起動します。
- **--interval <時間>**: チェックする時間間隔です。(デフォルト: 1 min)
- **--start-period <時間>**: 最初のチェックを実行する前の待ち時間です。(デフォルト: interval 設定の値)
- **--timeout <秒数>**: 設定された時間以内にヘルスチェックが終了しなかった場合は失敗となります。(デフォルト: 無し)

また、いくつかのタイミングでコマンドを実行させることができます：

- **set_healthcheck_start_command コマンド [コマンド引数]**: コンテナ起動後にヘルスチェックが初めて成功した際に実行されるコマンドです。

- **set_healthcheck_fail_command コマンド [コマンド引数]**: ヘルスチェックが retries 回失敗した後に実行されるコマンドです。このコマンドは set_healthcheck の --action 設定の前に実行されますので、コマンドだけを実行したい場合は --action none で無効化してください。

- **set_healthcheck_recovery_command コマンド [コマンド引数]**: ヘルスチェックが retries 回失敗した後に再び成功した際に実行されるコマンドです。コンテナを起動する際に成功せずに失敗した場合は、その 1 回目の成功の際に set_healthcheck_start_command で設定されたコマンドのみが実行されます。

例: set_healtcheck -- curl -s --fail http://localhost:8080/status
例: set_healthcheck_start_command abos-ctrl rollback-clone

```
armadillo:~# grep podman_atmark /var/log/messages
Jun 20 11:33:21 armadillo user.notice podman_atmark: my_container healthcheck is now healthy (was starting)
Jun 20 11:33:21 armadillo user.notice podman_atmark: my_container first healthy check: running abos-ctrl rollback-clone
Jun 20 11:40:21 armadillo user.notice podman_atmark: my_container healthcheck failed (from healthy, 1 / 3)
Jun 20 11:41:21 armadillo user.notice podman_atmark: my_container healthcheck failed (from healthy, 2 / 3)
Jun 20 11:42:21 armadillo user.notice podman_atmark: my_container healthcheck failed (from healthy, 3 / 3)
Jun 20 11:42:21 armadillo user.notice podman_atmark: my_container is unhealthy, restarting container
Jun 20 11:43:21 armadillo user.notice podman_atmark: my_container healthcheck is now healthy (was failed)
```

図 6.68 上記の例でエラーを発生させた際の起動ログ

6.9.4.17. 自動起動の無効化

set_autostart no または **set_autostart create**

Armadillo の起動時にコンテナを自動起動しないように設定できます。

create を指定した場合はコンテナは生成されており、podman start <name> で起動させることができます。

no を指定した場合は podman_start <name> で起動させることができます。



コンフィグに記載していないイメージはアップデートの際に削除されますので、そういったイメージに対して設定してください。

6.9.4.18. 実行コマンドの設定

set_command [コマンド]

コンテナを起動するときのコマンド。設定されなかった場合、コンテナイメージのデフォルトを使います。

例: set_command /bin/sh -c "echo bad example"

6.9.4.19. コンテナ起動前にコマンドを実行する

add_pre_command [コマンド]

コンテナを起動する直前に設定したコマンドを実行します。

Armadillo Base OS の環境で実行されてますので、ハードウェアの設定等に適切です。

また、複数のコマンドを実行する場合は順番に実行されます。設定したコマンドが 1 つでも失敗した場合は、コンテナは起動されません。

例: add_pre_command gpioset --daemonize CONx_y=1

6.9.4.20. podman run に引数を渡す設定

add_args [引数]

ここまでで説明した設定項目以外の設定を行いたい場合は、この設定で podman run に直接引数を渡すことができます。

例：add_args --cap-add=SYS_TTY_CONFIG --env=XDG_RUNTIME_DIR=/run/xdg_home

6.9.5. アットマークテクノが提供するイメージを使う

アットマークテクノは、動作確認環境として使用できる Debian ベースのイメージを提供しています。ここでは以下の 3 つの手順について説明します。

- ABOSDE からインストールする方法
- Docker ファイルからイメージをビルドする方法
- すでにビルド済みのイメージを使う方法

6.9.5.1. ABOSDE からインストールする

1. インストール用のプロジェクトを作成する

   VS Code の左ペインの [A6E] から [Atmark Container New Project] を実行し、表示されるディレクトリ選択画面からプロジェクトを保存するディレクトリを選択してください。保存先を選択すると、プロジェクト名を入力するダイアログが表示されるので、任意のプロジェクト名を入力してエンターキーを押してください。この操作により、選択した保存先に、入力したプロジェクト名と同名のディレクトリが作成されます。

   また、ここでは次のように設定しています。

   - **保存先**：ホームディレクトリ
   - **プロジェクト名**：my_project

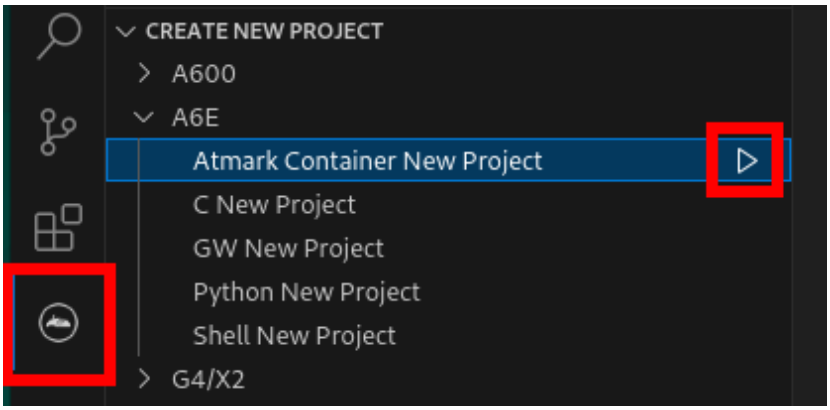


図 6.69 インストール用のプロジェクトを作成する

2. SWU イメージを作成する

   VS Code の左ペインの [my_project] から [Generate at-debian-image container setup swu] を実行してください。

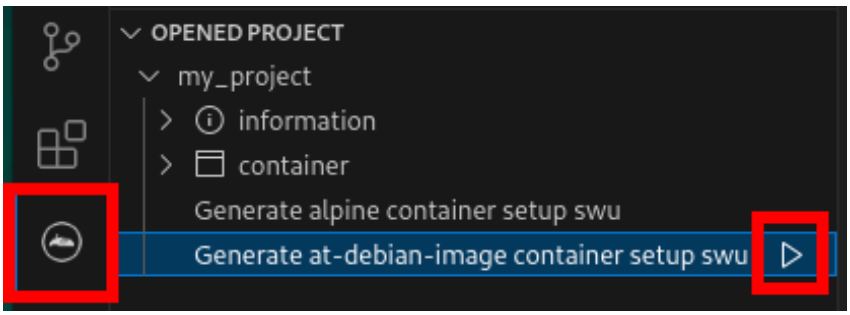


図 6.70 at-debian-image のコンテナイメージをインストールする SWU ファイルを作成する

   作成した SWU ファイルは `container_setup/at-debian-image/at-debian-image-armv7.swu` に保存されています。この SWU イメージを 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールしてください。

3. SBOM 生成に関わる設定を行う

   ABOSDE から作成した場合は SBOM が同時に生成されます。詳細は 「3.19. SBOM 生成に関わる設定を行う」 をご確認ください。SBOM の生成には以下の二つのファイルが必要です。

   - コンフィグファイル
   - desc ファイル

   SBOM の生成にはライセンス情報を示したコンフィグファイルを使用します。コンフィグファイルは `container_setup/at-debian-image-armv7.sbom_config.yaml.tmpl` になります。SWU イメージ作成時にこのコンフィグファイルからバージョン番号をアップデートした `container_setup/at-debian-image-armv7.sbom_config.yaml` が生成されます。

   リリース時にはコンフィグファイルの内容を確認し、正しい内容に変更してください。各項目の詳細な説明については SPDX specification v2.2.2 (https://spdx.github.io/spdx-spec/v2.2.2/) をご覧ください。SBOM に含めるコンテナイメージ等の情報については desc ファ

イルに記載されています。各項目の説明については 「6.37.2.4. SWU イメージと同時に SBOM を作成する」 をご覧ください。

6.9.5.2. Docker ファイルからイメージをビルドする

Armadillo-IoT ゲートウェイ A6E コンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から「Debian [VERSION] サンプル Dockerfile」 ファイル (at-debian-image-dockerfile-[VERSION].tar.gz) をダウンロードします。その後 podman build コマンドを実行します。

```
[armadillo ~]# tar xzf at-debian-image-dockerfile-[VERSION].tar.gz
[armadillo ~]# cd at-debian-image-dockerfile-[VERSION]
[armadillo ~]# abos-ctrl podman-storage --disk
[armadillo ~]# podman build -t at-debian-image:latest .
:
:（省略）
:
[armadillo ~]# podman images
REPOSITORY                  TAG         IMAGE ID      CREATED             SIZE
localhost/at-debian-image   latest      c8e8d2d55456  About a minute ago  233 MB
docker.io/library/debian    bullseye    723b4a01cd2a  18 hours ago        123 MB
```

図 6.71 Docker ファイルによるイメージのビルドの実行例

podman images コマンドにより at-debian-image がビルドされたことが確認できます。library/debian イメージはベースとなっている Debian イメージです。

6.9.5.3. ビルド済みのイメージを使用する

Armadillo-IoT ゲートウェイ A6E コンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から「Debian [VERSION] サンプルコンテナイメージ」 ファイル (at-debian-image-[VERSION].tar) をダウンロードします。その後 podman load コマンドを実行します。

```
[armadillo ~]# podman load -i at-debian-image-[VERSION].tar
:
:（省略）
:
[armadillo ~]# podman images
REPOSITORY                  TAG         IMAGE ID      CREATED       SIZE
localhost/at-debian-image   [VERSION]   93a4ec873ac5  17 hours ago  233 MB
localhost/at-debian-image   latest      93a4ec873ac5  17 hours ago  233 MB
```

図 6.72 ビルド済みイメージを load する実行例

podman images コマンドにより at-debian-image がビルドされたことが確認できます。

6.9.6. alpine のコンテナイメージをインストールする

alpine のコンテナイメージは、 ABOSDE を用いてインストールすることが可能です。「6.9.5.1. ABOSDE からインストールする」 を参照して、 インストール用のプロジェクトを作成しておいてください。

VS Code の左ペインの [my_project] から [Generate alpine container setup swu] を実行してください。

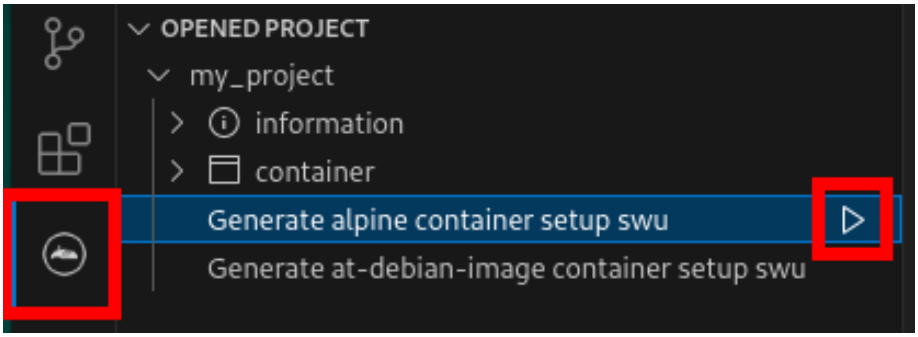


図 6.73 alpine のコンテナイメージをインストールする SWU ファイルを作成する

作成した SWU ファイルは container_setup/alpine.swu に保存されています。この SWU イメージを 「3.3.4.5. SWU イメージのインストール」 を参照して Armadillo へインストールしてください。

6.9.6.1. SBOM 生成に関わる設定を行う

ABOSDE から作成した場合は SBOM が同時に生成されます。詳細は 「3.19. SBOM 生成に関わる設定を行う」 をご確認ください。SBOM の生成には以下の二つのファイルが必要です。

- コンフィグファイル
- desc ファイル

SBOM の生成にはライセンス情報を示したコンフィグファイルを使用します。コンフィグファイルは container_setup/alpine.sbom_config.yaml.tmpl になります。SWU イメージ作成時にこのコンフィグファイルからバージョン番号をアップデートした container_setup/alpine.sbom_config.yaml が生成されます。

リリース時にはコンフィグファイルの内容を確認し、正しい内容に変更してください。各項目の詳細な説明については SPDX specification v2.2.2 (https://spdx.github.io/spdx-spec/v2.2.2/) をご覧ください。SBOM に含めるコンテナイメージ等の情報については desc ファイルに記載されています。各項目の説明については 「6.37.2.4. SWU イメージと同時に SBOM を作成する」 をご覧ください。

6.9.7. コンテナのネットワークを扱う

この章では、コンテナ内のネットワークを扱う方法について示します。

6.9.7.1. コンテナの IP アドレスを確認する

基本的にコンテナの IP アドレスは Podman イメージからコンテナを作成したときに自動的に割り振られます。コンテナに割り振られている IP アドレスはホスト OS 側からは podman inspect コマンドを用いて、以下のように確認することができます。

```
[armadillo ~]# vi /etc/atmark/containers/net_example.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# podman_start net_example
Starting 'net_example'
```

```
48ae479af65445674323567c17c5418dd4624292351e061bd2bd8a0add4cf150
[armadillo ~]# podman inspect --format '{{ .NetworkSettings.IPAddress }}' net_example
10.88.0.17
```

図 6.74 コンテナの IP アドレス確認例

コンテナ内の ip コマンドを用いて確認することもできます。

```
[armadillo ~]# podman exec net_example ip addr show eth0
3: eth0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 10.88.0.17/16 brd 10.88.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::40e5:98ff:feec:4b17/64 scope link
       valid_lft forever preferred_lft forever
```

図 6.75 ip コマンドを用いたコンテナの IP アドレス確認例

6.9.7.2. コンテナに固定 IP アドレスを設定する



podman はデフォルトで 10.88.0.0/16 を使います。

他に使用している IP アドレスと被った場合等はコンテナに別の IP アドレスを設定してください。

コンテナに固定 IP アドレスを設定するためには、最初にユーザ定義のネットワークを作成する必要があります。以下に 198.51.100.0/24 にユーザ定義のネットワークを作成する例を示します。

```
[armadillo ~]# vi /etc/atmark/containers/my_network.conf
set_type network
set_subnet 198.51.100.0/24
[armadillo ~]# podman_start my_network
Creating network 'my_network'
my_network
```

図 6.76 ユーザ定義のネットワーク作成例

コンテナを作成する際に、上記で作成したネットワークと設定したい IP アドレスを渡すことで、コンテナの IP アドレスを固定することができます。以下の例では、IP アドレスを 198.51.100.10 に固定します。

```
[armadillo ~]# vi /etc/atmark/containers/network_example.conf
set_image docker.io/alpine
set_command sleep infinity
set_network my_network
set_ip 198.51.100.10
[armadillo ~]# podman_start network_example
```

```
Starting 'network_example'
3ea8c9031bf833228908bd73d8929b1d543b189b436c218e0634e0d39409e100
```

図 6.77 IP アドレス固定のコンテナ作成例

コンテナの IP アドレスが、198.51.100.10 に設定されていることが確認できます。

```
[armadillo ~]# podman inspect --format '{{ .NetworkSettings.Networks.my_network.IPAddress }}'
network_example
198.51.100.10
```

図 6.78 コンテナの IP アドレス確認例

6.9.8. コンテナ内にサーバを構築する

この章では、コンテナ内で様々なサーバを構築する方法について示します。この章で取り上げているサーバは alpine の apk コマンドでインストールすることが可能です。

6.9.8.1. HTTP サーバを構築する

ここでは、HTTP サーバとして Apache と lighttpd の 2 種類を使用する場合について説明します。

・ Apache を使用する

alpine イメージからコンテナを作成し、そのコンテナ内に Apache をインストールします。コンテナ作成の際に、ホスト OS の 8080 番ポートをコンテナ内の 80 番ポートに転送する指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/apache_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 8080:80
[armadillo ~]# podman_start apache_example
Starting 'apache_example'
ea0a1ed9c2fe170a6db02e480300467510f4e844900efb35c7a24cc1a8653af2
[armadillo ~]# podman exec -it apache_example sh
[container ~]# apk upgrade && apk add apache2
[container ~]# httpd
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
10.88.0.2. Set the 'ServerName' directive globally to suppress this message
```

図 6.79 コンテナに Apache をインストールする例

他の PC などの Web ブラウザから、ホスト OS の IP アドレスの 8080 番ポートに接続すると、動作確認用ページが表示されます。デフォルトでは、/var/www/localhost/htdocs ディレクトリにファイルを置くことで Web ブラウザから閲覧できます。Apache の詳細な設定は、/etc/apache2 ディレクトリにある設定ファイルを編集することで変更可能です。

・ lighttpd を使用する

alpine イメージからコンテナを作成し、そのコンテナ内に lighttpd をインストールします。コンテナ作成の際に、ホスト OS の 8080 番ポートをコンテナ内の 80 番ポートに転送する指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/lighttpd_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 8080:80
[armadillo ~]# podman_start lighttpd_example
Starting 'lighttpd_example'
fd7ea338d09c5e8962654ed54bba17fb6a9ed4fca1b344e350bbf8f943d2f12b
[armadillo ~]# podman exec -it lighttpd_example sh
[container ~]# apk upgrade && apk add lighttpd
[container ~]# echo "<html><body>It works!</body></html>" > /var/www/localhost/htdocs/index.html
[container ~]# lighttpd -f /etc/lighttpd/lighttpd.conf
```

図 6.80 コンテナに lighttpd をインストールする例

lighttpd はデフォルトでは動作確認用ページが用意されていないため、上記の手順では簡単なページを /var/www/localhost/htdocs ディレクトリの下に配置しています。他の PC などの Web ブラウザから、ホスト OS の IP アドレスの 8080 番ポートに接続すると表示されます。lighttpd の詳細な設定は、/etc/lighttpd ディレクトリにある設定ファイルを編集することで変更可能です。

6.9.8.2. FTP サーバを構築する

ここでは、FTP サーバとして vsftp を使用する場合について説明します。alpine イメージからコンテナを作成し、そのコンテナ内に vsftpd をインストールします。コンテナ作成の際に、FTP 通信で使用するポートについてホスト OS 側からコンテナ内のポートに転送する指定と、コンテナ内の環境変数として PASV_ADDRESS にホスト OS 側の IP アドレスの指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/ftp_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 21:21 21100-21110:21100-21110
add_args --env=PASV_ADDRESS=<ホストの IP アドレス>
[armadillo ~]# podman_start ftp_example
Starting 'ftp_example'
efcf1ba752c2db9ae1a33ac11af3be71d95ac7b737ce9734730ebca602e57796
[armadillo ~]# podman exec -it ftp_example sh
[container ~]# apk upgrade && apk add vsftpd
```

図 6.81 コンテナに vsftpd をインストールする例

コンテナ内にユーザアカウントを作成し、このユーザで ftp ログインできるようにします。

```
[container ~]# adduser atmark
Changing password for atmark
New password: （パスワードを入力）
Retype password: （パスワードを入力）
passwd: password for atmark changed by root
```

図 6.82 ユーザを追加する例

作成したユーザで ftp ログインできるように、vsftpd の設定ファイルを編集します。

```
[container ~]# sed -i -e 's/anonymous_enable=YES/#anonymous_enable=YES/g' /etc/vsftpd/vsftpd.conf
[container ~]# sed -i -e 's/#local_enable=YES/local_enable=YES/g' /etc/vsftpd/vsftpd.conf
[container ~]# sed -i -e 's/#write_enable=YES/write_enable=YES/g' /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_enable=YES" >> /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_min_port=21100" >> /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_max_port=21110" >> /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_address=$PASV_ADDRESS" >> /etc/vsftpd/vsftpd.conf
```

図 6.83 設定ファイルの編集例

編集した設定ファイルを指定して vftpd を起動することにより、ftp 接続可能となります。ftp ログイン時のアカウントは前述の手順で作成したものを使用します。

```
[container ~]# vsftpd /etc/vsftpd/vsftpd.conf
```

図 6.84 vsftpd の起動例

6.9.8.3. Samba サーバを構築する

ここでは、Samba サーバの構築方法について説明します。alpine イメージからコンテナを作成し、そのコンテナ内に samba をインストールします。コンテナ作成の際に、samba で使用するポートについてホスト OS 側からコンテナ内のポートに転送する指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/smb_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 139:139 445:445
[armadillo ~]# podman_start smb_example
Starting 'smb_example'
6d81c01fe27b5a92ee6ea69de2f9a8dbb569d420c2f5f630ece1966c81824a1f
[armadillo ~]# podman exec -it smb_example sh
[container ~]# apk upgrade && apk add samba
```

図 6.85 コンテナに samba をインストールする例

コンテナ内にユーザアカウントを作成し、このユーザで samba にログインできるようにします。

```
[container ~]# adduser atmark
Changing password for atmark
New password: （パスワードを入力）
Retype password: （パスワードを入力）
passwd: password for atmark changed by root
[container ~]# pdbedit -a atmark
new password: （パスワードを入力）
retype new password: （パスワードを入力）
```

図 6.86 ユーザを追加する例

samba を起動すると、前述の手順で作成したユーザアカウントで他の PC などからログインすることができます。

```
[container ~]# smbd
```

図 6.87 samba の起動例

共有するディレクトリの指定などの詳細設定は /etc/samba/smb.conf ファイルを編集することで変更可能です。

6.9.8.4. SQL サーバを構築する

ここでは、RDMS として sqlite を使用する場合について説明します。alpine イメージからコンテナを作成し、そのコンテナ内に sqlite をインストールします。

```
[armadillo ~]# vi /etc/atmark/containers/sqlite_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /var/app/volumes/sqlite_db:/db
[armadillo ~]# podman_start sqlite_example
Starting 'sqlite_example'
114c5f1dbb7e81293dcb8fbe0c600b861626375b14cfe4023761acaa84fdcad1
[armadillo ~]# podman exec -it sqlite_example sh
[container ~]# apk upgrade && apk add sqlite
```

図 6.88 コンテナに sqlite をインストールする例

コンテナ内に入り、sqlite3 コマンドを実行すると sqlite のプロンプトが表示されデータベースの操作ができるようになります。

```
[container ~]# sqlite3 /db/mydb.sqlite
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite>
```

図 6.89 sqlite の実行例

6.9.9. コンテナからの poweroff 及び reboot

Armadillo Base OS は busybox init で shutdown と reboot を対応します。

busybox init で PID 1 に signal を送ることで shutdown や reboot となります。コンテナから signal を送るように、pid namespace を共有する必要がありますが、共有されたら kill で実行できます。

```
[armadillo ~]# vi /etc/atmark/containers/shutdown_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_args --pid=host
[armadillo ~]# podman_start shutdown_example
Starting 'shutdown_example'
c8e3b9b418fc72395db9f3c22b1eb69eb41eaaf790d3b7151047ef066cc4c8ff
[armadillo ~]# podman exec -ti shutdown_example sh
```

```
[container ~]# kill -USR2 1  (poweroff)
[container ~]# kill -TERM 1  (reboot)
```

図 6.90 コンテナから shutdown を行う

6.9.10. 異常検知

この章では、コンテナ内で動作しているアプリケーションが意図せずに停止してしまった際にシステムを再起動する方法について示します。

Armadillo では、U-Boot や Linux カーネルがバックグラウンドでハードウェアウォッチドッグを利用しています。ソフトウェアが正常に動作している間は、定期的にウォッチドッグタイマーをキックするため、再起動は発生しません。一方、U-Boot や Linux カーネルの動作が停止すると、ウォッチドッグタイマーをキックできなくなり、Armadillo は自動的に再起動します。この処理をアプリケーションから行う事も可能です。その場合は、アプリケーションが停止した場合に Armadillo を再起動する事ができます。

6.9.10.1. ハードウェアウォッチドッグタイマーを扱う

コンテナ内で動作するアプリケーションからハードウェアウォッチドッグタイマーを扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/watchdogN を渡す必要があります。以下は、/dev/watchdog0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/watchdog_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/watchdog0
[armadillo ~]# podman_start watchdog_example
Starting 'watchdog_example'
a5d329cca49d60423ce4155d72a119b8049a03dbd1d0277817a253e96dce7bc7
```

図 6.91 ハードウェアウォッチドッグタイマーを使うためのコンテナ作成例

ハードウェアウォッチドッグタイマーは、プログラム内からデバイスファイル /dev/watchdog0 を open した時点で起動します。コンテナ内に入ってハードウェアウォッチドッグタイマーを echo コマンドで起動する例を以下に示します。

```
[armadillo ~]# podman exec -it watchdog_example sh
[container ~]# echo > /dev/watchdog0
```

図 6.92 コンテナ内からハードウェアウォッチドッグタイマーを起動する実行例

ハードウェアウォッチドッグタイマーを起動した後、/dev/watchdog0 に（ V 以外の）任意の文字を書き込むことでハードウェアウォッチドッグタイマーをキックすることができます。

10 秒間（ V 以外の）任意の文字の書き込みがない場合は、システムが再起動します。



タイムアウト値の変更は util-linux に含まれるコマンドを利用して、 `wdctl --settimeout <秒数>` のように行う事ができます。 タイムアウトの有効値は 1~128 秒です。

```
[armadillo ~]# podman exec -it watchdog_example sh
[container ~]# echo a > /dev/watchdog0
```

図 6.93 ハードウェアウォッチドッグタイマーをキックする実行例



このコマンドを実行すると、必ず「watchdog: watchdog0: watchdog did not stop!」とエラーメッセージが表示されます。これは、ハードウェアの仕様により、一度起動したウォッチドッグタイマーを停止できない為です。機能的な問題はありません。エラーメッセージの出力を抑制するには、/dev/watchdog0 を open 後、close せずに都度 write を行うアプリケーションを作成する必要があります。

ハードウェアウォッチドッグタイマーを停止したい場合は、/dev/watchdog0 に V を書き込みます。停止後は Linux カーネルがタイマーを定期的にキックする状態に戻ります。

```
[armadillo ~]# podman exec -it watchdog_example sh
[container ~]# echo V > /dev/watchdog0
```

図 6.94 ハードウェアウォッチドッグタイマーを停止する実行例

6.10. ゲートウェイコンテナを動かす

Armadillo-IoT ゲートウェイ A6E にはゲートウェイコンテナをインストールして動作させることができます。本章は、ゲートウェイコンテナの概要と動かす方法について記載しています。

6.10.1. ゲートウェイコンテナの概要

ゲートウェイコンテナは各インターフェースの操作や取得するデータの設定、接続するクラウドの情報を設定するだけで、コンテナ内で動作するアプリケーションを修正することなく、クラウドにデータを送信することができるコンテナです。

ゲートウェイコンテナを利用して実施できる内容は下記の通りです。

表 6.11 利用できるインターフェース・機能

インターフェース	機能
RS-485 (ModbusRTU)	レジスタ読み出し
	レジスタ書き込み
接点入力 2ch	ポーリング監視
	エッジ検出
接点出力 2ch	指定レベル出力
アプリケーション LED	点灯/消灯操作
ユーザースイッチ	状態取得

表 6.12 利用できるクラウドベンダー・サービス

クラウドベンダー	クラウドサービス
AWS	AWS IoT Core
Azure	Azure IoT

クラウドベンダー	クラウドサービス
SORACOM	SORACOM Harvest

インターフェースやクラウドサービスの選択はコンフィグ設定で行う事ができます。また、センサーデータのログ出力やネットワーク断時のキャッシュ機能にも対応しています。



SORACOM を使用するためには SORACOM SIM が必要です。

6.10.2. ゲートウェイコンテナ利用の流れ

以下では、必要機器の接続やネットワークの設定は完了しているものとして説明を進めます。一連の流れは下記の通りです。

ゲートウェイコンテナでは AWS IoT Core と Azure IoT、また SORACOM Harvest への接続をサポートしています。それぞれについて、データの可視化までを行うことが出来る環境を構築するためのテンプレートを提供しています。

1. ゲートウェイコンテナのインストール
2. ゲートウェイコンテナ起動確認
3. 接続先の クラウド 環境を構築 (クラウドにデータを送信する場合)
    a. AWS IoT Core
    b. Azure IoT Hub
    c. SORACOM Harvest
4. コンフィグ 設定
    a. インターフェース設定
    b. 接続先クラウド設定
5. コンテナ起動・実行
6. コンテナ終了

6.10.3. ゲートウェイコンテナ起動確認

ゲートウェイコンテナは、デフォルトで Armadillo-IoT ゲートウェイ A6E に電源を入れると自動的に起動する設定となっています。Armadillo が起動し、ゲートウェイコンテナが起動・実行されると、 アプリケーション LED が点滅します。

6.10.4. 接続先の クラウド 環境を構築 (AWS)

AWS では、 AWS IoT Core と Amazon CloudWatch を組み合わせてデータの可視化を行います。本項では、 AWS 上で実施する設定を記載します。

手順中で使用するファイルは、Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から

「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) から予めダウンロードしておきます。

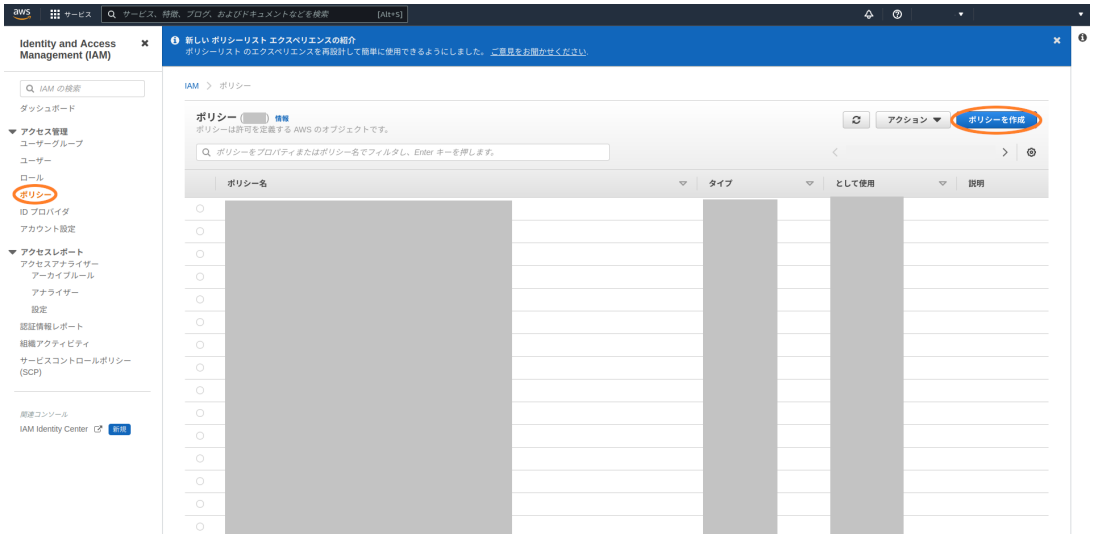
6.10.4.1. AWS アカウントを作成する

AWS アカウントの作成方法については、AWS 公式サイトの AWS アカウント作成の流れ https://aws.amazon.com/jp/register-flow/を参照してください。

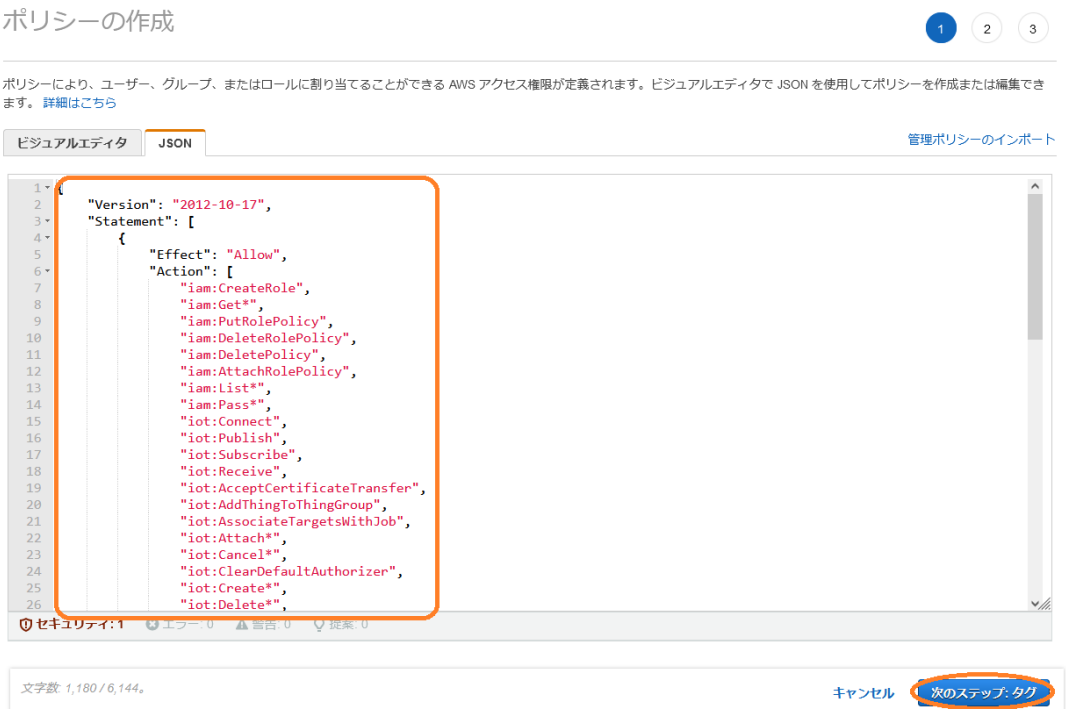
6.10.4.2. IAM ユーザーを作成する

AWS IAM (Identity and Access Management) は、AWS リソースへのアクセスを安全に管理するためのウェブサービスです。IAM により、誰を認証(サインイン)し、誰にリソースの使用を承認する(アクセス許可を持たせる)かを管理することができます。

1. IAM へ移動し、「アクセス管理」→「ポリシー」を開き、「ポリシー作成」をクリックします。

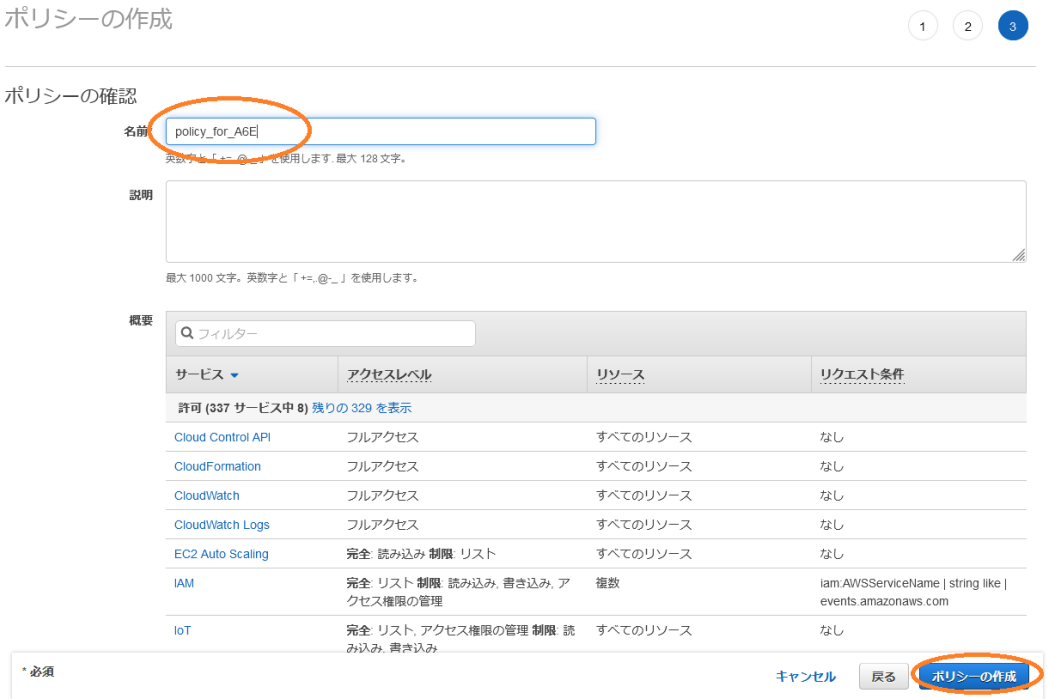


2. 「JSON」を選択し、「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) AWS フォルダ内の `a6e_aws_iam_policy.json` のファイルの内容を貼り付け、「次のステップ：タグ」をクリックします。

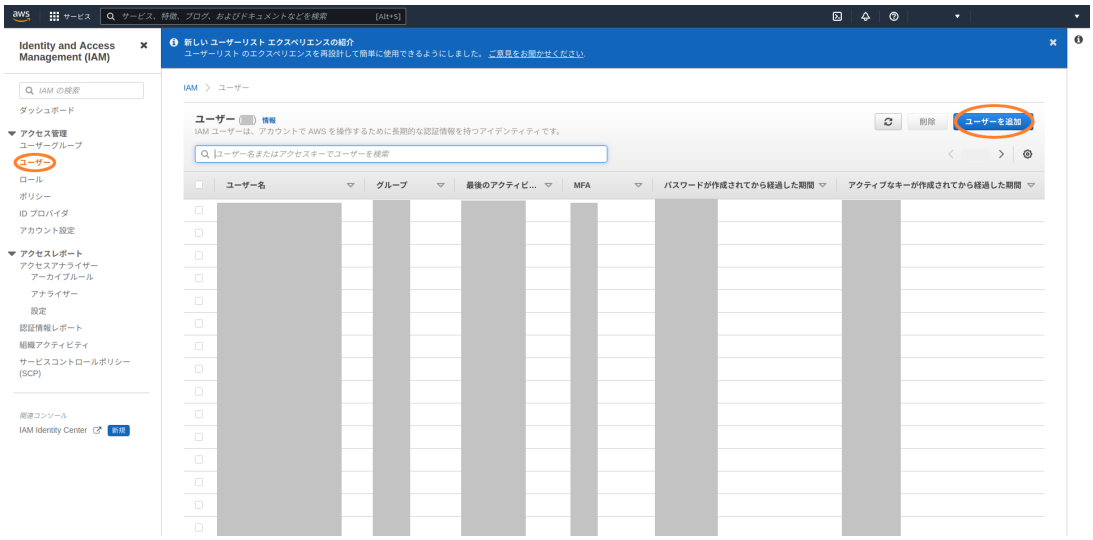


3. 何も選択せずに、「次のステップ：確認」をクリックします。

4. ポリシー名を入力し、「ポリシーの作成」をクリックします。ここでは、ポリシー名を "policy_for_A6E" としています。



5. IAM から、「アクセス管理」→「ユーザー」を開き、「ユーザーを追加」をクリックします。



6. 下記の通り入力、選択し、「次へ」をクリックします。

   - ユーザー名を入力する
   - 「AWS マネジメントコンソールへのユーザーアクセスを提供する - オプション」を選択する
   - コンソールパスワードは「自動生成されたパスワード」を選択する
   - 「ユーザーは次回のサインイン時に新しいパスワードを作成する必要があります (推奨)。」にチェックを入れる



7. 「ポリシーを直接アタッチする」をクリックし、先ほど作成したポリシーを選択して、「次へ」をクリックします。



8. 表示される内容を確認し、「ユーザーの作成」をクリックします。



9. 「.csv ファイルをダウンロード」をクリックし、"<ユーザー名>_credentials.csv" をダウンロードして、「ユーザーリストに戻る」をクリックします。

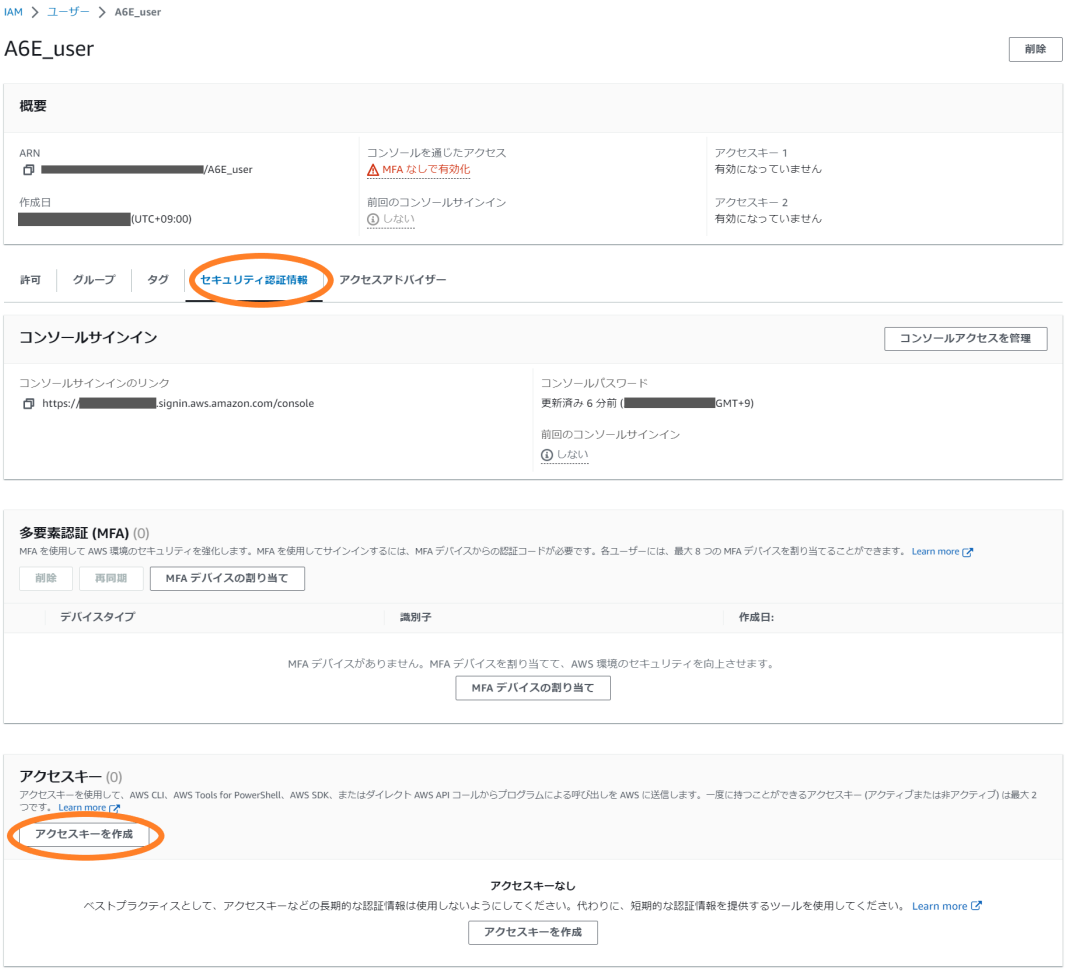


6.10.4.3. アクセスキーを作成する

1. 作成したユーザーをユーザーリストの中から選択します。



2. ユーザー情報画面の「セキュリティ認証情報」-「アクセスキーを作成」をクリックします。



3. 「AWS の外部で実行されるアプリケーション」を選択し、「次へ」をクリックします。



4. 「アクセスキーを作成」をクリックします。

5. 「.csv ファイルをダウンロード」をクリックし、"<ユーザー名>_accessKeys.csv" をダウンロードして、「完了」をクリックします。



6.10.4.4. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する

AWS IoT Core に登録する Thing 名は Armadillo のシリアル番号を使用します。環境設定時、パラメータに指定する必要があるため、下記のコマンドを実行しシリアル番号を取得します。

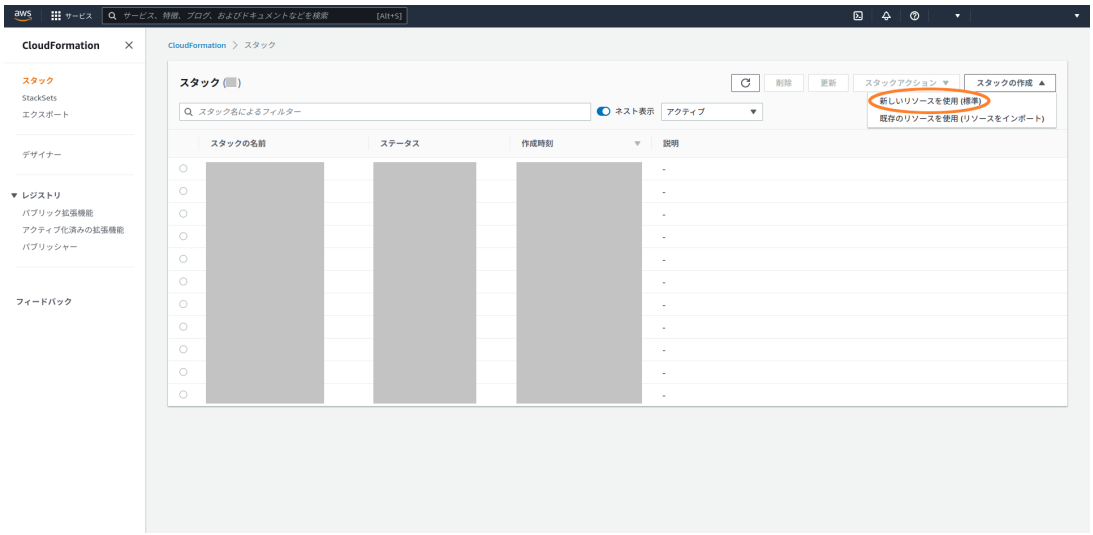
```
armadillo:~# hexdump -v -s 0xa0 -n 8 -e '/4 "%08X"' /sys/bus/nvmem/devices/imx-ocotp0/nvmem | cut
-c 5-
00CD11112222 ❶
```

❶ この場合、00CD11112222 がシリアル番号になります

6.10.4.5. AWS IoT Core と Amazon CloudWatch の設定を行う

AWS IoT Core に送信したデータを Amazon CloudWatch のダッシュボード上で可視化します。ここでは、CloudFormation を用いて AWS IoT Core と Amazon CloudWatch の設定を行います。

1. CloudFormation へ移動し、「スタックの作成」→「新しいリソースを使用(標準)」をクリックします。



2. 「テンプレートファイルのアップロード」で「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル（a6e-gw-container-cloudsetting-[VERSION].zip）AWS フォルダ内の `a6e_aws_cfn_template.yml` を選択し、「次へ」をクリックします。



3. スタック名を入力します。また、 「6.10.4.4. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する」 で取得したシリアル番号をパラメータに指定し、「次へ」をクリックします。

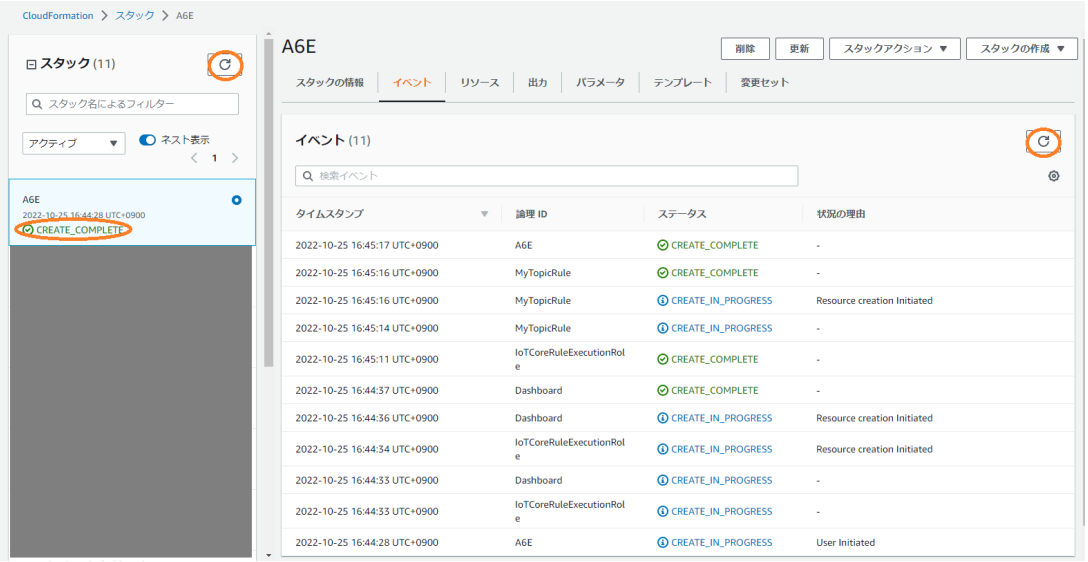


4. そのまま「次へ」をクリックします。

5. チェックボックスを選択し、「スタックの作成」をクリックします。



6. 作成したスタックのステータスが"CREATE_COMPLETE" になったら作成完了です。



6.10.4.6. 設定に必要となるパラメータを取得する

「3.16.4.2. 接続先クラウド情報の設定」 で設定するパラメータを取得します。

1. AWS IoT Core エンドポイント

    1. IoT Core へ移動し、サイドバー下部にある設定をクリックします。

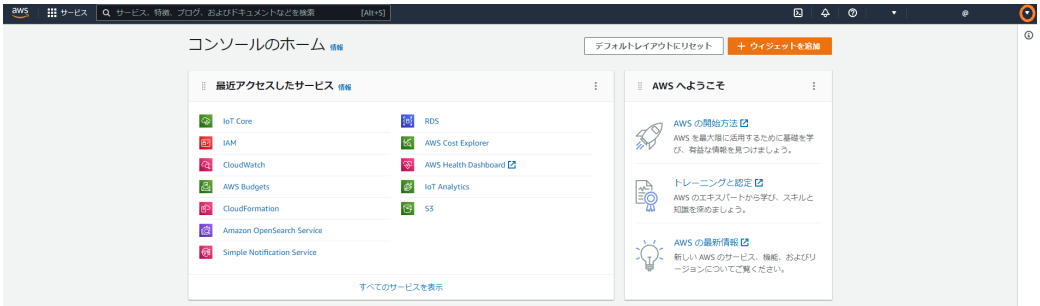


2. IoT Core エンドポイントが表示されます。

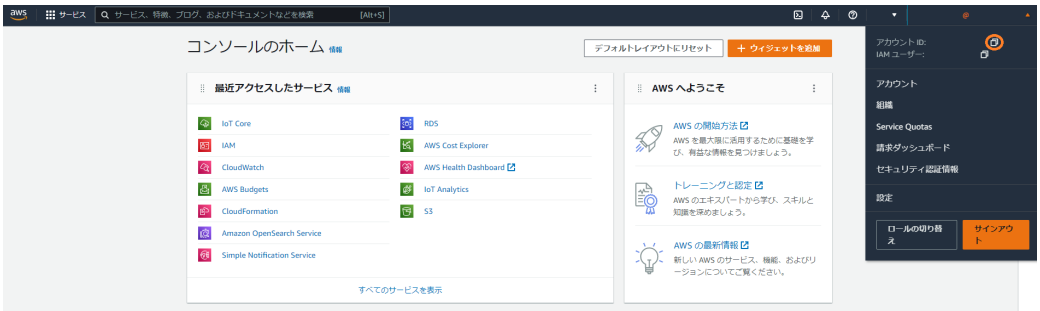


2. アカウント ID

1. AWS コンソール画面右上の ▼ をクリックします。



2. 下記画像の丸で囲んだマークをクリックすると、コピーすることができます。



6.10.5. 接続先の クラウド 環境を構築 (Azure)

Azure の場合は、 Azure IoT Hub にデータを送信します。本項では、 Azure portal 上で実施する設定を記載します。

手順中で使用するファイルは、Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) にアップロードしています。

6.10.5.1. Microsoft アカウントを作成する

Microsoft アカウントの作成については、Microsoft 公式ページ https://account.microsoft.com/ を参照してください。なお、サブスクリプションの設定も必要となります。

6.10.5.2. リソースグループを作成する

リソースグループの作成を行います。

1. Azure portal から [リソース グループ] を開き、[作成] を選択します。
2. サブスクリプションとリージョンを選択し、リソースグループ名を入力の後、[確認および作成] を選択します。



6.10.5.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う

ここでは、データの送信先となる Azure IoT Hub と、デバイスプロビジョニングのヘルパーサービスである Azure IoT Hub Device Provisioning Service (以降、DPS と記載) の設定を行います。



以下の手順はアットマークテクノが提供する設定ファイルを用いて設定を行っていますが、Azure portal で作成した Azure IoT Hub / DPS に接続することも可能です。DPS の個別登録機能を用いてデバイスプロビジョニングを行うため、以下のドキュメントを参考に DPS の設定を行ってください。 https://learn.microsoft.com/ja-jp/azure/iot-dps/quick-create-simulated-device-x509?tabs=windows&pivots=programming-language-ansi-c#create-a-device-enrollment

なお、上記手順中でアップロードするプライマリ証明書は、Armadillo 上の /var/app/volumes/gw_container/device/cert/device_cert.pem を使用してください。

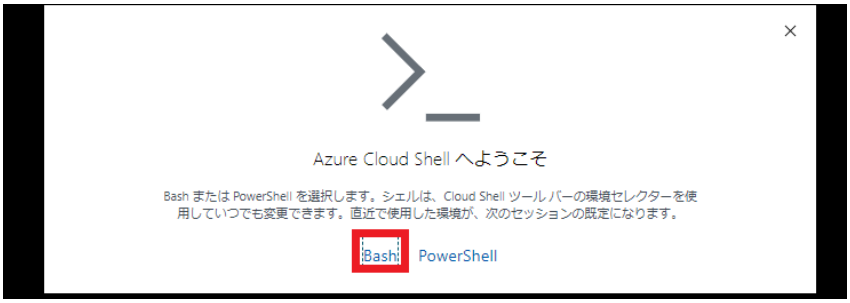


「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」v2.1.0 から、DPS のデバイスプロビジョニング方法が個別登録に変更となりました。v2.0.0 以前を使用してクラウド環境を構築および Azure portal で作成した DPS にグループ登録で設定を行った場合は、再度環境の構築および設定を行ってください。

1. Azure portal https://account.microsoft.com/ にサインインします。
2. Cloud Shell アイコンを選択し、 Azure Cloud Shell を起動します。



3. [Bash] を選択します。



4. ストレージアカウントの設定を行います。サブスクリプションを選択し、ストレージの作成をクリックすると自動的にストレージアカウントが作成されます。



5. Cloud Shell が起動したら、以下のコマンドで Armadillo-IoT ゲートウェイ A6E クラウド設定データをダウンロードします。

```
[Azure: ~]$ wget https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-a6e/
container/a6e-gw-container-cloudsetting-[VERSION].zip
[Azure: ~]$ unzip a6e-gw-container-cloudsetting-[VERSION].zip -d a6e-gw-container-cloud-
setting
[Azure: ~]$ cd a6e-gw-container-cloud-setting/Azure
```

図 6.95 Armadillo-IoT ゲートウェイ A6E クラウド設定データをダウンロードする

6. Cloud Shell 上でエディタを開き、コンフィグファイルを編集します。

```
[Azure: ~]$ code a6e_azure_create_hubdps.conf
# Common Config
resourceGroup="" ❶
certificateFilePath="./device_cert.pem"

# IoT Hub Config
iotHubName="" ❷
skuName="S1"
skuUnit=1
partitionCount=4

# DPS Config
provisioningServiceName="" ❸
```

図 6.96 コンフィグファイルを編集する

❶ リソースグループを指定します

❷ 作成する Azure IoT Hub 名を入力します

❸ 作成する DPS 名を入力します

```
# Common Config
resourceGroup="armadillo"
certificateFilePath="./device_cert.pem"

# IoT Hub Config
iotHubName="armadillo-iothub"
skuName="S1"
skuUnit=1
partitionCount=4

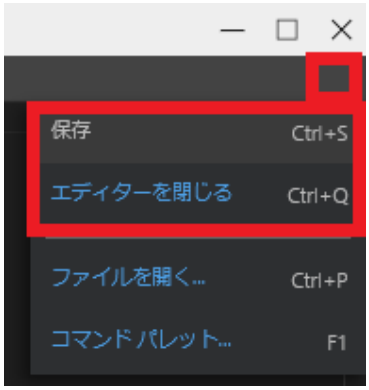
# DPS Config
provisioningServiceName="armadillo-dps"
```

図 6.97 コンフィグファイル設定例



Azure IoT Hub 名、 DPS 名はそれぞれグローバルで一意である必要があります。既に使用されている名称を指定した場合、エラーとなります。

コンフィグファイルの編集が終了したら、[保存] を行い、 [エディターを閉じる] を選択し、エディタを終了します。

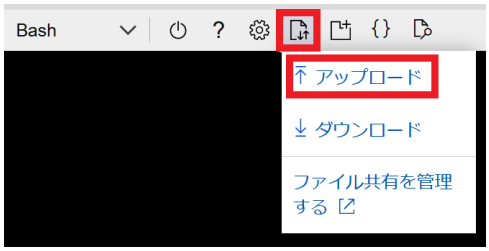


7. DPS に登録する証明書を Cloud Shell にアップロードします。

   証明書ファイルは Armadillo 上の /var/app/volumes/gw_container/device/cert/device_cert.pem を使用します。

> ゲートウェイアプリケーションのプロジェクト v1.1.0 以降を使用すると、 VS Code のタスクを使用してデバイス証明書を取得することができます。手順詳細は 「3.16.6.1. ゲートウェイコンテナアプリケーションが使用するデバイス証明書の取得」 をご確認ください。

   開発 PC にコピーした後、Cloud Shell の以下のアイコンを選択し、アップロードを行います。



   アップロード完了後、スクリプトと同階層に証明書ファイルをコピーします。

```
[Azure: ~]$ cp /home/<ユーザー名>/device_cert.pem .
```

8. 設定スクリプトを実行し、 Azure IoT Hub と DPS の設定を行います。

```
[Azure: ~]$ chmod +x a6e_azure_create_hubdps.sh
[Azure: ~]$ ./a6e_azure_create_hubdps.sh
Starting to create IoT Hub.
: (省略)
Starting to create DPS.
{
: (省略)
  "name": "xxxxx",
  "properties": {
: (省略)
```

```
    "idScope": "0ne12345678", ❶
: (省略)
  },
: (省略)
}
: (省略)
Starting to link between IoT Hub and DPS.
: (省略)
Starting to create enrollment.
: (省略)
Completed!
```

図 6.98 Azure IoT Hub と DPS の設定を実行する

❶ 環境設定時に使用するため、控えておきます

6.10.6. 接続先の クラウド 環境を構築 (SORACOM)

SORACOM を利用する場合は、データを SORACOM Harvest Data に送信します。本項では SORACOM Harvest Data を有効にする手順を説明します。

6.10.6.1. SORACOM アカウントを作成する

SORACOM アカウントの作成については、SORACOM 公式ページ https://users.soracom.io/ja-jp/guides/getting-started/create-account/ を参照して、SORACOM コンソールにログインできるアカウントを用意してください、なお、支払い方法についての設定も必要となります。

6.10.6.2. SIM を申し込む

SIM の申し込み方法については、SORACOM 公式ページ https://users.soracom.io/ja-jp/guides/getting-started/buy/ を参照してください。

6.10.6.3. SIM を SORACOM ユーザーコンソールに登録する

SIM を SORACOM ユーザーコンソールに登録する方法については、SORACOM 公式ページ https://users.soracom.io/ja-jp/guides/getting-started/register-sim/ を参照してください。

6.10.6.4. SIM 接続設定

SIM を装着し、接続設定を行ってください。Armadillo に電源が入っている場合は一度電源を切り、SIM を装着してから再起動してください。

6.10.6.5. SORACOM Harvest Data の有効化

SORACOM Harvest Data は、SIM グループ単位で機能の有効/無効を設定します。 SORACOM コンソールにログインして、下記の手順で設定を行ってください。

1. SORACOM コンソールメニューの「SIM 管理」を開きます。



2. SIM 一覧から使用する SIM にチェックを入れ、「操作」>「所属グループ変更」を選択します。



3. 「新しい所属グループ」から「新しいグループを作成…」を選択します。



4. コンソールメニューの「SIM グループ」を選択し、SIM グループ管理画面を開きます。



5. 「SORACOM Harvest Data 設定」を ON にして「保存」をクリックします。



6. 「SORACOM Harvest Data を有効にしますか」が表示されるので「有効にする」をクリックします。



7. 以上で指定した SIM から SORACOM Harvest Data へのデータ送信が有効になります。データの確認は 「6.10.8.4. SORACOM 上でのデータ確認」 を参照ください。

6.10.7. ゲートウェイコンテナの設定ファイル

利用したい内容に合わせて、 設定ファイルを編集します。設定内容はコンテナ起動時の内容が適用されるため、一度コンテナを終了させます。

```
[armadillo ~]# podman stop a6e-gw-container
a6e-gw-container
```

図 6.99 ゲートウェイコンテナを終了する



本マニュアルに記載しているゲートウェイコンテナの設定ファイルの内容は、最新バージョンの内容となります。

ご利用のゲートウェイコンテナのバージョンが最新ではない場合、ゲートウェイコンテナを最新のバージョンにアップデートするか、ゲートウェイコンテナのバージョンに対応した製品マニュアルをご参照ください。

製品マニュアルのバージョンとゲートウェイコンテナのバージョンについては Armadillo-IoT A6E の製品アップデートページをご参照ください。

設定ファイルの内容は「3.16.4.2. 接続先クラウド情報の設定」及び「3.16.4.3. インターフェース設定」を参照ください。

6.10.8. コンテナ起動・実行

設定ファイルの修正が完了したら、コンテナを起動します。コンテナが起動すると、設定に従ってコンテナ内のアプリケーションが実行される仕組みとなっています。

```
[armadillo ~]# podman_start a6e-gw-container
Starting 'a6e-gw-container'
a3b719c355de677f733fa8208686c29424be24e57662d3972bc4131ab7d145ad
```

「表 3.71. [DEFAULT] 設定可能パラメータ」 でクラウドにデータを送信する設定を行った場合は、クラウド接続後、アプリケーション LED の状態が点滅から点灯に変化します。

6.10.8.1. Armadillo からクラウドに送信するデータ

Armadillo からクラウドに送信するデータは以下の通りです。

- デバイス情報

表 6.13 デバイス情報データー覧

項目	概要
DevInfo_SerialNumber	シリアル番号
DevInfo_LAN_MAC_Addr	LAN MAC アドレス
DevInfo_ABOS_Ver	Armadillo Base OS バージョン
DevInfo_Container_Ver	コンテナイメージバージョン

- CPU 温度

表 6.14 CPU 温度データー覧

項目	概要
CPU_temp	CPU 温度

- 接点入力

表 6.15 接点入力データー覧

項目	概要
DI1_polling	DI1 のポーリング結果
DI2_polling	DI2 のポーリング結果
DI1_edge	DI1 のエッジ検出結果
DI2_edge	DI2 のエッジ検出結果

- 接点出力

  クラウドに送信するデータはありません。

- RS-485

表 6.16 RS-485 データー覧

項目	概要
RS485_Data1	RS485_Data1 の読み出し値
RS485_Data2	RS485_Data2 の読み出し値
RS485_Data3	RS485_Data3 の読み出し値
RS485_Data4	RS485_Data4 の読み出し値

- ユーザースイッチ

表 6.17 ユーザースイッチ関連データー覧

項目	概要
sw_state	ユーザースイッチの状態

クラウドにデータが届いているかどうかは、次項の方法で確認することができます。

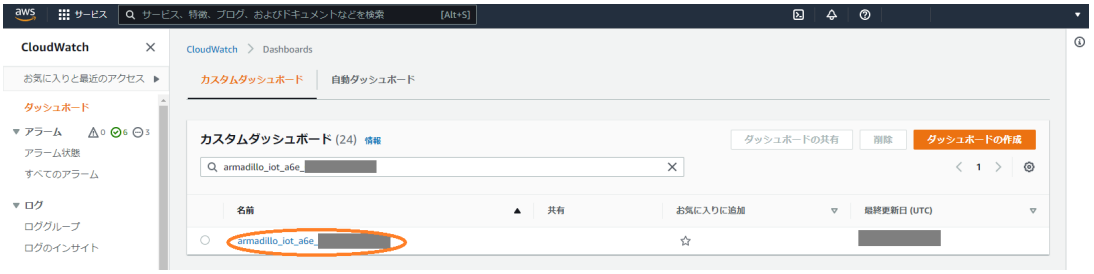
6.10.8.2. AWS 上でのデータ確認

Amazon CloudWatch ダッシュボードで、データが届いているかの確認を行う事ができます。

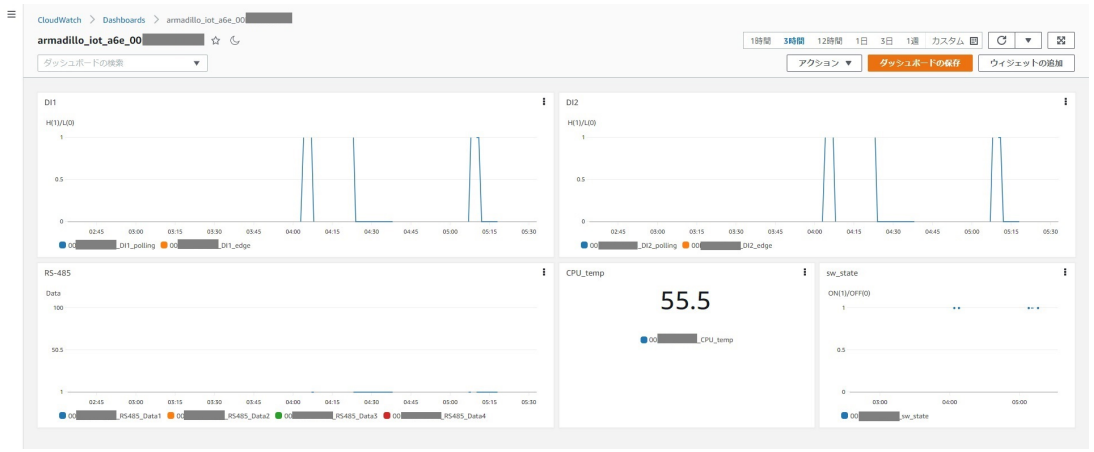
1. CloudWatch に移動し、「ダッシュボード」を選択します。



2. 「6.10.4.5. AWS IoT Core と Amazon CloudWatch の設定を行う」 で CloudWatch ダッシュボードが作成されています。ダッシュボード名は armadillo_iot_a6e_〈シリアル番号〉です。



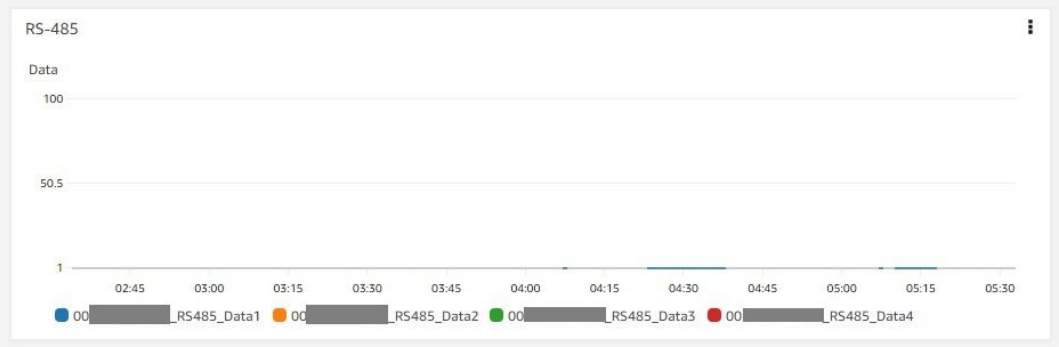
3. ダッシュボード名をクリックすると、下記のような画面が表示されます。



・ 接点入力



・RS-485



・CPU 温度



・ユーザースイッチ



また、実際にデバイスから届いているデータを確認する場合は、 AWS IoT Core の Device Shadow で確認を行います。

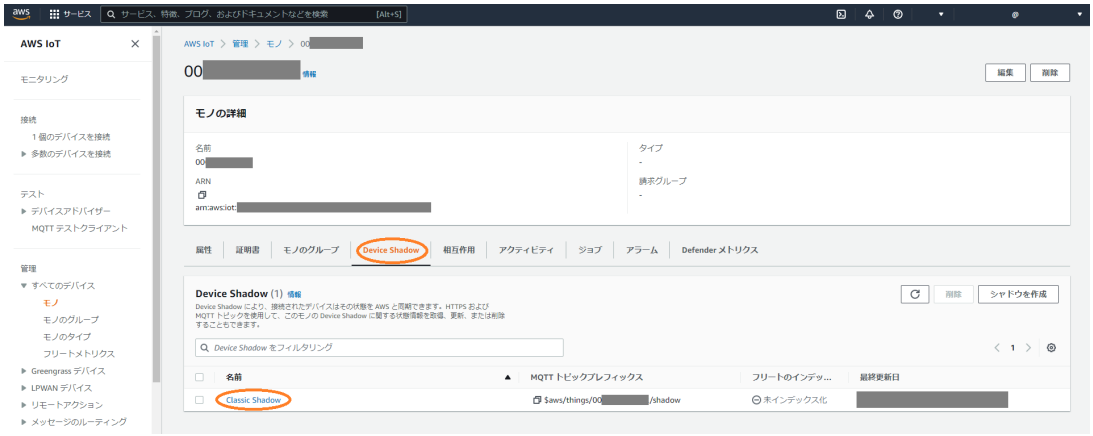
1. AWS IoT Core に移動し、「管理」→「すべてのデバイス」→「モノ」を選択します。



2. デバイスの名前は 「6.10.4.4. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する」で取得したシリアル番号で登録されています。



3. 「Device Shadow」の「Classic Shadow」を選択します。



4. 下記の通り、 Armadillo から送信されてきたデータを確認することができます。



6.10.8.3. Azure 上でのデータ確認



以下では可視化の手順を記載していますが、実際にデバイスから届いているデータを確認する場合は、Azure IoT Explorer を用いて確認することが可能です。詳細はこちらのドキュメント https://docs.microsoft.com/ja-jp/azure/iot-pnp/howto-use-iot-explorer をご参照ください。

Azure IoT Hub に登録されるデバイス ID は、デバイス認証に使用している証明書の CN となります。以下のコマンドで確認することが可能です。

```
[armadillo ~]# openssl x509 -noout -subject -in /var/app/volumes/
gw_container/device/cert/device_cert.pem | grep subject | awk '{print
$NF}'
```

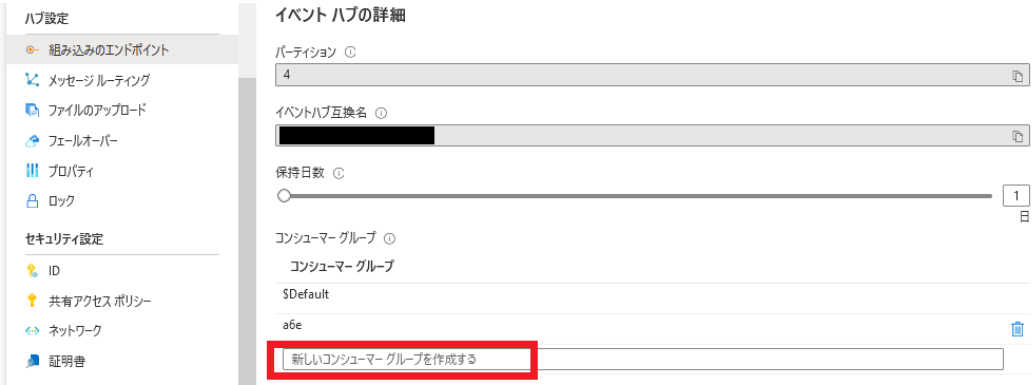
可視化の方法は様々ありますが、本書では一例として、Power BI を使用して Azure IoT Hub に送信したデータの可視化を行う方法を記載します。

以下の手順では、 「6.10.8.1. Armadillo からクラウドに送信するデータ」 のうち CPU_temp を例に記載します。

1. こちらのページで https://powerbi.microsoft.com/ja-jp/ Power BI アカウントを作成します。なお、 Pro アカウントでの登録が必要となります。

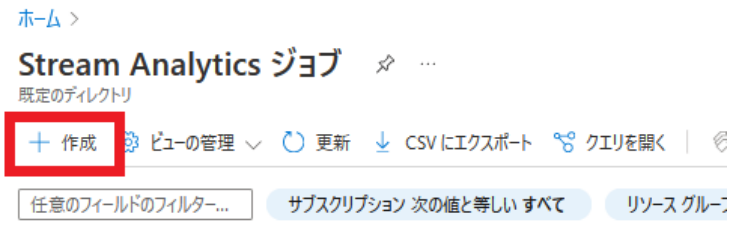
2. PowerBI にログインし、グループワークスペースを作成します。

3. Azure IoT Hub にコンシューマーグループを追加します。 Azure portal から [IoT Hub] を開き、「6.10.5.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 で作成した IoT Hub を選択します。[組み込みのエンドポイント] を選択し、[コンシューマーグループ] の下のテキストボックスに、新しいコンシューマーグループの名前を入力、保存します。



4. Azure IoT Hub のデータを Power BI のデータセットにルーティングする Azure Stream Analytics ジョブを作成します。

   Azure portal から [Stream Analytics ジョブ] を開き、 [Stream Analytics ジョブ] 概要ページで [作成] を選択します。



[基本] タブに、 「表 6.18. Azure Stream Analytics ジョブ設定値」 の情報を入力し、 [確認と作成] を選択した後、 [作成] を選択して Stream Analytics ジョブを作成します。

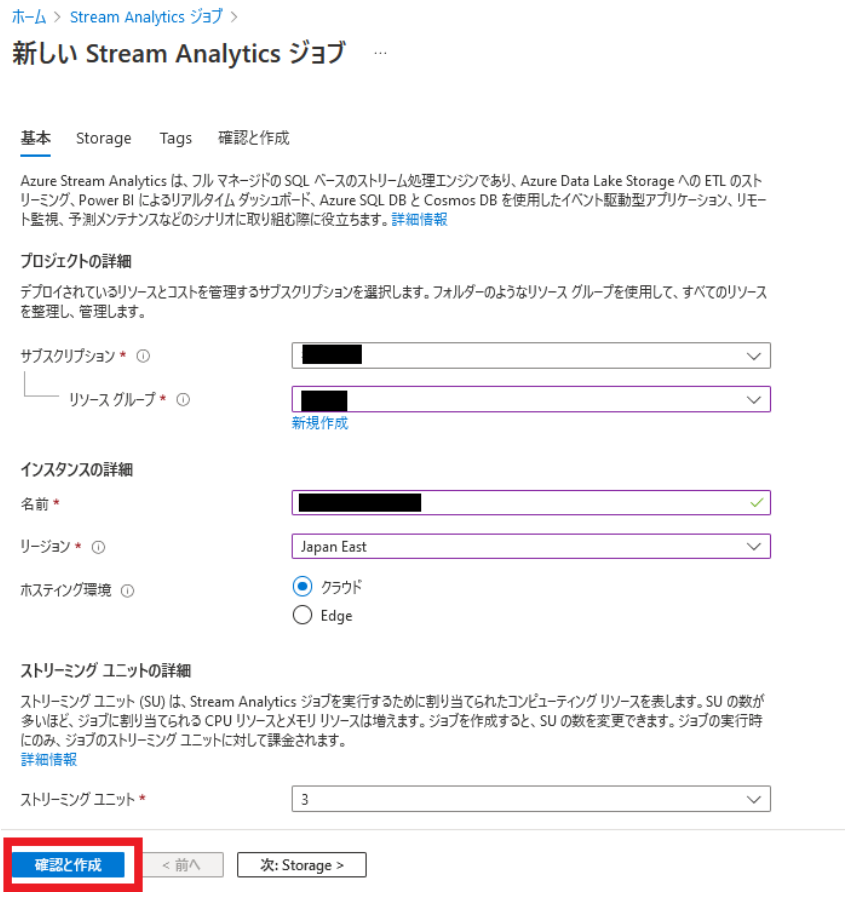


表 6.18 Azure Stream Analytics ジョブ設定値

項目	設定値
サブスクリプション	IoT Hub のサブスクリプション
リソースグループ	IoT Hub のサブスクリプション
名前	ジョブの名前(任意)
リージョン	IoT Hub のリージョン

5. Stream Analytics ジョブに入力を追加します。

   作成した Stream Analytics ジョブを開きます。



[ジョブ トポロジ] - [入力] から [ストリーム入力の追加] を選択し、ドロップダウンリスト内の [IoT Hub] を選択します。



「表 6.19. Azure Stream Analytics ジョブ入力設定値」 の情報を入力し、それ以外の内容はデフォルトのまま [保存] を選択します。

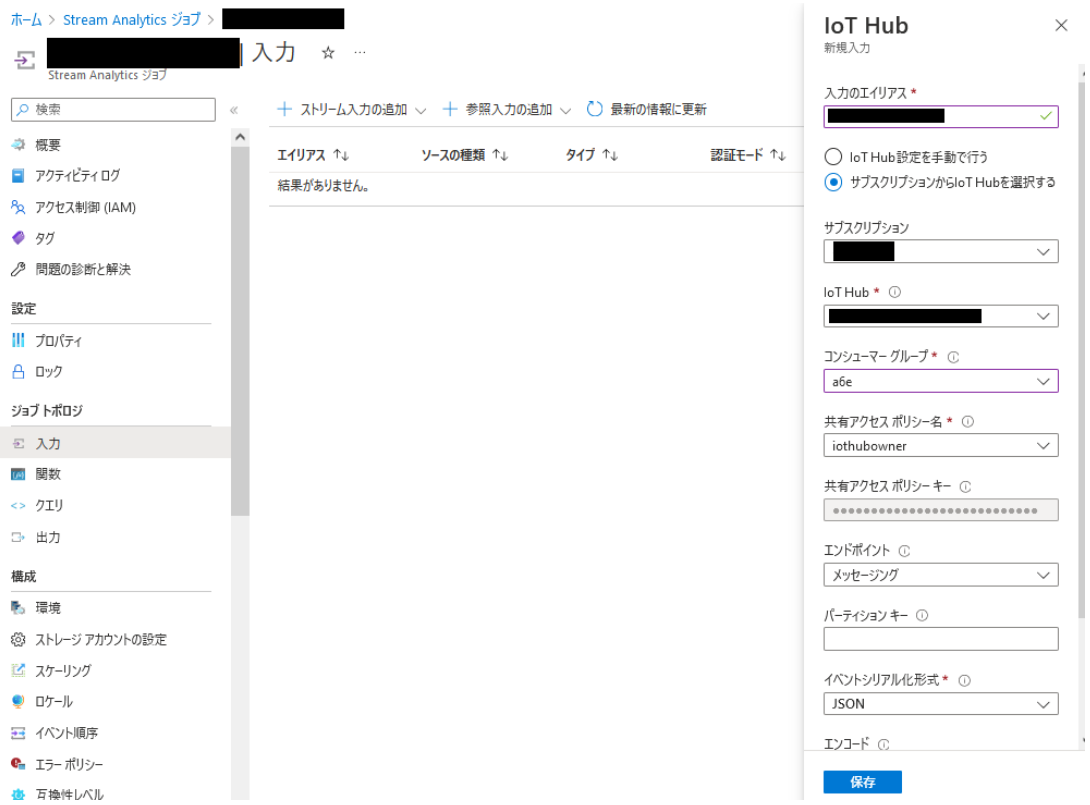
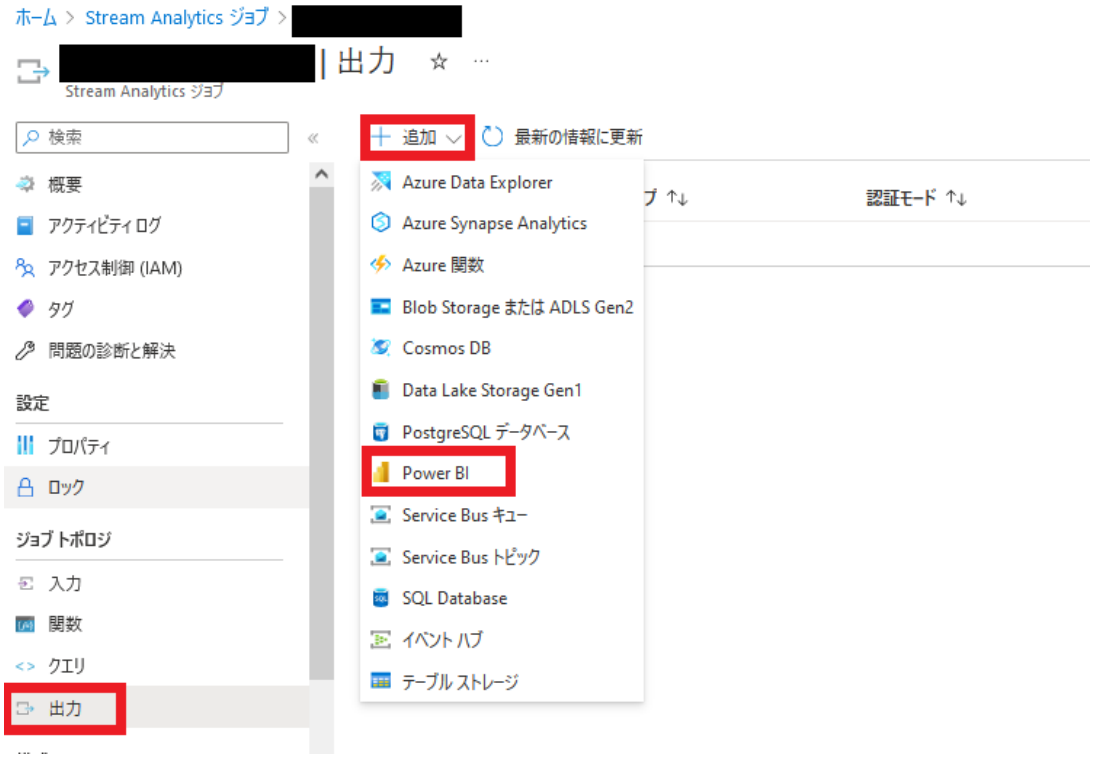


表 6.19 Azure Stream Analytics ジョブ入力設定値

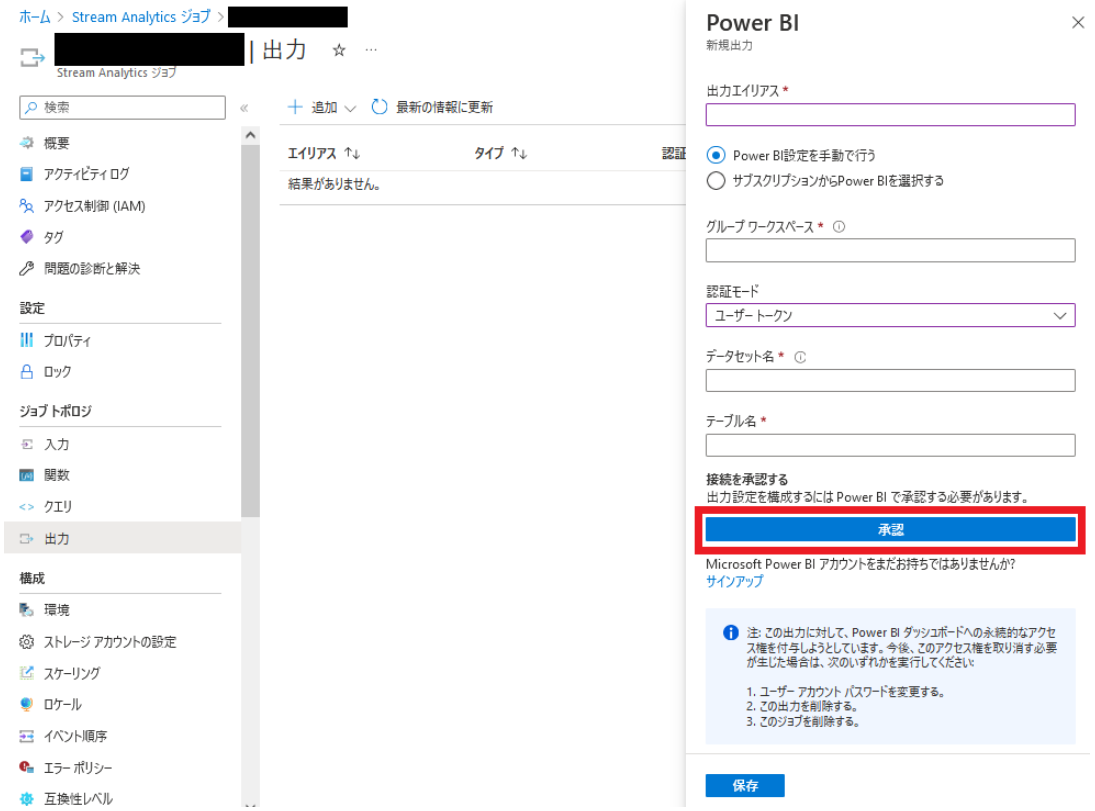
項目	設定値
入力のエイリアス	一意の名前を入力
サブスクリプションから IoT Hub を選択する	選択
サブスクリプション	IoT Hub 用のサブスクリプション
IoT Hub	使用する IoT Hub
コンシューマーグループ	作成したコンシューマーグループを選択
共有アクセスポリシー名	iothubowner

6. Stream Analytics ジョブに出力を追加します。なお、複数の値を PowerBI で可視化する場合は、値の数分の出力設定が必要になります。

   [ジョブ トポロジ] - [出力] から [追加] を選択し、ドロップダウンリスト内の [Power BI] を選択します。

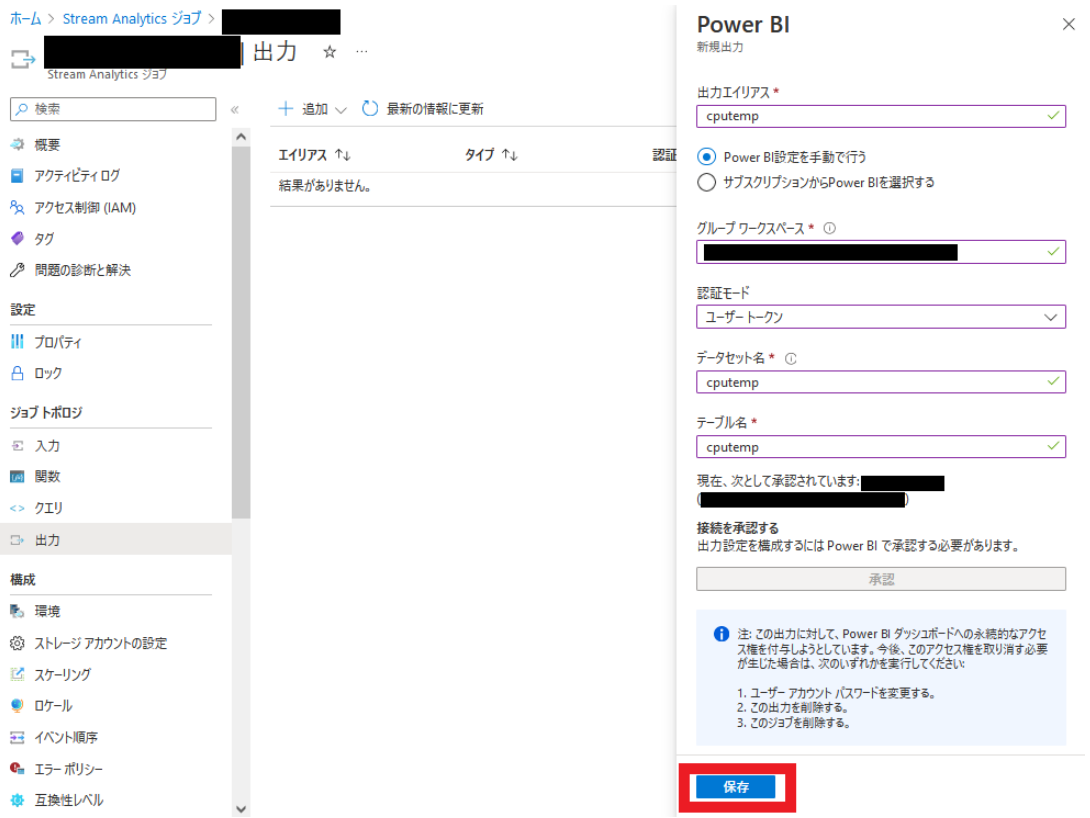


[認証モード] で「ユーザートークン」を選択、[接続を承認する] の [承認] を選択し、Power BI アカウントにサインインします。



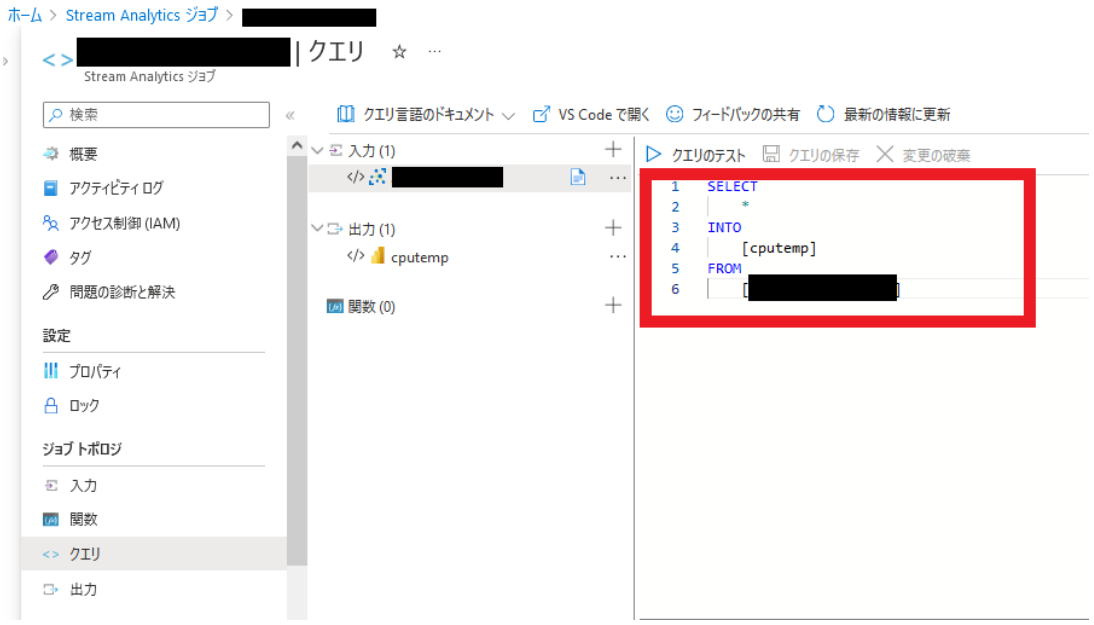
作成したグループワークスペースの ID を [グループワークスペース] に入力します。グループワークスペースの ID は、グループワークスペースの URL から取得することができます。[デー

タセット名] と [テーブル名] は任意の値を指定してください。ここではそれぞれ cputemp を指定しています。情報登録完了後、 [保存] を選択します。



7. Stream Analytics ジョブのクエリを構成します。

   [ジョブ トポロジ] の [クエリ] を選択します。



赤枠内にクエリを指定します。入力完了後、[クエリの保存] を選択してください。フォーマットは下記の通りです。 <パラメータ名> には、 「6.10.8.1. Armadillo からクラウドに送信するデータ」 の「項目」を指定してください。

```
SELECT
    <パラメータ名>,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [<ジョブ出力エイリアス名>]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE <パラメータ名> IS NOT NULL
```

これに従い、CPU_temp の場合は以下の通りとなります。

```
SELECT
    CPU_temp,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [cputemp]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE CPU_temp IS NOT NULL
```

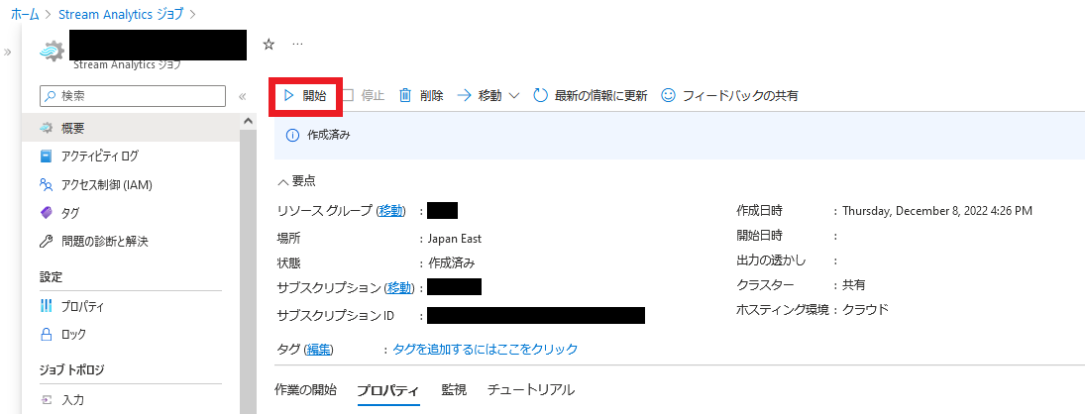
なお、複数の出力がある場合は、クエリ入力欄に下記の通り複数のクエリを列挙してください。INTO 句で指定するパラメータ(データセット名)が異なることに注意してください。

```
SELECT
    CPU_temp,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [cputemp]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE CPU_temp IS NOT NULL

SELECT
    DI1_polling,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [di1polling]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE DI1_polling IS NOT NULL
```

8. Stream Analytics ジョブを実行します。

   [概要] 画面で [開始] を選択します。



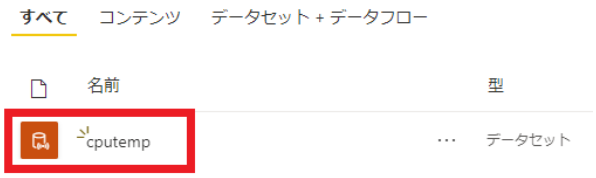
[ジョブの開始] 画面の [ジョブ出力の開始時刻] で [現在] が選択されていることを確認し、 [開始] を選択します。ジョブが正常に開始されると、[概要] 画面の [状態] が [実行中] に変わります。



9. ゲートウェイコンテナを停止している場合、下記のコマンドを実行しゲートウェイコンテナを開始します。

```
[armadillo ~]# podman_start a6e-gw-container
Starting 'a6e-gw-container'
a3b719c355de677f733fa8208686c29424be24e57662d3972bc4131ab7d145ad
```

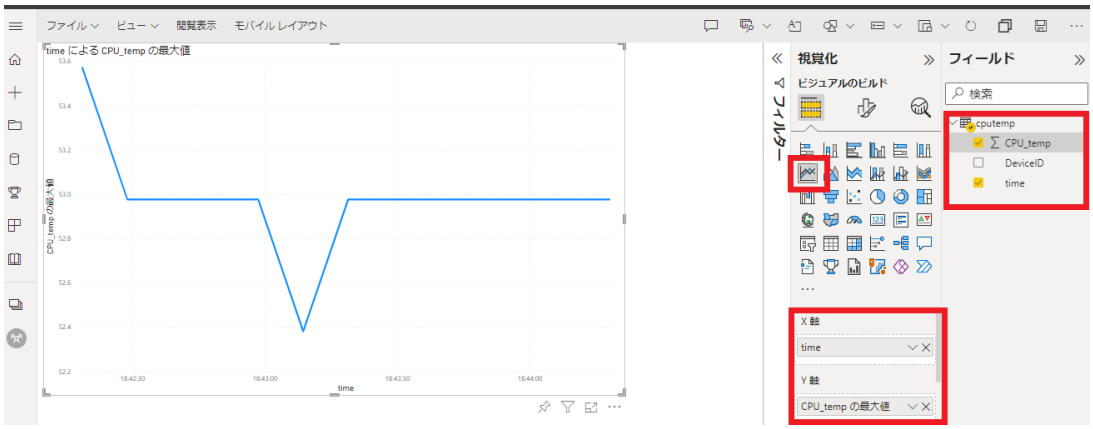
10. PowerBI アカウントにサインインし、使用したワークスペースを右側のメニューから選択すると、 Stream Analytics ジョブ出力で指定した名称のデータセットが作成されています。



11. データセットの [レポートの作成] を選択します。



12. [視覚化] で [折れ線グラフ] を選択、X 軸に EventEnqueuedUtcTime 、 Y 軸に CPU_temp を指定することにより、グラフ化を行うことが出来ます。各設定を行った後、 [保存] すると、レポートが作成されます。



13. 複数のデータセットが存在している場合は、それぞれについてレポートの作成を行います。なお、各レポートを一括して表示したい場合はダッシュボード機能を選択してください。手順についてはこちらのドキュメント https://learn.microsoft.com/ja-jp/power-bi/create-reports/service-dashboard-create を参照してください。

6.10.8.4. SORACOM 上でのデータ確認

SORACOM Harvest Data 上で、データが届いているかの確認を行う事ができます。また、SORACOM Lagoon を用いての可視化が可能です。

SORACOM Harvest Data 上での確認

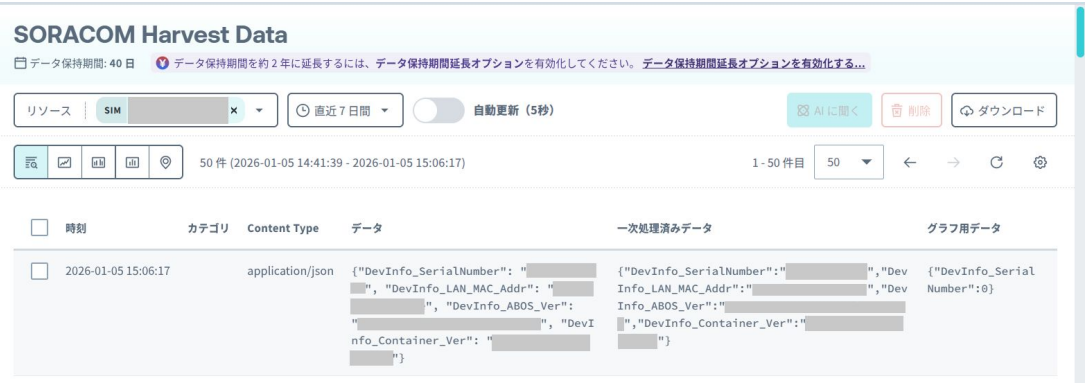
1. SORACOM コンソールメニュー内 「データ収集・蓄積・可視化」 > 「SORACOM Harvest Data」 を選択します。



2. 「リソース」横の ▼ を押して、データを確認したい SIM を選択します。



3. データの一覧が表示されます。



SORACOM Lagoon のダッシュボード設定

SORACOM Harvest に送られたデータについて、SORACOM Lagoon ダッシュボードに表示することが可能です。設定したグループに届いたすべてのデータを可視化する設定を行います。

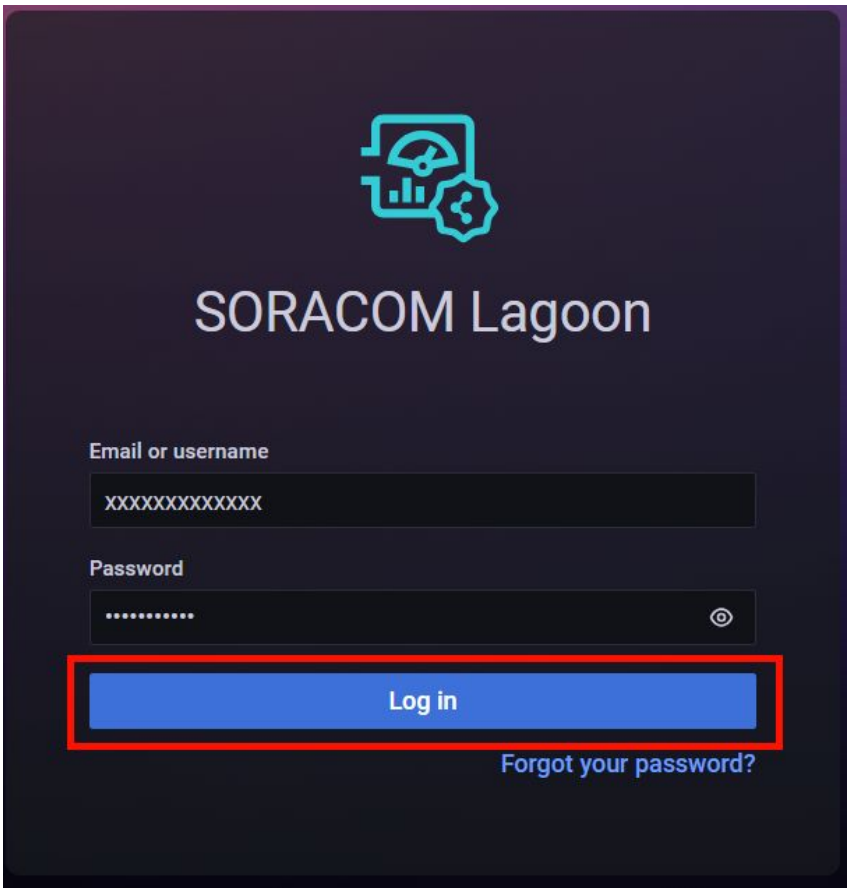
1. SORACOM コンソールメニュー内 「データ収集・蓄積・可視化」 > 「SORACOM Lagoon」を選択します。



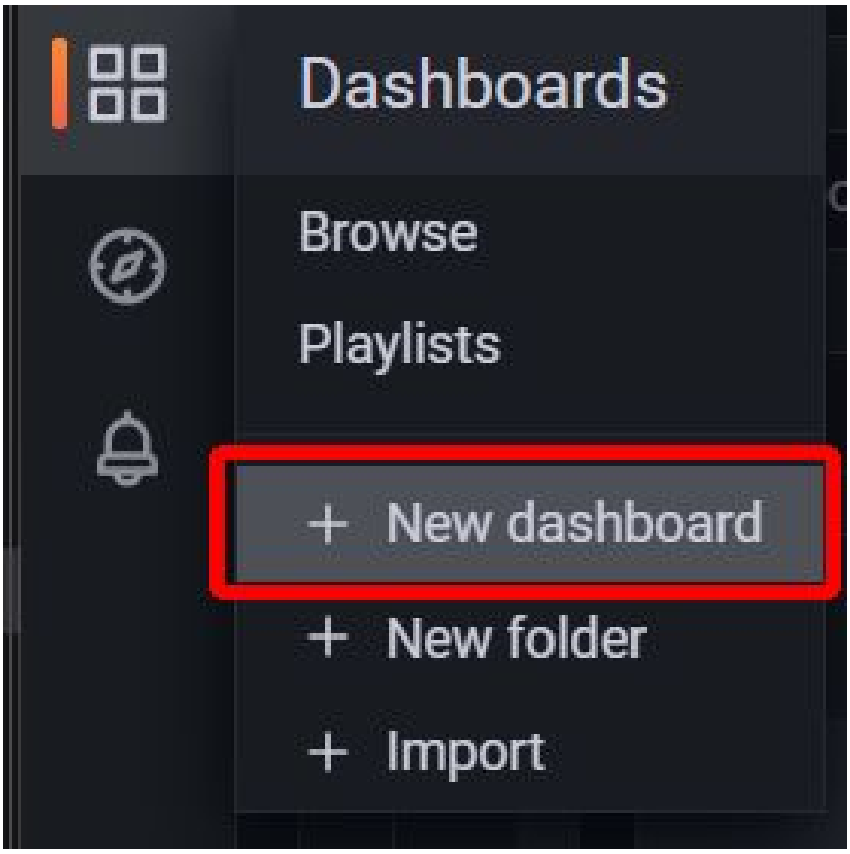
2. 「SORACOM Lagoon 3 コンソール」 を選択します。



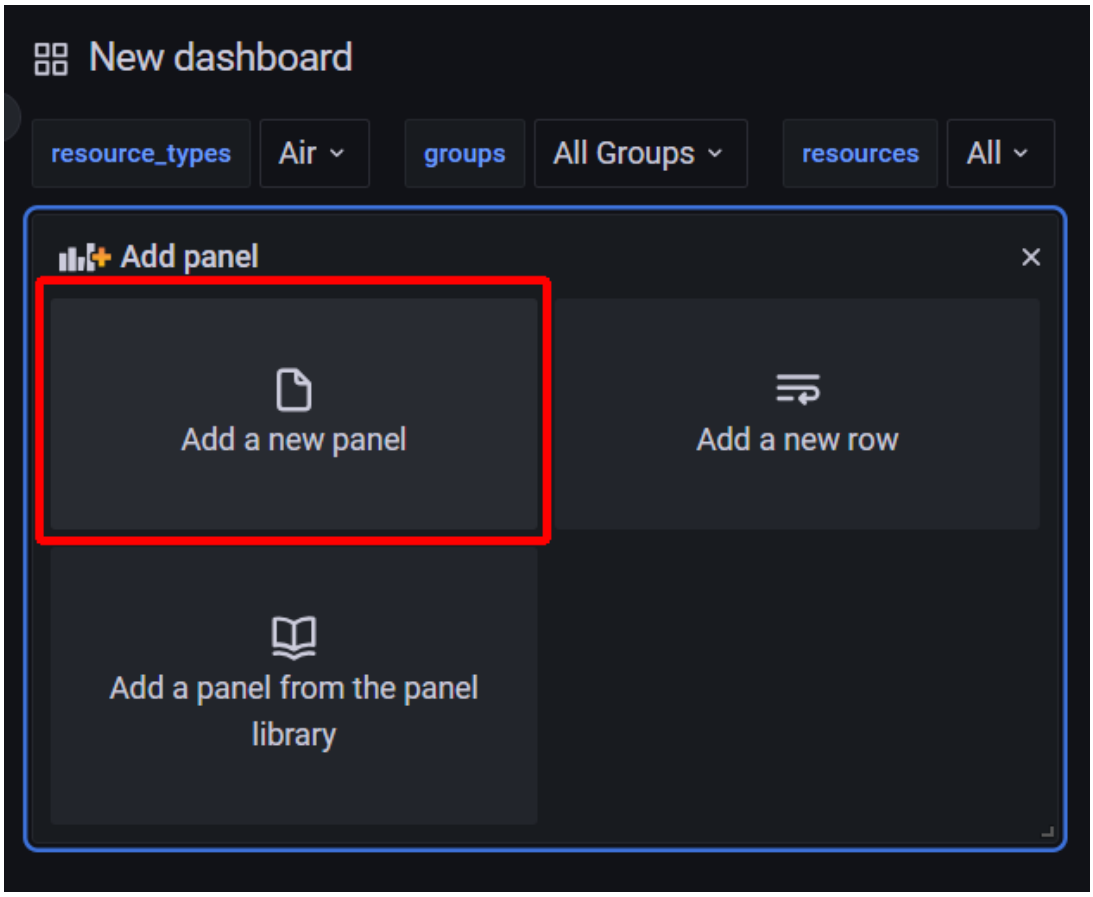
3. SORACOM Lagoon コンソールのログインページが開くので、SORACOM アカウントのメールアドレスとパスワードを入力してログインしてください。



4. 「Dashboards」>「+ New Dashboard」 を選択します。



5. 「Add a new panel」を選択します。



6. 保存したダッシュボードは一覧に表示されるので、選択して表示します。

6.10.9. クラウドからの操作

6.10.9.1. クラウドからのデータ設定

各インターフェースの設定については、 「3.16.4.3. インターフェース設定」 に記載している通り Armadillo 上の設定ファイルで行いますが、クラウドから設定値を変更することも可能です。

なお、クラウドからデータ設定を行うためには、 「表 3.71. [DEFAULT] 設定可能パラメータ」 の `cloud_config` を true に設定する必要があります。

設定を変更できる項目は以下の通りです。

- 接点入力設定
- 接点出力設定
- RS-485 レジスタ読み出し

下記の手順でデータを設定します。

- AWS

  AWS IoT Core の Device Shadow を更新して設定を行います。

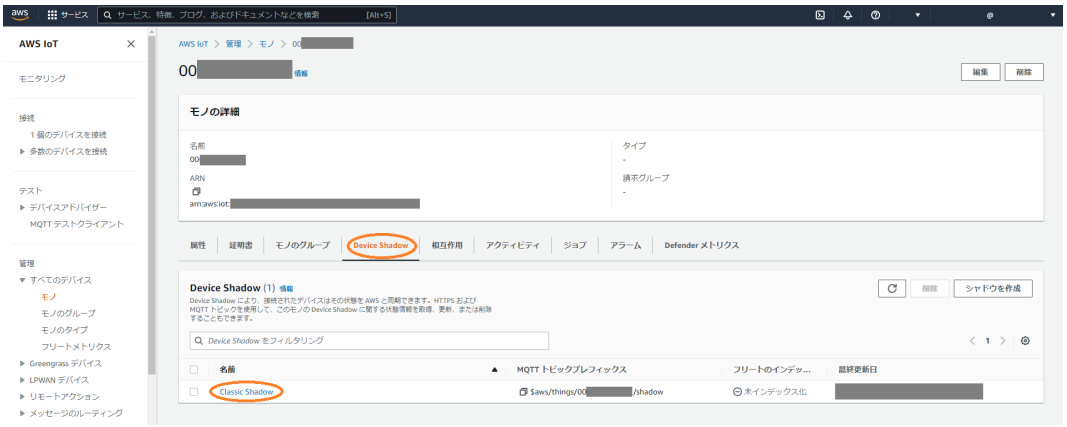
  1. AWS IoT Core に移動し、「管理」→「すべてのデバイス」→「モノ」を選択します。



2. デバイスの名前は 「6.10.4.4. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する」で取得したシリアル番号で登録されています。



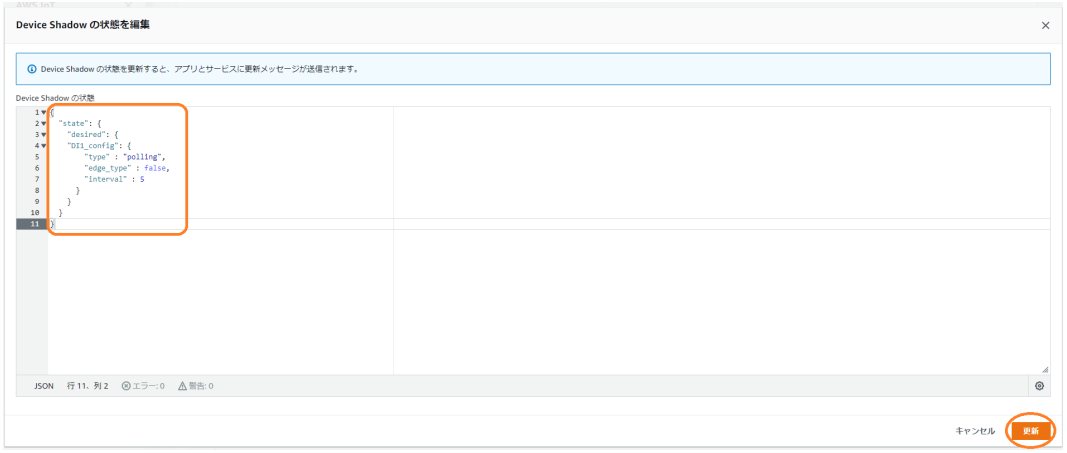
3. 「Device Shadow」の「Classic Shadow」を選択します。



4. Device Shadow ドキュメントの「編集」を選択します。



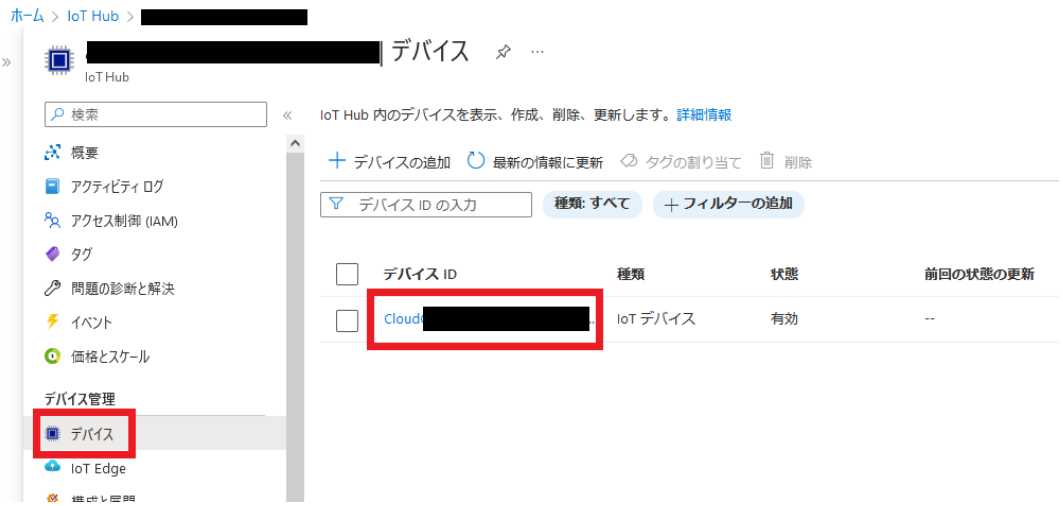
5. 入力画面が表示されるため、設定データを入力し「更新」をクリックします。



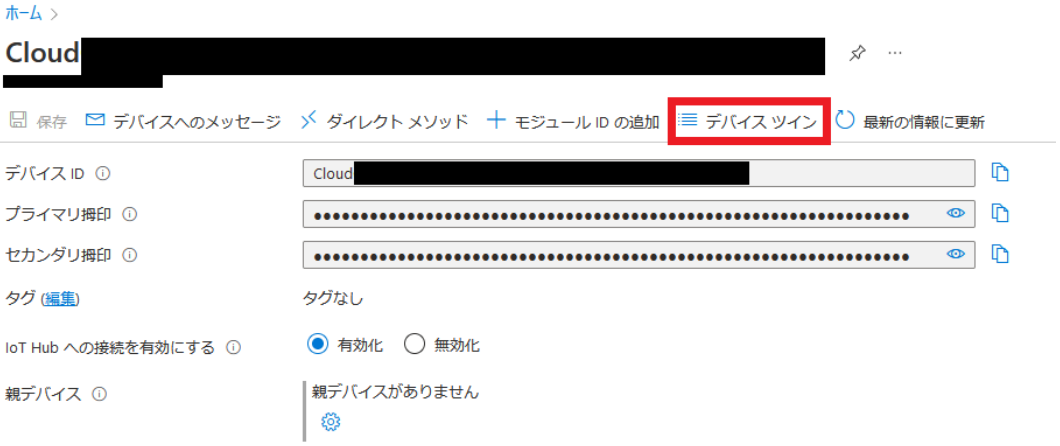
- Azure

  Azure IoT Hub のデバイスツインを更新して設定を行います。

  1. Azure portal から [IoT Hub] を開き、「6.10.5.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 で作成した IoT Hub を選択します。[デバイス] を選択し、一覧の中から該当するデバイス ID を選択します。



  2. [デバイスツイン] を選択します。



3. デバイスツイン編集画面が表示されるため、設定データを入力し「保存」をクリックします。



各機能それぞれ、下記の通りのフォーマットとなっています。

・ 接点入力設定

表 6.20 接点入力設定値

項目	概要	設定値	内容
type	動作種別	(空欄) or none	接点入力状態取得を行わない
		polling	ポーリング
		edge	エッジ検出

項目	概要	設定値	内容
edge_type	エッジ検出設定	falling	立ち下がりエッジ
		rising	立ち上がりエッジ
		both	両方
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "<制御ポート>_config": { ❶
        "type" : <polling or edge>,
        "edge_type" : <falling or rising or both>,
        "interval" : <読み出し間隔>
      }
    }
  }
}
```

❶ 制御ポートは DI1, DI2 のいずれかを指定してください

```
{
  "state": {
    "desired": {
    "DI1_config": {
        "type" : "polling",
        "edge_type" : falling,
        "interval" : 5
      }
    }
  }
}
```

図 6.100 接点入力制御シャドウ設定例

・Azure

フォーマットは下記の通りです。デバイスツインの "desired" プロパティに設定します。

```
{
  "properties": {
    "desired": {
    "<制御ポート>_config": { ❶
        "type" : <polling or edge>,
        "edge_type" : <falling or rising or both>,
        "interval" : <読み出し間隔>
      },
      :
    }
```

```
  }
}
```

❶ 制御ポートは DI1, DI2 のいずれかを指定してください

```
{
  "properties": {
    "desired": {
    "DI1_config": {
        "type" : "polling",
        "edge_type" : falling,
        "interval" : 5
      },
```

図 6.101 接点入力制御デバイスツイン設定例

・接点出力設定

クラウドから設定内容を受信したタイミングで接点出力動作を停止し、設定内容を更新します。

表 6.21 接点出力設定値

項目	概要	設定値	内容
output_state	出力状態	high	High
		low	Low
output_time	出力時間[sec]	1～3600	出力コマンド実行後に output_state で指定したレベルを出力する時間。 0 を指定すると出力値を固定します。
output_delay_time	出力遅延時間[sec]	0～3600	出力コマンド実行後、指定した時間遅延して出力します。

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "<制御ポート>_config": { ❶
        "output_state" : <high or low>,
        "output_time" : <出力時間>,
        "output_delay_time" : <出力遅延時間>
      }
    }
  }
}
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

```
{
  "state": {
```

```
    "desired": {
    "DO1_config": {
        "output_state" : "high",
        "output_time" : 10,
        "output_delay_time" : 10
      }
    }
  }
}
```

図 6.102 接点出力制御シャドウ設定例

・Azure

フォーマットは下記の通りです。デバイスツインの "desired" プロパティに設定します。

```
{
  "properties": {
    "desired": {
    "<制御ポート>_config": { ❶
        "output_state" : <high or low>,
        "output_time" : <出力時間>,
        "output_delay_time" : <出力遅延時間>
      },
      :
    }
  }
}
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

```
{
  "properties": {
    "desired": {
      "DO1_config": {
        "output_state" : "high",
        "output_time" : 10,
        "output_delay_time" : 10
      },
```

図 6.103 接点出力制御デバイスツイン設定例

・RS-485 レジスタ読み出し

表 6.22 RS-485 レジスタ読み出し設定値

項目	概要	設定値	内容
method	通信種別	none	RS-485 を利用しない
		rtu	Modbus-RTU
data_size	データサイズ	7 or 8	
baudrate	ボーレート	300～921600[bps]	通信速度を指定します

項目	概要	設定値	内容
parity	パリティビット	none	None
		odd	Odd
		even	Even
stop	ストップビット	1	1
		2	2
device_id	Modbus スレーブ機器のデバイス ID	1 ～ 247	
func_code	ファンクションコード	1 ～ 4	
register_addr	レジスタアドレス	機器依存	値を読み出すレジスタのアドレスを指定
register_count	読み出しレジスタ数	1 or 2	一度に読み出すレジスタ数を指定
endian	エンディアン設定	little	リトルエンディアン
		big	ビッグエンディアン
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します
data_offset	読み出し値に加算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値に加算する値を指定します
data_multiply	読み出し値と乗算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と乗算する値を指定します
data_divider	読み出し値と除算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と除算する値を指定します

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
      "RS485_Data<1～4>_config": { ❶
        "method" : <種別>,
        "baudrate" : <ボーレート>,
        "data_size": <データサイズ>,
        "parity" : <パリティ>,
        "stop" : <ストップビット>,
        "device_id" : <デバイス ID>,
        "func_code" : <ファンクションコード>,
        "register_addr" : <レジスタアドレス>,
        "register_count" : <読み出すレジスタ数>,
        "endian" : <エンディアン種別>,
        "interval" : <読み出し間隔>,
        "data_offset" : <データに加算する値>,
        "data_multiply" : <データに乗算する値>,
        "data_divider" : <データと除算する値>
      }
    }
  }
}
```

❶ 1～4 のいずれかを指定してください

```
{
  "state": {
    "desired": {
      "RS485_Data1_config": {
        "baudrate" : 9600,
        "parity" : "none",
        "stop" : 1,
        "device_id" : 1,
        "func_code" : 3,
        "register_addr" : 0,
        "register_count" : 2,
        "endian" : "big",
        "interval" : 30,
        "data_offset" : 0,
        "data_multiply" : 0,
        "data_divider" : 0
      }
    }
  }
}
```

図 6.104 RS-485 レジスタ読み出しシャドウ設定例

・ Azure

フォーマットは下記の通りです。デバイスツインの "desired" プロパティに設定します。

```
{
  "properties": {
    "desired": {
      "RS485_Data<1～4>_config": { ❶
        "method" : <種別>,
        "baudrate" : <ボーレート>,
        "data_size": <データサイズ>,
        "parity" : <パリティ>,
        "stop" : <ストップビット>,
        "device_id" : <デバイス ID>,
        "func_code" : <ファンクションコード>,
        "register_addr" : <レジスタアドレス>,
        "register_count" : <読み出すレジスタ数>,
        "endian" : <エンディアン種別>,
        "interval" : <読み出し間隔>,
        "data_offset" : <データに加算する値>,
        "data_multiply" : <データに乗算する値>,
        "data_divider" : <データと除算する値>
      },
      :
    }
  }
}
```

❶ 1～4 のいずれかを指定してください

```
{
  "properties": {
```

```
    "desired": {
      "RS485_Data1_config": {
        "baudrate" : 9600,
        "parity" : "none",
        "stop" : 1,
        "device_id" : 1,
        "func_code" : 3,
        "register_addr" : 0,
        "register_count" : 2,
        "endian" : "big",
        "interval" : 30,
        "data_offset" : 0,
        "data_multiply" : 0,
        "data_divider" : 0
      },
```

図 6.105 RS-485 レジスタ読み出しデバイスツイン設定例

6.10.9.2. クラウドからのデバイス制御

以下について、クラウドからデバイスの動きを制御することができます。

- 接点出力制御
  - 接点出力開始/終了の制御
- RS-485 レジスタ書き込み
  - 接続したデバイスのレジスタへの書き込み
- LED 点灯制御
  - アプリケーション LED の点灯 on/off 制御を行います

以下の手順で実行します。

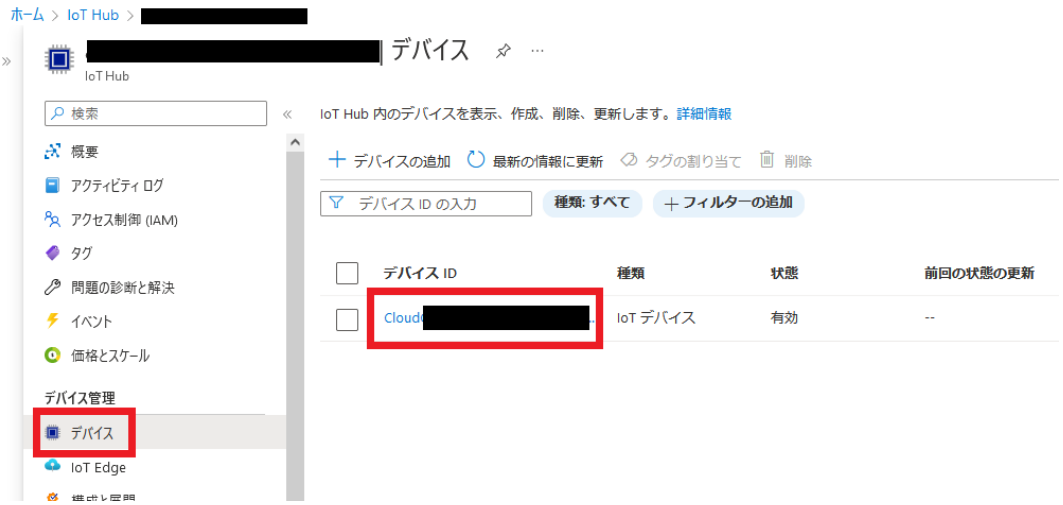
- AWS

  AWS IoT Core の デバイスシャドウを更新し、制御を行います。更新方法については 「6.10.9.1. クラウドからのデータ設定」 と同じです。

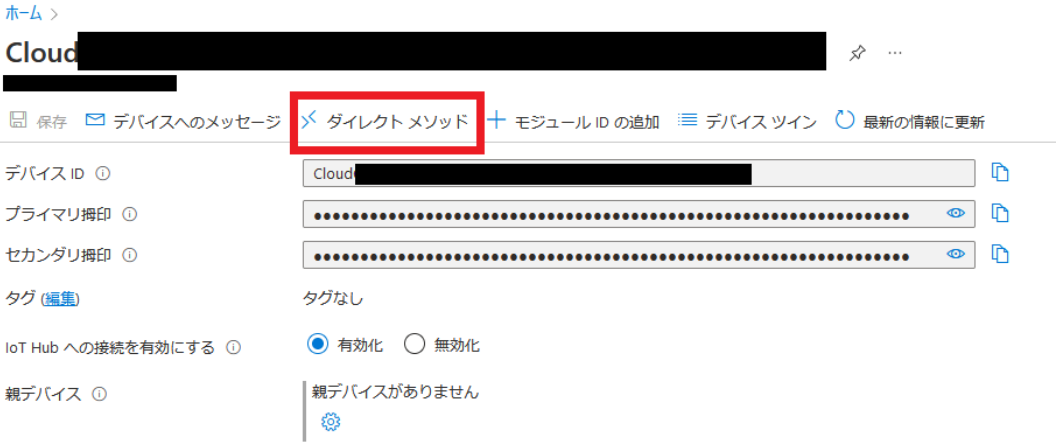
- Azure

  Azure IoT Hub のダイレクトメソッドを更新して設定を行います。

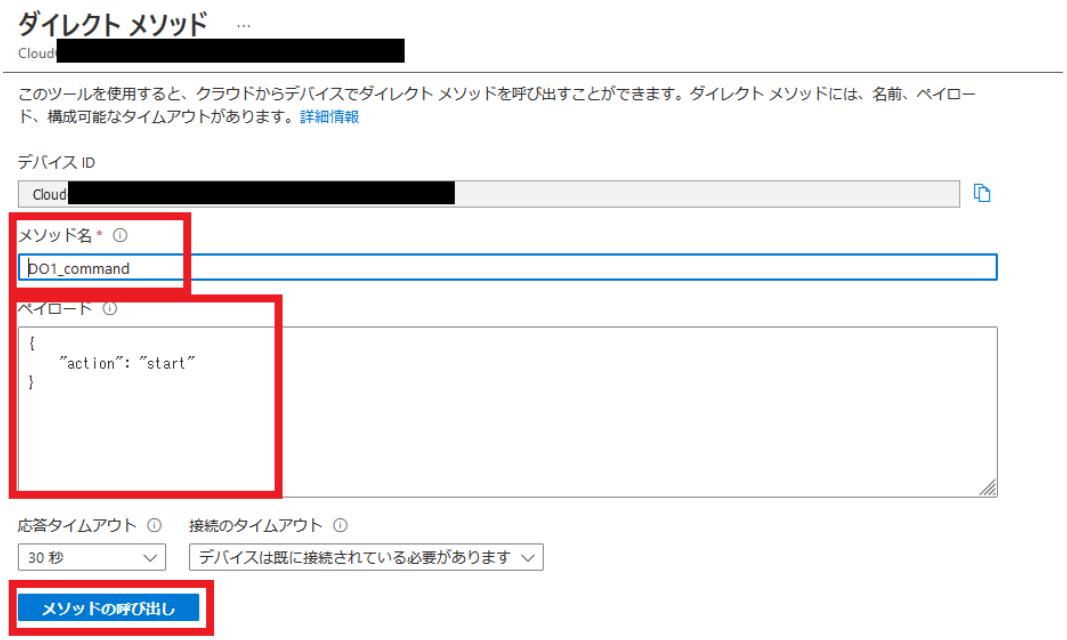
  1. Azure portal から [IoT Hub] を開き、「6.10.5.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 で作成した IoT Hub を選択します。[デバイス] を選択し、一覧の中から該当するデバイス ID を選択します。



2. [ダイレクトメソッド] を選択します。



3. ダイレクトメソッド集画面が表示されるため、メソッド名とペイロードを入力し、「メソッドの呼び出し」をクリックします。



各機能それぞれ、下記の通りのフォーマットとなっています。

・接点出力制御

表 6.23 接点出力制御設定値

項目	概要	設定値	内容
action	制御状態	start	出力開始。ゲートウェイコンテナ起動後に接点出力の設定内容を更新した場合は、一度接点出力制御で出力動作を停止してください。その後、出力開始することで更新内容を適用して動作を開始します。
		stop	接点出力動作を停止します。出力状態については、受信したタイミングの出力状態で固定します。

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "<制御ポート>_command": { ❶
        "action" : <start or stop>
      }
    }
  }
}
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

```
{
  "state": {
    "desired": {
    "DO1_command": {
        "action" : "start"
      }
    }
  }
}
```

図 6.106 接点出力制御シャドウ設定例

・ Azure

メソッド名とペイロードのフォーマットは下記の通りです。

・ メソッド名

```
<制御ポート>_command ❶
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

・ ペイロード

```
{"action" : <start or stop> }
```

各パラメータの設定例は下記の通りです。

・ メソッド名

```
DO1_command
```

・ ペイロード

```
{"action" : "start"}
```

・ RS-485 レジスタ書き込み

表 6.24 RS-485 レジスタ書き込みシャドウ設定値

項目	概要	設定値	内容
method	通信種別	rtu	Modbus-RTU
data_size	データサイズ	7 or 8	
baudrate	ボーレート	300～921600[bps]	通信速度を指定します
parity	パリティビット	none	None
		odd	Odd
		even	Even
stop	ストップビット	1	1
		2	2

項目	概要	設定値	内容
device_id	Modbus スレーブ機器のデバイス ID	1 ～ 247	
func_code	ファンクションコード	5 or 6 or 15 or 16	
register_addr	レジスタアドレス	機器依存	値を書き込むレジスタのアドレスを指定
write_data	レジスタに書き込む値	任意の値	

・ AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
      "RS485_command": {
        "method" : <種別>,
        "baudrate" : <ボーレート>,
        "data_size": <データサイズ>,
        "parity" : <パリティ>,
        "stop" : <ストップビット>,
        "device_id" : <デバイス ID>,
        "func_code" : <ファンクションコード>,
        "register_addr" : <レジスタアドレス>,
        "write_data": <書き込む値>
      }
    }
  }
}
```

```
{
  "state": {
    "desired": {
      "RS485_command": {
        "method" : "rtu",
        "baudrate" : 19200,
        "data_size": 8,
        "parity" : "none",
        "stop" : 1,
        "device_id" : 1,
        "func_code" : 5,
        "register_addr" : 3,
        "write_data": 1
      }
    }
  }
}
```

図 6.107 RS-485 レジスタ書き込みシャドウ設定例

・ Azure

メソッド名とペイロードのフォーマットは下記の通りです。

・ メソッド名

```
RS485_command
```

・ペイロード

```
{
  "method" : <種別>,
  "baudrate" : <ボーレート>,
  "data_size": <データサイズ>,
  "parity" : <パリティ>,
  "stop" : <ストップビット>,
  "device_id" : <デバイス ID>,
  "func_code" : <ファンクションコード>,
  "register_addr" : <レジスタアドレス>,
  "write_data": <書き込む値>
}
```

各パラメータの設定例は下記の通りです。

・メソッド名

```
RS485_command
```

・ペイロード

```
{
  "method" : "rtu",
  "baudrate" : 19200,
  "data_size": 8,
  "parity" : "none",
  "stop" : 1,
  "device_id" : 1,
  "func_code" : 5,
  "register_addr" : 3,
  "write_data": 1
}
```

・LED

表 6.25 LED 点灯制御設定値

項目	概要	設定値	内容
state	点灯状態	on	点灯
		off	消灯

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
```

```
    "led_command": {
        "state" : <on or off>
      }
    }
  }
}
```

```
{
  "state": {
    "desired": {
    "led_command": {
        "state" : "off"
      }
    }
  }
}
```

図 6.108 LED 点灯制御シャドウ設定例

- Azure

  メソッド名とペイロードのフォーマットは下記の通りです。

  - メソッド名

```
led_command
```

  - ペイロード

```
{"state" : <on or off>}
```

  各パラメータの設定例は下記の通りです。

  - メソッド名

```
led_command
```

  - ペイロード

```
{"state" : "off"}
```

6.10.10. コンテナの終了

　podman_start で起動したゲートウェイコンテナを終了させる場合は、以下のコマンドを実行してください。

```
[armadillo ~]# podman stop a6e-gw-container
```

6.10.11. ログ内容確認

「6.10.7. ゲートウェイコンテナの設定ファイル」 でログファイルにログを出力する設定にした場合、インターフェース部とクラウド部にわかれて、それぞれ以下のファイルに出力されます。

- インターフェース部
  - /var/app/volumes/gw_container/device/log/sensing_mgr.log
- クラウド部
  - /var/app/volumes/gw_container/device/log/cloud_agent.log

ログファイルは自動的にローテートされるように設定されています。ローテートされると、各ファイルの末尾に番号が付与されます。なお、ファイル数が 10 を超えた場合は古いファイルから削除されます。

また、ログファイルの内容はテキストデータであり、以下のようなフォーマットになっています。

```
出力日時 ログレベル : メッセージ
```

図 6.109 ログファイルのフォーマット

6.10.12. ゲートウェイコンテナの構成

ゲートウェイコンテナは下記の通り構成されています。コンテナ内外関わらず、誤ってファイルを削除した場合はインストールディスクで初期化を行ってください。

起動スクリプト

コンテナ起動時、下記のスクリプトを実行します。

- /usr/bin/gw-app.sh

ゲートウェイコンテナアプリケーション

ゲートウェイコンテナアプリケーションは下記に配置されています。

- /usr/lib/python3.12/site-packages/atgateway/

ボリュームマウント

以下のパスをコンテナ内でマウントしています。

ホストパス	コンテナパス	概要
/var/app/rollback/volumes/gw_container/cert	/cert/ca	デバイス認証関連ファイル
/var/app/rollback/volumes/gw_container/config	/config	ゲートウェイコンテナコンフィグファイル
/var/app/rollback/volumes/gw_container/src	/root/gw_container	ゲートウェイコンテナ main 関数
/var/app/volumes/gw_container/device/cert	/cert/device	デバイス証明書関連ファイル
/var/app/volumes/gw_container/device/log	/log	ゲートウェイコンテナ ログ

6.11. ゲートウェイコンテナアプリケーションを改造する

「3.16. ゲートウェイコンテナアプリケーションの開発」で説明したとおり、VS Code 上でゲートウェイコンテナアプリケーションの設定を行えますが、メインファイルを変更することで独自のアプリケーションを開始することも可能です。ゲートウェイコンテナアプリケーションの拡張例のファイルは app/example ディレクトリに配置してあります。実行する場合は app/example ディレクトリのファイル一式を app/src ディレクトリにコピーしてください。拡張例のゲートウェイコンテナでは以下の動作を実行します。

- 5 秒毎に Count_value のカウントアップ
- Count_value が 100 に達すると 0 クリア

Count_value がカウントアップしていく様子はログファイルで確認できます。ゲートウェイコンテナのログについての詳細は 「6.10.11. ログ内容確認」 をご参照ください。

```
2023-01-26 11:05:35,115 <INFO> : {'data': {'Count_value': 0, 'timestamp': 1674698730}}
2023-01-26 11:05:45,150 <INFO> : {'data': {'Count_value': 1, 'timestamp': 1674698735}}
2023-01-26 11:05:45,165 <INFO> : {'data': {'Count_value': 2, 'timestamp': 1674698740}}
2023-01-26 11:05:45,175 <INFO> : {'data': {'Count_value': 3, 'timestamp': 1674698745}}
2023-01-26 11:05:55,202 <INFO> : {'data': {'Count_value': 4, 'timestamp': 1674698750}}
2023-01-26 11:05:55,215 <INFO> : {'data': {'Count_value': 5, 'timestamp': 1674698755}}
2023-01-26 11:06:05,242 <INFO> : {'data': {'Count_value': 6, 'timestamp': 1674698760}}
2023-01-26 11:06:05,255 <INFO> : {'data': {'Count_value': 7, 'timestamp': 1674698765}}
2023-01-26 11:06:15,282 <INFO> : {'data': {'Count_value': 8, 'timestamp': 1674698770}}
2023-01-26 11:06:15,295 <INFO> : {'data': {'Count_value': 9, 'timestamp': 1674698775}}
2023-01-26 11:06:25,323 <INFO> : {'data': {'Count_value': 10, 'timestamp': 1674698780}}
2023-01-26 11:06:25,335 <INFO> : {'data': {'Count_value': 11, 'timestamp': 1674698785}}
2023-01-26 11:06:35,362 <INFO> : {'data': {'Count_value': 12, 'timestamp': 1674698790}}
```

図 6.110 ログファイルの Count_value の出力例

6.12. Web UI から Armadillo をセットアップする (ABOS Web)

ABOS Web は、Web ブラウザから Armadillo の動作設定を行う機能で、ABOS (Armadillo Base OS) を搭載する全ての Armadillo に対応しています。

詳細は、「3.9.1. ABOS Web とは」を参照してください。

6.12.1. ABOS Web ではできないこと

ABOS Web は、ABOS の詳細や Linux のコマンドシェルの操作に詳しくない方でも、簡単に Armadillo のセットアップを行なえることを目的にしています。そのための、Armadillo の動作設定を行う機能ですから、動作設定以外のこと、たとえば、Armadillo の動作状態を監視したりすることは、できません。さらに、Armadillo をインターネットから設定操作する、リモート操作もできません。セキュリティの観点から、ABOS Web は、同じ LAN 内からの接続しか受け付けないように実装しています。

ABOS Web でできる Armadillo の設定については、「6.12.2. ABOS Web の設定機能一覧と設定手順」を参照してください。なお、ABOS Web は OSS で提供していますので、現在の ABOS Web に無い設定機能を、ご自分で実装して機能追加することも可能です。

6.12.2. ABOS Web の設定機能一覧と設定手順

現在、ネットワークに関して ABOS Web で設定できるのは以下のものです。

- WWAN 設定
- WLAN 設定
- 各接続設定（各ネットワークインターフェースの設定）
- DHCP サーバー設定
- NAT 設定
- VPN 設定

これらについては、「3.9. ネットワーク設定」で紹介していますので、そちらを参照してください。

ネットワーク以外にも ABOS Web は以下の機能を持っています。

- コンテナ管理
- SWU インストール
- 時刻設定
- アプリケーション向けのインターフェース (Rest API)
- カスタマイズ

本章では、これらのネットワーク以外の設定項目について紹介します。

6.12.3. コンテナ管理

ABOS Web から Armadillo 上のコンテナを一覧表示して、コンテナごとに起動・停止を行うことができます。

ABOS Web のトップページから、"コンテナ管理"をクリックすると、「図 6.111. コンテナ管理」の画面に遷移します。



図 6.111 コンテナ管理

この画面では、ABOS 上にあるコンテナ全てについて、イメージ名やコンテナ名、現在状態を一覧表示します。コンテナの一覧表示欄で選択したコンテナに対し、起動と停止、および、コンテナから出力されたログの表示を行うことができます。



「3.9.12. VPN 設定」に記載のとおり、VPN 接続を設定すると、abos_web_openvpn のコンテナが作成されます。VPN 接続中は、このコンテナが動作状態になっており、このコンテナをコンテナ管理画面で停止すると、VPN 接続が切断されます。

6.12.4. SWU インストール

ABOS Web から PC 上の SWU イメージや HTTP サーバー上の SWU イメージを Armadillo にインストールすることができます。

SWU イメージについては、「3.3.4.1. SWU イメージとは」を参照してください。

ABOS Web のトップページから、"SWU インストール"をクリックすると、「図 6.112. SWU インストール」の画面に遷移します。



図 6.112 SWU インストール

この画面では、PC 上の SWU イメージファイルまたは、HTTP サーバー上の SWU イメージファイルの URL を指定して、Armadillo にインストールすることができます。Armadillo のソフトウェアのアップデート用に最初に行う設定で作成する initial_setup.swu が、まだ Armadillo にインストールされていなければ、"mkswu --init で作成した initial_setup.swu をインストールしてください。" というメッセージを画面上部に表示します。



図 6.113 最新バージョンに更新

この画面の "更新" を押すと、プルダウンメニューで指定されている SWU イメージをダウンロードして、 Armadillo にインストールします。プルダウンメニューに表示される SWU イメージは、 /etc/swupdate.watch に記述されている URL から取得しています。

青色に表示されている「こちら」を押すと、Armadillo Base OS の SWU イメージのダウンロードページにアクセスできます。

SWU イメージのインストール動作を実行する時には、進行状況を示すログを表示します。"現在の SWU で管理されているバージョン" 欄には、ABOS の各ソフトウェアコンポーネントの名前とバージョン情報を一覧表示します。

6.12.5. 時刻設定

SWU 管理対象ソフトウェアコンポーネントの一覧表示. ABOS Web から時刻に関する設定を行うことができます。

ABOS Web のトップページから "時刻設定" をクリックすると、以下の内容が表示されます。

「図 6.114. ネットワークタイムサーバーと同期されている場合の状況確認画面」 では Armadillo の現在時刻と、同期中のサーバーとの時間差を確認することができます。



図 6.114 ネットワークタイムサーバーと同期されている場合の状況確認画面

時刻が同期されてない状態では 「図 6.115. ネットワークタイムサーバーと同期されていない場合の状況確認画面」 の様に 「PC と同期する」ボタンを押すことで、 Armadillo の時刻を PC と同期することができます。



図 6.115 ネットワークタイムサーバーと同期されていない場合の状況確認画面

「図 6.116. ネットワークタイムサーバーの設定項目」 では NTP (ネットワークからの時刻同期）サーバーと Armadillo 起動時に同期するサーバーを設定することができます。



図 6.116 ネットワークタイムサーバーの設定項目

最後に、「図 6.117. タイムゾーンの設定項目」 では Armadillo Base OS で使用するタイムゾーンの変更ができます。コンテナには影響ありませんのでご注意ください。



図 6.117 タイムゾーンの設定項目

6.12.6. アプリケーション向けのインターフェース (Rest API)

コンテナやスクリプトから ABOS Web の一部の機能を使用できます。

6.12.6.1. Rest API へのアクセス権の管理

Rest API は ABOS Web のパスワードと Rest API 用のトークンで認証されます。

また、接続可能なネットワークにも制限をかけております。初期状態では、同一サブネットからのアクセスのみ許容しています。同一サブネット外の IP アドレスからアクセスしたい場合は設定が必要です。設定方法は 「3.9.2. ABOS Web へのアクセス」 を参照してください。

各リクエストは以下のどちらかの Authorization ヘッダーで認証されます：

- Basic (パスワード認証) : curl の -u :<password> 等で認証可能です。<password> の文字列は ABOS Web で設定したパスワードです。
- Bearer (トークン認証) : curl の -H "Authorization: Bearer <token> 等で認証可能です。<token> は /api/tokens であらかじめ生成した文字列です。

また、トークンには権限も設定できます。Admin で生成されたトークンはすべてのインターフェースにアクセスできますが、一部のインターフェースしか使用しない場合はそのインターフェースに必要な権限だけを持つトークンを生成してください。

トークンの管理は ABOS Web の「設定管理」ページで行えます:

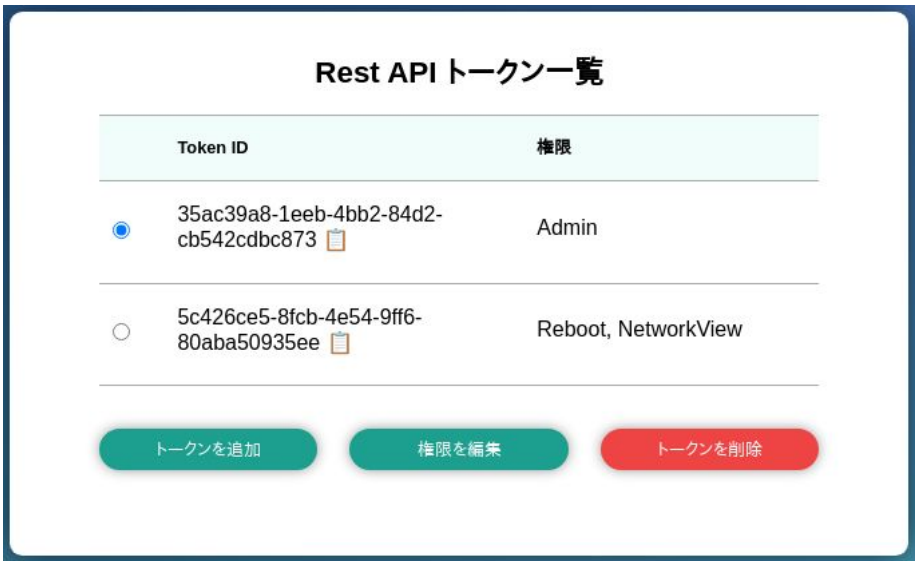


図 6.118 設定管理の Rest API トークン一覧表示



ABOS Web の バージョン 1.2.3 以降では、Token ID の横にあるクリップボードアイコンをクリックするとクリップボードにコピーすることができます。

6.12.6.2. Rest API 使用例の前提条件

各 Rest API の使用例を説明します。使用例では以下を前提としています。:

- ABOS Web に `https://armadillo.local:58080` でアクセスします。
- 「 `AUTH` 」環境変数に ABOS Web で生成したトークンを設定します。例： `AUTH="Authorization: Bearer 35ac39a8-1eeb-4bb2-84d2-cb542cdbc873"`
- `curl` コマンドを省略するため、以下のように alias を使用します：

```
[ATDE ~]$ alias curl_rest='curl -k -H "$AUTH" -w "¥nhttp code: %{http_code}¥n" '
```



コンテナから ABOS Web には「https://host.containers.internal:58080」でアクセスできます。



この章で説明する例では、`curl` のオプションに `-k` を指定して証明書を無視するようにしています。もし、証明書を使用したい場合は以下のように設定してください。

```
[ATDE ~]$ openssl s_client -showcerts -connect armadillo.local:58080 </
dev/null 2>/dev/null | openssl x509 -outform PEM > abosweb.pem
[ATDE ~]$ CERT="$PWD/abosweb.pem"
[ATDE ~]$ alias curl_rest='curl -H "$AUTH" --cacert "$CERT" -w "¥nhttp
code: %{http_code}¥n" '
```

6.12.6.3. Rest API の入力と出力

インターフェースの一部にはパラメータを取るものがあります。パラメータがある場合は json (Content-Type を application/json に設定する）と form（デフォルトの application/x-www-form-urlencoded でのパラメータ）のどちらでも使用可能です。

インターフェースの出力がある場合は json object で出力されます。今後のバージョンアップで json object のキーが増える可能性があるため、出力された値を処理する場合はその点に留意してください。

エラーの場合は json object の「error」キーに文字列のエラーが記載されています。http のステータスコードも 50x になります。

エラーの例：

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/tokens/
3b2d830d-2f64-4e76-9e59-316da82eefc4
{"error":"No such token"}
http code: 500
```

6.12.6.4. Rest API : トークン管理

トークン管理のためのインターフェースは以下のとおりです：

- **トークン一覧**
  GET "/api/tokens"
  必要権限: Admin
  パラメータ: 無し
  出力: トークンリスト

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/tokens
{"tokens":[{"token":"35ac39a8-1eeb-4bb2-84d2-cb542cdbc873","permissions":["Admin"]},
{"token":"5c426ce5-8fcb-4e54-9ff6-80aba50935ee","permissions":["Reboot","NetworkView"]}]}
http code: 200
```

- **トークン取得**
  GET "/api/tokens/<token>"
  必要権限: Admin
  パラメータ: 無し
  出力: トークン情報

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/tokens/35ac39a8-1eeb-4bb2-84d2-
cb542cdbc873
{"token":"35ac39a8-1eeb-4bb2-84d2-cb542cdbc873","permissions":["Admin"]}
http code: 200
```

- **トークン生成**
  POST "/api/tokens"
  必要権限: Admin
  パラメータ: 付与したい permissions 権限リスト(ない場合は「Admin」で生成されます)
  出力: 生成されたトークン情報

```
[ATDE ~]$ curl_rest -H "Content-type: application/json" -d '{"permissions": ["SwuInstall",
"ContainerView"]}' https://armadillo.local:58080/api/tokens
{"token":"3b2d830d-2f64-4e76-9e59-316da82eefc4","permissions":
["SwuInstall","ContainerView"]}
http code: 200
```

- **トークン編集 (存在しない場合は指定のトークンで生成されます)**
  POST "/api/tokens/{token_id}"
  必要権限: Admin
  パラメータ: 付与したい permissions 権限リスト(ない場合は編集しません)
  出力: 編集か生成されたトークン情報

```
[ATDE ~]$ curl_rest -X POST -d permissions=Poweroff -d permissions=ContainerAdmin https://
armadillo.local:58080/api/tokens/3b2d830d-2f64-4e76-9e59-316da82eefc4
{"token":"3b2d830d-2f64-4e76-9e59-316da82eefc4","permissions":["Poweroff","ContainerAdmin"]}
```

- **トークン削除**
  DELETE "/api/tokens/{token_id}"
  必要権限: Admin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/tokens/
3b2d830d-2f64-4e76-9e59-316da82eefc4
http code: 200
```

- **abos-web パスワード変更**
  POST "/api/password"
  必要権限: Admin
  パラメータ: password でハッシュ済みのパスワード文字列か hashed=false が設定されている場合は平文の文字列
  出力: 無し

```
[ATDE ~]$ PWD_HASH=$(openssl passwd -6)
Password:
Verifying - Password:
[ATDE ~]$ echo $PWD_HASH
$6$LuXQduN7L3PwbMaZ$txrw8vLJqEVUreQnZhM0CYMQ5U5B9b58L0mpVRULDiVCh2O46GKscq/
xsDPskjxg.x8ym0ri1/8NqFBu..IZE0
[ATDE ~]$ curl_rest --data-urlencode "password=$PWD_HASH" -X POST https://armadillo.local:
58080/api/password
http code: 200
```

6.12.6.5. Rest API : SWU

- **インストール済み SWU のバージョン情報取得**
  GET "/api/swu/versions"
  必要権限: SwuView
  パラメータ: 無し
  出力: Swupdate の各バージョン情報

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/swu/versions
{"extra_os.custom":"54","extra_os.container":"1","custom":"54","extra_os.initial_setup":"4",
"boot":"2020.4-at19","base_os":"3.18.4-at.6","extra_os.sshd":"1"}
http code: 200
```

- **アップデートステータス取得**
  GET "/api/swu/status"
  必要権限: SwuView
  パラメータ: 無し
  出力: rollback_ok: ロールバック状態 (false の場合は rollback されています)、last_update_timestamp: UTC の unix epoch (数字での日付)、 last_update_versions: 最新のアップデートで更新されたバージョン情報 (コンポーネント → [更新前のバージョン, 更新後のバージョン]。 更新前に存在しなかったコンポーネントの場合は null で記載されています)

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/swu/status
{"rollback_ok":true,"last_update_timestamp":1703208559,"last_update_versions":{"custom":
[null,"54"],"extra_os.custom":["53","54"]}}
http code: 200
```

- **SWU をファイルアップロードでインストール**
  POST "/api/swu/install/upload"
  必要権限: SwuInstall
  パラメータ: multipart/form-data で **swu** の転送
  出力: swupdate プロセスの出力 (stdout または stderr)、またはアップデートプロセスの出力ステータス (exit_code または exit_signal)

```
[ATDE ~]$ curl_rest -F swu=@"$HOME/mkswu/file.swu" https://armadillo.local:58080/api/swu/
install/upload
{"stdout":"SWUpdate v2023.05_git20231025-r0¥n"}
{"stdout":"¥n"}
{"stdout":"Licensed under GPLv2. See source distribution for detailed copyright notices.¥n"}
{"stdout":"¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [main] : Running on AGX4500 Revision at1¥n"}
{"stdout":"[INFO ] : SWUPDATE started :  Software Update started !¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [install_single_image] : Installing pre_script¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying
current os over¥n"}
: (省略)
{"stdout":"[INFO ] : SWUPDATE running :  [install_single_image] : Installing post_script¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [read_lines_notify] : swupdate triggering reboot!¥n"}
{"stderr":"Killed¥n"}
{"exit_code":0}
```

```
http code: 200
```

- **SWU を URL でインストール**
  POST "/api/swu/install/url"
  必要権限: SwuInstall
  パラメータ: url=<SWU をダウンロードできる URL>
  出力: swupdate プロセスの出力 (stdout または stderr)、またはアップデートプロセスの出力ステータス (exit_code または exit_signal)

```
[ATDE ~]$ curl_rest -d url=https://url/to/file.swu https://armadillo.local:58080/api/swu/
install/url
{"stdout":"Downloading https://url/to/file.swu...¥n"}
{"stdout":"SWUpdate v2023.05_git20231025-r0¥n"}
{"stdout":"¥n"}
{"stdout":"Licensed under GPLv2. See source distribution for detailed copyright notices.¥n"}
{"stdout":"¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [main] : Running on AGX4500 Revision at1¥n"}
{"stdout":"[INFO ] : SWUPDATE started :  Software Update started !¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [install_single_image] : Installing pre_script¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying
current os over¥n"}
: (省略)
{"stdout":"[INFO ] : SWUPDATE running :  [install_single_image] : Installing post_script¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers¥n"}
{"stdout":"[INFO ] : SWUPDATE running :  [read_lines_notify] : swupdate triggering reboot!¥n"}
{"stderr":"Killed¥n"}
{"exit_code":0}

http code: 200
```

6.12.6.6. Rest API : コンテナ操作

- **コンテナー覧**
  GET "/api/containers"
  必要権限: ContainerView
  パラメータ: 無し
  出力: 各コンテナの id, name, state, command, image 情報

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/containers
{"containers":
[{"id":"02616122dcea5bd75c551b29b2ef54f54e09f59c50ce3282684773bc6bfb86a8","name":"python_app
","state":"running","command":["python3","/vol_app/src/main.py"],"image":"localhost/
python_arm64_app_image:latest"}]}
http code: 200
```

- **コンテナログ取得**
  GET "/api/containers/{container}/logs"
  必要権限: ContainerView
  パラメータ: follow=true (podman logs -f と同様の効果)
  出力: podman logs プロセスの出力 (stdout または stderr)、またはアップデートプロセスの出力ステータス (exit_code または exit_signal)

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/containers/python_app/logs
{"stdout":"Some message¥n"}
{"exit_code":0}

http code: 200
```

follow=true を付与する例

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/containers/python_app/logs?follow=true
{"stdout":"Some message¥n"}
Ctrl-C で終了
```

- **コンテナ起動** (すでに起動されている場合は再起動されます)
  POST "/api/containers/{container}/start"
  必要権限: ContainerAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/containers/python_app/start

http code: 200
```

- **コンテナ停止**
  POST "/api/containers/{container}/stop"
  必要権限: ContainerAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/containers/python_app/stop

http code: 200
```

6.12.6.7. Rest API : ネットワーク設定

- **ネットワーク設定一覧**
  GET "/api/connections"
  必要権限: NetworkView
  パラメータ: 無し
  出力: ネットワーク設定一覧と各接続の uuid, name, state, ctype, 存在すれば device 情報

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/connections
{"connections":[{"name":"Wired connection
1","state":"activated","uuid":"18d241f1-946c-3325-974f-65cda3e6eea5","ctype":"802-3-
ethernet","device":"eth0"},{"name":"lo","state":"activated","uuid":"529ec241-f122-4cb2-843f-
ec9787b2aee7","ctype":"loopback","device":"lo"},
{"name":"podman0","state":"activated","uuid":"be4583bc-3498-4df2-
a31c-773d781433aa","ctype":"bridge","device":"podman0"},
{"name":"veth0","state":"activated","uuid":"03446b77-b1ab-47d0-98fc-
f167c3f3778a","ctype":"802-3-ethernet","device":"veth0"},{"name":"Wired connection
```

```
2","state":"","uuid":"181f44df-850e-36c1-a5a4-6e461c768acb","ctype":"802-3-ethernet"},
{"name":"Wired connection 3","state":"","uuid":"e4381368-6351-3985-
ba6e-2625c62b8d39","ctype":"802-3-ethernet"}]}

http code: 200
```

- **ネットワーク設定詳細取得**
  GET "/api/connections/{connection}"
  必要権限: NetworkView
  パラメータ: 無し（URL の connection は UUID または接続名で使用可能）
  出力: 接続の詳細情報（Network Manager のプロパティ）

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/connections/Wired%20connection%201
{"name":"Wired connection
1","state":"activated","uuid":"18d241f1-946c-3325-974f-65cda3e6eea5","ctype":"802-3-
ethernet","device":"eth0","props":{"802-3-ethernet.accept-all-mac-addresses":"-1","802-3-
ethernet.auto-negotiate":"no","802-3-ethernet.cloned-mac-address":"","802-3-
ethernet.duplex":"","802-3-ethernet.generate-mac-address-mask":"","802-3-ethernet.mac-
address":"","802-3-ethernet.mac-address-blacklist":"","802-3-ethernet.mtu":"auto","802-3-
ethernet.port":"","802-3-ethernet.s390-nettype":"","802-3-ethernet.s390-options":"","802-3-
ethernet.s390-subchannels":"","802-3-ethernet.speed":"0","802-3-ethernet.wake-on-
lan":"default","802-3-ethernet.wake-on-lan-password":"","GENERAL.CON-PATH":"/org/
freedesktop/NetworkManager/Settings/1","GENERAL.DBUS-PATH":"/org/freedesktop/NetworkManager/
ActiveConnection/
6","GENERAL.DEFAULT":"yes","GENERAL.DEFAULT6":"no","GENERAL.DEVICES":"eth0","GENERAL.IP-
IFACE":"eth0","GENERAL.MASTER-PATH":"","GENERAL.NAME":"Wired connection 1","GENERAL.SPEC-
OBJECT":"","GENERAL.STATE":"activated","GENERAL.UUID":"18d241f1-946c-3325-974f-65cda3e6eea5"
,"GENERAL.VPN":"no","GENERAL.ZONE":"","IP4.ADDRESS[1]":"198.51.100.123/16","IP4.DNS[1]":"192
.0.2.1","IP4.DNS[2]":"192.0.2.2","IP4.GATEWAY":"198.51.100.1","IP4.ROUTE[1]":"dst =
198.51.100.0/16, nh = 0.0.0.0, mt = 100","IP4.ROUTE[2]":"dst = 0.0.0.0/0, nh = 198.51.100.1,
mt = 100","IP6.ADDRESS[1]":"fe80::211:cff:fe00:b13/64","IP6.GATEWAY":"","IP6.ROUTE[1]":"dst
= fe80::/64, nh = ::, mt = 1024","connection.auth-
retries":"-1","connection.autoconnect":"yes","connection.autoconnect-
priority":"-999","connection.autoconnect-retries":"-1","connection.autoconnect-
slaves":"-1","connection.dns-over-tls":"-1","connection.gateway-ping-
timeout":"0","connection.id":"Wired connection 1","connection.interface-
name":"eth0","connection.lldp":"default","connection.llmnr":"-1","connection.master":"","con
nection.mdns":"-1","connection.metered":"unknown","connection.mptcp-
flags":"0x0","connection.multi-connect":"0","connection.permissions":"","connection.read-
only":"no","connection.secondaries":"","connection.slave-type":"","connection.stable-
id":"","connection.timestamp":"1703208824","connection.type":"802-3-
ethernet","connection.uuid":"18d241f1-946c-3325-974f-65cda3e6eea5","connection.wait-
activation-delay":"-1","connection.wait-device-
timeout":"-1","connection.zone":"","ipv4.addresses":"198.51.100.123/16","ipv4.auto-route-
ext-gw":"-1","ipv4.dad-timeout":"-1","ipv4.dhcp-client-id":"","ipv4.dhcp-
fqdn":"","ipv4.dhcp-hostname":"","ipv4.dhcp-hostname-flags":"0x0","ipv4.dhcp-
iaid":"","ipv4.dhcp-reject-servers":"","ipv4.dhcp-send-hostname":"yes","ipv4.dhcp-
timeout":"0","ipv4.dhcp-vendor-class-
identifier":"","ipv4.dns":"192.0.2.1,192.0.2.2","ipv4.dns-options":"","ipv4.dns-
priority":"0","ipv4.dns-search":"","ipv4.gateway":"198.51.100.1","ipv4.ignore-auto-
dns":"no","ipv4.ignore-auto-routes":"no","ipv4.link-local":"0","ipv4.may-
fail":"yes","ipv4.method":"manual","ipv4.never-default":"no","ipv4.replace-local-
rule":"-1","ipv4.required-timeout":"-1","ipv4.route-metric":"-1","ipv4.route-
table":"0","ipv4.routes":"","ipv4.routing-rules":"","ipv6.addr-gen-
mode":"eui64","ipv6.addresses":"","ipv6.auto-route-ext-gw":"-1","ipv6.dhcp-
duid":"","ipv6.dhcp-hostname":"","ipv6.dhcp-hostname-flags":"0x0","ipv6.dhcp-
```

```
iaid":"","ipv6.dhcp-send-hostname":"yes","ipv6.dhcp-timeout":"0","ipv6.dns":"","ipv6.dns-
options":"","ipv6.dns-priority":"0","ipv6.dns-search":"","ipv6.gateway":"","ipv6.ignore-
auto-dns":"no","ipv6.ignore-auto-routes":"no","ipv6.ip6-privacy":"-1","ipv6.may-
fail":"yes","ipv6.method":"auto","ipv6.mtu":"auto","ipv6.never-default":"no","ipv6.ra-
timeout":"0","ipv6.replace-local-rule":"-1","ipv6.required-timeout":"-1","ipv6.route-
metric":"-1","ipv6.route-table":"0","ipv6.routes":"","ipv6.routing-
rules":"","ipv6.token":"","proxy.browser-only":"no","proxy.method":"none","proxy.pac-
script":"","proxy.pac-url":""}}
http code: 200
```

- **ネットワーク設定の変更**
  PATCH "/api/connections/{connection}"
  必要権限: NetworkAdmin
  パラメータ: Network Manager で編集可能な値
  出力: 無し

```
[ATDE ~]$ curl_rest -X PATCH -d ipv4.method=manual -d ipv4.addresses=198.51.100.123/16
https://armadillo.local:58080/api/connections/Wired%20connection%201

http code: 200
```

- **ネットワークの接続**
  POST "/api/connections/{connection}/up"
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/connections/Wired%20connection
%201/up

http code: 200
```

- **ネットワークの切断**
  POST "/api/connections/{connection}/down"
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/connections/Wired%20connection
%201/down

http code: 200
```

- **ネットワーク設定の削除**
  DELETE "/api/connections/{connection}"
  必要権限: NetworkAdmin
  パラメータ: 無し出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/connections/178b8c95-
fcad-4bb1-8040-5a02b9ad046f
```

```
http code: 200
```



通信に使用しているネットワークの設定を削除した場合は Armadillo へアクセスできなくなりますので、ご注意ください。

6.12.6.8. Rest API : WLAN

- **無線ネットワークのリスト取得**
  GET "/api/wlan/scan"
  必要権限: NetworkView
  パラメータ: (任意)rescan=true/false, false を指定するとキャッシュされているスキャン結果を出力します。
  出力: リスト

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wlan/scan
[{"id":"my_ap","signal":74,"bssid":"04:42:1A:E4:78:0C","chan":44,"rate":"540 Mbit/
s","security":"WPA2 WPA3"},{"id":"other_ap","signal":65,"bssid":"AC:44:F2:56:22:38","chan":
1,"rate":"130 Mbit/s","security":"WPA2"}]
http code: 200
```

- *無線ネットワークの接続
  POST "/api/wlan/connect"
  必要権限: NetworkAdmin
  パラメータ： ssid, passphrase, ifname, bssid, hidden. ssid 以外は任意です。
  出力: 生成した接続の uuid

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wlan/connect -d ssid=my_ap -d
passphrase=my_passphrase
{"uuid":"178b8c95-fcad-4bb1-8040-5a02b9ad046f"}
http code: 200
```

- **無線ネットワーク アクセスポイントの設定**
  POST "/api/wlan/ap"
  必要権限: NetworkAdmin
  パラメータ: ssid, passphrase, bridge_addr, hw_mode/channel, interface.
  interface は任意です。hw_mode:2.4GHz を使用する場合は "g"、5GHz を使用する場合は "a" を設定します。
  channel: 2.4GHz の場合は 1 ～ 13、5GHz の場合は 36、40、44、48 を設定します。
  hw_mode/channel を設定しない場合は自動的に選択されますが、両方を未設定にすることはできません。
  bridge_addr は既に br_ap に eth0 等がブリッジされ、IP アドレスを持っている場合、設定しないとその IP アドレスがそのまま使用されます。
  出力: 無し

```
[ATDE ~]$ curl_rest -d ssid=my_ap -d passphrase=my_passphrase -d bridge_addr=198.51.100.1/24
-d channel=3 https://armadillo.local:58080/api/wlan/ap
```

```
http code: 200
```



アクセスポイントを設定するとクライアントの接続が無効になります。



クライアントの接続の削除は DELETE ”/api/connections/{connection}” で行えます。

- **無線ネットワーク アクセスポイントの削除**
  DELETE ”/api/wlan/ap”
  必要権限: NetworkAdmin
  パラメータ: interface (任意)
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/wlan/ap

http code: 200
```

6.12.6.9. Rest API : WWAN の設定

- **WWAN の設定追加**
  POST ”/api/wwan”
  必要権限: NetworkAdmin
  パラメータ: apn, user, password, auth_type (CHAP/PAP, デフォルト CHAP), mccmnc, ipv6 (bool、デフォルト true), no_connect (デフォルト false)
  apn 以外は任意です。
  no_connect を true にすると、設定のみを行い接続は行いません。
  出力: 追加された接続の uuid

```
[ATDE ~]$ curl_rest -d apn=provider.tld -d user=provider -d password=provider https://
armadillo.local:58080/api/wwan
{”uuid”:”ce603d3e-838b-4ac8-b7fd-6a3f1abe4003”}
http code: 200
```

- **WWAN の設定削除**
  DELETE ”/api/wwan”
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/wwan

http code: 200
```



WWAN の設定確認または一時的な切断は connection の API で行ってください。

- **IMEI の取得**
  GET "/api/wwan/imei"
  必要権限: NetworkView
  パラメータ: 無し
  出力: LTE モジュールの IMEI

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wwan/imei
{"imei":"XXXXXXXXXXXXXXX"}
http code: 200
```

- **電話番号の取得**
  GET "/api/wwan/phone_numbers"
  必要権限: NetworkView
  パラメータ: 無し
  出力: SIM カードに設定されている電話番号

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wwan/phone_numbers
{"phone_numbers":["XXXXXXXXXXX"]}
http code: 200
```

- **電波品質の取得** (ABOS 3.20.3-at.5 以降)
  GET "/api/wwan/signal_quality"
  必要権限: NetworkView
  パラメータ: 無し
  出力: 電波品質(mmcli コマンドで出力される signal quality 相当の値)

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wwan/signal_quality
{"signal_quality":"54"}
http code: 200
```

- **SMS 一覧の取得** (ABOS 3.20.3-at.5 以降)
  GET "/api/wwan/sms"
  必要権限: SmsView
  パラメータ: 無し
  出力: SMS メッセージ ID 一覧

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wwan/sms
{"message_ids":["0", "1"]}
http code: 200
```

- **SMS の取得** (ABOS 3.20.3-at.5 以降)
  GET "/api/wwan/sms/{message_id}"
  必要権限: SmsView
  パラメータ: 無し
  出力: SMS の内容

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/wwan/sms/0
{"sms":{"content":{"data":"--","number":"XXXXXXXXXXX","text":"message"},"dbus_path":"/org/
freedesktop/ModemManager1/SMS/0","properties":
{"pdu_type":"deliver","state":"received","storage":"me","timestamp":"20YY-MM-DDThh:mm:ss
+ZZ"}}}

http code: 200
```



message_id は SMS 一覧の取得で表示された値を使用します。

- **SMS の作成・送信** (ABOS 3.20.3-at.5 以降)
  POST "/api/wwan/sms"
  必要権限: SmsSend
  パラメータ: phone_number: 送信先電話番号, message: 送信するメッセージ
  出力: 無し

```
[ATDE ~]$ curl_rest -d phone_number=XXXXXXXXXXX -d message="message" https://armadillo.local:
58080/api/wwan/sms

http code: 200
```

- **SMS の削除** (ABOS 3.20.3-at.5 以降)
  DELETE "/api/wwan/sms/{message_id}"
  必要権限: SmsView
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/wwan/sms/0

http code: 200
```



message_id は SMS 一覧の取得で表示された値を使用します。

- **SMS の全削除** (ABOS 3.20.3-at.5 以降)
  DELETE "/api/wwan/sms"
  必要権限: SmsView
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/wwan/sms

http code: 200
```

- **LTE モジュールの再起動** (ABOS 3.22.1-at.5 以降)
  POST "/api/wwan/force_restart"
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/wwan/force_restart

http code: 200
```

6.12.6.10. Rest API : DHCP の設定

- **DHCP の設定確認**
  GET "/api/dhcp"
  必要権限: NetworkView
  パラメータ: 無し
  出力: interface, ip_addr, start_addr, end_addr, lease_time のリスト

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/dhcp
[{"interface":"br_ap","ip_addr":"198.51.100.1/24","start_addr":"198.51.100.10","end_addr":"1 ⏎
98.51.100.20","lease_time":"3600"}]
http code: 200
```

- **DHCP の設定**
  POST "/api/dhcp/{interface}"
  必要権限: NetworkAdmin
  パラメータ: start_addr, end_addr, lease_time
  lease_time を設定しなかった場合は 3600 (秒) とする
  出力: 無し

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/dhcp/br_ap -d start_addr=198.51.100.10 ⏎
-d end_addr=198.51.100.20
```

```
http code: 200
```

- DHCP の設定削除
  DELETE "/api/dhcp/{interface}"
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/dhcp/br_ap

http code: 200
```

6.12.6.11. Rest API : NAT の設定

- NAT (masquerade) の設定確認
  GET "/api/nat"
  必要権限: NetworkView
  パラメータ: 無し
  出力: NAT されている interface のリスト

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/nat
[{"interface":"eth0"}]
http code: 200
```

- NAT の設定
  POST "/api/nat/{interface}"
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/nat/eth0

http code: 200
```

- NAT の削除
  DELETE "/api/nat/{interface}"
  必要権限: NetworkAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/nat/eth0

http code: 200
```

- ポートフォワードの設定確認
  GET "/api/port_forwarding"
  必要権限: NetworkView
  パラメータ: 無し

出力: フォワードされてるポート

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/port_forwarding
[{"interface":"eth0","protocol":"tcp","dport":"22","destination":"127.0.0.1","destination_po
rt":"2222"}]
http code: 200
```

- **ポートフォワードの設定**
  POST "/api/port_forwarding"
  必要権限: NetworkAdmin
  パラメータ: interface, protocol (デフォルト tcp), dport, destination, destination_port
  出力: 無し

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/port_forwarding -d interface=eth0  -d
dport=22 -d de
stination=127.0.0.1 -d destination_port=2222

http code: 200
```

- **ポートフォワードの削除**
  DELETE "/api/port_forwarding"
  必要権限: NetworkAdmin
  パラメータ: interface, protocol (デフォルト tcp), dport, destination, destination_port
  出力: 無し

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/port_forwarding -X DELETE -H "Content-
Type: application/json" -d
'{"interface":"eth0","protocol":"tcp","dport":"22","destination":"127.0.0.1","destination_po
rt":"2222"}'

http code: 200
```

6.12.6.12. Rest API : VPN の設定

VPN クライアントは、現在 OpenVPN [https://openvpn.net/community/] をサポートしています。

- **VPN の設定** (ABOS 3.20.3-at.4 以降)
  POST "/api/vpn/openvpn"
  必要権限: VpnAdmin
  パラメータ: setting_name, conf, auth_type, username, password, cert, key, key_pass
  setting_name: 設定名です。任意の文字列を設定してください。
  conf: OpenVPN の設定ファイルです。
  auth_type: 認証方式です。userpass(ユーザ名とパスワード) または cert(証明書) を設定してください。
  username: auth_type が userpass の場合、ユーザ名を設定します。
  password: auth_type が userpass の場合、パスワードを設定します。
  cert: auth_type が cert の場合、証明書ファイルを設定します。OpenVPN の設定ファイルに含まれている場合は不要です。
  key: auth_type が cert の場合、鍵ファイルを設定します。OpenVPN の設定ファイルに含まれている場合は不要です。
  key_pass: 鍵がパスワードで保護されている場合、そのパスワードを設定します。

出力: 無し

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/vpn/openvpn -F setting_name=myvpn -F
conf=@"$HOME/conf.ovpn" -F auth_type=userpass -F username=myname -F password=mypass
http code: 200
```

図 6.119 ユーザ名とパスワード認証の例

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/vpn/openvpn  -F setting_name=myvpn -F
conf=@"$HOME/conf.ovpn" -F auth_type=cert -F cert=@"$HOME/cert.crt" -F key=@"$HOME/key.key" -F
key_pass=mykeypass
http code: 200
```

図 6.120 証明書認証の例



コンテナ内から VPN 設定の Rest API を使う場合、 Armadillo 上に alpine_openvpn コンテナイメージが存在していないと、正しく設定されません。このコンテナイメージが存在していない場合、先に ABOS Web の Web UI やコンテナ外からの Rest API で設定を行ってください。一度 alpine_openvpn コンテナイメージがインストールされれば、コンテナ内からも Rest API で設定できます。alpine_openvpn コンテナイメージは VPN 設定を削除しても残り続けますが、VPN 設定を削除した後に swupdate でアップデートを行うと削除されますので、その場合は再度 alpine_openvpn コンテナイメージのインストールを行う必要があります。

- **VPN の接続** (ABOS 3.20.3-at.4 以降)
  POST "/api/vpn/up"
  必要権限: VpnAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/vpn/up
http code: 200
```

- **VPN の切断** (ABOS 3.20.3-at.4 以降)
  POST "/api/vpn/down"
  必要権限: VpnAdmin
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/vpn/down
http code: 200
```

- **VPN 設定の削除** (ABOS 3.20.3-at.4 以降)
  DELETE "/api/vpn/openvpn"
  必要権限: VpnAdmin

パラメータ: 無し
出力: 無し

```
[ATDE ~]$ curl_rest -X DELETE https://armadillo.local:58080/api/vpn/openvpn
http code: 200
```

6.12.6.13. Rest API : 時刻の設定

- **時刻の状況確認**
  GET "/api/time/ntp_info"
  必要権限: TimeView
  パラメータ: 無し
  出力: time_now: epoch 形式の現在時刻、ntp_server_ip: 現在同期中のサーバーアドレス。同期されていない場合は「null」となります。ntp_server_offset: 現在同期中のサーバーとの時刻の遅れ（マイナスの場合は Armadillo がサーバーより早いです）

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/ntp_info
{"ntp_server_ip":"203.0.113.10","ntp_server_offset":"-0.000015824","time_now":1710139558}
http code: 200
```

- **NTP の設定確認**
  GET "/api/time/ntp_config"
  必要権限: TimeView
  パラメータ: 無し
  出力: servers: 同期する対象、 initstepslew: Armadillo 起動時に同期するかどうかの設定

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/ntp_config
{"servers":["pool pool.ntp.org iburst"],"initstepslew":"10 pool.ntp.org"}
http code: 200
```

- **NTP の設定**
  POST "/api/time/ntp_config"
  必要権限: TimeAdmin
  パラメータ: servers: 同期する対象、 initstepslew: Armadillo 起動時に同期するかどうかの設定。パラメータを送信しない場合は設定されません。値が空の場合は設定が削除されて、「 default 」の場合は Armadillo Base OS のデフォルトに戻ります。
  出力: 取得時と同じ

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/ntp_config -X POST -d "servers=server ⏎
203.0.113.10 iburst" -d "servers=server 203.0.113.11 iburst" -d "initstepslew="
{"servers":["server 203.0.113.10 iburst","server 203.0.113.11 iburst"],"initstepslew":null}
http code: 200
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/ntp_config -X POST -d ⏎
"servers=default&initstepslew=default"
{"servers":["pool pool.ntp.org iburst"],"initstepslew":"10 pool.ntp.org"}
http code: 200
```

- **タイムゾーンの確認**
  GET "/api/time/timezone"

必要権限: TimeView
パラメータ: 無し
出力: timezone: 使用されているタイムゾーン

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/timezone
{"timezone":"Asia/Tokyo"}
http code: 200
```

- **タイムゾーンの設定**
  POST "/api/time/timezone"
  必要権限: TimeAdmin
  パラメータ: timezone: 設定するタイムゾーン
  出力: 無し

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/timezone -X POST -d "timezone=Asia/
Tokyo"
http code: 200
```

- **時刻を強制的に設定する**
  POST "/api/time/set"
  必要権限: TimeAdmin
  パラメータ: timestamp: epoch 形式の時刻
  出力: 無し

```
[ATDE ~]$ curl_rest https://armadillo.local:58080/api/time/set -X POST -d "timestamp=$(date +%s)"
http code: 200
```

6.12.6.14. Rest API : 電源制御

- **再起動**
  POST "/api/reboot"
  必要権限: Reboot
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/reboot

http code: 200
```

- **停止**
  POST "/api/poweroff"
  必要権限: Poweroff
  パラメータ: 無し
  出力: 無し

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/poweroff

http code: 200
```

- **スリープ (サスペンド)** (ABOS 3.21.3-at.14 以降)
  POST "/api/sleep"
  必要権限: Sleep
  パラメータ: 無し
  出力: 無し (リクエストは起床時にリターンします)

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/sleep

http code: 200
```

- **スリープ禁止管理** (ABOS 3.21.3-at.14 以降)
  POST "/api/inhibit_sleep"
  必要権限: Sleep
  パラメータ: inhibit: true か空白で禁止し、false で許可します。禁止中に sleep 命令が入った場合は許可した直後に実行されます。禁止命令を複数実行した場合は同じ数の許可命令で再びサスペンド可能になります。
  出力: 無し

```
[ATDE ~]$ curl_rest -d inhibit=true https://armadillo.local:58080/api/inhibit_sleep

http code: 200
```

6.12.6.15. Rest API : ABOS Web 制御

- **リスタート**
  POST "/api/abosweb/restart"
  必要権限: AbosWebRestart
  パラメータ: 無し
  出力: コネクションリセット。ABOS Web はリスタートする前に一度終了するためコネクションリセットが発生します。

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/abosweb/restart

http code: 000
curl: (52) Empty reply from server
```

6.12.6.16. Rest API : ユーザー設定とユーザーデータの管理

- **ユーザー設定とユーザーデータの削除**
  POST "/api/reset_default"
  必要権限: ResetDefault
  パラメータ: 無し
  出力: abos-ctrl reset-default の出力 (stdout または stderr)、および出力ステータス (exit_code または exit_signal)

```
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/reset_default
{"stdout":"rm -f /etc/NetworkManager/system-connections/*¥n"}
{"stdout":"persist_file -r /etc/NetworkManager/system-connections¥n"}
{"stdout":"persist_file -r /etc/dnsmasq.d¥n"}
{"stdout":"rc-service dnsmasq restart¥n"}
```

```
{"stdout":"/etc/init.d/iptables save¥n"}
{"stdout":"sed -i -e '/NETAVARK/d' /etc/iptables/rules-save¥n"}
{"stdout":"persist_file /etc/iptables/rules-save¥n"}
{"stdout":"podman stop -a¥n"}
{"stdout":"find /var/app/volumes /var/log -mindepth 1 -delete¥n"}
{"stdout":"Starting clone to /dev/mmcblk0p1¥n"}
{"stdout":"Cloning rootfs¥n"}
{"stdout":"Updating appfs snapshots¥n"}
{"stdout":"Reusing up-to-date bootloader¥n"}
{"stdout":"Rollback clone successful¥n"}
{"stderr":"WARNING: Rebooting!¥n"}
{"exit_code":0}

http code: 200
```

6.12.6.17. Rest API : カスタムスクリプトの実行

ユーザが Armadillo に追加したスクリプトを Rest API を使用して実行することができます。実行したいスクリプトに実行権限を付与し、Armadillo の /etc/atmark/abos_web/customize_rest ディレクトリ下に置いてください。

実行に root 権限が必要なスクリプトの場合は、 以下のように /etc/doas.d/abos_web_customize.conf にスクリプトを追加してください。

```
[armadillo ~]# cat /etc/doas.d/abos_web_customize.conf
permit nopass abos-web-admin as root cmd /etc/atmark/abos_web/customize_rest/root_command.sh
```

- **任意のスクリプト実行**
  POST "/api/custom/{script}"
  必要権限: Custom
  パラメータ: **args** でスクリプトの引数を順番に指定できます。
  root を true に設定すると root 権限でスクリプトを実行します。
  出力: /etc/atmark/abos_web/customize_rest/{script} {args} {args...} を実行して、そのスクリプトの出力を stdout/stderr で返します。スクリプトが終了した際の出力ステータスは exit_code または exit_signal (どちらも int) です。

```
[armadillo ~]# cat /etc/atmark/abos_web/customize_rest/print_args.sh
#!/bin/sh

printf "arg: %s¥n" "$@"
[ATDE ~]$ curl_rest -X POST https://armadillo.local:58080/api/custom/print_args.sh ¥
      -H 'Content-type: application/json' -d '{"args": ["param", "second arg"], "root":false}'
{"stdout":"arg: param¥n"}
{"stdout":"arg: second arg¥n"}
{"exit_code":0}
```



標準の ABOS Web には最小限の権限しか与えていません。
root 権限でスクリプトを実行する場合、 Armadillo の故障やセキュリティにも関わりますので、十分注意して追加してください。

6.12.7. カスタマイズ

ABOS Web をお客様の最終製品へ組み込む場合に、ロゴ画像や背景色、メニューの文言などをカスタマイズすることができます。詳細は 「3.11. ABOS Web をカスタマイズする」 を参照してください。

6.12.8. ユーザー設定とユーザーデータの削除

カスタマイズと Rest API トークン以外の設定内容と、ユーザーデータを一括削除することができます。

ユーザーデータの削除では以下のデータを削除します。

- /var/app/volumes ディレクトリ下のファイルを全て
- /var/log ディレクトリ下のファイルを全て

ABOS Web のトップページから、「設定管理」ページへ移動し「ユーザー設定とユーザーデータの削除」にある「削除」ボタンを押すと削除できます。削除後は Armadillo が再起動するので引き続き ABOS Web を使用する場合は、再起動が完了してからアクセスしてください。

6.12.9. ABOS Web を停止する

「図 6.121. ABOS Web を停止する」に ABOS Web のサービスを停止する方法を示します。

```
[armadillo ~]# rc-update | grep abos-web ❶
             abos-web |      default
[armadillo ~]# rc-service abos-web status ❷
 * status: started
[armadillo ~]# rc-service abos-web stop ❸
abos-web                 | * Stopping abos-web ... [ ok ]
[armadillo ~]# rc-update del abos-web ❹
 * service abos-web deleted from runlevel default
[armadillo ~]# persist_file -d /etc/runlevels/default/abos-web ❺
```

図 6.121 ABOS Web を停止する

❶ OpenRC に ABOS Web のサービスが登録されていることを確認します。

❷ ABOS Web のサービスが起動していることを確認します。

❸ ABOS Web のサービスを停止します。

❹ サービスを管理している OpenRC から ABOS Web のサービスの登録を解除します。

❺ サービス設定ファイルの削除を永続化します。

ABOS Web を停止すると ABOS Web の Rest API も使用できなくなります。

6.12.10. ABOS Web を起動する

「図 6.122. ABOS Web を起動する」に ABOS Web のサービスを起動する方法を示します。

```
[armadillo ~]# rc-update | grep abos-web ❶
[armadillo ~]# rc-update add abos-web ❷
```

```
* service abos-web added to runlevel default
[armadillo ~]# rc-service abos-web start ❸
abos-web                | * Starting abos-web ... [ ok ]
[armadillo ~]# persist_file /etc/runlevels/default/abos-web ❹
```

図 6.122 ABOS Web を起動する

❶ OpenRC に ABOS Web のサービスが登録されていないことを確認します。何も出力されなければ登録されていません。

❷ サービスを管理している OpenRC に ABOS Web のサービスを登録します。

❸ ABOS Web のサービスを起動します。

❹ サービス設定ファイルを永続化します。

6.12.11. ABOS Web のセキュリティ対策

ABOS Web は開発時には便利ですが、運用時にはできることの多さ故に外部からの攻撃面になり得ます。ここでは、 ABOS Web が具備しているセキュリティ対策機能について紹介します。

6.12.11.1. 同一 LAN 以外からのアクセス禁止

「6.12.1. ABOS Web ではできないこと」でも紹介している通り、 ABOS Web は同一 LAN 内からのみ接続でき、それ以外の接続は拒否します。これによって ABOS Web はインターネット側からの攻撃面となり得ず、攻撃者は同一の LAN 内に侵入してから ABOS Web にアクセスする必要があります。

6.12.11.2. コンテナ以外のアクセスを制限

ABOS Web に設定を追加することで、コンテナからのみ REST API を使用することが可能です。

この設定を行うためには /etc/atmark/abos_web/init.conf を作成し、以下の内容を記載します。

```
command_args="--listen-addr 10.88.0.1:58080 --free-bind"
```

図 6.123 コンテナからのアクセスのみを許可する設定

作成後、ファイルを永続化して Armadillo を再起動します。

コマンド引数の意味は以下の通りです：

オプション	説明
--listen-addr 10.88.0.1:58080	コンテナのデフォルトネットワークでは 10.88.0.0/24 を利用しています。こちらのアドレスにバインドすると、コンテナ以外からアクセスできなくなります。
--free-bind	abos-web 起動時に podman0 インタフェースが存在しなくても正しく起動するためのオプションです。ABOS 3.21.3-at.14 以降が必要です。

6.12.11.3. ABOS Web のユーザ認証

ABOS Web はユーザ認証としてパスワードによる認証を行います。初回起動時には必ずパスワードの設定を求められます。このとき設定するパスワードは 8 文字以上のものに強制しています。

また、ユーザ認証のブルートフォースアタック対策として、パスワードを間違える等でユーザ認証に失敗した場合に、次回再認証までの 2 秒間はパスワードの入力を受け付けない実装になっています。これによって、機械的に総当りでパスワードを解析するまでの時間が大幅に増大することが見込めます。

6.12.11.4. 通信の暗号化

ABOS Web の通信は TLS によって暗号化されています。使用する TLS 証明書は ECDSA (secp384r1) の暗号技術を用いて生成されています。

6.13. ABOSDE から ABOS Web の機能を使用する

ABOSDE は以下に示す ABOS Web の情報取得や動作を行うことができます。

- Armadillo の SWU バージョンを取得する
- Armadillo のコンテナの情報を取得する
- Armadillo のコンテナを起動・停止する
- Armadillo のコンテナのログを取得する
- Armadillo に SWU をインストールする

ABOSDE は ABOS Web の Rest API を用いて通信を行っていますので、ABOS Web にパスワードでログインができる状態である必要があります。ABOS Web へのログインを行っていない場合は「3.9.1. ABOS Web とは」を参考にしてください。

ABOSDE から ABOS Web の機能を使用するには通信を行う対象の Armadillo を選択する必要があります。「図 6.124. ABOSDE で ローカルネットワーク上の Armadillo をスキャンする」 の赤枠で囲まれているボタンをクリックすることで、ローカルネットワーク上で ABOS Web が動作している Armadillo をスキャンすることができます。ただし、ATDE のネットワークを NAT に設定している場合は Armadillo がリストに表示されません。

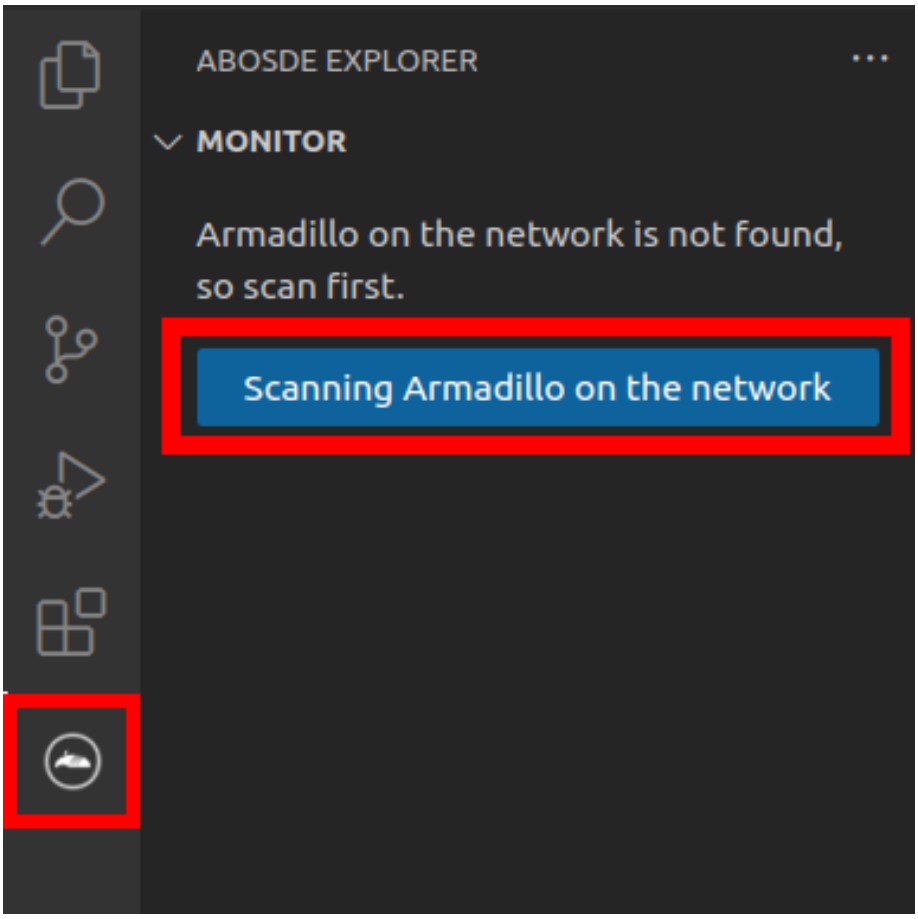


図 6.124 ABOSDE で ローカルネットワーク上の Armadillo をスキャンする

ABOSDE から ABOS Web に初めて通信を行う時、ABOS Web は通信に使用するためのトークンを発行します。そのため、ABOSDE では 「図 6.125. ABOSDE の ABOS Web パスワード入力画面」 のように ABOS Web のパスワードを求められますので、設定したパスワードを入力してください。

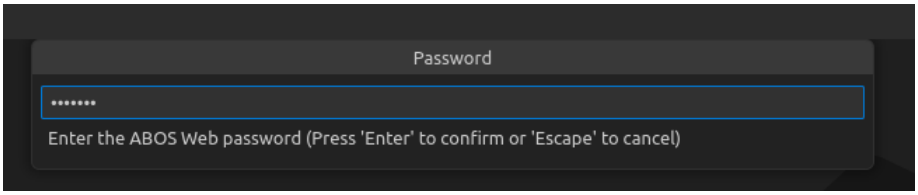


図 6.125 ABOSDE の ABOS Web パスワード入力画面

6.13.1. Armadillo の SWU バージョンを取得する

ローカルネットワーク上の Armadillo をスキャンした後に、「図 6.126. ABOSDE で Armadillo の SWU バージョンを取得」 の赤枠で囲まれているボタンをクリックすることで、選択した Armadillo の SWU バージョンを取得することができます。

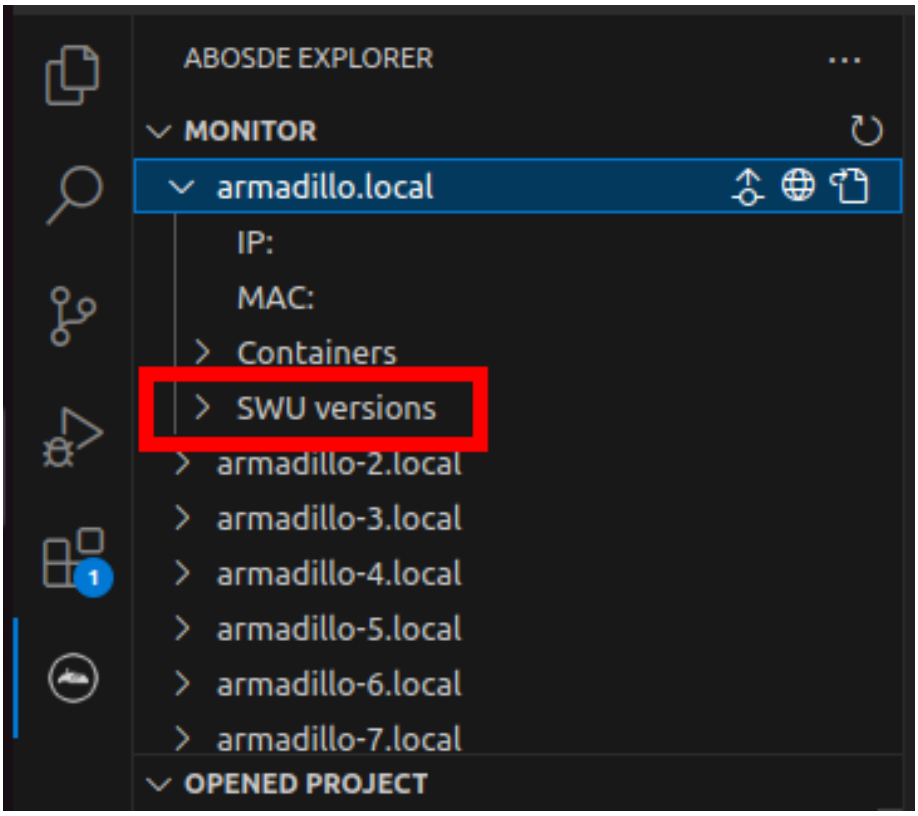


図 6.126 ABOSDE で Armadillo の SWU バージョンを取得

6.13.2. Armadillo のコンテナの情報を取得する

ローカルネットワーク上の Armadillo をスキャンした後に、「図 6.127. ABOSDE で Armadillo のコンテナ情報を取得」 の赤枠で囲まれているボタンをクリックすることで、選択した Armadillo のコンテナの情報を取得できます。表示されるコンテナの情報は以下の通りとなります。

- **state**：コンテナが起動中の場合は running、コンテナが停止中の場合は exited
- **image**：コンテナのイメージ名
- **command**：コンテナ起動時に実行しているコマンド

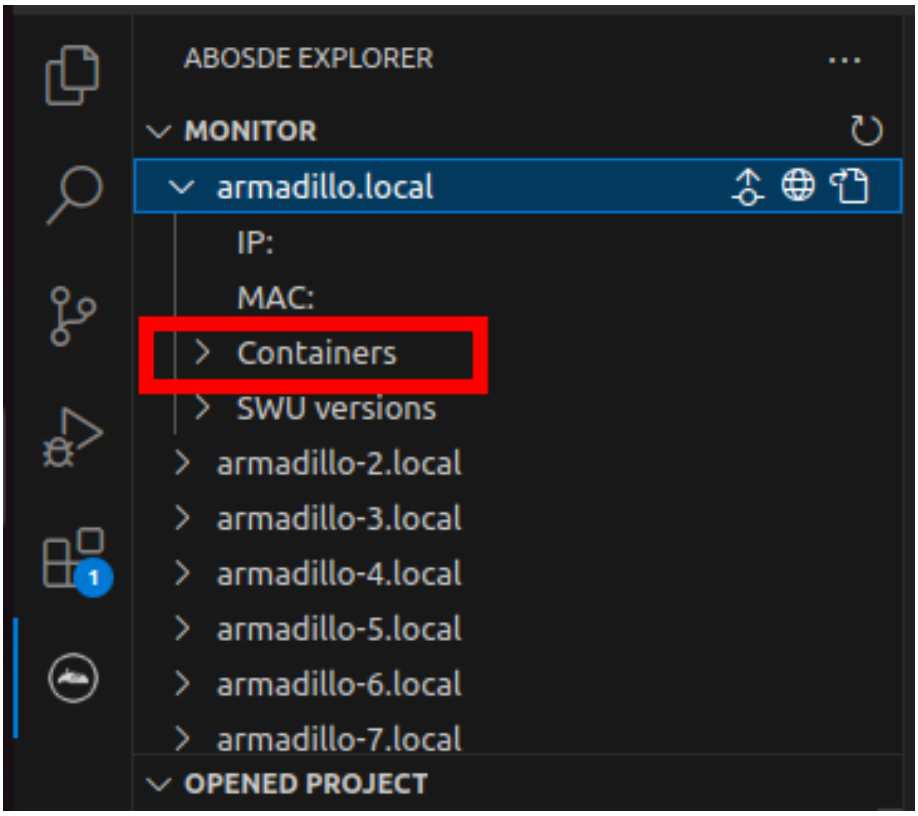


図 6.127 ABOSDE で Armadillo のコンテナ情報を取得

6.13.3. Armadillo のコンテナを起動・停止する

ローカルネットワーク上の Armadillo をスキャンした後に、「図 6.128. ABOSDE で Armadillo のコンテナを起動」 の赤枠で囲まれているボタンをクリックすることで、選択したコンテナを起動することができます。コンテナを起動できた場合はコンテナの status が running に変化します。また、「図 6.129. ABOSDE で Armadillo のコンテナを停止」 の赤枠で囲まれているボタンをクリックすることで、選択したコンテナを停止することができます。コンテナを停止できた場合はコンテナの status が exited に変化します。

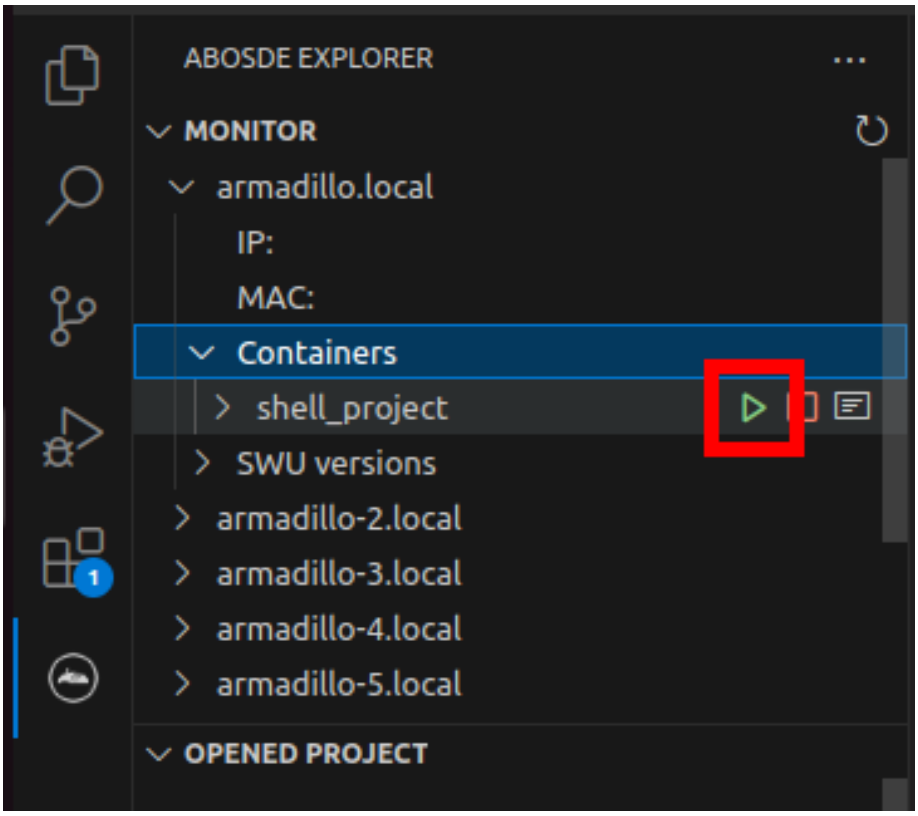


図 6.128 ABOSDE で Armadillo のコンテナを起動

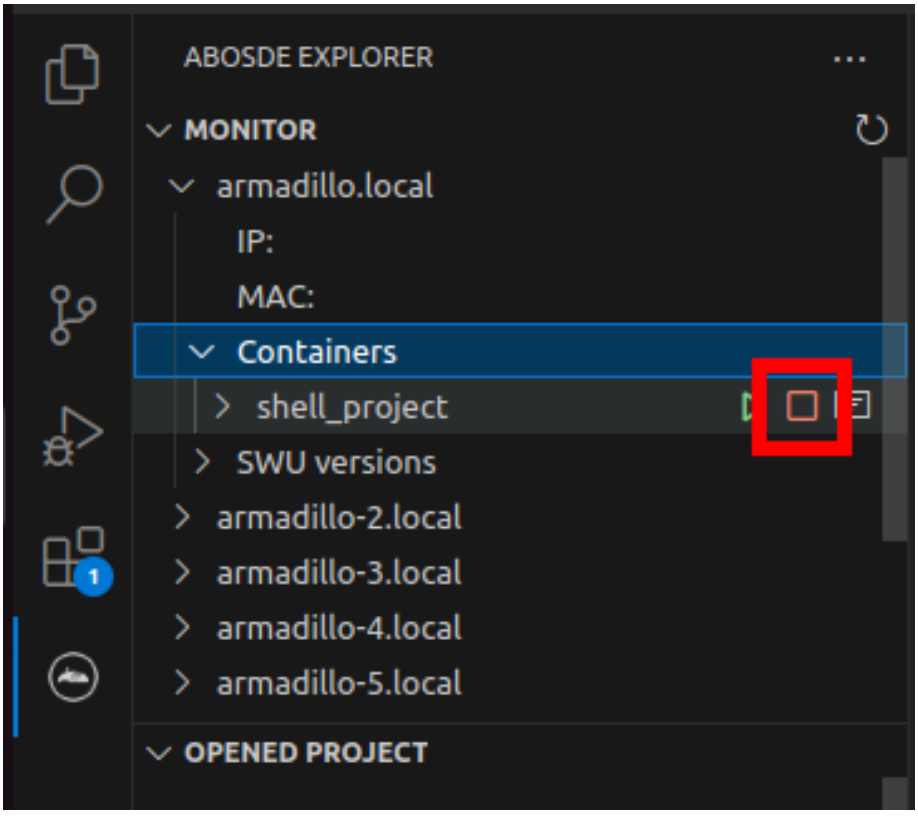


図 6.129 ABOSDE で Armadillo のコンテナを停止

6.13.4. Armadillo のコンテナのログを取得する

「図 6.130. ABOSDE で Armadillo のコンテナのログを取得」 の赤枠で囲まれているボタンをクリックすることで、コンテナが出力したログを取得することができます。ログは VS Code のテキストエディタに開かれます。コンテナが何もログを出力していない場合は表示されません。

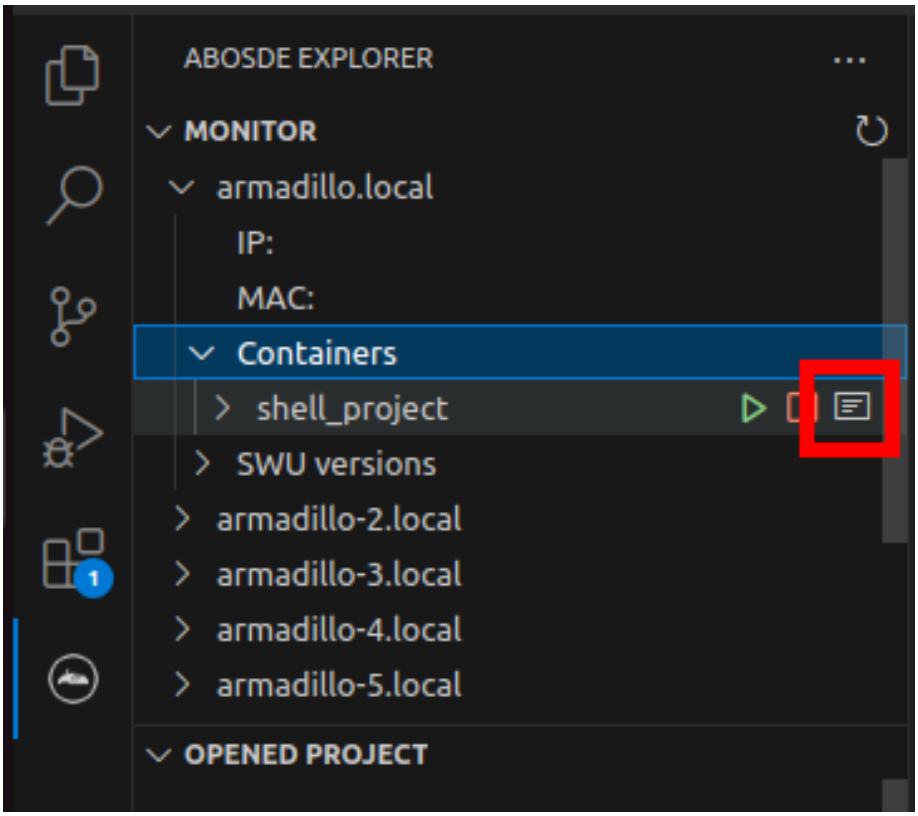


図 6.130 ABOSDE で Armadillo のコンテナのログを取得

6.13.5. Armadillo に SWU をインストールする

ローカルネットワーク上の Armadillo をスキャンした後に、「図 6.131. ABOSDE で Armadillo に SWU をインストール」 の赤枠で囲まれているボタンをクリックすることで、選択した Armadillo に SWU をインストールすることができます。SWU インストールのログは VS Code 画面下部の OUTPUT に表示されます。

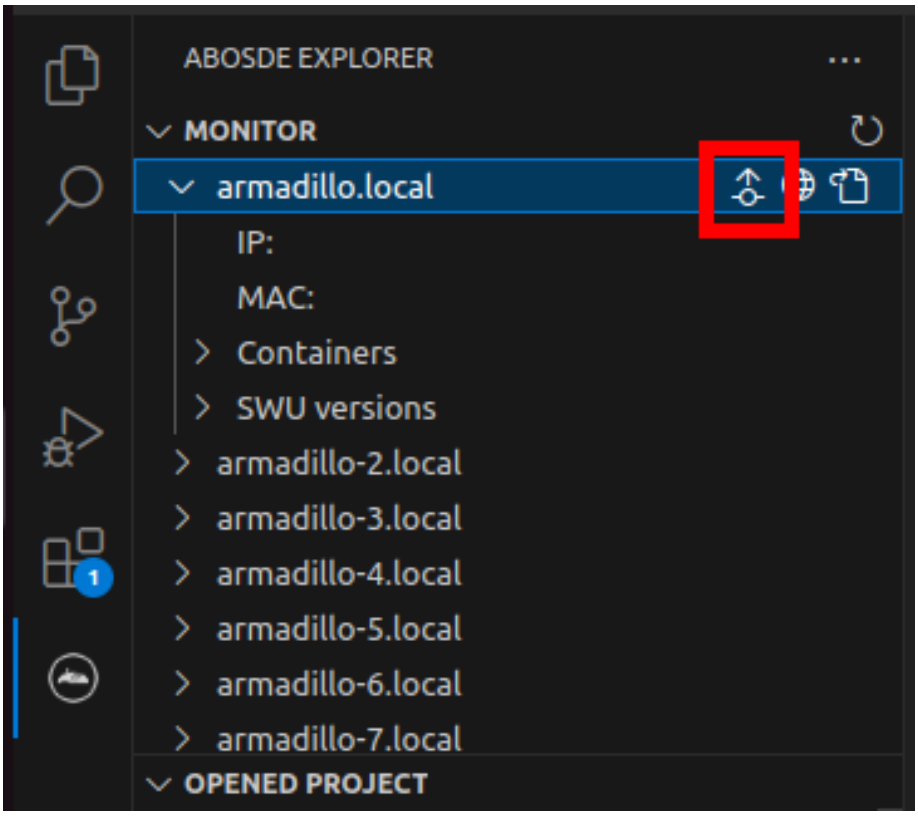


図 6.131 ABOSDE で Armadillo に SWU をインストール

6.14. ssh 経由で Armadillo Base OS にアクセスする

Armadillo-IoT ゲートウェイ A6E には openssh がインストールされていますが、デフォルトでは SSH サーバーが起動していません。

SSH サーバーを自動的に起動するようにするためには、以下のコマンドを実行してください。

```
[armadillo:~]# rc-update add sshd
 * service sshd added to runlevel default
[armadillo ~]# persist_file /etc/runlevels/default/sshd
[ 2819.277066] EXT4-fs (mmcblk0p1): re-mounted. Opts: (null)
[armadillo ~]# reboot
```

上記の例では、再起動後も設定が反映されるように、 `persist_file` コマンドで eMMC に設定を保存しています。

6.15. 入力電圧監視サービス (power-alertd) を使用する

バッテリー駆動時など入力電圧が変化する環境で入力電圧を周期的に監視し、設定値以上・以下に電圧が変化した際に行うアクションを定義することができる power-alertd サービスが存在します。



スタンダードタイプ WLAN/LAN モデルの場合、A/D コンバータが搭載されておらず、入力電圧を監視できないため、この機能を使用できません。

このサービスを使用することで、入力電圧が想定外の値になった際外部に通知する、システムが稼働不可能になる前に安全にシャットダウンするなどのアクションが可能となります。

本章では、本サービスの使用方法を説明します。

6.15.1. 入力電圧監視サービス (power-alertd) の設定

最初に設定ファイル /etc/atmark/power-alertd.conf を編集します。POWER_ALERTD_ARGS= の後に設定する引数を記載します。記載例を「図 6.132. /etc/atmark/power-alertd.conf の記載例」に示します。オプションの詳細を「表 6.26. POWER_ALERTD_ARGS に記載するオプションの説明」に示します。

```
POWER_ALERTD_ARGS="-u 11000 -a COMMAND"
```

図 6.132 /etc/atmark/power-alertd.conf の記載例

表 6.26 POWER_ALERTD_ARGS に記載するオプションの説明

オプション	説明
-o --over	入力電圧が指定電圧以上になった場合、-a / --action で指定した処理を行います。
-u --under	入力電圧が指定電圧以下になった場合、-a / --action で指定した処理を行います。
-c --critial	入力電圧が指定電圧以下になった場合、-a / --action で指定した処理を行い、60 秒後 poweroff コマンドで Armadillo の電源をオフにします。
-o --oneshot	入力電圧が -u / -o で指定した電圧以上・以下になった場合、 -a / --action で指定した処理を行い、power-alertd を終了します。
-i --interval	計測周期を秒で指定できます。指定しない場合 60 秒周期で計測します。

設定値の有効・永続化には「図 6.133. /etc/atmark/power-alertd.conf の永続化」に示すコマンドを使用します。

```
[armadillo ~]# persist_file -P /etc/atmark/power-alertd.conf
```

図 6.133 /etc/atmark/power-alertd.conf の永続化

6.15.2. 入力電圧監視サービス (power-alertd) の有効・無効化

設定ファイルを記載した後、入力電圧監視サービス (power-alertd) を有効にします。有効にする手順を「図 6.134. 入力電圧監視サービス (power-alertd) を有効にする」に示します。

```
[armadillo ~]# rc-update add power-alertd default
[armadillo ~]# persist_file -P /etc/runlevels/default/power-alertd
```

図 6.134 入力電圧監視サービス (power-alertd) を有効にする

入力電圧監視サービス (power-alertd) を無効にする手順を「図 6.135. 入力電圧監視サービス (power-alertd) を無効にする」に示します。

```
[armadillo ~]# rc-update del power-alertd default
[armadillo ~]# persist_file -d /etc/runlevels/default/power-alertd
```

図 6.135 入力電圧監視サービス (power-alertd) を無効にする

6.16. コマンドラインからネットワーク設定を行う

ここでは、コマンドラインによるネットワークの設定方法について説明します。

6.16.1. 接続可能なネットワーク

表 6.27 ネットワークとネットワークデバイス

ネットワーク	搭載モデル	ネットワークデバイス	出荷時の設定
Ethernet	全モデル	eth0	DHCP
LTE	Cat.1 bis	ppp0	無し
無線 LAN	Cat.1 bis[a], WLAN	wlan0	クライアントモード

[a]型番によっては、搭載/非搭載が異なります。

6.16.2. ネットワークの設定方法

Armadillo-IoT ゲートウェイ A6E では、通常の Linux システムと同様、ネットワークインターフェースの設定は NetworkManager を使用します。NetworkManager はすべてのネットワーク設定をコネクションとして管理します。コネクションには「どのようにネットワークへ接続するか」、「どのようにネットワークを作成するか」を記述し、 /etc/NetworkManager/system-connections/ に保存します。また、1 つのデバイスに対して複数のコネクションを保存することは可能ですが、1 つのデバイスに対して有効化にできるコネクションは 1 つだけです。

NetworkManager は、従来の /etc/network/interfaces を使った設定方法もサポートしていますが、本書では nmcli を用いた方法を中心に紹介します。

6.16.2.1. nmcli について

nmcli は NetworkManager を操作するためのコマンドラインツールです。「図 6.136. nmcli のコマンド書式」に nmcli の書式を示します。このことから、 nmcli は「オブジェクト (OBJECT) というものが存在し、それぞれのオブジェクトに対してコマンド (COMMAND) を実行する。」という書式でコマンドを入力することがわかります。また、オブジェクトそれぞれに help が用意されていることもここから読み取れます。

```
nmcli [ OPTIONS ] OBJECT { COMMAND | help }
```

図 6.136 nmcli のコマンド書式

6.16.3. nmcli の基本的な使い方

ここでは nmcli の、基本的な使い方を説明します。

6.16.3.1. コネクションの一覧表示

登録されているコネクションの一覧表示するには、「図 6.137. コネクションの一覧表示」に示すコマンドを実行します。[1]

```
[armadillo ~]# nmcli connection
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  ethernet  eth0
```

図 6.137 コネクションの一覧表示

表示された NAME については、以降 [ID] として利用することができます。

6.16.3.2. コネクションの有効化・無効化

コネクションを有効化するには、「図 6.138. コネクションの有効化」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection up [ID]
```

図 6.138 コネクションの有効化

コネクションを無効化するには、「図 6.139. コネクションの無効化」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection down [ID]
```

図 6.139 コネクションの無効化

6.16.3.3. コネクションの作成

コネクションを作成するには、「図 6.140. コネクションの作成」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection add con-name [ID] type [type] ifname [interface name]
```

図 6.140 コネクションの作成

[ID] にはコネクションの名前(任意)、[type] には ethernet、wifi といった接続タイプ、[interfacename] にはインターフェース名(デバイス)を入力します。これにより /etc/NetworkManager/system-connections/ に[ID]の名前でコネクションファイルが作成されます。このファイルを vi などで編集し、コネクションを修正することも可能です。

Armadillo-IoT ゲートウェイ A6E を再起動したときにコネクションファイルが消えてしまわないように、persist_file コマンドで永続化する必要があります。persist_file コマンドに関する詳細は 「6.2. persist_file について」 を参照してください。

```
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/<コネクションファイル名>
```

図 6.141 コネクションファイルの永続化

[1] nmcli connection show [ID] によって、より詳細な情報を表示することもできます。



別の Armadillo-IoT ゲートウェイ A6E からコネクションファイルをコピーした場合は、コネクションファイルのパーミッションを 600 に設定してください。600 に設定後、 nmcli c reload コマンドでコネクションファイルを再読込します。

```
[armadillo ~]# chmod 600 /etc/NetworkManager/system-connections/<コネクションファイル名>
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/<コネクションファイル名>
[armadillo ~]# nmcli c reload
```

swu イメージを使用してコネクションファイルのアップデートを行う場合は、swu イメージに含めるコネクションファイルのパーミッションを 600 に設定してから、swu イメージを作成してください。アップデート実行時には swu イメージ作成時のパーミッションが維持されるため、上記のコマンド実行手順は不要です。swu イメージに関しては 「3.3.4.5. SWU イメージのインストール」 を参考にしてください。

6.16.3.4. コネクションの削除

コネクションを削除するには、「図 6.142. コネクションの削除」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection delete [ID]
```

図 6.142 コネクションの削除

これにより /etc/NetworkManager/system-connections/ のコネクションファイルも同時に削除されます。コネクションの作成と同様に persist_file コマンドで永続化する必要があります。

```
[armadillo ~]# persist_file -d /etc/NetworkManager/system-connections/<コネクションファイル名>
```

図 6.143 コネクションファイル削除時の永続化

6.16.3.5. 固定 IP アドレスに設定する

「表 6.28. 固定 IP アドレス設定例」の内容に設定する例を、「図 6.144. 固定 IP アドレス設定」に示します。

表 6.28 固定 IP アドレス設定例

項目	設定
IP アドレス	192.0.2.10
マスク長	24
デフォルトゲートウェイ	192.0.2.1

```
[armadillo ~]# nmcli connection modify [ID] ¥
ipv4.method manual ipv4.addresses 192.0.2.10/24 ipv4.gateway 192.0.2.1
```

図 6.144 固定 IP アドレス設定

6.16.3.6. DHCP に設定する

DHCP に設定する例を、「図 6.145. DHCP の設定」に示します。

```
[armadillo ~]# nmcli connection modify [ID] ipv4.method auto
```

図 6.145 DHCP の設定



-ipv4.addresses のように、プロパティ名の先頭に "-" を付けることで設定したプロパティを削除することができます。反対に "+" を付けることでプロパティを追加することができます。

6.16.3.7. DNS サーバーを指定する

DNS サーバーを指定する例を、「図 6.146. DNS サーバーの指定」に示します。

```
[armadillo ~]# nmcli connection modify [ID] ipv4.dns 192.0.2.1
```

図 6.146 DNS サーバーの指定

6.16.3.8. コネクションの修正を反映する

有効化されているコネクションを修正した場合、かならず修正したコネクションを再度有効化してください。

```
[armadillo ~]# nmcli connection down [ID]
[armadillo ~]# nmcli connection up [ID]
```

図 6.147 コネクションの修正の反映

6.16.3.9. デバイスの一覧表示

デバイスの一覧(デバイス名、タイプ、状態、有効なコネクション)を確認するには、「図 6.148. デバイスの一覧表示」に示すコマンドを実行します。

```
[armadillo ~]# nmcli device
DEVICE  TYPE      STATE        CONNECTION
eth0    ethernet  connected    Wired connection 1
lo      loopback  unmanaged    --
```

図 6.148 デバイスの一覧表示

6.16.3.10. デバイスの接続

デバイスを接続するには、「図 6.149. デバイスの接続」に示すコマンドを実行します。

```
[armadillo ~]# nmcli device connect [ifname]
```

図 6.149 デバイスの接続



デバイスを接続するには、接続しようとしているデバイスの有効なコネクションが必要です。"Error: neither a valid connection nor device given" というメッセージが表示された場合には、`nmcli connection` などで有効なコネクションが存在するかを確認してください。

6.16.3.11. デバイスの切断

デバイスを切断するには、「図 6.150. デバイスの切断」に示すコマンドを実行します。

```
[armadillo ~]# nmcli device disconnect [ifname]
```

図 6.150 デバイスの切断

6.16.4. 有線 LAN の接続を確認する

有線 LAN で正常に通信が可能かを確認します。設定を変更した場合、必ず変更したインターフェースを再度有効化してください。

同じネットワーク内にある通信機器と PING 通信を行います。以下の例では、通信機器が「192.0.2.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping -c 3 192.0.2.20
PING 192.0.2.20 (192.0.2.20): 56 data bytes
64 bytes from 192.0.2.20: seq=0 ttl=64 time=3.056 ms
64 bytes from 192.0.2.20: seq=1 ttl=64 time=1.643 ms
64 bytes from 192.0.2.20: seq=2 ttl=64 time=1.633 ms

--- 192.0.2.20 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1.633/2.110/3.056 ms
```

図 6.151 有線 LAN の PING 確認



有線 LAN 以外のインターフェースが有効化されている場合、ルーティングの設定などにより、ネットワーク通信に有線 LAN が使用されない場合があります。確実に有線 LAN の接続確認をするために、有線 LAN 以外のインターフェースを無効化してください。

6.16.5. LTE

本章では、Armadillo-IoT ゲートウェイ A6E に搭載されている LTE モジュールの使用方法について説明します。



Armadillo-IoT ゲートウェイ A6E に搭載しております SIMCom 製 LTE 通信モジュール SIM7672G 、SIM7672JP は、ドコモ/KDDI の相互接続性試験を完了しています。ファームウェアのバージョンと相互接続性試験の完了状況の対応は以下のとおりです。

表 6.29 ファームウェアのバージョンと相互接続性試験の完了状況

LTE モジュール	ファームウェアバージョン	相互接続性試験完了状況
SIM7672G	2388B03SIM767XM5A_M	ドコモ/KDDI
SIM7672JP	2388B02SIM767XM5A	ドコモ/KDDI

ファームウェアのバージョンは以下のように確認できます。「Revision:」がバージョンになります。

```
[armadillo ~]# send-at /dev/ttymxc3 AT+SIMCOMATI echo sim7672
AT+SIMCOMATI
Manufacturer: SIMCOM INCORPORATED
Model: SIM7672G-MNGV
Revision: 2388B03SIM767XM5A_M
SIM767XM5_B03V01_250619
IMEI: 865031061369335
OK
```

上記のバージョンでない場合は、ファームウェアのアップデートをお願いします。ファームウェア本体、およびアップデート手順は https://armadillo.atmark-techno.com/resources/software/partner-solution/sim7672g-software で公開しています。

上記のバージョン以外のファームウェアを使用しても技適違反にはなりませんが、通信トラブル等があった際にキャリアのサポートが得られない等の可能性があります。

6.16.5.1. LTE データ通信設定を行う前に

LTE データ通信を利用するには、通信事業者との契約が必要です。契約時に通信事業者から貸与された nanoSIM(UIM カード)と APN 情報を準備します。



Armadillo-IoT ゲートウェイ A6E 動作検証済み nanoSIM (料金プラン)に関しては、 Armadillo サイトの「Armadillo-IoT ゲートウェイ 動作確認済み SIM 一覧」を確認ください。

Armadillo-IoT ゲートウェイ 動作確認済み SIM 一覧 [https://armadillo.atmark-techno.com/howto/armadillo-iot-tested-sim]



Armadillo-IoT ゲートウェイ A6E の電源が切断されていることを確認してから nanoSIM(UIM カード)を取り付けてください。



本製品は、nanoSIM スロットを搭載しています。

標準/microSIM サイズの SIM カードを nanoSIM サイズにカットしたもの、サイズの異なるものを使用すると、nanoSIM スロットが故障する原因となります。これらを使用し本製品が故障した場合は、保証期間内であっても保証適用外となります。

nanoSIM(UIM カード)の切り欠きを挿入方向に向け、刻印面を上にして挿入してください。挿入位置などは、「図 3.97. nanoSIM カードの接続例」 を参照してください。

APN の設定を行うには、「表 6.30. APN 設定情報」に示す情報が必要です。各設定値の文字長を超える設定はできませんので、SIM の料金プランを選択する際にはご注意ください。

表 6.30 APN 設定情報

項目	Armadillo-IoT ゲートウェイ A9E (SIM7672G/SIM7672JP 搭載モデル)
APN	最大 99 文字
ユーザー名	最大 64 文字
パスワード	最大 64 文字
認証方式	PAP または CHAP
PDP Type	IP のみをサポート

6.16.5.2. 省電力などの設定

LTE モジュール SIM7672G、SIM7672JP 起動時に設定する内容を、設定ファイル(/etc/atmark/sim7672-boot.conf)に記載します。

設定ファイルの記載例として、サンプルファイル(/etc/atmark/sim7672-boot.conf.example)がありますので、こちらをリネームまたはコピーしてご利用ください。

/etc/atmark/sim7672-boot.conf に設定できる内容を「表 6.31. sim7672-boot.conf の設定内容」に示します。

sim7672-boot.conf のフォーマットは以下の通りです。

- パラメータは、「パラメータ名=値」のフォーマットで記載してください。
- 行頭に # が存在する場合、その行を無視します。
- パラメーターが存在しない場合、その項目に関して何も設定をしません。

表 6.31 sim7672-boot.conf の設定内容

パラメーター名	初期値	設定可能値	説明
psm	disable	disable または tau,act-time	Power Save Mode の設定
edrx	disable	disable または eDRX の値	eDRX の設定

PSM (Power Save Mode) の設定値を「表 6.32. psm の tau と act-time に設定可能な値」に示します。disable にしない場合、tau (Periodic TAU cycle (T3412)) は act_time (Active time (T3324)) より大きい値にする必要があります。

表 6.32 psm の tau と act-time に設定可能な値

パラメーター名	設定可能値
tau (s=秒,m=分,h=時間)	2s,4s,6s…62s,90s,120s,150s…930s,16m,17m,18m…31m,40m,50m,60m…310m,6h,7h,8h…31h,40h,50h,60h…310h,320h,640h,960h…9920h
act-time (s=秒,m=分,h=時間)	2s,4s,6s…62s,1m,2m,3m…31m,36m,42m,48m…186m

eDRX (extended Discontinuous Reception) の設定値を「表 6.33. edrx に設定可能な値」に示します。

PSM と eDRX はどちらかの設定が有効となります。両方共設定した場合は PSM が優先されます。

表 6.33 edrx に設定可能な値

パラメーター名	設定可能値
edrx (秒)	5.12, 10.24, 20.48, 40.96, 61.44, 81.92, 102.4, 122.88, 143.36, 163.84, 327.68, 655.36, 1310.72, 2621.44, 5242.88, 10485.76

6.16.5.3. LTE のコネクションを作成する

「表 6.34. APN 情報設定例」の内容に設定する例を「図 6.152. LTE のコネクションの作成」に示します。

表 6.34 APN 情報設定例

項目	設定
APN	[apn]
ユーザー名	[user]
パスワード	[password]
ネットワークデバイス	ttyCommModem

```
[armadillo ~]# nmcli connection add type gsm ifname ttyCommModem apn [apn] user [user] password ⏎
[password]
Connection 'gsm-[wwan]' (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx) successfully added.
```

図 6.152 LTE のコネクションの作成

コネクション設定を永続化するには、以下のコマンドを入力してください。設定を永続化すると、Armadillo 起動時に自動的にデータ接続を行うようになります。

同一インタフェースへの設定が複数存在する場合、 gsm-ttyCommModem-1.nmconnection など後ろに数値が付与されますので、「図 6.152. LTE のコネクションの作成」 入力時のメッセージで生成されたファイル名を確認した上で永続化を実施ください。

```
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/gsm-ttyCommModem.nmconnection
```

図 6.153 LTE のコネクションの設定の永続化

6.16.5.4. ユーザー名とパスワード設定が不要な LTE のコネクションを作成する

ユーザー名とパスワード設定が不要な SIM カードをご利用の場合、「図 6.154. ユーザー名とパスワード設定が不要な LTE のコネクションの作成」 に示すとおり user と password を省略して設定してください。

```
[armadillo ~]# nmcli connection add type gsm ifname ttyCommModem apn [apn]
Connection 'gsm-[wwan]' (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx) successfully added.
```

図 6.154 ユーザー名とパスワード設定が不要な LTE のコネクションの作成

6.16.5.5. MCC/MNC を指定した LTE のコネクションを作成する

マルチキャリア SIM などを使用する際、MCC (Mobile Country Code) と MNC (Mobile Network Code) を指定してコネクションを作成すると LTE ネットワークに接続できることがあります。指定する場合は 「図 6.155. MCC/MNC を指定した LTE コネクションの作成」 に示すコマンドを実行してください。

[mccmnc] には 44010 などの数字を入力してください。実際に設定する値に関しては、ご契約の通信事業者へお問い合わせください。

```
[armadillo ~]# nmcli connection add type gsm ifname ttyCommModem apn [apn] user [user] password ⏎
[password] gsm.network-id [mccmnc]
```

図 6.155 MCC/MNC を指定した LTE コネクションの作成

6.16.5.6. PAP 認証を有効にした LTE のコネクションを作成する

LTE のコネクションの認証方式は、デフォルトで CHAP に設定されています。PAP 認証を有効にしたコネクションを作成する場合は「図 6.156. PAP 認証を有効にした LTE コネクションの作成」 に示すコマンドを実行してください。

```
[armadillo ~]# nmcli connection add type gsm ifname ttyCommModem apn [apn] user [user] password ⏎
[password] ppp.refuse-eap true ppp.refuse-chap true ppp.refuse-mschap true ppp.refuse-mschapv2 true ⏎
ppp.refuse-pap false
```

図 6.156 PAP 認証を有効にした LTE コネクションの作成



すでに LTE コネクションを作成済みの場合はコネクション設定を削除した後に、「図 6.156. PAP 認証を有効にした LTE コネクションの作成」を実施してください。

6.16.5.7. LTE コネクションを確立する

LTE コネクションの作成直後や設定変更後に再起動をせずにコネクションを確立するには、「図 6.157. LTE のコネクション確立」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection up gsm-ttyCommModem
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/x)
```

図 6.157 LTE のコネクション確立

6.16.5.8. LTE の接続を確認する

ご利用になるサービスとの通信を検証する、ICMP に対応しているアドレス (8.8.8.8 など) と PING 通信を行うなどの方法で LTE の接続を確認してください。

```
[armadillo ~]# ping -c 3 8.8.8.8 -I [network device]
```

図 6.158 LTE の PING 確認

[network device] には、「表 6.27. ネットワークとネットワークデバイス」を参照し、ご使用の製品モデルで使用している LTE のネットワークデバイスを指定してください。

6.16.5.9. LTE コネクションを切断する

LTE コネクションを切断するには、「図 6.159. LTE コネクションを切断する」に示すコマンドを実行します。LTE コネクションを切断する前に、LTE 再接続サービスを停止しないと再接続処理が実行される為、事前に停止します。

```
[armadillo ~]# rc-service connection-recover stop ❶
connection-recover| * Stopping connection-recover ... [ ok ]
[armadillo ~]# nmcli connection down gsm-ttyCommModem ❷
```

図 6.159 LTE コネクションを切断する

❶ LTE 再接続サービスを停止します。

❷ LTE コネクションを切断します。

6.16.5.10. LTE 再接続サービス

LTE 再接続サービスは、LTE のデータ接続の状態を定期的に監視し、切断を検出した場合に再接続を行うサービスです。

Cat.1 bis モデルは初期状態でこのサービスが有効になっております。



閉塞 LTE 網を使用する料金プランをご契約で本サービスをご利用になられる際の注意点。

コネクション状態確認時 PING 送付先の初期値は 8.8.8.8 ですが、この IP アドレスに対して ping 導通ができない場合、 ping 導通可能な IP アドレスを指定する必要があります。

SIM カードが挿入されており、NetworkManager に有効な LTE コネクションの設定がされているとき、初期設定では 120 秒に一度コネクションの状態を監視します。オプションで SIM カードの認識ができないときに Armadillo の再起動を実施することも可能です。

コネクションが無効になっている場合、切断状態と判定しコネクションを有効にします。

コネクションが有効になっている場合、特定の宛先に PING を実行します。PING がエラーになったとき切断状態と判定し、コネクションの無効化・有効化を行うことで再接続を実施します。

コネクションの無効化・有効化による再接続を実施しても PING がエラーになる場合、電波のオン・オフまたは LTE モジュールの電源をオン・オフを実施して LTE 再接続を試みます。どちらを実施するかは設定ファイルの WWAN_FORCE_RESTART_COUNT に依存します。

WWAN_FORCE_RESTART_COUNT が初期値の 10 である場合、1 から 9 回目は電波のオン・オフを実施し、10 回目は LTE モジュールの電源オン・オフを実施します。それ以降も NG が続く場合、同じく 10 回に一度 LTE モジュールの電源オン・オフを実施します。

LTE モジュールが検出できない状態が 2 回連続で発生した場合、LTE モジュールの再起動を実施します。

LTE 接続中状態が 3 回連続で発生した場合、LTE モジュールの再起動を実施します。

工場出荷状態で本サービスは有効化されており、システム起動時にサービスが自動的に開始されます。PING を実行する宛先は、初期設定では "8.8.8.8" です。

設定ファイルの記載例として、サンプルファイル(/etc/atmark/connection-recover.conf.example)がありますので、こちらをリネームまたはコピーしてご利用ください。

設定ファイルの概要を「表 6.35. 再接続サービス設定パラメーター」に示します。必要に応じて設定値を変更してください。

設定ファイルが存在しない場合は初期値で動作します。

表 6.35 再接続サービス設定パラメーター

パラメーター名	初期値	意味	変更
PRODUCT_NAME	-	製品名	不可
CHECK_INTERVAL_SEC	120	監視周期(秒)	可
PING_DEST_IP	8.8.8.8	コネクション状態確認時 PING 送付先	可
DEVICE	-	ネットワークデバイス名	不可
TYPE	-	ネットワークタイプ	不可
NETWORK_IF	-	ネットワーク I/F 名	不可
FORCE_REBOOT	FALSE	TRUE に設定すると PING 導通チェックが 4 回連続 NG だった場合、 Armadillo を再起動します。	可
REBOOT_IF_SIM_NOT_FOUND	FALSE	TRUE に設定すると SIM を検出できない状態が 2 回連続で発生した場合、 Armadillo を再起動します。	可
WWAN_FORCE_RESTART_COUNT	10	PING 導通確認を設定した回数連続で失敗した場合 LTE モジュールを再起動します。設定した回数に満たない場合 、電波のオフ・オン実施のみで LTE 再接続を試みます。	可

設定ファイル変更後、変更内容を永続化するには「図 6.160. LTE 再接続サービスの設定値を永続化する」に示すコマンドを実行してください。

```
[armadillo ~]# persist_file /etc/atmark/connection-recover.conf
```

図 6.160 LTE 再接続サービスの設定値を永続化する

LTE 再接続サービスの状態を確認するには、「図 6.161. LTE 再接続サービスの状態を確認する」に示すコマンドを実行してください。

```
[armadillo ~]# rc-service connection-recover status
 * status: started
```

図 6.161 LTE 再接続サービスの状態を確認する

LTE 再接続サービスを停止するには、「図 6.162. LTE 再接続サービスを停止する」に示すコマンドを実行してください。

```
[armadillo ~]# rc-service connection-recover stop
connection-recover| * Stopping connection-recover ... [ ok ]
```

図 6.162 LTE 再接続サービスを停止する

LTE 再接続サービスを開始するには、「図 6.163. LTE 再接続サービスを開始する」に示すコマンドを実行してください。

```
[armadillo ~]# rc-service connection-recover start
connection-recover| * Starting connection-recover ... [ ok ]
```

図 6.163 LTE 再接続サービスを開始する

独自に接続状態を確認するサービスを実装されるなどの理由で標準の LTE 再接続サービスが不要な場合、「図 6.164. LTE 再接続サービスを無効にする」に示す手順で再接続サービスを永続的に無効にできます。

```
[armadillo ~]# rc-service connection-recover stop ❶
connection-recover| * Stopping connection-recover ... [ ok ]
[armadillo ~]# rc-update del connection-recover default ❷
service connection-recover removed from runlevel default
[armadillo ~]# persist_file -d /etc/runlevels/default/connection-recover ❸
```

図 6.164 LTE 再接続サービスを無効にする

❶ 再接続サービスを停止します。

❷ 再接続サービスを無効にします。

❸ サービス設定ファイルの削除を永続化します。

LTE 再接続サービスを無効化した後、再度有効にする場合、「図 6.165. LTE 再接続サービスを有効にする」に示す手順を実行してください。

```
[armadillo ~]# rc-update add connection-recover default ❶
service connection-recover added to runlevel default
[armadillo ~]# rc-service connection-recover start ❷
connection-recover| * Starting connection-recover ... [ ok ]
[armadillo ~]# persist_file /etc/runlevels/default/connection-recover ❸
```

図 6.165 LTE 再接続サービスを有効にする

❶ 再接続サービスを有効にします。

❷ 再接続サービスを開始します。

❸ サービス設定ファイルを永続化します。

6.16.5.11. ModemManager - mmcli について

ここでは ModemManager と mmcli について説明します。

Armadillo-IoT ゲートウェイ A6E にはネットワークを管理する NetworkManager とは別に、モデムを管理する ModemManager がインストールされています。ModemManager はモバイルブロードバンドデバイス(LTE モジュールなど)の操作および、接続状況の管理などを行います。

ModemManager のコマンドラインツールである **mmcli** を使用することで、LTE 通信の電波強度や SIM カードの情報(電話番号や IMEI など)を取得することが可能です。mmcli の詳しい使いかたについては **man mmcli** を参照してください。

ModemManager はモデムデバイスに応じたプラグインを選択して動作します。 Armadillo-IoT ゲートウェイ A6E では simtech-sim7672 という名称のプラグインで動作しています。

6.16.5.12. mmcli - 認識されているモデムの一覧を取得する

認識されているモデムの一覧を取得するには、「図 6.166. 認識されているモデムの一覧を取得する」に示すコマンドを実行します。

Armadillo Base OS では、Armadillo Base OS が使用している LTE モジュール番号を取得するコマンド **mm-modem-num** を用意しております。

```
[armadillo:~]# mmcli -L
    /org/freedesktop/ModemManager1/Modem/0 [SIMCOM INCORPORATED] SIM7672G-MNGV
```

図 6.166 認識されているモデムの一覧を取得する

6.16.5.13. mmcli - モデムの情報を取得する

モデムの情報を取得するには、「図 6.167. モデムの情報を取得する」に示すコマンドを実行します。

```
armadillo:~# mmcli -m $(mm-modem-num)
  --------------------------------
  General  |                 path: /org/freedesktop/ModemManager[number1]/Modem/[number2]
           |            device id: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  --------------------------------
  Hardware |         manufacturer: SIMCOM INCORPORATED
```

```
            |                model: SIM7672G-MNGV
            |    firmware revision: XXXXXXXXXXXXXXXXXXX
            |            supported: lte
            |              current: lte
            |         equipment id: XXXXXXXXXXXXXXX
:（省略）
```

図 6.167 モデムの情報を取得する



モデムの情報を取得するには、SIM カードが取り付けられている必要があります。正しく SIM カードが取り付けられていることを確認してください。

6.16.5.14. mmcli - SIM の情報を取得する

SIM の情報を取得するには、「図 6.168. SIM の情報を取得する」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m $(mm-modem-num)
:（省略）
SIM       |     primary sim path: /org/freedesktop/ModemManager1/SIM/[number] # [number] を次のコマ ⏎
ンドで使用
:（省略）

[armadillo ~]# mmcli -i [number]
  ------------------------------
  General    |              path: /org/freedesktop/ModemManager1/SIM/0
  ------------------------------
  Properties |            active: yes
             |              imsi: XXXXXXXXXXXXXX
             |             iccid: XXXXXXXXXXXXXXXXXX
             |       operator id: XXXXX
             |     operator name: XXXXXXXXXX
```

図 6.168 SIM の情報を取得する

6.16.5.15. mmcli - 回線情報を取得する

回線情報を取得するには、「図 6.169. 回線情報を取得する」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m $(mm-modem-num)
:（省略）
  Bearer   |               paths: /org/freedesktop/ModemManager1/Bearer/[number] # [number] を次の ⏎
コマンドで使用
:（省略）

[armadillo ~]# mmcli -b [number]
  ------------------------------------
  General            |           path: /org/freedesktop/ModemManager1/Bearer/[bearer number]
                     |           type: default
  ------------------------------------
```

```
Status              |      connected: yes
                    |      suspended: XX
                    |    multiplexed: XX
                    |     ip timeout: XX
------------------------------------
Properties          |            apn: XXXXXXXXXXX
                    |        ip type: XXXXX
```

図 6.169 回線情報を取得する

6.16.6. 無線 LAN

本章では、Armadillo-IoT ゲートウェイ A6E に搭載されている無線 LAN モジュールの使用方法について説明します。

例として、WPA2-PSK(AES)のアクセスポイントに接続します。WPA2-PSK(AES)以外のアクセスポイントへの接続方法などについては、man nm-settings を参考にしてください。また、以降の説明では、アクセスポイントの ESSID を[essid]、パスフレーズを[passphrase]と表記します。

6.16.6.1. 無線 LAN アクセスポイントに接続する

無線 LAN アクセスポイントに接続するためには、次のようにコマンドを実行してコネクションを作成します。

```
[armadillo ~]# nmcli device wifi connect [essid] password [passphrase]
```

図 6.170 無線 LAN アクセスポイントに接続する

作成されたコネクションの ID は nmcli connection コマンドで確認できます。

```
[armadillo ~]# nmcli connection
NAME                UUID                                  TYPE      DEVICE
atmark-4f           e051a1df-6bd7-4bcf-9c71-461af666316d  wifi      wlan0
Wired connection 1  f147b8e8-4a17-312d-a094-8c9403007f6a  ethernet  --
```

図 6.171 無線 LAN のコネクションが作成された状態



NetworkManager の仕様により、無線 LAN の接続にはランダムな MAC アドレスが使用されます。搭載されている無線 LAN モジュール固有の MAC アドレスを使用したい場合は、以下の例のように NetworkManager の設定を変更し、再起動を行ってください。

```
[armadillo ~]# echo "[device-mac-randomization]" > /etc/NetworkManager/
conf.d/no-random-mac.conf
[armadillo ~]# echo "wifi.scan-rand-mac-address=no" >> /etc/
NetworkManager/conf.d/no-random-mac.conf
[armadillo ~]# echo "[connection-mac-randomization]" >> /etc/
NetworkManager/conf.d/no-random-mac.conf
[armadillo ~]# echo "wifi.cloned-mac-address=permanent" >> /etc/
NetworkManager/conf.d/no-random-mac.conf
```

```
[armadillo ~]# persist_file /etc/NetworkManager/conf.d/no-random-
mac.conf
```

6.16.6.2. 無線 LAN の接続を確認する

無線 LAN で正常に通信が可能か確認します。

同じネットワーク内にある通信機器と PING 通信を行います。以下の例では、通信機器が「192.0.2.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping 192.0.2.20
```

図 6.172 無線 LAN の PING 確認



無線 LAN 以外のコネクションが有効化されている場合、ネットワーク通信に無線 LAN が使用されない場合があります。確実に無線 LAN の接続確認をする場合は、事前に無線 LAN 以外のコネクションを無効化してください。

6.16.7. 無線 LAN アクセスポイント (AP) として設定する

WLAN+BT コンボ搭載モデルの無線 LAN をアクセスポイント (以降 AP) として設定する方法を説明します。AP 設定は hostapd というソフトウェアと、 DNS/DHCP サーバである dnsmasq というソフトウェアを使用します。

hostapd と dnsmasq は Armadillo Base OS にデフォルトでインストール済みとなっているため、インストール作業は不要です。インストールされていない場合は、 Armadillo Base OS を最新バージョンに更新してください。



アクセスポイントモード (AP) と ステーションモード (STA) の同時利用はできません。

6.16.7.1. bridge インターフェースを追加する

NetworkManager を使用し bridge インターフェース (br0) を追加します。同時に AP の IP アドレスも設定します。ここでは 192.0.2.1 を設定しています。

```
[armadillo ~]# nmcli con add type bridge ifname br0
[armadillo ~]# nmcli con mod bridge-br0 ipv4.method manual ipv4.address "192.0.2.1/24" bridge.stp
off
```

```
[armadillo ~]# nmcli con up bridge-br0
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/bridge-br0.nmconnection ❶
```

図 6.173 bridge インターフェースを作成する

❶ 設定ファイルを永続化します。

また、NetworkManager のデフォルト状態では定期的に wlan0 のスキャンを行っています。スキャン中は AP の性能が低下するため wlan0 を NetworkManager の管理から外します。

```
[armadillo ~]# vi /etc/NetworkManager/conf.d/90_disable_wlan0.conf
[device_wlan0]
match-device=interface-name:wlan0
managed=0

[armadillo ~]# persist_file /etc/NetworkManager/conf.d/90_disable_wlan0.conf
[armadillo ~]# nmcli d set wlan0 managed no ❶
```

図 6.174 wlan0 インターフェースを NetworkManager の管理から外す

❶ nmcli で NetworkManager をリスタートせずに設定します。

6.16.7.2. hostapd を設定する

hostapd の設定ファイルの雛形として用意してある /etc/hostapd/hostapd.conf.example をコピーして使用します。

```
[armadillo ~]# cp /etc/hostapd/hostapd.conf.example /etc/hostapd/hostapd.conf
[armadillo ~]# vi /etc/hostapd/hostapd.conf
hw_mode=a ❶
channel=44 ❷
ssid=myap ❸
wpa_passphrase=myap_pass ❹
interface=wlan0 ❺
bridge=br0
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
driver=nl80211
country_code=JP
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
disassoc_low_ack=1
preamble=1
wmm_enabled=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
ieee80211ac=1
ieee80211ax=1
ieee80211n=1
```

```
ieee80211d=1
ieee80211h=1
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2

[armadillo ~]# persist_file /etc/hostapd/hostapd.conf ❻
[armadillo ~]# rc-service hostapd start ❼
[armadillo ~]# rc-update add hostapd ❽
[armadillo ~]# persist_file /etc/runlevels/default/hostapd ❾
```

図 6.175 hostapd.conf を編集する

❶ 5GHz であれば a を、2.4GHz であれば g を設定します。

❷ 使用するチャンネルを設定します。

❸ 子機から接続するための任意の SSID を設定します。この例では myap を設定しています。

❹ 子機から接続するための任意のパスフレーズを設定します。この例では myap_pass を設定しています。

❺ Armadillo-IoT ゲートウェイ A6E では interface には wlan0 を設定します。

❻ 設定ファイルを永続化します。

❼ hostapd を起動します。

❽ Armadillo 起動時に hostapd が自動的に起動されるようにします。

❾ hostapd 自動起動の設定を永続化します。



W53 / W56 のチャンネルグループは工事設計認証未取得のため使用しないでください。5GHz 帯アクセスポイント（AP）モードで指定できるチャンネルは W52 に属する 36 / 40 / 44 / 48 の 4 チャンネルのみです。

hostapd.conf で channel=0 または channel=acs_survey を指定することでチャンネル自動選択（ACS）が有効になりますが、この際には acs_exclude_dfs=1 を必ず併記してください。これを省略すると DFS 対象の W53 / W56 が選択候補に含まれ、意図せず電波法に抵触するおそれがあります。

2.4GHz 帯のみを利用する構成では acs_exclude_dfs=1 は必須ではありませんが、後から 5GHz 帯へ切り替える際の設定漏れ防止のため 2.4GHz 帯運用時でも予防的に指定することを推奨します。

5GHz 帯の利用の可否や条件については 「1.2.2. 取扱い上の注意事項」の「電波に関する注意事項(5GHz 帯無線)」をご参照ください。

6.16.7.3. dnsmasq を設定する

dnsmasq の設定ファイルを以下の内容で作成し /etc/dnsmasq.d/ 下に配置します。ファイル名は任意ですが、拡張子は .conf としてください。ここでは dhcp.conf としています。

```
[armadillo ~]# vi /etc/dnsmasq.d/dhcp.conf
interface=br0
bind-dynamic
dhcp-range=192.0.2.10, 192.0.1.2, 24h ❶

[armadillo ~]# persist_file /etc/dnsmasq.d/dhcp.conf ❷
[armadillo ~]# rc-service dnsmasq restart ❸
```

図 6.176 dnsmasq の設定ファイルを編集する

❶ 子機に割り当てる IP アドレスの範囲とリース期間を設定します。

❷ 設定ファイルを永続化します。

❸ dnsmasq を再起動します。

hostapd と dnsmasq の起動完了後、子機から hostapd.conf で設定した SSID とパスフレーズで接続できます。

6.16.8. ファイアウォールの設定方法

開放しているポートが存在すると攻撃者の標的になる可能性があります。開発したサーバーが使用するポートに対して、アクセスできる IP アドレスを制限することでセキュリティ上のリスクを低減することができます。

ここでは、iptables コマンドを使用した、パケットフィルタリングによるアクセス制限方法を紹介します。



デフォルトでは iptables サービスは無効になっています。サービスを有効にするためには以下のコマンドを実行してください。

```
[armadillo ~]# rc-update add iptables ❶
[armadillo ~]# persist_file /etc/runlevels/default/iptables ❷
```

❶ サービスを有効にします。

❷ サービスの有効を永続化します。

「図 6.177. 特定のポートに対する IP アドレスのフィルタリング」の例では、Armadillo の特定のポートに対して、特定の IP アドレスからのアクセスのみを受け入れるようにします。この例では、<送信元 IP アドレス> は Armadillo にパケットを送信する IP アドレス、<ポート番号> はパケットを受け入れる Armadillo のポート番号、<プロトコル> は通信プロトコルのことを指します。また、<ポート番号> はパケットを受け入れる Armadillo のポート番号のことを指します。

```
[armadillo ~]# iptables -I INPUT -s <送信元 IP アドレス> -p <プロトコル> --dport <ポート番号> -j ⏎
ACCEPT ❶
[armadillo ~]# iptables -A INPUT -p <プロトコル> --dport <ポート番号> -j REJECT ❷
[armadillo ~]# iptables -L ❸
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     〈プロトコル〉 --  〈送信元 IP アドレス〉 anywhere            〈プロトコル〉 dpt:〈ポート番号〉
REJECT     〈プロトコル〉 --  anywhere             anywhere             〈プロトコル〉 dpt:〈ポート番号
〉 reject-with icmp-port-unreachable
... 省略

[armadillo ~]# /etc/init.d/iptables save ❹
[armadillo ~]# persist_file /etc/iptables/rules-save ❺
```

図 6.177 特定のポートに対する IP アドレスのフィルタリング

❶ <ポート番号> に <送信元 IP アドレス> から送られてきたパケットを受け入れるように設定します。

❷ <ポート番号> に <送信元 IP アドレス> 以外から送信されてきたパケットを拒否するように設定します。

❸ 想定通りに設定されているか確認します。

❹ 上記の設定を設定ファイル /etc/iptables/rules-save に保存します。

❺ 保存した設定ファイルを永続化します。

「図 6.177. 特定のポートに対する IP アドレスのフィルタリング」はあくまで一例ですが、このように iptables コマンドを用いることで開発したサーバーにアクセスできる IP アドレスを制限することができます。

上記の設定を削除する場合は「図 6.178. 特定のポートに対する IP アドレスのフィルタリングの設定を削除」に示すコマンドを実行してください。

```
[armadillo ~]# iptables -L --line-number ❶
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     〈プロトコル〉 --  〈送信元 IP アドレス〉 anywhere            〈プロトコル〉 dpt:〈ポート番号〉
2    REJECT     〈プロトコル〉 --  anywhere             anywhere             〈プロトコル〉 dpt:〈ポー
ト番号〉 reject-with icmp-port-unreachable
... 省略

[armadillo ~]# iptables -D INPUT 2 ❷
[armadillo ~]# iptables -D INPUT 1 ❸
[armadillo ~]# /etc/init.d/iptables save ❹
[armadillo ~]# persist_file /etc/iptables/rules-save ❺
```

図 6.178 特定のポートに対する IP アドレスのフィルタリングの設定を削除

❶ 削除する設定の番号（num）を確認します。ここでは 1 番と 2 番の設定を削除します。

❷ 2 番の設定を削除します。

❸ 1 番の設定を削除します。

❹ 上記の設定を設定ファイル /etc/iptables/rules-save に保存します。

❺ 保存した設定ファイルを永続化します。

6.17. コマンドラインからストレージを使用する

ここでは、SDHC カードを接続した場合を例にストレージの使用方法を説明します。以降の説明では、共通の操作が可能な場合に、SD/SDHC/SDXC カードを SD カードと表記します。



SDXC/microSDXC カードを使用する場合は、事前に「6.17.1. ストレージのパーティション変更とフォーマット」を参照してフォーマットを行う必要があります。これは、Linux カーネルが exFAT ファイルシステムを扱うことができないためです。通常、購入したばかりの SDXC/microSDXC カードは exFAT ファイルシステムでフォーマットされています。

Linux では、アクセス可能なファイルやディレクトリは、一つの木構造にまとめられています。あるストレージデバイスのファイルシステムを、この木構造に追加することを、マウントするといいます。マウントを行うコマンドは、 mount です。

mount コマンドの典型的なフォーマットは、次の通りです。

```
mount [-t fstype] device dir
```

図 6.179 mount コマンド書式

-t オプションに続く fstype には、ファイルシステムタイプを指定します。ファイルシステムタイプの指定は省略可能です。省略した場合、mount コマンドはファイルシステムタイプを推測します。この推測は必ずしも適切なものとは限りませんので、事前にファイルシステムタイプが分かっている場合は明示的に指定してください。FAT32 ファイルシステムの場合は vfat 、EXT3 ファイルシステムの場合は ext3 を指定します。



通常、購入したばかりの SDHC カードは FAT32 または exFAT ファイルシステムでフォーマットされています。

device には、ストレージデバイスのデバイスファイル名を指定します。microSD カードのパーティション 1 の場合は /dev/mmcblk1p1 、パーティション 2 の場合は /dev/mmcblk1p2 となります。

dir には、ストレージデバイスのファイルシステムをマウントするディレクトリを指定します。

microSD スロット (CON1) に SDHC カードを挿入し、以下に示すコマンドを実行すると、 /media ディレクトリに SDHC カードのファイルシステムをマウントすることができます。microSD カード内のファイルは、/media ディレクトリ以下に見えるようになります。

```
[armadillo ~]# mount -t vfat /dev/mmcblk1p1 /media
[armadillo ~]# ls /media
```

```
   :
   :
```

図 6.180 ストレージのマウント

ストレージを安全に取り外すには、アンマウントという作業が必要です。アンマウントを行うコマンドは、 umount です。オプションとして、アンマウントしたいデバイスがマウントされているディレクトリを指定します。

```
[armadillo ~]# umount /media
```

図 6.181 ストレージのアンマウント

6.17.1. ストレージのパーティション変更とフォーマット

通常、購入したばかりの SDHC カードや USB メモリは、一つのパーティションを持ち、FAT32 ファイルシステムでフォーマットされています。

パーティション構成を変更したい場合、 fdisk コマンドを使用します。 fdisk コマンドの使用例として、一つのパーティションで構成されている microSD カードのパーティションを、2 つに分割する例を「図 6.182. fdisk コマンドによるパーティション変更」に示します。一度、既存のパーティションを削除してから、新たにプライマリパーティションを二つ作成しています。先頭のパーティションには 100MByte、二つめのパーティションに残りの容量を割り当てています。先頭のパーティションは /dev/mmcblk1p1 、二つめは /dev/mmcblk1p2 となります。 fdisk コマンドの詳細な使い方は、man ページ等を参照してください。

```
[armadillo ~]# fdisk /dev/mmcblk1

Welcome to fdisk (util-linux 2.29.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-7744511, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-7744511, default 7744511): +100M

Created a new partition 1 of type 'Linux' and of size 100 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
```

```
Partition number (2-4, default 2): 2
First sector (206848-7744511, default 206848):
Last sector, +sectors or +size{K,M,G,T,P} (206848-7744511, default 7744511):

Created a new partition 2 of type 'Linux' and of size 3.6 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
[  447.905671]  mmcblk1: p1 p2
Syncing disks.
```

図 6.182 fdisk コマンドによるパーティション変更

FAT32 ファイルシステムでストレージデバイスをフォーマットするには、 mkfs.vfat コマンドを使用します。また、EXT2 や EXT3、 EXT4 ファイルシステムでフォーマットするには、mkfs.ext2 や mkfs.ext3 、 mkfs.ext4 コマンドを使用します。microSD カードのパーティション 1 を EXT4 ファイルシステムでフォーマットするコマンド例を、次に示します

```
[armadillo ~]# mkfs.ext4 /dev/mmcblk1p1
```

図 6.183 EXT4 ファイルシステムの構築

6.18. コマンドラインから CPU の測定温度を取得する

Armadillo-IoT ゲートウェイ A6E の温度センサーは、i.MX6ULL の TEMPMON(Temperature Monitor)を利用しています。

起動直後の設定では、i.MX6ULL の測定温度が 100℃以上になった場合、Linux カーネルが /sbin/poweroff コマンドを実行し、システムを停止します。

6.18.1. 温度を取得する

/sys/class/thermal/thermal_zone0/temp ファイルの値を読み出すことによって、i.MX6ULL の測定温度を取得することができます。

```
[armadillo ~]# cat /sys/class/thermal/thermal_zone0/temp
50000 ❶
```

図 6.184 i.MX6ULL の測定温度を取得する

❶ 温度はミリ℃ の単位で表示されます。この例では 50.000℃ を示しています。

6.19. アナログ入力インターフェースの電源制御を行う

アナログ入力インターフェース(CON21)の電源制御が可能です。アナログ入力インタフェースを使用しないタイミングがある場合などに使用ください。



アナログ入力インターフェースに電圧または電流を印加した状態で、アナログ入力インターフェースの電源をオフしないでください。内部回路が故障する可能性があります。

以下に電源オフ、電源オン、再起動のコマンドを示します。

```
[armadillo ~]# ain-power-ctrl off
Powered off the analog input block
```

図 6.185 アナログ入力インターフェースの電源オフ

```
[armadillo ~]# ain-power-ctrl on
Powered on the analog input block
```

図 6.186 アナログ入力インターフェースの電源オン

```
[armadillo ~]# ain-power-ctrl reboot
Powered off the analog input block
Powered on the analog input block
```

図 6.187 アナログ入力インターフェースの再起動

6.20. SMS を利用する

Armadillo-IoT ゲートウェイ A6E は、LTE モジュール を使用した SMS の送受信を行うことができます。SMS の送信、受信した SMS の確認および削除などの操作は ModemManager の mmcli コマンドで行うことができます。

本章では mmcli コマンドでの SMS の使用方法について説明します。

6.20.1. 初期設定

SMS が利用可能な SIM を挿入して Armadillo-IoT ゲートウェイ A6E の電源を入れると、ModemManager が必要な初期設定を行い、 SMS が利用可能になります。

SMS の受信は自動的に行われます。

「図 6.188. 言語設定」に示すコマンドを実行し、言語設定を行います。

```
[armadillo ~]# export LANG="ja_JP.UTF-8"
```

図 6.188 言語設定

6.20.2. SMS を送信する

SMS を作成するには、「図 6.189. SMS の作成」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m $(mm-modem-num) --messaging-create-sms="number=[送信先電話番号],text='[SMS
本文]'"
```

図 6.189 SMS の作成

SMS の作成に成功すると、以下のように SMS 番号が表示されます。SMS 番号は送信時に使用します。

```
Successfully created new SMS: /org/freedesktop/ModemManager1/SMS/[SMS 番号]
```

図 6.190 SMS 番号の確認

「図 6.191. SMS の送信」に示すコマンドを実行し、SMS 送信を行います。 [SMS 番号] には、 SMS の作成時に表示された番号を指定します。

```
[armadillo ~]# mmcli -s [SMS 番号] --send
```

図 6.191 SMS の送信

6.20.3. SMS を受信する

SMS を送信可能な端末から Armadillo-IoT ゲートウェイ A6E に SMS を送信すると、 Armadillo-IoT ゲートウェイ A6E は自動的に SMS を受信します。

また、 LTE モジュールの内蔵ストレージに 10 件 SMS を保存した状態で Armadillo-IoT ゲートウェイ A6E に SMS を送信した場合は、Armadillo-IoT ゲートウェイ A6E は受信を行いません。

受信を行うには、 LTE モジュールの内蔵ストレージに保存している SMS を削除するか、他のストレージに移動する必要があります。

6.20.4. SMS 一覧を表示する

「図 6.192. SMS の一覧表示」のコマンドを実行することで、 SMS 一覧を表示できます。

末尾が "(sent)" となっているものが送信した SMS で "(received)" となっているものが受信した SMS です。

```
[armadillo ~]# mmcli -m $(mm-modem-num) --messaging-list-sms
Found 7 SMS messages:
        /org/freedesktop/ModemManager1/SMS/0 (received)
        /org/freedesktop/ModemManager1/SMS/1 (received)
        /org/freedesktop/ModemManager1/SMS/2 (received)
        /org/freedesktop/ModemManager1/SMS/3 (received)
        /org/freedesktop/ModemManager1/SMS/4 (sent)
        /org/freedesktop/ModemManager1/SMS/5 (received)
        /org/freedesktop/ModemManager1/SMS/6 (sent)
```

図 6.192 SMS の一覧表示

6.20.5. SMS の内容を表示する

SMS の内容を表示するには、「図 6.193. SMS の内容を表示」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号]
  ----------------------------------
  Content    |              number: XXXXXXXXXX
             |                text: hello world
  ----------------------------------
  Properties |            PDU type: deliver
             |               state: received
             |             storage: me
             |                smsc: +XXXXXXXXXXX
             |           timestamp: XXXXXXXXXXX+XX
```

図 6.193 SMS の内容を表示

受信した SMS は自動的に LTE モジュールの内蔵ストレージに保存されます。Armadillo-IoT ゲートウェイ A6E に搭載されている、LTE モジュールには、最大 10 件まで SMS を保存することが可能です。

SMS の内容を表示した際の「storage: **me**」は、 LTE モジュールの内蔵ストレージに SMS が保存されていることを意味しています。

「storage: **sm**」と表示された場合、 SIM カードのストレージに SMS が保存されています。 SIM カードのストレージに保存できる SMS の件数は SIM カードによって異なります。

ストレージに保存されている SMS は、Armadillo-IoT ゲートウェイ A6E の電源を切断してもデータが保持されます。

6.20.6. SMS を削除する

SMS を削除するには、「図 6.194. SMS の削除」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m $(mm-modem-num) --messaging-delete-sms=[SMS 番号]
```

図 6.194 SMS の削除

6.20.7. SMS を他のストレージに移動する

SIM カードのストレージに SMS を移動するには、「図 6.195. SIM カードのストレージに SMS を移動」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号] --store-in-storage="sm"
```

図 6.195 SIM カードのストレージに SMS を移動

LTE モジュールの内蔵ストレージに SMS を移動するには、「図 6.196. LTE モジュールの内蔵ストレージに SMS を移動」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号] --store-in-storage="me"
```

図 6.196 LTE モジュールの内蔵ストレージに SMS を移動

6.21. ボタンやキーを扱う

buttond サービスを使用することで、ボタンやキー入力をトリガーとする処理を簡単に実装できます。

/etc/atmark/buttond.conf に BUTTOND_ARGS を指定することで、動作を指定することができます:

- --short <key> --action "command": 短押しの設定。キーを 1 秒以内に離せば短押しと認識し "command"を実行します。認識する最大時間は --time <time_ms> オプションで変更可能です。
- --long <key> --action "command": 長押しの設定。キーを 5 秒押し続けたタイミングで "command" を実行します。長押しと認識する最低時間は --time <time_ms> オプションで変更可能です。
- 1 つのキーに対して複数の設定が可能です。長押しの設定が複数ある場合、押したままの状態だと一番長い時間に設定されている "command" を実行します。途中でキーを離した場合は、キーを離した時間に応じた "command" を実行します。(例：buttond --short <key> --action "cmd1" --long <key> --time 2000 --action "cmd2" --long <key> --time 10000 --action "cmd3" <file> を実行した場合、1 秒以内に離すと "cmd1"、2 秒以上 10 秒以内に離すと "cmd2"、10 秒を越えたら "cmd3" を実行します)。
  - 短押し設定を複数指定する場合、時間の短い設定を先に指定してください。 0.5 秒、1 秒を設定したい場合、1 秒 → 0.5 秒の順番で指定すると 0.5 秒が無視されます。
- --exit-timeout <time_ms>: 設定した時間の後に buttond を停止します。起動時のみに対応したい場合に使えます。
- キーの設定の --exit-after オプション：キーのコマンドを実行した後に buttond を停止します。キーの対応を一回しか実行しないように使えます。

6.21.1. SW1 の短押しと長押しの対応

以下にデフォルトを維持したままで SW1 の短押しと長押しのそれぞれの場合にコマンドを実行させる例を示します。

```
[armadillo ~]# vi /etc/atmark/buttond.conf ❶
BUTTOND_ARGS="$BUTTOND_ARGS --short prog1 --action 'date >> /tmp/shortpress'"
BUTTOND_ARGS="$BUTTOND_ARGS --long prog1 --time 5000 --action 'date >> /tmp/longpress'"
[armadillo ~]# persist_file /etc/atmark/buttond.conf ❷
[armadillo ~]# rc-service buttond restart ❸
buttond          | * Stopping button watching daemon ...                                [ ok ]
buttond          | * Starting button watching daemon ...                                [ ok ]
[armadillo ~]# cat /tmp/shortpress ❹
Tue Mar 22 17:16:42 JST 2022
Tue Mar 22 17:16:43 JST 2022
[armadillo ~]# cat /tmp/longpress
Tue Mar 22 17:16:48 JST 2022
```

図 6.197 buttond で SW1 を扱う

❶ buttond の設定ファイルを編集します。この例では、短押しの場合 /tmp/shotpress に、5 秒以上の長押しの場合 /tmp/longpress に日付を出力します。

❷ 設定ファイルを保存します。

❸ buttond サービスを再起動させます。ここでは再起動後短押しを 2 回、長押しを 1 回行ったとします。

❹ 押された回数を確認します。

6.21.2. USB キーボードの対応

USB キーボードや他の入力デバイスにも対応できます。

1. デバイスを接続してから、 buttond でデバイス名とキーコードを確認します。

```
[armadillo ~]# buttond -vvv /dev/input/* /dev/input/by-*/* ❶
Skipping directory /dev/input/by-id
Skipping directory /dev/input/by-path
[78972.042] /dev/input/event2 4 4 458976: non-keyboard event ignored
[78972.042] /dev/input/event2 LEFTCTRL (29) pressed: ignored ❷
[78972.042] /dev/input/by-id/usb-0566_3029-event-kbd 4 4 458976: non-keyboard event ignored
[78972.042] /dev/input/by-id/usb-0566_3029-event-kbd LEFTCTRL (29) pressed: ignored
[78972.042] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd 4 4 458976: ⏎
non-keyboard event ignored
[78972.042] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd LEFTCTRL ⏎
(29) pressed: ignored
[78972.130] /dev/input/event2 4 4 458976: non-keyboard event ignored
[78972.130] /dev/input/event2 LEFTCTRL (29) released: ignored
[78972.130] /dev/input/by-id/usb-0566_3029-event-kbd 4 4 458976: non-keyboard event ignored
[78972.130] /dev/input/by-id/usb-0566_3029-event-kbd LEFTCTRL (29) released: ignored
[78972.130] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd 4 4 458976: ⏎
non-keyboard event ignored
[78972.130] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd LEFTCTRL ⏎
(29) released: ignored
```

図 6.198 buttond で USB キーボードのイベントを確認する

❶ buttond を -vvv で冗長出力にして、すべてのデバイスを指定します。

❷ 希望のキーを押すと、LEFTCTRL が三つのパスで認識されました。 一番安定する by-id のパスを控えておきます。

2. USB デバイスを外すこともありますので、-i (inotify) で管理されてる入力デバイスとして追加します。そうしないとデバイスを外したときに buttond が停止します。

```
[armadillo ~]# vi /etc/atmark/buttond.conf
BUTTOND_ARGS="$BUTTOND_ARGS -i /dev/input/by-id/usb-0566_3029-event-kbd"
BUTTOND_ARGS="$BUTTOND_ARGS --short LEFTCTRL --action 'podman_start ⏎
button_pressed_container'"
[armadillo ~]# persist_file /etc/atmark/buttond.conf
[armadillo ~]# rc-service buttond restart
```

図 6.199 buttond で USB キーボードを扱う

6.21.3. Armadillo 起動時にのみボタンに反応する方法

Armadillo 起動時にのみ、例として SW1 の長押しに反応する方法を紹介します。

/etc/local.d/boot_switch.start に稼働期間を指定した buttond を起動させる設定を記載します。

buttond が起動してから 10 秒以内に SW1 を一秒以上長押しすると myapp のコンテナの親プロセスに USR1 信号を送ります（アプリケーション側で信号を受信して、デバッグモードなどに切り替える想定です）。SW1 が Armadillo 起動前に押された場合は、buttond の起動一秒後に実行されます。

```
[armadillo ~]# vi /etc/local.d/boot_switch.start
#!/bin/sh

buttond /dev/input/by-path/platform-gpio-keys-event ¥ ❶
        --exit-timeout 10000 ¥ ❷
        --long PROG1 --time 1000 --exit-after ¥ ❸
                --action "podman exec myapp kill -USR1 1" & ❹
[armadillo ~]# chmod +x /etc/local.d/boot_switch.start
[armadillo ~]# persist_file /etc/local.d/boot_switch.start
```

図 6.200 buttond で SW1 を Armadillo 起動時のみ受け付ける設定例

❶ SW1 の入力を /dev/input/by-path/platform-gpio-keys-event ファイルの PROG1 として認識できます。

❷ buttond 起動後 10 秒経過すると終了します。

❸ SW1 を一度検知した後すぐに終了します。

❹ サービスとして動作させる必要がないため & を付けてバックグラウンド起動します。

6.22. 動作中の Armadillo の温度を測定する

この章では、Armadillo Base OS 搭載製品を組み込んだユーザー製品の熱設計時に役立つ温度プロファイラツールである「atmark-thermal-profiler」について紹介します。

6.22.1. 温度測定の重要性

Armadillo は製品ごとに動作温度範囲が設定されていますが、それらはあくまでも標準筐体に放熱材と共に取り付けて使用した場合の目安であり、実運用時には自作の筐体の使用や放熱の有無などで記載のスペック通りにならない場合があります。また、 Armadillo には CPU または SoC が特定の温度以上になると、自動的にシャットダウンするサーマルシャットダウン機能が搭載されています。そのため、現実的には Armadillo を組み込んだ製品を運用時と同等の環境で動作させつつ、実際に温度を計測して実運用時の CPU 及び SoC 温度がどの程度まで上がるか、サーマルシャットダウンは起こらないかを確かめる必要があります。

Armadillo Base OS 搭載製品では、動作中の Armadillo の各種温度等を取得し CSV 形式で出力する atmark-thermal-profiler を利用することができますので、温度測定に役立てることができます。

6.22.2. atmark-thermal-profiler をインストールする

atmark-thermal-profiler は apk パッケージで公開されていますので、apk add コマンドでインストールすることが可能です。

```
[armadillo ~]# apk upgrade
[armadillo ~]# apk add atmark-thermal-profiler
```

図 6.201 atmark-thermal-profiler をインストールする



atmark-thermal-profiler はデバッグ(開発)用途で温度情報を収集及び解析するツールです。atmark-thermal-profiler は、他の apk パッケージと同様に persist_file -a コマンドで永続的にインストールしておくことが可能ですが、ログの保存のために Armadillo が起動している間 eMMC への書き込みを続けるので、 Armadillo を組み込んだ製品の運用時に動かしたままにしておくことは推奨しません。

atmark-thermal-profiler を永続的にインストールする場合は、運用時には必ず削除してください。

6.22.3. atmark-thermal-profiler を実行・停止する

「図 6.202. atmark-thermal-profiler を実行する」に示すコマンドを実行することで、 atmark-thermal-profiler が動作を開始します。

```
[armadillo ~]# rc-service atmark-thermal-profiler start
```

図 6.202 atmark-thermal-profiler を実行する

「図 6.203. atmark-thermal-profiler を停止する」に示すコマンドを実行することで、 atmark-thermal-profiler が動作を停止します。

```
[armadillo ~]# rc-service atmark-thermal-profiler stop
```

図 6.203 atmark-thermal-profiler を停止する

6.22.4. atmark-thermal-profiler が出力するログファイルを確認する

atmark-thermal-profiler は、インストール直後から自動的に温度や CPU 負荷率、Load Average などの情報を 30 秒に 1 度の周期で集め、/var/log/thermal_profile.csv に追記していきます。

```
[armadillo ~]# head /var/log/thermal_profile.csv
DATE,ONESHOT,CPU_TMEP,SOC_TEMP,LOAD_AVE,CPU_1,CPU_2,CPU_3,CPU_4,CPU_5,USE_1,USE_2,USE_3,USE_4,USE_ ⏎
5
2022-11-30T11:11:05+09:00,0,54,57,0.24,/usr/sbin/rngd -b -p /run/rngd.pid -q -O jitter:buffer_size: ⏎
4133 -O jitter:refill_thresh:4133 -O jitter:thread_count:1,/usr/sbin/chronyd -f /etc/chrony/ ⏎
chrony.conf,[kworker/1:3H-kb],podman network inspect podman,/usr/sbin/NetworkManager -n,22,2,2,0,0,
:（省略）
```

図 6.204 ログファイルの内容例

thermal_profile.csv の 1 行目はヘッダ行です。各列についての説明を「表 6.36. thermal_profile.csv の各列の説明」に記載します。

表 6.36 thermal_profile.csv の各列の説明

ヘッダ	説明
DATE	その行のデータ取得日時です。 "年-月-日 T 時:分:秒+タイムゾーン" の形式で出力されます。
ONESHOT	この列が 1 の行のデータは、サーマルシャットダウンを含むシャットダウンが実行された時に取得されたことを示します。
CPU_TEMP	計測時点の CPU 温度を示します。単位は℃です。
SOC_TEMP	計測時点の SoC 温度を示します。単位は℃です。製品よっては非対応で、その場合は空白になります。
LOAD_AVE	計測時点から直近 1 分間の Load Average です。
CPU_1	計測時点の CPU 使用率 1 位のプロセスです。
CPU_2	計測時点の CPU 使用率 2 位のプロセスです。
CPU_3	計測時点の CPU 使用率 3 位のプロセスです。
CPU_4	計測時点の CPU 使用率 4 位のプロセスです。
CPU_5	計測時点の CPU 使用率 5 位のプロセスです。
USE_1	計測時点の CPU 使用率 1 位のプロセスの CPU 使用率です。
USE_2	計測時点の CPU 使用率 2 位のプロセスの CPU 使用率です。
USE_3	計測時点の CPU 使用率 3 位のプロセスの CPU 使用率です。
USE_4	計測時点の CPU 使用率 4 位のプロセスの CPU 使用率です。
USE_5	計測時点の CPU 使用率 5 位のプロセスの CPU 使用率です。

6.22.5. 温度測定結果の分析

atmark-thermal-profiler を使用して得られたログファイルの内容を分析してみます。

6.22.5.1. サーマルシャットダウン温度の確認

予め、使用している Armadillo が何℃でサーマルシャットダウンするか確認しておきます。ここでは、 Armadillo Base OS を搭載している Armadillo-IoT ゲートウェイ G4 を例とします。他の製品では得られる結果が異なる場合があることに注意してください。

```
[armadillo ~]# cat /sys/class/thermal/thermal_zone0/trip_point_1_temp
105000 ❶
[armadillo ~]# cat /sys/class/thermal/thermal_zone1/trip_point_1_temp
105000 ❷
```

図 6.205 サーマルシャットダウン温度の確認(Armadillo-IoT ゲートウェイ G4 を例に)

❶ CPU のサーマルシャットダウン温度です。ミリ℃で表記されているので、105℃でサーマルシャットダウンすることがわかります。

❷ SoC のサーマルシャットダウン温度です。ミリ℃で表記されているので、105℃でサーマルシャットダウンすることがわかります。

6.22.5.2. 温度測定結果のグラフ化

atmark-thermal-profiler が出力するログ(thermal_profile.csv)は CSV ファイルなので、各種表計算ソフトでインポートしてグラフ化することが可能です。これにより Armadillo 動作中の温度の変化が可視化され、得られる情報が見やすくなります。

「図 6.206. Armadillo-IoT ゲートウェイ G4 で取得した温度のグラフ」は Armadillo-IoT ゲートウェイ G4 上で一定期間 atmark-thermal-profiler を実行して取得した thermal_profile.csv を Google ス

プレッドシートでグラフ化したものです。例のために、途中で stress-ng コマンドを実行して CPU に負荷を与えた後、 stress-ng コマンドを停止して CPU と SoC の温度が下がるのを待った際のデータです。

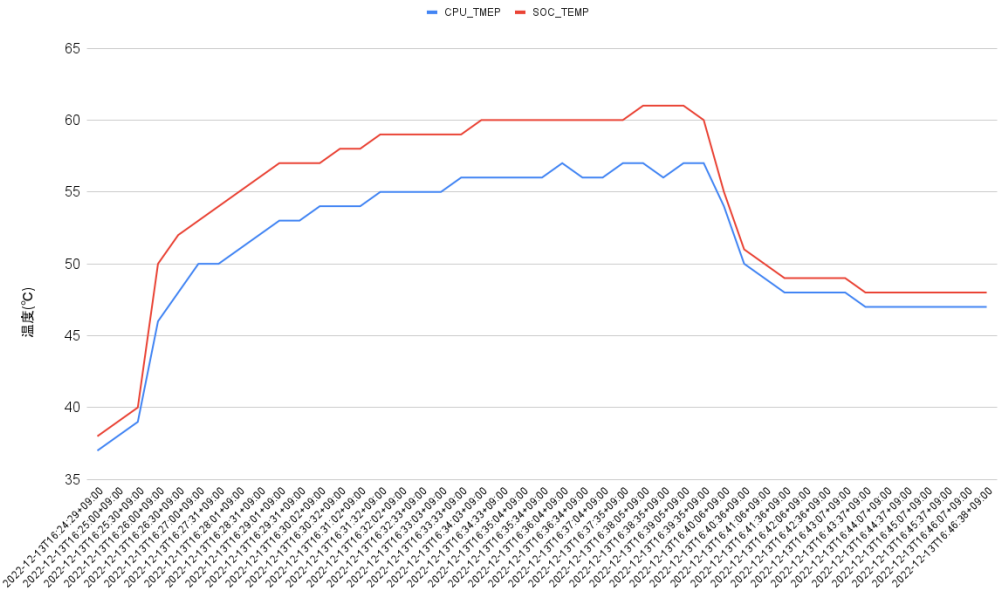


図 6.206 Armadillo-IoT ゲートウェイ G4 で取得した温度のグラフ

グラフの縦軸は温度(℃)で、横軸は時間です。青い線は CPU の温度、赤い線は SoC の温度を表しています。このグラフと、「6.22.5.1. サーマルシャットダウン温度の確認」で得たサーマルシャットダウン温度を見比べると、 CPU に負荷をかけた際であっても SoC の温度は 60℃ 前後ほどまでしか上がらず、この条件で動く Armadillo が温度的にどれほど余裕を持っているかをひと目で確認できます。

6.22.5.3. CPU 使用率の確認

atmark-thermal-profiler は、時間毎の温度だけでなく CPU 使用率と CPU 使用率の高いプロセスについても取得して記録します。CPU 使用率については thermal_profile.csv の CPU_1～CPU_5 列と、USE_1～USE_5 列を参照してください。各列について詳しくは「表 6.36. thermal_profile.csv の各列の説明」にまとまっています。

一般的に CPU 使用率が高くなると、 CPU 周辺の温度も高くなります。そのため、測定した温度が高い場合は、 CPU 使用率の高いプロセスに注目して、 CPU を無駄に使用している意図しない処理が行なわれていないかなどを確認することをおすすめします。

6.22.6. Armadillo Twin から Armadillo の温度を確認する

atmark-thermal-profiler の他に、Armadillo Twin からも温度や CPU 負荷率等の情報を確認することができます。詳細は Armadillo Twin ユーザーマニュアル 「デバイス監視アラートを管理する」[https://manual.armadillo-twin.com/management-device-monitoring-alert/] をご確認ください。

6.23. 電源を安全に切るタイミングを通知する

Armadillo Base OS には、シャットダウン中に電源を切っても安全なタイミングで通知する機能があります。通知は GPIO 出力を用いて行います。どの GPIO 出力ピンを使うのかを Device Tree で設定します。Device Tree で通知用に設定された GPIO 出力ピンの出力レベルを変化させる動作は、シャット

ダウン中に実行される signal_indicator が行います。通知用に設定した GPIO 出力ピンに割り当てた名前を、signal_indicator の設定ファイルに記述することにより、電源を切っても安全なタイミングで、その GPIO 出力ピンの出力が変化します。

以下、signal_indicator の設定手順と、通知用の Device Tree の設定手順を順に述べます。Device Tree の設定は、DTS overlays を使用して行います。



以下の手順は Armadillo Base OS v3.17.2-at.4 以降に存在します /boot/armadillo-iotg-a6e-stdwn-ind-do1.dtbo または /boot/armadillo-iotg-a6e-stdwn-ind-con8-pin7.dtbo を使用します。v3.17.2-at.4 より前のバージョンをご利用の場合はアップデートしてのご利用をお願いします。

6.23.1. signal_indicator の設定

signal_indicator を設定するには、/etc/conf.d ディレクトリに indicator_signals という名前で、次の内容のテキストファイルを作成してください。

```
stdwn_led=STDWN_IND
```

図 6.207 /etc/conf.d/indicator_signals の記述内容

「図 6.208. /etc/conf.d/indicator_signals の永続化」 に示すコマンドで /etc/conf.d/indicator_signals の追加を永続化します。

```
[armadillo ~]# persist_file -vp /etc/conf.d/indicator_signals
```

図 6.208 /etc/conf.d/indicator_signals の永続化

6.23.2. DTS overlays の設定

以下の 「6.23.2.1. CON6(入出力インターフェース)の接点出力 1 を使用する」、「6.23.2.2. CON8(拡張インターフェース)の 7 番ピンを使用する」 どちらかの設定を行ってください。これ以外のピンを使用する場合はデバイスツリーを記載してビルドする必要があります。

6.23.2.1. CON6(入出力インターフェース)の接点出力 1 を使用する



CON6(入出力インターフェース)の接点出力ピンは、ゲートウェイコンテナアプリケーションも使用しており、接点出力 1 を本機能に割り当てると、ゲートウェイコンテナアプリケーションでは、そのピンを使用できなくなります。本機能に接点出力 1 を割り当てる場合は、ゲートウェイコンテナアプリケーションで接点出力 1 を使わないように設定してください。

接点出力に対するゲートウェイコンテナアプリケーションの設定は、「6.10.7. ゲートウェイコンテナの設定ファイル」 を参照してください。ゲートウェイコンテナアプリケーションで接点出力 1 を使用する場合、本機能には CON8(拡張インターフェース)のピンを割り当ててください。

「6.29.4. DTS overlays によるカスタマイズ」 を参考にして /boot/overlays.txt に armadillo-iotg-a6e-stdwn-ind-do1.dtbo を追加してください。

6.23.2.2. CON8(拡張インターフェース)の 7 番ピンを使用する

「6.29.4. DTS overlays によるカスタマイズ」 を参考にして /boot/overlays.txt に armadillo-iotg-a6e-stdwn-ind-con8-pin7.dtbo を追加してください。

6.23.3. 動作確認

ここまで述べた設定を行うと、シャットダウン動作中に通知が行われるようになります。シャットダウン動作の最後の方で、以下のメッセージを出力するのと同じタイミングで通知を行います。つまり、通知用に割り当てた GPIO 出力ピンの出力レベルが、0/Low から 1/High に変わります。シャットダウンが完了して SoC (CPU) への給電がオフすると、出力レベルが 0/Low に戻ります。出力レベルが 0/Low から 1/High に変化した時点以降であれば、Armadillo-IoT A6E の電源を切っても安全です。

```
indicator_signals        | * Signaling external devices we are shutting down ...
```

図 6.209 indicator_signals のコンソール出力

6.24. Armadillo Base OS をアップデートする

Armadillo Base OS は SWUpdate によってアップデートすることができます。

アップデートする際には、rootfs ファイルシステムにインストールされたファイルをすべて消して、アップデートの中身と /etc/swupdate_preserve_files に記載されているファイルで新しい rootfs を作ります。「6.5. swupdate_preserve_files について」 を参照してください。

アップデートでファイルを削除してしまった場合に abos-ctrl mount-old で前のシステムを read-only でマウントして、削除されたファイルをコピーすることもできます。

6.25. ロールバック状態を確認する

Armadillo Base OS のルートファイルシステムが破損し起動できなくなった場合、自動的に以前のバージョンで再起動します。

abos-ctrl status コマンドでロールバックされてるかどうかを確認できます。

```
[armadillo ~]# abos-ctrl status
Currently booted on /dev/mmcblk0p1
Last update on Fri Jun  7 16:03:37 JST 2024, updated: ❶
  boot: 2020.4-at23-00001-g01508f65b8 -> 2020.4-at23
  base_os: 3.19.1-at.3.20240523.pc.gtr -> 3.19.1-at.4
rollback-status: OK: available, no auto-rollback ❷
```

図 6.210 abos-ctrl status の例

❶ 最新のアップデートの日付と内容が出力されています。

❷ 「表 6.37. rollback-status の出力と意味」「表 6.38. rollback-status 追加情報の出力と意味」 に示す状態と追加情報が出力されています。

表 6.37 rollback-status の出力と意味

出力	説明
OK	ロールバックされていません。
rolled back	ロールバックされています。

表 6.38 rollback-status 追加情報の出力と意味

出力	説明
no fallback (fresh install)	初期化状態。
no fallback	何かの理由で B 面が起動できない状態になっています（アップデート失敗後等）。
auto-rollback enabled (post-update)	アップデート直後でまだ再起動していない状態です。再起動して失敗した場合にロールバックが発生します。
auto-rollback enabled (cloned)	abos-ctrl rollback-clone コマンドを実行した後の状態です。同じくロールバック可能です。
available, no auto-rollback	アップデートの後に正常に起動できたので、自動ロールバックが無効になっていますが abos-ctrl rollback --allow-downgrade コマンドで手動ロールバック可能です。



Armadillo Base OS 3.19.1-at.4 以下のバージョンでは「アップデート直後」の概念がなかったため、ステータスは「no fallback」（B 面がない状態）、「optimal」（ロールバック可能）、と「rolled back」の 3 択だけでした。

必要な場合（例えば、自分のアプリケーションがアップデート直後に問題があった場合）、 abos-ctrl rollback で手動のロールバックも可能です。ロールバックにエラーがなければ、再起動してロールバックを完了します。

なお、/var/at-log/atlog に切り替えの際に必ずログを書きますので、調査の時に使ってください。以下の例では、Armadillo Base OS を更新した後に起動できないカーネルをインストールして、起動できなかったためにロールバックされました。

```
[armadillo ~]# cat /var/at-log/atlog
Jun  7 16:03:37 armadillo NOTICE swupdate: Installed update to /dev/mmcblk0p2: ¥
boot: 2020.4-at22 -> 2020.4-at23, base_os: 3.19.1-at.3 -> 3.19.1-at.4
Jun  7 16:11:39 armadillo NOTICE swupdate: Installed update to /dev/mmcblk0p1: ¥
extra_os.kernel: unset -> 5.10.218-1
Jun  7 16:12:18 armadillo WARNING uboot: reset by wdt
Jun  7 16:12:42 armadillo WARNING uboot: reset by wdt
Jun  7 16:13:06 armadillo WARNING uboot: reset by wdt
Jun  7 16:13:09 armadillo WARNING uboot: Counted 3 consecutive unfinished boots
Jun  7 16:13:09 armadillo WARNING uboot: Rolling back to mmcblk0p2
```

図 6.211 /var/at-log/atlog の内容の例

6.26. Armadillo 起動時にコンテナの外でスクリプトを実行する

起動時に何かスクリプトを走らせるためにはコンテナとして実行することを推奨します。 「6.9.4. コンテナ起動設定ファイルを作成する」 を参照してください。

コンテナで実行不可能な場合に、「local」サービスを使うことができます: /etc/local.d ディレクトリに.start ファイルを置いておくと起動時に実行されて、 .stop ファイルは終了時に実行されます。

```
[armadillo ~]# vi /etc/local.d/date_test.start ❶
#!/bin/sh

date > /tmp/boottest
[armadillo ~]# chmod +x /etc/local.d/date_test.start ❷
[armadillo ~]# persist_file /etc/local.d/date_test.start ❸
[armadillo ~]# reboot
:（省略）
[armadillo ~]# cat /tmp/boottest ❹
Tue Mar 22 16:36:12 JST 2022
```

図 6.212 local サービスの実行例

❶ スクリプトを作ります。

❷ スクリプトを実行可能にします。

❸ スクリプトを保存して、再起動します。

❹ 実行されたことを確認します。

6.27. u-boot の環境変数の設定

u-boot の環境変数を変更するには /boot/uboot_env.d/ ディレクトリに環境変数が書かれた設定ファイルを配置します。

ファイルの構文は fw_setenv が扱うことができるもので、以下のとおりです：

- # で始まる行はコメントと扱われる為、無視されます。また、 環境変数への代入を示す = がない場合も無視されます。
- [変数]=[値] で変数を設定します。スペースや引用符を含め他の文字は有効ですので、変数の名前と値に不要な文字を入れないように注意してください。
- [変数]= で変数を消します。値がない場合に変数が消去されます。

このファイルによるアップデート内容は swupdate でアップデートする際に適用されます。

実行中のシステムに影響がありませんので、設定ファイルを swupdate で転送しない場合はファイル永続化後に fw_setenv -s /boot/uboot_env.d/[ファイル名] で変数を書き込んでください。

swupdate でファイルを転送した場合には、変数はすぐに利用されます。

```
[armadillo ~]# vi /boot/uboot_env.d/no_prompt ❶
# bootdelay を -2 に設定することで u-boot のプロンプトを無効化します
bootdelay=-2
[armadillo ~]# persist_file -v /boot/uboot_env.d/no_prompt ❷
'/boot/uboot_env.d/no_prompt' -> '/mnt/boot/uboot_env.d/no_prompt'
[armadillo ~]# fw_setenv -s /boot/uboot_env.d/no_prompt ❸
Environment OK, copy 0
```

```
[armadillo ~]# fw_printenv | grep bootdelay ❹
bootdelay=-2
```

図 6.213 uboot_env.d のコンフィグファイルの例

❶ コンフィグファイルを生成します。

❷ ファイルを永続化します。

❸ 変数を書き込みます。

❹ 書き込んだ変数を確認します。



mkswu バージョン 4.4 以降が必要です。必要な場合はアップデートしてください。

```
[ATDE ~]$ sudo apt update && sudo apt upgrade
```

書き方は、 /usr/share/mkswu/examples/uboot_env.desc を参考にしてください。



「6.30.1. ブートローダーをビルドする」 の際に u-boot のデフォルトを変更した場合や、u-boot のプロンプトで「setenv」や「saveenv」を実行しても、 /boot/uboot_env.d/00_defaults によって変更がアップデートの際にリセットされます。

00_defaults のファイルは Base OS の一部で更新されることもありますので、変更を望む場合は別のファイルを作って設定してください。ファイルはアルファベット順で処理されます。 00_defaults にある変数を後のファイルにも設定した場合はそのファイルの値だけが残ります。

主要な u-boot の環境変数を以下に示します。

表 6.39 u-boot の主要な環境変数

環境変数	説明	デフォルト値
console	コンソールのデバイスノードと、UART のボーレート等を指定します。	ttymxc2,115200
bootcount	起動回数を示します。初回起動時に 1 となり、起動に失敗する度にインクリメントされます。ユーザーランドの reset_bootcount サービスが起動されると、この値はクリアされます。この値が"bootlimit"を越えた場合はロールバックします。ロールバックの詳細については、「3.3.4.4. ロールバック(リカバリー)」を参照してください。	1
bootlimit	"bootcount"のロールバックを行うしきい値を指定します。	3
upgrade_available	1 以上の場合は bootcount を管理してロールバック可能になります。0 か空の場合はロールバックできません。値を abos-ctrl status で確認できます。	状況による

環境変数	説明	デフォルト値
bootdelay	保守モードに遷移するためのキー入力を待つ時間を指定します(単位:秒)。次の値は特別な意味を持ちます。 ・ -1: キー入力の有無に関らず保守モードに遷移します。 ・ -2: キー入力の有無に関らず保守モードに遷移しません。 ・ -3: キー入力の有無に関らず保守モードに遷移しません。ただし、SW1 が押下されている場合は保守モードに遷移します。	0
image	Linux カーネルイメージファイルのパスです。"mmcdev"で指定されたデバイスの、"mmcpart"で指定されたパーティションのルートディレクトリからの相対パスで指定します。	boot/uImage
fdt_file	DTB ファイルのパスです。"mmcdev"で指定されたデバイスの、"mmcpart"で指定されたパーティションのルートディレクトリからの相対パスで指定します。	boot/armadillo.dtb
overlays_list	DT overlay の設定ファイルのパスです。"mmcdev"で指定されたデバイスの、"mmcpart"で指定されたパーティションのルートディレクトリからの相対パスで指定します。DT overlay の詳細については、「6.29.4. DTS overlays によるカスタマイズ」を参照してください。	boot/overlays.txt
mmcautodetect	mmc デバイスの自動検出機能の有効/無効を指定します。yes を指定した場合のみ、u-boot が起動された mmc デバイスが自動的に mmcdev として利用されます。	yes
mmcdev	"image"や"fdt_file"で指定されたファイルが配置してある mmc デバイスのインデックスを指定します。インデックスと mmc デバイスの対応は次の通りです。 ・ 0: eMMC ・ 1: microSD/microSDHC/microSDXC カード "mmcautodetect"に yes が指定されている場合は、u-boot の起動時に上書きされます。	0
mmcpart	"image"や"fdt_file"で指定されたファイルが配置してある、"mmcdev"で指定された mmc デバイスのパーティション番号を指定します。"mmcautodetect"に yes が指定されている場合は、u-boot の起動時に上書きされます。	1
mmcroot	ルートファイルシステムが配置されているデバイスノードと、マウントオプションを指定します。"mmcautodetect"に yes が指定されている場合は、u-boot の起動時に上書きされます。overlayfs が正しく機能しなくなる場合があるので、ro の指定は変更しないでください。	/dev/mmcblk0p1 rootwait ro
optargs	Linux カーネル起動時パラメータを指定します。"quiet"を削除すると、コンソールに起動ログが出力されるようになりますが、起動時間が長くなります。"usbcore.authorized_default=0" は USB 接続制御機能で使用される値です。 この値は USB 接続制御機能の設定によって変化します。この値は直接編集しないでください。 USB 接続制御機能については「3.10. USB デバイスの接続を許可する」を参照してください。	quiet usbcore.authorized_default=0
loadaddr	Linux カーネルが RAM にロードされる物理アドレスを指定します。	0x80800000
fdt_addr	DTB が RAM にロードされる物理アドレスを指定します。	0x83500000
overlay_addr	DT overlay のワーク領域として利用される RAM の物理アドレスを指定します。	0x83520000

6.28. SD ブートの活用

本章では、microSD カードから直接起動(以降「SD ブート」と表記します)する手順を示します。SD ブートを活用すると、microSD カードを取り替えることでシステムイメージを変更することができます。本章に示す手順を実行するためには、容量が 8Gbyte 以上の microSD カードを必要とします。



SD ブートを行った場合、ブートローダーの設定は **microSD カード** に保存されます。



WLAN 搭載モデルでは、SD コントローラ(uSDHC2) を WLAN/BT コンボモジュールが使用するため SD ブートができません。

6.28.1. ブートディスクの作成

1. ブートディスクイメージをビルドします

   「6.30.3. Alpine Linux ルートファイルシステムをビルドする」 で説明されているソースツリー alpine/build-rootfs にあるスクリプト `build_image` と 「6.30.1. ブートローダーをビルドする」でビルドした `u-boot-dtb.imx` を利用します。

```
[ATDE ~/build-rootfs-[VERSION]]$ ./build_image.sh --board a6e ¥
          --boot ~/u-boot-[VERSION]/u-boot-dtb.imx
:（省略）
[ATDE ~/build-rootfs-[VERSION]]$ ls baseos-6e*img
baseos-6e-[VERSION].img
```

1. ブートディスクイメージの書き込み

   ブートディスクイメージの書き込みは「3.1.4.1. 初期化インストールディスクの作成」を参照してください。参照先では初期化インストールディスクの場合の手順を示していますが、ここでビルドしたイメージについても同じ手順になります。



microSD カードのパーティション構成は次のようになっています。

表 6.40 microSD カードのパーティション構成

パーティション	オフセット	サイズ	説明
-	0	10MiB	ブートローダー
1	10MiB	300MiB	A/B アップデートの A 面パーティション
2	310MiB	300MiB	A/B アップデートの B 面パーティション
3	610MiB	50MiB	ログ用パーティション
4	660MiB	200MiB	ファームウェア
5	860MiB	残り	アプリケーション用パーティション

gdisk で確認すると次のようになります。

```
[ATDE ~]$ sudo gdisk -l /dev/sdb
GPT fdisk (gdisk) version 1.0.6

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 60506112 sectors, 28.9 GiB
Model:
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 44B816AC-8E38-4B71-8A96-308F503238E3
Partition table holds up to 128 entries
Main partition table begins at sector 20448 and ends at sector 20479
First usable sector is 20480, last usable sector is 60485632
Partitions will be aligned on 2048-sector boundaries
Total free space is 0 sectors (0 bytes)

Number  Start (sector)    End (sector)  Size       Code  Name
   1           20480          634879   300.0 MiB   8300  rootfs_0
   2          634880         1249279   300.0 MiB   8300  rootfs_1
   3         1249280         1351679   50.0 MiB    8300  logs
   4         1351680         1761279   200.0 MiB   8300  firm
   5         1761280        60485632   28.0 GiB    8300  app
```

6.28.2. SD ブートの実行

「6.28.1. ブートディスクの作成」で作成したブートディスクから起動する方法を説明します。

1. Armadillo-IoT ゲートウェイ A6E に電源を投入する前に、ブートディスクを CON1(SD インターフェース)に挿入します。また、SW2 を 起動デバイスは microSD 側設定します。SW2 に関しては、「図 3.180. スイッチの状態と起動デバイス」 を参照ください。

2. 電源を投入します。

```
U-Boot 2020.04 (Oct 25 2022 - 10:37:29 +0900)

CPU:   i.MX6ULL rev1.1 at 396 MHz
Model: Atmark Techno Armadillo-IoT Gateway A6E Board
DRAM:  512 MiB
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... *** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Saving Environment to MMC... Writing to redundant MMC(1)... OK
switch to partitions #0, OK
mmc1 is current device
flash target is MMC:1
Net:
```

```
Warning: ethernet@2188000 using MAC address from ROM
eth0: ethernet@2188000
Fastboot: Normal
Normal Boot
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1 is current device
11660400 bytes read in 524 ms (21.2 MiB/s)
Booting from mmc ...
38603 bytes read in 22 ms (1.7 MiB/s)
Loading fdt boot/armadillo.dtb
## Booting kernel from Legacy Image at 80800000 ...

...中略...

Welcome to Alpine Linux 3.16
Kernel 5.10.149-1-at on an armv7l (/dev/ttymxc2)

armadillo login:
```

6.29. Device Tree をカスタマイズする

拡張基板を追加するなど、拡張インターフェース(CON8)のピンを使用する場合、ATDE 上のアプリケーション at-dtweb を利用して Device Tree をカスタイマイズすることが可能です。

at-dtweb は、 Web ブラウザ上のマウス操作で dtbo ファイルおよび desc ファイルを生成することができます。

カスタマイズの対象は拡張インターフェース(CON8)です。



拡張インターフェース(CON8) の 12、13 ピン は I2C4 として予約しています。

追加で I2C4 にデバイスを接続する場合は、+Di8+Ai4 拡張基板を検出するために使用している I2C アドレス 0x53、0x55、0x56、0x57 を使用しないように設計してください。

6.29.1. at-dtweb の更新

「図 3.3. ソフトウェアをアップデートする」 のコマンドを実行し最新版への更新を行ってください。

6.29.2. at-dtweb の起動

1. at-dtweb の起動開始

   at-dtweb の起動を開始するには、デスクトップ左上のアプリケーションの「システムツール」から「at-dtweb」を選択してください。

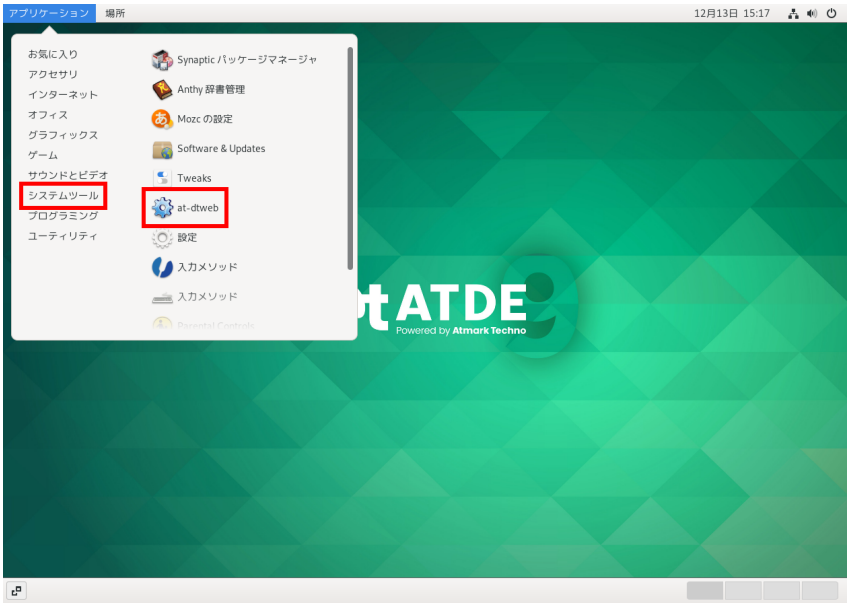


図 6.214 at-dtweb の起動開始

コマンドライン上からでも、at-dtweb コマンドで起動できます。

```
[ATDE ~]$ at-dtweb
```

1. ボードの選択

   ボードを選択します。Armadillo-IoT_A6E を選択して、「OK」をクリックします。

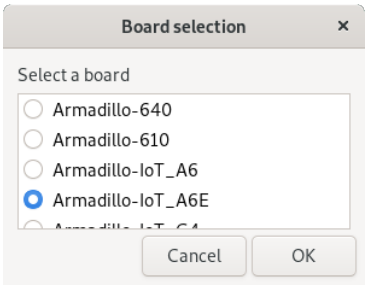


図 6.215 ボード選択画面

2. Linux カーネルディレクトリの選択

   Linux カーネルディレクトリを選択します。コンフィギュレーション済みの Linux カーネルディレクトリを選択して、「OK」をクリックします。

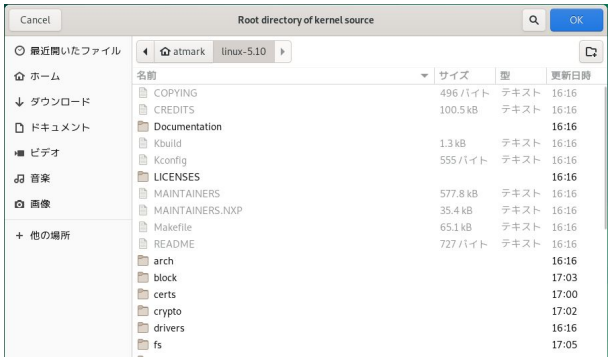


図 6.216 Linux カーネルディレクトリ選択画面

3. at-dtweb の起動完了

   at-dtweb が起動し、次のように画面が表示されます。

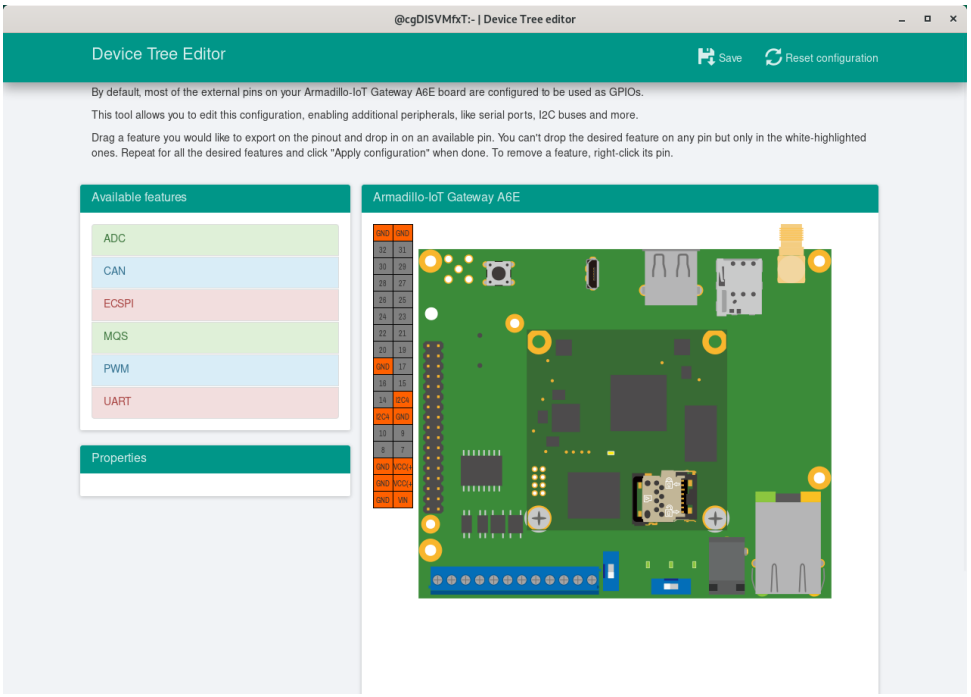


図 6.217 at-dtweb 起動画面



Linux カーネルは、事前にコンフィギュレーションされている必要があります。コンフィギュレーションの手順については「6.30. Armadillo のソフトウェアをビルドする」を参照してください。

6.29.3. Device Tree をカスタマイズ

6.29.3.1. 機能の選択

機能の選択は、ドラッグ&ドロップで行います。画面左上の「Available features」から有効にしたい機能をドラッグし、画面右側の「Armadillo-IoT Gateway A6E」の白色に変化したピンにドロップします。例として CON8 8/9 ピンを UART1(RXD/TXD) に設定します。



何も機能が選択されていないピンには GPIO の機能が割り当てられます。GPIO の機能が自動的に割り当てられるのを防ぐには、「Unset GPIO」を対象のピンに割り当ててください。

at-dtweb で作成した dtbo と 独自にカスタマイズして作成した dtbo (*-customize.dtbo など）を Armadillo に両方適用する場合、at-dtweb で自動的に GPIO に設定された pinctrl と、カスタマイズにより意図して設定した pinctrl が競合する可能性があります。

このような競合を防ぐためには、独自にカスタマイズしたデバイスツリーで使用したピンに対して、at-dtweb で「Unset GPIO」を割り当ててください。

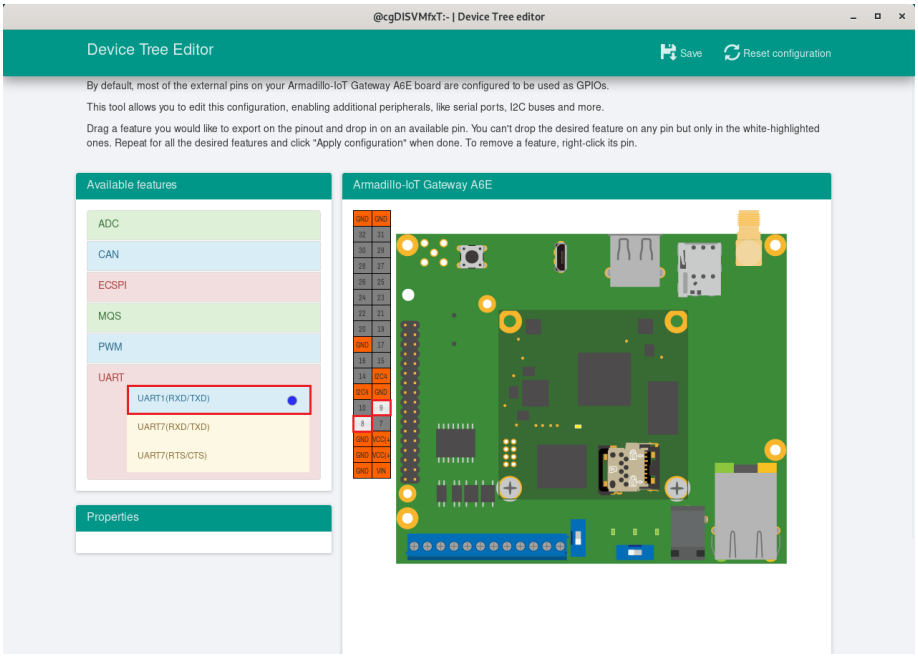


図 6.218 UART1(RXD/TXD) のドラッグ

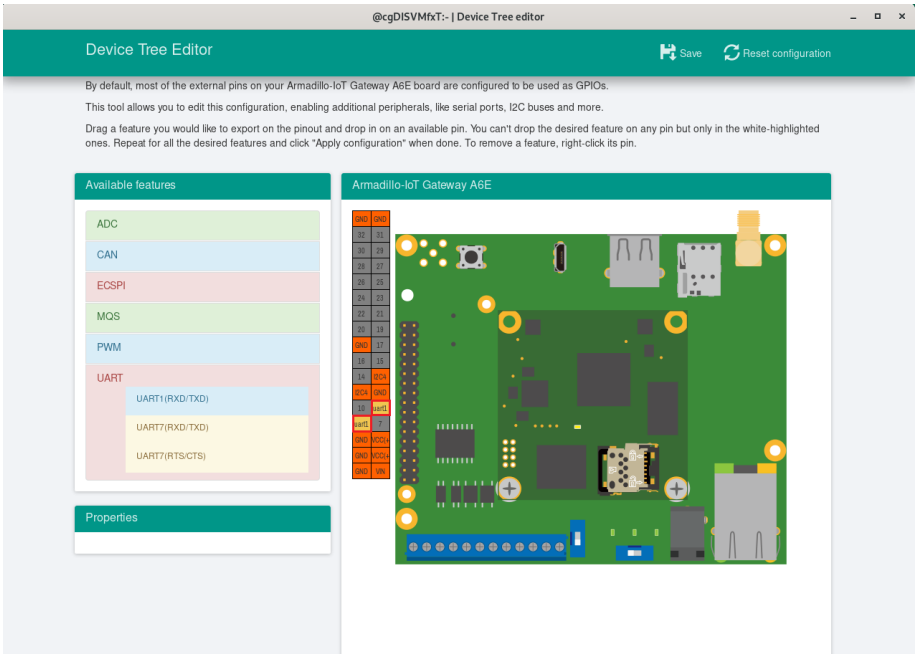


図 6.219 CON8 8/9 ピンへのドロップ

6.29.3.2. 信号名の確認

画面右側の「Armadillo-IoT Gateway A6E」にドロップして設定したピンを左クリックすると信号名が表示されます。どのピンがどの信号に対応しているのかを確認することができます。

例として UART1(RXD/TXD) の信号名を確認します。

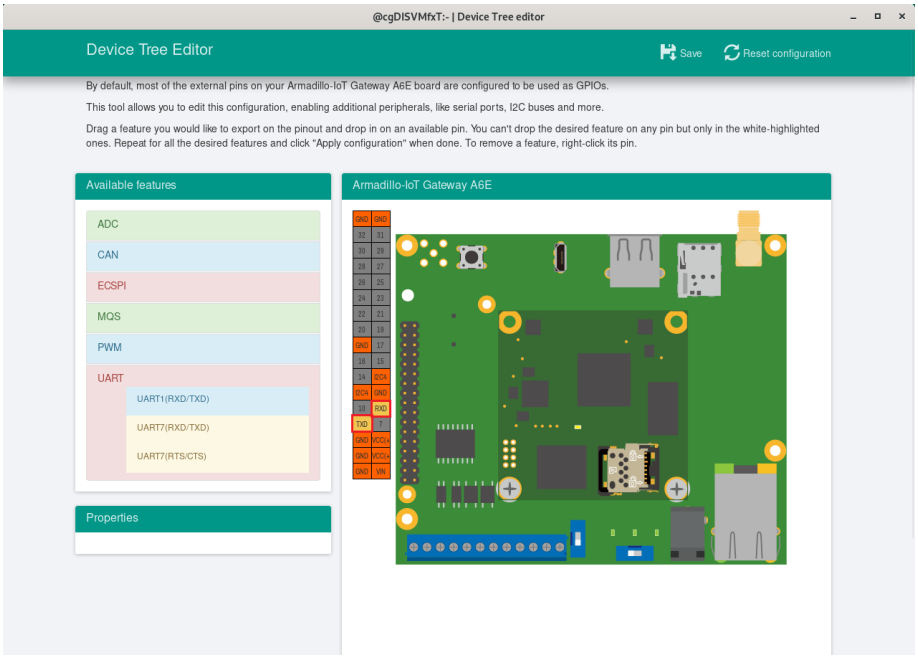


図 6.220 信号名の確認



再度ピンを左クリックすると機能名の表示に戻ります。

6.29.3.3. プロパティの設定

いくつかの機能にプロパティを設定することができます。画面右側の「Armadillo-IoT Gateway A6E」に選択した機能を左クリックすると、画面左下の「Properties」からプロパティを選択することができます。

例として CON8 29-32 ピンの ECSPI1 の spi-max-frequency プロパティを設定します。

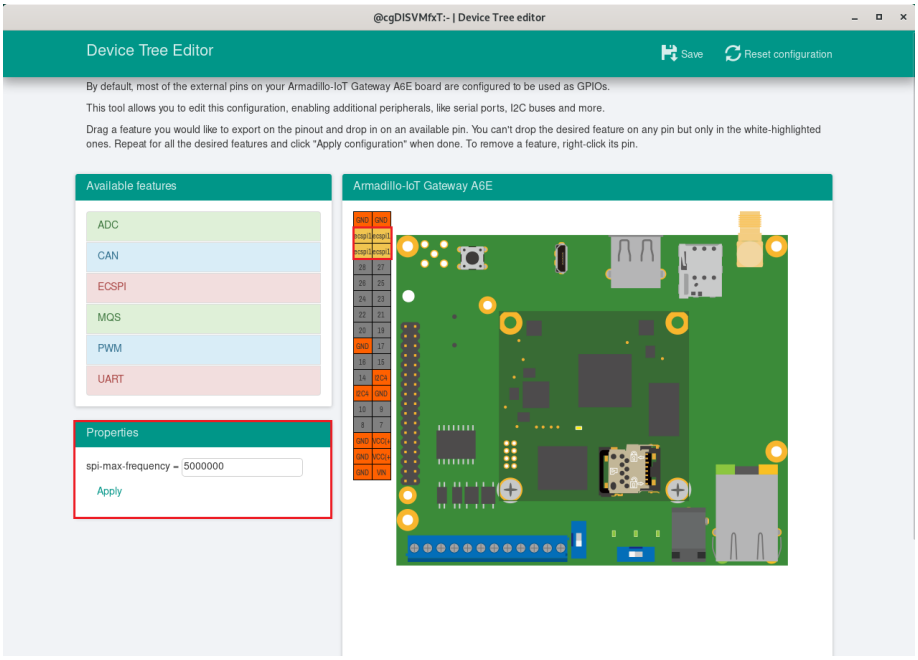


図 6.221 プロパティの設定

設定したプロパティを確定させるには「Apply」をクリックします。

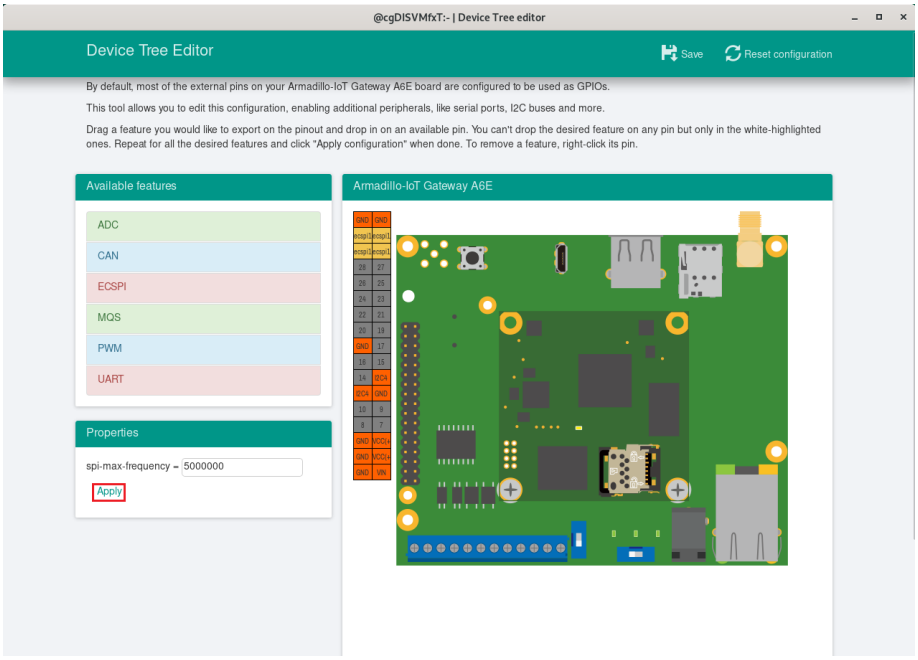


図 6.222 プロパティの保存

6.29.3.4. 機能の削除

全ての機能を削除する場合は、画面右上の「Reset configuration」をクリックします。機能ごとに削除する場合は、画面右側の「Armadillo-IoT Gateway A6E」のピンを右クリックして「Remove」をクリックします。

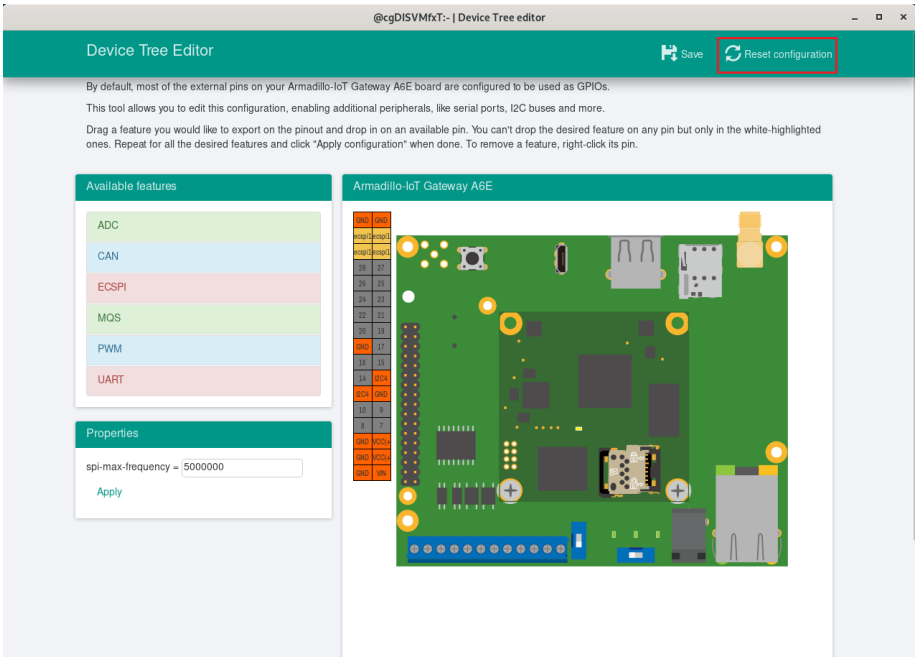


図 6.223 全ての機能の削除

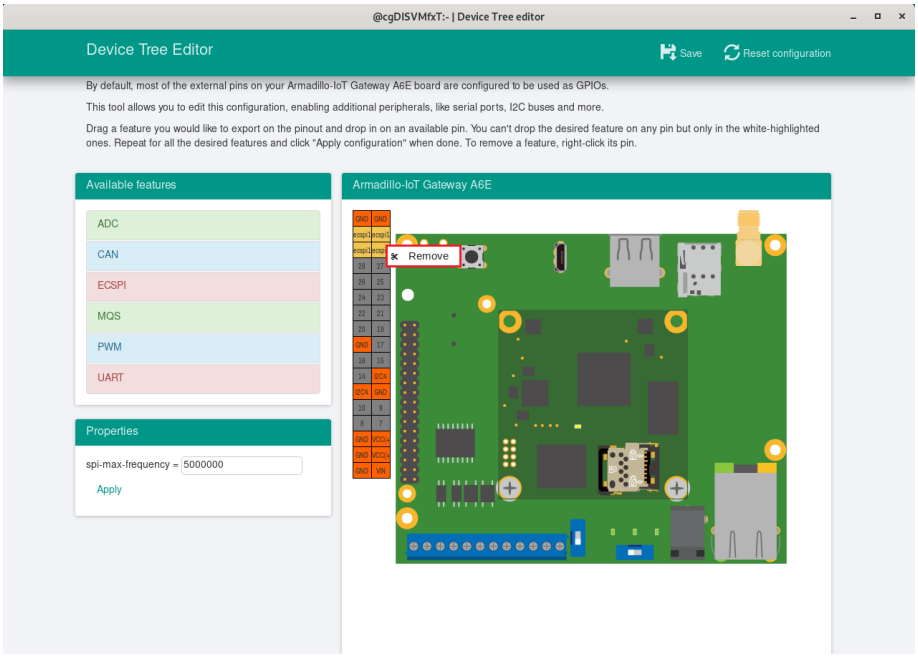


図 6.224 ECSPI1 の削除

6.29.3.5. dtbo/desc の生成

dtbo ファイルおよび desc ファイルを生成するには、画面右上の「Save」をクリックします。

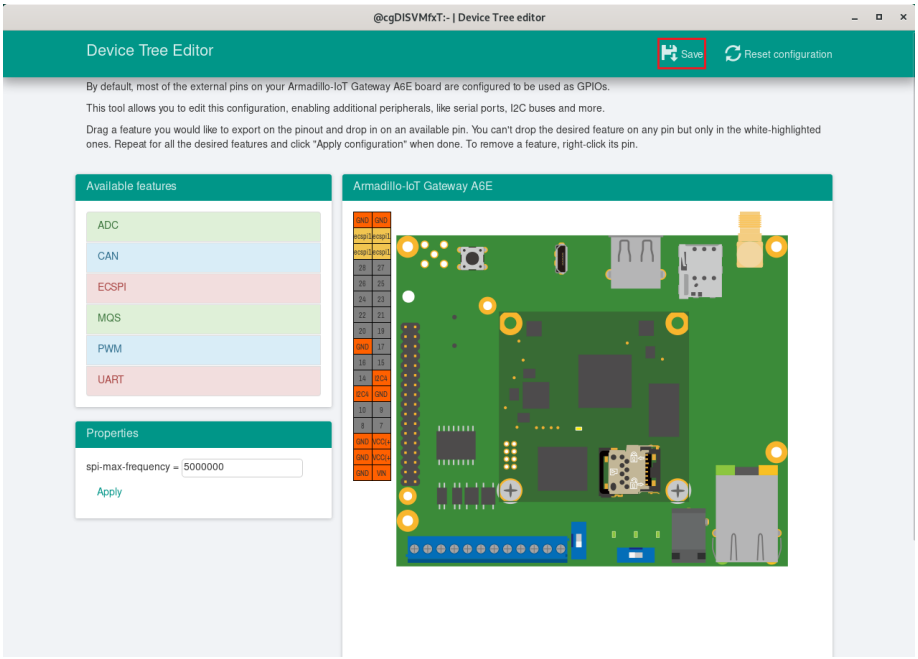


図 6.225 dtbo/desc ファイルの生成

以下の画面ようなメッセージが表示されると、dtbo ファイルおよび desc ファイルの生成は完了です。

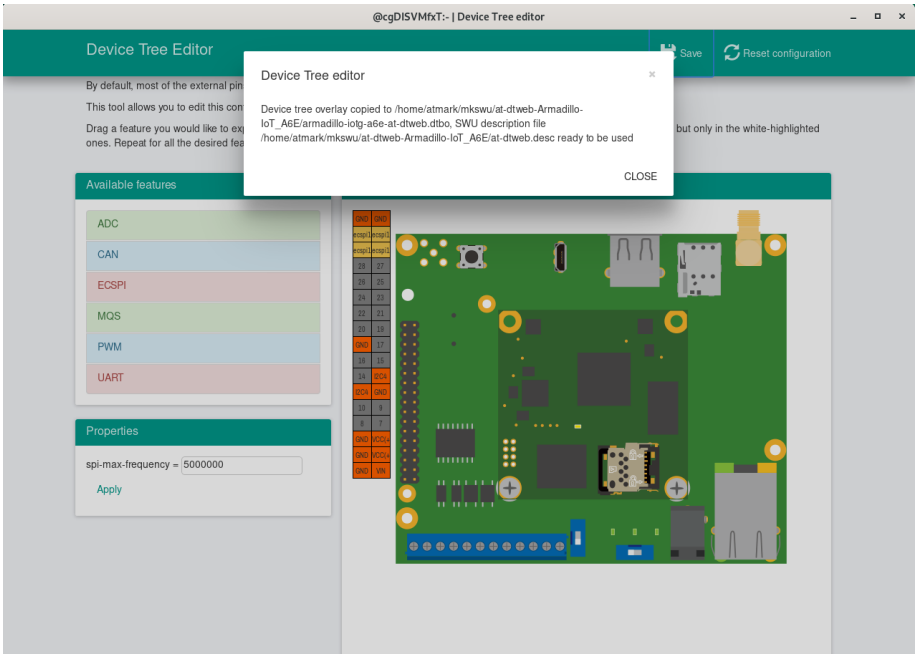


図 6.226 dtbo/desc の生成完了

ビルドが終了すると、ホームディレクトリ下の mkswu/at-dtweb-Armadillo-IoT_A6E/ディレクトリに、DTS overlays ファイル(dtbo ファイル)と desc ファイルが生成されます。Armadillo-IoT ゲートウェイ A6E 本体に書き込む場合は、mkswu コマンドで desc ファイルから SWU イメージを生成してアップデートしてください。

```
[ATDE ~]$ ls ~/mkswu/at-dtweb-Armadillo-IoT_A6E/
armadillo-iotg-a6e-at-dtweb.dtbo  update_overlays.sh
at-dtweb.desc                     update_preserve_files.sh
[ATDE ~]$ cd ~/mkswu/at-dtweb-Armadillo-IoT_A6E/
[ATDE ~/mkswu/at-dtweb-Armadillo-IoT_A6E]$ mkswu at-dtweb.desc ❶
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
at-dtweb.swu を作成しました。
```

❶ SWU イメージを生成します。

SWU イメージを使ったアップデートの詳細は 「3.3.4.5. SWU イメージのインストール」 を参照してください。

6.29.4. DTS overlays によるカスタマイズ

Device Tree は「DTS overlay」(dtbo) を使用することでも変更できます。

DTS overlay を使用することで、通常の dts の更新が自動的に入りつづける状態で dts の変更でしかできない設定を行うことができます。

/boot/overlays.txt に fdt_overlays を dtbo 名で設定することで、u-boot が起動時にその DTS overlay を通常の dtb と結合して起動します。

複数の DTS overlay を使う場合は以下の例のようにスペースで別けたファイル名を記載することができます。

```
[armadillo ~]# ls /boot/ ❶
armadillo-610.dtb
armadillo-640.dtb
armadillo-iotg-a6e-at-dtweb.dtbo
armadillo-iotg-a6e-di8ai4-1st.dtbo
armadillo-iotg-a6e-di8ai4-2nd.dtbo
armadillo-iotg-a6e-di8ai4-3rd.dtbo
armadillo-iotg-a6e-di8ai4-4th.dtbo
armadillo-iotg-a6e-els31.dtbo
armadillo-iotg-a6e-ems31.dtbo
armadillo-iotg-a6e-lwb5plus.dtbo
armadillo-iotg-a6e-sim7672.dtbo
armadillo-iotg-a6e-stdwn-ind-con8-pin7.dtbo
armadillo-iotg-a6e-stdwn-ind-do1.dtbo
armadillo-iotg-a6e.dtb
armadillo.dtb
overlays.txt
uImage
uboot_env.d

[armadillo ~]# persist_file /boot/armadillo-iotg-a6e-at-dtweb.dtbo ❷
[  441.860885] EXT4-fs (mmcblk0p1): re-mounted. Opts: (null)

[armadillo ~]# vi /boot/overlays.txt ❸
fdt_overlays=armadillo-iotg-a6e-sim7672.dtbo armadillo-iotg-a6e-at-dtweb.dtbo

[armadillo ~]# persist_file -vp /boot/overlays.txt ❹
'/boot/overlays.txt' -> '/mnt/boot/overlays.txt'
Added "/boot/overlays.txt" to /etc/swupdate_preserve_files

[armadillo ~]# reboot ❺
: (省略)
Applying fdt overlay: armadillo-iotg-a6e-sim7672.dtbo ❻
Applying fdt overlay: armadillo-iotg-a6e-at-dtweb.dtbo
: (省略)
```

図 6.227 /boot/overlays.txt の変更例

❶ at-dtweb で作成した dtbo ファイルを USB メモリや microSD カード等の外部記憶装置を用いて転送し、 /boot/ ディレクトリ下に配置します。

❷ 配置した dtbo ファイルを保存します。

❸ /boot/overlays.txt ファイルに「armadillo-iotg-a6e-at-dtweb.dtbo」を追加します。ファイルが存在しない場合は新規に作成してください。このファイルの詳細については 「6.29.4. DTS overlays によるカスタマイズ」 を参照してください。

❹ /boot/overlays.txt を保存し、アップデートの場合でも保存します。

❺ overlay の実行のために再起動します。

❻ シリアルコンソールの場合に、u-boot によるメッセージを確認できます。

6.29.4.1. 提供している DTS overlay

以下の DTS overlay を用意しています：

- armadillo-iotg-a6e-sim7672.dtbo: LTE Cat.1 bis モジュール搭載モデルで自動的に使用します。
- armadillo-iotg-a6e-lwb5plus.dtbo: WLAN+BT コンボモジュール搭載モデルで自動的に使用します。
- armadillo-iotg-a6e-stdwn-ind-do1.dtbo: /boot/overlays.txt に記載することで使用できます。使用方法は「6.23. 電源を安全に切るタイミングを通知する」を参照ください。
- armadillo-iotg-a6e-stdwn-ind-con8-pin7.dtbo: /boot/overlays.txt に記載することで使用できます。使用方法は「6.23. 電源を安全に切るタイミングを通知する」を参照ください。
- armadillo-iotg-a6e-di8ai4-1st.dtbo: +Di8+Ai4 拡張ボード接続時に自動的に使用します。

6.29.5. 独自の DTS overlay を追加する

標準イメージで提供している DTS overlay や at-dtweb で作成できないような DTS overlay が必要となった場合に、独自の DTS overlay を作成し Armadillo へ適用する手順を示します。

1. 「6.30.2. Linux カーネルをビルドする」 を参照の上、最新版カーネルのビルドまで実施してください。

   以下、ATDE のホームディレクトリに linux-[VERSION] ディレクトリができている前提で進めます。

2. カスタマイズ用に用意しています arch/arm/boot/dts/armadillo-600-customize.dts を編集します。

```
[ATDE ~/linux-[VERSION]]$ vi arch/arm/boot/dts/armadillo-600-customize.dts
```

図 6.228 armadillo-600-customize.dts の編集

3. 編集したファイルをビルドします。

```
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- dtbs
  DTC     arch/arm/boot/dts/armadillo-600-customize.dtbo
```

図 6.229 編集した dts ファイルのビルド

4. ビルドしてできた armadillo-600-customize.dtbo を Armadillo の /boot/ に配置します。

```
[armadillo ~]# ls /boot/armadillo-600-customize.dtbo
/boot/armadillo-600-customize.dtbo
```

図 6.230 ビルドした DTS overlay ファイルを Armadillo に配置

5. 配置した dtbo を永続化します。このとき、配置した dtbo が SWUpdate 時に消去されてしまわないように、 -p オプションを付与して dtbo を swupdate_preserve_files に追記させます。

```
[armadillo ~]# persist_file -vp /boot/armadillo-600-customize.dtbo
Added "/boot/armadillo-600-customize.dtbo" to /etc/swupdate_preserve_files
'/mnt/boot/armadillo-600-customize.dtbo' -> '/target/boot/armadillo-600-customize.dtbo'
```

図 6.231 ビルドした DTS overlay ファイルを永続化

6. /boot/overlays.txt に armadillo-600-customize.dtbo を追記し、/boot/overlays.txt を永続化します。

```
[armadillo ~]# vi /boot/overlays.txt
fdt_overlays=armadillo-iotg-a9e-sim7672.dtbo armadillo-600-customize.dtbo ❶
[armadillo ~]# persist_file /boot/overlays.txt
```

図 6.232 /boot/overlays.txt の編集と永続化

❶ Cat.1 bis モデルの例です。すでに別の dtbo ファイルが記載されている場合、スペースを挿入して後ろに追加してください。

7. Armadillo を再起動し、動作確認をします。

6.30. Armadillo のソフトウェアをビルドする

ここでは、Armadillo-IoT ゲートウェイ A6E で使用するソフトウェアのビルド方法を説明します。

6.30.1. ブートローダーをビルドする

ここでは、ATDE 上で Armadillo-IoT ゲートウェイ A6E 向けのブートローダーイメージをビルドする方法を説明します。

1. ソースコードの取得

   Armadillo-IoT ゲートウェイ A6E ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/boot-loader] から「ブートローダー ソース」ファイル (u-boot-[VERSION].tar.gz) を「図 6.233. ブートローダーのソースコードをダウンロードする」に示す手順でダウンロードします。

```
[ATDE ~]$ wget https://download.atmark-techno.com/armadillo-iot-a6e/bootloader/u-boot-
[VERSION].tar.gz
[ATDE ~]$ tar xf u-boot-[VERSION].tar.gz
[ATDE ~]$ cd u-boot-[VERSION]
```

図 6.233 ブートローダーのソースコードをダウンロードする

2. デフォルトコンフィギュレーションの適用

   「図 6.234. デフォルトコンフィギュレーションの適用」に示すコマンドを実行します。

```
[ATDE ~/u-boot-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- armadillo-iotg-
a6e_defconfig
  HOSTCC  scripts/basic/fixdep
```

```
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

図 6.234 デフォルトコンフィギュレーションの適用

3. ビルド

   ブートローダーのビルドを実行するには、「図 6.235. ブートローダーのビルド」に示すコマンドを実行します。

```
[ATDE ~/u-boot-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
:
:（省略）
:
  SHIPPED dts/dt.dtb
  FDTGREP dts/dt-spl.dtb
  CAT     u-boot-dtb.bin
  CFGS    u-boot-dtb.cfgout
  MKIMAGE u-boot-dtb.imx
  OBJCOPY u-boot.srec
  COPY    u-boot.bin
  SYM     u-boot.sym
  COPY    u-boot.dtb
  CFGCHK  u-boot.cfg
```

図 6.235 ブートローダーのビルド

4. インストール

   ビルドしたブートローダーは、以下に示すどちらかの方法でインストールしてください。

   - 「3.3.4.5. SWU イメージのインストール」 でインストールする

     mkswu の初期化を行った後に 提供されているスクリプトを使って SWU イメージを作成してください。

```
[ATDE ~/u-boot-[VERSION]]$ echo 'swdesc_boot u-boot-dtb.imx' > boot.desc
[ATDE ~/u-boot-[VERSION]]$ mkswu boot.desc
boot.swu を作成しました。
```

図 6.236 ブートローダーを SWU でインストールする方法

     作成された boot.swu のインストールについては 「3.3.4.5. SWU イメージのインストール」を参照ください。

   - 「6.28.1. ブートディスクの作成」 でインストールする

     手順を参考にして、ビルドされた u-boot-dtb.imx を使ってください。

6.30.2. Linux カーネルをビルドする

ここでは、Armadillo-IoT ゲートウェイ A6E 向けの Linux カーネルイメージをビルドする方法を説明します。



Armadillo-IoT ゲートウェイ A6E では、基本的には Linux カーネルイメージをビルドする必要はありません。「6.30.3. Alpine Linux ルートファイルシステムをビルドする」の手順を実施することで、標準の Linux カーネルイメージがルートファイルシステムに組み込まれます。

標準の Linux カーネルイメージは、アットマークテクノが提供する linux-at という Alpine Linux 用のパッケージに含まれています。

カスタマイズした Linux カーネルイメージを利用する場合は、以下に示す手順を参照してください。

1. ソースコードの取得

   Armadillo-IoT ゲートウェイ A6E Linux カーネル [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/linux-kernel] から「Linux カーネル」ファイル (linux-at-a6-[VERSION].tar) をダウンロードして、「図 6.237. Linux カーネルソースコードの展開」に示すコマンドを実行して展開します。

```
[ATDE ~]$ tar xf linux-at-a6-[VERSION].tar
[ATDE ~]$ tar xf linux-at-a6-[VERSION]/linux-[VERSION].tar.gz
[ATDE ~]$ cd linux-at-a6-[VERSION]/linux-[VERSION]
```

図 6.237 Linux カーネルソースコードの展開

2. デフォルトコンフィギュレーションの適用

   「図 6.238. Linux カーネルデフォルトコンフィギュレーションの適用」に示すコマンドを実行します。

```
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- armadillo-iotg-a6e_defconfig
```

図 6.238 Linux カーネルデフォルトコンフィギュレーションの適用

3. Linux カーネルコンフィギュレーションの変更

   コンフィギュレーションの変更を行わない場合はこの手順は不要です。変更する際は、「図 6.239. Linux カーネルコンフィギュレーションの変更」に示すコマンドを実行します。

```
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

図 6.239 Linux カーネルコンフィギュレーションの変更

コマンドを実行するとカーネルコンフィギュレーション設定画面が表示されます。カーネルコンフィギュレーションを変更後、"Exit"を選択して「Do you wish to save your new kernel configuration? (Press <ESC><ESC> to continue kernel configuration.)」で "Yes" を選択し、 カーネルコンフィギュレーションを確定します。

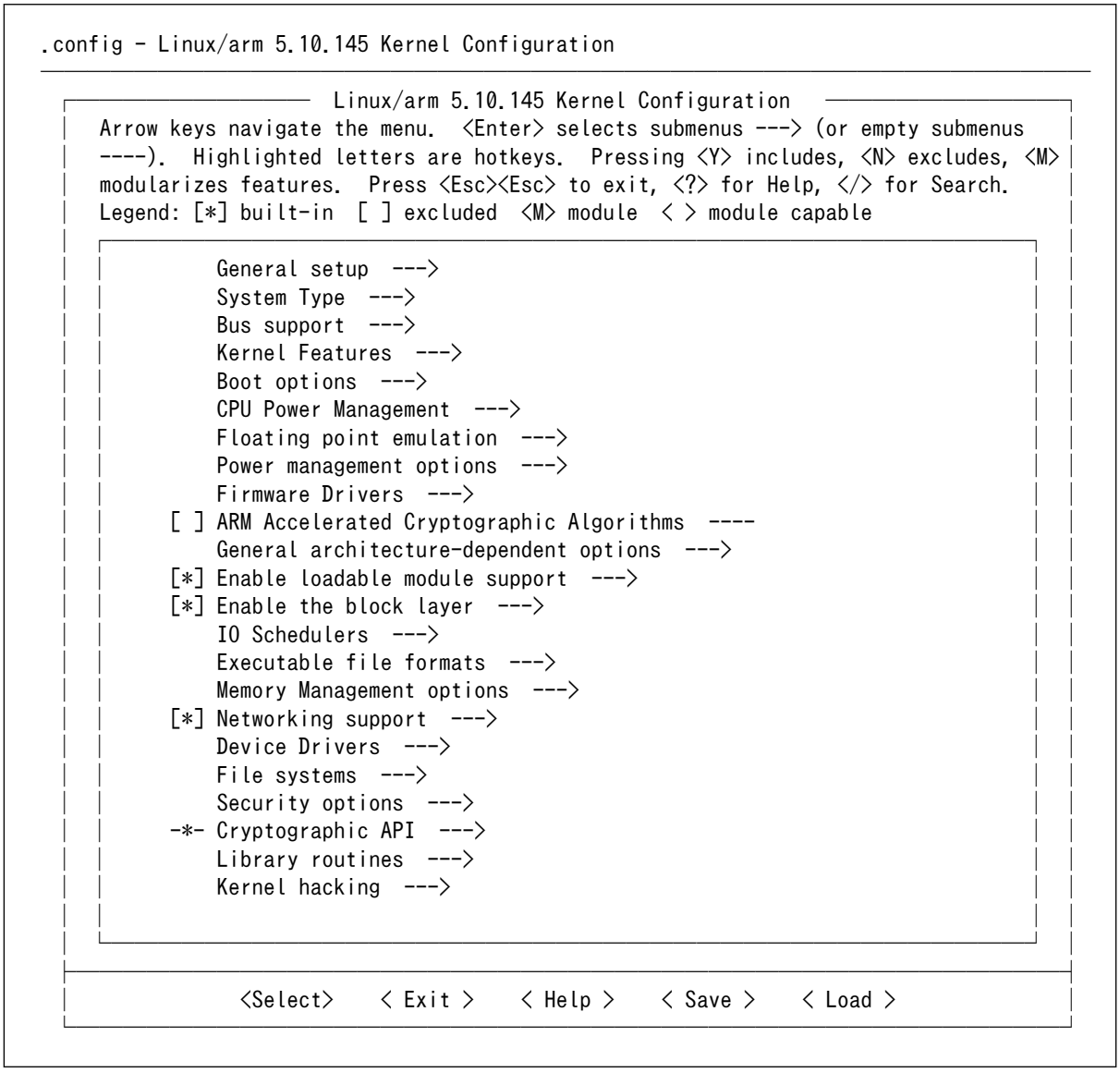


図 6.240 Linux カーネルコンフィギュレーション設定画面



Linux Kernel Configuration メニューで"/"キーを押下すると、カーネルコンフィギュレーションの検索を行うことができます。カーネルコンフィギュレーションのシンボル名(の一部)を入力して"Ok"を選択すると、部分一致するシンボル名を持つカーネルコンフィギュレーションの情報が一覧されます。

4. ビルド

Linux カーネルをビルドするには、「図 6.241. Linux カーネルのビルド」に示すコマンドを実行します。

```
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x82000000 uImage
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

図 6.241 Linux カーネルのビルド

5. インストール

ビルドしたカーネルは、以下に示すどちらかの方法でインストールしてください。

・「3.3.4.5. SWU イメージのインストール」 でインストールする

mkswu の初期化を行った後に 提供されているスクリプトを使って SWU イメージを作成してください。

```
[ATDE ~/linux-[VERSION]]$ /usr/share/mkswu/examples/kernel_update_plain.install.sh ~/
mkswu/kernel.desc
Installing kernel in /home/atmark/mkswu/kerneltest ...
'arch/arm/boot/uImage' -> '/home/atmark/mkswu/kernel/uImage'
'arch/arm/boot/dts/armadillo-610-at-dtweb.dtb' -> '/home/atmark/mkswu/kernel/
armadillo-610-at-dtweb.dtb'
: (省略)
  INSTALL arch/arm/crypto/chacha-neon.ko
  INSTALL arch/arm/crypto/curve25519-neon.ko
: (省略)
  DEPMOD  [VERSION]
Updated /home/atmark/mkswu/kernel.desc version from [PREV_VERSION] to [VERSION]
Done installing kernel, run `mkswu "/home/atmark/mkswu/kernel.desc"` next.
[ATDE ~/linux-[VERSION]]$ mkswu ~/mkswu/kernel.desc
/home/atmark/mkswu/kernel.swu を作成しました
```

図 6.242 Linux カーネルを SWU でインストールする方法

作成された kernel.swu のインストールについては 「3.3.4.5. SWU イメージのインストール」を参照ください。



この kernel.swu をインストールする際は /etc/swupdate_preserve_files の更新例 の様に /boot と /lib/modules を維持するように追加します。カーネルをインストールした後に Armadillo Base OS を更新しても、この kernel.swu のカーネルが維持されます。

標準のカーネルに戻りたいか、以下の 「図 6.243. Linux カーネルを build_rootfs でインストールする方法」 で Armadillo Base OS の更新のカーネルを使用したい場合は /etc/swupdate_preserve_files から /boot と /lib/modules の行を削除してください。

・ build_rootfs で新しいルートファイルシステムをビルドする

build_rootfs を展開した後に以下のコマンドでインストールしてください。

```
[ATDE ~/linux-[VERSION]]$ BROOTFS=$HOME/build-rootfs-[VERSION] ❶
[ATDE ~/linux-[VERSION]]$ sed -i -e '/^linux-at/d' "$BROOTFS/a6e/packages" ❷
[ATDE ~/linux-[VERSION]]$ cp -v arch/arm/boot/uImage "$BROOTFS/a6e/resources/boot/"
'arch/arm/boot/uImage' -> '/home/atmark/build-rootfs-v3.17-at.3/a6e/resources/boot/
uImage'
[ATDE ~/linux-[VERSION]]$ cp -v arch/arm/boot/dts/armadillo*.{dtb,dtbo} "$BROOTFS/a6e/
resources/boot/"
'arch/arm/boot/dts/armadillo-610-at-dtweb.dtb' -> '/home/atmark/build-rootfs-v3.17-at.3/
a6e/resources/boot/armadillo-610-at-dtweb.dtb'
: (省略)
[ATDE ~/linux-[VERSION]]$ rm -rfv "$BROOTFS/a6e/resources/lib/modules" ❸
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
INSTALL_MOD_PATH="$BROOTFS/a6e/resources" -j5 modules_install
  INSTALL arch/arm/crypto/chacha-neon.ko
  INSTALL arch/arm/crypto/curve25519-neon.ko
: (省略)
  DEPMOD  [VERSION]
```

図 6.243 Linux カーネルを build_rootfs でインストールする方法

❶ build_rootfs のディレクトリ名を設定します。これによって、長いディレクトリ名を何度も入力する必要が無くなります。

❷ アットマークテクノが提供するカーネルをインストールしない様に、 linux-at-a6@atmark と記載された行を削除します。

❸ 別のカーネルをすでにインストールしている場合は、新しいモジュールをインストールする前に古いモジュールを削除する必要があります。

6.30.3. Alpine Linux ルートファイルシステムをビルドする

ここでは、build-rootfs を使って、 Alpine Linux ルートファイルシステムを構築する方法を説明します。

build-rootfs は、ATDE 上で Armadillo-IoT ゲートウェイ A6E 用の Alpine Linux ルートファイルシステムを構築することができるツールです。

1. ビルドに必要なパッケージのアップデート

   「図 3.3. ソフトウェアをアップデートする」 を実行します。

2. build-rootfs の入手

   Armadillo-IoT ゲートウェイ A6E 開発用ツール [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/tools] から 「Alpine Linux ルートファイルシステムビルドツール」 ファイル (build-rootfs-[VERSION].tar.gz) を次のようにダウンロードします。

```
[ATDE ~/]$ wget https://download.atmark-techno.com/armadillo-iot-a6e/tool/build-rootfs-
latest.tar.gz
[ATDE ~/]$ tar xf build-rootfs-latest.tar.gz
[ATDE ~/]$ cd build-rootfs-[VERSION]
```

3. Alpine Linux ルートファイルシステムの変更

a6e ディレクトリ以下のファイルを変更することで、ルートファイルシステムをカスタマイズすることができます。



common と a6e ディレクトリ直下にある fixup や packages などの同名ファイルは、それぞれのファイルを連結して利用されます。パッケージの削除などを行う場合は、common ディレクトリ以下のファイルも確認してください。

common と a6e 内のサブディレクトリにある同名ファイルは、a6e のファイルが利用されます。

build-rootfs に含まれるファイルの説明は次の通りです。

表 6.41 build-rootfs のファイル説明

ファイル	説明
a6e/resources/*	配置したファイルやディレクトリは、そのままルートファイルシステム直下にコピーされます。ファイルを追加する場合は、このディレクトリに入れてください。
a6e/packages	このファイルに記載されているパッケージはルートファイルシステムにインストールされます。パッケージを追加する場合はこのファイルに追加してください。
a6e/fixup	このファイルに記載されているコマンドはパッケージのインストールが完了した後に実行されます。
a6e/image_firstboot/*	配置したファイルやディレクトリは、「6.28.1. ブートディスクの作成」や「3.3.6.1. インストールディスクの作成」の手順のようにブートディスクイメージを作成する際、そのままルートファイルシステム直下にコピーされます。
a6e/image_installer/*	配置したファイルやディレクトリは、「3.3.6.1. インストールディスクの作成」の手順のようにインストールディスクイメージを作成する際、そのままインストーラーにコピーされます。ルートファイルシステムに影響はありません。
a6e/image_common/*	配置したファイルやディレクトリは、ブートディスクイメージおよびインストールディスクイメージを作成する際、ルートファイルシステム、インストーラにそれぞれコピーされます。



利用可能なパッケージは以下のページで検索することができます。

Alpine Linux Packages https://pkgs.alpinelinux.org/packages

Alpine Linux ルートファイルシステムを起動している Armadillo でも検索することができます。

```
[armadillo ~]# apk update
[armadillo ~]# apk search ruby
ruby-test-unit-rr-1.0.5-r0
ruby-rmagick-5.1.0-r0
ruby-public_suffix-5.0.0-r0
:
:（省略）
:
```

```
ruby-mustache-1.1.1-r5
ruby-nokogiri-1.13.10-r0
```

4. ビルド

次のコマンドを実行します。

パッケージをインターネット上から取得するため回線速度に依存しますが、ビルドには数分かかります。

```
[ATDE ~/build-rootfs-[VERSION]]$ ./build_rootfs.sh -b a6e
use default(outdir=/home/atmark/git/build-rootfs)
use default(output=baseos-6e-ATVERSION.tar.zst)
:
: (略)
:
> Creating rootfs archive
-rw-r--r--    1 root     root     231700480 Oct 11 07:18 rootfs.tar
ERROR: No such package: .make-alpine-make-rootfs
================================================
footprint[byte]  tarball[byte]  packages
      229904000       74942331  alpine-base coreutils chrony ...(省略)
================================================
done.
```



リリース時にバージョンに日付を含めたくないときは --release を引数に追加してください。



任意のパス、ファイル名で結果を出力することもできます。

```
[ATDE ~/build-rootfs-[VERSION]]$ ./build_rootfs.sh -b a6e ~/
alpine.tar.zst
:
: (略)
:
[ATDE ~/build-rootfs-[VERSION]]$ ls ~/alpine.tar.zst
~/alpine.tar.zst
```

「Alpine Linux ルートファイルシステムビルドツール」のバージョンが 3.18-at.7 以降を使用している場合は、ビルドが終わると SBOM も [output].spdx.json として出力されます。ライセンス情報等を記載するためのコンフィグファイルはデフォルトは baseos_sbom.yaml となっています。コンフィグファイルを変更する場合は --sbom-config <config> に引数を入れてください。SBOM が不要な場合は --nosbom を引数に追加してください。

SBOM のライセンス情報やコンフィグファイルの設定方法については 「6.37.2.3. ビルドしたルートファイルシステムの SBOM を作成する」 をご覧ください。

5. インストール

ビルドしたルートファイルシステムは、以下に示すどちらかの方法でインストールしてください。

- 「3.3.4.5. SWU イメージのインストール」 でインストールする

  mkswu の初期化を行った後に 提供されているスクリプトを使って SWU イメージを作成してください。

```
[ATDE ~/build-rootfs-[VERSION]]$ vi OS_update.desc
swdesc_tar --version base_os [VERSION] ¥
    --preserve-attributes baseos-6e-[VERSION].tar.zst
[ATDE ~/build-rootfs-[VERSION]]$ mkswu OS_update.desc
OS_update.swu を作成しました。
```

  作成された OS_update.swu のインストールについては 「3.3.4.5. SWU イメージのインストール」 を参照ください。

- 「6.28.1. ブートディスクの作成」 でインストールする

  手順を実行すると、ビルドされた baseos-6e-[VERSION].tar.zst が自動的に利用されます。



アットマークテクノがリリースするファームウェアはバージョンのサフィックスとして"-at.[数字]"を含めています。オリジナルのサフィックスをつける場合は、"-at.[数字]"を使用しないことを強く推奨します。

6.31. eMMC のデータリテンション

eMMC は主に NAND Flash メモリから構成されるデバイスです。NAND Flash メモリには書き込みしてから 1 年から 3 年程度の長期間データが読み出されないと電荷が抜けてしまう可能性があります。その際、電荷が抜けて正しくデータが読めない場合は、eMMC 内部で ECC (Error Correcting Code) を利用してデータを訂正します。しかし、訂正ができないほどにデータが化けてしまう場合もあります。そのため、一度書いてから長期間利用しない、高温の環境で利用するなどのケースでは、データ保持期間内に電荷の補充が必要になります。電荷の補充にはデータの読み出し処理を実行し、このデータの読み出し処理をデータリテンションと呼びます。

Armadillo-IoT ゲートウェイ A6E に搭載の eMMC は、eMMC 自身にデータリテンション機能が備わっており、A6E に電源が接続されて eMMC に電源供給されている状態で、eMMC 内部でデータリテンション処理が自動実行されます。

6.32. 動作ログ

6.32.1. 動作ログについて

Armadillo-IoT ゲートウェイ A6E ではシステムが出力するログの一部は、一般的な /var/log ディレクトリではなく、/var/at-log ディレクトリに出力されます。/var/at-log は、ルートファイルシステムとは別のパーティションになっているので、ルートファイルシステムに障害が発生した場合でも、/var/at-log のパーティションが無事であれば、ログファイルを取り出して、不具合等の解析に利用することができます。

Armadillo-IoT ゲートウェイ A6E で /var/log 配下に出力するログに関しては 「6.32.5. /var/log/ 配下のログに関して」 を参照ください。

6.32.2. 動作ログを取り出す

ログファイルは /var/at-log ディレクトリ内に atlog というファイル名で作成されているので、これを任意のディレクトリにコピーすることで取り出せます。もし、eMMC 上のルートファイルシステムが壊れてしまい起動できない場合は、microSD カードから起動することでログファイルを取り出すことができます。



/var/at-log/atlog はファイルサイズが 3MiB になるとローテートされ /var/at-log/atlog.1 に移動されます。

/var/at-log/atlog.1 が存在する状態で、更に /var/at-log/atlog のファイルサイズが 3MiB になった場合は、 /var/at-log/atlog の内容が /var/at-log/atlog.1 に上書きされます。 /var/at-log/atlog.2 は生成されません。

6.32.3. ログファイルのフォーマット

ログファイルの内容はテキストデータであり、以下のようなフォーマットになっています。

```
日時 armadillo ログレベル 機能: メッセージ
```

図 6.244 動作ログのフォーマット

atlog には以下の内容が保存されています。

- インストール状態のバージョン情報
- swupdate によるアップデートの日付とバージョン変更
- abos-ctrl / uboot の rollback 日付
- uboot で wdt による再起動があった場合にその日付

6.32.4. ログ用パーティションについて

ログ出力先である /var/at-log ディレクトリには、GPP である /dev/mmcblk0gp1 パーティションがマウントされています。このパーティションに論理的な障害が発生した場合は、/dev/mmcblk0gp1 のデータを /dev/mmcblk0gp2 にコピーし、/dev/mmcblk0gp1 は FAT ファイルシステムでフォーマットされます。このパーティションの障害チェックはシステム起動時に自動的に実行されます。

6.32.5. /var/log/ 配下のログに関して

「表 6.42. /var/log/ 配下のログ」 に Armadillo-IoT ゲートウェイ A6E で /var/log/ 配下に出力するログを示します。

最大ファイルサイズを超えると 「表 6.42. /var/log/ 配下のログ」 の「ファイル名」の 2 行目に記載されたファイル名にコピーします。

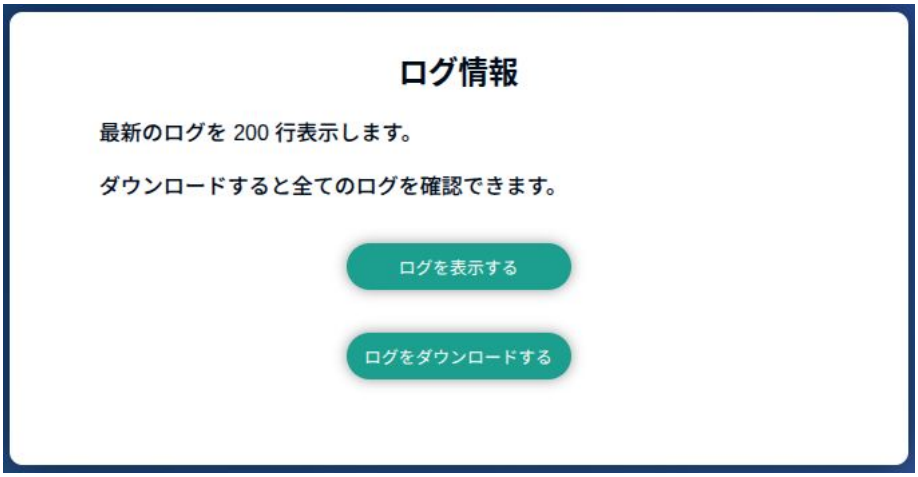
その状態から更に最大ファイルサイズを超えた場合、「表 6.42. /var/log/ 配下のログ」の「ファイル名」の 2 行目に記載されたファイル名に上書きします。

表 6.42 /var/log/ 配下のログ

ファイル名	説明	最大ファイルサイズ	最大ファイル数
/var/log/messages /var/log/messages.0	通常のログです。	4MiB	2
/var/log/armadillo-twin-agent/agent_log /var/log/armadillo-twin-agent/agent_log.1	Armadillo Twin Agent の動作ログです。	1MiB	2

6.32.6. ABOS Web でログを確認する

ログは ABOS Web の「設定管理」ページにある「ログ情報」でも確認およびダウンロードすることができます。



- **ログを表示する**

  `/var/log/messages` の最新の 200 行を表示します。

- **ログをダウンロードする**

  `/var/log` ディレクトリと `/var/at-log` ディレクトリをまとめて圧縮したファイルをダウンロードできます。より詳細なログを確認したい場合は、こちらを使用してください。

6.33. ATDE のバックアップと移行

コンピューター間で ATDE を移動させたい場合、基本的には VirtualBox の管理画面から「仮想アプライアンスのエクスポート」機能を使用して `.ova` 形式でエクスポートし、移行先のコンピューターでインポートすることで簡単に実現できます。

ただし、次のようなケースでは上記の方法では対処できません。

- より新しいマイナーバージョンの ATDE をインストールして作業内容を引き継ぎたい
- 何らかの原因で現在使用している ATDE が故障したため、新たに ATDE を用意して作業内容を引き継ぎたい
- 移行するデータを必要最小限にしたい

・ エクスポートした ATDE を長期間経過後にインポートする際、互換性などの問題でインポートできない可能性を避けるため、どのデバイスでも中身を確認できるように必要なデータだけを確実にバックアップしたい

このような場合は、以下のコマンドでディレクトリやファイルを TAR アーカイブ化します。ディレクトリやファイルを直接保存する方法では、隠しファイルや隠しディレクトリのコピー漏れ、権限・所有者・所有グループの変更によるエラーが発生する可能性がありますが、TAR アーカイブ形式ではこれらの問題を回避できます。

```
[ATDE ~/]$ tar czf atde_archive.tar.gz mkswu/ .ssh/ .abosde/ $(ls -d secureboot_*/ 2>/dev/null)
my_project/
```

図 6.245 ATDE のバックアップ用アーカイブの作成

上記のコマンドでは、ホームディレクトリ直下にある以下のディレクトリを atde_archive.tar.gz という名前のアーカイブにまとめています。

・ mkswu/ : swupdate.key 等の mkswu 用の鍵が格納されています。必ず保存してください。

・ .ssh/ : SSH 鍵が格納されているディレクトリ

・ .abosde/ : ABOSDE で使用する ABOS Web のトークンや、お気に入り機能で登録した名前と MAC アドレスの対応関係などが格納されているディレクトリ

・ secureboot_*/ : セキュアブートの鍵が格納されているディレクトリ（ディレクトリが存在する場合のみ）

・ my_project/ : ユーザーが作成したプロジェクトのディレクトリ

上記以外にも必要なディレクトリがあれば、スペース区切りで適宜追加してください。

移行先の ATDE で以下のコマンドを実行し、アーカイブを展開します。

```
[ATDE ~/]$ tar xzf atde_archive.tar.gz
```

図 6.246 アーカイブの展開

6.34. ATDE・Linux でインストールディスクを作成する

ATDE や Linux でインストールディスクイメージからインストールディスクを作成する方法を紹介します。（Windows でインストールディスクを作成する方法については「3.1.4.1. 初期化インストールディスクの作成」を参照してください。参照先は初期化インストールディスクイメージを使用した方法ですが、それ以外のインストールディスクイメージでも同様の手順で行うことができます。）

ここでは、例として ATDE で初期化インストールディスクを作成する 2 種類の方法（CUI・GUI）を記載しています。初期化インストールディスクイメージ以外のインストールディスクイメージでも同様の手順で行うことができます。

いずれの方法でも、まずは次の共通の手順を実施してください。

1. 1 GB 以上の microSD カードを用意してください。

2. 標準のインストールディスクイメージをダウンロードします。Armadillo-IoT ゲートウェイ A6E インストールディスクイメージ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/disc-image] から「Armadillo Base OS インストールディスクイメージ」をダウンロードしてください。

6.34.1. GUI でインストールディスクを作成する

1. ダウンロードした zip ファイルを展開します。図のとおり、zip ファイルを選択して右クリックし、「ここで展開」をしてください。

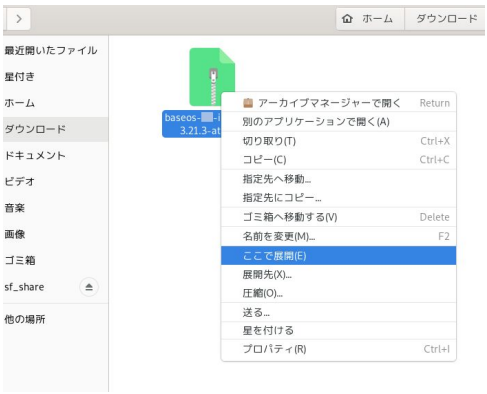


図 6.247 zip ファイルを展開

2. ATDE に microSD カードを接続します。詳しくは「3.1.2.7. 取り外し可能デバイスの使用」を参考にしてください。

3. 展開したフォルダ内にあるインストールディスクイメージ（.img）をダブルクリックして、「ディスクイメージをリストア」のウィンドウを表示します。



図 6.248 展開したフォルダ内にある img ファイルをダブルクリック



図 6.249 ディスクイメージをリストア

4. microSD カードを指定します。[転送先]から接続した microSD カードに対応するものを選びます。

> [転送先]は細心の注意を払って選んでください。[転送先]には ATDE のシステムディスク（右側に VBOX HARDDISK や /dev/atde9-vg* と記載されているもの）も表示されていますが、**これらは決して選択しないでください。**誤って選択して書き込んでしまった場合、ATDE が破壊される可能性があります。
>
> また、他に接続されているデバイスに意図せず書き込んでしまった場合、そのデバイスのデータが上書きされてしまいます。選んだ[転送先]が目的の microSD カードなのかどうか、しっかりと確認してください。microSD カードやカードリーダーの挿抜による[転送先]の表示の変化で確かめることもできます。



図 6.250 microSD カードを指定

5. [リストアを開始]ボタンをクリックして、書き込みを開始します。

6. 確認のウィンドウが現れるので、今一度確認した上で[リストア]をクリックします。



図 6.251 確認のウィンドウ

7. パスワードが要求されるので入力します。



図 6.252 パスワードの要求

8. 書き込みが完了したら、 microSD カードを取り外します。

6.34.2. CUI でインストールディスクを作成する

1. ATDE に microSD カードを接続します。詳しくは「3.1.2.7. 取り外し可能デバイスの使用」を参考にしてください。

2. microSD カードのデバイス名を確認します

```
[ATDE ~]$ ls /dev/sd?
/dev/sda  /dev/sdb
[ATDE ~]$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 7.22 GiB, 7751073792 bytes, 15138816 sectors
Disk model: SD/MMC
:（省略）
```

3. microSD カードがマウントされている場合、アンマウントします。

```
[ATDE ~]$ mount
:（省略）
/dev/sdb1 on /media/52E6-5897 type ext2
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=cp437,iocharset
=utf8,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
[ATDE ~]$ sudo umount /dev/sdb1
```

4. ダウンロードしたファイルを展開し、以下の dd コマンドで img ファイルを microSD カードに書き込んでください。

```
[ATDE ~]$ unzip baseos-6e-installer-[VERSION].zip
[ATDE ~]$ sudo dd if=baseos-6e-installer-[VERSION].img ¥
                 of=/dev/sdb bs=1M oflag=direct status=progress
```

5. 書き込みが完了したら、 microSD カードを取り外します。

6.35. シリアル通信ソフトウェア(minicom)

Linux や ATDE で使用できるシリアル通信ソフトウェアとして minicom の使用方法を紹介します。

6.35.1. シリアル通信ソフトウェア(minicom)のセットアップ



ATDE9 v20240925 以降の ATDE では以下の設定を実施した状態のイメージを配布しています。これより前のバージョンの場合は、次の手順に沿って minicom のシリアル通信設定を実施してください。

minicom を使用して Armadillo とシリアルコンソール経由で通信を行うためには、「表 6.43. シリアル通信設定」 のとおりにあらかじめ設定しておく必要があります。ここでは、その設定手順について説明します。また、minicom を起動する端末の横幅を 80 文字以上にしてください。横幅が 80 文字より小さい場合、コマンド入力中に表示が乱れることがあります。

表 6.43 シリアル通信設定

項目	設定
転送レート	115,200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

1. 「図 6.253. minicom の設定の起動」に示すコマンドを実行し、minicom の設定画面を起動してください。

```
[ATDE ~]$ sudo LANG=C minicom --setup
```

図 6.253 minicom の設定の起動

2. 「図 6.254. minicom の設定」が表示されますので、「Serial port setup」を選択してください。

```
            +-----[configuration]------+
            | Filenames and paths      |
            | File transfer protocols  |
            | Serial port setup        |
            | Modem and dialing        |
            | Screen and keyboard      |
            | Save setup as dfl        |
            | Save setup as..          |
            | Exit                     |
            | Exit from Minicom        |
            +--------------------------+
```

図 6.254 minicom の設定

3. 「図 6.255. minicom のシリアルポートの設定」が表示されますので、A キーを押して Serial Device を選択してください。

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

図 6.255 minicom のシリアルポートの設定

4. Serial Device に使用するデバイスファイル名として /dev/ttyUSB0 を入力して Enter キーを押してください。



デバイスファイル名の確認方法

デバイスファイル名は、環境によって /dev/ttyS0 や /dev/ttyUSB1 など、本書の実行例とは異なる場合があります。

その場合は以下の方法でデバイスファイル名を確認してください。

Linux で PC と Armadillo 側のシリアルポートを接続した場合、コンソールに以下のようなログが表示されます。ログが表示されなくても、dmesg コマンドを実行することで、ログを確認することができます。

例. シリアルポート接続時のログ. 上記の例では Armadillo 側のシリアルポートが ttyUSB0 に割り当てられたことが分かります。

5. F キーを押して Hardware Flow Control を No に設定してください。
6. G キーを押して Software Flow Control を No に設定してください。
7. キーボードの E キーを押してください。「図 6.256. minicom のシリアルポートのパラメータの設定」 が表示されます。

```
                    +---------[Comm Parameters]----------+
                    |                                    |
                    |     Current: 115200 8N1            |
                    | Speed            Parity      Data  |
                    | A: <next>        L: None     S: 5  |
                    | B: <prev>        M: Even     T: 6  |
                    | C:   9600        N: Odd      U: 7  |
                    | D:  38400        O: Mark     V: 8  |
                    | E: 115200        P: Space          |
                    |                                    |
                    | Stopbits                           |
                    | W: 1             Q: 8-N-1          |
```

```
                    | X: 2            R: 7-E-1          |
                    |                                   |
                    |                                   |
                    | Choice, or <Enter> to exit?       |
                    +-----------------------------------+
```

図 6.256 minicom のシリアルポートのパラメータの設定

8. 「図 6.256. minicom のシリアルポートのパラメータの設定」では、転送レート、データ長、ストップビット、パリティの設定を行います。

9. 現在の設定値は「Current」に表示されています。それぞれの値の内容は「図 6.257. minicom シリアルポートの設定値」を参照してください。

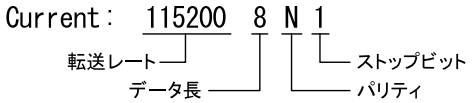


図 6.257 minicom シリアルポートの設定値

10. E キーを押して、転送レートを 115200 に設定してください。

11. Q キーを押して、データ長を 8、パリティを None、ストップビットを 1 に設定してください。

12. Enter キーを 2 回押して、「図 6.254. minicom の設定」に戻ってください。

13. 「図 6.254. minicom の設定」から、「Save setup as dfl」を選択し、設定を保存してください。

14. 「Exit from Minicom」を選択し、minicom の設定を終了してください。



Ctrl-a に続いて z キーを入力すると、minicom のコマンドヘルプが表示されます。

6.35.2. minicom の起動

ATDE の場合は、minicom を起動する前にデバイスを ATDE に接続してください。VirtualBox の左上のメニューから「デバイス」→「USB」からシリアルコンソールとして使用する取り外し可能デバイスを選択してください。

表 6.44 シリアルコンソールとして使用する取り外し可能デバイス

デバイス	デバイス名
USB シリアル変換 IC	Silicon Labs CP2102N USB to UART Bridge Controller

「図 6.258. minicom 起動方法」のようにして、minicom を起動してください。また、minicom を起動する端末の横幅を 80 文字以上にしてください。横幅が 80 文字より小さい場合、コマンド入力中に表示が乱れることがあります。

```
[ATDE ~]$ sudo LANG=C minicom --wrap --device /dev/ttyUSB0
```

図 6.258 minicom 起動方法



デバイスファイル名は、環境によって /dev/ttyS0 や /dev/ttyUSB1 など、本書の実行例とは異なる場合があります。



minicom がオープンする /dev/ttyS0 や /dev/ttyUSB0 といったデバイスファイルは、 root または dialout グループに属しているユーザーしかアクセスできません。

ユーザーを dialout グループに入れることで、以降、sudo を使わずに minicom で /dev/ttyUSB0 をオープンすることができます。

```
[ATDE ~]$ sudo usermod -aG dialout atmark
[ATDE ~]$ LANG=C minicom --wrap --device /dev/ttyUSB0
```

6.35.3. minicom の終了

minicom を終了させるには、まず Ctrl-a に続いて q キーを入力します。その後、以下のように表示されたら「Yes」にカーソルを合わせて Enter キーを入力すると minicom が終了します。

```
+-----------------------+
| Leave without reset? |
|     Yes       No      |
+-----------------------+
```

図 6.259 minicom 終了確認

6.36. vi エディタを使用する

vi エディタは、Armadillo に標準でインストールされているテキストエディタです。本書では、Armadillo の設定ファイルの編集などに vi エディタを使用します。

vi エディタは、ATDE にインストールされてる gedit や emacs などのテキストエディタとは異なり、モードを持っていることが大きな特徴です。vi のモードには、コマンドモードと入力モードがあります。コマンドモードの時に入力した文字はすべてコマンドとして扱われます。入力モードでは文字の入力ができます。

本章で示すコマンド例は ATDE で実行するよう記載していますが、Armadillo でも同じように実行することができます。

6.36.1. vi の起動

vi を起動するには、以下のコマンドを入力します。

```
[ATDE ~]# vi [file]
```

図 6.260 vi の起動

file にファイル名のパスを指定すると、ファイルの編集(file が存在しない場合は新規作成)を行います。vi はコマンドモードの状態で起動します。

6.36.2. 文字の入力

文字を入力するにはコマンドモードから入力モードへ移行する必要があります。コマンドモードから入力モードに移行するには、「表 6.45. 入力モードに移行するコマンド」に示すコマンドを入力します。入力モードへ移行後は、キーを入力すればそのまま文字が入力されます。

表 6.45 入力モードに移行するコマンド

コマンド	動作
i	カーソルのある場所から文字入力を開始
a	カーソルの後ろから文字入力を開始

「i」、「a」それぞれのコマンドを入力した場合の文字入力の開始位置を「図 6.261. 入力モードに移行するコマンドの説明」に示します。

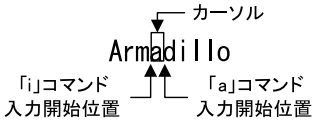


図 6.261 入力モードに移行するコマンドの説明

入力モードからコマンドモードに戻りたい場合は、ESC キーを入力することで戻ることができます。現在のモードが分からなくなった場合は、ESC キーを入力し、一旦コマンドモードへ戻ることにより混乱を防げます。



日本語変換機能を OFF に

vi のコマンドを入力する時は ATDE の日本語入力システム(Mozc)を OFF にしてください。日本語入力システムの ON/OFF は、半角/全角キーで行うことができます。



vi での文字削除

コンソールの環境によっては BS(Backspace)キーで文字が削除できず、「^H」文字が入力される場合があります。その場合は、「6.36.4. 文字の削除」で説明するコマンドを使用し、文字を削除してください。

6.36.3. カーソルの移動

方向キーでカーソルの移動ができますが、コマンドモードで「表 6.46. カーソルの移動コマンド」に示すコマンドを入力することでもカーソルを移動することができます。

表 6.46 カーソルの移動コマンド

コマンド	動作
h	左に 1 文字移動
j	下に 1 文字移動
k	上に 1 文字移動
l	右に 1 文字移動

6.36.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 6.47. 文字の削除コマンド」に示すコマンドを入力します。

表 6.47 文字の削除コマンド

コマンド	動作
x	カーソル上の文字を削除
dd	現在行を削除

「x」コマンド、「dd」コマンドを入力した場合に削除される文字を「図 6.262. 文字を削除するコマンドの説明」に示します。

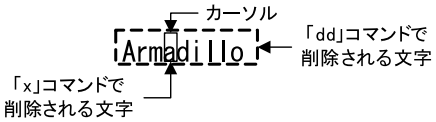


図 6.262 文字を削除するコマンドの説明

6.36.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 6.48. 保存・終了コマンド」に示します。

表 6.48 保存・終了コマンド

コマンド	動作
:q!	変更を保存せずに終了
:w[file]	ファイルを file に指定して保存
:wq	ファイルを上書き保存して終了

保存と終了を行うコマンドは「：」(コロン)からはじまるコマンドを使用します。"："キーを入力すると画面下部にカーソルが移り入力したコマンドが表示されます。コマンドを入力した後 Enter キーを押すことで、コマンドが実行されます。

6.37. セキュリティ

近年、組み込みデバイスの増加に伴い、ハッキングによる脅威も増加の一途を辿っています。サイバー攻撃による被害はそのデバイスの使用者はもちろんですが、信用の失墜や損害賠償といった開発元の企業が被る損害も計り知れません。そのようなリスクを最小限に抑えるために、開発及び運用時にセキュリティを意識することは重要です。

6.37.1. SWUpdate と暗号化について

`mkswu --init` の時に暗号化を有効にする場合は AES でファイルを暗号化します。

現在使われてる SWUpdate の暗号化はコマンドやメタデータを含む sw-description ファイルは暗号化されてません。そのため、通信の暗号化（HTTPS で送信するなど）を使うことを推奨します。

6.37.2. SBOM の提供

アットマークテクノでは ABOS 及び ABOS 上で動作する標準ソフトウェアの SBOM を提供しています。また、開発したソフトウェアの SWU イメージを作成するタイミングで SBOM を生成することができます。SBOM 生成手順は 「6.37.2.3. ビルドしたルートファイルシステムの SBOM を作成する」もしくは 「6.37.2.4. SWU イメージと同時に SBOM を作成する」 を参照ください。

6.37.2.1. SBOM について

SBOM（Software Bill of Materials: ソフトウェア部品表) は、ソフトウェアを構成するコンポーネントやソフトウェア間の依存関係、ライセンス情報を記したリストです。経済産業省は、ソフトウェアサプライチェーンが複雑化する中で、急激に脅威が増しているソフトウェアのセキュリティを確保するための管理手法の一つとして SBOM の導入を推進しています。SBOM の導入はソフトウェアのトレーサビリティを確保し、脆弱性残留リスクの低減、脆弱性対応期間の低減に繋がります。アットマークテクノが提供する SBOM は ISO/IEC5962 で国際標準となっている SPDX2.2 のフォーマットに準拠しています。

SPDX2.2 の詳細については以下のドキュメントをご参照ください。

The Software Package Data Exchange® (SPDX®) Specification Version 2.2.2 [https://spdx.github.io/spdx-spec/v2.2.2/]

アットマークテクノの提供する mkswu コマンドでは SWU を作成するタイミングで SBOM を生成することができます。

6.37.2.2. SBOM の利点

SBOM の利点はソフトウェアのサプライチェーン攻撃への対応です。ソフトウェアのセキュリティ対策は日々見直されており、トレーサビリティが明らかになることで、ソフトウェアに含まれる脆弱性に速やかに対処することが可能になります。SBOM はトレーサビリティを辿るのに優れており、加えて、脆弱性スキャンツールを用いることで、表面化していない脆弱性の発見に利用できます。脆弱性スキャンツールには例として、Google が提供する osv-scanner が挙げられます。脆弱性に関する詳細なリンクや、脆弱性の深刻度を示す CVSS(Common Vulnerability Scoring System) を出力します。アットマークテクノが提供する SBOM は osv-scanner のスキャンに対応しています。

osv-scanner を用いた SBOM のスキャンについては「3.20. 生成した SBOM をスキャンする」をご参照ください。

アットマークテクノが提供している ABOS は GPLv3（GNU General Public License 第 3 版）のソフトウェアを含まない構成で提供しています。OSS（オープンソース・ソフトウェア）利用者に広く普及している GPLv3 は、インストール用情報の開示義務、関連する特許ライセンスの許諾について定める条項が含まれ、組み込み機器に適用する際の妨げになる場合があります。SBOM にはパッケージのライセンス情報が含まれているため、GPLv3 ライセンスが含まれているかどうかの検出を可能にします。

6.37.2.3. ビルドしたルートファイルシステムの SBOM を作成する

「6.30.3. Alpine Linux ルートファイルシステムをビルドする」 を実行すると、OS_update.swu と同じ場所に SBOM を作成します。SBOM を作成するには、作成する対象のファイルとライセンス情報等を記載するためのコンフィグファイルが必要となります。また、baseos-6e-[VERSION].tar.zst から、アーカイブに含まれるパッケージ情報やファイル情報を SBOM に記載します。

ライセンス情報等を記載するためのコンフィグファイルの例は以下のコマンドで確認することができます。各項目に関する説明はコメントに記載しておりますので、必要に応じて値を変更してください。

各項目の詳細な説明については SPDX specification v2.2.2 (https://spdx.github.io/spdx-spec/v2.2.2/) をご覧ください。

```
[ATDE ~/build-rootfs-[VERSION]]$ cat submodules/make-sbom/config.yaml
```

作成したコンフィグファイルと、baseos-6e-[VERSION].tar.zst から OS_update.swu の SBOM を作成します。

```
[ATDE ~/build-rootfs-[VERSION]]$ ./build_sbom.sh -i OS_update.swu -c <コンフィグファイル> -f
baseos-6e-[VERSION].tar.zst
INFO:root:created OS_update.swu.spdx.json
```

作成される SBOM は OS_update.swu.spdx.json になります。json 形式で ISO/IEC5962 で国際標準となっている SPDX2.2 のフォーマットに準拠しています。

アットマークテクノが提供しているソフトウェアの SBOM はソフトウェアダウンロード [https://armadillo.atmark-techno.com/armadillo-iot-a6e/resources/software]の各ソフトウェアダウンロードページからダウンロードすることができます。

6.37.2.4. SWU イメージと同時に SBOM を作成する

「5.4.1. SWU イメージの作成」 の実行時に SBOM を作成する方法について説明します。SWU イメージは desc ファイルから作成されます。この desc ファイルに SBOM 作成に必要な情報についても記載します。

コンフィグファイルを作成する

SBOM を作成するには、作成する対象のファイルとライセンス情報等を記載するためのコンフィグファイルが必要となります。コンフィグファイルについて指定がない場合はデフォルトのコンフィグファイルで SBOM を作成します。デフォルトのコンフィグファイルは /usr/share/make-sbom/config/config.yaml にあります。このファイルは SBOM 作成ツールによって配置されます。コンフィグファイルを編集するために、例としてカレントディレクトリにコピーします。リリース時には正しいコンフィグファイルの内容を記載してください。

```
[ATDE ~]$ cp /usr/share/make-sbom/config/config.yaml .
[ATDE ~]$ vi config.yaml
```

ライセンス情報等を記載するためのコンフィグファイルの例は以下のコマンドで確認することができます。各項目に関する説明はコメントに記載しておりますので、必要に応じて値を変更してください。各項目の詳細な説明については SPDX specification v2.2.2 (https://spdx.github.io/spdx-spec/v2.2.2/) をご覧ください。

「6.37.2.4. SWU イメージと同時に SBOM を作成する」 で desc ファイルに編集したコンフィグファイルのパスを指定します。

desc ファイルを編集する

SBOM 作成のために、desc ファイルに記載する項目を以下に示します。

表 6.49 desc ファイルの設定項目

項目	設定値	説明
swdesc_option BUILD_SBOM=<mode>	auto(デフォルト): SBOM 作成ツールがある場合作成する	SBOM を作成するかどうか。記載がない場合は auto が選択される
	yes: SBOM を作成する。SBOM 作成ツールがない場合はエラーする	
	no: SBOM を作成しない	
swdesc_option sbom_config_yaml=<path>	ファイルパス	コンフィグファイルのパスを指定する。記載がない場合はデフォルトのコンフィグファイルを使用する
swdesc_sbom_source_file <path>	ファイルパス	SBOM に含めるファイルを指定する。記載がない場合は SBOM に含まれない

以下に desc ファイルの記載例について示します。

```
swdesc_option component=make_sbom
swdesc_option version=1
swdesc_option BUILD_SBOM=yes❶
swdesc_option sbom_config_yaml=config.yaml❷

swdesc_sbom_source_file manifest.json❸
```

図 6.263 desc ファイルの追加例

❶ SBOM を作成するように設定します。例として必ず作成するように "yes" を指定します。

❷ コンフィグファイルのパスを設定します。例としてカレントディレクトリにある config.yaml を指定します。

❸ SBOM に含めたいファイルがある場合に指定します。例として manifest.json を指定します。

desc ファイルの作成が出来たら 「5.4.1. SWU イメージの作成」 を実行すると、SWU イメージと同じ場所に SBOM が作成されます。desc ファイルの内容によっては SBOM 作成に数分かかります。作成される SBOM のファイル名は <SWU イメージ名>.spdx.json になります。json 形式で ISO/IEC5962 で国際標準となっている SPDX2.2 のフォーマットに準拠しています。

6.37.3. 不正な USB デバイスの接続を拒否する



USB 接続制御機能は、 ABOS バージョン 3.20.3-at.8 以降で対応しています。



この機能に関して、ABOS バージョン 3.21.3-at.12 以降であれば ABOS Web で設定することも可能です。

「3.10. USB デバイスの接続を許可する」をご参照ください。

```
[armadillo ~]# abos-ctrl usb-filter help
Usage: abos-ctrl usb-filter [action [arguments]]
```

```
Possible actions:
  enable: enabling the USB filter function
  disable: disabling the USB filter
  list-devices [--verbose]: list currently connected USB devices
  list-rules [--verbose]: list currently USB device allowlist
  reset-rules [--force]: reset USB device allowlist
                         with --force option, do not prompt
  allow-device [--vendor-id VID] [--product-id PID]
               [--model MODEL] [--usb-interfaces INTERFACES]
               [--serial SERIAL]: add device to the allowlist
                                  Omitted parameters are stored as wildcard
                                  parameters can be checked with list-devices and
                                  list-rules --verbose flag
  allow-device {DEVID}: allow connected USB devices specified by ID to connect,
                        and add device's information to the allowlist
                        ID can be checked with `abos-ctrl usb-filter list-devices`
  block-device {DEVID}: block alllowed USB devices specified by ID to connect,
                        and delete device's information from the allowlist
                        ID can be checked with `abos-ctrl usb-filter list-devices`
  allow-class [CLASS]...: create allow rule for each USB device class
                          the string for 'CLASS' can be found by running
                          `abos-ctrl usb-filter allow-class`
  remove-rule {RULEID}...: remove the allow rule specified by ID
                           ID can be checked with `abos-ctrl usb-filter list-rules`
  help: show this message
```

図 6.264 USB 接続制御機能を管理するコマンド

SWUpdate から USB 接続制御機能の設定を変更する手順を Armadillo サイトの Howto で公開していますので、必要な場合はご参照ください。

Armadillo サイト Howto 「SWUpdate を用いて USB 接続制御機能の設定を行う」

https://armadillo.atmark-techno.com/howto/setting-usb-filter-with-swupdate

6.37.3.1. USB 接続制御機能を有効/無効化する

現在 USB 接続制御機能が有効か無効かは「図 6.265. USB 接続制御機能の状態を確認する」に示すコマンドを実行することで確認できます。

```
[armadillo ~]# abos-ctrl usb-filter
Currently USB filter is disabled.
```

図 6.265 USB 接続制御機能の状態を確認する

「図 6.266. USB 接続制御機能を有効化する」に示すコマンドを実行することで、 USB 接続制御機能を有効化できます。

```
[armadillo ~]# abos-ctrl usb-filter enable
USB filter enabled.
please reboot to apply rules to currently connected devices
```

図 6.266 USB 接続制御機能を有効化する

有効化したあとに接続された USB デバイスは、設定した許可ルールにしたがって許可/拒否されます。デフォルトでは全てのデバイスは拒否されます。

「図 6.267. USB 接続制御機能を無効化する」に示すコマンドを実行することで、 USB 接続制御機能を無効化できます。

```
[armadillo ~]# abos-ctrl usb-filter disable
USB filter disabled.
please reboot to apply rules to currently connected devices
```

図 6.267 USB 接続制御機能を無効化する

6.37.3.2. 接続済みの USB デバイスの一覧を表示する

「図 6.268. 接続されている USB デバイスをリストする」に示すコマンドを実行することで、 Armadillo に接続されている全ての USB デバイスのリストを表示します。

```
[armadillo ~]# abos-ctrl usb-filter list-devices
     1  block "001:003" "046d" "08e5" "HD_Pro_Webcam_C920" ":0e0100:0e0200:010100:010200:" ⏎
"046d_HD_Pro_Webcam_C920" "/devices/platform/soc@0/32f10100.usb/38100000.dwc3/xhci-hcd.1.auto/ ⏎
usb1/1-1"
```

図 6.268 接続されている USB デバイスをリストする

1 行につき 1 つのデバイスの情報をスペース区切りで示しています。各列が何を示しているかは「表 6.50. デバイスリストの各列の意味」を参照してください。

表 6.50 デバイスリストの各列の意味

1 列目	そのデバイスに割り当たっている ID です。USB デバイスを許可/拒否する際に識別子として使用されます
2 列目	そのデバイスが現在許可(allow)されているか。拒否(block)されているかを示します
3 列目	そのデバイスに割り当たっている USB バス番号とデバイス番号のペアです
4 列目	そのデバイスのベンダー ID です
5 列目	そのデバイスのモデル ID です
6 列目	そのデバイスのモデル名です
7 列目	そのデバイスの USB クラスコードです。デバイスによっては複数存在するものもあり、":"(コロン)区切りで表示されます
8 列目	そのデバイスのシリアル番号です。一般に同じ製品であっても個体ごとに一意な値です
9 列目	そのデバイスに割り当たっている/sys 以下のディレクトリパスです

6.37.3.3. USB デバイスの接続を許可する

「図 6.268. 接続されている USB デバイスをリストする」のコマンドを実行して、2 列目が「block」となっているデバイスは拒否されているデバイスです。このデバイスに対して、「図 6.269. 指定した

USB デバイスを許可する」に示すコマンドを実行することで接続を許可し、今後再接続されたとしても許可します。

```
[armadillo ~]# abos-ctrl usb-filter allow-device 1 ❶
allowed the following device:
    "001:003" "046d" "08e5" "HD_Pro_Webcam_C920" ":0e0100:0e0200:010100:010200:"
"046d_HD_Pro_Webcam_C920" "/devices/platform/soc@0/32f10100.usb/38100000.dwc3/xhci-hcd.1.auto/
usb1/1-1" ❷
```

図 6.269 指定した USB デバイスを許可する

❶ この例では、「図 6.268. 接続されている USB デバイスをリストする」のコマンドを実行して ID が 1 のデバイスを許可しています

❷ 許可されたデバイスの情報が表示されます

また、ベンダー/モデル ID やシリアル番号などの情報を前もって知っている場合は、それらの情報を用いて USB デバイスの接続を許可することも可能です。

```
[armadillo ~]# abos-ctrl usb-filter allow-device ¥
--vendor-id "046d" ¥
--product-id "08e5" ¥
--model "HD_Pro_Webcam_C920" ¥
--usb-interfaces ":0e0100:0e0200:010100:010200:" ¥
--serial "046d_HD_Pro_Webcam_C920"
allowed the following device:
    "046d" "08e5" "HD_Pro_Webcam_C920" ":0e0100:0e0200:010100:010200:" "046d_HD_Pro_Webcam_C920"
```

図 6.270 パラメータで指定した USB デバイスを許可する

「図 6.270. パラメータで指定した USB デバイスを許可する」において、1 つ以上のパラメータが指定されていない場合、その指定しなかったパラメータはワイルドカードとして扱われます。この場合、デバイスを接続したときに、指定したパラメータさえ完全に一致していれば、ワイルドカードとして扱われているパラメータが何であってもデバイスの接続が許可されます。



Armadillo に接続されている USB デバイスであれば 「図 6.271. Armadillo に接続している USB デバイスのパラメータを調べる」に示すコマンドを実行することで、「図 6.270. パラメータで指定した USB デバイスを許可する」に指定すべきパラメータを調べることができます。

```
[armadillo ~]# abos-ctrl usb-filter list-devices -v
     1  block "001:003" --vendor-id "046d" --product-id "08e5" --model
"HD_Pro_Webcam_C920" --usb-interfaces ":0e0100:0e0200:010100:010200:" --
serial "046d_HD_Pro_Webcam_C920" "/devices/platform/soc@0/32f10100.usb/
38100000.dwc3/xhci-hcd.1.auto/usb1/1-1"
```

図 6.271 Armadillo に接続している USB デバイスのパラメータを調べる

6.37.3.4. USB デバイスの接続を拒否する

「図 6.268. 接続されている USB デバイスをリストする」のコマンドを実行して、2 列目が「allow」となっているデバイスは許可されているデバイスです。このデバイスに対して、「図 6.272. 指定した USB デバイスを拒否する」に示すコマンドを実行することで接続を拒否し、今後再接続されたとしても拒否します。

```
[armadillo ~]# abos-ctrl usb-filter block-device 1 ❶
blocked the following device:
    "001:003" "046d" "08e5" "HD_Pro_Webcam_C920" ":0e0100:0e0200:010100:010200:"
"046d_HD_Pro_Webcam_C920" "/devices/platform/soc@0/32f10100.usb/38100000.dwc3/xhci-hcd.1.auto/
usb1/1-1" ❷
```

図 6.272 指定した USB デバイスを拒否する

❶ この例では、「図 6.268. 接続されている USB デバイスをリストする」のコマンドを実行して ID が 1 のデバイスを拒否しています

❷ 拒否されたデバイスの情報が表示されます

6.37.3.5. USB デバイスクラス単位で USB デバイスの接続を許可する

「6.37.3.3. USB デバイスの接続を許可する」の手順では、 USB デバイスのベンダー ID やプロダクト ID、シリアル番号などが完全一致したデバイスのみを許可します。そのため同じメーカーの同じデバイスであっても、シリアル番号が一致しないと許可されません。ここでは、「USB メモリの接続はどのメーカーのどの製品であっても全て許可するが、 USB カメラなどの接続は拒否する」といったルールを手軽に作成する機能を紹介します。

abos-ctrl usb-filter allow-class コマンドを用いて、指定した USB デバイスクラスのみを許可することができます。例として、「図 6.273. 指定した USB デバイスクラスを許可する」では、 USB MassStorage クラスのデバイス(USB メモリなど)を許可します。

```
[armadillo ~]# abos-ctrl usb-filter allow-class MassStorage
```

図 6.273 指定した USB デバイスクラスを許可する

「図 6.273. 指定した USB デバイスクラスを許可する」の例では、引数として"MassStorage"を指定していますが、他にも指定できるデバイスクラスがあります。「図 6.274. 指定可能な USB デバイスクラスを確認する」に示すコマンドを実行することで、引数として指定できる文字列を確認できます。

```
[armadillo ~]# abos-ctrl usb-filter allow-class
supported USB device classes:
        Audio
        CDC
        HID
        Physical
        Image
        Printer
        MassStorage
        Hub
        CDCdata
        SmartCard
```

```
        ContentSecurity
        Video
        PersonalHealthCare
```

図 6.274 指定可能な USB デバイスクラスを確認する

各 USB デバイスクラスについては、 Defined Class Codes [https://www.usb.org/defined-class-codes] を参照してください。

1 つのデバイスで複数のデバイスクラスを持っている場合、そのデバイスの全てのデバイスクラスが許可されていなければ認識されません。例えば、「図 6.268. 接続されている USB デバイスをリストする」の 1 番目に認識されている USB カメラは Video と Audio の 2 つのデバイスクラスを持っています。このデバイスは、 Video と Audio 両方のデバイスクラスを許可している場合に認識されます。

6.37.3.6. 定義済みの USB デバイス許可ルールを表示する

「図 6.269. 指定した USB デバイスを許可する」や「図 6.272. 指定した USB デバイスを拒否する」、「図 6.273. 指定した USB デバイスクラスを許可する」を実行すると、 USB デバイスの許可ルールが更新されます。許可ルールに記載されているデバイスは、接続されたときに認識され、使用できます。

「図 6.275. 定義済みの USB デバイス許可ルールを表示する」に示すコマンドを実行することで現在の許可ルールの一覧を表示できます。

```
[armadillo ~]# abos-ctrl usb-filter list-rules
     1  class MassStorage
     2  device "046d" "08e5" "HD_Pro_Webcam_C920" ":0e0100:0e0200:010100:010200:"
"046d_HD_Pro_Webcam_C920"
```

図 6.275 定義済みの USB デバイス許可ルールを表示する

各行の最初の数値は、そのルールに割り当たった ID です。この ID は ルールを個別に削除する際に使用されます。詳細は「6.37.3.7. 定義済みの USB デバイス許可ルールを削除する」を参照してください。

2 列目の文字列は、そのルールがデバイスクラス単位の許可ルールであるか、デバイス固有の情報に基づいた許可ルールであるかを示します。
この列が class であれば、当該のルールはデバイスクラス単位の許可ルール(「6.37.3.5. USB デバイスクラス単位で USB デバイスの接続を許可する」によって追加されたルール)であり、3 列目の文字列は許可する USB デバイスクラス名です。
この列が device であれば、当該のルールはデバイス固有の情報に基づいた許可ルール(「6.37.3.3. USB デバイスの接続を許可する」によって追加されたルール)であり、以降の文字列はその条件に一致したデバイスが接続されると接続が許可されるルールです。この場合の 3 列目以降の文字列の意味を「表 6.51. 2 列目が device のときの許可ルールリストの各列の意味」に示します。

表 6.51 2 列目が device のときの許可ルールリストの各列の意味

3 列目	このルールによって許可されるベンダー ID です
4 列目	このルールによって許可されるモデル ID です
5 列目	このルールによって許可されるモデル名です
6 列目	このルールによって許可される USB クラスコードです。複数ある場合も全てが完全一致した場合のみ許可されます
7 列目	このルールによって許可されるシリアル番号です

1 つのルールにつき 1 行で表記され、接続したデバイスが全てのルールのうちどれか 1 つでも満たしていれば許可されます。

6.37.3.7. 定義済みの USB デバイス許可ルールを削除する

定義した許可ルールは削除することができます。削除することでそのデバイスは再接続時に接続が拒否され使用できなくなります。

「図 6.275. 定義済みの USB デバイス許可ルールを表示する」に示す状況のときに、"MassStorage"の許可を削除する例を「図 6.276. 定義済みの USB デバイス許可ルールを削除する」に示します。

```
[armadillo ~]# abos-ctrl usb-filter remove-rule 1
[armadillo ~]# abos-ctrl usb-filter list-rules
     1  device "046d" "08e5" "HD_Pro_Webcam_C920" ":0e0100:0e0200:010100:010200:"
"046d_HD_Pro_Webcam_C920"
```

図 6.276 定義済みの USB デバイス許可ルールを削除する

6.37.4. EdgeLock SE050 を利用したキーストレージ

NXP の EdgeLock SE050 は IoT アプリケーション向けのセキュアエレメントです。フラッシュメモリを内蔵しており、保存された秘密鍵を外部に露出することなく暗号処理に利用できます。ここでは Egelock SE050 をキーストレージとして利用する方法を説明します。



EdgeLock SE050 の詳細については以下のデータシートを参照してください。

SE050 datasheet

https://www.nxp.com/docs/en/data-sheet/SE050-DATASHEET.pdf

6.37.4.1. EdgeLock SE05x Plug & Trust Middleware

EdgeLock SE050 を利用するためのソフトウェアとして、NXP から ミドルウェア EdgeLock SE05x Plug & Trust Middleware (以下、Plug & Trust Middleware) が提供されています。ただし、ビルド済みバイナリは提供されないため利用環境に合わせてビルドの必要があります。

Armadillo Base OS では Alpine Linux と Debian GNU/Linux 向けに Plug & Trust Middleware のビルド済みパッケージを提供しています。以下は Plug & Trust Middleware 周辺のソフトウェアスタックです。提供中のビルド済みパッケージは赤い四角のブロックになります。

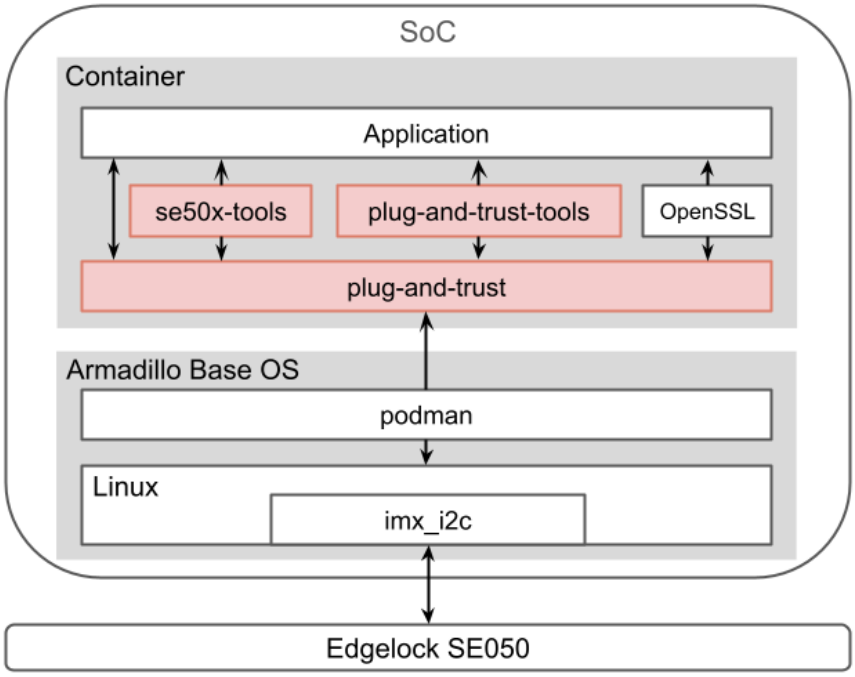


図 6.277 Plug & Trust Middleware の周辺のソフトウェアスタック

提供しているパッケージは以下のとおりです。

表 6.52 Plug & Trust Middle のパッケージ

パッケージ	内容
plug-and-trust	Plug & Trust Middleware のビルド済みライブラリ群。OpenSSL engine を含む。
plug-and-trust-dev	Plug & Trust Middleware の開発用ファイル。アプリケーション開発に利用できます。
plug-and-trust-tools	Plug & Trust Middleware に含まれるサンプルアプリケーション。最小限の動作確認ツール (se05x_Minimal) や機能や固有情報を取得するツール (se05x_GetInfo) など。
se05x-tools	非対称暗号向けの鍵の読み書きツール。

Armadillo Base OS が対応するディストリビューションとバージョンは以下のとおりです。

表 6.53 Plug & Trust Middleware のサポート

ディストリビューション	バージョン
Alpine Linux	3.21
Debian GNU/Linux	11 (bullseye)
Debian GNU/Linux	12 (bookworm)



本ガイド作成時点では利用したバージョンは以下のとおりです。

- Plug & Trust Middleware : 04.02.00
- plug-and-trust : 4.2.0

・ se05x-tools : 1.2.2

6.37.4.2. EdgeLock SE050 を有効にする

EdgeLock SE050 は ENA ピンがアサートされると有効化されます。

Armadillo-IoT ゲートウェイ A6E の EdgeLock SE050 はデフォルトで自動的に有効化され、スリープモード中は無効化されるようになっているため、設定は不要です。

6.37.4.3. Plug & Trust Middleware のインストール

Debian と Alpine Linux に対応しますが、ここでは Alpine Linux で作業を進めます。

コンテナを立ち上げるために、/etc/atmark/containers ディレクトリに、plug-and-trust.conf ファイルを以下の内容で作成します。

```
[armadillo ~]# vi /etc/atmark/containers/plug-and-trust.conf
set_image docker.io/alpine
add_armadillo_env
add_devices "${AT_SE_PARAM%:*}"
add_volumes /etc/apk:/etc/apk:ro
set_command sleep infinity
```



Debian を利用する場合は、at-debian の利用をお勧めします。at-debian はアットマークテクノによる動作確認済みの環境です。apt-get install を利用してインストールしてください。

「6.9.5. アットマークテクノが提供するイメージを使う」を参考にしてください。

ベースとなる alpine コンテナをインストールします。

```
[armadillo ~]# abos-ctrl podman-rw pull docker.io/alpine
```

コンテナを起動し、コンテナの中に入ります。

```
[armadillo ~]# podman_start plug-and-trust
[armadillo ~]# podman exec -it plug-and-trust sh
```



以下のエラーが出る場合は、Armadillo Base OS のバージョンを最新にするか、plug-and-trust.conf を書き換えて再度実行してみてください。

```
[armadillo ~]# podman_start plug-and-trust
error: plug-and-trust: device  does not exist
[armadillo ~]# vi /etc/atmark/containers/plug-and-trust.conf
```

```
set_image docker.io/alpine
export AT_SE_PARAM="$(device-info --se-param)"
add_args --env=AT_SE_PARAM
add_devices "${AT_SE_PARAM%:*}"
add_volumes /etc/apk:/etc/apk:ro
set_command sleep infinity
```

パッケージをインストールします。

```
[container /]# cd
[container ~]# apk add se05x-tools plug-and-trust-tools
```

利用のために環境変数を設定します。

```
[container ~]# export OPENSSL_CONF=/etc/plug-and-trust/openssl11_sss_se050.cnf
[container ~]# export EX_SSS_BOOT_SSS_PORT="$AT_SE_PARAM"
```

インストールと環境設定が終わったら、se05x_GetInfo で EdgeLock SE050 の動作確認を確認します。アクセスに成功すると以下のようなログが出力されます。

```
[container ~]# se05x_GetInfo
App   :INFO :PlugAndTrust_v04.02.00_20220524
App   :INFO :Running se05x_GetInfo
App   :INFO :Using PortName='/dev/i2c-1:0x48' (ENV: EX_SSS_BOOT_SSS_PORT=/dev/i2c-1:0x48)
sss   :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
App   :WARN :No SemsLite Applet Available.
App   :INFO :Running se05x_GetInfo
App   :INFO :Using PortName='/dev/i2c-1:0x48' (ENV: EX_SSS_BOOT_SSS_PORT=/dev/i2c-1:0x48)
sss   :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
App   :WARN :#####################################################
App   :INFO :uid (Len=18)
      04 00 50 01    EA 31 EF DE    83 23 DA 04    0F 81 AA 2A
      1E 90
App   :WARN :#####################################################
App   :INFO :Applet Major = 3
App   :INFO :Applet Minor = 1
App   :INFO :Applet patch = 1
App   :INFO :AppletConfig = 6FFF
App   :INFO :With    ECDSA_ECDH_ECDHE
App   :INFO :With    EDDSA
App   :INFO :With    DH_MONT
App   :INFO :With    HMAC
App   :INFO :With    RSA_PLAIN
App   :INFO :With    RSA_CRT
```

```
App   :INFO :With    AES
App   :INFO :With    DES
App   :INFO :With    PBKDF
App   :INFO :With    TLS
App   :INFO :With    MIFARE
App   :INFO :With    I2CM
:（省略）
```

以上で Plug & Trust Middleware を利用するための準備を終わります。

6.37.4.4. Plug & Trust Middleware を活用する

ここからは実際に Plug & Trust Middleware を用いて EdgeLock SE050 を利用する方法を説明していきます。

EdgeLock SE050 の保存された鍵を利用する

plug-and-trust パッケージには OpenSSL engine のライブラリが含まれており、OpenSSL を利用して間接的に EdgeLock SE050 を操作することができます。ここでは例として、EdgeLock SE050 のチップ製造時に書き込まれた鍵を使って OpenSSL で署名と署名の検証を行います。

まず、次のコマンドでリファレンスキーを取得します。

- keyid = 0xF0000100
- Cloud connection key 0 (prime256v1)

```
[container ~]# se05x_getkey 0xF0000100 refkey.pem "$AT_SE_PARAM"
:(省略)
[container ~]# ls
refkey.pem ❶
```

❶ refkey.pem が生成されます。



チップ製造時に書き込まれた鍵について

「6.37.4.5. 補足説明」 を参照してください。



リファレンスキーについて

「6.37.4.5. 補足説明」 を参照してください。

次のコマンドで署名して sig.bin を生成します。message.txt は任意のファイルを用意してください。取得したリファレンスキー refkey.pem を利用します。

```
[container ~]# echo "This is test" > message.txt
[container ~]# openssl dgst -sha256 -sign refkey.pem -out sig.bin message.txt
```

```
ssse-flw: EmbSe_Init(): Entry
App   :INFO :Using PortName='/dev/i2c-1:0x48' (ENV: EX_SSS_BOOT_SSS_PORT=/dev/i2c-1:0x48)
sss   :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
ssse-flw: Version: 1.0.5
ssse-flw: EmbSe_Init(): Exit
ssse-flw: Control Command EMBSE_LOG_LEVEL; requested log level = 4
[container ~]# ls
message.txt  refkey.pem   sig.bin ❶
```

❶ sig.bin が生成されます。

署名が正しいかどうかを確認するために署名を検証します。

```
[container ~]# openssl dgst -sha256 -prverify refkey.pem -signature sig.bin message.txt
ssse-flw: EmbSe_Init(): Entry
App   :INFO :Using PortName='/dev/i2c-1:0x48' (ENV: EX_SSS_BOOT_SSS_PORT=/dev/i2c-1:0x48)
sss   :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
ssse-flw: Version: 1.0.5
ssse-flw: EmbSe_Init(): Exit
ssse-flw: Control Command EMBSE_LOG_LEVEL; requested log level = 4
Verified OK
```



クラウドサービスに接続するために事前書き込みされた鍵と証明書を利用する方法は以下を参照してください(Armadillo-IoT ゲートウェイ G4 向けの内容ですが、本機種でも同じ手順でご利用いただけます)。

EdgeLock SE050 を使用して AWS IoT Core へ接続する

https://armadillo.atmark-techno.com/howto/connect_to_aws_iot_core_for_aiot_g4

EdgeLock SE050 を使用して Azure IoT Hub へ接続する

https://armadillo.atmark-techno.com/howto/connect_azure_iot_hub_for_aiot_g4

ユーザーが生成した鍵を保存して利用する

EdgeLock SE050 ではユーザーが生成した鍵を保存することもできます。一度保存された鍵は、そのまま暗号処理に利用することができるようになります。ここでは例として、ユーザー鍵を保存して openssl cms で暗号化と復号を実行してみます。

以降の例で生成されるファイルの役割を説明します。

- key.pem: openssl で生成された秘密鍵
- cert.pem: openssl で生成された自己証明書（公開鍵）
- refkey.pem: SE050 に保存した秘密鍵 key.pem を使用するためのリファレンスキー
- message.txt: 任意のファイル
- message.enc: message.txt を暗号化したファイル
- message.dec: message.enc を復号したファイル

以降の作業のため、「6.37.4.4. Plug & Trust Middleware を活用する」で生成したファイルは削除します。

```
[container ~]# rm -rf *
```

まず、ECC の曲線 prime256v1 の鍵を生成します。証明書の情報はお任せします。

```
[container ~]# openssl req -x509 -nodes -days 3650 -newkey ec ¥
-pkeyopt ec_paramgen_curve:prime256v1 -keyout key.pem -out cert.pem
```

生成した鍵と自己証明書を EdgeLock SE050 に保存します。EdgeLock SE050 では keyid が 0x00000001 から 0x7BFFFFFF の範囲で開放されており、 50kB のストレージが搭載されています。ただし、今後アットマークテクノが一部ストレージを利用する予定であるため、Armadillo-IoT ゲートウェイ A6E で利用できる keyid およびストレージ容量は以下の通りとなります。

- keyid: 0x00000001 ～ 0x7BFEFFFF
- ストレージ容量: 35kB

```
[container ~]# se05x_setkey -f 0x10 key.pem "$AT_SE_PARAM"
[container ~]# se05x_setkey -f 0x11 cert.pem "$AT_SE_PARAM"
```

SE050 に秘密鍵 key.pem を保存したので、流出を防ぐために key.pem を削除します。

```
[container ~]# rm -rf key.pem
```



Secure Object についての詳しい情報は以下を参照してください。

SE050A/B/C/D/F APDU Specification

https://www.nxp.com/search?keyword=AN12413



秘密鍵を EdgeLock SE050 に保存に成功したあとは、利用する際に EdgeLock SE050 にアクセスすることになります。そのため、ファイルシステム上の key.pem は基本的に必要ありません。セキュリティ上、削除することをお勧めします。

0x10 に保存した秘密鍵を使用するためのリファレンスキー refkey.pem を生成します。

```
[container ~]# se05x_getkey 0x10 refkey.pem "$AT_SE_PARAM"
```



リファレンスキーについて

「6.37.4.5. 補足説明」 を参照してください。

自己証明書に含まれる公開鍵 cert.pem を使って暗号化します。message.txt は任意のファイルを用意してください。

```
[container ~]# echo "This is test" > message.txt
[container ~]# openssl cms -encrypt -binary -aes256 -in message.txt ¥
-out message.enc cert.pem
[container ~]# ls
cert.pem  message.enc  message.txt  refkey.pem ❶
```

❶ message.txt を暗号化した message.enc が生成されます。

以下はあくまで例ですが、暗号化されたファイルは以下のような内容になります。

```
[container ~]# cat message.enc
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=enveloped-data; name="smime.p7m"
Content-Transfer-Encoding: base64

MIIBWgYJKoZIhvcNAQcDoIIBSzCCAUcCAQIxggECoYH/AgEDoFGhTzAJBgcqhkjO
PQIBA0IABPFaZ4KiT2MfZYLMgyhKmTRMeU8+dOwLVW9qe5l6QuFgPUmIVq5Q/jC1
teAvZVli90gmGmvH+GqkFy2rVKpvYEowGAYJK4EFEIZIPwACMAsGCWCGSAFlAwQB
LTCBjDCBiTBdMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQQIDApTb21lLVN0YXRlMSEw
HwYDVQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQCFB7+NC1Xtvz6O3ptiaPN
hHqnkocSBCgVCiONik9gab2iWroRv4VLp35RKBGY/YRv4OytUgEzOV7ejHTzoFPa
MDwGCSqGSIb3DQEHATAdBglghkgBZQMEASoEEL4YFn0m8QCBHSBJZHeWV16AEEWr
AlzrhnvA3jiOYZCX4e4=
```

EdgeLock SE050 のキーストレージにある秘密鍵を使って復号します。

```
[container ~]# openssl cms -decrypt -in message.enc -out message.dec ¥
-inkey refkey.pem
```

```
[container ~]# cat message.dec
This is test
```

ユーザーが保存した鍵を削除する

「6.37.4.4. Plug & Trust Middleware を活用する」で EdgeLock SE050 にユーザーが生成した鍵を削除する方法を紹介します。注意点としまして、複数保存した鍵の中から 1 つだけを選んで削除することはできません。EdgeLock SE050 内に保存されているユーザーが保存した鍵をすべて削除し、その後復元できませんのでよく理解した上でご使用ください。



「6.37.4.5. 補足説明」で紹介している、事前に書き込まれた鍵と証明書は削除されません。

EdgeLock SE050 内に保存されているユーザーが保存した鍵をすべて削除するコマンドを「図 6.278. EdgeLock SE050 内に保存されているユーザーが保存した鍵をすべて削除する」に示します。

```
[container ~]# export EX_SSS_BOOT_SSS_PORT="$AT_SE_PARAM"
[container ~]# se05x_reset -y
:(省略)
secure objects have been successfully removed.
```

図 6.278 EdgeLock SE050 内に保存されているユーザーが保存した鍵をすべて削除する

削除後、保存されていた keyid から鍵を読み出そうとしても失敗します。

```
[container ~]# se05x_getkey 0x10 refkey "$AT_SE_PARAM"
:(省略)
failed to retrieve handle with keyid.
keyid may be invalid. please check keyid.
failure occurred in sss api (kStatus_SSS_Fail)
```

図 6.279 削除したユーザーが保存した鍵を読み出そうとしてみる

6.37.4.5. 補足説明

事前書き込みされた鍵と X.509 証明書

EdgeLock SE050 のフラッシュメモリにはチップ製造時に書き込まれたチップ固有な鍵と NXP に署名された X.509 証明書を保持しています。それらの鍵や証明書は Armadillo-IoT ゲートウェイ A6E を購入後にすぐにそのままの状態でクラウドサービスなどの PKI に利用できます。事前に書き込みがされた状態で空き容量はあるので、ユーザーの鍵や証明書を追加することも可能です。



事前書き込みされた鍵と証明書については以下のドキュメントを参照してください。利用可能な keyid、鍵の種類、想定する用途が記載されています。

なお、Armadillo-IoT ゲートウェイ A6E に搭載されている EdgeLock SE050 は Variant C です。

SE050 configuration

https://www.nxp.com/search?keyword=AN12436

リファレンスキー

EdgeLock SE050 は秘密鍵が外部に露出しません。これは、いったん鍵を EdgeLock SE050 に書き込むと秘密鍵を抜き出すことができない (削除や書き換えは可能) という仕様で実現されています。攻撃者がチップの中にある、どこかのサイトの認証資格などを有する証明書を得るためには、物理的にチップ自体も必要ということになってきます。

秘密鍵を抜き出すことができない仕様で、どのように秘密鍵を利用するかというと、リファレンスキーと呼ばれる特殊なファイルで代用します。リファレンスキーは秘密鍵のフォーマットで保存されますが、公開鍵部分のみ有効で、それ以外の値は管理用やダミーの情報になっています。リファレンスキーをファイルシステムに置いておいて、EdgeLock SE050 を利用する時にそのファイルを秘密鍵のように使うと、Plug & Trust Middleware に含まれる OpenSSL の engine のライブラリがフックして EdgeLock SE050 にアクセスします。

e05x-tools

se05x-tools は EdgeLock SE050 をキーストレージとして利用するためのツールです。アットマークテクノから Armadillo Base OS 向けにリリースされています。se05x-tools の内部では Plug & Trust Middleware を利用しています。

対応する非対称暗号鍵の種類は以下のとおりです。

se05x-tools でサポートされる ECC カーブ

- prime192v1, secp224r1, prime256v1, secp384r1, secp521r1

se05x-tools でサポートされる RSA 鍵長

- 1024bit, 2048bit, 3072bit, 4096bit



Secure Object についての詳しい情報は以下を参照してください。

SE050A/B/C/D/F APDU Specification

https://www.nxp.com/search?keyword=AN12413

6.37.5. デバッグ機能を閉じる

ここでは、デバッグ機能を閉じる方法とその影響について説明します。

デバッグ機能は開発の効率を向上させるために開発フェーズではなくてはならないものです。製品が市場に出荷されると市場不良の解析にも効果を発揮します。しかし、開発者にとって便利であるということは、同時に攻撃者にとっても便利な解析手段となり得ます。想定される製品の運用形態によって、デバッグ機能が必須である運用形態もあります。セキュリティリスクとのトレードオフになる可能性があるので、有効にするかどうかを検討することをお勧めします。



デバッグ機能を閉じることは開発者と攻撃者だけでなく、**アットマークテクノによる解析についてもトレードオフになります。閉じられたインターフェースを利用した解析ができなくなることをご留意いただきますようお願いします。**

6.37.5.1. JTAG と SD ブートを無効化する

Armadillo-IoT ゲートウェイ A6E の出荷時は、JTAG ポートは有効なままで出荷されます。JTAG が有効なままですと攻撃者が悪意のあるコードを実行する、メモリをダンプして鍵などセキュアな情報を取り出すなどの行為が可能となります。そのため、市場に出荷される最終段階では JTAG を無効化することをお勧めします。

次に、SD ブートは SD メディアを挿すだけで起動する便利な機能です。その反面、SD メディアの盗難や流出によってシステムへの侵入、SD ブートを利用したセキュアブート鍵に対する攻撃が考えられます。これらのリスクは、SD ブートを無効にすることで排除することが可能です。しかし、SD ブートはシステムの復旧の役割も担っています。SD ブートを無効化することによって、eMMC ブートでは起動できない状態に陥った場合、合わせて JTAG の無効化が合わせて設定されていると、二度と復旧することができないデバイスになる (廃棄するしかない) 可能性があることを考慮してください。

ソフトウェアの開発が完了し量産段階に入ると、量産用 Armadillo にソフトウェアを書き込むためのインストールディスクが必要になります。Armadillo Base OS では開発が完了した Armadillo 上でコマンドを実行することにより、その Armadillo のルートファイルシステム及びブートローダーをそのままインストールディスクイメージとして作成することができます。詳細な情報は「4.4.4. 開発したシステムをインストールディスクにする」を参照してください。

JTAG と SD ブートを無効化したインストールディスクを作成するには `abos-ctrl make-installer` コマンドを実行する前に、`abos-ctrl installer-setting` コマンドで設定します。



作成したインストールディスクを使用して初期化した Armadillo の JTAG と SD ブートを無効にする設定であり、開発用の Armadillo の JTAG と SD ブートが無効になることはありません。

```
[armadillo /]# abos-ctrl installer-setting
Would you like to disable JTAG in the installer ? [y/N] ❶
y
JTAG disabled setting for production Armadillo has been configured.
Would you like to disable SD boot in the installer ? [y/N] ❷
y
SD boot disabled setting for production Armadillo has been configured.
```

図 6.280 インストールディスクの JTAG と SD ブートを無効化する

❶ JTAG を無効化する場合は y を入力します。無効化しない場合は何も入力せず Enter キーを押してください。

❷ SD ブートを無効化する場合は y を入力します。無効化しない場合は何も入力せず Enter キーを押してください。

現在の設定値を確認するには abos-ctrl check-secure コマンドを実行します。disabled の場合は無効化する設定になっています。

```
[armadillo /]# abos-ctrl check-secure
- JTAG access disabled for mass production.
- SD boot access disabled for mass production.
```

図 6.281 JTAG と SD ブートの設定値を確認する

設定をリセットするには --reset オプションを付けて abos-ctrl installer-setting コマンドを実行します。

```
[armadillo /]# abos-ctrl installer-setting --reset
cleaned up all settings.
```

図 6.282 JTAG と SD ブートの設定値をリセットする

JTAG と SD ブートの無効化設定を実行した後に、 abos-ctrl make-installer コマンドを実行してインストールディスクを作成します。コマンド実行前に、Armadillo がインターネットに接続されており、かつ 10GB 以上の空き容量がある microSD カードが挿入されていることを確認してください。 microSD カード内のデータはインストールディスク作成時に上書きされて消えてしまうので、必要なデータは予めバックアップを取っておいてください。コマンド実行例は以下のようになります。

```
[armadillo /]# abos-ctrl make-installer
Checking if /dev/mmcblk1 can be used safely...
It looks like your SD card does not contain an installer image
Download baseos-6e-installer-latest.zip image from armadillo.atmark-techno.com (~170M) ? [y/N] ❶
y
WARNING: it will overwrite your SD card!!

Downloading and extracting image to SD card...
Finished writing baseos-6e-installer-[VERSION].img, verifying written content...

Would you like to create a windows partition?
That partition would only be used for customization script at the end of
install, leave at 0 to skip creating it.
Custom partition size (MB, [0] or 16 - 29014): 500 ❷
Checking and growing installer main partition
Trying to install mkfs.exfat (exfatprogs) in memory from internet
fetch https://download.atmark-techno.com/alpine/v3.19/atmark/armv7/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/main/armv7/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/community/armv7/APKINDEX.tar.gz
(1/1) Installing exfatprogs (1.2.2-r0)
Executing busybox-1.36.1-r15.trigger
OK: 148 MiB in 197 packages
exfatprogs version : 1.2.2
Creating exFAT filesystem(/dev/mmcblk1p2, cluster size=131072)
```

```
Writing volume boot record: done
Writing backup volume boot record: done
Fat table creation: done
Allocation bitmap creation: done
Upcase table creation: done
Writing root directory entry: done
Synchronizing...

exFAT format complete!
Resize device id 1 (/dev/mmcblk1p1) from 520.00MiB to max
Installer will disable JTAG access
Installer will disable SD boot after installation
Copying boot image
Copying rootfs
Copying appfs
At subvol app/snapshots/volumes
At subvol app/snapshots/boot_volumes
At subvol app/snapshots/boot_containers_storage
Cleaning up and syncing changes to disk...
Installer updated successfully!
```

図 6.283 インストールディスクを作成する

❶ y を入力します。

❷ インストールディスク内にインストールログを保存したい場合など、自由に使用できる第 2 パーティションを指定したサイズ作成します。

「Installer updated successfully!」と表示されれば、正常に microSD カードにインストールディスクイメージを書き込むことができています。 Armadillo から microSD カードを抜去してください。この microSD カードを使って量産用 Armadillo にインストールを行うと、その Armadillo は JTAG と SD ブートが無効化された状態となります。

6.37.6. コンテナの最小化

開発時に使用するパッケージが最新でセキュリティに問題がないと考えられていても、後に脆弱性が発見される可能性があります。

コンテナ内のパッケージに脆弱性が見つかった場合、開発者は公開されたパッチを適用する必要があります。しかし、インストールされているパッケージが多いと、アップデートが必要なパッケージを把握するのが困難になることがあります。

その結果、メンテナンスされていないパッケージが残り、セキュリティリスクが高まる可能性があります。

このようなリスクを回避するためには、コンテナにインストールするパッケージを必要最低限に抑えることが重要です。

コンテナを最小化する主な利点は以下の通りです。

- セキュリティリスクの低減：不要なパッケージやサービスを減らすことで、脆弱性の発生箇所が少なくなり、攻撃対象領域が縮小します。
- 保守性の向上：インストールされているパッケージが少ないため、アップデートや脆弱性対応が容易になります。

- コストの削減：コンテナイメージが小さくなることで、ネットワーク経由でのダウンロードや転送が速くなり、ストレージの消費も抑えられます。これにより、転送コストや時間の削減につながります。
- ライセンス管理の簡素化：含まれるソフトウェアが少ないため、ライセンス管理や監査が容易になります。

VS Code の拡張機能「Armadillo Base OS Development Environment（ABOSDE）」では、コンテナにインストールされるパッケージを最小化する工夫がされています。

もし ABOSDE を使用しない場合でも、以下の内容を参考にすることで、コンテナのセキュリティを向上させることができます。

6.37.6.1. コンテナに使用する OS の選択

ABOSDE が提供する Shellscript/Python/C 言語のプロジェクトでは、arm64v8 と arm32v7 の両方で Debian または Alpine コンテナを選択できます。

Debian コンテナでは、ディストリビューションとして `bullseye-slim` を使用しています。

以下は、Debian と Alpine のコンテナ内パッケージを比較した表です。（以降、2025/7/1 時点の情報です。）

表 6.54 Debian と Alpine の比較

比較項目	Debian	Alpine
パッケージ管理システム	apt	apk
インストール済みパッケージ数	88 個	16 個
利用可能パッケージ数	豊富	Debian に比べて少ない

Alpine は利用可能なパッケージ数が少ないため、開発難易度が高くなる可能性があります。しかし、Alpine を採用することで、Debian よりもコンテナ内のパッケージ数を減らすことができます。

以下は、Python プロジェクトを Debian と Alpine でビルドした場合のコンテナイメージサイズの比較例です。どちらも LED を点滅させるアプリケーションを動作させています。

表 6.55 Python プロジェクトにおける Debian と Alpine コンテナのサイズ比較

アーキテクチャ	Debian	Alpine
arm32v7	117 MB	64.8 MB
arm64v8	139 MB	76.1 MB

この例では、どちらのアーキテクチャにおいても Python プロジェクトのコンテナサイズは Alpine コンテナベースが Debian コンテナベースの約半分であることが分かります。

ABOSDE におけるコンテナ OS の選択方法は「3.17.3.4. ABOSDE におけるコンテナ OS の選択方法」を参照してください。

6.37.6.2. Debian コンテナにインストールするパッケージを減らす方法

Debian を使用する場合、依存パッケージを減らすことでコンテナサイズを小さくできます。

ABOSDE が提供する Dockerfile では、デフォルトで必要最低限のパッケージのみをインストールするオプションが指定されています。

```
:（省略）
```

```
# Add extra packages to containers/packages.txt
ARG PACKAGES
RUN apt-get update && apt-get upgrade -y ¥
    && apt-get install -y --no-install-recommends ${PACKAGES} ¥ ❶
    && apt-get clean

:(省略)
```

図 6.284 Debian コンテナの Dockerfile の例

❶ --no-install-recommends オプションを指定しています。

--no-install-recommends オプションを使用すると、必要最低限のパッケージのみをインストールできます。

以下は、Python プロジェクトのサンプルプログラムを使用して、--no-install-recommends の有無による Debian コンテナのサイズを比較した結果です。(2025/7/1 時点)

表 6.56 Python プロジェクトにおける --no-install-recommends の有無による Debian コンテナのサイズ比較

アーキテクチャ	オプション無し	オプションあり
arm32v7	317 MB	117 MB
arm64v8	418 MB	139 MB

この例から、どちらのアーキテクチャでも --no-install-recommends オプションを指定することで、コンテナサイズを約 1/3 に削減できることが分かります。

6.38. オプション品

本章では、Armadillo-IoT ゲートウェイ A6E のオプション品について説明します。

表 6.57 Armadillo-IoT ゲートウェイ A6E 関連のオプション品

名称	型番
AC アダプタ（12V/2.0A φ2.1mm）温度拡張品 効率レベル VI 品	OP-AC12V4-00
Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)	OP-CASEA6E-PLA-20
+Di8+Ai4 拡張基板	-[a]

[a]アットマークテクノ 営業部までお問い合わせください。

6.38.1. Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)

6.38.1.1. 概要

Armadillo-IoT ゲートウェイ A6E 用のプラスチック製ケースです。

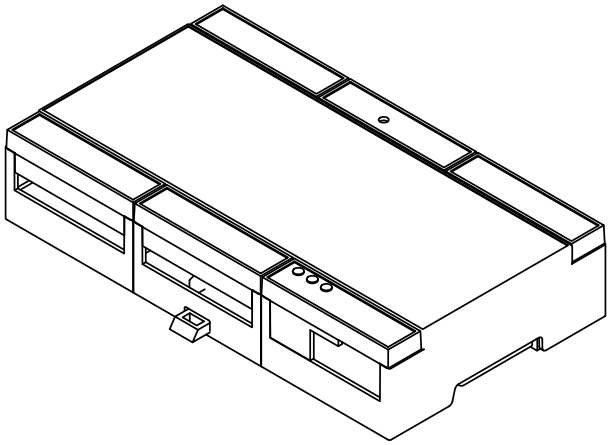


図 6.285 ケース外観図



Armadillo-IoT ゲートウェイ A6E　標準ケースセットロング(9M) は Armadillo-IoT ゲートウェイ A6E +Di8+Ai4 に付属しています。ケースのみ必要なお客様のためにオプション品として別売りもしています。

表 6.58 Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)について

商品名	Armadillo-IoT ゲートウェイ A6E 標準ケースセットロング(9M)
型番	OP-CASEA6E-PLA-20
内容	ケース(トップ/ボトム)、カバーパーツ(A(アンテナ穴有り)/A(アンテナ穴無し)/B(アンテナ穴有り)/B(アンテナ穴無し)/C/D)、LED ライトパイプ、フック

表 6.59 ケース(トップ/ボトム)の仕様

材質	PC/ABS 混合樹脂
難燃性	UL94 V-0
色	グレー
使用温度範囲	-20～90℃

表 6.60 フックの仕様

材質	PRO 樹脂
難燃性	UL94 V-0
色	黒
使用温度範囲	-20～90℃

表 6.61 カバーパーツ A/B/C/D の仕様

材質	PC 樹脂
難燃性	UL94 V-0
色	グレー
使用温度範囲	-20～90℃



最高使用温度よりも高い温度で保管または使用した場合、樹脂ケースが変形する可能性があります。

6.38.1.2. 組み立て

組み立て方法については Armadillo-IoT ゲートウェイ A6E +Di8+Ai4 製品マニュアルの「組み立てと分解」の章をご参照ください。

6.38.1.3. 形状図

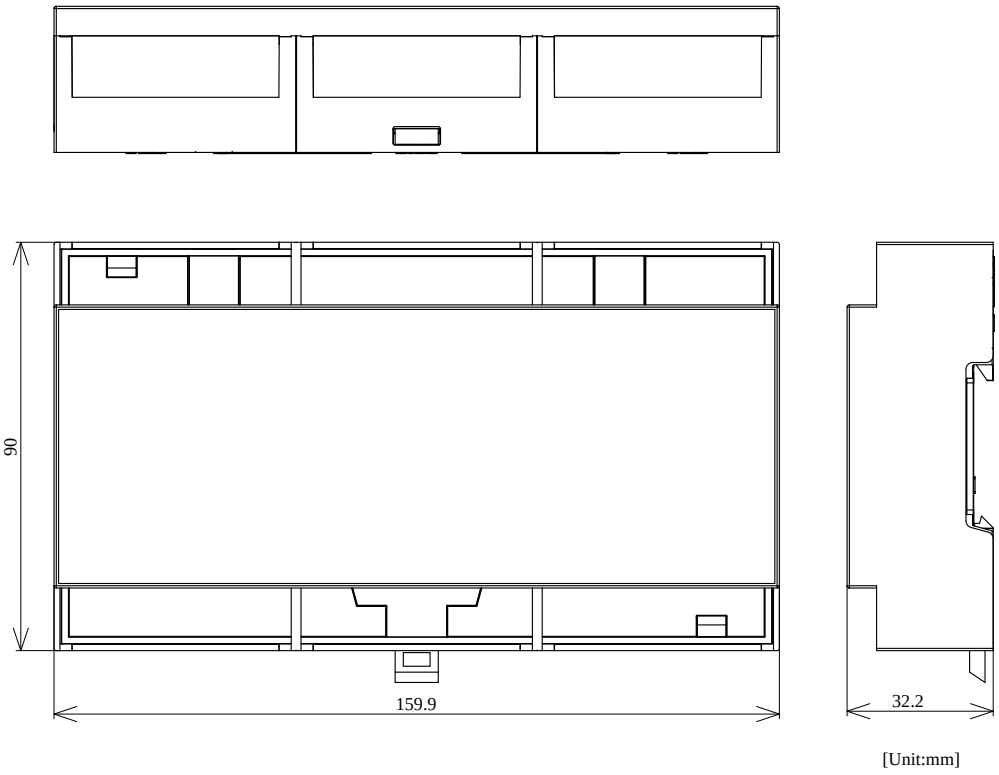


図 6.286 形状図 ケース外形(トップとボトムを組み合わせた状態)

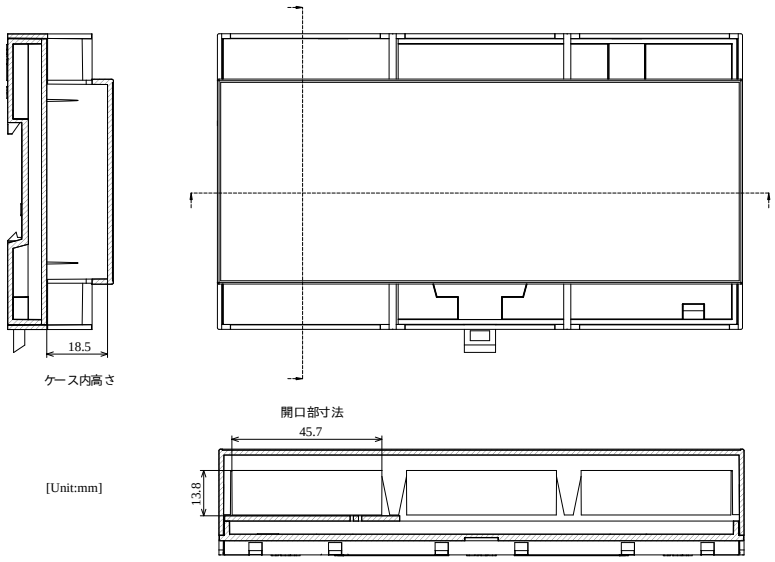


図 6.287 形状図 ケース内高さおよび開口部寸法

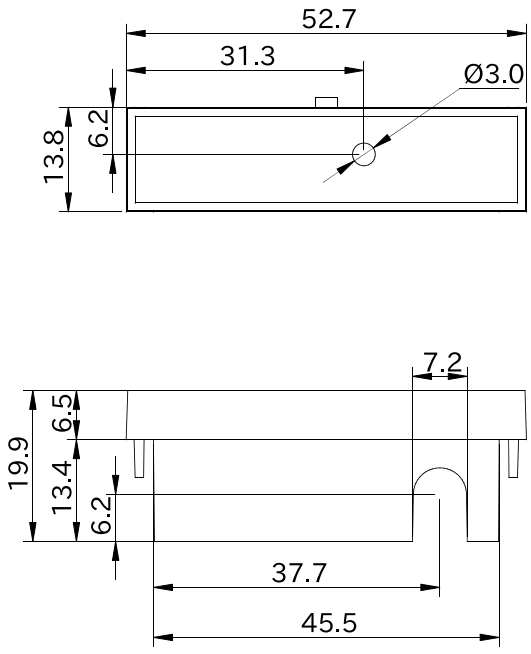


図 6.288 形状図 カバーパーツ A (アンテナ穴有り)

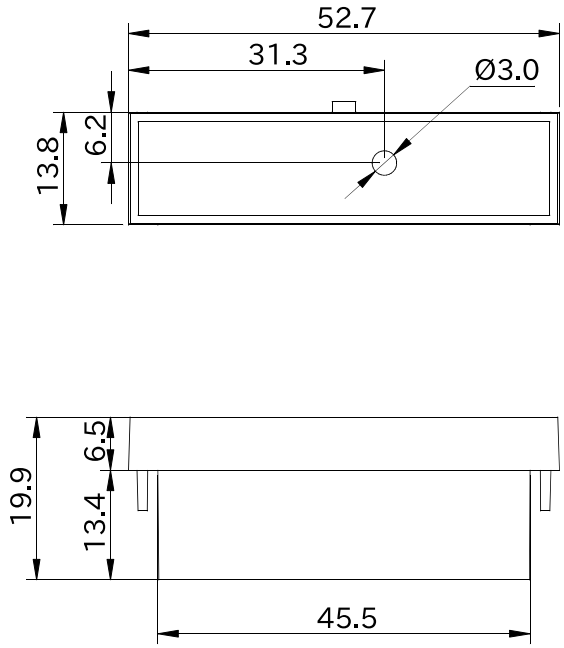


図 6.289 形状図 カバーパーツ A (アンテナ穴無し)

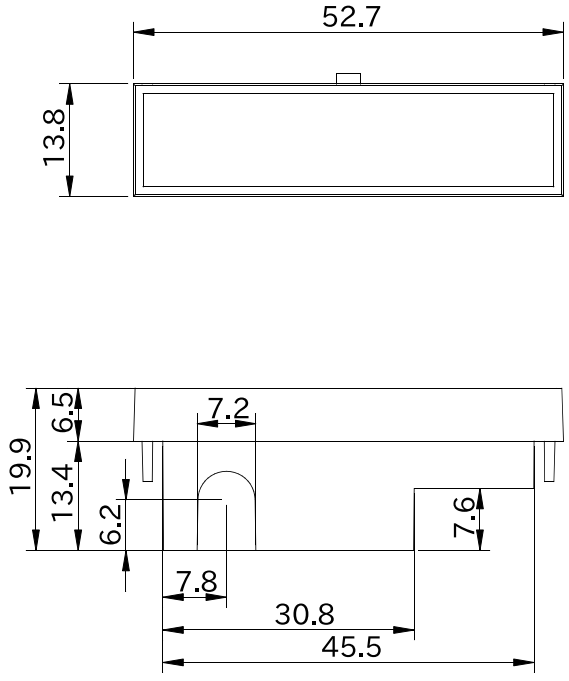


図 6.290 形状図 カバーパーツ B (アンテナ穴有り)

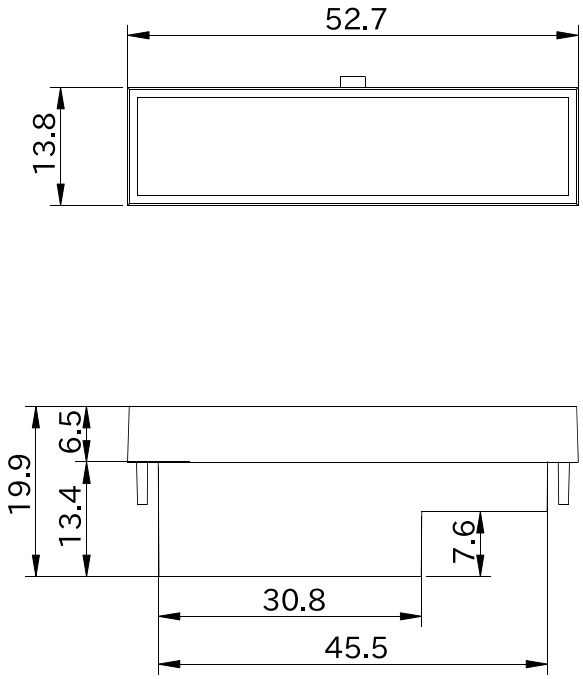


図 6.291 形状図 カバーパーツ B (アンテナ穴無し)

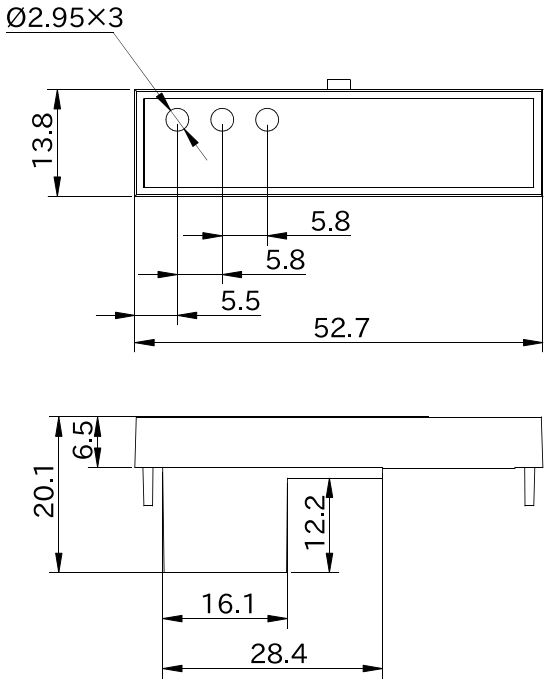


図 6.292 形状図 カバーパーツ C

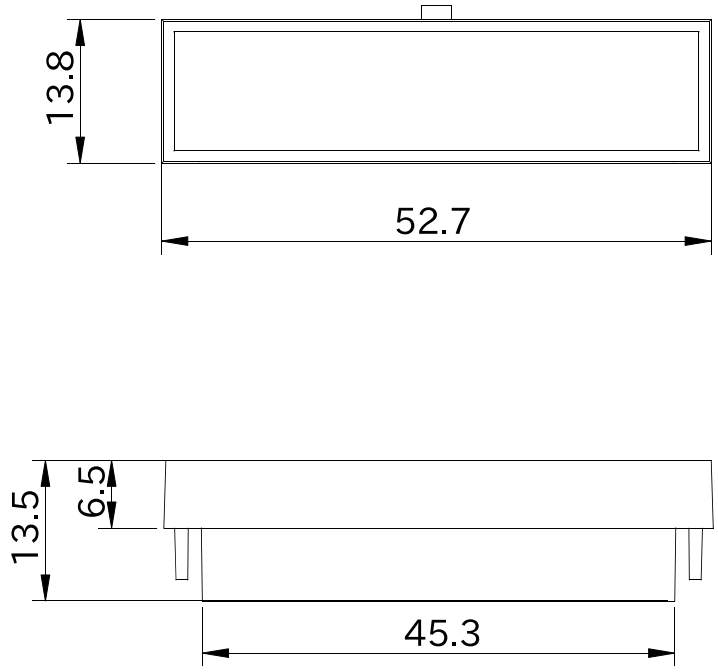


図 6.293 形状図 カバーパーツ D

6.38.2. +Di8+Ai4 拡張基板

+Di8+Ai4 拡張基板は Armadillo-IoT ゲートウェイ A6E に接続することで、アナログ入力ポートと接点入力ポートを増設できる拡張基板です。

+Di8+Ai4 拡張基板の仕様は次の通りです。

表 6.62 +Di8+Ai4 拡張基板の仕様

接点入力	**入力点数**		8 点
	定格入力電圧		DC 8~26.4 V
	入力インピーダンス		4.7 kΩ
	入力 ON 電流		2.0 mA 以上
	入力 OFF 電流		0.2 mA 以下
	絶縁耐圧		2kV
外部電源制御出力	**定格電圧**		最大 48 V
	定格電流		最大 500 mA
	出力形式		無極性
	絶縁耐圧		2kV
アナログ入力	**入力点数**		4 点
	定格入力電圧		DC 0.1~5.1 V
	定格入力電流		0.5mA～20.4mA
	入力インピーダンス	**電圧入力**	1MΩ
		電流	0.2 mA 以下



+Di8+Ai4 拡張基板をカスケード接続した場合、2 枚目以降の+Di8+Ai4 拡張基板の外部電源制御出力は 1 枚目の外部電源制御出力と同じ挙動をするためご注意ください。

6.38.2.1. ブロック図

+Di8+Ai4 拡張基板のブロック図は次の通りです。

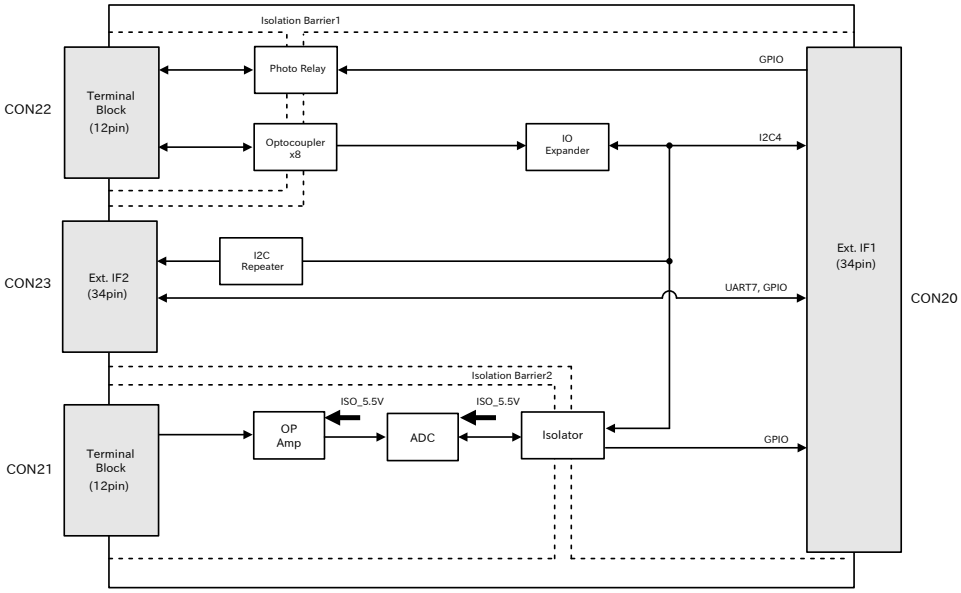


図 6.294 +Di8+Ai4 拡張基板のブロック図

6.38.2.2. インターフェース仕様

+Di8+Ai4 拡張基板のインターフェース仕様について説明します。

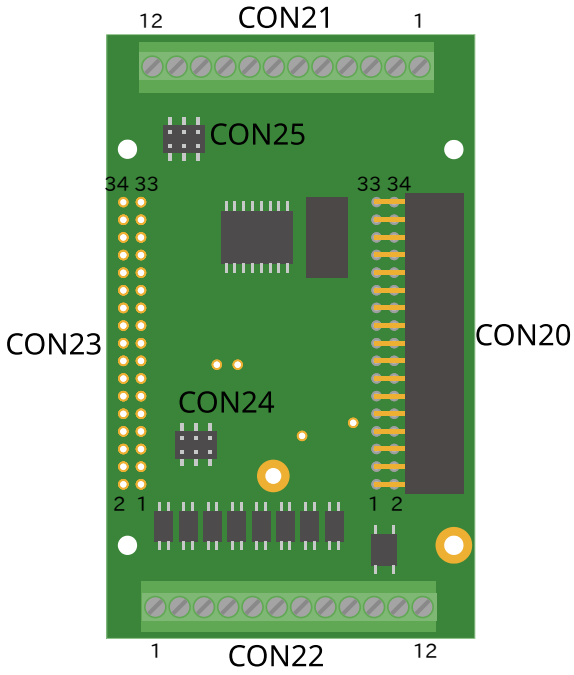


図 6.295 +Di8+Ai4 拡張基板のインターフェースレイアウト

表 6.63 +Di8+Ai4 拡張基板のインターフェース一覧

名称	インターフェース名
CON20	拡張インターフェース 1
CON21	アナログ入力インタフェース
CON22	接点入力/外部電源制御出力インタフェース
CON23	拡張インターフェース 2
CON24	設定用ピンヘッダ 1
CON25	設定用ピンヘッダ 2

- CON20 は Armadillo-IoT ゲートウェイ A6E の拡張インターフェース、もしくは同じ+Di8+Ai4 拡張基板の CON23 との接続コネクタです。
- CON21 については「3.7.13. アナログ入力を使用する」をご覧ください。
- CON22 については「3.7.11. 接点入力を使用する」、「3.7.25. 外部電源制御出力を使用する」をご覧ください。
- CON23 は+Di8+Ai4 拡張基板に拡張基板を接続するためのインタフェースです。
- CON24/CON25 は+Di8+Ai4 拡張基板をカスケード接続した際に使用する設定用ピンヘッダです。詳細は「6.38.3.2. 各+Di8+Ai4 拡張基板に必要な設定」をご覧ください。

表 6.64 CON20 信号配列

ピン番号	信号名	I/O	説明
1	VIN	Power	Armadillo-IoT ゲートウェイ A6E の電源端子と直結
2	GND	Power	電源(GND)
3	+5V	Power	電源(5V)
4	GND	Power	電源(GND)
5	+3.3V	Power	電源(3.3V)
6	GND	Power	電源(GND)
7	VIN MONITOR	Out	入力電圧監視用 ADC 入力
8	-	-	システムで使用
9	VOUT_ONOFF	In	外部電源制御出力の制御信号
10	ADDIO	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続
11	GND	Power	電源(GND)
12	I2C4_SCL	In	i.MX6ULL の UART2_TX_DATA に接続、+Di8+Ai4 拡張基板内部の ADC、GPIO エキスパンダーとの通信に使用する i2c 信号(SCL)
13	I2C4_SDA	In/Out	i.MX6ULL の UART2_RX_DATA に接続、+Di8+Ai4 拡張基板内部の ADC、GPIO エキスパンダーとの通信に使用する i2c 信号(SDA)
14	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA00 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
15	GPIO EX INT	Out	接点入力を制御している GPIO エキスパンダーの割り込み信号出力
16	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA02 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
17	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA03 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
18	GND	Power	電源(GND)
19	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続、起動デバイス設定スイッチ(SW2)が eMMC 側に設定されている時 10kΩ プルアップ(VCC_3.3V)、SD 側に設定されている時 10kΩ プルダウンされます。
20	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA06 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルアップ(VCC_3.3V)されています。
21	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA07 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。

ピン番号	信号名	I/O	説明
22	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA08 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
23	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA09 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
24	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA10 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
25	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続、起動デバイス設定スイッチ(SW2)が eMMC 側に設定されている時 10kΩ プルダウン、SD 側に設定されている時 10kΩ プルアップ(VCC_3.3V)されます。
26	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA15 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
27	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA16 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
28	DIO	In/Out	拡張入出力、i.MX6ULL の LCD_DATA17 ピンに接続、Armadillo-IoT ゲートウェイ A6E 上で 10kΩ プルダウンされています。
29	DIO	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
30	DIO	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
31	5V_PW_EN	In	+Di8+Ai4 拡張基板内の 5V 電源の ON-OFF を制御する信号
32	ADC INT	Out	+Di8+Ai4 拡張基板内の ADC の割り込み信号
33	GND	Power	電源(GND)
34	GND	Power	電源(GND)

表 6.65 CON23 信号配列

ピン番号	信号名	I/O	説明
1	-	-	未接続
2	GND	Power	電源(GND)
3	+5V	Power	電源(5V)
4	GND	Power	電源(GND)
5	+3.3V	Power	電源(3.3V)
6	GND	Power	電源(GND)
7	-	-	未接続
8	-	-	未接続
9	VOUT_ONOFF	Out	外部電源制御出力の制御信号
10	ADDIO	In/Out	
11	GND	Power	電源(GND)
12	I2C4_SCL	Out	i2c 信号(SCL) i2c リピーター出力
13	I2C4_SDA	Out	i2c 信号(SDA) i2c リピーター出力
14	DIO	In/Out	CON20 の 14 ピンと接続
15	GPIO EX INT	Out	接点入力を制御している GPIO エキスパンダーの割り込み信号出力
16	DIO	In/Out	CON20 の 16 ピンと接続
17	DIO	In/Out	CON20 の 17 ピンと接続
18	GND	Power	電源(GND)
19	DIO	In/Out	CON20 の 19 ピンと接続
20	DIO	In/Out	CON20 の 20 ピンと接続
21	DIO	In/Out	CON20 の 21 ピンと接続
22	DIO	In/Out	CON20 の 22 ピンと接続
23	DIO	In/Out	CON20 の 23 ピンと接続
24	DIO	In/Out	CON20 の 24 ピンと接続
25	DIO	In/Out	CON20 の 25 ピンと接続
26	DIO	In/Out	CON20 の 26 ピンと接続
27	DIO	In/Out	CON20 の 27 ピンと接続
28	DIO	In/Out	CON20 の 28 ピンと接続
29	DIO	In/Out	CON20 の 29 ピンと接続
30	DIO	In/Out	CON20 の 30 ピンと接続

ピン番号	信号名	I/O	説明
31	5V_PW_EN	Out	+Di8+Ai4 拡張基板内の 5V 電源の ON-OFF を制御する信号
32	ADC INT	Out	+Di8+Ai4 拡張基板内の ADC の割り込み信号
33	GND	Power	電源(GND)
34	GND	Power	電源(GND)

6.38.2.3. 基板形状図

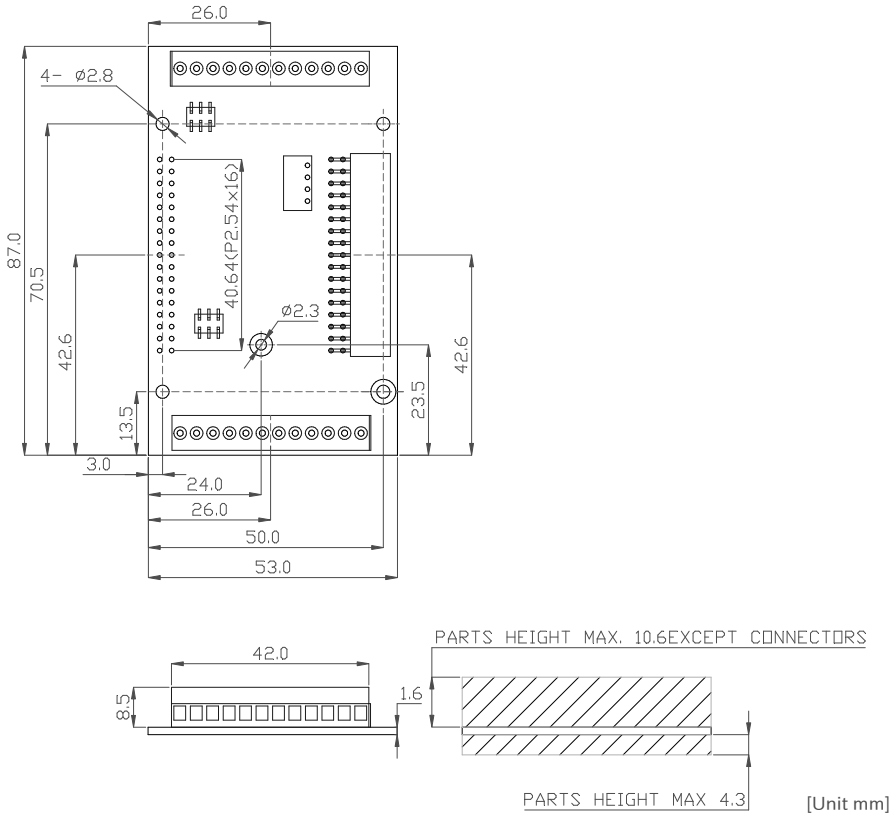


図 6.296 +Di8+Ai4 拡張基板基板形状図

6.38.3. +Di8+Ai4 拡張基板のカスケード接続

Armadillo-IoT ゲートウェイ A6E は以下の図に示すように、+Di8+Ai4 拡張基板部分をカスケード接続することで接点入力とアナログ入力ポートを増設することができます。

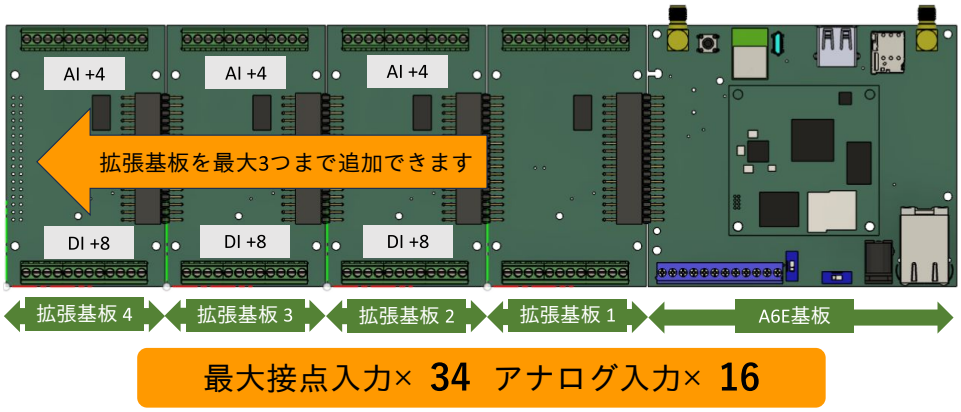


図 6.297 +Di8+Ai4 拡張基板のカスケード接続

+Di8+Ai4 拡張基板をカスケード接続した状態の本製品が入る筐体は弊社では取り扱いがございませんのでご注意ください

カスケード接続する際に、以下の PCN が適用される前の+Di8+Ai4 拡張基板を混ぜると、「6.1.2.3. アナログ入力電圧閾値設定」のアナログ入力で sleep 状態から起床する機能をご使用いただけませんのでご注意ください。

https://armadillo.atmark-techno.com/change_notification/2023-024

外部電源制御出力はカスケード接続した際に全ての+Di8+Ai4 拡張基板の外部電源制御出力が同じ動作をする設計になっております。ポートの数は増えますが、個別の制御はできませんのでご注意ください。

6.38.3.1. +Di8+Ai4 拡張基板の組付け方法

+Di8+Ai4 拡張基板をカスケード接続する場合、以下の組付け方法のイメージ図を参考に、できる限り平行に 2 枚の基板を組み付けてください。

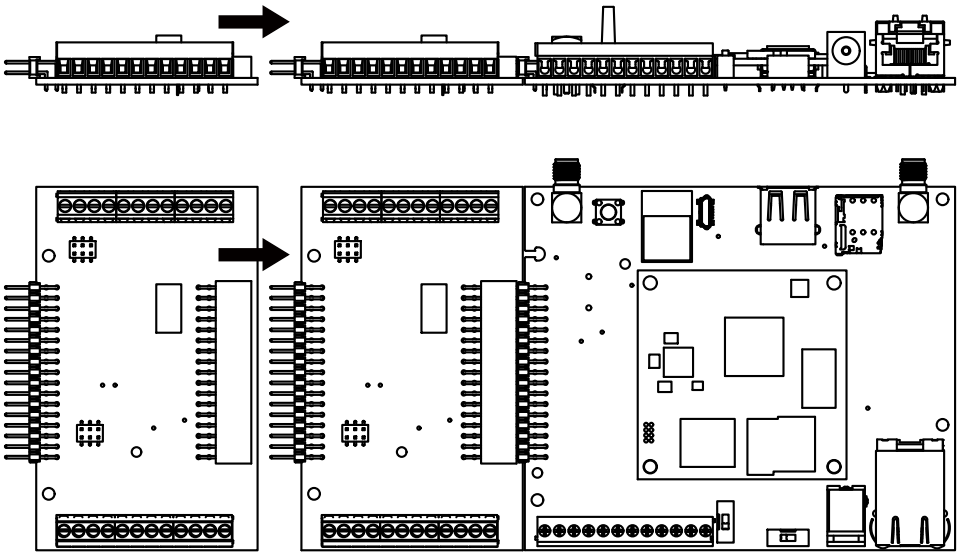


図 6.298 +Di8+Ai4 拡張基板の組付け方法

下図のように両基板を傾けた状態で接続すると接触不良やコネクタの破損につながる可能性がありますのでご注意ください。

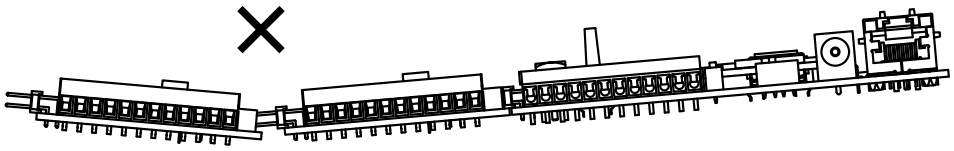


図 6.299 悪い組付け例

両基板は、ピンヘッダとピンソケットとの嵌合のみで接続されます。基板を曲げるなど無理な力を加えると接触不良やコネクタの破損につながる可能性がありますので、ご使用の際には、他の基板やケース等に固定して使用することを推奨します。

6.38.3.2. 各+Di8+Ai4 拡張基板に必要な設定

カスケード接続する場合、以下の図中に赤枠で示すピンヘッダで各+Di8+Ai4 拡張基板の設定を行う必要があります。ピンヘッダはそれぞれの+Di8+Ai4 拡張基板に 2 箇所あり、両方に同じ設定を行う必要があります。

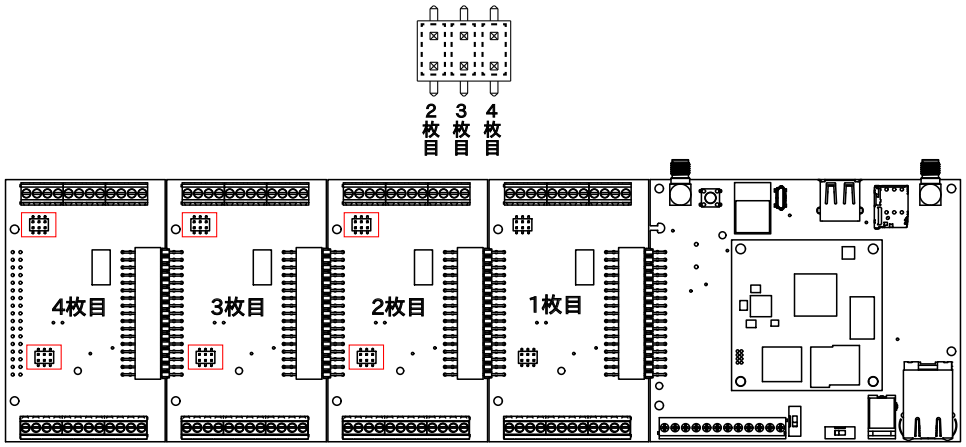


図 6.300 ピンヘッダの位置と設定

本マニュアルで+Di8+Ai4 拡張基板は Armadillo-IoT ゲートウェイ A6E 本体に近い側から 1 枚目、2 枚目、3 枚目、4 枚目と定義しており、それぞれでショートさせるべき列が指定されています。+Di8+Ai4 拡張基板の順番とショートさせる列の関係を以下の表に示します。

表 6.66 +Di8+Ai4 拡張基板のピンヘッダ設定

順番	ショートさせる列
1 枚目	ショートしない
2 枚目	左から 1 列目
3 枚目	左から 2 列目
4 枚目	左から 3 列目



ショートさせるピンヘッダの列を間違えた場合、ソフトウェアから見たインターフェースの順番が変わってしまい、誤動作の原因になりますので、ショートさせる列を間違わないようご注意ください

6.38.3.3. 接点入力の使用方法

使用方法は 「3.7.11. 接点入力を使用する」 と同様ですが、どの +Di8+Ai4 拡張基板（何枚目）に属する接点入力かによって 「表 3.35. 接点入力に対応する GPIO 番号」 の gpiochip に続く数字が異なります。

1 枚目の +Di8+Ai4 拡張基板 では gpiochip6 です。以降は 1 枚増えるごとに番号が 1 ずつ加算されるため、N 枚目では gpiochip(5+N) となります。この対応関係を 「表 6.67. +Di8+Ai4 拡張基板の順番と gpiochip に続く数字の対応関係」 に示します。

表 6.67 +Di8+Ai4 拡張基板の順番と gpiochip に続く数字の対応関係

順番	gpiochipX
1 枚目	gpiochip6
2 枚目	gpiochip7
3 枚目	gpiochip8
4 枚目	gpiochip9



ABOS 3.22.2-at.6 以前のバージョンではこの対応関係は上表と異なります。旧対応については「Armadillo-IoT ゲートウェイ A6E 製品マニュアル v3.5.0」を参照してください。

なお +Di8+Ai4 拡張基板 を 1 枚のみ接続している場合については、ABOS のバージョンに関わらず、1 枚目は共通して gpiochip6 です。

6.38.3.4. アナログ入力の使用方法

使用方法は「3.7.13. アナログ入力を使用する」の内容と同じですが、何枚目の+Di8+Ai4 拡張基板のアナログ入力なのかによって、/sys/bus/iio/devices/iio:deviceX/name の"X"の値が変わります。

表 6.68 "deviceX"と+Di8+Ai4 拡張基板の順番の関係

順番	deviceX
1 枚目	device1
2 枚目	device2
3 枚目	device3
4 枚目	device4

Armadillo-IoT ゲートウェイ A6E 製品マニュアル

改訂履歴

バージョン	年月日	改訂内容
3.0.0	2025/05/29	・初版発行
3.1.0	2025/06/25	・「3.16. ゲートウェイコンテナアプリケーションの開発」 の節の GW コンテナをインストールする方法を SWU イメージを使用する説明に変更 ・GW コンテナがプリインストールされていることを前提とした記述を修正 ・「6.37. セキュリティ」 を追加 ・「6.12.6.14. Rest API : 電源制御」 に sleep と inhibit_sleep に関する説明を追記 ・「3.12. ABOS Web から Armadillo の電源を操作する」 の節を記載 ・「6.29. Device Tree をカスタマイズする」 の節に GPIO の機能が自動的に割り当てられるのを防ぐ方法を記載 ・「3.7.8. USB デバイスを使用する」 の節に Type-A コネクタの電源を制御する方法を記載 ・「6.1.6. スリープ動作の禁止と禁止解除」 の節を記載 ・その他軽微な修正
3.2.0	2025/07/30	・「3.1.4.1. 初期化インストールディスクの作成」 で紹介する手順を Windows での手順に限定。従来の ATDE 及び Linux での作成方法は 「6.34. ATDE・Linux でインストールディスクを作成する」 を参照してください。 ・「3.1.7. シリアルコンソールを使用する」 で紹介するツールを Tera Term に変更。従来の minicom を用いたシリアルコンソールの操作方法については 「6.35. シリアル通信ソフトウェア (minicom)」 を参照してください。 ・「6.12.11.2. コンテナ以外のアクセスを制限」 を追加 ・「3.18.3.7. 実行ファイルのビルド」 を追加 ・「3.17.3.2. ディレクトリ構成」 に resources_python と python_launch に関する説明を追記 ・「3.9.6. ABOS Web のパスワード変更」に パスワードリセットに関する説明を追記 ・「3.11. ABOS Web をカスタマイズする」 に 製品型番のカスタマイズに関する説明を追記 ・「6.32.6. ABOS Web でログを確認する」 を追加 ・「6.37.6. コンテナの最小化」 を追加 ・誤記修正 ・その他軽微な修正
3.3.0	2025/08/28	・「6.16.5. LTE」に、SIMCom 製 LTE 通信モジュール SIM7672G で KDDI の相互認証試験が完了した旨を記載 ・その他軽微な修正
3.4.0	2025/09/24	・「3.7.10. GPIO を使用する」 において、 sysfs GPIO クラスの使用が非推奨になったのに伴って当該の記載を削除 ・「6.12.11.2. コンテナ以外のアクセスを制限」に command_args に渡すコマンド引数の説明を追加 ・「6.12.6.9. Rest API : WWAN の設定」 に 「LTE モジュールの再起動」 を追加 ・「6.12.4. SWU インストール」 に ABOS を最新バージョンに更新する機能の説明を追加 ・「3.10. USB デバイスの接続を許可する」 に 標準出荷状態で許可されている USB デバイスについての説明を追加

Armadillo-IoT ゲートウェイ A6E 製品マニュアル

		・軽微な修正
3.5.0	2025/10/29	・ABOS Web の UI が変更されたことによる修正 ・「6.10.9.2. クラウドからのデバイス制御」 を追加 ・「表　3.76.　[RS485_Data1，RS485_Data2，RS485_Data3，RS485_Data4] 設定可能パラメータ」 、「表 6.22. RS-485 レジスタ読み出し設定値」 及び「表 6.24. RS-485 レジスタ書き込みシャドウ設定値」 内の誤記を修正 ・「3.9.7.1. WWAN 再接続機能設定」 を追加 ・「表　6.10.　add_armadillo_env で追加される環境変数」 に AT_MODEL_NAME を追加 ・軽微な修正
3.6.0	2025/11/26	・「6.38.3.3. 接点入力の使用方法」 において、 +Di8+Ai4 拡張基板をカスケード接続した場合の基板と gpiochip 番号の対応関係が仕様変更されたことに関連した修正 ・「表 1.4. 無線 LAN 機能の利用条件」 を追加 ・「6.16.7.2. hostapd を設定する」 内に、 hostapd で ACS を使用する際のチャンネル選択に関する警告を追加 ・軽微な修正
3.7.0	2025/12/24	・「6.33. ATDE のバックアップと移行」 を追加 ・軽微な修正
3.8.0	2026/01/28	・「6.10.6. 接続先の クラウド 環境を構築 (SORACOM)」 や、「6.10.8.4. SORACOM 上でのデータ確認」など、SORACOM のクラウドへの接続方法に関する説明を追加 ・「3.9.11.3. IP アドレスフィルタリング設定」 を追加 ・「3.8.4. ログの設計」 内で、ユーザーアプリケーションがログを残す方法の説明を追加 ・「6.9.10. 異常検知」 内で、「ソフトウェアウォッチドッグタイマー」と説明していた箇所が、正しくは「ハードウェアウォッチドッグタイマー」であったため修正するとともに、より詳細な説明を追加 ・SIM7672JP に関する記述を追加 ・軽微な修正

Armadillo-IoT ゲートウェイ A6E 製品マニュアル
Version 3.8.0
2026/01/28