Электроника

Виктор Петин



Arduino и Raspberry Pi в проектах Internet of Things

- исходные коды скетчей Arduino из книги
- исходные коды библиотек Arduino
- исходные коды python-скриптов для проектов с Raspberry Pi

+ Материалы на www.bhv.ru

Интернет вещей — будущее, которое уже наступило. Не опоздай!

В. Петин
Arduino и Raspberry Pi в проектах Internet of Things

УДК 004.4
ББК 32.973.26-018.2
П29

Петин В. А.

П29 Arduino и Raspberry Pi в проектах Internet of Things. — СПб.: БХВ-Петербург, 2016. — 320 с.: ил. — (Электроника)

ISBN 978-5-9775-3646-2

Рассмотрено создание простых устройств в рамках концепции Интернета вещей (IoT, Internet of Things) на базе популярной платформы Arduino и микрокомпьютера Raspberry Pi. Показана установка и настройка среды разработки приложений Arduino IDE, а также среда макетирования Frizing. Описаны технические возможности, особенности подключения и взаимодействия различных датчиков и исполнительных устройств. Показана организация доступа разрабатываемых проектов к сети Интернет, отправка и получение ими данных с использованием популярных облачных IoT сервисов: Narodmon, ThingSpeak, Xively, Weaved, Blynk, Wyliodrin и др. Уделено внимание обмену данными с помощью платы GPRS/GSM Shield. Рассмотрен проект создания собственного сервера для сбора по сети данных с различных устройств на платформе Arduino. Показано как использовать фреймворк WebIOPi для работы с Raspberry Pi. Приведены примеры использования Wi-Fi-модуля ESP8266 в проектах «Умный дом». На сайте издательства размещен архив с исходными кодами программ и библиотек.

Для интересующихся современной электроникой

УДК 004.4
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Марины Дамбиевой*

Подписано в печать 12.01.16.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 25,8.
Тираж 1200 экз. Заказ № 1237.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3646-2

© Петин В. А., 2016
© Оформление, издательство "БХВ-Петербург", 2016

Оглавление

Глава 1. Интернет вещей (вместо введения) 7

Глава 2. Среда программирования Arduino IDE 9
2.1. Установка Arduino IDE 10
2.1.1. В ОС Windows 10
2.1.2. В ОС Linux 12
2.1.3. В Mac OS X 13
2.2. Настройка Arduino IDE 13

Глава 3. Среда разработки Fritzing 17
3.1. Загрузка и установка среды Fritzing 17
3.2. Главное окно среды Fritzing 17
3.3. Создание схемы соединений 20
3.4. Создание принципиальной схемы 21
3.5. Добавление компонентов в среду Fritzing 22

Глава 4. Arduino и аналоговые датчики 25
4.1. Аналоговые датчики (сенсоры) 25
4.2. Arduino и датчик температуры LM335 27
4.3. Arduino, Ethernet Shield/W5100 и облачные сервисы 29
4.3.1. Отправка данных на сайт «Народный мониторинг» 33
4.3.2. Чтение данных с фоторезистора 39
4.3.3. Отправка данных в сервис ThingSpeak 41
4.4. Arduino и инфракрасные датчики расстояния SHARP 49
4.4.1. Подключение датчиков Sharp к Arduino 51
4.4.2. Подсчет количества посетителей магазина 53
4.4.3. Приложение ThingTweet сервиса ThingSpeak 55
4.4.4. Отправка данных о количестве посетителей в Twitter из Arduino 57

Глава 5. Использование Arduino в качестве контроллера исполнительных устройств 63
5.1. Arduino и электромагнитное реле 63
5.1.1. Электромагнитное реле 63
5.1.2. Устройство и принцип работы электромагнитного реле 64
5.1.3. Подключение реле к Arduino 65

5.2. Arduino и твердотельное реле 67
5.3. Arduino и диммер 68
5.3.1. Диммер 68
5.3.2. Подключение диммера к Arduino 69
5.3.3. Скетч управления диммером 70
5.4. Arduino и сервоприводы 72
5.4.1. Принципы управления сервоприводами 73
5.4.2. Управление сервоприводом с помощью Arduino 75
5.5. Arduino и библиотека TinyWebServer 77
5.5.1. Использование файлов с SD-карты для формирования веб-страниц 78
5.5.2. Включение/выключение реле с веб-страницы 79
5.5.3. Веб-страница для управления реле 80
5.5.4. Веб-страница для управления сервоприводом 85

Глава 6. Arduino и устройства I²C 89
6.1. Обзор протокола I²C 89
6.2. Arduino и библиотека Wire 93
6.3. Arduino и датчик освещенности BH1750 на шине I²C 96
6.4. Arduino и сервис Xively 99
6.4.1. Отправка данных в сервис Xively 102
6.4.2. Получение данных из сервиса Xively 105
6.5. Arduino и датчик влажности и температуры SHT21 на шине I²C 107
6.6. Arduino и сервис Xively (продолжение) 110
6.6.1. Отправка мультиданных в сервис Xively 110
6.6.2. Получение мультиданных из сервиса Xively 113
6.7. Arduino и часы реального времени на шине I²C 116
6.8. Arduino и SD-карта: чтение и запись данных 121

Глава 7. Arduino и 1-Wire 125
7.1. Технология 1-Wire 125
7.2. Применение 1-Wire 128
7.3. Интерфейс 1-Wire 129
7.3.1. Обмен информацией по шине 1-Wire 130
7.3.2. Протокол обмена информацией 1-Wire 133
7.4. Arduino и цифровой датчик температуры DS18B20 135
7.4.1. Цифровой датчик температуры DS18B20 135
7.4.2. Использование библиотеки OneWire для получения данных температуры с датчика DS18B20 138

Глава 8. Сервер для сбора данных с Ethernet-модулей датчиков, установленных на Arduino 141
8.1. Датчики влажности DHT11 и DHT22 141
8.1.1. Подключение датчиков DHT к Arduino 143
8.1.2. Библиотека DHT 143
8.2. Модуль датчика движения HC-SR501 145
8.3. Модуль датчика звука FC-04 148
8.4. Ethernet-модуль датчиков на Arduino 149
8.5. Сервер сбора данных 153

Глава 9. Обмен данными с помощью платы GPRS/GSM Shield 157
9.1. Отправка и получение SMS-сообщений 159
9.2. Отправка данных на сайт «Народный мониторинг» 162

Глава 10. Проект Blynk: управление Arduino с планшета 169
10.1. Начало работы: тестовый пример 170
10.2. Управление с планшета исполнительными устройствами, подключенными к Arduino 178
10.3. Отправка данных из Arduino на экран планшета 181

Глава 11. Микрокомпьютер Raspberry Pi 187
11.1. Технические характеристики и возможности Raspberry Pi 188
11.2. Установка операционной системы 191
11.3. Первоначальная настройка ОС Raspbian 194
11.3.1. Меню конфигурации 194
11.3.2. Настройка сетевых параметров 196
11.3.3. Настройка доступа по Wi-Fi 196
11.3.4. Подключение 3G-модема 199
11.4. Интерфейс GPIO 203
11.4.1. Управление GPIO из оболочки bash 206
11.4.2. Управление GPIO командами языка Python 206
11.5. Raspberry Pi и датчик температуры DS18B20 на шине 1-Wire 209
11.5.1. Подключение датчика DS18B20 к Raspberry Pi 209
11.5.2. Отправка данных с датчика DA18B20 в сервис «Народный мониторинг» 212
11.6. Raspberry Pi и датчик освещенности BH1750 на шине I^2C 215
11.6.1. Подключение датчика BH1750 к Raspberry Pi 215
11.6.2. Получение на Raspberry Pi данных с датчика BH1750 217

Глава 12. WebIOPi — веб-интерфейс и облако для Raspberry Pi 219
12.1. Установка WebIOPi на ОС Raspbian 219
12.2. Задание пользовательского пароля WebIOPi 221
12.3. Настройка сервера WebIOPi 222
12.4. Javascript-библиотека webiopi.js 223
12.4.1. Функции библиотеки webiopi.js 224
12.5. Проект управления веб-камерой на сервоприводах 230
12.6. WebIOPi — подключение устройств 236
12.7. Доступ к устройству из сервиса Weaved 238
12.7.1. Установка сервиса Weaved 238
12.7.2. Подключение к Raspberry Pi в сервисе Weaved 242

Глава 13. Проект Wyliodrin: управление удаленными устройствами из браузера 245
13.1. Добавление устройства в профиль 246
13.2. Запись образа Wyliodrin на SD-карту 247
13.2.1. ...в ОС Windows 248
13.2.2. ...в ОС Linux 249
13.2.3. ...в Mac OS 250
13.2.4. ...в ОС Raspbian 250
13.3. Запись на SD-карту настроек Wyliodrin 250

13.4. Подключение Raspberry Pi к Wyliodrin 251
13.5. Создание приложения в графической среде программирования 253
13.6. Включение/выключение светодиода с веб-страницы 257
13.7. Подключение платы Arduino к сервису Wyliodrin 259
13.7.1. ...с помощью библиотеки Firmata 259
13.7.2. ...без использования библиотеки Firmata 266
13.8. Совместная работа Raspberry Pi и платы GrovePi 270
13.9. Обмен сообщениями между платами Raspberry Pi через сервис Wyliodrin 274
13.10. Отправка данных в сервис Wyliodrin с мобильного устройства 277

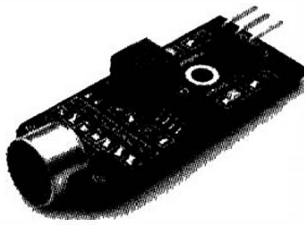
Глава 14. Wi-Fi модуль ESP8266 283
14.1. Режим AT-команд 284
14.2. Прошивка NodeMCU 289
14.2.1. Запуск веб-сервера 291
14.2.2. Подключение к ESP8266 модулей датчиков средствами языка Lua 292
14.3. Проект Home`s Smart 294
14.3.1. Прошивка и первоначальная настройка модуля ESP8266 296
14.3.2. Обновление прошивки через Интернет 300
14.3.3. Подключение датчиков к модулю ESP8266 300
Подключение датчика температуры DS18B20 300
Подключение датчика влажности DHT11 (DHT22) 302
Подключение I^2C-датчика освещенности BH1750 302
14.4. Отправка данных с модуля ESP8266 на сайт «Народный мониторинг» 306
14.5. Отправка данных с модуля ESP8266 в сервис ThingSpeak 307
14.6. Подключение дисплея WH1602 к плате модуля ESP8266 310
14.7. Управление выводами GPIO модуля ESP8266 311
14.8. Работа с прерываниями модуля ESP8266 312
14.9. Управление каналами ШИМ модуля ESP8266 313
14.10. Планировщик задач модуля ESP8266 314

Заключение 315

Приложение. Описание электронного архива 316

Предметный указатель 317

ГЛАВА 1



Интернет вещей (вместо введения)

Internet of Things (IoT, Интернет вещей) — это концепция «умного дома», где все (или многие) бытовые приборы и системы управляются через Интернет.

Идея Интернета вещей впервые возникла еще в 1999 году у Кевина Эштона — исследователя из Массачусетского технологического института (MIT), предложившего тогда концепцию системы управления через Интернет промышленными объектами. Интернет вещей предполагает оснащение каждого устройства, будь то пылесос, холодильник или стиральная машина, модулем подключения к Интернету с возможностью взаимодействия его с домашним компьютером или смартфоном домовладельца.

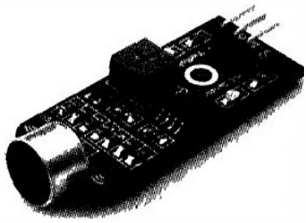
С появлением Интернета вещей автоматически решится множество самых разных проблем: от индивидуального комфорта и безопасности, когда «умный» дом будет оценивать и контролировать собственное состояние, до ликвидации пробок на дорогах, когда машины сами станут договариваться со светофорами об оптимальном трафике. Холодильники смогут следить за просроченными продуктами, лекарства подскажут время приема, портфель в дождливую погоду напомнит хозяину, что тот забыл зонтик, а автомобиль сам выдержит безопасную дистанцию в потоке машин и покажет, где и как лучше припарковаться.

Интернет вещей — это не только множество различных приборов и датчиков, объединенных между собой проводными и беспроводными каналами связи и подключенных к сети Интернет, а более тесная интеграция реального и виртуального миров, в котором общение осуществляется между людьми и устройствами.

В этой книге мы познакомимся с практическими примерами создания простейших устройств для Интернета вещей на базе популярного контроллера Arduino и микрокомпьютера Raspberry Pi. Чтобы Arduino и Raspberry Pi стали полноценными устройствами для Интернета вещей, их необходимо оснастить датчиками и исполнительными устройствами и предоставить им доступ к сети Интернет. Соответственно, мы рассмотрим работу Arduino и Raspberry Pi с различными датчиками и устройствами, а также организацию доступа их к сети с дальнейшей отправкой данных в известные облачные сервисы и получением их оттуда.

А последняя, 14-я, глава книги посвящена использованию в качестве устройства для Интернета вещей нового, но быстро набирающего популярность Wi-Fi-модуля ESP8266.

ГЛАВА 2



Среда программирования Arduino IDE

Разработка собственных приложений на базе плат, совместимых с архитектурой Arduino, осуществляется в официально бесплатной среде программирования Arduino IDE. Среда предназначена для написания, компиляции и загрузки собственных программ в память микроконтроллера, установленного на плате Arduino-совместимого устройства. Основой среды разработки является язык Processing/Wiring — это фактически обычный C++, дополненный простыми и понятными функциями для управления вводом/выводом на контактах устройства. Для операционных систем Windows, Mac OS и Linux существуют свои версии среды. Скачать среду Arduino IDE можно с ее официального сайта: **www.arduino.cc**.

Последняя версия Arduino IDE — 1.6.5 — имеет множество улучшений по сравнению с предыдущими. Вот далеко неполный их список:

- включена поддержка значительного количества платформ;
- организовано определение и отображение плат в меню списка портов вместе с последовательным портом;
- увеличена скорость компиляции;
- добавлено автосохранение при компиляции/загрузке скетча;
- в основу монитора последовательного порта положена современная библиотека JSSC (вместо старой RXTX), что дало возможность повысить его быстродействие;
- для опций **Найти/Заменить** организовано несколько вкладок;
- улучшено множество библиотек Arduino IDE (String, Serial, Print и пр.);
- обновлены инструменты и компиляторы (avr-gss, arm-gss, avrdude, bossac);
- переработан интерфейс командной строки;
- добавлен вывод информации о размере скетча и использовании памяти;
- в редакторе теперь отображаются номера строк;
- меню с большим количеством строк имеют полосы прокрутки;
- организована загрузка устройства Arduino Yun через сеть;

- улучшен класс HardwareSerial;
- увеличены стабильность и производительность USB;
- библиотека SPI теперь поддерживает транзакции для улучшения совместимости при использовании одновременно нескольких SPI-устройств;
- появилась возможность настроить подменю с конфигурациями;
- устранены проблемы загрузки на Leonardo, Micro и Yun;
- усовершенствованы библиотеки для Arduino — в частности: Bridge, TFT, Ethernet, Robot_Control, SoftwareSerial, GSM — и устранены проблемы в их работе;
- устранено множество незначительных ошибок пользовательского интерфейса.

2.1. Установка Arduino IDE

2.1.1. В ОС Windows

Отправляемся на страницу **http://arduino.cc/en/Main/Software** (рис. 2.1), выбираем версию для операционной системы Windows и скачиваем соответствующий архивный файл. Его объем составляет чуть более 80 Мбайт и содержит все необходимое, в том числе и драйверы. По окончании загрузки распаковываем скачанный файл в удобное для себя место.



Рис. 2.1. Страница загрузки официального сайта Arduino

Для установки драйверов подключаем устройство (пусть это будет Arduino Uno) к компьютеру — на контроллере должен загореться индикатор питания (зеленый светодиод). Начавшаяся тут же попытка Windows автоматически установить драйверы заканчивается сообщением: **Программное обеспечение драйвера не было установлено**.

Не беда: открываем Диспетчер устройств и в составе устройств находим значок Arduino Uno — он там помечен восклицательным знаком. Щелкаем правой кнопкой мыши на этом значке и в открывшемся окне выбираем опцию **Обновить драйверы** и далее — **Выполнить поиск драйверов на этом компьютере**. Указываем путь к драйверам — ту папку на компьютере, куда распаковывали скачанный архив, — пусть это будет папка drivers каталога установки Arduino (например, C:\arduino-1.6.5\drivers). Игнорируем все предупреждения Windows и получаем в результате сообщение: **Обновление программного обеспечения для данного устройства завершено успешно**. В заголовке окна будет указан и COM-порт, на который установлено устройство.

Осталось запустить среду разработки Arduino IDE (рис. 2.2). Как уже отмечалось ранее, в новой версии Arduino IDE в списке доступных портов отображается и название подключенной платы Arduino.

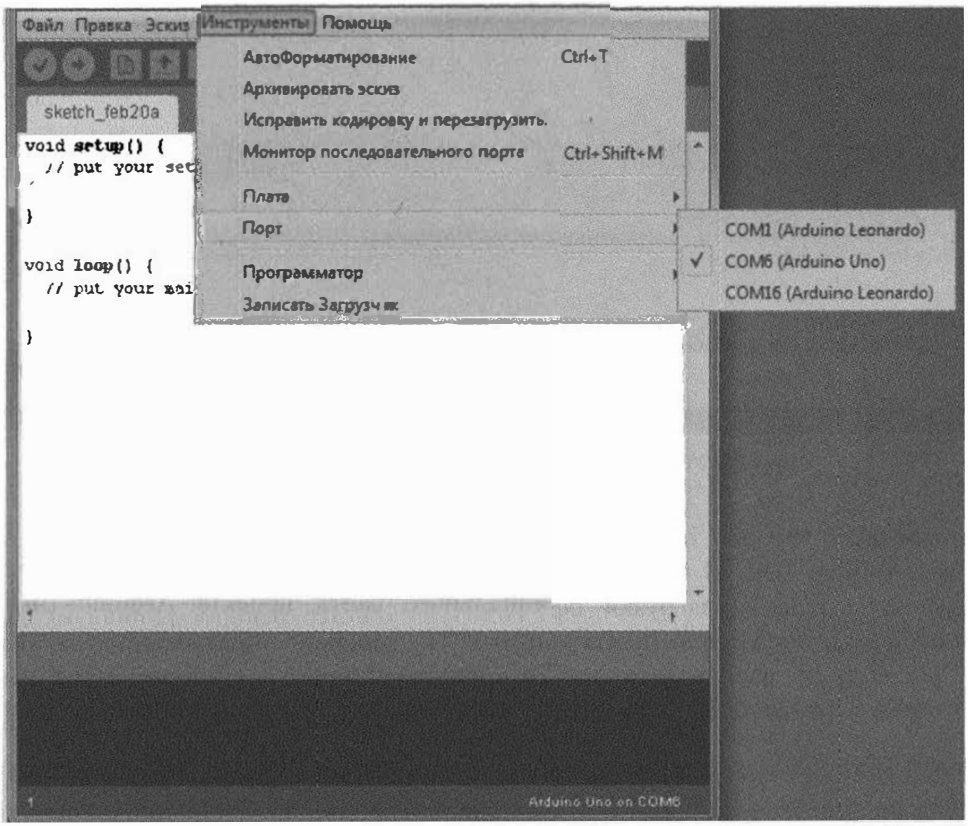


Рис. 2.2. Среда разработки Arduino IDE: выбор порта

2.1.2. В ОС Linux

В Linux Ubuntu среда Arduino IDE устанавливается еще проще, поскольку она находится в репозитории стандартных приложений Linux.

Итак, в меню Ubuntu **Приложения | Центр приложений Ubuntu | Загрузить приложение** выбираем из списка доступных программ Arduino IDE, затем в списке разделов выбираем **Инструменты разработчика**, в списке следующего уровня — **Все приложения** и в следующем открывшемся списке — **Arduino IDE**. В открывшемся окне (рис. 2.3) щелкаем левой кнопкой мыши на значке этой программы — справа от нее появляется кнопка **Установить**, нажимаем на эту кнопку, и среда устанавливается автоматически. Для запуска Arduino IDE выбираем опцию меню **Приложения | Программирование | Arduino IDE**.

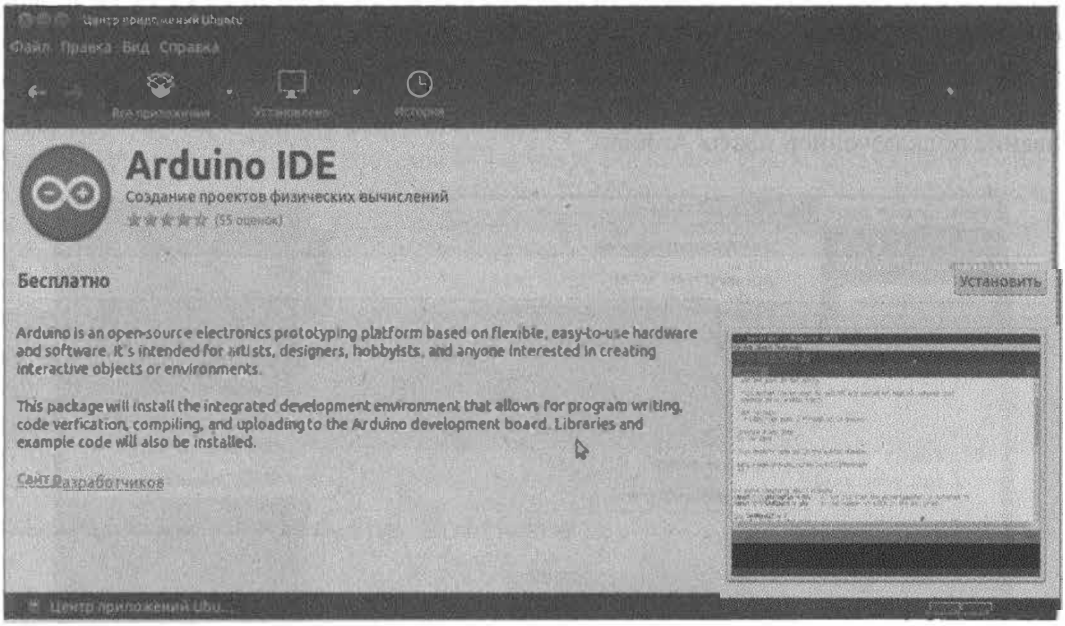


Рис. 2.3. Установка Arduino IDE из центра приложений Ubuntu

Надо заметить, что при таком способе устанавливается не последняя версия программы Arduino IDE. И чтобы работать именно с ее последней версией, нужно скачать со страницы загрузки официального сайта проекта Arduino (**https://www.arduino.cc/en/Main/Software**) архив с версией программы для Linux (см. рис. 2.1) и распаковать его в желаемое место — например, в /home/<user>/Arduino. Осталось для запуска программы выполнить из терминала команды:

```
cd ~/Arduino
./arduino
```

2.1.3. В Mac OS X

Для установки Arduino IDE в операционной системе Mac OS X, как и в предыдущих случаях, скачиваем со страницы загрузки официального сайта проекта Arduino (**https://www.arduino.cc/en/Main/Software**) архив с версией программы для OS X (см. рис. 2.1), распаковываем его и копируем содержимое архива в папку **Программы** — после чего значок **Arduino** появляется в списке программ **Launchpad** (рис. 2.4).

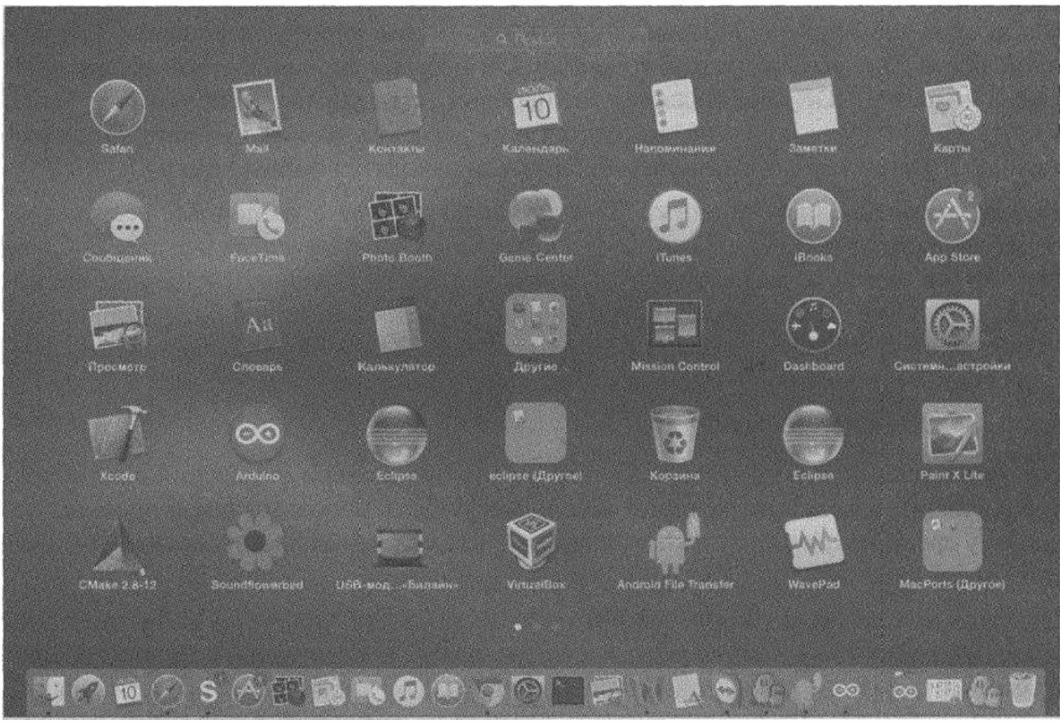


Рис. 2.4. Установка Arduino IDE в OS X

2.2. Настройка Arduino IDE

Среда разработки Arduino состоит из:

- редактора программного кода;
- области сообщений;
- окна вывода текста;
- панели инструментов с кнопками часто используемых команд;
- нескольких меню.

Программа, написанная в среде Arduino, носит название *скетч*. Скетч пишется в текстовом редакторе, который имеет цветовую подсветку создаваемого про-

граммного кода. Во время сохранения и экспорта проекта в области сообщений появляются пояснения и информация об ошибках. Окно вывода текста показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины.

Дополнительная функциональность может быть добавлена разрабатываемым скетчам с помощью *библиотек*, представляющих собой специальным образом оформленный программный код, реализующий некоторый функционал, который можно подключить к создаваемому проекту. Специализированных библиотек существует множество. Обычно библиотеки пишутся так, чтобы упростить решение той или иной задачи и скрыть от разработчика детали программно-аппаратной реализации. Среда Arduino IDE поставляется с набором стандартных библиотек: Serial, EEPROM, SPI, Wire и др. Они находятся в подкаталоге libraries каталога установки Arduino. Внутри каталога с именем библиотеки находятся файлы *.cpp и *.h.

Необходимые библиотеки могут быть также загружены с различных ресурсов — папка загруженной библиотеки просто копируется в каталог стандартных библиотек (тот же самый подкаталог libraries каталога установки Arduino). Многие библиотеки снабжаются примерами, расположенными в папке examples. Если библиотека установлена правильно, то она появляется в меню **Эскиз | Импорт библиотек**. Выбор библиотеки в меню приведет к добавлению в исходный код строчки:

```
#include <имя библиотеки.h>
```

Эта директива подключает заголовочный файл библиотеки с описанием ее объектов, функций и констант, которые теперь могут быть использованы в проекте, поскольку среда Arduino станет компилировать создаваемый проект уже вместе с указанной библиотекой.

При загрузке скетча используется загрузчик (bootloader) Arduino — небольшая программа, загружаемая в микроконтроллер на плате. Она позволяет загружать программный код без использования дополнительных аппаратных средств. Работа загрузчика распознается по миганию светодиода на цифровом выводе D13.

Перед загрузкой скетча требуется задать необходимые параметры в меню **Инструменты | Плата** (рис. 2.5) и **Инструменты | Порт** (см. рис. 2.2).

Современные платформы Arduino перед загрузкой перезагружаются автоматически. На старых же платформах для этого необходимо нажать кнопку перезагрузки. На большинстве плат во время процесса загрузки мигают светодиоды RX и TX.

Монитор последовательного порта (Serial Monitor) отображает данные, посылаемые в платформу Arduino (плату USB или плату последовательной шины). Для отправки данных необходимо ввести в соответствующее поле текст и нажать кнопку **Отправить** (Send) или клавишу <Enter> (рис. 2.6), после чего из выпадающего списка в правом нижнем углу окна монитора выбрать скорость передачи, соответствующую значению `Serial.begin` в скетче.

На Mac OC или в Linux при подключении мониторинга последовательной шины платформа Arduino будет перезагружена (скетч начнется сначала).

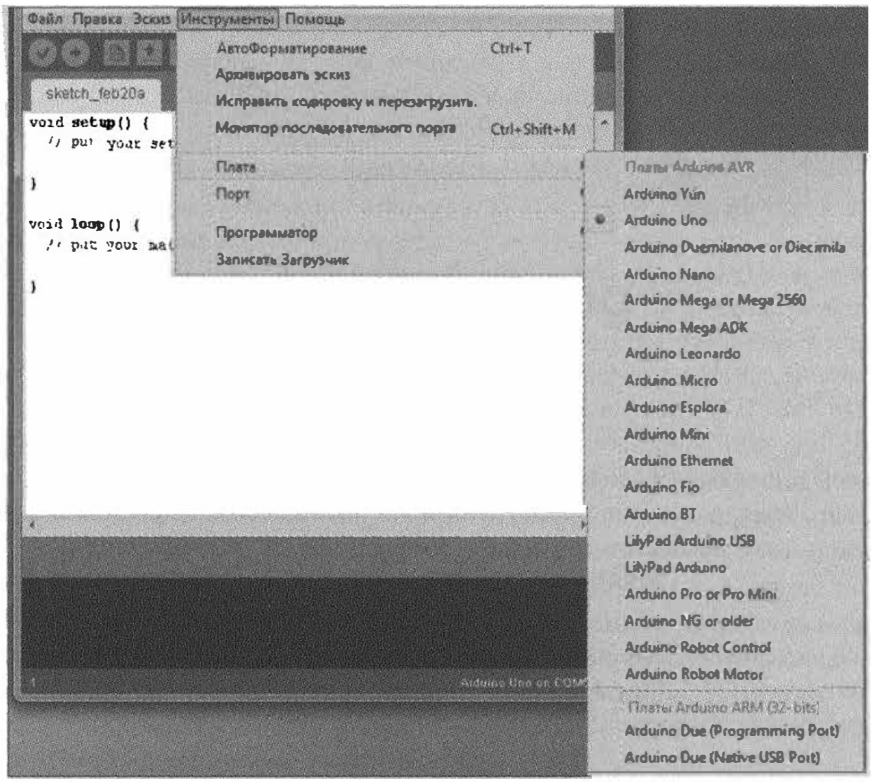


Рис. 2.5. Arduino IDE: выбор платы

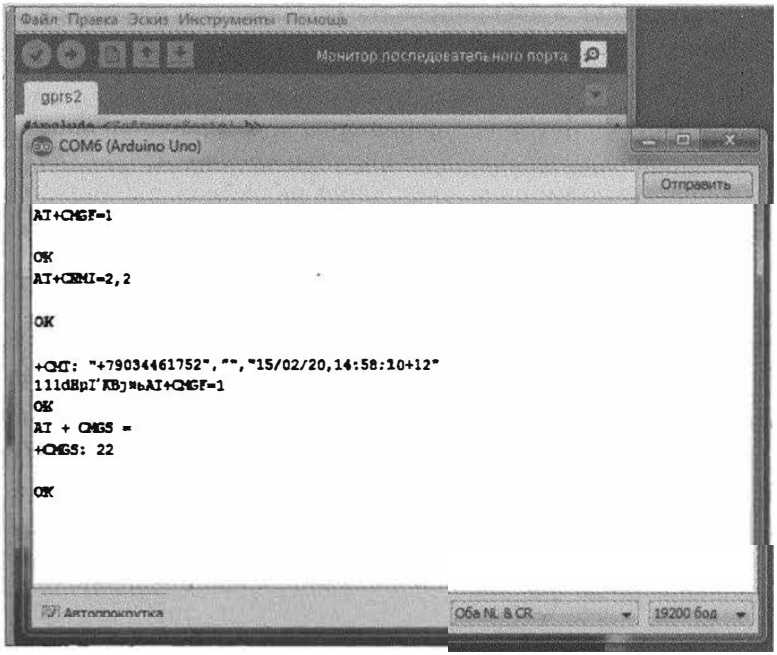
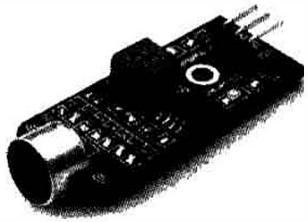


Рис. 2.6. Arduino IDE: монитор последовательного порта

ГЛАВА 3



Среда разработки Fritzing

Среда разработки Fritzing представляет собой инструмент разработчика с открытым исходным кодом, предназначенный для прототипирования и создания программного кода на базе Arduino, а также и для обучения пользователей этим процедурам. Среда работает как на Windows, так и на Mac OS и на Linux.

С помощью Fritzing можно создать принципиальную схему устройства и оформить ее представление в виде соединения макетов элементов, разработать печатную плату, написать код программы и загрузить его в плату Arduino. В отличие от других систем проектирования, у Fritzing простой интерфейс, который делает разработку электронных схем интуитивно понятной.

3.1. Загрузка и установка среды Fritzing

Для установки среды разработки Fritzing перейдите на страницу ее загрузки (**http://fritzing.org/download/?donation=0**) и выберите свою операционную систему (рис. 3.1). На момент подготовки этой книги актуальна версия среды 0.9.2b. Для установки программы на компьютер следуйте инструкциям, приведенным на странице загрузки.

Scan & DjVu
Bookingolz

3.2. Главное окно среды Fritzing

В первый раз открыв среду разработки Fritzing, вы увидите окно, изображенное на рис. 3.2.

Для разработки схемы соединения элементов разрабатываемого нами проекта переключимся на вкладку **Макетная плата** (рис. 3.3).

В правой части окна находятся панели инструментов со всеми элементами и опциями. Они содержат как основные компоненты: провода, кнопки, резисторы и пр., так и различные специализированные компоненты — например, платы Arduino и датчики.

Если компонент настраивается, то в панели инструментов **Инспектор** отображаются настраиваемые параметры для этого компонента (рис. 3.4).

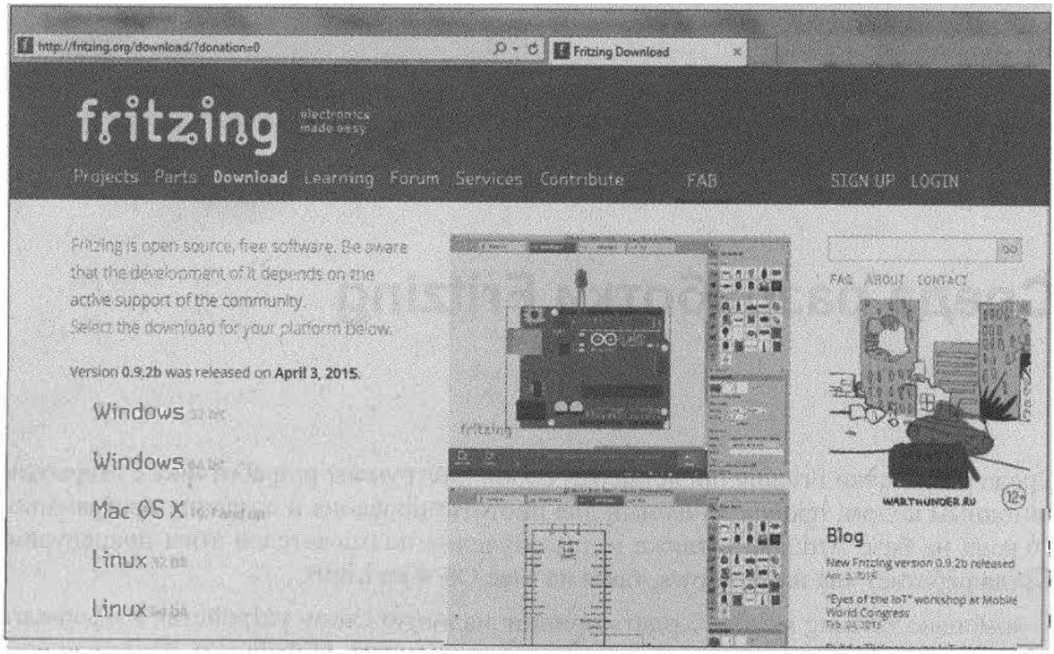


Рис. 3.1. Страница загрузки Fritzing

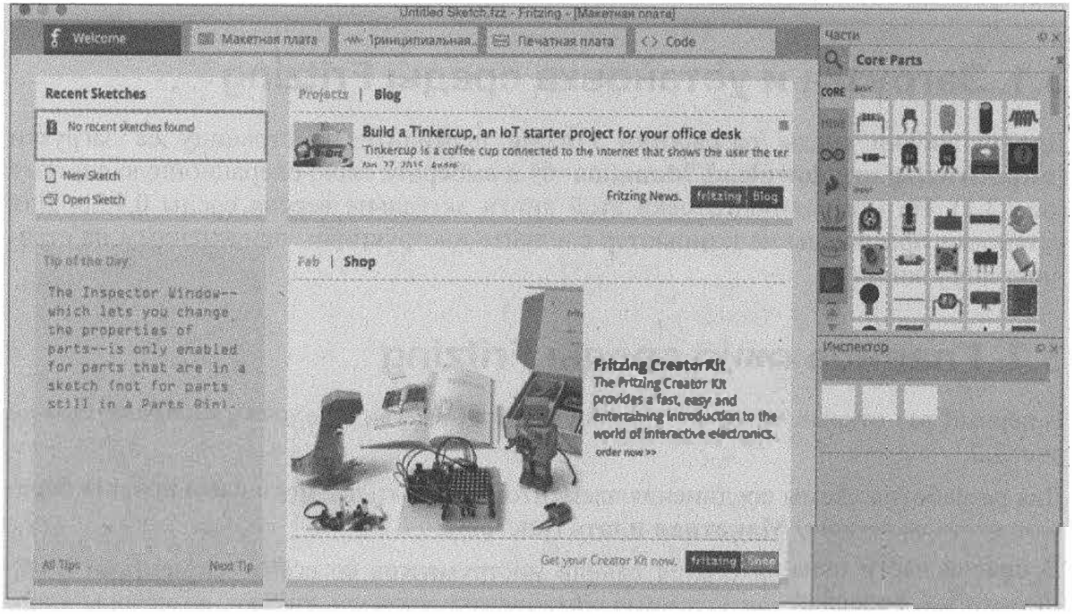


Рис. 3.2. Окно программы Fritzing

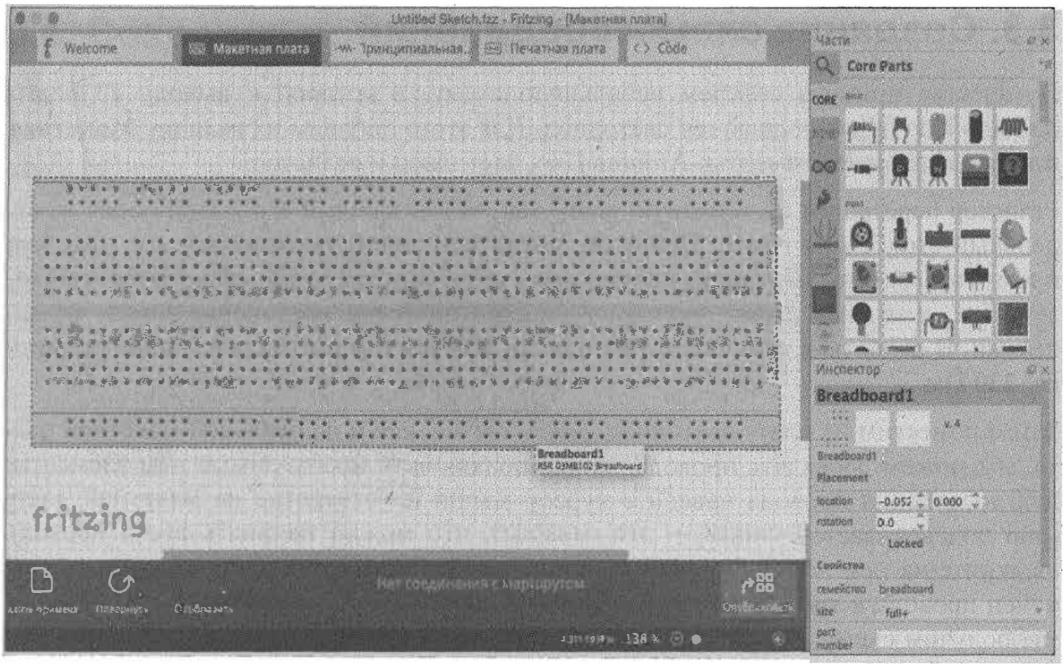


Рис. 3.3. Вкладка **Макетная плата**

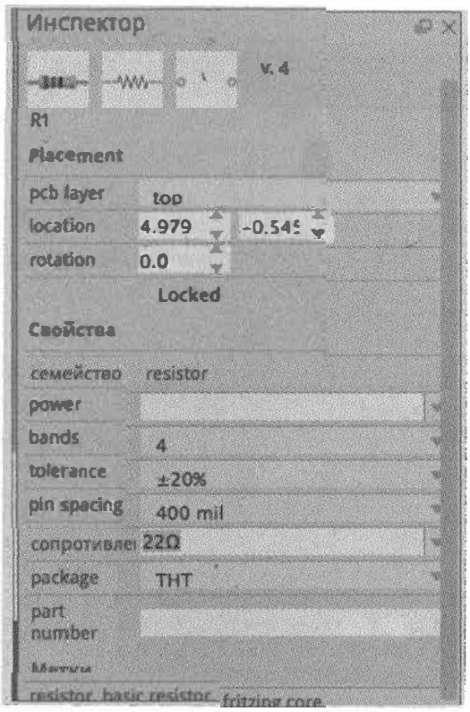


Рис. 3.4. Панель **Инспектор** для настройки параметров компонента

3.3. Создание схемы соединений

В качестве примера соберем небольшую схему, в которой к выводу 13 платы Arduino Uno будет подключен светодиод. Для этого добавим на вкладку **Макетная плата** следующие элементы: Arduino Uno, светодиод и резистор.

Резистор перетащим на макетную плату так, чтобы каждый его вывод попал на отдельный столбец на плате. При этом, когда компонент подключается к тому или иному отверстию макетной платы, весь столбец отверстий становится светло-зеленым, а зеленая линия указывает на электрическое соединение между отверстиями. В свойствах компонента на панели **Инспектор** выберем номинал резистора: **220 Ом**.

Затем поместим на плату светодиод. Здесь так же, как и на реальной макетной плате, мы можем добавлять провода, чтобы подключать необходимые нам элементы: для добавления провода наведите курсор мыши на отверстие на макетной плате (оно подсвечивается синим — это означает, что можно начинать вести провод), щелкните на нем левой кнопкой мыши и, не отпуская кнопки, перетащите второй конец провода в требуемую точку. Добавив таким образом необходимые провода, мы получим схему, представленную на рис. 3.5. Чтобы использовать ее в дальнейшем, необходимо сохранить ее в нужном формате (командой меню **Файл | Экспорт**).

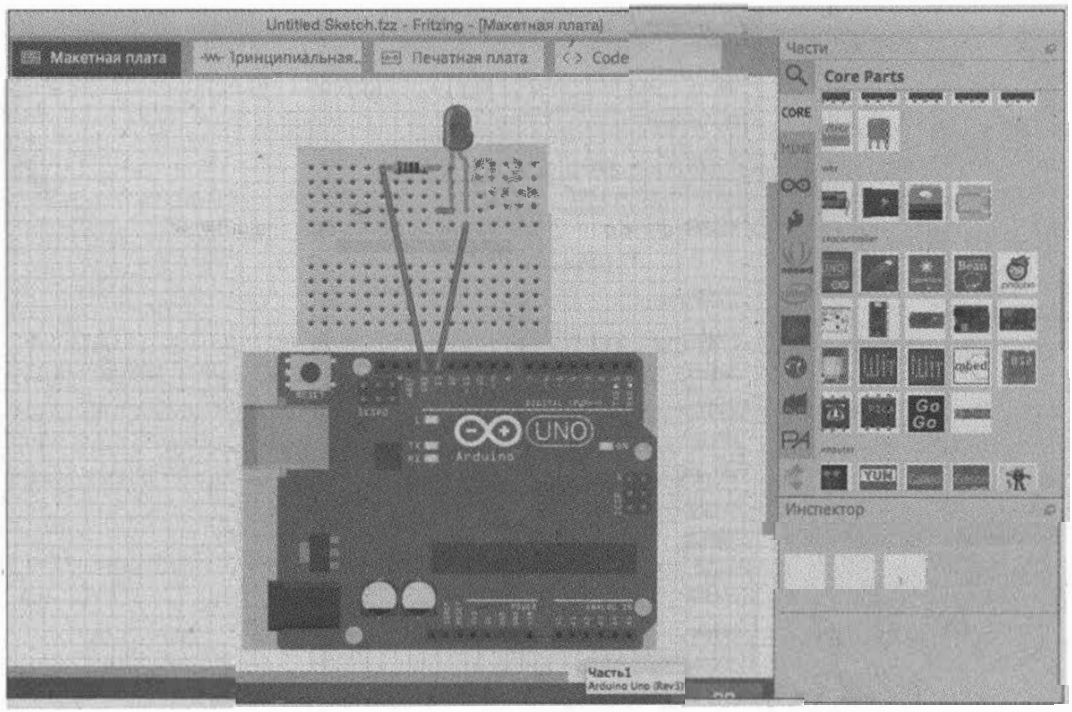


Рис. 3.5. Схема подключения светодиода к выводу 13 Arduino Uno

3.4. Создание принципиальной схемы

В предыдущем разделе мы рассмотрели создание *схемы соединений*. Теперь создадим на основе этой схемы (см. рис. 3.5) *принципиальную схему* устройства, для чего переключимся на вкладку **Принципиальная схема** (рис. 3.6).

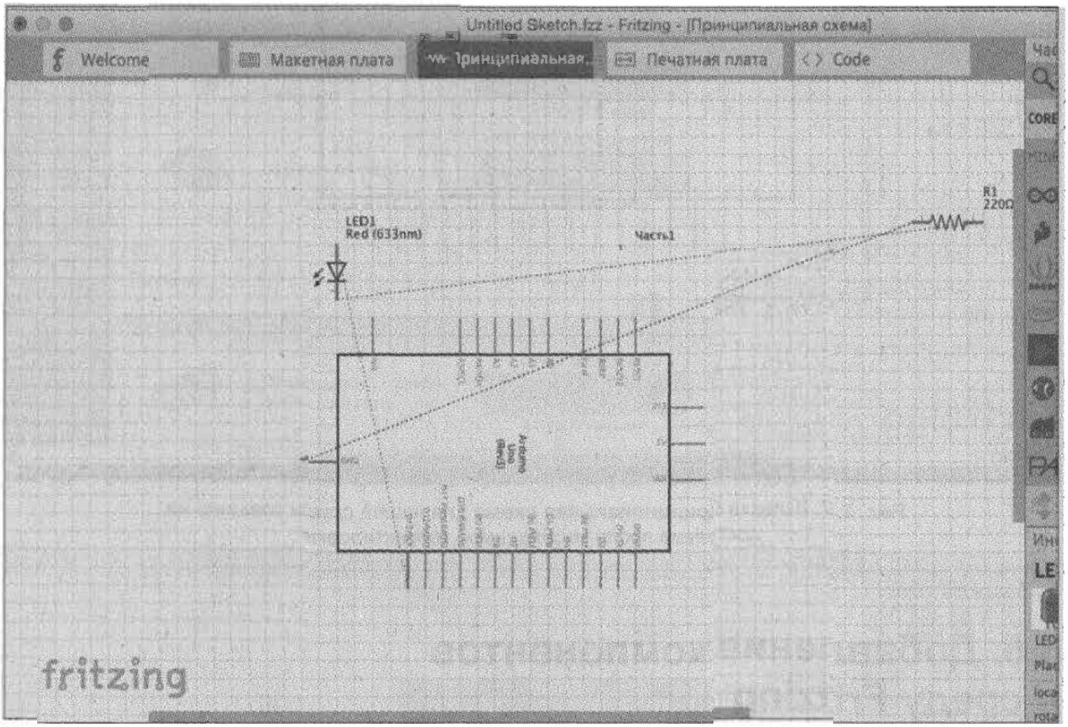


Рис. 3.6. Вкладка **Принципиальная схема** для нашей схемы соединений: исходное состояние

Как можно видеть, среда Fritzing создала все необходимые соединения, надо только привести их к более опрятному виду. Для этого в меню Fritzing выбираем команду **Вид | Подогнать окно** — чтобы автоматически сцентрировать и отмасштабировать схему на рабочем поле. Затем, перетаскивая и поворачивая компоненты, постараемся добиться того, чтобы проводники не пересекались, или же количество таких пересечений стало минимальным (рис. 3.7).

После этого в левом нижнем углу окна среды Fritzing нажимаем кнопку **Автотрассировка**, и схема приводится в порядок. Теперь ее можно сохранить для публикации в удобном формате (командой меню **Файл | Экспорт**).

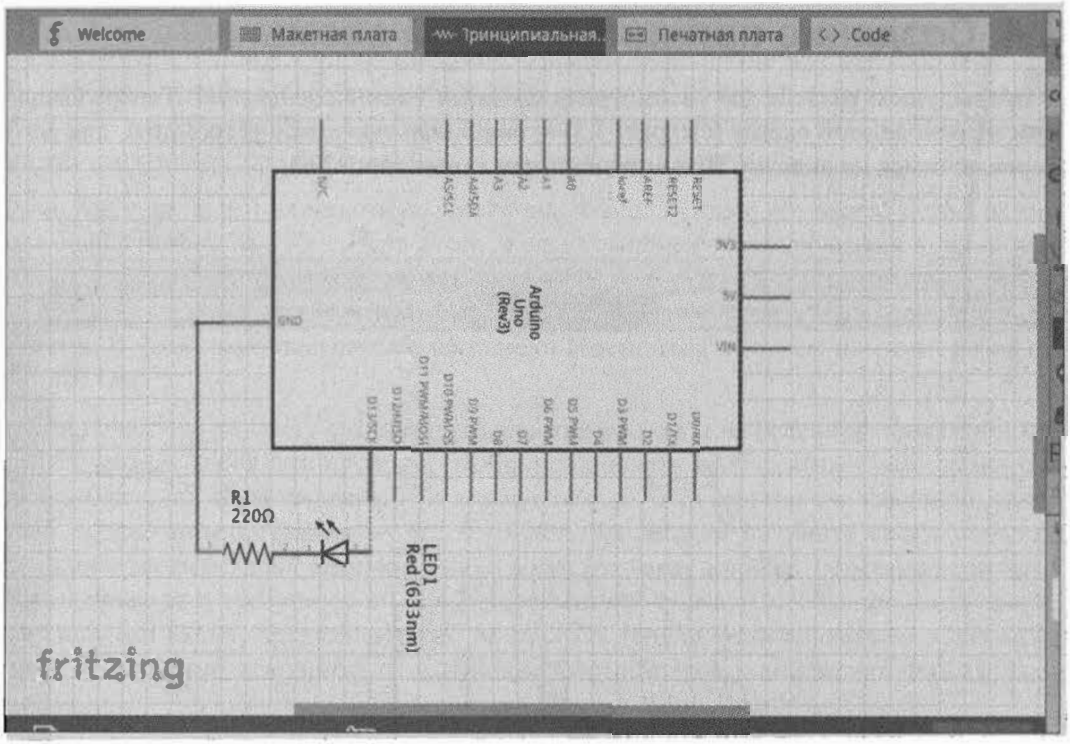


Рис. 3.7. Вкладка **Принципиальная схема** для нашей схемы соединений: состояние после необходимой корректировки

3.5. Добавление компонентов в среду Fritzing

Панели инструментов в правой части окна Fritzing содержат библиотеки уже имеющихся распространенных элементов, а также некоторые модули ряда производителей. Все это мы можем использовать сразу после установки Fritzing. Но рано или поздно возникает необходимость добавить какой-либо компонент или шилд, с которым мы планируем работать далее. Как же добавить новый компонент в библиотеку Fritzing?

Вы можете создать собственный компонент с нуля или найти готовые компоненты на просторах Интернета. Большое количество таких компонентов собрано на русскоязычном ресурсе «Роботоша» по адресу: **http://robotosha.ru/arduino/fritzing-library.html** (рис. 3.8). Скачаем оттуда какой-нибудь компонент — например, 16-кнопочную клавиатуру (файл KEYPAD **4x4.fzpz**) и добавим ее в библиотеку Fritzing.

Итак, выбираем раздел, в который мы собираемся добавить новый компонент, — пусть это будет, например: **MINE | My Parts** (Мои компоненты). Щелкаем правой кнопкой мыши, выбираем пункт **Import** и в открывшемся окне выбираем скачан-

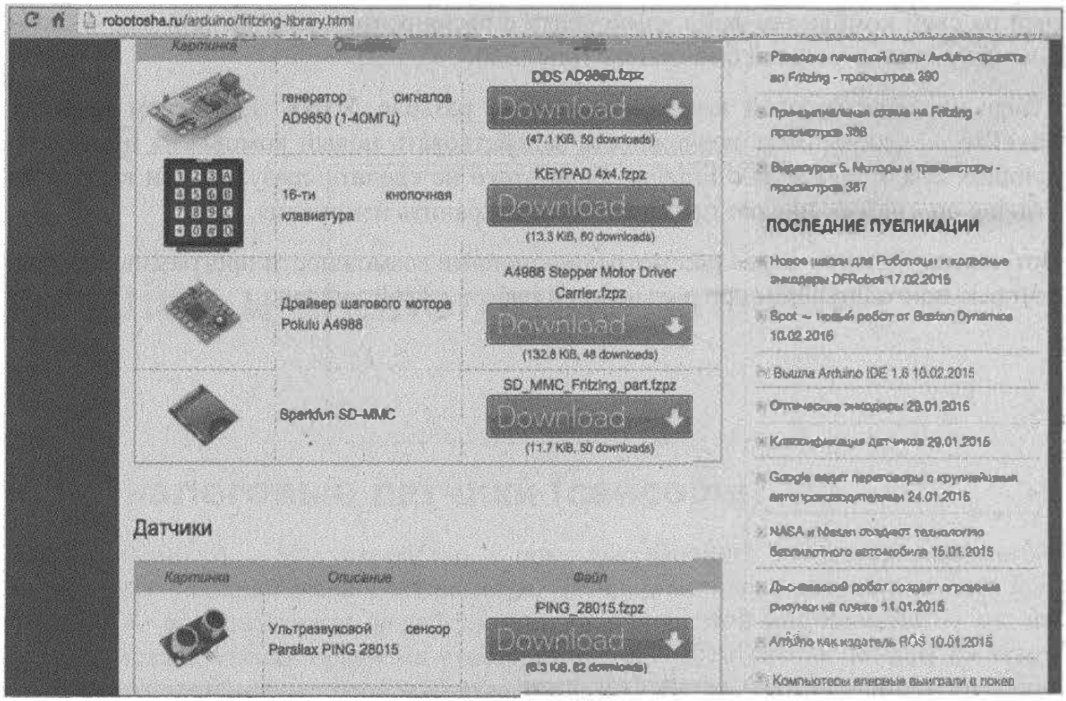


Рис. 3.8. Компоненты для Fritzing на сайте «Роботоша»

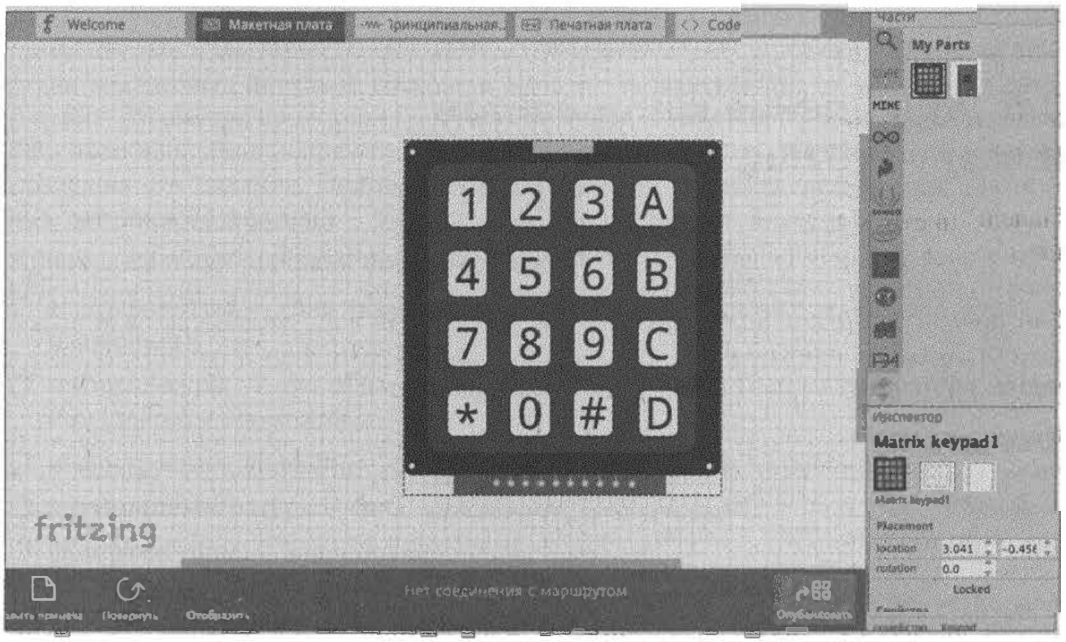


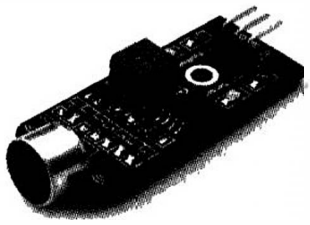
Рис. 3.9. Компонент добавлен в раздел **MINE**

ный на свой компьютер файл компонента с расширением fzpz, fbz или fzbz — компонент появится в нашей библиотеке (рис. 3.9).

Теперь щелкнем правой кнопкой мыши на разделе **MINE** и выберем команду **SaveBin** — чтобы иметь возможность задействовать новый компонент и в последующих сессиях работы с Fritzing. Если этого не сделать сразу, то при выходе из Fritzing программа еще раз предложит вам сохранить изменения.

Вот, пожалуй, и все — мы рассмотрели основные возможности программы Fritzing, которые пригодятся нам при дальнейшей работе с этой книгой.

ГЛАВА 4



Arduino и аналоговые датчики

4.1. Аналоговые датчики (сенсоры)

Датчик (сенсор) — это устройство, с помощью которого мы измеряем значение какого-либо технологического параметра. Датчики позволяют определять, что происходит во внешней среде, и действовать на основе этой информации, — их, наверное, можно назвать органами чувств системы. Любой датчик состоит из чувствительного элемента и преобразовательного устройства, выполняющего преобразование входного воздействия на датчик любой физической величины в сигнал, удобный для дальнейшего использования.

Самый простой тип датчиков — *аналоговые* датчики (первичные преобразователи), применяемые в системах непрерывного измерения и регулирования. Принцип действия аналоговых датчиков состоит в том, что в зависимости от изменения измеряемого параметра происходит соответствующее изменение их выходного сигнала. При этом выходное напряжение датчика может принимать значение от 0 В до напряжения его питания. Впрочем, обычно рабочий диапазон напряжений аналоговых датчиков более узкий.

Примерами таких датчиков могут служить:

- акселерометры — для обнаружения наклона (используются в смартфонах и планшетах);
- магнитометры — для обнаружения магнитных полей (используются при создании цифровых компасов);
- инфракрасные датчики — для определения расстояния до объекта;
- датчики температуры — для определения температуры;
- фоторезисторы — для измерения освещенности.

Между измеряемой величиной датчиков и возвращаемым напряжением установлена определенная зависимость. Например, чем больше величина, тем больше напряжение. Или, наоборот, — чем больше величина, тем напряжение меньше. Так, инфракрасный датчик расстояния Sharp GP2Y0A02YK измеряет расстояние от него до объекта — и чем меньше расстояние, тем больше напряжение. Если объект нахо-

дится на расстоянии 20 см, сенсор выдает на сигнальном проводе примерно 2,5 В. На расстоянии 60 см — примерно 1 В, а на расстоянии 150 см — примерно 0,4 В (рис. 4.1). Иногда зависимость более сложная: напряжение растет до определенного значения, затем падает пропорционально ему. Здесь все зависит от конкретного датчика.

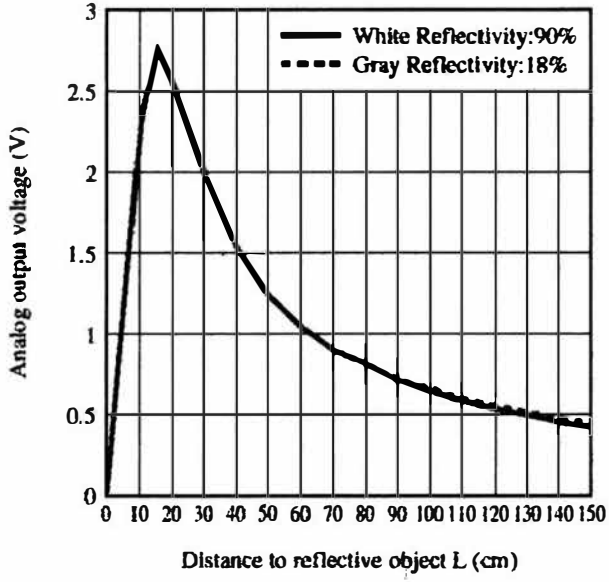


Рис. 4.1. Диаграмма зависимости выходного напряжения от расстояния для датчика Sharp GP2Y0A02YK

Диаграммы для других аналоговых датчиков можно найти в документации на них или получить экспериментально.

Преимуществом сенсоров с аналоговым сигналом является крайняя простота их использования с Arduino. А главный недостаток аналогового сигнала — неустойчивость к внешним шумам: если провод от сенсора до микроконтроллера будет достаточно длинным, он начнет работать как антенна и улавливать внешние электромагнитные поля, — т. е. провод сам станет влиять на выходное напряжение и тем самым искажать показания датчика. Поэтому разумный предел длины провода для аналогового сенсора — не более 50 см.

Аналого-цифровые преобразователи (АЦП)

Датчики, выдающие аналоговый сигнал (напряжение), оцифровываются в управляющей программе с помощью аналого-цифровых преобразователей (АЦП). Микроконтроллеры Atmega, используемые в Arduino, содержат шестиканальный АЦП. Разрешение такого преобразователя составляет 10 битов, что позволяет на выходе получать значения от 0 до 1023.

Различные платы Arduino имеют разные количества аналоговых входов. Например, платы UNO и Duemilanovo имеют 6 аналоговых входов, Nano — 8, Yin — 12, Mega — 16. Основным применением аналоговых входов большинства платформ Arduino явля-

ется чтение показаний аналоговых датчиков, для чего в Arduino существует стандартная функция `analogRead()`. Вот пример чтения данных с аналогового входа A0:

```
int value1=analogRead(A0);
```

Диапазон входного напряжения от 0 до 5 В в программе проецируется на диапазон целочисленных значений от 0 до 1023. Для масштабирования значения к другим значениям можно использовать функцию `map()`:

```
int value2=map(value1,0,1024,min,max);
```

4.2. Arduino и датчик температуры LM335

Познакомимся с работой аналоговых датчиков температуры на примере датчика LM335 (рис. 4.2) — недорогого температурного чувствительного элемента с диапазоном от –40 до +100 °С и с точностью в 1 °С.

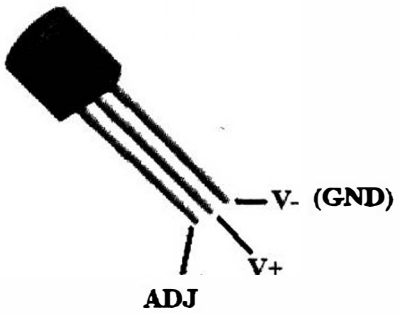


Рис. 4.2. Датчик температуры LM335

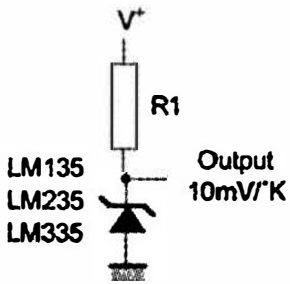


Рис. 4.3. Типовая схема включения датчика LM335

По принципу действия датчик LM335 представляет собой стабилитрон, у которого напряжение стабилизации зависит от температуры, — при повышении температуры на один градус Кельвина напряжение стабилизации увеличивается на 10 милливольт. Типовая схема включения датчика (соответствует типовой схеме включения стабилитрона) показана на рис. 4.3 (номинал резистора R1 равен 2,2 кОм).

При взгляде на эту типовую схему сразу возникает вопрос о напряжении питания (V^+), подаваемом на датчик с учетом указанного номинала резистора R1. Ответ содержится в технической документации на датчик (Datasheet), где сказано, что нормальная работа изделия гарантируется в диапазоне токов 0,45–5,00 миллиампер. Следует заметить, что предел в 5 мА превышать не следует, поскольку датчик станет перегреваться и измерять собственную температуру.

Согласно Datasheet датчик проградуирован в абсолютной шкале Кельвина. При температуре –273,15 °С (это абсолютный ноль по Кельвину) рассматриваемый датчик должен показать и нулевое напряжение. А при увеличении температуры на каждый градус выходное напряжение стабилитрона станет возрастать на целых 10 мВ, или на 0,010 В.

Температура 25 °С — единственная точка калибровки датчика. При этой температуре на выходе датчика должно присутствовать напряжение 298,15 × 0,010 = = 2,9815 В. Для калибровки датчика используется схема, представленная на рис. 4.4.

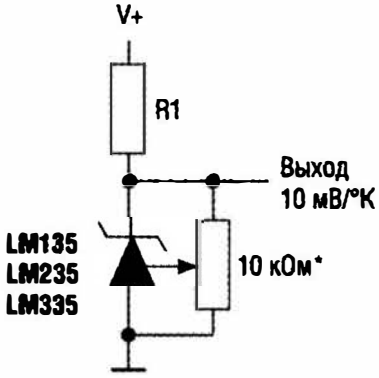


Рис. 4.4. Схема калибровки датчика LM335

Итак, подключаем датчик температуры LM335 по схеме, представленной на рис. 4.5, и пишем скетч считывания данных с датчика и вывода показаний в последовательный порт (листинг 4.1). Вывод результатов измерения данных датчиком LM335 в последовательный порт показан на рис. 4.6.

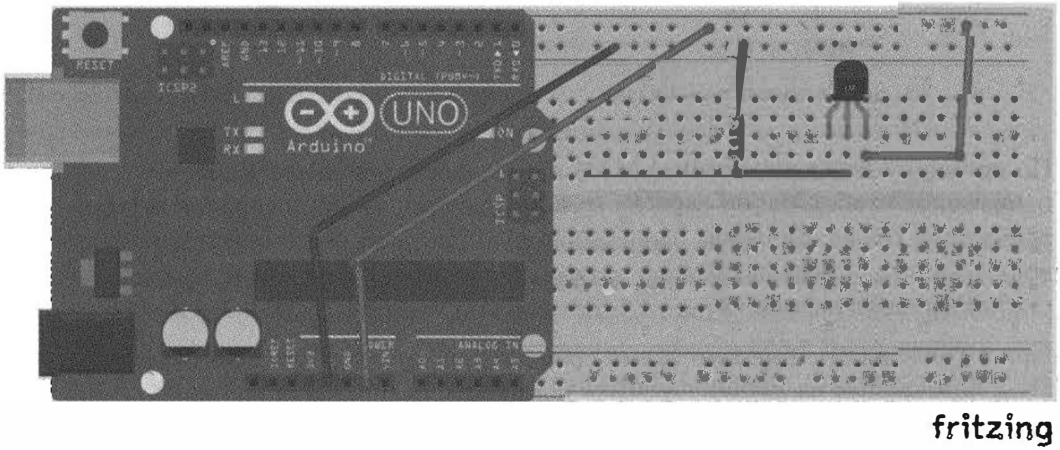


Рис. 4.5. Схема подключения датчика LM335 к Arduino

Листинг 4.1

```
int lm335=0;     // подключение датчика к аналоговому входу A0
void setup()
 {
  Serial.begin(9600);
 }
```

```
void loop()
 {
  double val = analogRead(lm335);  // чтение
  double voltage = val*5.0/1024;   // перевод в значение в вольтах
  double temp = voltage*100 - 273.15;  // перевод в градусы Цельсия
  Serial.print(" temp =  ");
  Serial.println(temp);
  delay(1000);
 }
```

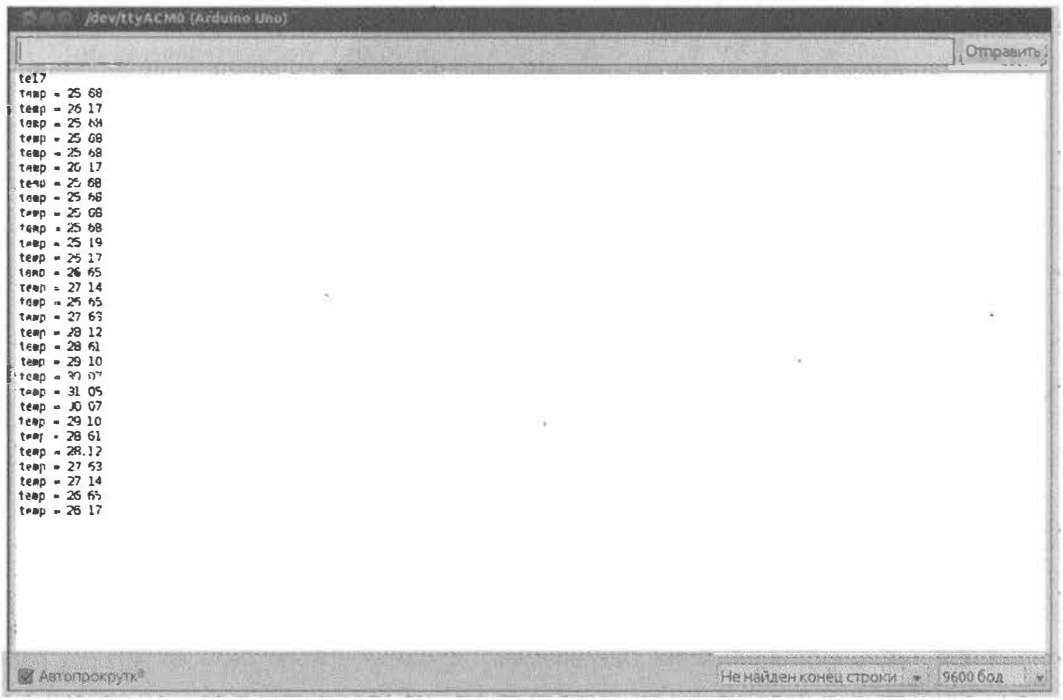


Рис. 4.6. Вывод результатов измерения данных датчиком LM335 в последовательный порт

4.3. Arduino, Ethernet Shield/W5100 и облачные сервисы

Реализуя концепцию Internet of Things (Интернет вещей), согласно которой доступ к данным датчиков должен быть обеспечен из любой точки мира, давайте сейчас организуем отправку данных из Arduino в облачные сервисы, для чего необходимо предоставить плате Arduino доступ к сети Интернет.

Самый распространенный способ предоставления такого доступа — использование платы Ethernet Shield (рис. 4.7) — платы расширения, которая состыковывается с платой Arduino посредством сопряженных разъемов и дает ей возможность, вы-

ступая в роли сетевого устройства, общаться по проводной сети с аналогичными устройствами, с обычными компьютерами, принтерами, сервисами в Интернете и прочими сетевыми ресурсами.

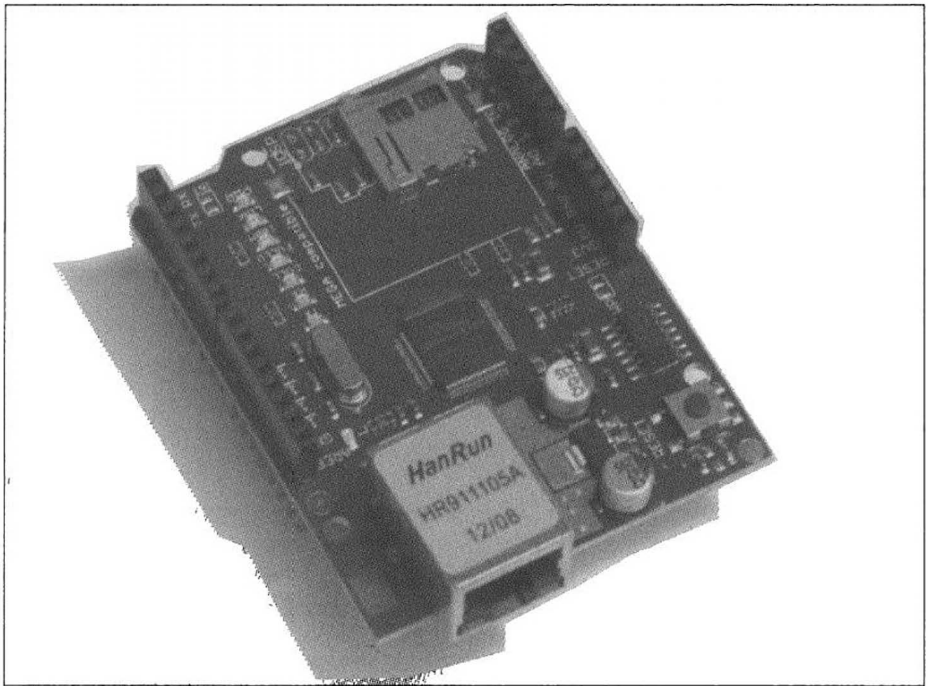


Рис. 4.7. Плата Ethernet Shield Rev3

Плата Ethernet Shield

Плата Ethernet Shield основана на чипе Wiznet W5100, который поддерживает как TCP-, так и UDP-протоколы. Одновременно открытыми могут быть до четырех подключений.

Плата обладает стандартным Ethernet-портом для подключения к сети с помощью патч-корда «витая пара» и набором контактов для сопряжения с Arduino. Для общения между собой Ethernet Shield и Arduino задействуют контакты 4-й и с 10-го по 13-й, поэтому их использование в других целях в присутствии платы расширения невозможно.

Для программирования сетевого взаимодействия служит библиотека Ethernet из стандартного дистрибутива. При использовании этой библиотеки необходимо указывать MAC-адрес платы (уникальный идентификатор любого сетевого устройства). В более новых версиях платы Ethernet Shield ее MAC-адрес можно найти на наклейке на плате. Если такой наклейки нет, то просто введите любую подходящую цифробуквенную комбинацию, — главное, чтобы в вашей сети не было устройств с совпадающими MAC-адресами.

Плата Ethernet Shield также оборудована слотом для карты памяти формата MicroSD, на которой можно хранить ресурсы, раздаваемые по сети. Для взаимодействия с SD-картой может быть использована библиотека sdfatlib.

Последняя версия платы Ethernet Shield Rev3 полностью совместима с Arduino Mega2560.

Итак, с учетом сказанного, для отправки данных в облачные сервисы мы задействуем плату Arduino с подключенной к ней платой Ethernet Shield, выступающей в качестве веб-клиента, и создадим на этой основе простой пример получения нашим устройством IP-адреса по DHCP. Подключается Ethernet Shield к плате Arduino так же просто, как и любой другой шилд, — просто состыкуйте их вместе.

Соединив Ethernet Shield с платой Arduino, подключим плату Arduino к USB-порту компьютера, а к Ethernet Shield подсоединим сетевой кабель (рис. 4.8).

> ***Совет***
>
> **Следует учесть, что установка других шилдов поверх Ethernet Shield весьма затруднительна. Это связано с большими размерами разъема RJ-45, служащего для подключения сетевого кабеля, поэтому, если вы хотите использовать другие шилды, лучше их размещать между Arduino и Ethernet Shield.**

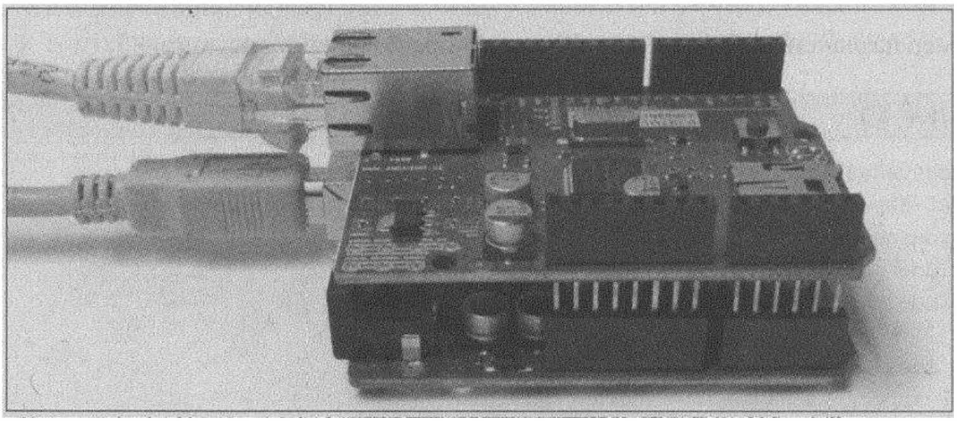


Рис. 4.8. Подключение плат Ethernet Shield Rev3 и Arduino

Пример получения IP-адреса по DHCP представлен в листинге 4.2.

Листинг 4.2

```
// Получение IP-адреса по DHCP
// MAC-адрес Ethernet Shield (можно увидеть на наклейке на плате) или
// произвольный, но уникальный в сети
#include <Ethernet.h>
#include <SPI.h>

byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  }
  // запуск Ethernet-соединения
  if (Ethernet.begin(mac) == 0) {
```

```
    Serial.println("Failed to configure Ethernet using DHCP");
    for (;;)
      ;
  }
  // печать в последовательный порт полученного по DHCP адреса
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }
  Serial.println();
}

void loop() {;}
```

Пример назначения статического IP-адреса представлен в листинге 4.3.

Листинг 4.3

```
// Получение статического IP-адреса
// MAC-адрес Ethernet Shield (можно увидеть на наклейке на плате) или
// произвольный, но уникальный в сети
#include <Ethernet.h>
#include <SPI.h>

byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
// IP-адрес, назначаемый Ethernet Shield:
byte ip[] = { 192, 168, 0, 111 };
// IP-адрес, dns сервера:
byte sdns[] = { 192, 168, 1, 1 };
// адрес шлюза:
byte gateway[] = { 192, 168, 0, 1 };
// маска:
byte subnet[] = { 255, 255, 255, 0 };

void setup() {
  Serial.begin(9600);
  // запуск Ethernet-соединения
  Ethernet.begin(mac, ip, sdns, gateway, subnet);
  delay(1000);
  Serial.println(Ethernet.localIP());
}
void loop() {;}
```

Получив доступ в Интернет, можно отправлять данные с платы Arduino в облачные сервисы. В следующем разделе мы рассмотрим пример такой отправки данных — на сайт «Народный мониторинг».

4.3.1. Отправка данных на сайт «Народный мониторинг»

Сайт «Народный мониторинг» (**http://www.narodmon.ru**) — проект по сбору и отображению на карте мира показаний температуры, давления, влажности и т. п. практически в реальном времени по фактическому их состоянию (а не на основе прогнозов) от различных датчиков среды, установленных как на улице для публичного доступа, так и в помещении для приватного, а также с веб-камер для частного или публичного доступа. Передавать показания датчиков на **narodmon.ru** можно посредством протоколов TCP/UDP или HTTP GET/POST. Минимальный интервал передачи показаний датчика — 5 минут (если передавать чаще, то возможна блокировка).

Чтобы стать участником проекта, необходимо зарегистрироваться: заходим на сайт **http://www.narodmon.ru**, выбираем команду меню **Вход | Стать участником проекта** и в открывшуюся форму (рис. 4.9) вводим адрес электронной почты, куда будут нам отправлены логин и пароль для входа в профиль.

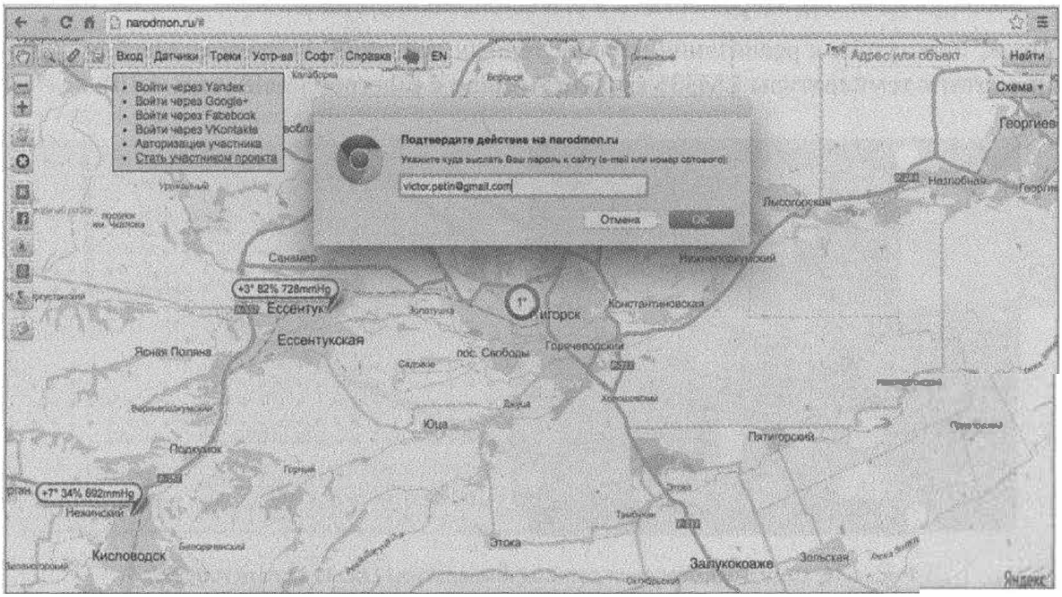


Рис. 4.9. Регистрация на сайте «Народный мониторинг»

Теперь настроим передачу и отображение показаний нашего, подключенного к Arduino, датчика температуры LM335 из *разд. 4.1*. Для добавления датчика на карту необходимо выполнить следующие действия:

1. Подключить устройство мониторинга (в нашем случае Arduino) к источнику питания и к сети Интернет (с помощью платы Ethernet Shield).
2. Настроить передачу показаний на сайт **narodmon.ru** с интервалом 5–15 минут (напомню, если чаще, то возможна блокировка).

3. Авторизоваться на сайте **narodmon.ru**, используя свой логин (e-mail или номер мобильного телефона) и пароль, полученные при регистрации.
4. В разделе **Мои Датчики** добавить устройство, введя его уникальный алфавитно-цифровой код (MAC-адрес). Добавление возможно только после успешной передачи показаний на сервер и верно указанного MAC-адреса.
5. Выбрать тип данных нашего датчика: **температура**. Если у нас подключены к Arduino еще несколько датчиков: влажности, давления и др., то выбираем все актуальные.
6. Установить доступ к показаниям для каждого датчика: **публичный** (виден всем) или **приватный** (только вам).
7. Указать названия для устройства и подключенных к нему датчиков.
8. Выполнить привязку к карте устройства мониторинга, указав полный адрес его размещения или геокоординаты, щелкнув для этого по строке с адресом в поле **Устройство** раздела **Мои Датчики**.
9. Уточнить свое местоположение, переместив маркер своего устройства на карте, щелкнув на опции **Переместить** в его всплывающем окне.

Что ж, приступим к реализации перечисленных действий. И прежде всего подключим датчик температуры LM335 к плате Arduino с Ethernet Shield (рис. 4.10).

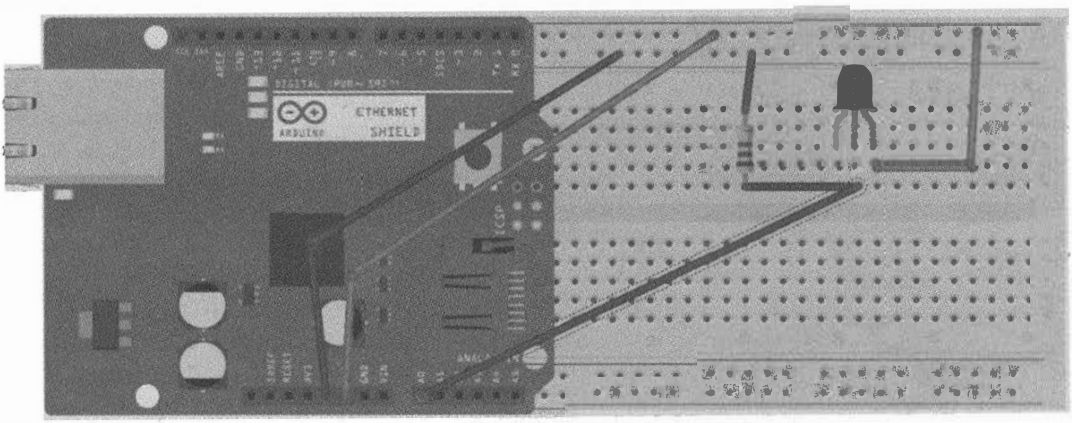


Рис. 4.10. Схема подключения датчика температуры LM335 к плате Arduino, состыкованной с Ethernet Shield/W5100

Далее загрузим на плату Arduino рекомендованный пример подключения к сервису **narodmon.ru** от **student-proger.ru**, изменив его под свои требования (сетевые параметры и датчики). Содержимое скетча представлено в листинге 4.4.

Листинг 4.4

```
#include <SPI.h>
#include <Ethernet.h>
```

```
byte mac[] = {  0x94, 0xDE, 0x80, 0x3A, 0x90, 0xC9 }; // MAC-адрес Arduino
const unsigned long postingInterval = 600000;  // интервал между отправками
                                               // данных - 10 минут
// IP-адрес, назначаемый Ethernet Shield:
byte ip[] = { 192, 168, 0, 119 };
// IP-адрес, dns сервера:
byte sdns[] = { 192, 168, 1, 1 };
// адрес шлюза:
byte gateway[] = { 192, 168, 0, 28 };
// маска:
byte subnet[] = { 255, 255, 255, 0 };

IPAddress server(94,19,113,221); // IP-адрес сервера
//IPAddress server(91,122,49,168); // IP-адрес сервера
EthernetClient client;
unsigned long lastConnectionTime = 0;    // время последней передачи данных
boolean lastConnected = false;           // состояние подключения
char replyBuffer[160];

void setup()
  {
  Serial.begin(9600);
  // Ethernet connection:
  Ethernet.begin(mac,ip,sdns,gateway,subnet);
  // секунда для инициализации Ethernet
  delay(1000);
  // первое соединение через 15 секунд после запуска
  lastConnectionTime = millis()-postingInterval+15000;
}

void loop()
  {
  // если не подключены, и прошло определенное время, то делаем замер,
  // переподключаемся и отправляем данные
  if (!client.connected() && (millis() - lastConnectionTime > postingInterval))
    {
    // формирование HTTP-запроса
    memset(replyBuffer, 0, sizeof(replyBuffer));
    strcpy(replyBuffer,"ID=");
    // конвертируем MAC-адрес
    for (int k=0; k<6; k++)
      {
      int b1=mac[k]/16;
      int b2=mac[k]%16;
      char c1[2],c2[2];

      if (b1>9) c1[0]=(char)(b1-10)+'A';
      else c1[0] = (char)(b1) + '0';
```

```
      if (b2>9) c2[0]=(char)(b2-10)+'A';
      else c2[0] = (char)(b2) + '0';

      c1[1]='\0';
      c2[1]='\0';

      strcat(replyBuffer,c1);
      strcat(replyBuffer,c2);
      }
   strcat(replyBuffer,"&");
   strcat(replyBuffer,"3351C4BA0200003B");
   strcat(replyBuffer,"=");
   char temp[3];
   double tmpd=(analogRead(A0)*5.0/1024)*100-273.15;
   int tmpi=int(tmpd);
   itos(tmpi,temp);
   strcat(replyBuffer,temp);
   strcat(replyBuffer,'\0');
   // отправляем запрос
   httpRequest();
   }
  // храним последнее состояние подключения
  lastConnected = client.connected();
  }
// функция отправки запроса
void httpRequest() {
  if (client.connect(server, 80))
    {
    // send the HTTP POST request:
    client.println("POST http://narodmon.ru/post.php HTTP/1.0");
    client.println("Host: narodmon.ru");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.println(len(replyBuffer));
    client.println();
    client.println(replyBuffer);
    client.println();
    lastConnectionTime = millis();
    }
  else
    {
    client.stop();
    }
  }

// размер данных
int len(char *buf)
```

```
{
  int i=0;
  do
    {
    i++;
    } while (buf[i]!='\0');
  return i;
}
// функция int to string
void itos(int n, char bufp[3]) //
{
  char buf[3]={'0','0','\0'};
  int i = 1;
  while (n > 0)
    {
    buf[i] = (n % 10)+48;
    i--;
    n /= 10;
    }
  for (i=0; i<3; i++)
    bufp[i]=buf[i];
  }
```

Для передачи данных на сайт «Народный мониторинг» мы используем резервный протокол передачи HTTP POST/GET на URL: **http://narodmon.ru/post.php**.

При этом HTTP-заголовки для POST будут следующими:

```
POST http://narodmon.ru/post.php HTTP/1.0\r\n
Host: narodmon.ru\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: NN(кол-во байт в строке данных ниже)\r\n
\r\n
ID=MAC&mac1=value1&...&macN=valueN[&time=UnixTime][&name=NAME][&lat=LAT][&lng=L
NG]
```

Скетч запускает Ethernet-соединение, плата получает IP-адрес в Сети, один раз в 5 минут считываются данные с датчика температуры, формируется строка с данными для отправки на сервер «Народный мониторинг», и данные отправляются с использованием протокола HTTP POST.

> ***Электронный архив***
>
> Скетч, соответствующий листингу 4.4, можно найти в файле arduino_scetches_04_04_04.ino сопровождающего книгу электронного архива.

Теперь авторизуемся на сайте, используя логин и пароль, пришедшие на нашу электронную почту, выбираем команду меню **Датчики | Мои Датчики | Добавить устройство** и вводим MAC-адрес нашего устройства. Если данные уже были отправлены на сайт, устройство будет добавлено (рис. 4.11).

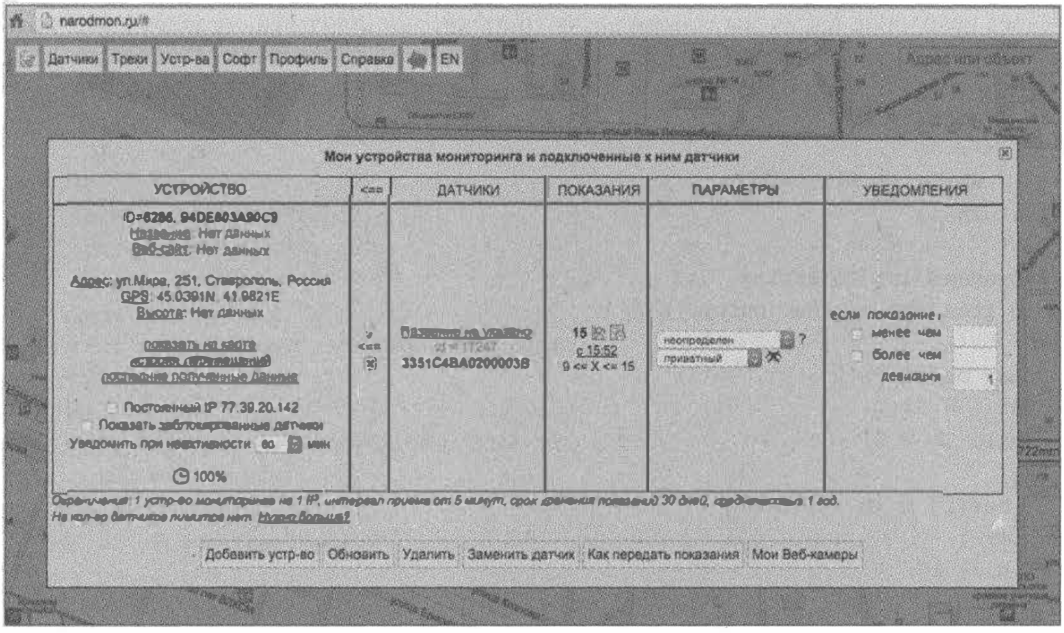


Рис. 4.11. Добавление устройства на сайте «Народный мониторинг»

Затем выбираем тип данных для нашего датчика (температура), устанавливаем доступ к показаниям (**приватный**), указываем название устройства и выполняем привязку к карте устройства мониторинга, указав полный адрес щелчком по строке с адресом. После чего выбираем опцию **показать на карте** (см. рис. 4.11) и в случае необходимости корректируем положение на карте с помощью плавающего меню (рис. 4.12).

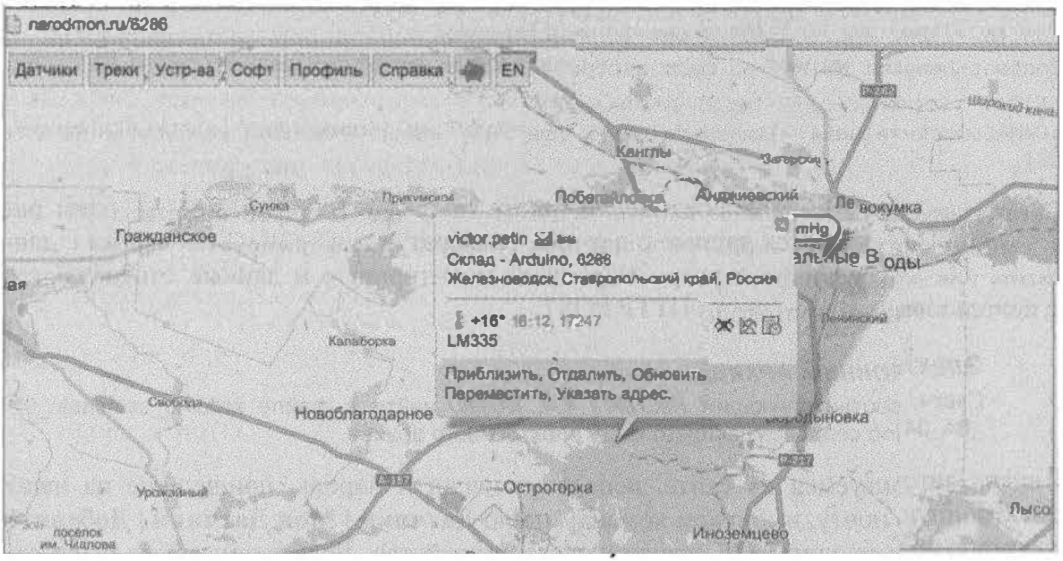


Рис. 4.12. Добавление устройства на карту сайта «Народный мониторинг»

Через некоторое время мы можем посмотреть временной график изменения данных датчика на нашем устройстве. Для этого выбираем команду меню **Профиль | Мои Датчики** и значок графика для выбранного датчика. На графике (рис. 4.13) представлено изменение переданных данных датчика во времени.

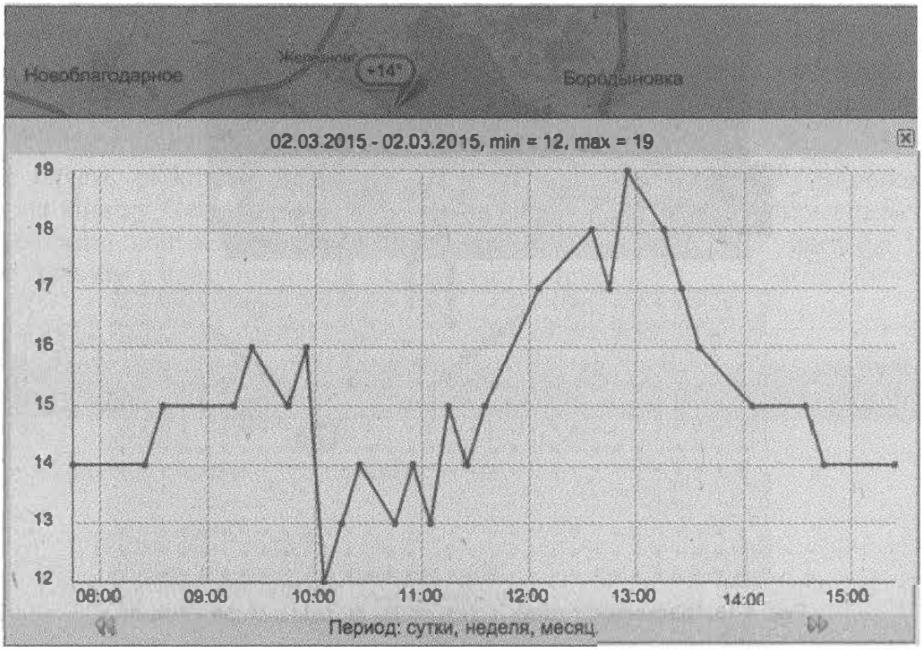


Рис. 4.13. Временной график переданных показаний на сайте «Народный мониторинг»

4.3.2. Чтение данных с фоторезистора

Рассмотрим теперь еще один аналоговый датчик — фоторезистор (рис. 4.14). Чаще всего с помощью фоторезисторов осуществляют измерение освещенности. Дело в том, что в темноте сопротивление фоторезистора весьма велико, но когда на него попадает свет, это сопротивление падает пропорционально освещенности.

Для схемы измерения освещенности (рис. 4.15) необходимо собрать делитель напряжения, в котором верхнее плечо будет представлено фоторезистором, а ниж-

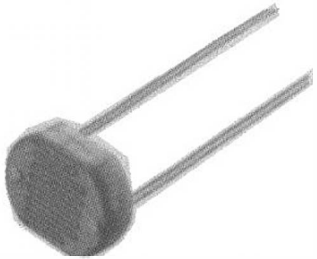


Рис. 4.14. Фоторезистор

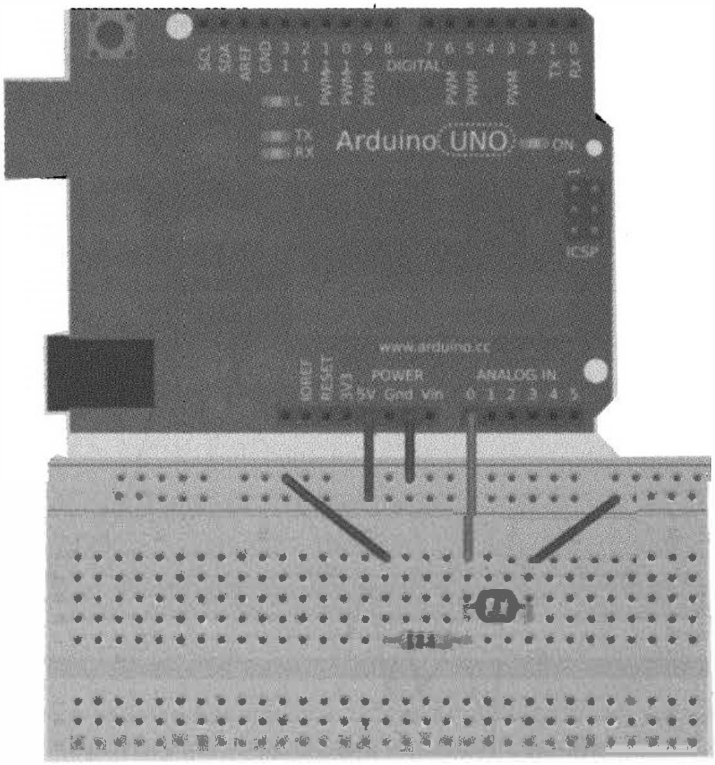


Рис. 4.15. Графическая схема подключения фоторезистора к Arduino

нее — обычным резистором достаточно большого номинала (мы применим здесь резистор 10 кОм). Среднее же плечо делителя подключается к аналоговому входу A0 Arduino.

Скетч, который измеряет напряжение на аналоговом входе и отправляет его в последовательный порт, представлен в листинге 4.5.

Листинг 4.5

```
// Получение данных с фоторезистора и отправка в последовательный порт
int light;
void setup()
  {
  Serial.begin(9600);
  }

void loop()
  {
  light = analogRead(0);
  Serial.println(light);
  delay(100);
  }
```

При уменьшении освещенности фоторезистора (можно просто затенить его рукой) сопротивление его будет возрастать, что приведет к уменьшению напряжения на аналоговом входе.

4.3.3. Отправка данных в сервис ThingSpeak

Сервис ThingSpeak (**https://thingspeak.com**) — открытая платформа данных для проектов Internet of Things, включающая в себя сбор данных с датчиков в реальном времени, обработку этих данных, их визуализацию и использование в приложениях и плагинах.

Чтобы начать работу с сервисом ThingSpeak, необходимо зарегистрироваться, нажав на кнопку **GetStarted Now** в стартовом окне сервиса. В открывшемся окне регистрации заполняем требуемые поля формы (рис. 4.16), нажимаем на кнопку **Create Account** и сразу попадаем в свой аккаунт (рис. 4.17).

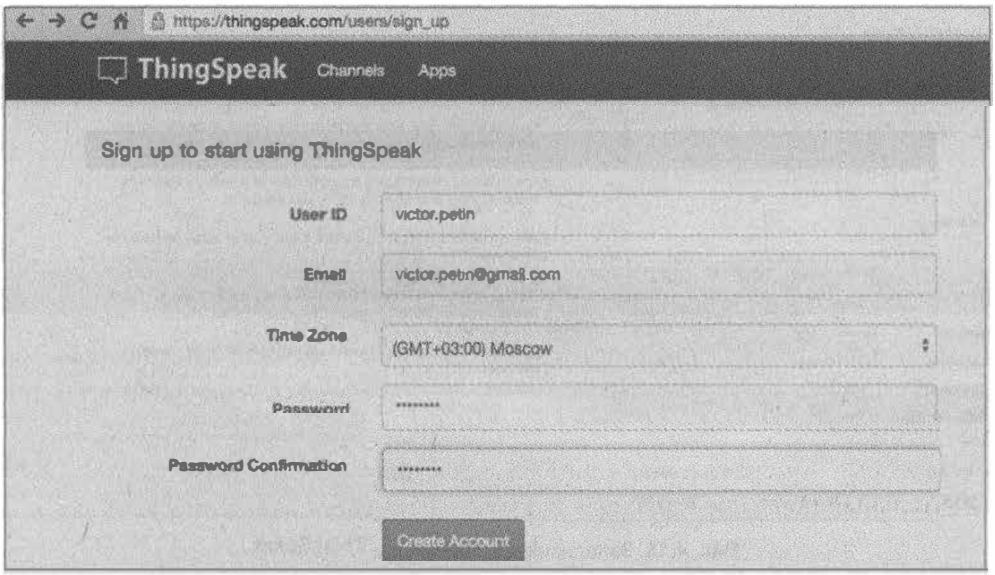


Рис. 4.16. Форма регистрации ThingSpeak

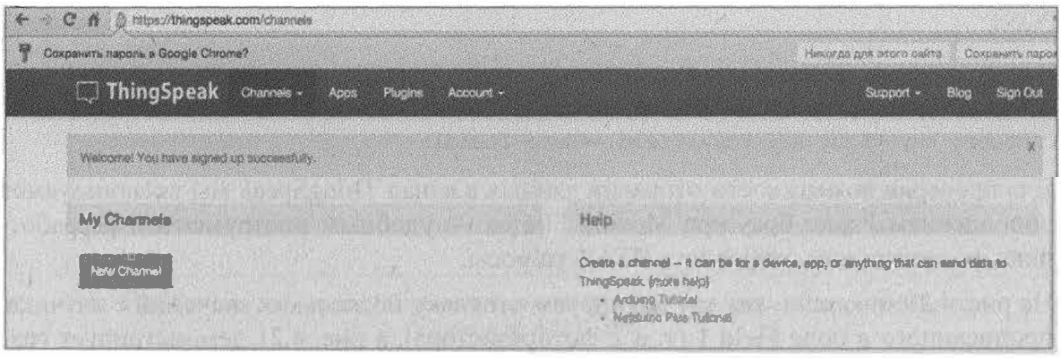


Рис. 4.17. Успешная регистрация в сервисе ThingSpeak

Здесь нам надо создать канал (Channel), в котором будут храниться наши данные. Каждый канал включает в себя восемь полей для любого типа данных, три поля местоположения и одно поле состояния. Таким образом, один канал мы можем использовать для отправки и хранения данных с одного устройства, имеющего не более восьми датчиков.

Для создания канала нажимаем на кнопку **New Channel**, заполняем поля, как показано на рис. 4.18, и сохраняем канал, нажав на кнопку **Save Channel**. Все — канал **Склад** создан (рис. 4.19). Как можно видеть на рис. 4.18, мы в качестве датчика задействовали в создаваемом канале фоторезистор, прописав его в поле **Field 1**.

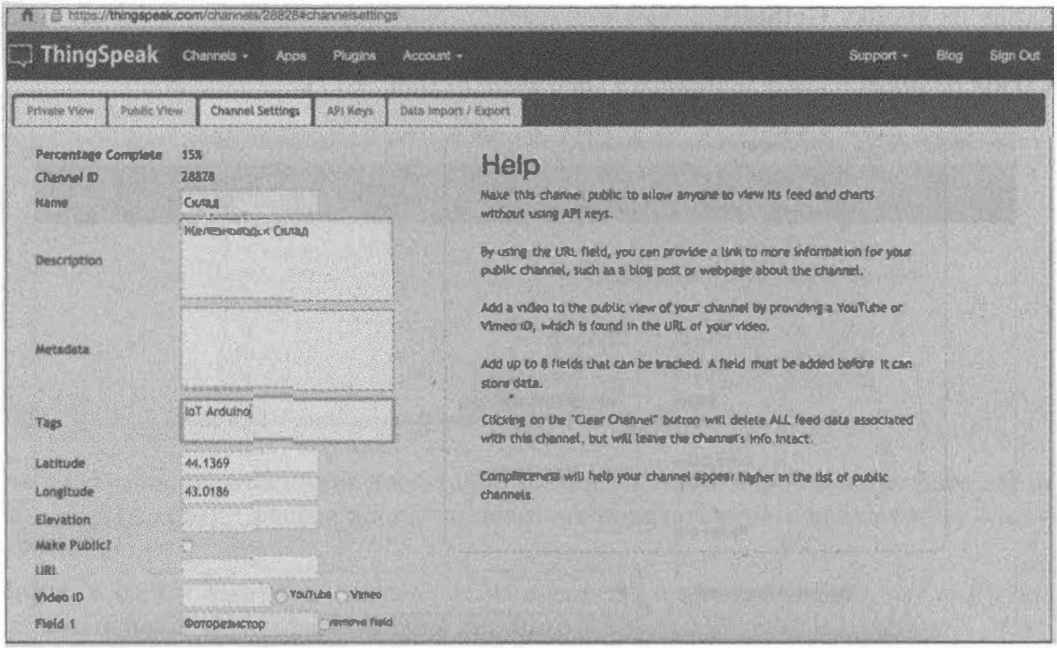


Рис. 4.18. Заполнение полей канала ThingSpeak

Отправка данных в канал осуществляется с использованием протокола HTTP POST:

```
URL: http://api.thingspeak.com/update
Content Type: application/x-www-form-urlencoded
Content: key=<Your API Key>&field1=<data field1>
```

Для проверки возможности отправки данных в канал ThingSpeak мы воспользуемся дополнением Poster браузера Mozilla Firefox — удобным инструментом разработчика, позволяющим отправлять HTTP-запросы.

На рис. 4.20 показано, как мы эмулируем отправку нескольких значений с датчика, прописанного в поле **Field 1** (т. е. с фоторезистора), а рис. 4.21 демонстрирует график канала, отображающий поступившие данные.

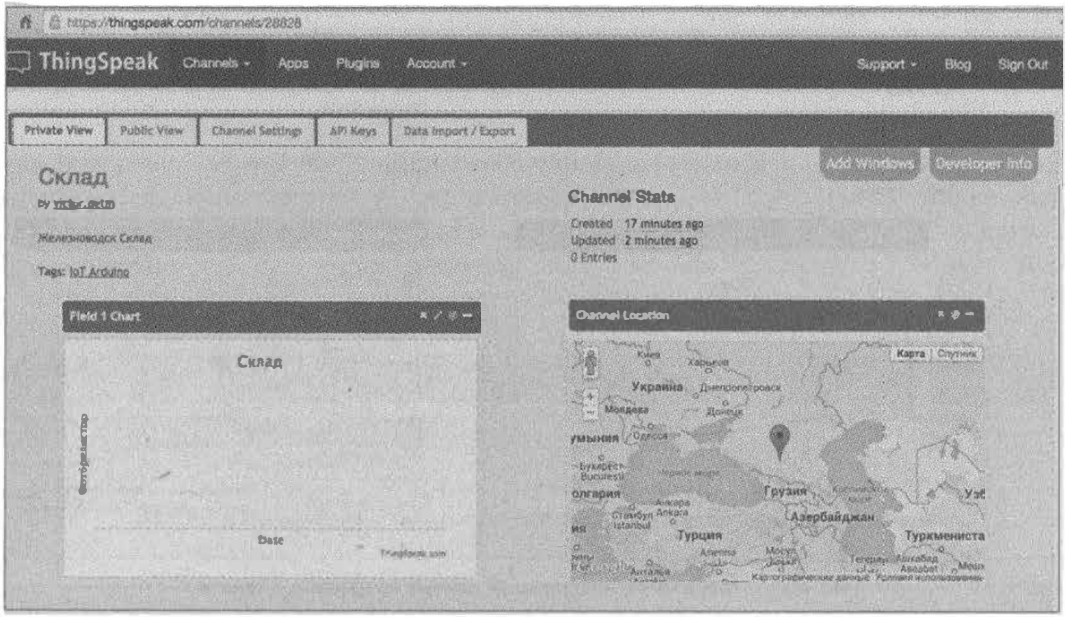


Рис. 4.19. Канал ThingSpeak создан

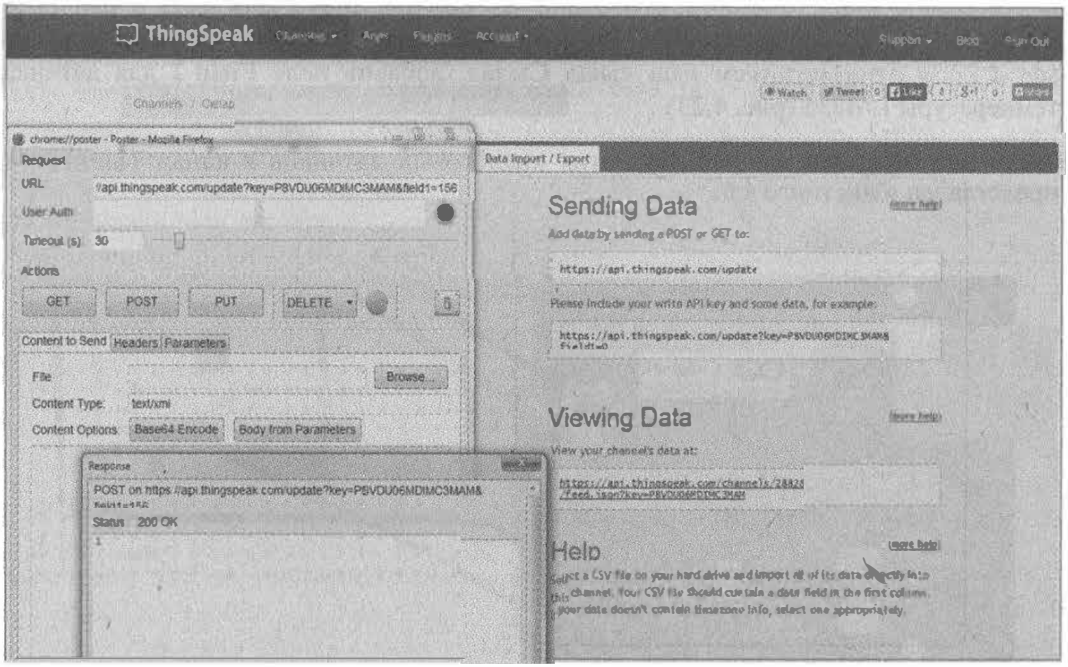


Рис. 4.20. Отправка данных в ThingSpeak через HTTP POST с помощью дополнения Poster браузера Mozilla Firefox

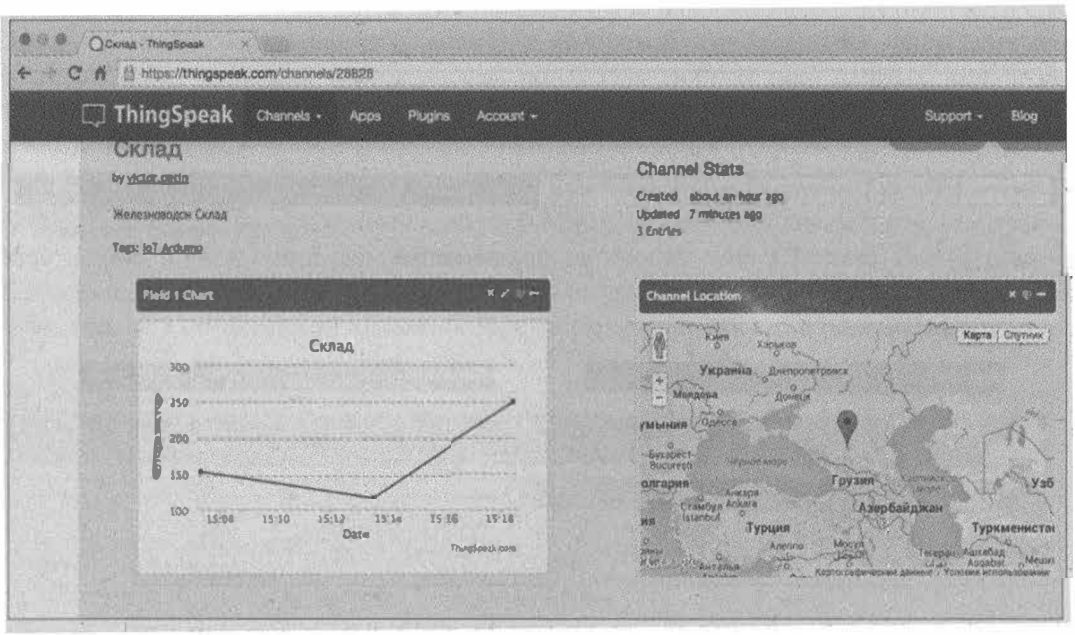


Рис. 4.21. Отображение поступивших данных на графике в канале

А сейчас мы осуществим отправку в ThingSpeak данных с реальных датчиков, подключенных к плате Arduino, состыкованной с Ethernet Shield. Для этого подключим к ней датчик температуры LM335 и фоторезистор по схеме, приведенной на рис. 4.22, и отредактируем наш канал **Склад**, добавив поле **Field 2** для датчика температуры LM335 (рис. 4.23).

Скетч для отправки данных двух этих аналоговых датчиков в канал ThingSpeak представлен в листинге 4.6.

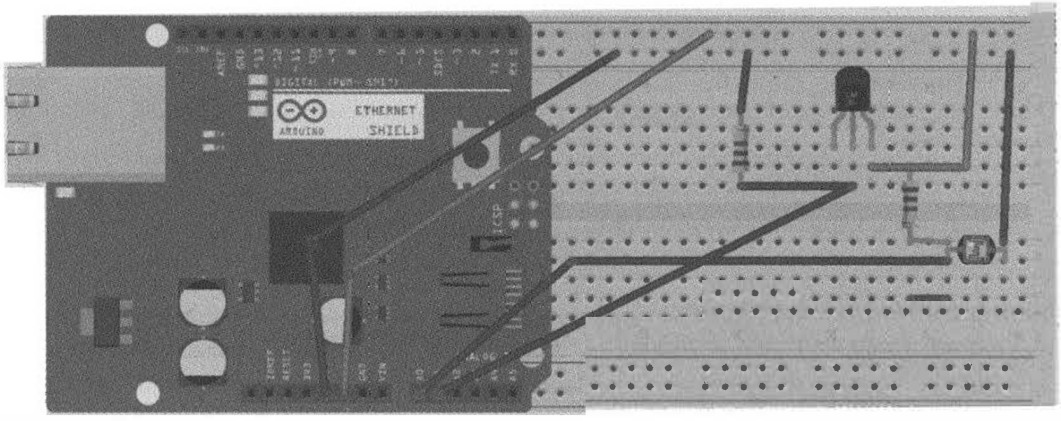


Рис. 4.22. Схема подключения датчиков температуры и освещенности (фоторезистора) к Arduino

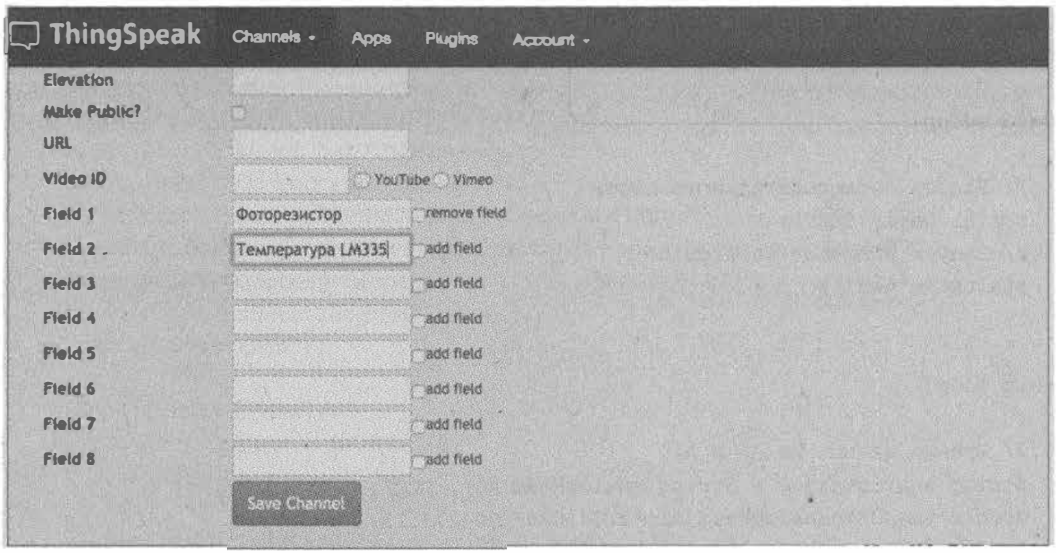


Рис. 4.23. Добавление нового поля (**Field 2**) в канал **Склад** сервиса ThingSpeak

Листинг 4.6

```
#include <SPI.h>
#include <Ethernet.h>

// уникальный MAC-адрес платы Arduino
byte mac[] = { 0xD4, 0x28, 0xB2, 0xFF, 0xA0, 0xA1 };

// IP-адрес, назначаемый Ethernet Shield:
byte ip[] = { 192, 168, 0, 119 };
// IP-адрес, dns сервера:
byte sdns[] = { 192, 168, 1, 1 };
// адрес шлюза:
byte gateway[] = { 192, 168, 0, 28 };
// маска:
byte subnet[] = { 255, 255, 255, 0 };

// ThingSpeak Settings
char thingSpeakAddress[] = "api.thingspeak.com";
String writeAPIKey = "P8VDU06MDIMC3MAM";
// Интервал отправки данных на сервер - 16 сек
const int updateThingSpeakInterval = 16 * 1000;

// служебные переменные
long lastConnectionTime = 0;
boolean lastConnected = false;
int failedCounter = 0;
```

```
// Инициализация Arduino Ethernet client
EthernetClient client;

void setup()
  {
  // Запуск последовательного порта
  Serial.begin(9600);
  // Запуск Ethernet на Arduino
  startEthernet();
  }

void loop()
  {
  // Чтение данных из A0 и A1
  String analogValue0 = String(analogRead(A0), DEC);
  double temp335=analogRead(A1)*5.0/1024*100-273.15;
  String analogValue1 = String(temp335);
  // печать запроса в последовательный порт
  if (client.available())
    {
    char c = client.read();
    Serial.print(c);
    }

  // Разъединение с ThingSpeak
  if (!client.connected() && lastConnected)
    {
    Serial.println("...disconnected");
    Serial.println();
    client.stop();
    }

  // отправка данных в канал ThingSpeak
  if(!client.connected() && (millis() - lastConnectionTime >
updateThingSpeakInterval))
    {
    updateThingSpeak("field1="+analogValue0+"&field2="+analogValue1);
    Serial.println(analogValue0);
    Serial.println(analogValue1);
    }
  // При кол-ве неуспешных попыток >10 - перезапуск интернет-соединения
  if (failedCounter > 10 ) {startEthernet();}
  lastConnected = client.connected();
}
// отправка данных в канал ThingSpeak
void updateThingSpeak(String tsData)
```

```
  {
  if (client.connect(thingSpeakAddress, 80))
    {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");

    client.print(tsData);
    lastConnectionTime = millis();
    if (client.connected())
      {
      Serial.println("Connecting to ThingSpeak...");
      Serial.println();

      failedCounter = 0;
      }
    else
      {
      failedCounter++;
      Serial.println("Connection to ThingSpeak failed ("+String(failedCounter,
DEC)+")");
      Serial.println();
      }
    }
  else
    {
    // увеличение счетчика неуспешных попыток отправки данных
    failedCounter++;
    Serial.println("Connection to ThingSpeak Failed ("+String(failedCounter,
DEC)+")");
    Serial.println();
    lastConnectionTime = millis();
    }
  }
// Перезапуск интернет-соединения
void startEthernet()
  {
  client.stop();
  Serial.println("Connecting Arduino to network...");
  Serial.println();
  delay(1000);
  Ethernet.begin(mac,ip,sdns,gateway,subnet);
```

```
  delay(1000);
    Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++)
   {
    // печать IP-адреса
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
    }
  Serial.println();
  delay(1000);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 4.6, можно найти в файле arduino_scetches_04_04_06.ino сопровождающего книгу электронного архива.

Загружаем этот скетч в плату Arduino и через некоторое время наблюдаем в окне сервиса ThingSpeak графики, построенные на основе отправленных в наш канал значений температуры и освещенности (рис. 4.24).

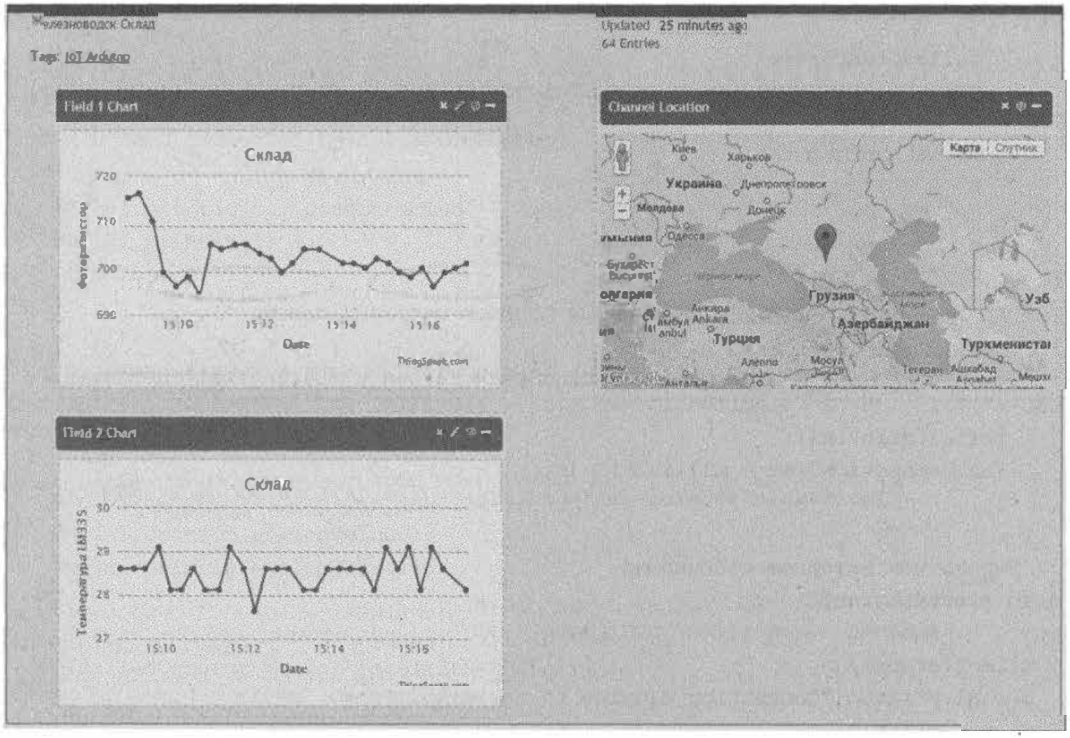


Рис. 4.24. Отображение поступивших из платы Arduino данных на графиках в канале сервиса ThingSpeak

4.4. Arduino и инфракрасные датчики расстояния SHARP

Для измерения расстояния до объекта используются аналоговые оптические датчики, основанные на методе триангуляции. Самые распространенные из них — это инфракрасные (Infra-Red, IR) датчики расстояния с выходным аналоговым напряжением, производимые фирмой Sharp (рис. 4.25).

В датчиках Sharp установлен инфракрасный (IR) светодиод (LED), излучающий через фокусирующую линзу узкий световой луч в инфракрасном диапазоне. Отраженный от объекта луч направляется через другую линзу на позиционно-чувствительный фотоэлемент (Position-Sensitive Detector, PSD). Угол отражения луча от объекта, а соответственно, и место на элементе PSD, куда попадает отраженный луч, зависят от расстояния до объекта (рис. 4.26). А проводимость элемента PSD зависит от того, в какое место на нем попадает отраженный луч. Проводимость эта преобразуется в напряжение, оцифровывая которое аналого-цифровым преобразователем микроконтроллера, можно вычислить расстояние до объекта.

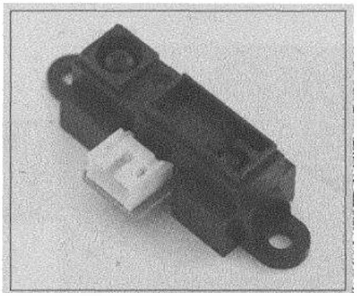


Рис. 4.25. Инфракрасный датчик расстояния Sharp

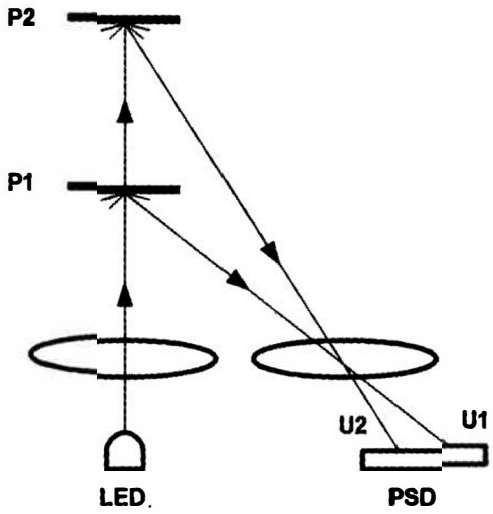


Рис. 4.26. Путь светового луча инфракрасного измерителя расстояния при различных расстояниях до объекта

Выход датчика расстояния Sharp обратно пропорциональный — с увеличением расстояния его значение медленно уменьшается (рис. 4.27). Датчики, в зависимости от их типа, имеют границы измерения, в пределах которых их выход может быть признан надежным. Измерение максимального реального расстояния ограничивают два фактора: уменьшение интенсивности отраженного света и невозможность PSD регистрировать незначительные изменения местоположения отраженного луча. Так, при измерении расстояния до сильно удаленных объектов выход датчика оста-

ется приблизительно таким же, как и при измерении минимальных расстояний. Дело в том, что минимально измеряемое датчиками Sharp расстояние ограничено их особенностью, проявляющейся в резком падении выходного напряжения при малых расстояних до объекта (в зависимости от типа датчика: от 4-х до 20 см). По существу, это означает, что одному значению выходного напряжения соответствуют два расстояния: очень близкое и очень далекое. Для предотвращения возникновения такой проблемы следует избегать слишком близкого приближения датчиков к объекту.

В целом график зависимости напряжения от расстояния хоть и не является линейным, однако в пределах допустимых расстояний график обратной величины вы-

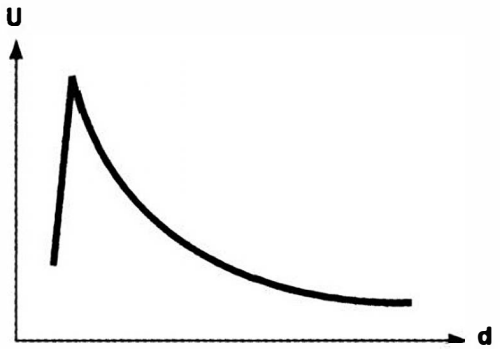


Рис. 4.27. График зависимости напряжения U от расстояния d для датчиков Sharp

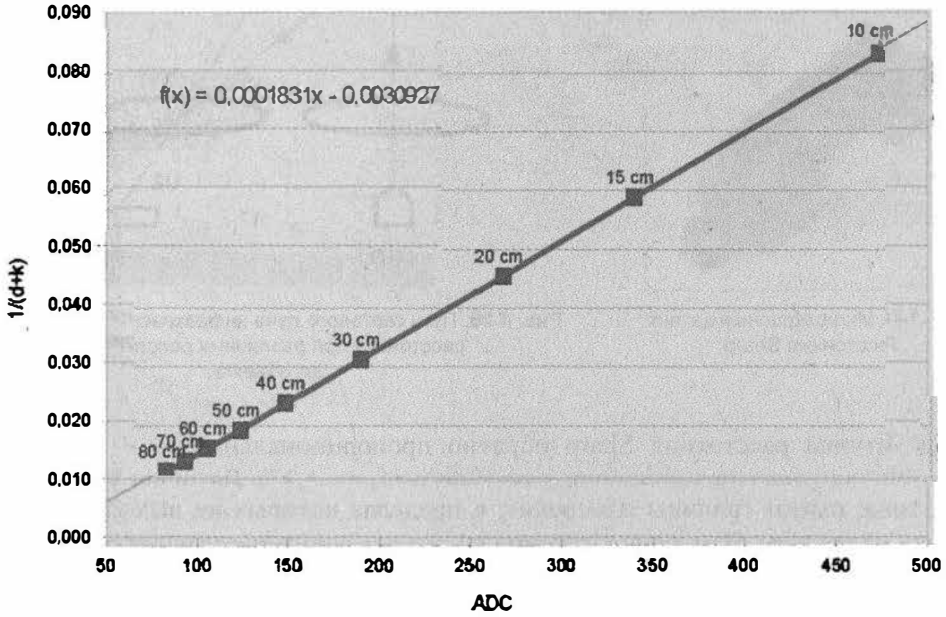


Рис. 4.28. График зависимости выходного напряжения датчика GP2Y0A21YK от расстояния до объекта

ходного напряжения и расстояния к линейности приближается достаточно близко, и с его помощью довольно просто получить формулу преобразования напряжения в расстояние. Для нахождения такой формулы необходимо точки этого графика ввести в какую-либо программу обработки табличных данных и из них создать новый график, а на основе точек этого графика автоматически вычислить линию тренда. На рис. 4.28 приведен график связи исправленной обратной величины выходного напряжения инфракрасного датчика GP2Y0A21YK с расстоянием вместе с линейной линией тренда. Выходное напряжение для упрощения формулы уже переведено в 10-битное значение аналого-цифрового преобразователя с опорным напряжением +5 В.

4.4.1. Подключение датчиков Sharp к Arduino

Работать с сенсором Sharp очень просто — достаточно подключить к нему питание и завести его вывод Vo на аналоговый вход Arduino (рис. 4.29). Значение получаемой функции `analogRead` представляет собой целое число в диапазоне от 0 до 1023. Таким образом, чтобы узнать напряжение на выходе сенсора, необходимо значение на аналоговом входе Arduino умножить на 0,0048828125 (5 В / 1024). А расстояние мы вычисляем по формуле:

```
distance = volts × 0,0001831 - 0,003097
```

Содержимое скетча, выдающего расстояние, измеряемое датчиком Sharp, в последовательный порт, представлено в листинге 4.7.

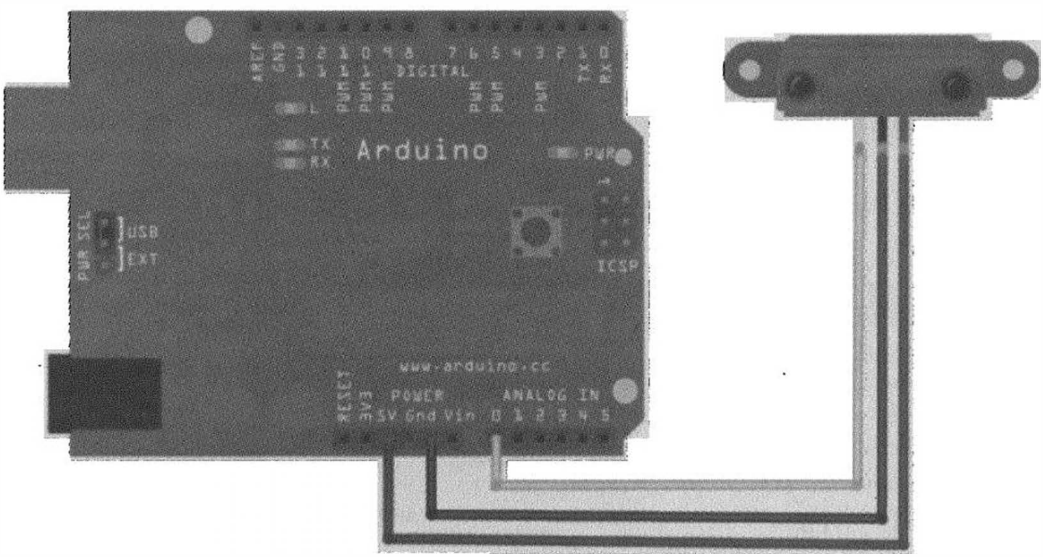


Рис. 4.29. Подключение к Arduino инфракрасного датчика расстояния Sharp GP2Y0A21YK

Листинг 4.7

```
int IRpin = 0;  // аналоговый пин для подключения выхода Vo сенсора

void setup() {
 Serial.begin(9600);   // старт последовательного порта
}

void loop() {
 // 5V/1024 = 0.0048828125
 // считываем значение сенсора и переводим в напряжение
 float volts = analogRead(IRpin)*0.0048828125;
 // и в расстояние в см
 float distance=32*pow(volts,-1.10);
 Serial.print(distance);       // выдаем в порт значение

 delay(100);
}
```

При чтении данных на каждой итерации цикла иногда для одного и того же расстояния приходят разные значения сигнала. Дело в том, что датчик передает сигнал на аналоговый порт с некоторой амплитудой, и когда итерация в момент считывания данных приходится на провал, измеренное значение оказывается отличным от реального. В листинге 4.8 представлен код скетча, осуществляющего сглаживание значений, получаемых с датчика расстояния.

Листинг 4.8

```
const int IRpin = A0;          // аналоговый пин для подключения выхода Vo
сенсора
int valuel;          // для хранения аналогового значения

void setup() {
Serial.begin(9600);       // Запуск последовательного порта
}

void loop() {
Serial.println(irRead(), DEC);
 // получаем сглаженное значение и переводим в напряжение
 float volts = analogRead(IRpin)*0.0048828125;
 // и в расстояние в см
 float distance=32*pow(volts,-1.10);
 Serial.print(distance);       // выдаем в порт значение
 delay(200);
}
```

```
// Усреднение нескольких значений для сглаживания
int irRead() {
int averaging = 0;        // переменная для суммирования данных

// Получение 5 значений
for (int i=0; i<5; i++) {
 value1 = analogRead(IRpin);
 averaging = averaging + value1;
 delay(55);    // Ожидание 55 ms перед каждым чтением
}
value1 = averaging / 5;    // усреднить значения
return(value1);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 4.8, можно найти в файле arduino_scetches_04_04_08.ino сопровождающего книгу электронного архива.

Точность измерений показаний датчика впрямую зависит от напряжения питания АЦП — естественно, для лучшего результата необходимо питать АЦП от отдельного источника калиброванного стабилизированного напряжения.

4.4.2. Подсчет количества посетителей магазина

Создадим пример отправки в сервис Twitter данных о количестве посетителей магазина за определенный промежуток времени. Упростим задачу, предполагая, что вход осуществляется через неширокую дверь, а для входа и выхода посетителей служат разные двери.

На входе поставим инфракрасный датчик Sharp GP2Y0A21YK0F с диапазоном измерений 20–150 см таким образом, чтобы при проходе человека показания имели значения от 10 до 50 см, а при отсутствии людей в просвете двери — 80 см. Каждые 10 минут данные о посещаемости будут отправляться в Twitter.

Сначала напишем код обнаружения прохода человека через дверной проем (листинг 4.9).

Листинг 4.9

```
const int IRpin = A0; // аналоговый пин для подключения выхода Vo сенсора
int value1; // для хранения аналогового значения
unsigned long timevisitors;    // время прохода
int count_visitors=0;          // переменная подсчета посетителей

void setup() {
Serial.begin(9600); // Запуск последовательного порта
}
```

```
void loop() {
// получаем сглаженное значение и переводим в напряжение
value1=irRead();
//Serial.println(value1, DEC);
if(value1>50)  // фиксация прохода
  {
  timevisitors=millis();
  while(irRead()>50) ;
  if(millis()-timevisitors>300)   // > минимального времени прохода
   {
   Serial.println("passage!!!");
   count_visitors=count_visitors+1;  // увеличение счетчика
   Serial.print("count_visitors=");
   Serial.println(count_visitors);
   }
  }
delay(200);
}
// Усреднение нескольких значений для сглаживания
int irRead() {
int averaging = 0; // переменная для суммирования данных

// Получение 5 значений
for (int i=0; i<5; i++) {
value1 = analogRead(IRpin);
averaging = averaging + value1;
delay(55); // Ожидание 55 ms перед каждым чтением
}
value1 = averaging / 5; // усреднить значения
return(value1);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 4.9, можно найти в файле arduino_scetches_04_04_09.ino сопровождающего книгу электронного архива.

Итак, датчик обнаруживает попадание человека в дверной проем. Теперь мы ожидаем завершения его прохода через проем, и если это время больше минимального времени прохода (взмах руки, пролет постороннего предмета и пр. мы отсекаем), то инкрементируем счетчик посетителей и выводим данные в последовательный порт (рис. 4.30).

Нам осталось добавить отправку в Twitter полученных данных с периодичностью в 10 минут — этим мы в следующих разделах и займемся.

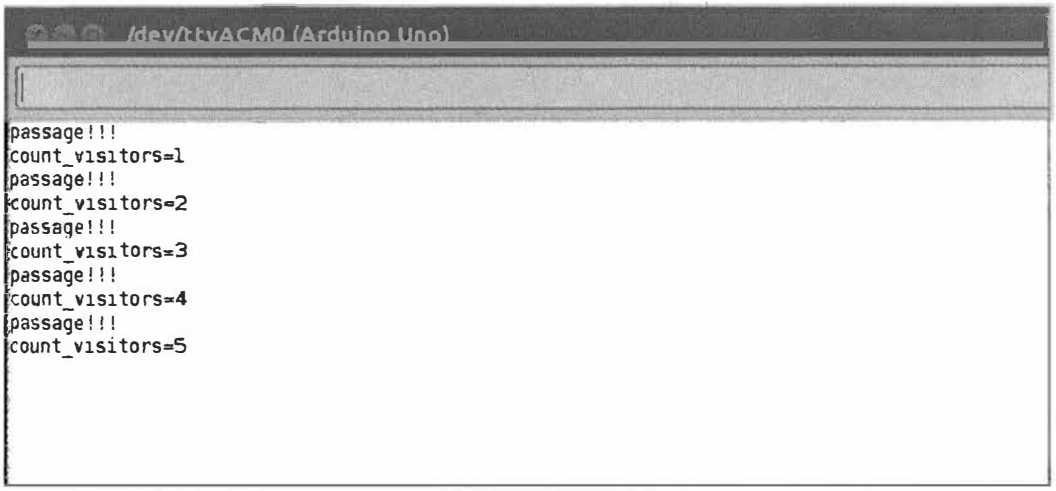


Рис. 4.30. Вывод в последовательный порт данных о проходе посетителей

4.4.3. Приложение ThingTweet сервиса ThingSpeak

Приложение ThingTweet сервиса ThingSpeak позволяет вашим устройствам послать сообщение в ваш профиль в Twitter, используя API ThingSpeak. При этом сервис ThingSpeak действует для Twitter в качестве прокси-сервера — чтобы ваши устройства отправляли сообщения в Twitter без необходимости реализовывать открытую аутентификацию (OAuth).

Что ж, авторизуемся в сервисе ThingSpeak и выберем пункт меню **Apps** — откроется окно выбора приложений (рис. 4.31).

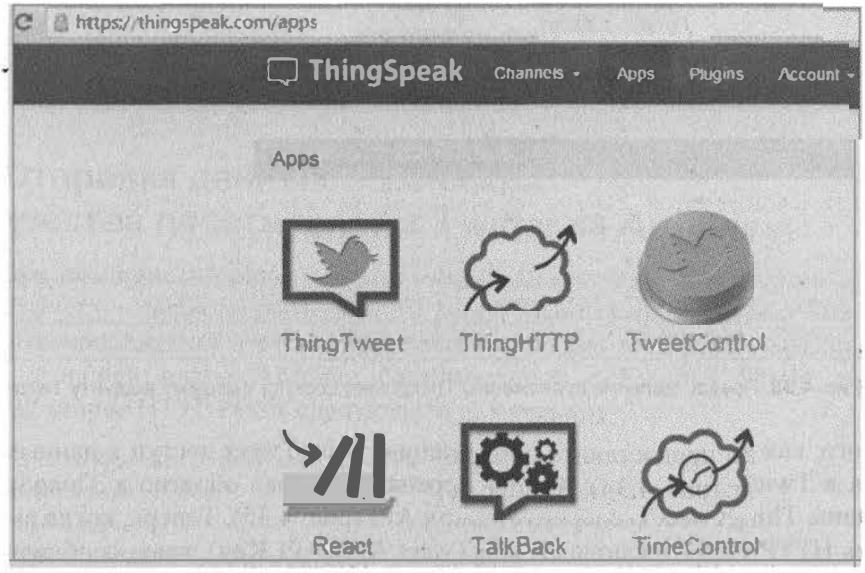


Рис. 4.31. Окно выбора приложения в сервисе ThingSpeak



Рис. 4.32. Окно перехода в аккаунт Twitter

Выбираем здесь приложение ThingTweet и попадаем на страницу со ссылкой перехода в ваш аккаунт Twitter (рис. 4.32).

В следующем окне (уже на странице Twitter) необходимо разрешить приложению ThingTweet доступ к данным вашего аккаунта Twitter, нажав на кнопку **Авторизовать** (рис. 4.33).

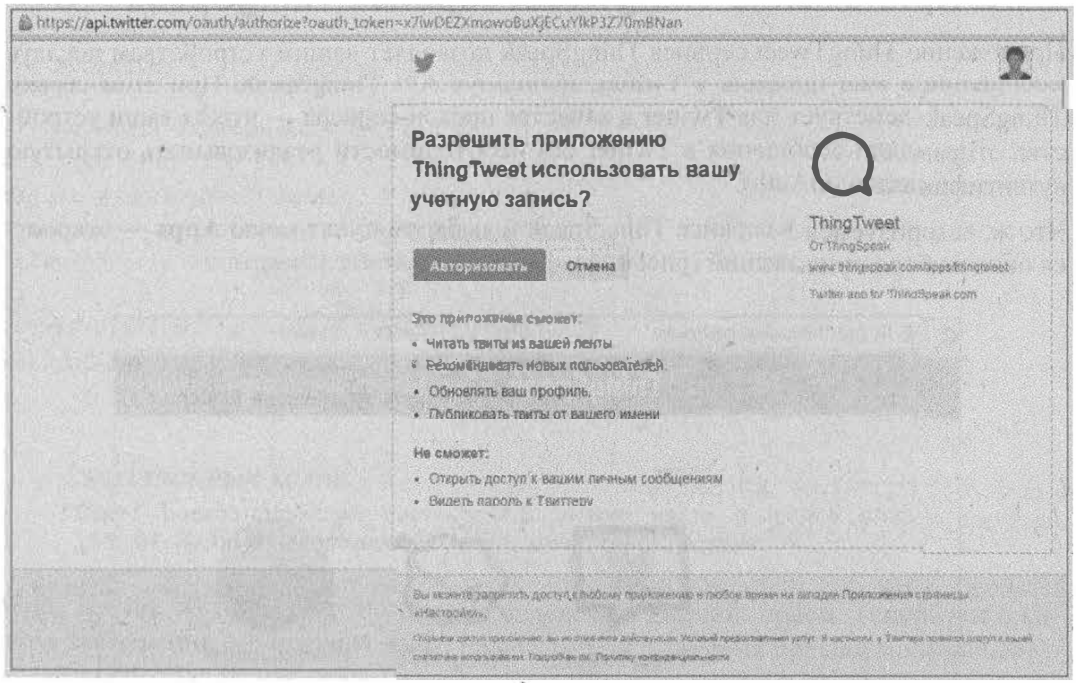


Рис. 4.33. Предоставление приложению ThingTweet доступа к вашему аккаунту Twitter

После того, как вы предоставите приложению ThingTweet доступ к данным вашего аккаунта в Twitter (рис. 4.34), Twitter перенаправит вас обратно в ThingSpeak, где приложение ThingTweet сгенерирует ключ API (рис. 4.35). Теперь, когда вы станете посылать HTTP POST с ключом ThingTweet API (API Key), ваше сообщение будет передано в Twitter.



Рис. 4.34. Доступ приложению к вашему аккаунту Twitter предоставлен

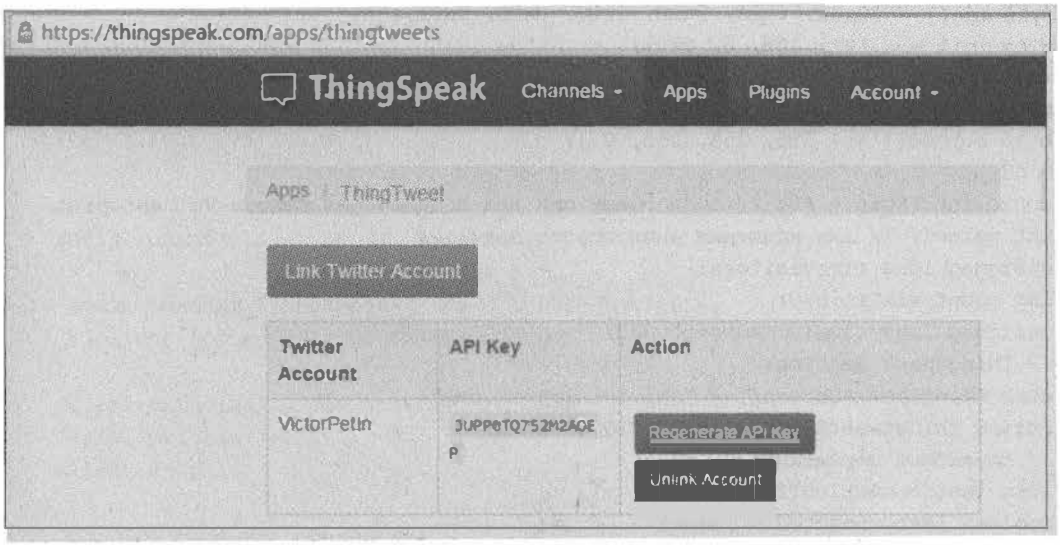


Рис. 4.35. Генерация API-ключей для перенаправления данных в Twitter

4.4.4. Отправка данных о количестве посетителей в Twitter из Arduino

Продолжим создание примера из *разд. 4.3.2*. Нам сейчас необходимо добавить в него отправку с периодичностью в 10 минут данных приложению ThingSpeak, которое перенаправит их в Twitter. Для этого один раз в 10 минут мы станем отправлять с нашей платы Arduino, состыкованной с Ethernet Shield, в сервис ThingSpeak запрос HTTP POST следующего содержания:

```
POST /apps/thingtweet/1/statuses/update HTTP/1.1
Host: api.thingspeak.com
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: <length>api_key=<thingtweet_api_key>&status=<message>
```

где:

- <length> — количество символов в сообщении;
- <thingtweet_api_key> — полученный API-ключ JUPP0TQ752M2AOEP (см. рис. 4.35);
- <message> — сообщение.

Итак, загружаем в нашу плату Arduino скетч, представленный в листинге 4.10.

Листинг 4.10

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
byte ip[] = { 192, 168, 0, 97 };
byte sdns[] = { 192, 168, 1, 1 };
byte gateway[] = { 192, 168, 0, 28 };
byte subnet[] = { 255, 255, 255, 0 };

const int IRpin = A0; // аналоговый пин для подключения выхода Vo сенсора
int value1; // для хранения аналогового значения
unsigned long timevisitors;
int count_visitors=0;
unsigned long time10minute=500000;
// ThingSpeak Settings
char thingSpeakAddress[] = "api.thingspeak.com";
String thingtweetAPIKey = "JUPP0TQ752M2AOEP";
// служебные переменные
long lastConnectionTime = 0;
boolean lastConnected = false;
int failedCounter = 0;
// инициализация Arduino Ethernet Client
EthernetClient client;

void setup()
  {
  Serial.begin(9600); // Запуск последовательного порта
  startEthernet();
  delay(1000);
  }

void loop()
 {
 value1=irRead();
 if(value1>50)
   {
   timevisitors=millis();
```

```
   while(irRead()>50) ;
   if(millis()-timevisitors>300)
     {
     Serial.println("passage!!!");
     count_visitors=count_visitors+1;
     Serial.print("count_visitors=");
     Serial.println(count_visitors);
     }
   }
 delay(200);
 if(millis()-time10minute>600000)
  {
  String message="last 10 minute "+String(count_visitors)+" visitors";
  updateTwitterStatus(message);
  time10minute=millis();
  }
   // печать ответа от сервера в монитор
  while (client.available())
    {
    char c = client.read();
    Serial.print(c);
    }
  // отключение от ThingSpeak
  if (!client.connected() && lastConnected)
   {
    Serial.println("...disconnected");
    Serial.println();
    client.stop();
   }
  // рестарт сетевого соединения
  if (failedCounter > 3 ) {startEthernet();}
  lastConnected = client.connected();
}

// Усреднение нескольких значений для сглаживания
int irRead()
 {
 int averaging = 0; // переменная для суммирования данных

 // Получение 5 значений
 for (int i=0; i<5; i++)
  {
  value1 = analogRead(IRpin);
  averaging = averaging + value1;
  delay(55); // Ожидание 55 ms перед каждым чтением
  }
```

```
 value1 = averaging / 5; // усреднить значения
 return(value1);
 }

void startEthernet()
 {
  client.stop();

  Serial.println("Connecting Arduino to network...");
  Serial.println();

  delay(1000);
  Ethernet.begin(mac, ip, sdns, gateway, subnet);
  Serial.println(Ethernet.localIP());

  // Connect to network amd obtain an IP address using DHCP
  Ethernet.begin(mac, ip, sdns, gateway, subnet);
  Serial.println(Ethernet.localIP());

  delay(1000);
}

void updateTwitterStatus(String tsData)
 {
  if (client.connect(thingSpeakAddress, 80))
   {
    // данные HTTP POST D
    tsData = "api_key="+thingtweetAPIKey+"&status="+tsData;

    client.print("POST /apps/thingtweet/1/statuses/update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");

    client.print(tsData);
    lastConnectionTime = millis();
    if (client.connected())
     {
      Serial.println("Connecting to ThingSpeak...");
      Serial.println();
      count_visitors=0;
      failedCounter = 0;
     }
```

```
    else
     {
      failedCounter++;
      Serial.println("Connection to ThingSpeak failed ("+String(failedCounter,
                                                              DEC)+")");
      Serial.println();
     }
   }
  else
   {
    failedCounter++;
    Serial.println("Connection to ThingSpeak Failed ("+String(failedCounter,
                                                              DEC)+")");
    Serial.println();
    lastConnectionTime = millis();
  }
}
```

Электронный архив

Скетч, соответствующий листингу 4.10, можно найти в файле arduino_scetches_04_04_10.ino сопровождающего книгу электронного архива.

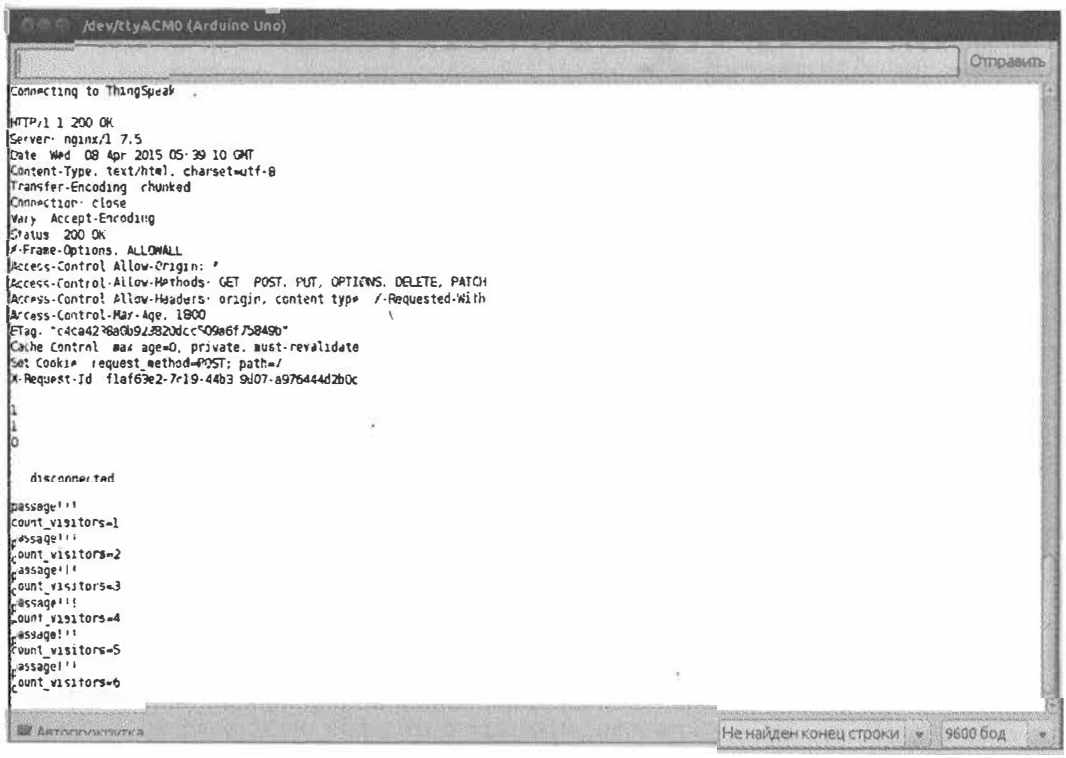


Рис. 4.36. Отображение процесса подсчета посетителей и отправки данных в мониторе последовательного порта

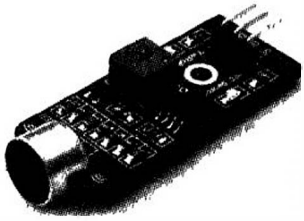
Здесь при загрузке программы мы прежде всего устанавливаем настройки сети для платы Ethernet Shield, после чего в основном цикле программы ведем по срабатыванию инфракрасного датчика подсчет посетителей в переменной `count_visitors` и по прошествии 10 минут `(millis()-time10minute>10*60*1000)` отправляем серверу `api.thingspeak.com` HTTP POST с сообщением `"last 10 minute xx visitors"`. Затем обнуляем счетчик посетителей `count_visitors` и опять ждем 10 минут для отправки следующих данных. После трех неудачных попыток соединения с сервером переподключаем Ethernet Shield к Сети.

На рис. 4.36 показано отображение в мониторе последовательного порта Arduino IDE процесса подсчета посетителей и отправки данных в ThingSpeak, а на рис. 4.37 — публикация этих сообщений в Twitter.



Рис. 4.37. Публикация отправляемых из Arduino данных в Twitter

ГЛАВА 5



Использование Arduino в качестве контроллера исполнительных устройств

Концепция Internet of Things (Интернет вещей) предполагает не только удаленное получение данных о состоянии объектов, но и удаленное управление исполнительными устройствами.

> ***Исполнительные устройства***
>
> Исполнительные устройства — это элементы автоматики, создающие управляющее воздействие на объект управления. Они изменяют положение или состояние регулирующего органа объекта управления таким образом, чтобы управляемый параметр соответствовал заданному значению. Исполнительное устройство, или механизм (actuator), преобразует электрическую энергию в механическую для воздействия на управляемый процесс.
>
> В качестве исполнительных механизмов могут быть задействованы световые и звуковые устройства, электромагнитные клапаны, электродвигатели постоянного (DC) и переменного (AC) тока, сервоприводы, релейные системы и многое другое.

Плата Arduino, имеющая доступ к сети с помощью, например, платы Ethernet Shield, вполне способна выступать в роли контроллера удаленных исполнительных устройств. В этой главе мы рассмотрим использование платы Arduino для организации веб-сервера, получающего команды управления по HTTP и преобразующего их в команды управления исполнительными устройствами, подключенными к цифровым выводам Arduino.

5.1. Arduino и электромагнитное реле

5.1.1. Электромагнитное реле

Реле́ (от фр. relais) — электрическое или электронное устройство (ключ), предназначенное для замыкания и размыкания различных участков электрических цепей. Изобретение реле стало великим открытием, перевернувшим мир. Произошло это в далеком 1831 году. Позже, в 1937 году, Сэмюэл Морзе изобрел коммуникационное реле и применил его в телеграфном аппарате. Сегодня электрические, пневматиче-

ские и механические реле используются весьма широко и выпускаются в огромном количестве самых разнообразных их видов.

Наиболее распространенными являются электромагнитные реле — электромеханические устройства, замыкающие и/или размыкающие контакты внешней электрической цепи при подаче в обмотку реле управляющего электрического тока. Этот ток порождает магнитное поле, вызывающее перемещение ферромагнитного якоря реле, механически связанного с электрическими контактами внешней электрической цепи. Последующее перемещение контактов коммутирует эту цепь.

К реле можно подключить лампочку, вентилятор, электромагнитный клапан для управления поливом и программно управлять этими устройствами изменением состояния на цифровых выводах Arduino.

5.1.2. Устройство и принцип работы электромагнитного реле

Рассмотрим устройство электромагнитного реле на примере широко используемого в связке с Arduino реле SRD-05VDC фирмы SONGLE (рис. 5.1). Это реле управляется напряжением 5 вольт и способно коммутировать до 10 ампер постоянного тока напряжением 30 В и переменного тока напряжением 250 В.

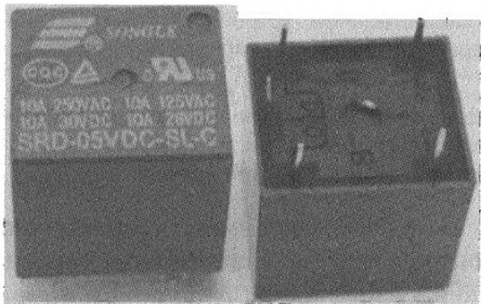


Рис. 5.1. Реле SRD-05VDC

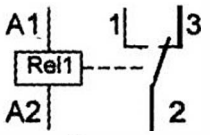


Рис. 5.2. Схема электромагнитного реле

Реле имеет две раздельных цепи, никак не связанные между собой: цепь управления, представленную контактами A1 и A2, и управляемую цепь с контактами 1, 2 и 3 (рис. 5.2). Между контактами A1 и A2 расположен металлический сердечник, при протекании тока по обмотке которого к нему притягивается подвижный якорь 2. Контакты 1 и 3 закреплены неподвижно. Стоит отметить, что якорь подпружинен, и пока мы не пропустим ток через обмотку сердечника, якорь будет удерживаться прижатым к контакту 3. При подаче в обмотку сердечника управляющего тока, сердечник, как уже говорилось, превращается в электромагнит, и якорь прижимается к контакту 1. При обесточивании обмотки сердечника пружина снова возвращает якорь к контакту 3.

5.1.3. Подключение реле к Arduino

При подключении реле к Arduino возникает одна немаловажная проблема — контакт микроконтроллера не может обеспечить мощность, необходимую для нормальной работы катушки реле. Поэтому нам необходимо усилить управляющий ток — что решается включением транзистора в схему управления реле. Удобнее всего для усиления применять n-p-n-транзистор, включенный по схеме с общим эмиттером (ОЭ), — как показано на рис. 5.3. При таком способе можно подключать нагрузку с напряжением питания бóльшим, чем питание микроконтроллера. Резистор на базе транзистора — ограничительный. Его номинал может варьироваться в широких пределах (1–10 кОм) — в любом случае транзистор будет работать в режиме насыщения.

В этой схеме может быть применен любой n-p-n-транзистор, поскольку коэффициент его усиления практически не имеет значения — достаточно подобрать транзистор таким образом, чтобы он соответствовал нужному нам току коллектора и напряжению коллектор-эмиттер (напряжение, которым запитывается нагрузка). На основании личного опыта могу порекомендовать для этой схемы транзистор C945.

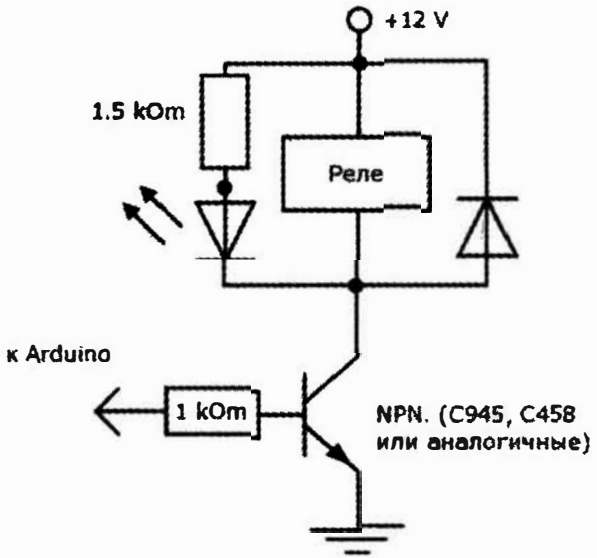


Рис. 5.3. Схема подключения реле к Arduino (p-канальное управление)

Для включения реле, подключенного по схеме с ОЭ, на вывод Arduino необходимо подать 1, для выключения — 0. Скетч, обеспечивающий работу схемы с реле, приведен в листинге 5.1.

Листинг 5.1

```
int relayPin = 10;          // подключение Arduino к выводу D10
void setup()
```

```
  {
  //
  pinMode(relayPin, OUTPUT);  // настроить вывод как выход (OUTPUT)
  }

// the loop function runs over and over again forever
void loop()
  {
  digitalWrite(relayPin, HIGH);   // включить реле
  delay(5000);
  digitalWrite(relayPin, LOW);    // выключить реле
  delay(5000);
  }
```

Существуют и готовые модули для Arduino, содержащие сразу несколько реле с необходимой обвязкой (рис. 5.4). Однако в таких модулях обычно используется n-канальное управление (рис. 5.5), при котором реле включается подачей на вывод Arduino низкого уровня.

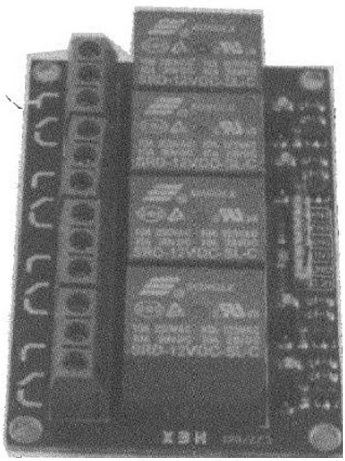


Рис. 5.4. Модуль из 4 реле

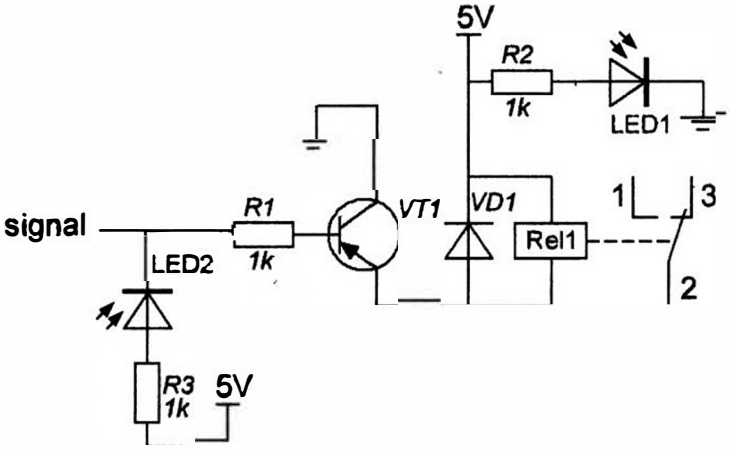


Рис. 5.5. Схема подключения реле к Arduino (n-канальное управление)

5.2. Arduino и твердотельное реле

Твердотельные реле (Solid State Relay, SSR) применяют в промышленном оборудовании — там, где нужны большая надежность и малые габариты (рис. 5.6). Во всех твердотельных оптоэлектронных реле коммутация цепей нагрузки осуществляется бесконтактно — за счет управления встроенными полупроводниковыми элементами. Как правило, это тиристоры или симисторы (для коммутации переменного тока) и транзисторы (для постоянного тока).

Такая конструкция дает твердотельным реле ряд преимуществ перед обычными электромагнитными — ведь как и в обычных реле, в твердотельных существует гальваническая развязка между напряжением катушки и напряжением на силовых контактах. Только в электромеханических реле это достигается за счет разнесения в пространстве, а в твердотельных — за счет оптической развязки, поскольку на входе реле установлен оптрон.

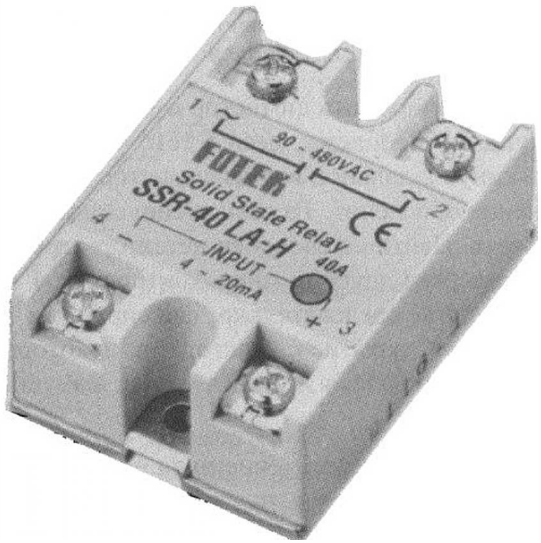


Рис. 5.6. Твердотельное реле

Твердотельные реле потребляют и теряют гораздо меньше энергии при работе, имеют меньшие габариты, высокое быстродействие, гораздо более длительный срок службы и все это — абсолютно бесшумно! Но есть у них и серьезный недостаток — высокая цена.

Для подключения твердотельного реле к Arduino минусовой контакт управляющей сети подсоединяется к «земле», а плюсовой контакт — к цифровому выводу Arduino. Выходного тока контакта Arduino для срабатывания твердотельного реле вполне достаточно.

5.3. Arduino и диммер

5.3.1. Диммер

Переключение нагрузки переменного тока с помощью Arduino довольно просто: используется либо механическое реле, либо твердотельное реле с оптически изолированным симистором. Становится сложнее, если необходимо, используя Arduino, плавно уменьшать (диммировать) яркость лампы переменного тока: просто ограничивать ток симистором не представляется возможным из-за необходимости в мощном симисторе и, как следствие, необходимости рассеивания большого количества тепла. Неэффективно это также и с точки зрения использования энергии. Правильный способ реализации этой задачи — применение регулирования фазы: симистор полностью открыт, но только в одной части синусоидальной волны переменного тока.

> ***Пояснение***
>
> *Диммером* называется устройство, обычно используемое для регулировки яркости свечения ламп накаливания или светодиодов.

Можно, конечно, и просто открывать симистор с помощью Arduino на некоторое количество микросекунд, но проблема здесь в том, что невозможно заранее предсказать, в какой части синусоидальной волны симистор откроется и, следовательно, какой уровень затемнения обеспечит. То есть, в синусоидальной волне нам необходимо иметь точку отсчета. Найти ее способен детектор пересечения нуля — схема, которая сообщает Arduino (или другому микроконтроллеру), когда синусоидальная волна проходит через нуль и, следовательно, фиксирует на этой синусоидальной волне определенную точку. Открытие симистора на некоторое количество микросекунд, начиная от пересечения нуля, как раз и даст нам предсказуемый уровень затемнения.

Сделать такую схему несложно: пересечение нуля берется непосредственно из выпрямленного сетевого переменного тока, что и дает сигнал каждый раз, когда волна проходит через нуль. Поскольку синусоида сначала проходит двухфазное выпрямление, сигнал пересечения нуля подается независимо от того, вверх или вниз идет синусоидальная волна. Этот сигнал и может быть использован для вызова прерывания Arduino.

Схема диммера изображена на рис. 5.7. Сетевое напряжение 220 В через два резистора по 30 кОм идет к мостовому выпрямителю, который выдает двухфазный выпрямленный сигнал на оптрон 4N25. Светодиод в этом оптроне при низком уровне работает на частоте 100 Гц, а на коллектор выходит сигнал высокого уровня с частотой 100 Гц в соответствии с синусоидальной волной. Сигнал с 4N25 подается на прерывающий вывод Arduino (или другого микропроцессора). Программа прерываний дает сигнал определенной длины на один из портов ввода/вывода. Сигнал с порта ввода/вывода уходит в нашу схему и открывает светодиод в симисторном оптроне MOC3021, который запускает оптотиристор.

Для затемнения реально пригодны только обычные лампы накаливания. Впрочем, наша схема также будет работать и с галогенными лампами, но это сократит срок их службы. А вот с компактными люминесцентными лампами (КЛЛ) схема работать не будет — если только они не выполнены специально с возможностью диммирования.

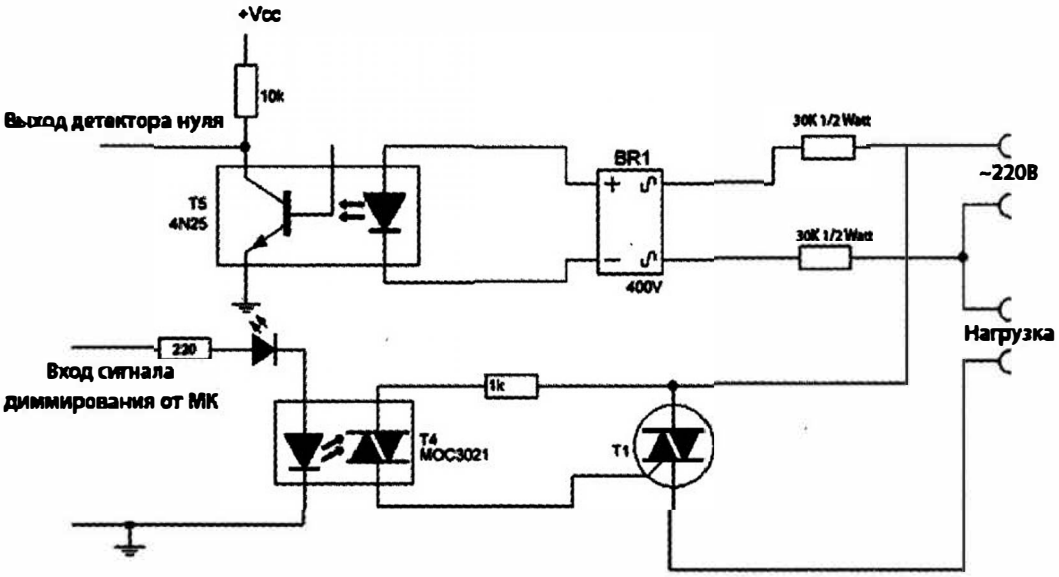


Рис. 5.7. Схема реализации диммера

5.3.2. Подключение диммера к Arduino

Подключим к плате Arduino диммер (рис. 5.8), приобретенный мной на сайте **http://carduino.ru**. Схема соединений показана на рис. 5.9: выход детектора нуля подсоединен к цифровому входу D2 платы Arduino — им мы воспользуемся для вызова прерывания, а цифровой вход D4 служит в качестве входа диммирования сигнала микроконтроллером.

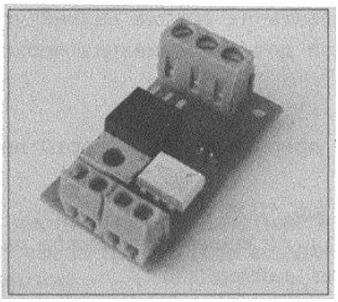


Рис. 5.8. Диммер

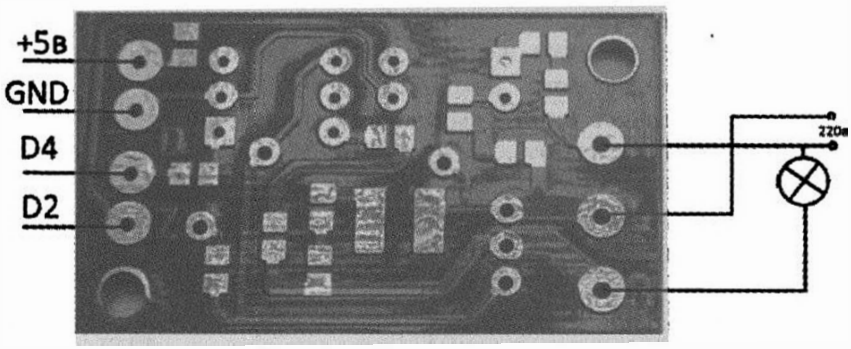


Рис. 5.9. Подключение диммера к Arduino

5.3.3. Скетч управления диммером

Напишем теперь скетч управления яркостью подключенной к диммеру лампы посредством отправки команд в последовательный порт Arduino. В листинге 5.2 приведен пример такого скетча, взятый у разработчика диммера, примененного в *разд. 5.3.2*.

В скетче используется библиотека Cyberlib.h, которую можно найти на сайте **http://cyber-place.ru**. Библиотека представляет набор макросов для низкоуровневой работы с портами Arduino, что дает существенный прирост скорости записи/чтения цифровых портов по сравнению со встроенными функциями Arduino, а также уменьшает размер используемой памяти.

Листинг 5.2

```
#include <CyberLib.h>              // Библиотека от Cyber-Place.ru
volatile uint8_t tic, Dimmer1=255;
uint8_t data;

void setup()
  {
  D4_Out;          // настраиваем порт на выход
  D4_Low;          // установить на выходе низкий уровень сигнала
  // настраиваем порт на вход для отслеживания прохождения сигнала
  // через ноль
  D2_In;
  // настроить срабатывание прерывания interrupt0 на pin 2 на
  // низкий уровень
  attachInterrupt(0, detect_up, LOW);
  StartTimer1(halfcycle, 40); // время для одного разряда ШИМ
  StopTimer1(); // остановить таймер
  UART_Init(57600); // инициализация порта
  }
```

```
// обработчики прерываний
void halfcycle()  // прерывания таймера
  {
  tic++;  // счетчик
  if(Dimmer1 < tic ) D4_High; // управляем выходом
  }

void  detect_up()  // обработка внешнего прерывания: сработает по переднему
                   // фронту
  {
 tic=0;             // обнулить счетчик
 ResumeTimer1();    // запустить таймер
 // перепрограммировать прерывание на другой обработчик
 attachInterrupt(0, detect_down, HIGH);
 }

void  detect_down()  // обработка внешнего прерывания: сработает по заднему
                     // фронту
{
 StopTimer1(); // остановить таймер
 D4_Low;       // логический ноль на выходы
 tic=0;        // обнулить счетчик
 // перепрограммировать прерывание на другой обработчик
attachInterrupt(0, detect_up, LOW);
}

void loop()
{
  Start
  if (UART_ReadByte(data))
 {
   if(data>47 && data<59)
    {
      data=225-(data-48)*25;
      Dimmer1=data;
    }
 }
 End
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.2, можно найти в файле arduino_scetches_05_05_02.ino, а библиотеку Cyberlib.h — в папке arduino_libraries\CyberLib сопровождающего книгу электронного архива.

5.4. Arduino и сервоприводы

Сервопривод (он же «серва», servo, рулевая машинка) — устройство, обеспечивающее преобразование сигнала в строго соответствующее этому сигналу перемещение (как правило — поворот) исполнительного устройства. Представляет собой прямоугольную коробку с мотором, схемой, редуктором и выходным валом, который может поворачиваться на строго фиксированный угол, определяемый входным сигналом. Как правило, этот угол имеет предел в 60 градусов, а иногда и в 180. Бывают сервоприводы и постоянного вращения.

На вал сервопривода надевается рычаг в форме круга, крестовины или перекладинки для передачи вращающего движения на рабочий орган. Выполнив поворот, вал остается в том же положении, пока не придет иной управляющий сигнал. Смысл сервопривода заключен в гарантированном выполнении заданной команды — если внешняя сила не позволит выполнить поворот на нужный угол, сервопривод все равно закончит движение по окончании действия мешающего внешнего воздействия. Воспрепятствовать этому может лишь разрушение сервопривода, снятие внешнего управляющего сигнала или пропадание напряжения питания.

Большинство сервоприводов внешне похожи друг на друга (рис. 5.10) — как правило, это прямоугольный корпус (внутри которого прячутся мотор, шестерни и управляющая схема) с крепежными ушками по бокам и выходным валом, расположенным на верхней крышке. К валу, как уже отмечалось, крепится сменный (идущий в комплекте) передаточный элемент в виде диска с отверстиями по кругу, крестовины с отверстиями на концах всех его перекладин или рычага, насаженного на выходной вал либо своим центром, либо одним из концов. Чаще всего этот сменный элемент белого цвета.

Иногда сервопривод имеет форму цилиндра — тогда его верхняя часть при работе поворачивается вокруг своей оси. Некоторые сервоприводы вместо выходного вала оснащаются шкивом (катушкой). На такой шкив наматывается (или с него сматывается) тросик, с помощью которого управляют каким-то внешним приспособлением — например, парусом в моделях судов или яхт.



Рис. 5.10. Сервопривод

5.4.1. Принципы управления сервоприводами

Сервопривод — электрическое исполнительное устройство, подключаемое с помощью трех проводов к устройству управляющему (контроллеру) и источнику питания.

По способу управления сервоприводы подразделяются на аналоговые и цифровые. *Аналоговые* управляются аналоговым сигналом — буквально частотой, параметры которой задаются с помощью широтно-импульсной модуляции (ШИМ). *Цифровые* сервоприводы управляются цифровым сигналом, представляющим собой кодовые команды, передаваемые по последовательному интерфейсу. Аналоговые сервоприводы намного дешевле цифровых.

Управление сервоприводом осуществляется с помощью импульсов переменной длительности, посылаемых по сигнальному проводу. Параметрами этих импульсов являются минимальная длительность, максимальная длительность и частота повторения. Угол поворота сервопривода определяется длительностью посылаемого на него импульса (это и называется *широтно-импульсной модуляцией*). Так, импульс в 1,5 миллисекунды (1,5 мс) диктует мотору поворот на 90 градусов, что и соответствует его нейтральному положению. Ожидает сервопривод импульса каждые 20 мс.

Нейтральное положение сервопривода определяется как положение, в котором сервопривод обладает одинаковым потенциалом вращения в обоих направлениях. Важно отметить, что различные сервоприводы обладают разными ограничениями в своем вращении, но они все имеют нейтральное положение, и это положение всегда находится в районе длительности импульса в 1,5 мс.

Когда сервопривод получает команду на перемещение, его управляющий орган перемещается в это положение и удерживает его. Если внешняя сила действует на сервопривод, когда он удерживает заданное положение, сервопривод будет сопротивляться перемещению из этого положения. Максимальная величина силы, которую может выдерживать сервопривод, характеризует вращающий момент сервопривода. Однако сервопривод не навсегда сохраняет свое положение — импульсы позиционирования должны повторяться, информируя сервопривод о необходимости сохранения положения.

Когда импульс, посылаемый на сервопривод, оказывается короче 1,5 мс, сервопривод поворачивает выходной вал на несколько градусов против часовой стрелки и удерживает это положение. Если же приходит импульс шире, чем 1,5 мс, выходной вал поворачивается на несколько градусов в противоположном направлении. Минимальная и максимальная ширина импульса, который управляет сервоприводом, является свойством конкретного сервопривода. Различные марки, и даже различные сервоприводы одной марки, обладают различным минимумом и максимумом. Как правило, ширина минимального импульса составляет примерно 1 мс, а ширина максимального — 2 мс.

Различаются сервоприводы и габаритами. Существуют так называемые *стандартные* сервоприводы. Их габариты и вес в общем модельном ряду соответствуют некоторым средним значениям. Они самые дешевые — в пределах 10–20 долларов.

При уменьшении или увеличении размеров сервопривода в сторону от стандартных значений цена сервопривода возрастает пропорционально величине этих отклонений. Как и самые маленькие (микросервы), так и самые большие супермощные сервоприводы, — это самые дорогие устройства, цена их может доходить до сотен долларов.

Сервоприводы различаются также материалом шестеренок. Самые дешевые сервоприводы оснащены шестернями из пластмассы. Более дорогие — с одной выходной шестерней из металла. Самые дорогие — с металлическими шестернями. Соответственно виду материала изменяется нагрузочная способность сервопривода. Самый слабый сервопривод — с пластиковыми шестернями, самый мощный — с металлическими.

Различия сервоприводов определяются и типом подшипников. Наиболее дешевые не имеют подшипников вовсе: пластмассовые шестерни на пластмассовых валах крутятся в отверстиях пластмассовых пластин, соединяющих шестерни в единый редуктор, — это самые недолговечные сервоприводы. Более дорогие и, соответственно, более долговечные сервоприводы имеют металлическую — обычно, латунную — втулку на выходном валу. Сервы подороже имеют на выходном валу, на который приходится самая большая нагрузка, настоящий подшипник: шариковый или роликовый. Шариковый — дешевле, роликовый — компактнее и легче. В еще более дорогих сервоприводах на всех (металлических!) шестернях стоят подшипники. Это — самые долговечные и надежные устройства.

Сервоприводы различаются и по толщине. Она может сильно варьироваться при одинаковых размерах по высоте и длине. Чем меньше толщина, тем выше цена, поскольку в узком корпусе труднее разместить шестерни.

Наконец, сервоприводы различаются по фирме-производителю. Наиболее раскрученные бренды продают самые дорогие сервоприводы. При этом в их ассортименте будут как дорогие, так и дешевые сервоприводы, но даже самый простенький и дешевый стандартный сервопривод крупного бренда стоит дороже, а иногда и существенно дороже, чем аналогичный сервопривод с наклейкой менее раскрученного имени и, тем более, с именем никому не известной фирмы.

Очень мощные и очень дорогие сервоприводы могут иметь любую внешнюю форму, определяемую назначением сервопривода. Так, сервоприводы для андроидов (человекоподобных роботов) могут быть шаровидными или дисковидными. Но обычно сервопривод — это все-таки черный параллелепипед с белой фитюлькой на верхней крышке.

Сервоприводы применяются в основном в промышленности (станки с числовым программным управлением и т. п.). В них установлены, как правило, мощные двигатели, в том числе и шаговые, что позволяет использовать их в автоматических манипуляторах (в промышленных роботах).

Сервоприводы для моделирования имеют гораздо более худшие характеристики, чем промышленные. Основным их параметром является момент сил, приложенных к «качалке» на заданном расстоянии от оси. Обычно такой момент измеряется в кг/см. Самые слабые сервоприводы тянут 1–2 кг/см. Самые мощные непромыш-

ленные сервоприводы — более 100 кг/см. Момент этот также зависит от питающего напряжения.

Немаловажным параметром является и скорость вращения вала сервопривода, как правило, измеряемая в сек/60°. Скорость сервопривода обычно важна, если он применяется совместно с гироскопом или акселерометром (например, в моделях вертолетов).

5.4.2. Управление сервоприводом с помощью Arduino

Для управления сервоприводом в Arduino имеется стандартная библиотека Servo. Однако на платах, отличных от Mega, задействование этой библиотеки отключает возможность использования функции `analogWrite()` (ШИМ) на контактах 9 и 10 (вне зависимости, подключены к этим контактам сервы или нет). В то же время на платах Mega без влияния на функциональность ШИМ могут использоваться до 12 сервоприводов, но дальнейшее увеличение количества сервомашинок до 23-х отключит ШИМ на контактах 11 и 12.

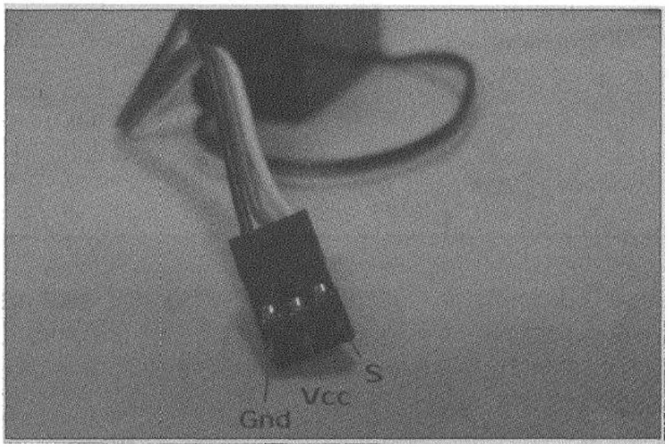


Рис. 5.11. Подключение сервопривода

Подключается сервопривод к плате Arduino тремя проводами (рис. 5.11): Vcc (питание), Gnd («земля») и S (сигнальный). Красный провод (питание) может быть подключен на плате Arduino к выводу +5 В. Черный или коричневый провод («земля») подключается к выводу Arduino GND, оранжевый, желтый или белый сигнальный провод подключается к цифровому выводу контроллера Arduino. Следует отметить, что мощные сервоприводы могут создавать большую нагрузку — в этом случае их следует запитывать отдельно (не через выход +5 В Arduino). То же самое верно и для случая подключения сразу нескольких сервоприводов.

Итак, подключаем сервопривод к Arduino по схеме, показанной на рис. 5.12, и загружаем скетч, представленный в листинге 5.3.

Листинг 5.3

```
#include <Servo.h>  // подключение библиотеки Servo
Servo myservo;      // создание экземпляра объекта Servo
int pos = 0;        // переменная для хранения текущей позиции сервопривода

void setup()
  {
  myservo.attach(9);      // подключает переменную servo к выходу 9
  }

void loop()
  {
  for(pos = 0; pos <= 180; pos += 1)  // вращение в одну сторону с шагом 1
   {
    myservo.write(pos);               // переместить в новую позицию
    delay(15);                        //  пауза 15 ms для перемещения сервы
   }
  for(pos = 180; pos>=0; pos-=1)      // вращение в одну сторону с шагом 1
  {
    myservo.write(pos
    delay(15);
  }
}
```

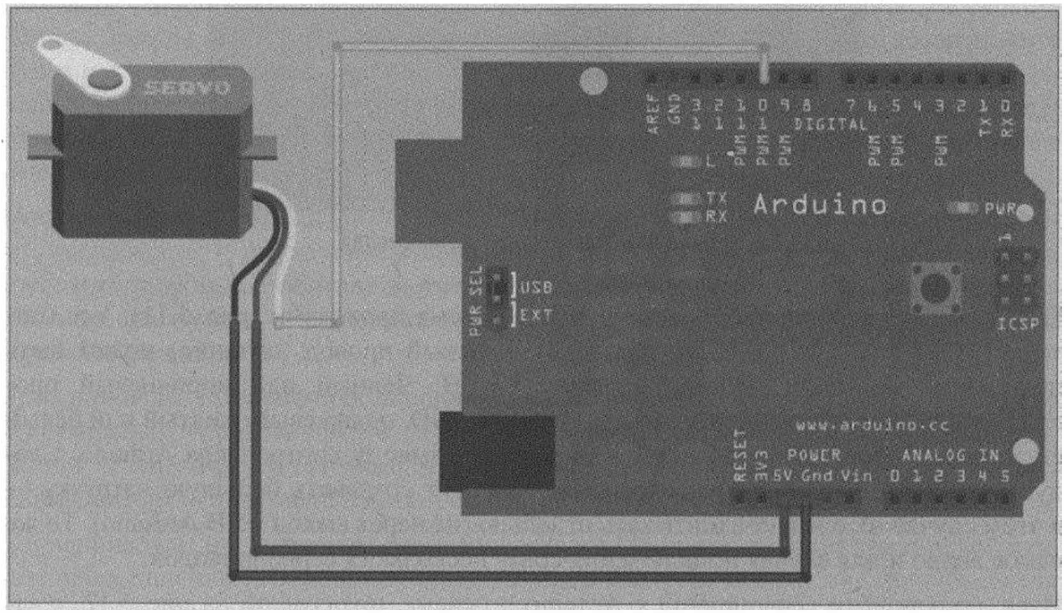


Рис. 5.12. Схема подключения сервопривода

5.5. Arduino и библиотека TinyWebServer

Библиотека TinyWebServer является небольшой и расширяемой реализацией HTTP-сервера, предназначенного для работы в ограниченном количестве пространства памяти плат Arduino. Для подключения к сети он использует плату Ethernet Shield/W5100, состыкованную с платой Arduino в обычном порядке.

С помощью библиотеки TinyWebServer логика работы контроллера теперь может быть полностью отделена от отображения страниц. При этом Arduino не обязательно формировать страницы полностью — веб-страницы, изображения и другой контент может быть скопирован пользователем вручную на SD-карту или загружен через сервер HTTP.

Файлы библиотеки TinyWebServer можно скачать со страницы: **https://github.com/ovidiucp/TinyWebServer**. Работа библиотеки TinyWebServer зависит от внешней библиотеки Flash версии 5.0, файлы которой можно скачать со страницы: **http://arduiniana.org/libraries/flash/**. При использовании новых версий Arduino IDE (новее, чем 1.5) необходимо изменить файл Flash.h, включив в него сразу после `#include <AVR / pgmspace.h>` следующие строки:

```
#if ARDUINO >= 150
typedef char prog_char __attribute__((__progmem__));
#endif
```

Для использования библиотеки TinyWebServer необходимо в скетч включить следующие строки:

```
#include <Ethernet.h>
#include <Flash.h>
#include <SD.h>
#include <TinyWebServer.h>
```

Функционал библиотеки TinyWebServer осуществляет класс `TinyWebServer`. Метод конструктора принимает два аргумента: первый — список обработчиков, когда клиентом HTTP запрашиваются определенные адреса, второй — список имен заголовков HTTP, для которых необходима реализация обработчиков. В листинге 5.4 представлен пример реализации библиотеки TinyWebServer с одним обработчиком.

Листинг 5.4

```
// MAC-адрес сетевой платы
static uint8_t mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// обработчик для страницы "/" (вывод на страницу "Hello World!")
boolean index_handler(TinyWebServer& web_server)
   {
   web_server.send_error_code(200);
   web_server << F("<html><body><h1>Hello World!</h1></body></html>\n");
   return true;
  }
```

```
// список заголовков и привязанных к ним обработчиков
TinyWebServer::PathHandler handlers[] = {
  // Register the index_handler for GET requests on /
  {"/", TinyWebServer::GET, &index_handler },
  {NULL}, // The array has to be NULL terminated this way
};
// создать веб-сервер
TinyWebServer web = TinyWebServer(handlers, NULL);

void setup()
 {
  Serial.begin(115200);
  EthernetDHCP.begin(mac);
  web.begin();
  }
void loop()
  {
  web.process();
 }
```

Здесь в цикле `loop()` для отслеживания HTTP-запросов мы должны вызвать метод веб-сервера `process()`. При отсутствии запросов этот метод осуществляет возврат, а при поступлении запроса он блокирует выполнение цикла, пока запрос не будет обработан.

5.5.1. Использование файлов с SD-карты для формирования веб-страниц

На плате Ethernet Shield/W5100 размещен слот для флеш-карты формата MicroSD, на которую пользователем могут быть записаны те или иные файлы. TinyWebServer позволяет задействовать эти файлы для формирования веб-страниц, что избавляет нас от необходимости использовать для этого память Arduino.

В листинге 5.5 представлен код обработчика для выдачи веб-сервером страницы, содержимое которой находится в файле на SD-карте.

Листинг 5.5

```
boolean file_handler(TinyWebServer& web_server) {
  char* filename = TinyWebServer::get_file_from_path(web_server.get_path());
  if (!filename) {
    web_server.send_error_code(404);
    web_server << "Could not parse URL";
  } else {
    TinyWebServer::MimeType mime_type
      = TinyWebServer::get_mime_type_from_filename(filename);
```

```
    web_server.send_error_code(mime_type, 200);
    if (file.open(filename, O_READ)) {
      web_server.send_file(file);
      file.close();
    } else {
      web_server << "Could not find file: " << filename << "\n";
    }
    free(filename);
  }
  return true;
}
```

Список имен заголовков для этого обработчика следующий:

```
TinyWebServer::PathHandler handlers[] = {
  {"/" "*", TinyWebServer::GET, &file_handler },
  {NULL},
}
```

Запись `"/" "*"` указывает, что обработчик вызывается для любого адреса (URL), следующего после `"/"`. При запросе `/content1` на веб-страницу будет выводиться содержимое файла content1 либо сообщение **Could not find file:content1** при его отсутствии на SD-карте.

5.5.2. Включение/выключение реле с веб-страницы

Напишем скетч для удаленного включения/выключения нескольких реле, подсоединенных к плате Arduino, состыкованной с Ethernet Shield. Для этого создадим на Arduino веб-сервер, используя библиотеку TinyWebServer.

Прежде всего подключим к цифровым выводам Arduino три реле. Контакты микроконтроллера не могут обеспечить мощность, необходимую для нормальной работы катушек реле. Поэтому нужно усилить ток — добавить n-p-n-транзистор, включенный по схеме с ОЭ, согласно схеме, приведенной на рис. 5.3. В этом случае выдача высокого уровня на контакт Arduino приведет к включению реле, подача низкого уровня — к его выключению.

Для того чтобы наш проект выглядел современно, мы воспользуемся технологией AJAX, которая позволяет отправлять данные на сервер и получать данные с сервера без перезагрузки страницы.

Технология AJAX

В нормальных веб-приложениях пользователи заполняют поля форм и нажимают кнопку **Submit** (Подтвердить). После этого форма передается на сервер полностью, сервер обрабатывает сценарий (обычно PHP или Java, возможно, CGI-процесс или что-то в этом роде), а потом передает назад всю новую страницу. Эта страница может быть HTML-страницей с новой формой с некоторыми заполненными данными, либо страницей подтверждения, либо страницей с какими-то выбранными вариантами, зависящими от введенных в оригинальную форму данных. Естественно, пока сценарий или программа на сервере не обработается и не возвратится новая форма, пользова-

тели должны ждать. Их экраны очистятся и будут перерисовываться по мере поступления новых данных от сервера. Вот где проявляется низкая интерактивность — пользователи не получают немедленной обратной реакции и определенно чувствуют себя не так, как при работе с настольными приложениями.

Технология AJAX по существу помещает технологию JavaScript и объект XMLHttpRequest между вашей веб-формой и сервером. Когда пользователи заполняют формы, данные передаются в некий JavaScript-код, а не прямо на сервер. Вместо этого JavaScript-код собирает данные формы и сам передает запрос на сервер. Пока это происходит, форма на экране пользователя не мелькает, не мигает, не исчезает и не блокируется. Другими словами, код JavaScript передает запрос в фоновом режиме — пользователь даже не замечает, что происходит запрос на сервер. Более того, запрос передается асинхронно, а это означает, что ваш JavaScript-код (и пользователь) не ожидает ответа сервера. То есть, пользователи могут продолжать вводить данные, прокручивать страницу и работать с приложением.

Когда же сервер передает данные обратно в ваш JavaScript-код (все еще находящийся в вашей веб-форме), тот решает, что делать с данными. Он может обновить поля формы «на лету», придавая свойство немедленности вашему приложению, — пользователи получают новые данные без подтверждения или обновления их форм. JavaScript-код может даже получить данные, выполнить какие-либо вычисления и передать еще один запрос, и все это без вмешательства пользователя! В этом заключается мощь XMLHttpRequest — он может общаться с сервером по своему желанию, а пользователь даже не догадывается о том, что происходит на самом деле. В результате мы получаем в веб-форме динамичность, чувствительность и высокую интерактивность настольного приложения вместе со всеми возможностями Интернета.

5.5.3. Веб-страница для управления реле

Создадим HTML-страницу для удаленного изменения статуса трех реле с использованием технологии AJAX. Веб-страница представляет собой форму, на которой расположены три элемента `radio`, — каждый такой элемент показывает статус одного из реле: **ON** — включено, **OFF** — выключено (рис. 5.13).

Добавим в форму еще один элемент `input`, в который станем передавать ответ сервера на ajax-запрос веб-страницы на изменения статуса реле. HTML-код формы представлен в листинге 5.6.

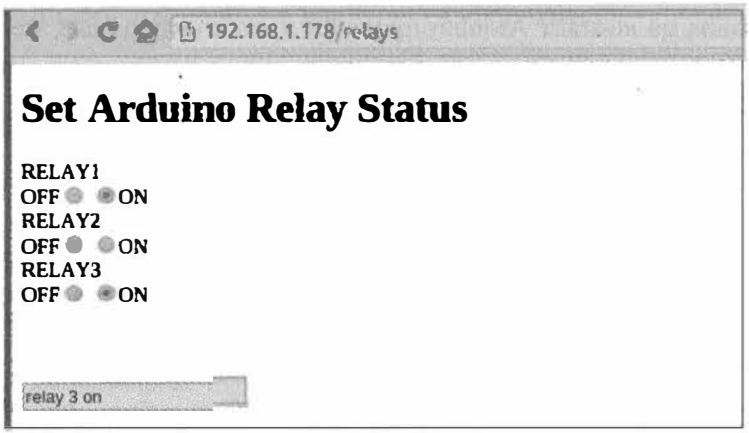


Рис. 5.13. Веб-страница для удаленного включения/выключения реле, подключенных к Arduino

Листинг 5.6

```
<form name=form1 action='return false;'>
   RELAY1<br>
   OFF<input type=radio name=relay1 value=0 checked
onclick="SetArduinoOutput(1,0);">
   <input type=radio name=relay1 value=1 onclick="SetArduinoOutput(1,1);">ON
   <br>RELAY2<br>
   OFF<input type=radio name=relay2 value=0 checked
onclick="SetArduinoOutput(2,0);">
   <input type=radio name=relay2 value=1 onclick="SetArduinoOutput(2,1);">ON
   <br>RELAY3<br>
   OFF<input type=radio name=relay3 value=0 checked
onclick="SetArduinoOutput(3,0);">
   <input type=radio name=relay3 value=1 onclick="SetArduinoOutput(3,1);">ON
   <br><br><br><br>
   <input type=text name="res1" id="res1" value="result"/>
</form>
```

При изменении значение элемента `radio` (с `on` на `off` или наоборот) вызывается js-функция `SetArduinoOutput()`, которая создает объект XMLHttpRequest для отправки данных на сервер и получения ответа с сервера с использованием AJAX. В листинге 5.7 представлен код js-функции `SetArduinoOutput()`.

Листинг 5.7

```
<script>
   function SetArduinoOutput(pin,value)
     {
      nocache = "&nocache=" + Math.random() * 1000000;
      var request = new XMLHttpRequest();
      request.onreadystatechange = function()
         {
         if (this.readyState == 4)
            {
             if (this.status == 200)
               {
                if (this.responseXML != null)
                  {
                  // разбор XML ответа сервера
                  document.getElementById("res1").value =
                  this.responseXML.getElementsByTagName('message')[0].
                  childNodes[0].nodeValue;
                  }
               }
```

```
        request.open("GET", "setrelay"+"&pin="+pin.toString()+"&value="
              +value.toString()+nocache, true);
        request.send(null);
    }
</script>
```

Код запрашиваемой страницы должен находиться в файле RELAYS, расположенном на SD-карте. Эта страница подгружается при URL-запросе `/relays` к TinyWebServer на Arduino. Вот список имен заголовков HTTP:

```
TinyWebServer::PathHandler handlers[] = {
  {"/", TinyWebServer::GET, &index_handler },
  {"/setrelay" "*", TinyWebServer::GET, &set_relays },
  {"/" "*", TinyWebServer::GET, &file_handler },
  {NULL},
};
```

При обращении по URL `http://<ipArduino>/relays` вызывается обработчик `file_handler()`, который выведет на страницу содержимое файла RELAYS с SD-карты. Код обработчика представлен в листинге 5.8.

Листинг 5.8

```
boolean file_handler(TinyWebServer& web_server)
   {
   if(!has_filesystem)
      {
      web_server.send_error_code(500);
      web_server << F("Internal Server Error");
      return true;
      }
  char* filename = TinyWebServer::get_file_from_path(web_server.get_path());
  if(!filename)
     {
     web_server.send_error_code(400);
     web_server << F("Bad Requestl");
     return true;
     }
  send_file_name(web_server, filename);
  free(filename);
  return true;
 }

void send_file_name(TinyWebServer& web_server, const char* filename)
  {
  TinyWebServer::MimeType mime_type
    = TinyWebServer::get_mime_type_from_filename(filename);
```

```
  if (file.open(&root, filename, O_READ))
     {
     web_server.send_error_code(200);
     web_server.send_content_type(mime_type);
     web_server.end_headers();

     Serial << F("Read file "); Serial.println(filename);
     web_server.send_file(file);
     file.close();
     }
  else
     {
     web_server.send_error_code(404);
     web_server.send_content_type("text/plain");
     web_server.end_headers();

     Serial << F("Could not find file: "); Serial.println(filename);
     web_server << F("404 - ERROR ") << filename << "\n";
     }
  }
```

Данные на сервер страница отправляет методом GET по URL `http://<ipArduino>/setrelay`. При поступлении на TinyWebServer этого запроса вызывается обработчик `set_relays()`, который должен обработать полученные методом GET данные, выделить номер реле и значение для него, включить (или выключить) реле и отправить серверу в формате XML ответ следующего содержания:

```
<xml version = "1.0" ?>"
<inputs>
    <message>relay X on(off)
    </message>
</inputs>
```

Содержимое обработчика `set_relays()` представлено в листинге 5.9.

Листинг 5.9

```
// включение/выключение реле по запросу
boolean set_relays(TinyWebServer& web_server)
  {
  char* filename = TinyWebServer::get_file_from_path(web_server.get_path());
  String str1(filename);
  Serial.println(str1);
  // установка реле
  digitalWrite(pinrelays[convert1(str1,13,14)-1],convert1(str1,21,22));

  if(!filename)
    {
    web_server.send_error_code(400);
```

```
    web_server << F("Bad Request1");
    return true;
    }
  web_server.send_error_code(200);
  web_server.send_content_type("text/xml");
  web_server.end_headers();
  web_server << F("<?xml version = \"1.0\" ?>");
  web_server << F("<inputs>");
  web_server << F("<message>relay ")<<str1.substring(13,14);
  if(convert1(str1,21,22)>0)
      web_server << F(" on");
  else
      web_server << F(" off");
  web_server << F("</message>");
  web_server << F("</inputs>");
  return true;
}
// выделение нужных значений из GET-данных
int convert1(String strstr,int start1,int end1)
  {
  char *strint=new char[2];
  String str3;
  str3=strstr.substring(start1,end1);str3.toCharArray(strint,2);
  return atoi(strint);
  }
```

При этом индексная страница на сервере (рис. 5.14) представляет собой меню для выбора страницы включения/выключения реле **Set relays** (`/relays`) или управления сервоприводами **Set servos** (`/servos`) — эту опцию мы рассмотрим в следующем разделе. При поступлении запроса индексной страницы вызывается обработчик `index_handler()` (листинг 5.10).

Листинг 5.10

```
// главная страница
boolean index_handler(TinyWebServer& web_server)
  {
  web_server.send_error_code(200);
  web_server.send_content_type("text/html");
  web_server.end_headers();
  web_server << F("<html><body><h1>Menu</h1>");
  web_server << F("<a href='/relays'>Set relays</a><br>");
  web_server << F("<br>");
  web_server << F("<a href='/servos'>Set servos</a><br>");
  web_server << F("<br>");
  return true;
  }
```

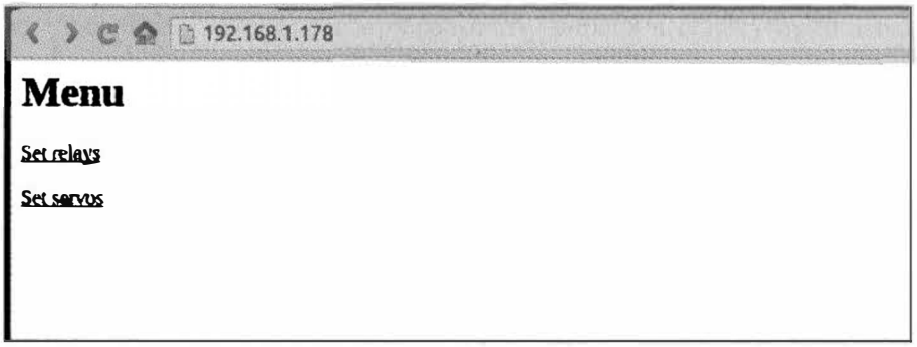


Рис. 5.14. Индексная страница нашего TinyWebServer

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингам, описывающим управление реле из веб-страницы, созданной с использованием технологии AJAX, можно найти в файле arduino_scetches_05_05_1214.ino сопровождающего книгу электронного архива.

5.5.4. Веб-страница для управления сервоприводом

Добавим на наш TinyWebServer страницу управления сервоприводом (рис. 5.15). С помощью HTML-элемента `range` (ползунок) мы станем задавать угол поворота сервопривода, а в элемент `input` — передавать ответ сервера на ajax-запрос веб-страницы по изменению угла поворота сервопривода. HTML-код такой формы представлен в листинге 5.11.

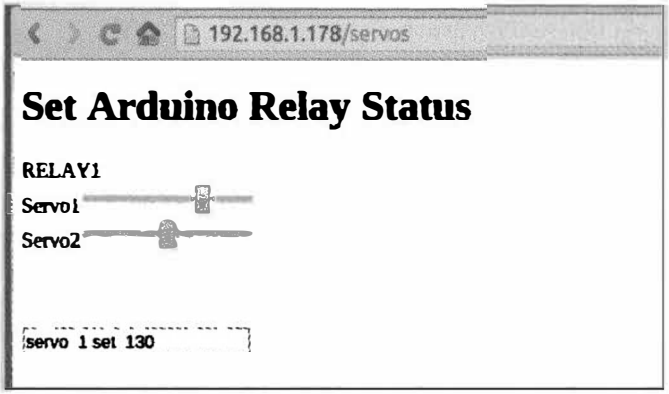


Рис. 5.15. Веб-страница для удаленного управления сервоприводами, подключенными к Arduino

Листинг 5.11

```
<form name=form1 action='return false;'>
        Servo1<input type=range name=servo1 min=0 max=180 step=1 value=90
                    onchange="SetArduinoOutput(1,this.value);">
        <br>
```

```
        Servo2<input type=range name=servo2 min=0 max=180 step=1 value=90
                    onchange="SetArduinoOutput(2,this.value);">
        <br><br><br><br>
        <input type=text name=res1 id=res1 value="result">
  </form>
```

При щелчке мышью по линии движения ползунка (элемент `range`) или по его отпусканию по завершении передвижения вызывается js-функция `SetArduinoOutput()`, которая создает объект XMLHttpRequest для отправки данных на сервер и получения ответа с сервера с использованием AJAX. В листинге 5.12 представлен код js-функции `SetArduinoOutput()`.

Листинг 5.12

```
<script>
     function SetArduinoOutput(pin,value)
        {
         var value1="";
         if(value<10)
              value1=value1+"00"+value;
         else if(value<100)
              value1=value1+"0"+value;
         else
              value1=value1+value;
          nocache = "&nocache=" + Math.random() * 1000000;
          var request = new XMLHttpRequest();
          request.onreadystatechange = function()
              {
                if (this.readyState == 4)
                   {
                    if (this.status == 200)
                        {
                        if (this.responseXML != null)
                            {
                            // разбор XML ответа сервера
                            document.getElementById("res1").value =
                            this.responseXML.getElementsByTagName('message')
                            [0].childNodes[0].nodeValue;
                            }
                        }
                   }
              }
            request.open("GET",
"setservo"+"&pin="+pin+"&value="+value1+nocache, true);
            request.send(null);
        }
</script>
```

Код запрашиваемой страницы находится в файле SERVOS, расположенном на SD-карте. Эта страница подгружается при URL запросе `/servos` к TinyWebServer на

Arduino. Добавляем этот запрос и обработчик запроса к списку имен заголовков HTTP:

```
TinyWebServer::PathHandler handlers[] = {
  {"/", TinyWebServer::GET, &index_handler },
  {"/setservo" "*", TinyWebServer::GET, &set_servos },
  {"/setrelay" "*", TinyWebServer::GET, &set_relays },
  {"/" "*", TinyWebServer::GET, &file_handler },
  {NULL},
};
```

При обращении по URL `http://<ipArduino>/servos` вызывается обработчик `file_handler()`, который выведет на страницу содержимое файла SERVOS с SD-карты. С кодом этого обработчика мы уже знакомы (см. листинг 5.8).

Данные на сервер страница отправляет методом GET по URL **http://<ipArduino>/setservo**. При поступлении на TinyWebServer этого запроса вызывается обработчик `set_servos()`, который должен обработать полученные методом GET данные, выделить номер сервопривода и угол поворота, выдать команду поворота для сервопривода и отправить серверу в формате XML ответ следующего содержания:

```
<xml version = "1.0" ?>"
<inputs>
    <message>servo X set XXX
    </message>
</inputs>
```

Далее для управления сервоприводом используем библиотеку Arduino Servo:

1. Подключаем библиотеку Servo:

   ```
   #include <Servo.h>
   ```

2. Создаем экземпляры объектов:

   ```
   Servo myservo1;
   Servo myservo2;
   ```

3. Подключаем переменные `myservo1` и `myservo2` к выводам Arduino:

   ```
   myservo1.attach(2);
   myservo2.attach(3);
   ```

4. Поворачиваем сервоприводы на угол, равный значению ползунка на веб-странице:

   ```
   myservo1.write(convert1(str1,21,24,4));
   myservo2.write(convert1(str1,21,24,4));
   ```

Содержимое обработчика `set_servos()` представлено в листинге 5.13.

Листинг 5.13

```
// управление поворотом сервопривода по запросу
boolean set_servos(TinyWebServer& web_server)
```

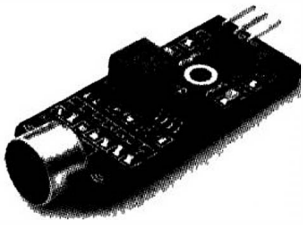
```
  {
  char* filename = TinyWebServer::get_file_from_path(web_server.get_path());
  String str1(filename);
  Serial.println(str1);
  if(!filename)
     {
     web_server.send_error_code(400);
     web_server << F("Bad Request1");
     return true;
     }
  if(convert1(str1,13,14,2)==1)
       myservo1.write(convert1(str1,21,24,4));    // поворот серво 1
  else if(convert1(str1,13,14,2)==2)
       myservo2.write(convert1(str1,21,24,4));    //  поворот серво 2
  else
       ;
  // ответ серверу
  web_server.send_error_code(200);
  web_server.send_content_type("text/xml");
  web_server.end_headers();
  web_server << F("<?xml version = \"1.0\" ?>");
  web_server << F("<inputs>");
  web_server << F("<message>servo  ")<<str1.substring(13,14);
  web_server << F(" set  ")<<convert1(str1,21,24,4);
  web_server << F("</message>");
  web_server << F("</inputs>");
  return true;
}
// выделение нужных значений из GET-данных
int convert1(String strstr,int start1,int end1, int kol)
  {
  char *strint=new char[kol];
  String str3;
  str3=strstr.substring(start1,end1);str3.toCharArray(strint,kol);
  return atoi(strint);
  }
```

Электронный архив

Скетч, соответствующий листингам, описывающим управление сервоприводами из веб-страницы, созданной с использованием технологии AJAX, можно найти в файле arduino_scetches_05_05_1517.ino сопровождающего книгу электронного архива.

В следующих главах мы еще вернемся к использованию библиотеки TinyWebServer для удаленного доступа к показаниям датчиков Arduino. Вернемся и к удаленному управлению исполнительными устройствами, подключенными к Arduino, но уже с использованием платы GPS/GPRS Shield.

ГЛАВА 6



Arduino и устройства I²C

В *главе 4* мы рассмотрели работу плат Arduino с аналоговыми датчиками, но может ли Arduino общаться с более сложными устройствами? Разумеется! — Arduino способен реализовать имеющиеся возможности, взаимодействуя через различные интерфейсы со множеством внешних компонентов. Чтобы упростить обмен данными между микроконтроллером и огромным количеством всевозможных модулей, создан ряд интегральных схем (ИС), реализующих стандартизированные цифровые протоколы связи. В этой главе мы рассмотрим работу Arduino с устройствами I²C.

6.1. Обзор протокола I²C

Разработанная фирмой Philips шина I²C (Inter-Integrated Circuit) — это двунаправленная асинхронная шина с последовательной передачей данных и возможностью адресации до 128 устройств. Физически шина I²C содержит две сигнальные линии, одна из которых (SCL) предназначена для передачи тактового сигнала, а вторая (SDA) — для обмена данными. Для управления линиями применяются выходные каскады с открытым коллектором, поэтому линии шины должны быть подтянуты к источнику питания +5 В через резисторы сопротивлением 1–10 кОм (в зависимости от физической длины линий и скорости передачи данных). Длина соединительных линий в стандартном режиме может достигать двух метров, скорость передачи — до 100 Кбит/с. Суммарная емкость линий не должна превышать 400 пФ, входная емкость на каждую ИС должна находиться в пределах 5–10 пФ.

Отличительные особенности протокола:

- двунаправленный обмен по обеим линиям;
- высокая скорость обмена — до 100 Кбит/с и выше;
- возможность адресации до 128 устройств;
- простота программной реализации Master-абонента;
- временна́я независимость процесса передачи.

Все абоненты шины делятся на два класса: Master и Slave. Устройство Master генерирует тактовый сигнал (SCL) и, как следствие, является ведущим. Оно может

самостоятельно выходить на шину и адресовать любое Slave-устройство с целью передачи или приема информации. Все Slave-устройства «слушают» шину на предмет обнаружения собственного адреса и, распознав его, выполняют предписываемую операцию. Кроме того, возможен так называемый Multi Master — режим, когда на шине установлено несколько Master-абонентов, которые либо совместно разделяют общие Slave-устройства, либо попеременно являются то Master-устройствами — когда сами инициируют обмен информацией, то Slave-устройствами — когда находятся в режиме ожидания обращения от другого Master-устройства. Режим Multi Master требует арбитража и распознавания конфликтов. Естественно, он сложнее в реализации (имеется в виду программная реализация) и, как следствие, реже используется в реальных изделиях.

В начальный момент времени — в режиме ожидания — обе линии, SCL и SDA, находятся в состоянии логической единицы (транзистор выходного каскада с ОК закрыт). В режиме передачи (рис. 6.1) бит данных SDA стробируется положительным импульсом SCL. Смена информации на линии SDA производится при нулевом состоянии линии SCL. Slave-устройство может «придерживать» линию SCL в нулевом состоянии — например, на время обработки очередного принятого байта, при этом Master-устройство обязано дождаться освобождения линии SCL, прежде чем продолжить передачу информации.

Для синхронизации пакетов шины I^2C различают два условия: Start и Stop, ограничивающих начало и конец информационного пакета (рис. 6.2). Для кодирования

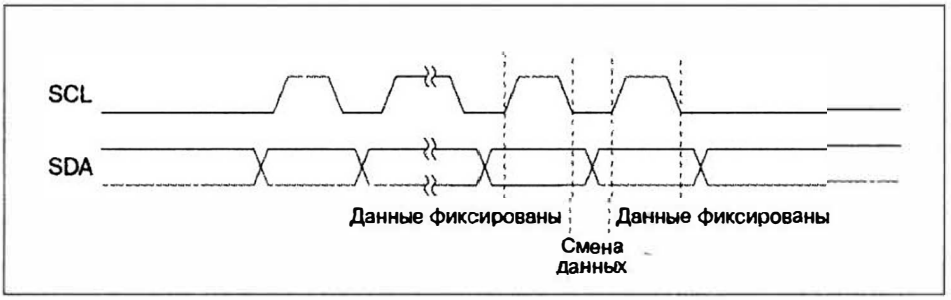


Рис. 6.1. Диаграмма процесса передачи данных по шине I^2C

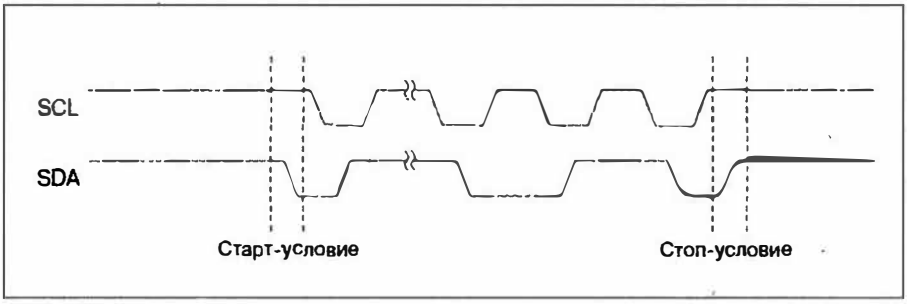


Рис. 6.2. Диаграмма START/STOP условия шины I^2C

этих условий используется изменение состояния линии SCL, что недопустимо при передаче данных. Start-условие образуется при отрицательном перепаде линии SDA, когда линия SCL находится в единичном состоянии, и, наоборот, Stop-условие образуется при положительном перепаде линии SDA при единичном состоянии линии SCL.

Передача данных начинается по первому положительному импульсу на линии SCL (рис. 6.3), которым стробируется старший бит первого информационного байта. Каждый информационный байт (8 битов) содержит 9 тактовых периодов линии SCL. В девятом такте устройство-получатель выдает подтверждение (ACK) — отрицательный импульс, свидетельствующий о «взаимопонимании» передатчика и получателя. Сразу отметим, что любой абонент шины — как Master, так и Slave — может в разные моменты времени быть как передатчиком, так и получателем, и в соответствии с режимом обязан либо принимать, либо выдавать сигнал ACK, отсутствие которого интерпретируется как ошибка.

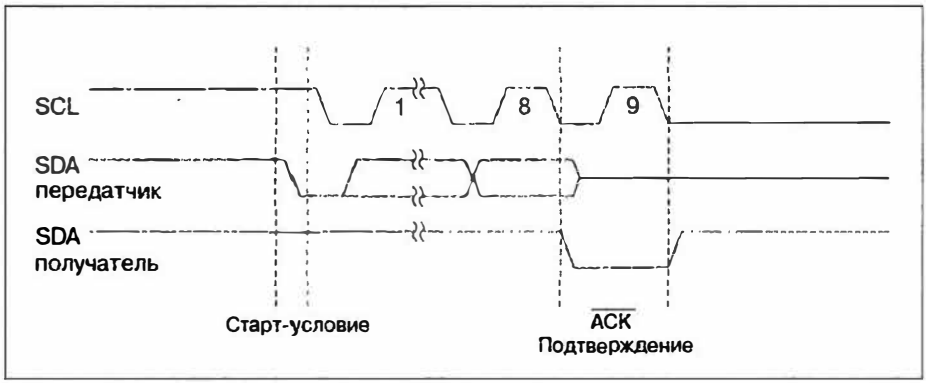


Рис. 6.3. Диаграмма подтверждения приема байта на шине I²C

Временна́я диаграмма сигналов SCL и SDA шины I²C показана на рис. 6.4 (здесь S обозначает Start-условие, а P — Stop-условие). Значения временны́х характеристик шины приведены в табл. 6.1.

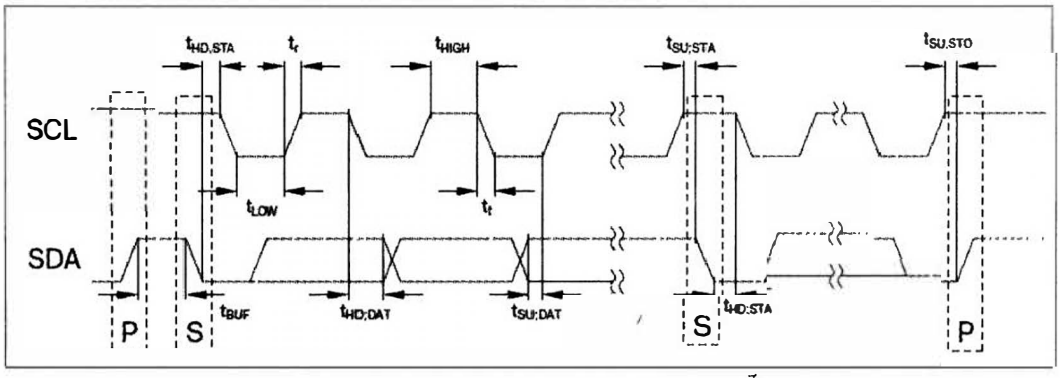


Рис. 6.4. Временная диаграмма работы шины I²C

Таблица 6.1. Значения временных характеристик шины I^2C

Параметр	Обозначение	min	max	Единица
Частота сигнала SCL	fSCL	0	100	кГц
Свободная шина	t_{BUF}	4.7	–	мкс
Фиксация START-условия	$t_{HD,ST}$	4.0	–	мкс
Длительность LOW полупериода SCL	t_{LOW}	4.7	–	мкс
Длительность HIGH полупериода SCL	t_{HIGH}	4.0	–	мкс
Готовность повторного START-условия	$t_{SU.STA}$	4.7	–	мкс
Удержание данных	$t_{HD.DAT}$	0	–	мкс
Готовность данных	$t_{SU.DAT}$	250	–	нс
Фронт сигналов SCL и SDA	t_r	–	1000	
Спад сигналов SCL и SDA	t_f	–	300	
Готовность STOP-условия	$t_{SU;STO}$	4.0	–	мкс

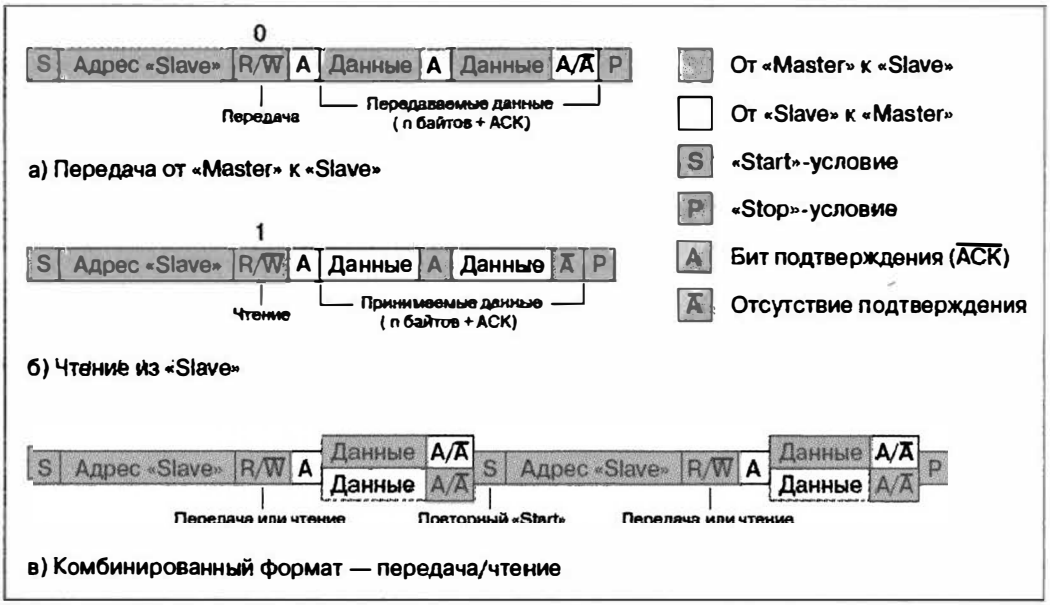


Рис. 6.5. Формат операций чтения/записи

Чтобы начать операцию обмена, устройство Master выдает на шину Start-условие, за которым следует байт с адресом Slave-устройства (рис. 6.5), состоящий из семибитового адреса устройства (занимает биты 1–7) и однобитового флага операции R/W (бит 0), определяющего направление обмена, причем 0 означает передачу от Master к Slave (рис. 6.5, *а*), а 1 — чтение из Slave (рис. 6.5, *б*). Все биты передаются по шине I^2C в порядке старший-младший, т. е. первым передается 7-й бит, последним 0-й. За адресом могут следовать один или более информационных байтов

(в направлении, определенном флагом R/W), биты которых стробируются сигналом SCL из Master-устройства.

При совершении операции чтения Master-абонент должен сопровождать прочитанный байт сигналом ACK, если необходимо прочитать следующий байт, и не выдавать сигнала ACK, если собирается закончить чтение пакета.

Допускается и многократное возобновление Slave-адреса в одном цикле передачи, т. е. передача повторного Start-условия без предварительного Stop-условия. Такой принцип широко применяется в управлении I²C абонентами, когда выдача нового Start-условия служит для синхронизации начала нового пакета данных, сопровождаемого, например, новым управляющим словом, уточняющим адресацию пакета.

Логическая реализация протоколов на шине I²C не нормируется документами фирмы Philips, содержащими формальные описания шины, и может быть произвольной для каждой конкретной ИС.

Преимущества интерфейса I²C:

- необходим всего один микроконтроллер для управления набором устройств;
- используется всего два проводника для подключения многих устройств;
- возможна одновременная работа нескольких ведущих (master) устройств, подключенных к одной шине I²C;
- стандарт предусматривает «горячее» подключение и отключение устройств в процессе работы системы;
- фильтр, встроенный в микросхемы, реализующие интерфейс, подавляет всплески, обеспечивая целостность данных.

Недостатки интерфейса I²C:

- ограничение на емкость линии — 400 пФ;
- несмотря на простоту протокола, программирование контроллера I²C затруднено из-за обилия на шине возможных нештатных ситуаций. По этой причине большинство систем используют I²C с единственным ведущим (master) устройством и распространенные драйверы поддерживают только монопольный режим обмена по I²C;
- трудность локализации неисправности, если одно из подключенных устройств ошибочно устанавливает на шине состояние низкого уровня.

6.2. Arduino и библиотека Wire

Библиотека Wire позволяет Arduino взаимодействовать с различными устройствами по интерфейсу I²C/TWI. На платах Arduino версии R3 (с «распиновкой» 1.0) линии SDA (данные) и SCL (тактовые импульсы), связанные с этим интерфейсом, расположены на разъеме возле контакта AREF. В Arduino Due реализовано два интерфейса I²C/TWI: линии одного из них (SDA1 и SCL1) расположены возле вывода AREF, линии второго — на выводах 20 и 21. Расположение выводов I²C/TWI для различных плат Arduino показано в табл. 6.2.

Таблица 6.2. Расположение выводов I²C/TWI для различных плат Arduino

Плата Arduino	Выводы I²C/TWI
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

Начиная с версии языка Arduino 1.0, библиотека Wire наследует функции класса `Stream`, что позволяет ей быть совместимой с другими библиотеками, осуществляющими запись и чтение данных. Поэтому методы `send()` и `receive()` были в ней заменены методами `read()` и `write()`.

Библиотека Wire использует следующие методы:

- `void begin();void begin(uint8_t address);void begin(int address);`

  Описание: инициализация библиотеки Wire и подключение к шине I²C в качестве Master или Slave. Вызывается только один раз.

  Параметр: `address` — 7-битный адрес устройства (если оно работает в режиме Slave). Если параметр не указан, то контроллер подключается к шине в роли Master.

  Возвращаемое значение: нет.

- `uint8_t requestFrom(uint8_t address, uint8_t quantity);`

  Описание: используется мастером для запроса байта от ведомого устройства (Slave). Байты могут быть получены с помощью методов `available()` и `read()`.

  Параметры:

  - `address` — 7-битный адрес устройства для запроса байтов данных;
  - `quantity` — количество запрошенных байтов.

  Возвращаемое значение: число считанных байтов.

- `void beginTransmission(uint8_t address);`

  Описание: начало передачи I²C для ведомого устройства (Slave) с заданным адресом, с последующими вызовом метода `write()` для добавления последовательности байтов в очередь предназначенных для передачи и выполнением самой передачи данных методом `endTransmission()`.

  Параметр: `address` — 7-битный адрес устройства для передачи.

  Возвращаемое значение: нет.

- `uint8_t endTransmission(void);`

  Описание: завершает передачу данных для ведомого устройства (Slave), которое было начато методом `beginTransmission()`, и фактически осуществляет передачу байтов, которые были поставлены в очередь методом `write()`.

Параметры: нет.

Возвращаемое значение: байт, который указывает статус передачи:

- 0 — успех;
- 1 — данных слишком много, и они не помещаются в буфер передачи (размер буфера задается определением `#define BUFFER_LENGTH 32`);
- 2 — получили NACK на передачу адреса;
- 3 — получили NACK на передачу данных;
- 4 — другая ошибка.

❒ `size_t write(uint8_t data);size_t write(const uint8_t *data, size_t quantity);`

Вызов:

- `Wire.write(value);`
- `Wire.write(string);`
- `Wire.write(data, length);`

Описание: записывает данные от ведомого устройства (Slave) в ответ на запрос мастера (Master), или записывает очередь байтов для передачи от мастера к ведомому устройству (в промежутках между вызовами методов `beginTransmission()` и `endTransmission()`).

Параметры:

- `value` — значение для отправки как единичный байт;
- `string` — строка для отправки как последовательность байтов;
- `data` — массив байтов для отправки;
- `length` — число байтов для передачи.

Возвращаемое значение: число записанных байтов.

❒ `int available(void);`

Описание: возвращает количество байтов, доступных для получения. Этот метод должно быть вызван на мастере (Master) после вызова `requestFrom()` или ведомым устройством (Slave) внутри обработчика `onReceive()`.

Параметры: нет.

Возвращаемое значение: число байтов, доступных для чтения.

❒ `int read(void);`

Описание: считывает байт, который был передан от ведомого устройства (Slave) к мастеру (Master) после вызова `requestFrom()` или был передан от Master к Slave.

Параметры: нет.

Возвращаемое значение: следующий полученный байт.

Пример использования этого метода приведен в листинге 6.1.

Листинг 6.1

```
#include <Wire.h>
void setup()
  {
   Wire.begin();            // подключение к шине I2C
   Serial.begin(9600);      // запуск последовательного порта
  }
void loop()
  {
   Wire.requestFrom(2, 6);    // запрос 6-го байта от
                              // устройства с адресом 2
   while(Wire.available())    // пока есть, что читать
     {
     char c = Wire.read();    // получаем байт (как символ)
     Serial.print(c);         // печатает в порт
     }
   delay(500);
  }
```

- `void onReceive( void (*function)(int) );`

  Описание: регистрирует функцию, которая вызывается, когда ведомое устройство (Slave) получает данные от мастера (Master).

  Параметр: `function` — функция, которая вызывается, когда Slave получает данные. Обработчик должен принимать один параметр `int` (число байтов, считанных от мастера) и ничего не возвращать.

  Возвращаемое значение: нет.

- `void onRequest( void (*function)(void) );`

  Описание: регистрирует функцию, которая вызывается, когда мастер запрашивает данные из этого ведомого устройства.

  Параметр: `function` — функция, которая будет вызываться.

  Возвращаемое значение: нет.

6.3. Arduino и датчик освещенности BH1750 на шине I²C

В *главе 4* для измерения уровня освещенности мы использовали фоторезистор. Но если необходимо знать величину освещенности в физических единицах (люксах), он не удобен, поскольку тогда нам пришлось бы делать для такого датчика измерительную схему и калибровать полученные с него данные. Для получения точных величин освещенности в люксах удобнее использовать датчик BH1750, который посредством цифрового интерфейса сможет выдавать нам готовые значения.

Датчик BH1750 обладает следующими характеристиками:

- цифровой интерфейс: I²C;
- высокое разрешение: до 0,5 Лк;
- малый потребляемый ток и наличие функции «спящего» режима;
- фильтрация световых шумов: 50/60 Гц;
- малая зависимость от источника света (лампа накаливания, светодиод и т. д.);
- малое влияние инфракрасного излучения;
- возможность выбрать два адреса микросхемы для I²C интерфейса (можно подключить одновременно два таких датчика к одной шине);
- не требует калибровки, что максимально удобно для применения в любых проектах;
- малые габариты;
- напряжение питания: 2,4–3,6 В;
- ток потребления: 120 мкА;
- ток потребления в спящем режиме: 0,01 мкА;
- измеряемая длина волны: 560 нм;
- точность в режиме высокого разрешения: 1 Лк;
- точность в режиме низкого разрешения: 4 Лк;
- период измерения в режиме высокого разрешения: 120 мс;
- период измерения в режиме низкого разрешения: 16 мс;
- 16-битный АЦП.

Для использования с платой Arduino датчик выпускается в виде модуля (рис. 6.6).

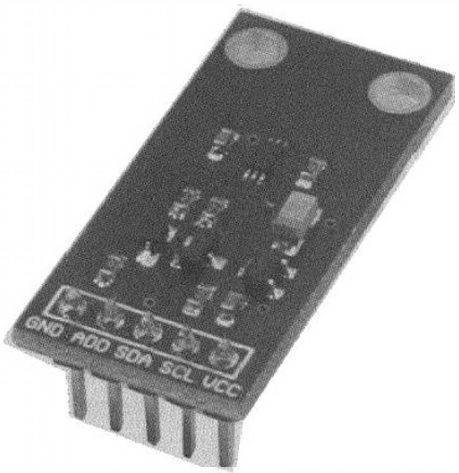


Рис. 6.6. Модуль датчика BH1750

Итак, подключим модуль датчика BH1750 к плате Arduino (рис. 6.7) и напишем скетч получения значений освещенности в люксах. Для работы с модулем мы воспользуемся библиотекой BH1750, которая является «оберткой» для библиотеки Wire. Содержимое скетча представлено в листинге 6.2.

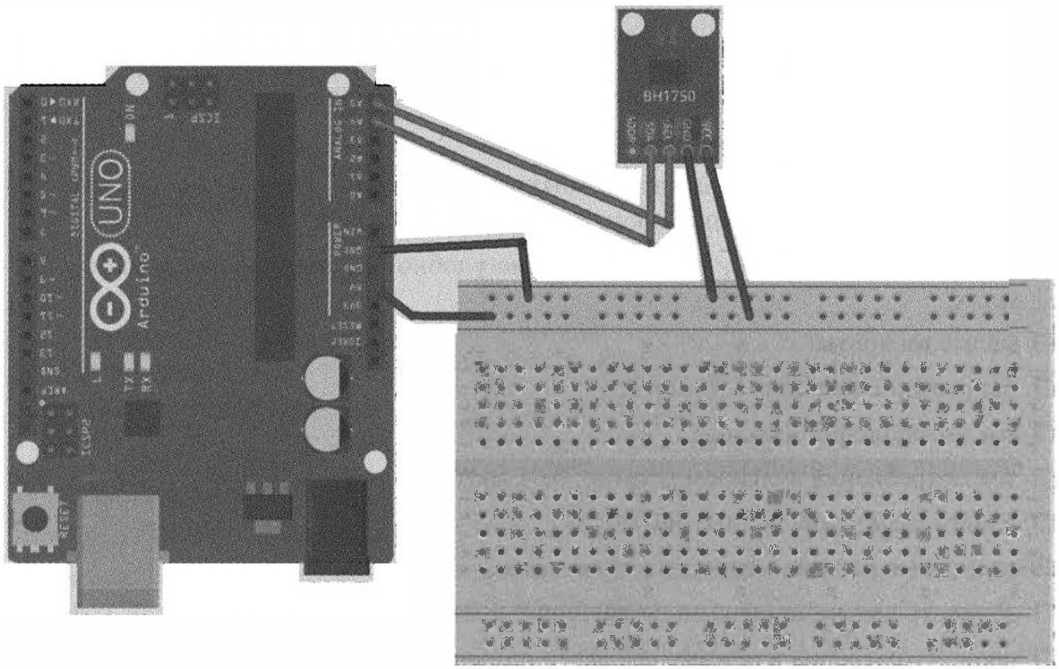


Рис. 6.7. Схема соединения модуля BH1750 с платой Arduino UNO

Листинг 6.2

```
// подключение библиотек
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;
void setup()
  {
  Serial.begin(9600);   // запуск последовательного порта
  lightMeter.begin();   // подключение к I2C
  Serial.println("Running...");
  }
void loop()
   {
  uint16_t lux = lightMeter.readLightLevel();  // получение значения
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(1000);
  }
```

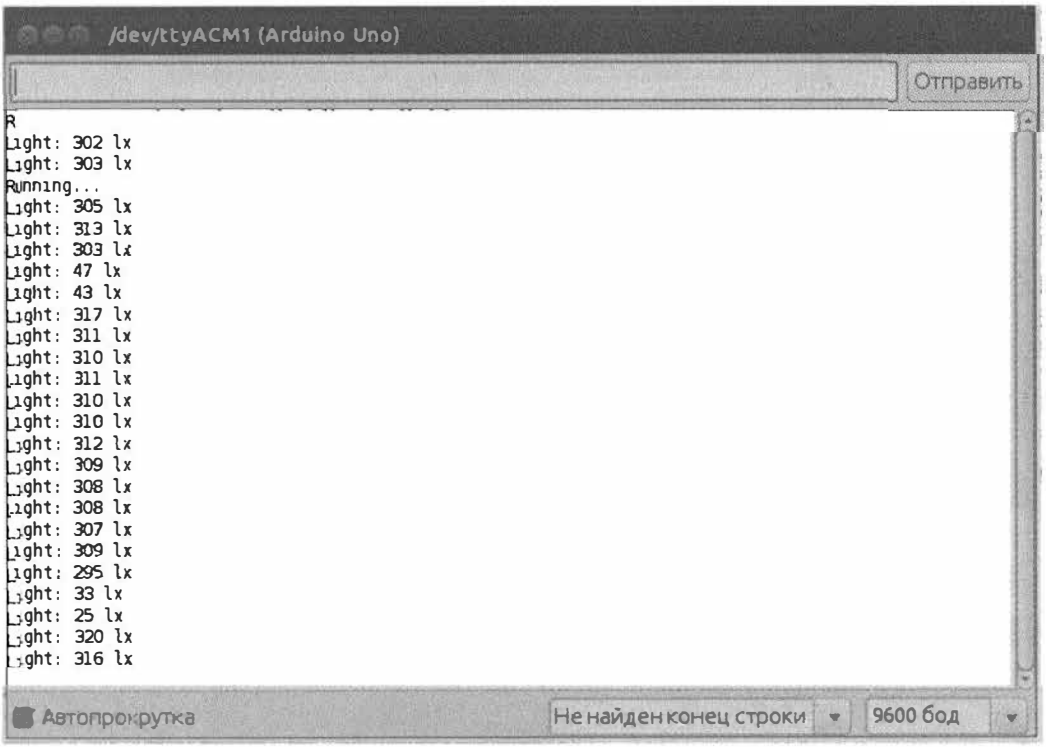


Рис. 6.8. Вывод показаний модуля BH1750 в монитор последовательного порта Arduino IDE

На рис. 6.8 показан вывод показаний модуля BH1750 в монитор последовательного порта Arduino IDE.

6.4. Arduino и сервис Xively

Для отображения информации с датчиков Arduino служит и облачный сервис Xively (**https://xively.com**), также реализующий возможности концепции Internet of Things.

Чтобы пользоваться этим сервисом, сначала необходимо в нем зарегистрироваться, — на главной странице сервиса (рис. 6.9) нажимаем кнопку **GET STARTED** и в открывшейся форме регистрации (рис. 6.10) заполняем все необходимые поля.

Зарегистрировавшись, мы можем войти в свой профиль и в разделе **DEVELOP** (рис. 6.11) добавить новое устройство, нажав на кнопку **+ Add Device**. Откроется форма (рис. 6.12), в поля которой мы вводим название (здесь: **Zheleznovodsk**), описание нового устройства (здесь: **Zheleznovodsk sanatoriy**), выбираем вид доступа к нему (приватный или публичный) и нажимаем кнопку **Add Device**, в результате чего попадаем в Xively Workbench — центр, объединяющий все датчики добавленного нами устройства. Здесь отображается информация об устройстве, последние запросы, текущие значения каналов, ключей API, триггеров и многое другое.

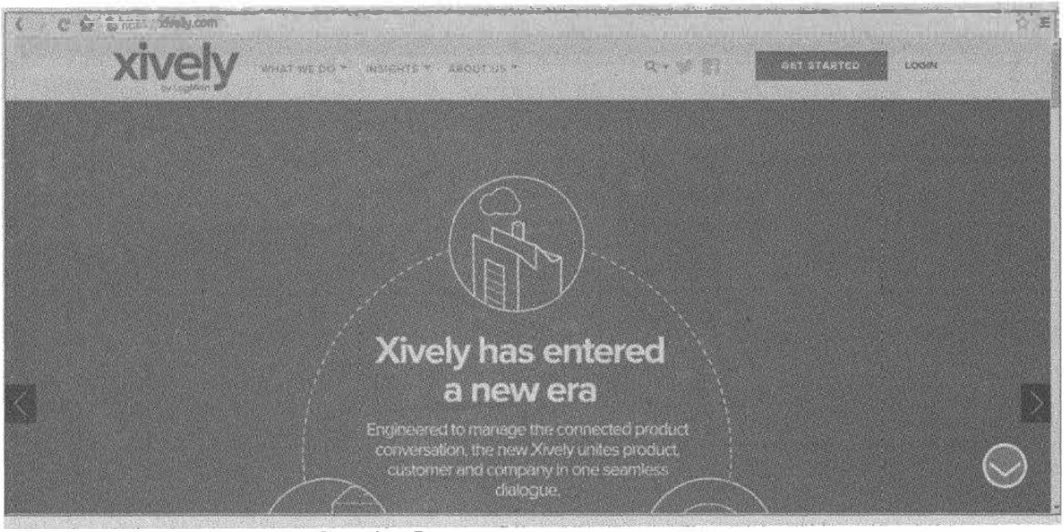


Рис. 6.9. Главная страница сервиса Xively

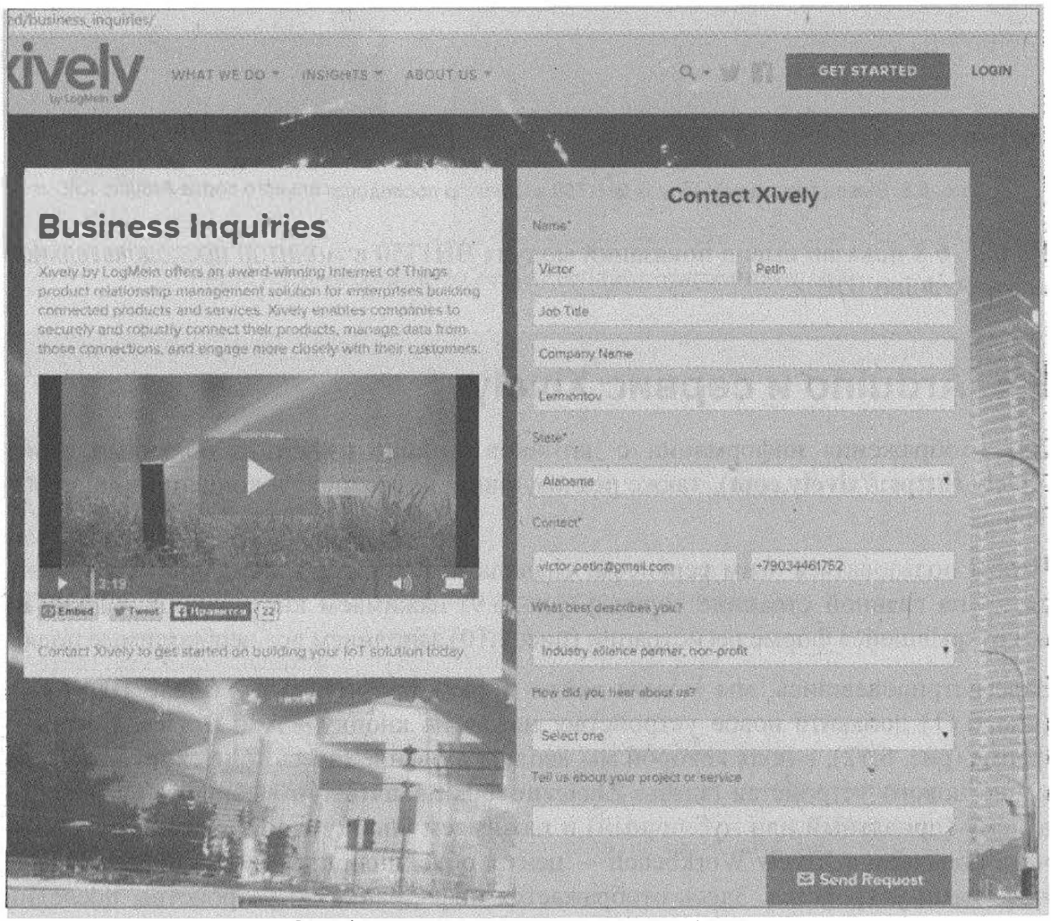


Рис. 6.10. Форма регистрации сервиса Xively

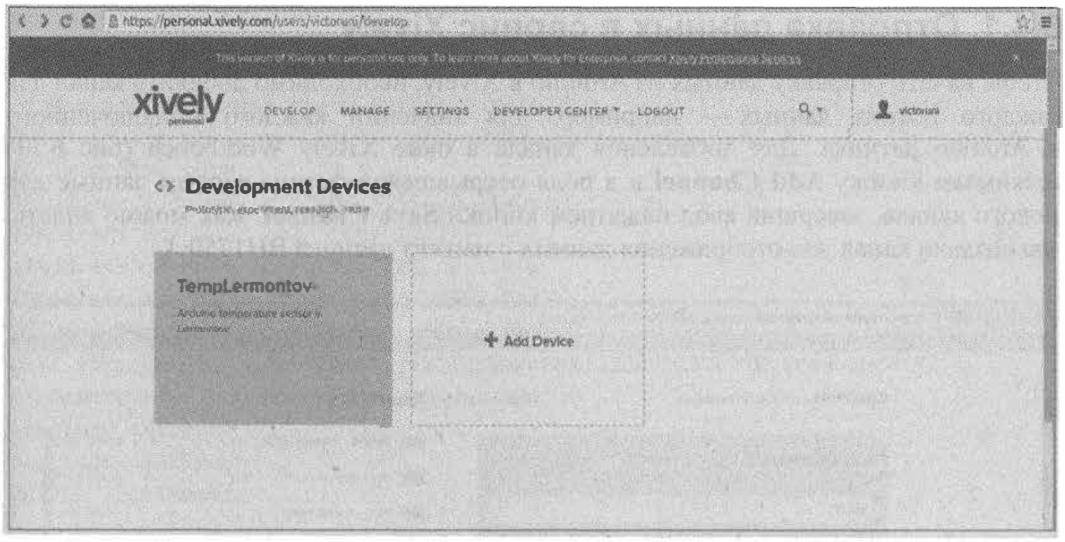


Рис. 6.11. Добавление нового устройства в Xively

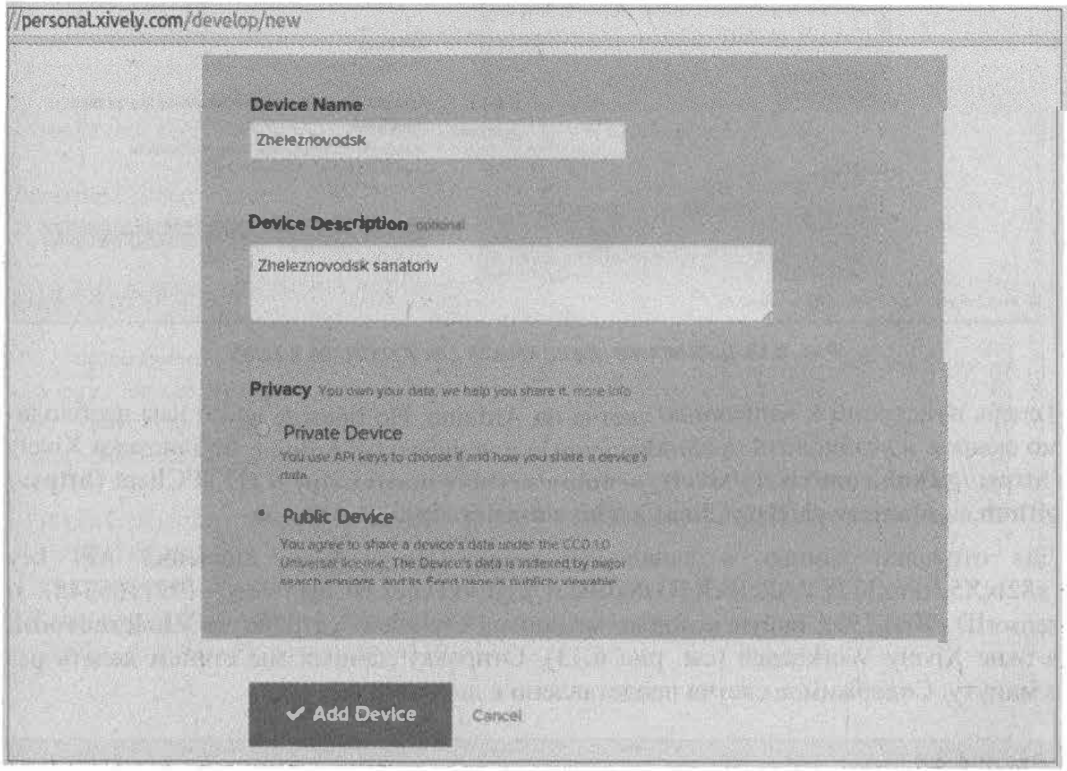


Рис. 6.12. Форма для ввода данных нового устройства в Xively

6.4.1. Отправка данных в сервис Xively

Чтобы начать отправку данных из Arduino в Xively, необходимо добавить канал для каждого потока данных — например, для значений каждого подключенного к Arduino датчика. Для добавления канала в окне Xively Workbench (рис. 6.13) нажимаем кнопку **Add Channel** и в поля открывшейся формы вводим данные для нового канала, завершив ввод нажатием кнопки **Save Channel**. Как можно видеть, мы создали канал для отображения данных с нашего датчика BH1750.

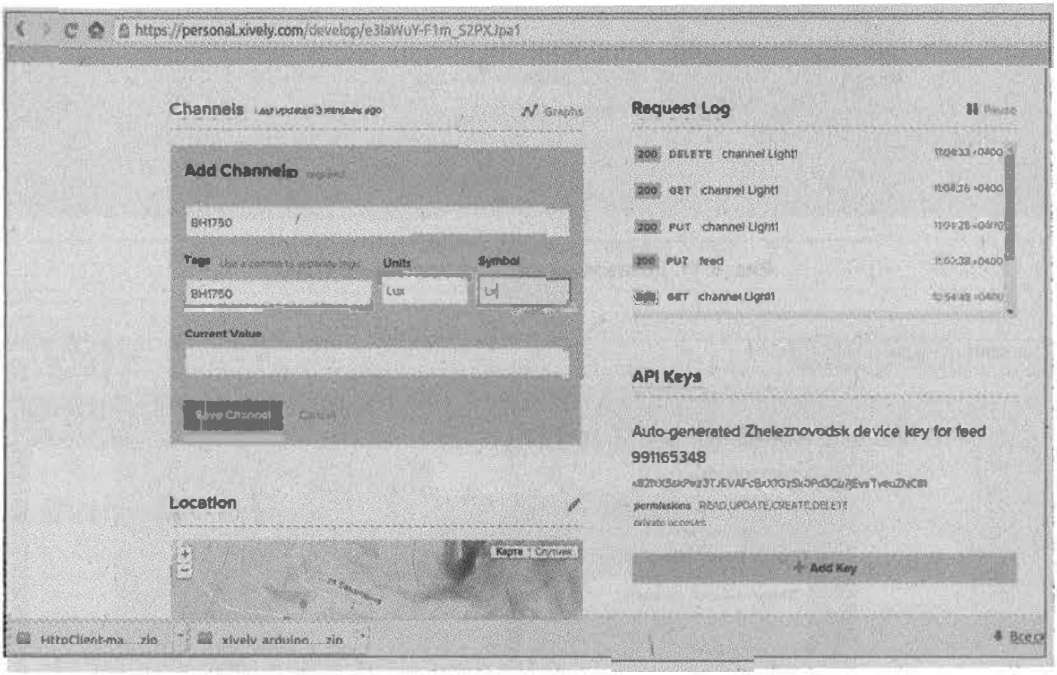


Рис. 6.13. Добавление нового канала для устройства в Xively

Теперь приступим к написанию скетча на Arduino. Но прежде всего нам необходимо скачать и установить предназначенные для работы с Arduino библиотеки Xively (**https://github.com/xively/xively_arduino/archive/master.zip**) и HTTPClient (**https://github.com/amcewen/HttpClient/archive/master.zip**).

Для отправки данных в канал нам также понадобятся значения API key (x82lxX5skPez3TJEVAFcBxX1GzSk0Pd3Cu7jEvsTveuZNC8t), feed (991165348) и sensorID (BH1750), найти которые можно на странице устройства **Zheleznovodsk** в окне Xively Workbench (см. рис. 6.13). Отправку данных мы станем делать раз в минуту. Содержимое скетча представлено в листинге 6.3.

Листинг 6.3

```
// Подключение библиотек
#include <SPI.h>
#include <Ethernet.h>
```

```
#include <HttpClient.h>
#include <Xively.h>

// Mac address нашей платы Ethernet Shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
// Настройки сети
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);
// Your Xively key to let you upload data
char xivelyKey[] = "x82lxX5skPez3TJEVAFcBxX1GzSk0Pd3Cu7jEvsTveuZNC8t";
// Подключение библиотек для датчика BH1750
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;

// Данные для канала
char sensorId[] = "BH1750";
XivelyDatastream datastreams[] = {
   XivelyDatastream(sensorId, strlen(sensorId), DATASTREAM_FLOAT),
};
// Направить поток данных в канал feed
XivelyFeed feed(991165348, datastreams, 1 /* количество каналов */);

EthernetClient client;
XivelyClient xivelyclient(client);

void setup()
  {
  // подключение последовательного порта
  Serial.begin(9600);
  // запуск датчика BH1750
  lightMeter.begin();

  Serial.println("Starting single datastream upload to Xively...");
  Serial.println();
  // Подключение к сети
  Ethernet.begin(arduino_mac,arduino_ip,dns_ip,gateway_ip,subnet_mask);
  Serial.println(Ethernet.localIP());
  }

void loop()
  {
  // получение данных с BH1750
  uint16_t lux = lightMeter.readLightLevel();
  datastreams[0].setFloat(lux);
```

```
Serial.print("BH1750 sensor value ");
Serial.println(datastreams[0].getFloat());

Serial.println("Uploading it to Xively");
// Получение ответа от Xively
int ret = xivelyclient.put(feed, xivelyKey);
Serial.print("xivelyclient.put returned ");
Serial.println(ret);
// задержка 60 сек
delay(60000);
}
```

Итак, устанавливаем в разъемы платы Arduino Uno плату Ethernet Shield, подключаем Ethernet Shield к сети, подключаем датчик BH1750 и загружаем на плату Arduino скетч из листинга 6.3. Затем открываем монитор последовательного порта Arduino IDE и наблюдаем процесс отправки данных датчика BH1750 в сервис Xively (рис. 6.14).

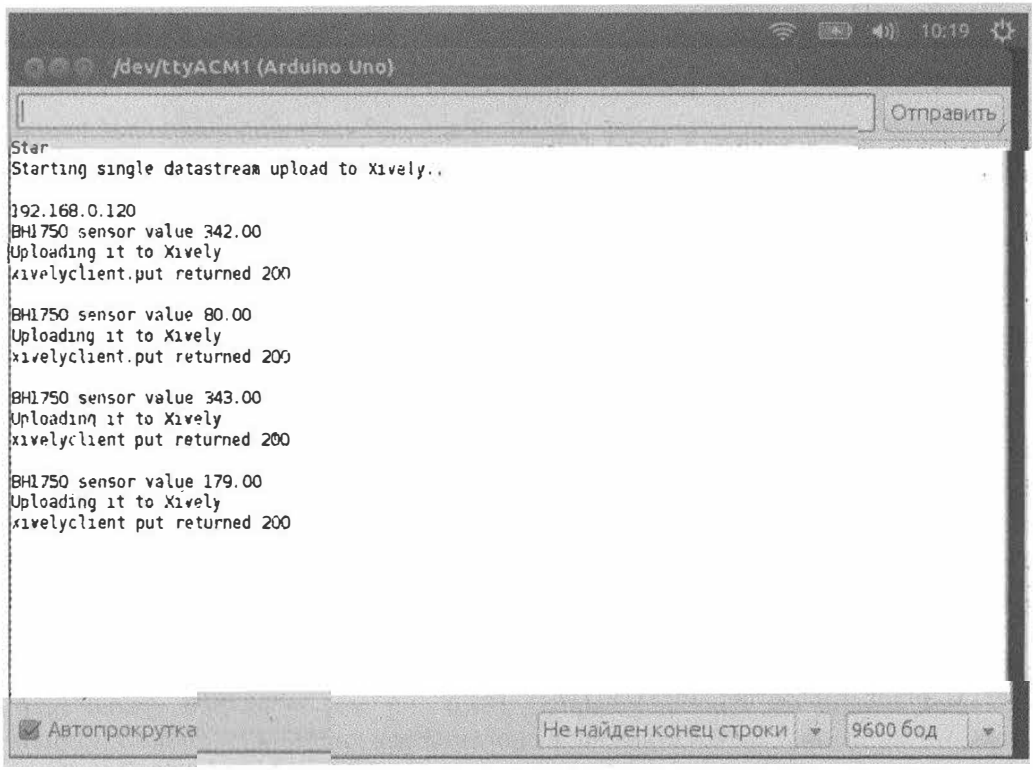


Рис. 6.14. Контроль из монитора последовательного порта процесса отправки данных в сервис Xively

Через некоторое время мы можем зайти в свой профиль в Xively, выбрать устройство и канал BH1750 и увидеть графическое отображение отправленных из Arduino данных (рис. 6.15).

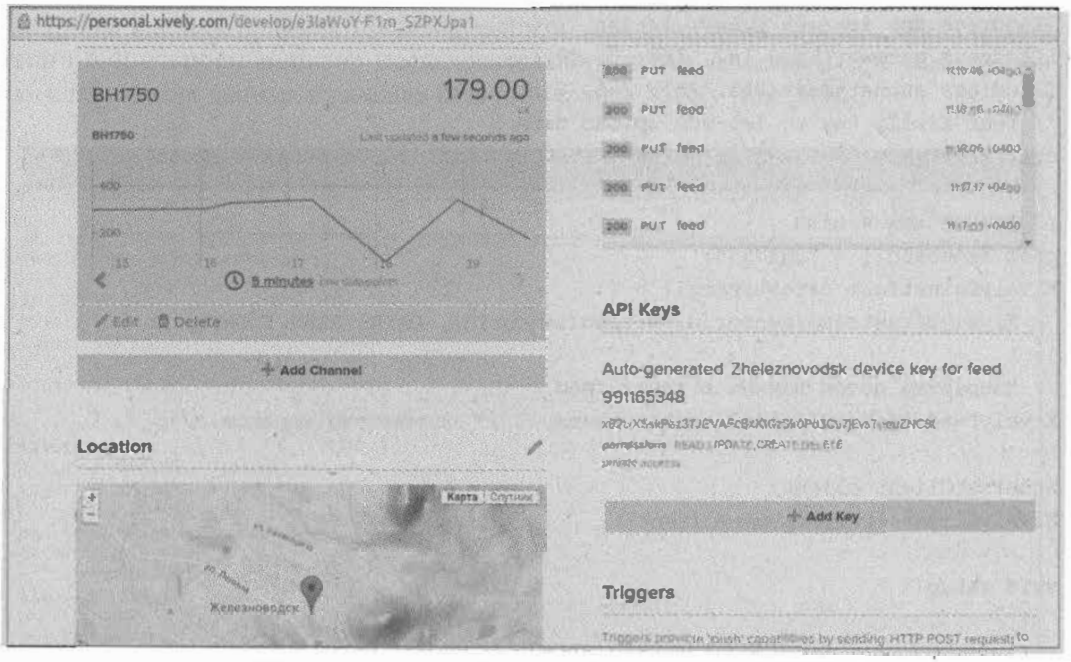


Рис. 6.15. Графическое отображение поступающих в канал данных

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.3, можно найти в файле arduino_scetches_06_06_03.ino сопровождающего книгу электронного архива.

6.4.2. Получение данных из сервиса Xively

Мы можем не только отправлять данные в сервис Xively, но и получать из него данные того или иного канала.

Для этого подключим другую плату Arduino с Ethernet Shield на борту к сети и настроим для нее получение данных из канала BH1750. Содержимое скетча представлено в листинге 6.4.

Листинг 6.4

```
// Подключение библиотек
#include <SPI.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>

// Mac address нашей платы Ethernet Shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
// Настройки сети
IPAddress arduino_ip ( 192, 168, 0, 120);
```

```
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);
// Your Xively key to let you upload data
char xivelyKey[] = "x82lxX5skPez3TJEVAFcBxX1GzSk0Pd3Cu7jEvsTveuZNC8t";

// Данные для канала
char sensorId[] = "BH1750";
XivelyDatastream datastreams[] = {
   XivelyDatastream(sensorId, strlen(sensorId), DATASTREAM_FLOAT),
};
// Направить поток данных в канал feed
XivelyFeed feed(991165348, datastreams, 1 /* количество каналов */);

EthernetClient client;
XivelyClient xivelyclient(client);

void setup()
  {
  // подключение последовательного порта
  Serial.begin(9600);
  Serial.println("Starting single datastream upload to Xively...");
  Serial.println();
  // Подключение к сети
  Ethernet.begin(arduino_mac,arduino_ip,dns_ip,gateway_ip,subnet_mask);
  Serial.println(Ethernet.localIP());
  }

void loop()
  {
  // Получение ответа от Xively
  int ret = xivelyclient.get(feed, xivelyKey);
  Serial.print("xivelyclient.get returned ");
  Serial.println(ret);
  // если есть данные - обработать и вывести
  if (ret > 0)
    {
    Serial.println("Datastream is...");
    Serial.println(feed[0]);
    Serial.print("Light is: ");
    Serial.println(feed[0].getFloat());
    }
  // задержка 60 сек
  delay(60000UL);
  }
```

Загрузим в плату Arduino, предназначенную для получения данных, скетч из листинга 6.4, откроем монитор последовательного порта Arduino IDE и увидим процесс получения данных из канала BH1750 сервиса Xively (рис. 6.16).

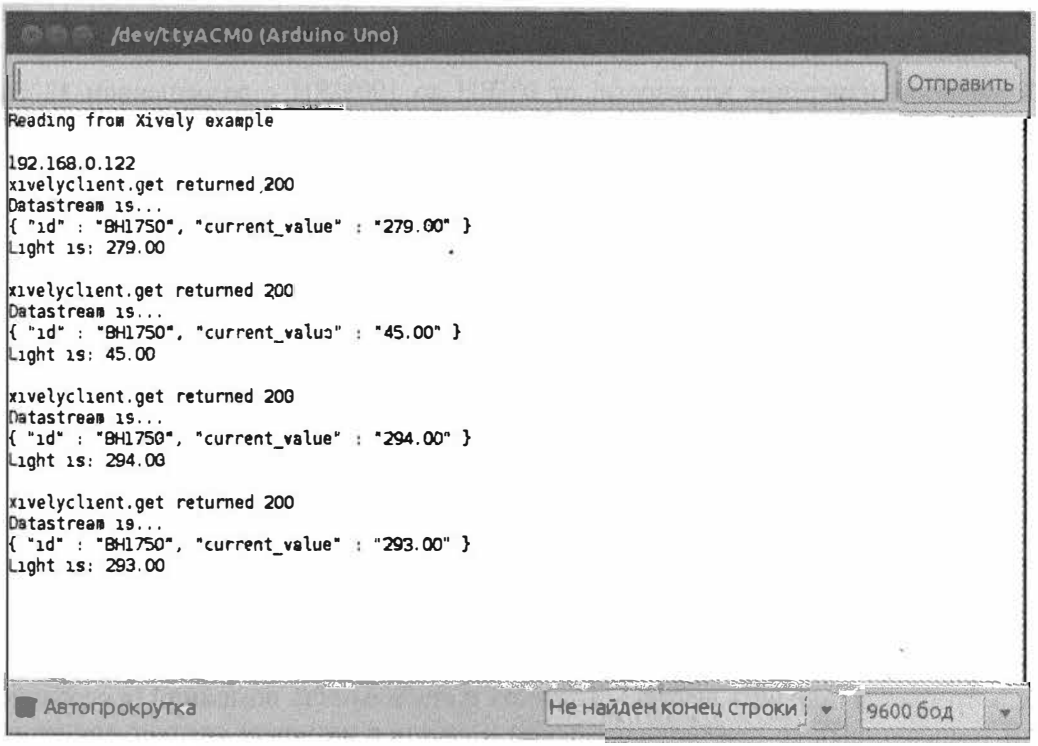


Рис. 6.16. Вывод в последовательный порт данных, полученных из сервиса Xively

Электронный архив

Скетч, соответствующий листингу 6.4, можно найти в файле arduino_scetches_06_06_04.ino сопровождающего книгу электронного архива.

В следующем разделе мы рассмотрим отправку и получение данных для нескольких каналов сервиса Xively.

6.5. Arduino и датчик влажности и температуры SHT21 на шине I²C

Познакомимся с еще одним полезным датчиком, работающим по протоколу I²C, — датчиком SHT21 (рис. 6.17). По заявлению производителя (фирма Sensirion), это самый маленький в мире датчик влажности и температуры. Датчик обладает следующими отличительными особенностями:

- миниатюрный размер: 3×3×1,1 мм — это, как уже отмечено, самый маленький датчик влажности в мире;

- напряжение питания: 2,1–3,6 В;
- всего 6 ножек по бокам корпуса (несмотря на миниатюрный размер, микросхему вполне можно запаять обычным паяльником);
- диапазон измерения температуры: от –40 до +125 °С с разрешением 14 бит (LSB = 0,01 °С);
- диапазон измерения влажности: от 0%RH до 100%RH с разрешением 12 бит (LSB = 0,04%RH);
- потребление в режиме измерения — 300 μА, в спящем режиме — 0,15 μА;
- встроенный детектор разряда батареи выставляет флаг, если напряжение питания опускается ниже 2,25 В;
- встроенный нагреватель для самодиагностики датчиков.

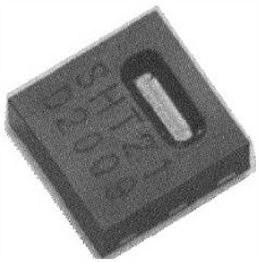


Рис. 6.17. Датчик влажности и температуры SHT21

Если требуется получить высокую точность и стабильность показаний (в особенности температуры), то при разводке платы и работе с датчиком следует придерживаться ряда правил. Это касается не только SHT21, но и любых датчиков температуры/влажности, от которых требуется высокая точность. Дело в том, что показания датчика могут оказаться ложными, если сам датчик и/или воздух вокруг него нагреваются в процессе работы. Причин для этого может быть несколько:

- *саморазогрев датчика из-за непрерывной работы*. Действительно, если мы постоянно заставляем датчик измерять температуру или влажность, пересчитывать их и выдавать данные в линию, то он будет выделять тепло. А из-за маленького размера корпуса (напомню, SHT21 — самый маленький датчик влажности в мире) он скоро начнет нагреваться.

  Решение этой проблемы напрашивается само собой — опрашивайте датчик как можно реже. Хорошо, если отношение времени работы ко времени простоя будет около 10%. Так, SHT21 требуется 70 мс для измерения температуры и 25 мс для измерения влажности. В сумме: 95 мс. Получается, что для того чтобы датчик не разогревался, нам надо проводить замеры не чаще чем раз в 1 секунду. Учитывая, что температура и влажность обычно изменяются медленно, увеличив задержку между опросами, мы ничего не потеряем;
- *влияние источников тепла, расположенных в непосредственной близости от датчика*. Передавать датчику свое тепло могут как различные силовые элемен-

ты, особенно если они работают в линейном режиме и сильно нагреваются, так и мощные контроллеры, работающие на высокой частоте. Большая часть такого тепла передается через медную фольгу на текстолите (печатный монтаж), поэтому рекомендуется размещать датчик как можно дальше от источников тепла и делать информационные дорожки к нему потоньше. При промышленном производстве в плате вокруг датчика делают прорези, отделяя его от остальной платы;

- *передача тепла посредством конвекции*. Проще говоря, горячий воздух от тех же силовых элементов может попасть в область действия датчика и привести к изменению его показаний. Способ борьбы один — отгородить датчик от разогретого воздуха перегородками. Впрочем, если наше устройство не замуровано в корпус, то этой напасти можно не бояться;
- *солнечный свет* тоже очень сильно влияет на показания термометра. Кстати, делать «крышу» над датчиком, если он стоит на открытом воздухе, нужно не только для защиты от солнца, но и от дождя.

Если устройство находится в корпусе, то датчик надо располагать так, чтобы контакт с воздухом из окружающей среды был наибольшим. Самый хороший вариант — когда внешний воздух свободно проходит около датчика, а от нагретого воздуха из корпуса датчик защищен перегородкой.

Для работы с Arduino датчик выпускается в виде модуля (рис. 6.18) с преобразователем питания от 5 В.

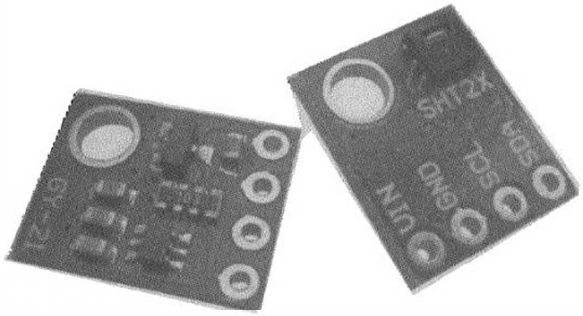


Рис. 6.18. Модуль датчика SHT21

Схема подключения модуля датчика SHT21 к плате Arduino показана на рис. 6.19.

Для работы с модулем можно использовать библиотеку SHT2x. В листинге 6.5 представлен пример считывания с датчика SHT21 данных влажности и температуры и вывод их в монитор последовательного порта.

Листинг 6.5

```
#include <Wire.h>
#include <SHT2x.h>
void setup()
```

```
  {
  Wire.begin();
  Serial.begin(9600);
  }
void loop()
  {
  Serial.print("Humidity(%RH): ");
  Serial.print(SHT2x.GetHumidity());
  Serial.print(" Temperature(C): ");
  Serial.println(SHT2x.GetTemperature());
  delay(1000);
  }
```

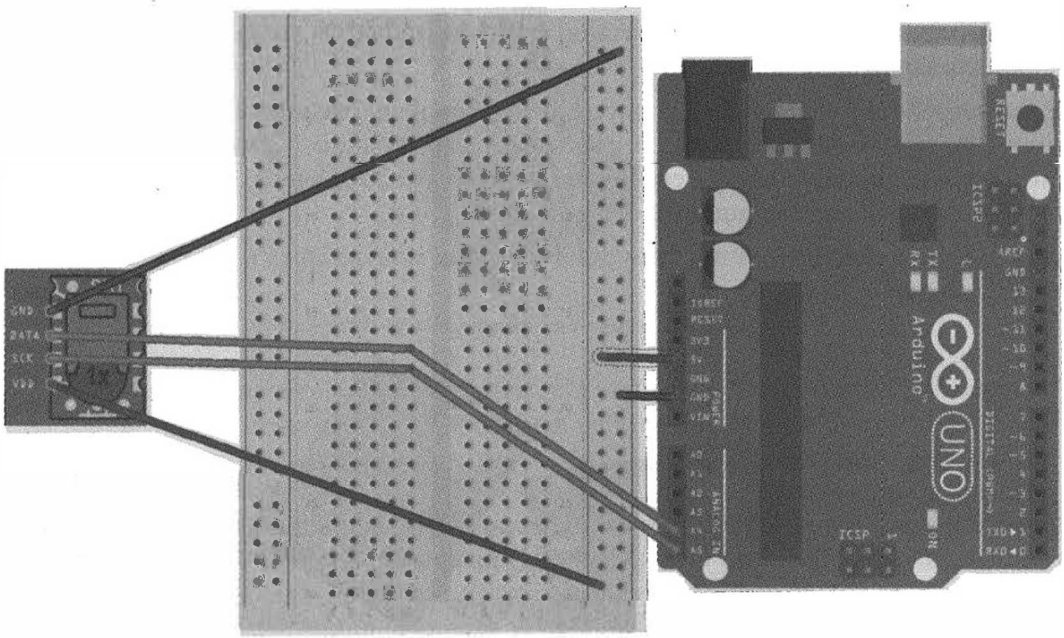


Рис. 6.19. Схема подключения модуля датчика SHT21 к плате Arduino

6.6. Arduino и сервис Xively (продолжение)

6.6.1. Отправка мультиданных в сервис Xively

Добавим в схему, приведенную на рис. 6.7, датчик SHT21 и настроим дополнительно к отправке в сервис Xively данных освещенности (от датчика BH1750) также и отправку данных влажности и температуры. Для этого в сервисе Xively для нашего устройства **Zheleznovodsk** добавим еще два канала: SHT-H — для отправки значений влажности и SHT-T — для отправки значений температуры (рис. 6.20).

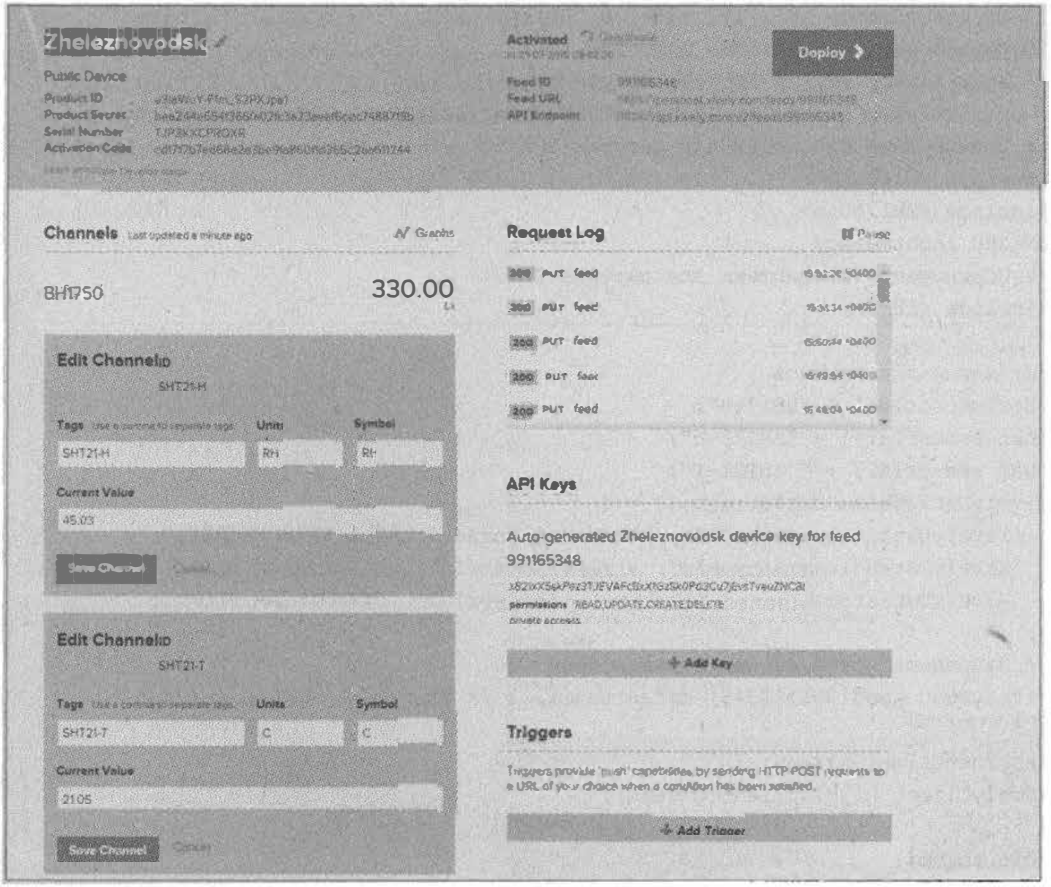


Рис. 6.20. Создание дополнительных каналов для отправки в сервис Xively данных от датчика SHT21

Изменим соответственно и скетч из листинга 6.3, добавив в него отправку данных в несколько каналов устройства Xively. Измененный скетч представлен в листинге 6.6. Загрузим его в плату Arduino и увидим в сервисе Xively графическое отображение поступающих с датчиков BH1750 и SH21 данных (рис. 6.21).

Листинг 6.6

```
// Подключение библиотек
#include <SPI.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>

// Mac address нашей платы Ethernet Shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
// Настройки сети
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
```

```
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);
// Your Xively key to let you upload data
char xivelyKey[] = "x82lxX5skPez3TJEVAFcBxX1GzSk0Pd3Cu7jEvsTveuZNC8t";
// Подключение библиотек для датчика BH1750
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;
// Подключение библиотеки для датчика SHT21
#include <SHT2x.h>

// Данные для каналов
char sensorId[] = "BH1750";
char sensorIdl[] = "SHT21-H";
char sensorId2] = " SHT21-T";
XivelyDatastream datastreams[] = {
   XivelyDatastream(sensorId, strlen(sensorId), DATASTREAM_FLOAT),
   XivelyDatastream(sensorId1, strlen(sensorId1), DATASTREAM_FLOAT),
   XivelyDatastream(sensorId2, strlen(sensorId2), DATASTREAM_FLOAT),
};
// Направить поток данных в канал feed
XivelyFeed feed(991165348, datastreams, 3 /* количество каналов */);

EthernetClient client;
XivelyClient xivelyclient(client);

void setup()
  {
  // подключение последовательного порта
  Serial.begin(9600);
  // запуск датчика BH1750
  lightMeter.begin();

  Serial.println("Starting single datastream upload to Xively...");
  Serial.println();
  // Подключение к сети
  Ethernet.begin(arduino_mac,arduino_ip,dns_ip,gateway_ip,subnet_mask);
  Serial.println(Ethernet.localIP());
  }

void loop()
  {
  // получение данных с BH1750
  uint16_t lux = lightMeter.readLightLevel();
  datastreams[0].setFloat(lux);
  datastreams[1].setFloat(SHT2x.GetHumidity());
  datastreams[2].setFloat(SHT2x.GetTemperature());
```

```
Serial.println("Uploading it to Xively");
// Получение ответа от Xively
int ret = xivelyclient.put(feed, xivelyKey);
Serial.print("xivelyclient.put returned ");
Serial.println(ret);
// задержка 60 сек
delay(60000);
}
```

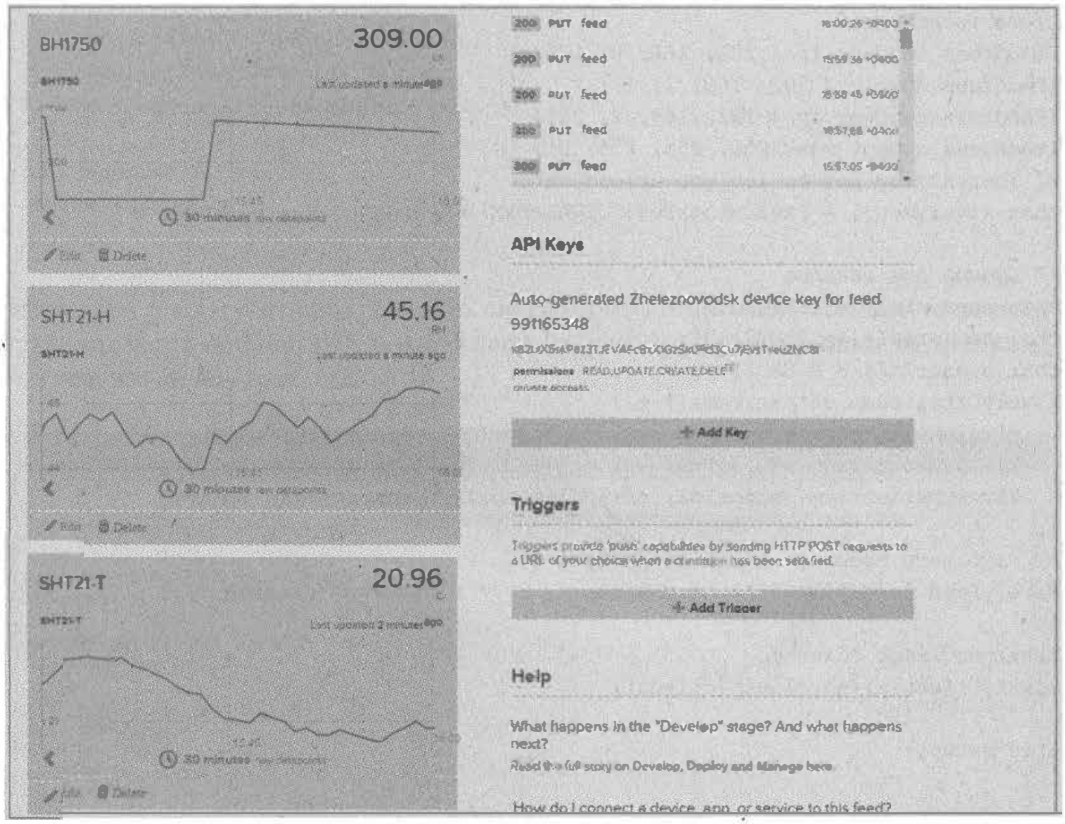


Рис. 6.21. Графическое отображение поступающих в канал данных

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.6, можно найти в файле arduino_scetches_06_06_06.ino сопровождающего книгу электронного архива.

6.6.2. Получение мультиданных из сервиса Xively

Изменим скетч из листинга 6.4 для получения данных из нескольких созданных нами каналов: BH1750, SHT21-H и SHT21-T. Содержимое скетча представлено в листинге 6.7.

Листинг 6.7

```
// Подключение библиотек
#include <SPI.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>

// Mac address нашей платы Ethernet Shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
// Настройки сети
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);
// Your Xively key to let you upload data
char xivelyKey[] = "x82lxX5skPez3TJEVAFcBxX1GzSk0Pd3Cu7jEvsTveuZNC8t";

// Данные для каналов
char sensorId[] = "BH1750";
char sensorId1[] = "SHT21-H";
char sensorId2] = " SHT21-T";
XivelyDatastream datastreams[] = {
   XivelyDatastream(sensorId, strlen(sensorId), DATASTREAM_FLOAT),
   XivelyDatastream(sensorId1, strlen(sensorId1), DATASTREAM_FLOAT),
   XivelyDatastream(sensorId2, strlen(sensorId2), DATASTREAM_FLOAT),
};
// Направить поток данных в канал feed
XivelyFeed feed(991165348, datastreams, 3 /* количество каналов */);

EthernetClient client;
XivelyClient xivelyclient(client);

void setup()
  {
  // подключение последовательного порта
  Serial.begin(9600);
  Serial.println("Starting single datastream upload to Xively...");
  Serial.println();
  // Подключение к сети
  Ethernet.begin(arduino_mac,arduino_ip,dns_ip,gateway_ip,subnet_mask);
  Serial.println(Ethernet.localIP());
  }

void loop()
  {
  // Получение ответа от Xively
  int ret = xivelyclient.get(feed, xivelyKey);
```

```
  Serial.print("xivelyclient.get returned ");
  Serial.println(ret);
  // если есть данные - обработать и вывести
  if (ret > 0)
    {
    Serial.println("Datastream is...");
    Serial.println(feed[0]);
    Serial.print("Light is: ");
    Serial.println(feed[0].getFloat());
    Serial.print("Humidity is: ");
    Serial.println(feed[1].getFloat());
    Serial.print("Temp is: ");
    Serial.println(feed[2].getFloat());
    }
  // задержка 60 сек
  delay(60000UL);
  }
```

Загружаем в плату Arduino скетч из листинга 6.7, открываем монитор последовательного порта Arduino IDE и наблюдаем процесс получения данных из трех каналов сервиса Xively (рис. 6.22).

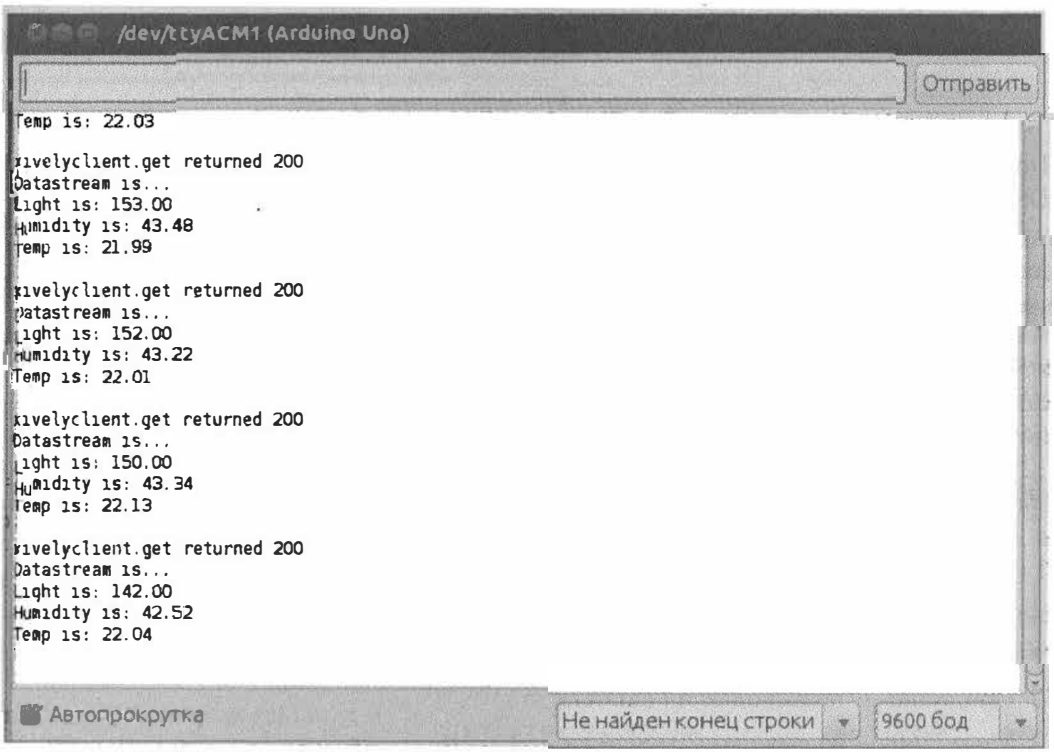


Рис. 6.22. Вывод в последовательный порт данных, полученных из сервиса Xively

Электронный архив

Скетч, соответствующий листингу 6.7, можно найти в файле arduino_scetches_06_06_07.ino сопровождающего книгу электронного архива.

6.7. Arduino и часы реального времени на шине I²C

Познакомимся с еще одним I²C-устройством, весьма часто необходимым при создании систем на Arduino, — микросхемой часов реального времени (RTC). Допустим, ваша плата Arduino собирает данные с подсоединенных к ней датчиков и ведет запись данных на внешний накопитель — например, на SD-карту. Чтобы анализировать находящиеся на карте данные, необходимо записывать их вместе с временной меткой, — тут и пригодится микросхема часов реального времени.

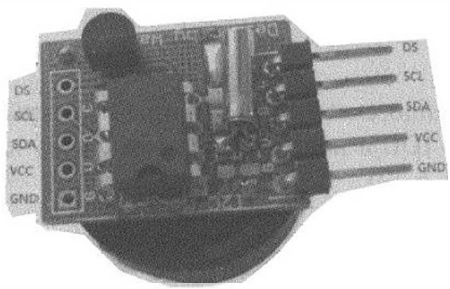


Рис. 6.23. Модуль RTC на микросхеме DS1307

Подключаемая к микроконтроллеру при помощи шины I²C, микросхема Dallas DS1307 (рис. 6.23) представляет собой часы реального времени с календарем и дополнительной памятью nvSRAM (56 байтов). Количество дней в месяце она способна с учетом високосных лет рассчитать до 2100 года. В микросхеме DS1307 имеется встроенная схема, определяющая аварийное отключение питания и автоматически подключающая резервную батарейку. При этом отсчет времени продолжается, и после восстановления питания часы показывают правильное время. Имеется в этой микросхеме и программируемый генератор прямоугольных импульсов, позволяющий вырабатывать одну из четырех частот (1 Гц, 4096 Гц, 8192 Гц или 32 768 Гц). Часы подключаются по протоколу I²C всего двумя проводами — через выводы SCL и SDA интерфейса I²C. Выводы, через которые подключаются часы к шине питания, необходимо дополнительно подтянуть с помощью резисторов 2 кОм.

Контакты SCL и SDA на разных платах Arduino расположены на различных выводах:

- Uno, Nano — на выводах A4 (SDA) и A5 (SCL);
- Mega2560 — на выводах 20 (SDA) и 21 (SCL);
- Leonardo — на выводах 2 (SDA) и 3 (SCL).

Вывод SDA часов подключается к выводу SDA контроллера, а вывод SDL часов — соответственно, к выводу SDL контроллера.

Схема подключения микросхемы часов реального времени DS1307 и жидкокристаллического (LCD) индикатора WH1602 к плате Arduino показана на рис. 6.24.

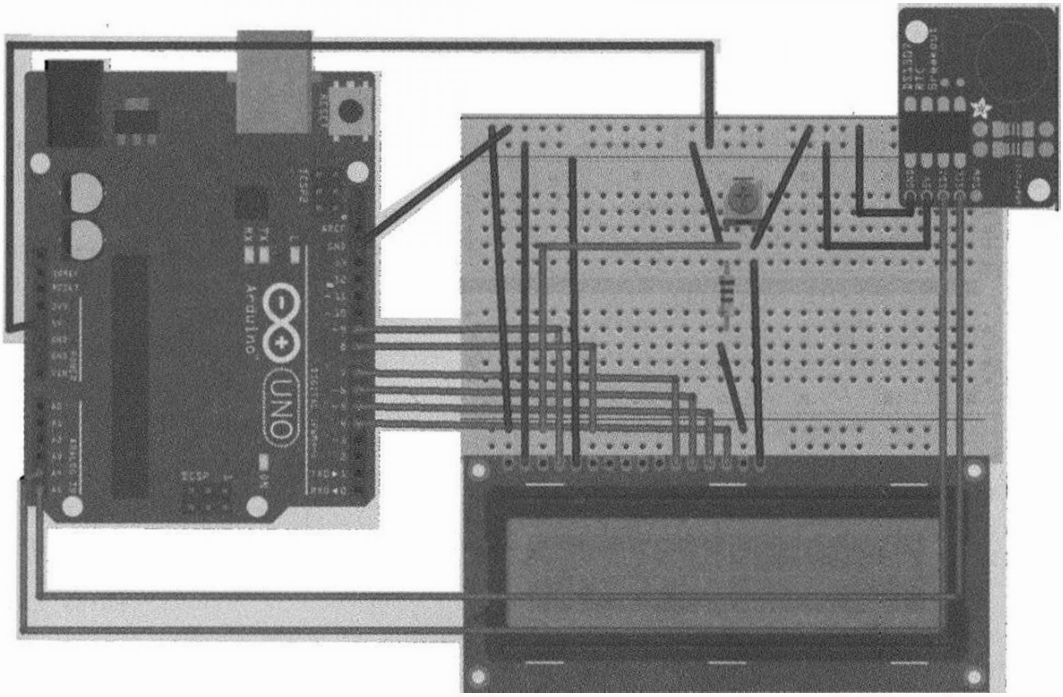


Рис. 6.24. Схема подключения к Arduino модуля часов реального времени DS1307 и жидкокристаллического (LCD) индикатора WH1602

Напишем скетч для вывода даты и времени, получаемых с микросхемы часов реального времени DS1307, на экран LCD-индикатора WH1602 (листинг 6.8). При написании скетча мы воспользуемся библиотекой Time, которая является «оберткой» для библиотеки DS1307, и библиотекой Wire, предназначенной для работы с I²C-устройствами. Для работы с LCD-индикатора нам пригодится также библиотека LiquidCrystal.

Листинг 6.8

```
// подключение библиотек для RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
// подключение библиотеки для lcd
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
```

```
void setup()
  {
  lcd.begin(16, 2);     // установить размерность дисплея
  }
void loop()
  {
  tmElements_t tm;
  if (RTC.read(tm))    // получение времени
    {
    print2digits(tm.Hour,0,0);
    lcd.print(':');
    print2digits(tm.Minute,3,0);
    lcd.print(':');
    print2digits(tm.Second,6,0);
    print2digits(tm.Day,0,1);
    lcd.print('/');
    print2digits(tm.Month,3,1);
    lcd.print('/');
    lcd.print(tmYearToCalendar(tm.Year));
    }
  else
    {
    if (RTC.chipPresent())
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("DS1307 is stopped");
      }
    else
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("DS1307 read error");
      }
    delay(9000);
    }
  delay(1000);
  }
// процедура вывода на дисплей с добавлением до двух цифр
void print2digits(int number,int col, int str)
  {
  lcd.setCursor(col, str);
  if (number >= 0 && number < 10)
    {lcd.print("0");}
    lcd.print(number);
  }
```

При запуске скетча на экране дисплея мы увидим неверное время и неверную дату. Дело в том, что при отсутствии питания значение времени в микросхеме DS1307 сбрасывается на 00:00:00 01/01/2000. Чтобы при отключении питания время не сбрасывалось, предусмотрено аварийное питание модуля от батарейки 3 В.

Для установки времени в библиотеке имеется функция `RTC.write(tmElements_t tm)`. Добавим в наш скетч возможность установки данных RTC по последовательному порту отправкой в модуль строки вида: `dd/mm/YYYY hh:mm:ss`. Содержимое скетча показано в листинге 6.9.

Листинг 6.9

```
// подключение библиотек для RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
// подключение библиотеки для lcd
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
// строка, собираемая из данных, приходящих в последовательный порт
String inputString = "";
boolean stringComplete = false;  // флаг комплектности строки

void setup()
  {
  Serial.begin(9600);       // запустить последовательный порт
  lcd.begin(16, 2);         // установить размерность дисплея
  }

void loop()
  {
  tmElements_t tm;


  // ожидание конца строки для анализа поступившего запроса:
  if (stringComplete)
    {
    tm.Day=(int(inputString[0])-48)*10+(int(inputString[1])-48);
    tm.Month=(int(inputString[3])-48)*10+(int(inputString[4])-48);
    tm.Year=CalendarYrToTm((int(inputString[6])-
            48)*1000+(int(inputString[7])-48)*100+
            (int(inputString[8])-48)*10+(int(inputString[9])-48));

    tm.Hour=(int(inputString[11])-48)*10+(int(inputString[12])-48);
    tm.Minute=(int(inputString[14])-48)*10+(int(inputString[15])-48);
    tm.Second=(int(inputString[17])-48)*10+(int(inputString[18])-48);
    RTC.write(tm);  // записать время в RTC
```

```
    // очистить строку
    inputString = "";
    stringComplete = false;
    }
  if (RTC.read(tm))
    {
    print2digits(tm.Hour,0,0);
    lcd.print(':');
    print2digits(tm.Minute,3,0);
    lcd.print(':');
    print2digits(tm.Second,6,0);
    print2digits(tm.Day,0,1);
    lcd.print('/');
    print2digits(tm.Month,3,1);
    lcd.print('/');
    lcd.print(tmYearToCalendar(tm.Year));
    }
  else
    {
    if (RTC.chipPresent())
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("DS1307 is stopped");
      }
    else
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("DS1307 read error");
      }
    delay(9000);
    }
  delay(1000);
  }
// процедура вывода на дисплей с добавлением до двух цифр
void print2digits(int number,int col, int str)
  {
  lcd.setCursor(col, str);
  if (number >= 0 && number < 10)
    {lcd.print("0");}
  lcd.print(number);
  }
// получение данных по последовательному порту
void serialEvent()
  {
  while (Serial.available())
```

```
    { // получить очередной байт:
    char inChar = (char)Serial.read();
    // добавить в строку
    inputString += inChar;
    // /n - конец передачи
    if (inChar == '\n')
      {stringComplete = true;}
    }
}
```

Теперь установим время из монитора последовательного порта отправкой в модуль строки `dd/mm/YYYY hh:mm:ss` и увидим на экране дисплея отображение верных даты и времени.

ЭЛЕКТРОННЫЙ АРХИВ

Скетчи, соответствующие листингам 6.8 и 6.9, можно найти в файлах arduino_scetches_06_06_08.ino и _06_09.ino сопровождающего книгу электронного архива.

6.8. Arduino и SD-карта: чтение и запись данных

Если вашим Arduino-проектам не хватает памяти (а объем энергонезависимой памяти EEPROM в платах Arduino совсем небольшой), можно использовать внешние носители. И одним из самых простых по подключению к платам Arduino является накопитель на SD-карте: мы можем подсоединиться к SD-карте напрямую, а можем и использовать соответствующие модули.

Итак, подсоединим модуль SD Card к плате Arduino и напишем пример сохранения на SD-карте данных от датчика BH1750. Для анализа данных освещенности потребуется знать и время производства замеров, поэтому мы задействуем и модуль часов реального времени (RTC). Схема, соответствующая нашим пожеланиям, показана на рис. 6.25.

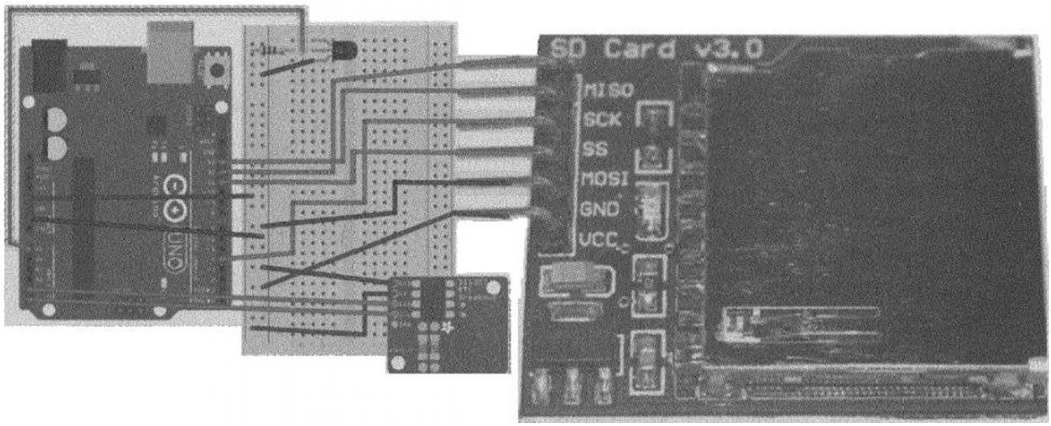


Рис. 6.25. Схема подключения к Arduino модулей SD Card и DS1307

При написании скетча мы воспользуемся библиотекой SD для работы с SD-картой, а также библиотеками Time и DS1307 для работы с модулем RTC. Содержимое скетча показано в листинге 6.10: каждые 5 минут мы считываем данные с датчика BH1750 и заносим время измерения и данные в файл вида d-m-Y. В начале суток создаем новый файл на новый день.

Листинг 6.10

```
// подключение библиотек для RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
// подключение библиотеки для SD
#include <SD.h>

File myFile;
String sfilename;
char filename[20];
const int LM335=A0;   // для подключения LM335
tmElements_t tm;
unsigned long millis1=0;
// Подключение библиотек для датчика BH1750
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;

void setup()
  {
  // запуск датчика BH1750
  lightMeter.begin();
  }

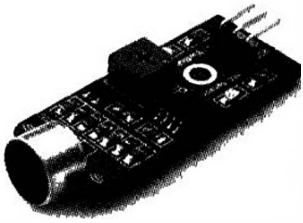
void loop()
 {
 // проверка, прошло 5 минут?
 if(millis()-millis1>5*60*000)
   {
   millis1=millis();
   // получить имя файла для текущего дня
   sfilename=get_file_name();
   sfilename.toCharArray(filename,20);
   // открыть файл или создать
   myFile = SD.open(filename, FILE_WRITE);
   // получить данные BH1750
   uint16_t lux = lightMeter.readLightLevel();
```

```
    // получить время H:m
    // создать запись для файла
    record=get_time();
    record+=" ";
    record+=String(lux);
    myFile.println(record);
    myFile.close();
    }
// получение времени дня
String get_time()
 {
  String time1;
  RTC.read(tm);
  if(tm.Hour()<10)
   time1="0"+String(tm.Hour(),DEC);
  else
   time1=String(tm.Hour(),DEC);
  if(tm.Minute()<10)
   time1+=":0"+String(tm.Minute(),DEC);
  else
   time1+=":"+String(tm,Minute(),DEC);
  return time1;
 }
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.10, можно найти в файле arduino_scetches_06_06_10.ino сопровождающего книгу электронного архива.

ГЛАВА 7



Arduino и 1-Wire

В этой главе мы рассмотрим примеры использования Arduino в связке с устройствами, основанными на протоколе 1-Wire.

7.1. Технология 1-Wire

1-Wire (от англ. — единственный провод) — зарегистрированная торговая марка корпорации Dallas Semiconductor для системотехники шины устройств связи Dallas Semiconductor. Протокол обеспечивает низкоскоростной интерфейс для данных — обычно 16,4 Кбит/с (максимум 125 Кбит/с в режиме overdrive). Для связи с устройством необходимо лишь два провода: на данные и заземление. Дело в том, что интегральная схема включает конденсатор емкостью 800 пФ для питания от линии данных (так называемое *питание от паразитного источника*). При этом электропитание схемы осуществляется за счет конденсатора, который заряжается в период наличия высокого уровня напряжения на линии данных. Здесь следует учитывать, что связь с устройствами, использующими паразитное питание, возможна только на коротких линиях, на длинных же линиях приходится задействовать дополнительный провод питания.

Протокол регламентирован разработчиками для применения в четырех основных сферах-приложениях:

- приборы в специальных корпусах MicroCAN для решения проблем идентификации, переноса или преобразования информации (технология iButton);
- программирование встроенной памяти интегральных компонентов;
- идентификация элементов оборудования и защита доступа к ресурсам электронной аппаратуры;
- системы автоматизации (технология сетей 1-Wire).

Первое из этих направлений широко известно на мировом рынке и уже давно пользуется заслуженной популярностью. Второе — с успехом обеспечивает возможность легкой перестройки функций полупроводниковых компонентов, производимых фирмой Dallas Semiconductor и имеющих малое количество внешних выводов.

Третье — позволяет обеспечить недорогую, но достаточно эффективную идентификацию и надежную защиту самого разнообразного оборудования. Что касается четвертого применения, то реализация локальных распределенных систем на базе шины 1-Wire является на сегодня де-факто наиболее оптимальным решением для большинства практических задач автоматизации. В настоящее время Dallas Semiconductor поставляет широкую номенклатуру однопроводных компонентов различных функциональных назначений для реализации самых разнообразных сетевых приложений. Поэтому имеется огромное число конкретных примеров использования интерфейса 1-Wire для целей автоматизации в самых различных областях, и все больше разработчиков проявляют интерес к этой технологии.

Так в чем же особенность этого сетевого стандарта? Ведь в качестве среды для передачи информации по однопроводной линии чаще всего возможно использование обычного телефонного кабеля и, следовательно, скорость обмена в этом случае не велика. Однако если внимательно проанализировать большинство объектов, требующих автоматизации, то больше чем для 60% из них предельная скорость обслуживания в 16,3 Кбит/с будет более чем удовлетворительной. А вот и другие преимущества технологии 1-Wire:

- простое и оригинальное решение адресуемости абонентов;
- несложный протокол;
- простая структура линии связи;
- малое потребление компонентов;
- легкое изменение конфигурации сети;
- значительная протяженность линий связи;
- исключительная дешевизна всей технологии в целом.

Все эти преимущества отражают очевидную рациональность и высокую эффективность технологии 1-Wire при решении задач комплексной автоматизации в самых различных областях деятельности.

Сеть 1-Wire представляет собой информационную сеть, использующую для осуществления цифровой связи одну линию данных (DATA) и один возвратный (или земляной) провод (RET). Таким образом, для реализации среды обмена этой сети могут быть применены доступные кабели, содержащие неэкранированную витую пару той или иной категории и даже обычный телефонный провод. Такие кабели при их прокладке не требуют наличия какого-либо специального оборудования, а ограничение максимальной длины однопроводной линии регламентировано разработчиками на уровне 300 м.

Основой архитектуры сетей 1-Wire является топология *общей шины*, когда каждое из устройств подключено непосредственно к единой магистрали, без каких-либо каскадных соединений или ветвлений. При этом в качестве базовой используется структура сети с одним ведущим (или мастером) и многочисленными ведомыми. Впрочем, существует и ряд специфических приемов организации работы однопроводных систем в режиме мультимастера.

Конфигурация любой сети 1-Wire может произвольно меняться в процессе ее работы, не создавая помех дальнейшей эксплуатации и работоспособности всей системы в целом, если при этих изменениях соблюдаются основные принципы организации однопроводной шины. Эта возможность достигается благодаря присутствию в протоколе интерфейса 1-Wire специальной команды поиска ведомых устройств (поиск ПЗУ), которая позволяет быстро определить новых участников информационного обмена. Благодаря наличию в составе любого устройства, снабженного интерфейсом 1-Wire, индивидуального уникального адреса (отсутствие совпадения адресов для компонентов, когда-либо выпускаемых Dallas Semiconductor, гарантируется самой фирмой-производителем), такая сеть имеет практически неограниченное адресное пространство. При этом каждый из однопроводных компонентов сразу готов к использованию в составе сети 1-Wire без каких-либо дополнительных аппаратно-программных модификаций.

Однопроводные компоненты представляют собой самотактируемые полупроводниковые устройства, в основе обмена информацией между которыми лежит управление длительностью импульсных сигналов в однопроводной среде и их измерение. Передача сигналов для интерфейса 1-Wire — асинхронная и полудуплексная, а вся информация, циркулирующая в сети, воспринимается абонентами либо как команды, либо как данные. Команды сети генерируются мастером и обеспечивают различные варианты поиска и адресации ведомых устройств, определяют активность на линии даже без непосредственной адресации отдельных компонентов, управляют обменом данными в сети.

Стандартная скорость работы сети 1-Wire, изначально нормированная на уровне 16,3 Кбит/с, была выбрана, во-первых, исходя из обеспечения максимальной надежности передачи данных на большие расстояния, и, во-вторых, с учетом быстродействия наиболее широко распространенных типов универсальных микроконтроллеров, которые в основном должны использоваться при реализации ведущих устройств шины 1-Wire. Эта скорость обмена может быть снижена до любой возможной благодаря введению принудительной задержки между передачей в линию отдельных битов данных (растягиванию временных слотов протокола). Увеличение скорости обмена в сети 1-Wire выше значения 16,3 Кбит/с приводит к сбоям и ошибкам при работе на магистрали 1-Wire длиной более 1 м. Однако если длина линии 1-Wire не превышает 0,5 м, то скорость обмена может быть значительно увеличена за счет перехода на специальный режим ускоренной передачи — overdrive (до 125 Кбит/с), который допускается для отдельных типов однопроводных компонентов. Как правило, такой режим обмена аппаратно реализован для однопроводных компонентов, имеющих большой объем встроенной памяти и предназначенных для эксплуатации на небольшой, но качественной и не перегруженной другими устройствами линии 1-Wire. Типичным примером таких компонентов являются микросхемы семейства iButton.

При реализации интерфейса 1-Wire используются стандартные КМОП/ТТЛ логические уровни сигналов, а питание большинства однопроводных компонентов может осуществляться от внешнего источника с рабочим напряжением в диапазоне от 2,8 до 6,0 В. Причем, такой источник может быть расположен либо непосредственно

возле компонента (например, батарея в составе микросхем iButton), либо энергия от него может поступать по отдельной линии магистрали 1-Wire. Альтернативой применению внешнего питания служит так называемый *механизм паразитного питания*, действие которого заключается в использовании каждым из ведомых компонентов линии 1-Wire электрической энергии импульсов, передаваемых по шине данных, аккумулируемой затем специальной, встроенной в микросхему, емкостью. Кроме того, отдельные однопроводные компоненты сетей 1-Wire могут использовать особый режим питания по шине данных, когда энергия к приемнику поступает непосредственно от мастера по шине DATA магистрали, при этом обмен информацией в сети принудительно прекращается.

7.2. Применение 1-Wire

О признании шины 1-Wire в качестве международного стандарта и серьезности отношения к этому интерфейсу со стороны маститых разработчиков и производителей электроники говорят многочисленные факты. Например, нет практически ни одного универсального микроконтроллера, в литературе по применению которого не обсуждались бы способы организации на его базе мастера линии 1-Wire.

Рационально использование технологии 1-Wire при построении систем автоматизации контроля и управления для разнообразного рассредоточенного оборудования, когда не требуется высокая скорость при обслуживании, но необходима существенная гибкость и наращиваемость при невысоких затратах на реализацию. Приведем некоторые примеры активного применения устройств 1-Wire:

- автоматизация;
- метеостанции;
- многоточечные системы контроля температуры — применяются при мониторинге микроклимата самых различных объектов, при ревизии любых операций по переработке и хранению пищи и медикаментов, для контроля температуры во многих областях промышленного производства, в холодильной технике и т. п.;
- технология «Умный дом» (Home Automation) — комплекс автоматики, который управляет инженерными системами жилища (освещением, отоплением, вентиляцией, кондиционированием, энергоснабжением, водоснабжением, электроприводами, оборудованием пожарных и охранных систем и прочим);
- идентификация и защита электронных изделий — например, маркирование и защита картриджей для принтеров и копировальных аппаратов, а также любых требующих этого электронных изделий;
- менеджмент автономных источников энергии — строгая и полная идентификация источников энергии, сохранение в памяти встроенного в батарею электронного устройства особенностей ее изготовления и индивидуальных технических характеристик, наиболее полный мониторинг их основных эксплуатационных параметров на протяжении всего срока службы, а также формирование коррект-

ного управляющего воздействия, связанного с восстановлением заряда обслуживаемого аккумулятора;

- ограничения и сопряжение с промышленными и глобальными сетями.

Русскоязычную информацию об однопроводных компонентах, наиболее подходящих для организации сетей 1-Wire, можно посмотреть в сети Интернет по адресу **http://www.elin.ru/1-Wire/?topic=component**. Полные списки всех однопроводных устройств и приспособлений для их поддержки можно получить на интернет-сайте компании-производителя Dallas Semiconductor: **http://www.maxim-ic.com**.

Устройства 1-Wire достаточно широко представлены в российских интернет-магазинах по приемлемым, достаточно низким, ценам.

7.3. Интерфейс 1-Wire

Интерфейс 1-Wire, как уже отмечалось ранее, разработан фирмой Dallas Semiconductor (ныне MAXIM) в конце 1990-х годов. Этот интерфейс интересен тем, что для двустороннего обмена по нему требуется всего одна линия, причем на эту линию можно повесить сразу несколько устройств. Правда, потребуются также общий провод («земля») и — но не всегда — провод питания. Протокол очень прост и легко реализуется на микроконтроллере программно. На рис. 7.1 представлена блок-схема аппаратной реализации 1-Wire.

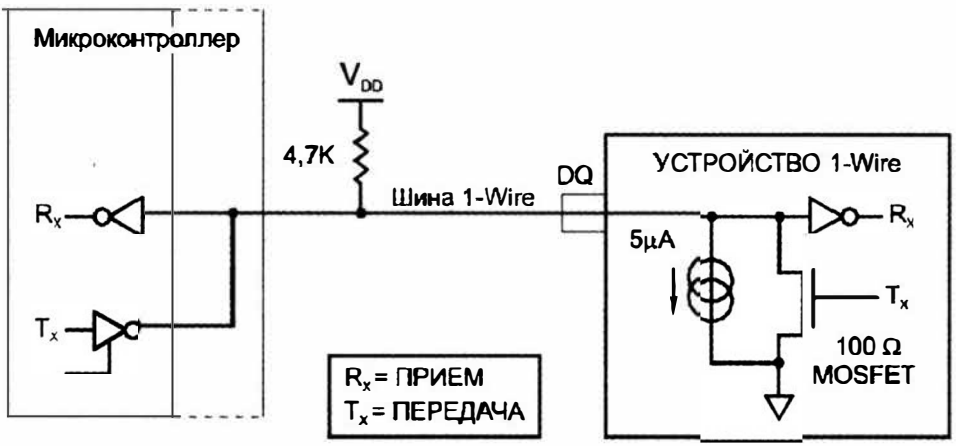


Рис. 7.1. Блок-схема аппаратной реализации 1-Wire

Вывод **DQ** устройства представляет собой вход КМОП-логического элемента, который может быть зашунтирован (замкнут на общий провод) полевым транзистором. Сопротивление канала этого транзистора в открытом состоянии — около 100 Ом. Когда транзистор заперт, имеется небольшой ток утечки (примерно 5 мкА) на общий провод.

Шина 1-Wire должна быть «подтянута» отдельным резистором к напряжению питания (может быть от 3 до 5 В — нужно уточнять по характеристикам подключае-

мого устройства). Сопротивление этого резистора 4,7 К, однако это значение подходит лишь для весьма коротких линий. Если шина используется для подключения устройств на большее расстояние, то сопротивление подтягивающего резистора необходимо уменьшить (это сопротивление зависит от величины максимального втекающего тока линии **DQ** конкретного устройства 1-Wire).

Примечательный момент — как уже отмечалось, некоторые устройства 1-Wire могут использовать так называемое *паразитное/фантомное питание* (Parasite power) — при котором питание устройства осуществляется от линии данных за счет встроенного конденсатора, который заряжается в период наличия высокого уровня напряжения на линии данных. Здесь следует учитывать, что связь с устройствами, использующими паразитное питание, допустима только на коротких линиях. На длинных линиях могут присутствовать непонятные побочные эффекты. Поэтому, если возможно, такого типа питания устройств следует избегать.

7.3.1. Обмен информацией по шине 1-Wire

Рассмотрим, как происходит обмен информацией по шине 1-Wire.

- Передача информации возможна только выдачей низкого уровня в линию, т. е. замыканием ее на общий провод. В высокий логический уровень линия вернется сама из-за наличия подтягивающего резистора (теперь становится понятно, что наличие внешнего подтягивающего резистора — обязательное условие работы 1-Wire).
- Обмен происходит по инициативе ведущего устройства (обычно — микроконтроллера).
- Обмен информацией начинается с подачи импульса сброса (`RESET pulse`) на линию.
- Устройства 1-Wire предусматривают «горячее» подключение.
- При подключении устройство 1-Wire выдает в линию **DQ** импульс присутствия (`PRESENCE pulse`). Этот же импульс выдается при обнаружении сигнала `RESET`.
- Обмен информацией ведется так называемыми *тайм-слотами* — один слот содержит один бит информации.
- Данные передаются побайтно — бит за битом, начиная с младшего байта. Достоверность данных (проверка отсутствия искажений) гарантируется путем подсчета циклической контрольной суммы (CRC).

Микроконтроллер (МК) формирует импульс `RESET`, переводя в низкий логический уровень шину 1-Wire и удерживая ее не менее 480 микросекунд. Затем МК «отпускает» шину, и напряжение возвращается к высокому логическому уровню — время «отпускания» зависит от емкости линии и сопротивления подтягивающего резистора. Протокол 1-Wire ограничивает это время диапазоном от 15 до 60 микросекунд, что и влияет на выбор подтягивающего резистора (на время возврата линии к высокому уровню большее влияние оказывает емкость линии, но, чаще всего, мы изменить ее не можем). Обнаружив импульс `RESET`, ведомое устройство формирует от-

ветный импульс PRESENCE. Для этого устройство «прижимает» линию **DQ** к «земле» и удерживает ее в таком состоянии от 60 до 240 микросекунд. Затем устройство так же «отпускает» шину. После этого устройству еще дается время для завершения внутренних процедур инициализации. Таким образом, МК должен приступить к любому обмену с устройством не ранее, чем через 480 микросекунд после завершения импульса RESET. Соответственно, процедура инициализации, с которой начинается обмен данными между устройствами, длится минимум 960 микросекунд (рис. 7.2).



Рис. 7.2. Процедура инициализации

Теперь рассмотрим процедуры обмена битами информации, которые осуществляются определенными тайм-слотами — жестко лимитированными по времени последовательностями смены уровней сигнала в линии 1-Wire (рис. 7.3).

Различают 4 типа тайм-слотов:

- передача «1» от МК;
- передача «0» от МК;
- прием «1» от устройства;
- прием «0» от устройства.

Тайм-слот всегда начинает микроконтроллер, «прижимая» шину к «земле». Длительность тайм-слота находится в пределах от 60 до 120 микросекунд. Между тайм-слотами всегда должен быть интервал не менее 1 микросекунды (определяется параметрами ведомого устройства). Тайм-слоты передачи отличаются от тайм-слотов приема поведением микроконтроллера:

- при передаче МК только формирует сигналы;
- при приеме МК еще и опрашивает уровень сигнала в линии 1-Wire.

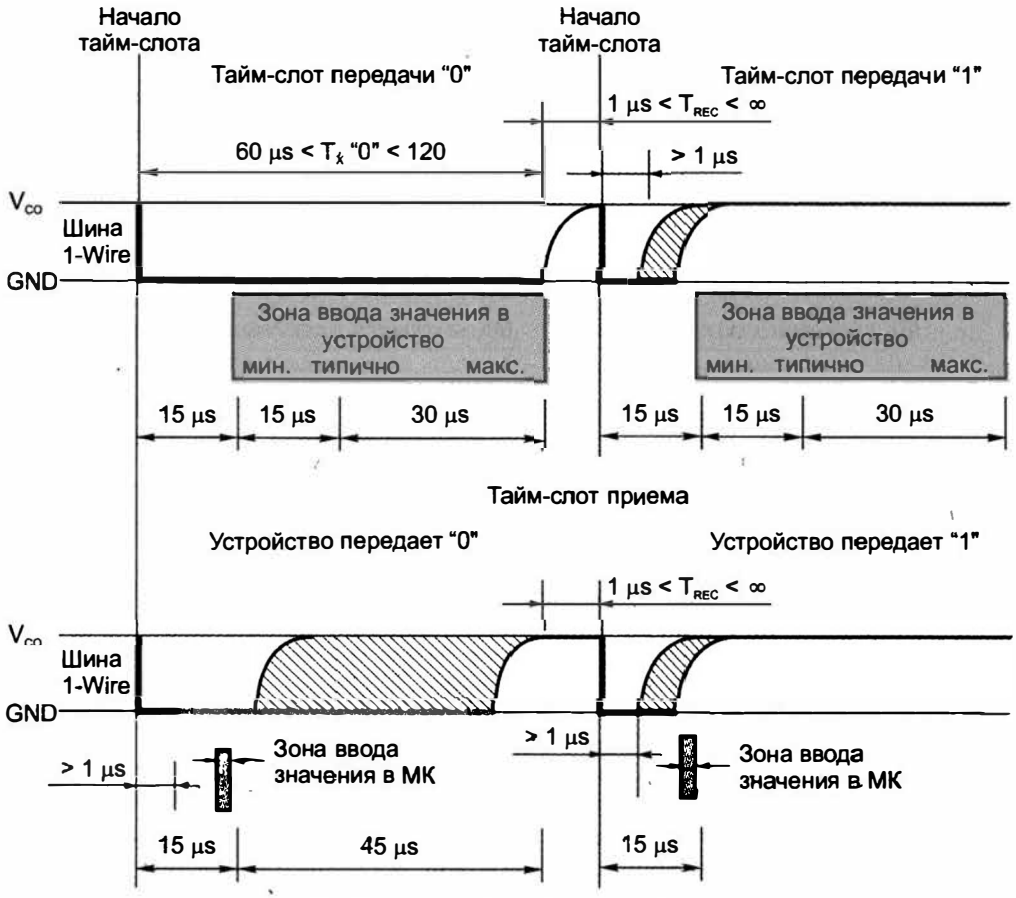


Рис. 7.3. Процедура обмена битами информации

Тайм-слот передачи «0» заключается просто в «прижимании» шины 1-Wire к «земле» в течение всей длительности тайм-слота. Передача «1» осуществляется путем «отпускания» шины 1-Wire со стороны МК не ранее чем через 1 микросекунду после начала тайм-слота, но не позже чем через 15 микросекунд. Ведомое устройство опрашивает уровень в шине 1-Wire в течение временного интервала (показанного на рис. 7.3 в виде серого прямоугольника), т. е. начиная с 15-й микросекунды от начала тайм-слота и заканчивая 60-й микросекундой его от начала (для большинства устройств — около 30-й микросекунды от начала тайм-слота).

Заштрихованная на рис. 7.3 область — это область «нарастания» уровня в шине 1-Wire, которая зависит от емкости линии и сопротивления подтягивающего резистора. Тайм-слоты приема информации отличаются тем, что МК формирует только начало тайм-слота (так же, как при передаче «1»), а затем управление уровнем шины 1-Wire берет на себя устройство, а МК осуществляет ввод этого уровня так же в определенной зоне временных интервалов. Зона эта, как видно из рис. 7.3, довольно мала. Поскольку заштрихованная область — область неопределенности, то для ввода микроконтроллеру остается даже не промежуток, а скорее конкретный мо-

мент, когда он должен ввести уровень сигнала из линии. Этот момент времени — 14-я или 15-я микросекунда от начала тайм-слота.

Резюмируем:

- микроконтроллер начинает тайм-слот, «прижимая» шину 1-Wire к логическому «0» в течение 1 микросекунды;
- последующий уровень зависит от типа тайм-слота: для приема и передачи «1» уровень должен стать высоким, а для передачи «0» — оставаться низким вплоть до конца тайм-слота, т. е. от 60 до 120 микросекунд;
- принимая данные, МК должен считывать уровень в шине 1-Wire в промежутке от 13-й до 15-й микросекунды тайм-слота;
- микроконтроллер должен обеспечить интервал между тайм-слотами не менее 1 микросекунды (лучше — больше, максимальное значение не ограничено).

Для достижения требуемых временных интервалов надо следовать простым правилам:

- все сигналы, которые должен формировать МК, следует формировать по принципу необходимого минимума длительности (т. е. немного больше, чем указанная минимальная длительность);
- от устройства следует ожидать сигналов по принципу наихудшего (т. е. ориентироваться на самые худшие варианты временных параметров сигнала).

7.3.2. Протокол обмена информацией 1-Wire

Каждое устройство 1-Wire обладает уникальным идентификационным 64-битным номером, программируемым на этапе производства микросхемы:

- первые 8 битов — номер серии устройства (список некоторых кодов семейств 1-Wire-устройств приведен в табл. 7.1);
- следующие 48 битов — уникальный серийный номер;
- последние 8 битов — CRC-код предыдущих 56 битов информации.

Фирма-производитель гарантирует, что не найдется двух микросхем с одинаковым идентификационным номером. Нетрудно посчитать, что устройств одного типа может быть выпущено 281 474 976 710 655 (десятичное представление 0xFFFFFFFFFFFF — 48 битов, или 6 байтов идентификационного номера).

Предположим, что на шине 1-Wire имеется более одного устройства. В этом случае перед микроконтроллером встают две проблемы: определение количества имеющихся устройств и выбор (адресация) одного конкретного из них для обмена данными. Номера некоторых устройств наносятся прямо на корпус микросхем (например, для ключей-таблеток — iButton), а номера других можно определить при помощи специальных программ или устройств. Итак, пусть мы знаем номера всех устройств 1-Wire на шине. Алгоритм работы с ними следующий:

1. Микроконтроллер посылает импульс RESET, и все имеющиеся устройства выдают PRESENCE.

2. Микроконтроллер посылает в шину команду, которую принимают все устройства. Определено несколько общих команд для всех типов устройств 1-Wire, есть так же и команды, уникальные для отдельных типов устройств.
3. После того, как микроконтроллер выдаст команду READ ROM, от устройства поступит 8 байтов его собственного уникального адреса — микроконтроллер должен их принять. Любая процедура обмена данными с устройством должна быть завершена полностью либо прервана посылкой сигнала RESET.
4. Если отправлена команда MATCH ROM, то после нее микроконтроллер должен передать 8 байтов адреса конкретного устройства, с которым будет осуществляться последующий обмен данными.
5. Приняв эту команду, каждое устройство сравнивает передаваемый адрес со своим собственным. Все устройства, адрес которых не совпал, прекращают анализ и выдачу сигналов в линии 1-Wire, а опознавшее адрес устройство продолжает работу. Теперь все данные, передаваемые МК, будут попадать только к этому «адресованному» устройству.
6. Если устройство на шине единственное — можно ускорить процесс взаимодействия с ним при помощи команды SKIP ROM. Получив эту команду, устройство сразу считает адрес совпавшим, хотя никакого адреса за этой командой не следует. Некоторые процедуры не требуют приема от устройства никаких данных, в этом случае команду SKIP ROM можно использовать для передачи какой-то информации сразу всем устройствам — например, для одновременного запуска цикла измерения температуры несколькими датчиками-термостатами типа DS18S20.

Общие команды для всех типов устройств 1-Wire представлены в табл. 7.1.

Таблица 7.1. Общие команды для всех типов устройств 1-Wire

Команда	Значение байта	Описание
SEARCH ROM	0xF0	Поиск адресов — используется при универсальном алгоритме определения количества и адресов подключенных устройств
READ ROM	0x33	Чтение адреса устройства — используется для определения адреса единственного устройства на шине
MATCH ROM	0x55	Выбор адреса — используется для обращения к конкретному адресу устройства из многих подключенных
SKIP ROM	0xCC	Игнорировать адрес — используется для обращения к единственному устройству на шине, при этом адрес устройства игнорируется (можно обращаться к неизвестному устройству)

Прием и передача байтов всегда начинается с младшего бита. Порядок следования байтов при передаче и приеме адреса устройства так же ведется от младшего к старшему. Порядок передачи другой информации зависит от конкретного устройства.

Алгоритм поиска устройств 1-Wire следующий:

1. Поиск начинается с импульса RESET от ведущего устройства и принятия PRESENCE от ведомых.
2. Затем посылается 1 байт команды:
   - 0xF0 — осуществляется поиск всех устройств на линии;
   - 0xEC — поиск среди устройств, находящихся в состоянии тревоги (alarm state).
3. Устройства отправляют первый бит своего уникального номера. Если несколько устройств передают свой бит одновременно, результирующий бит на линии получится как результат операции логического И (AND).
4. Следующий бит, который отправляют устройства, — это дополнение первого бита (если первый бит был 1, то будет 0 и наоборот: если был 0 — теперь будет 1). На основании этих двух битов ведущее устройство может сделать вывод о первом бите устройств на линии.
5. Далее микроконтроллер отправляет принятый бит назад. И теперь продолжат работу только те ведомые устройства, у которых этот бит установлен. Если же устройство такого бита не имеет, оно должно перейти в режим ожидания до следующего сигнала RESET.
6. Такая «двубитная передача» повторяется для всех следующих 63 битов ROM.
7. Все устройства на линии, кроме одного, перейдут в состояние ожидания, а код ROM этого единственного устройства будет известен.

7.4. Arduino и цифровой датчик температуры DS18B20

7.4.1. Цифровой датчик температуры DS18B20

DS18B20 (рис. 7.4) — цифровой термометр с программируемым разрешением от 9 до 12 битов, способный сохранять данные замеров в памяти EEPROM прибора. DS18B20 обменивается данными по шине 1-Wire и при этом может быть как единственным устройством на линии, так и работать в группе. Все процессы на шине управляются центральным микропроцессором.

Диапазон измерений датчика: от –55 до +125 °C с точностью 0,5°C в диапазоне от –10 до +85 °C. В дополнение DS18B20 может питаться напряжением линии данных (так называемое *питание от паразитного источника*) при отсутствии внешнего источника напряжения.

Каждый датчик типа DS18B20 имеет уникальный 64-битный последовательный код, который позволяет общаться с множеством датчиков DS18B20, установленных на одной шине. Первые 8 битов — код серии (для DS18B20 — 28h), затем 48 битов уникального номера и в конце 8 битов CRC-кода. Такой принцип позволяет ис-

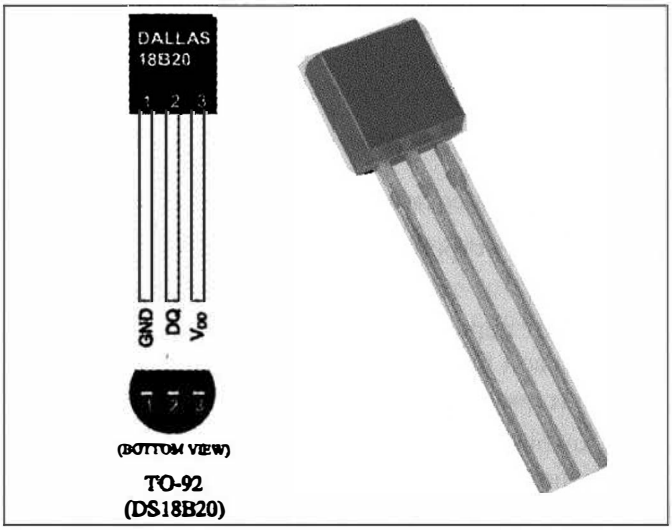


Рис. 7.4. Цифровой датчик температуры DS18B20

пользовать один микропроцессор, чтобы контролировать множество датчиков DS18B20, распределенных по большому участку.

Характеристики датчика DS18S20:

- интерфейс 1-Wire;
- измеряемая температура: от –55 до +125 °С;
- точность 0,5 °С в диапазоне от –10 до +85 °С;
- температура считывается 9-ю битами данных;
- время на конвертацию температуры — 750 мс (максимальное).

Данные о температуре хранятся в оперативной памяти датчика (рис. 7.5). Память состоит из оперативной ROM и энергонезависимой EEPROM:

- первые два байта — содержат данные об измеренной температуре;
- третий и четвертый байты хранят верхний (TH) и нижний (TL) пределы температуры;
- пятый и шестой — не задействованы;
- седьмой и восьмой — байты-счетчики. Они могут использоваться для более точного измерения температуры;
- девятый байт хранит CRC-код предыдущих восьми.

Кроме общих для всех типов 1-Wire-устройств команд, представленных в табл. 7.1, датчик может выполнять следующие команды:

- `Alarm Search [ECh]` — операция этой команды идентична операции поиска адресов `[F0h]`, за исключением того, что в данном случае ответят только те датчики,

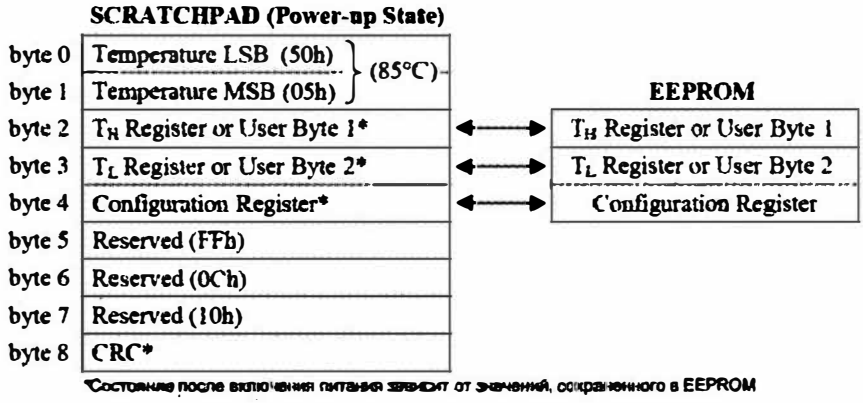


Рис. 7.5. Карта памяти DS18B20

у которых при последнем измерении температура вышла за установленные пределы (выше TH или ниже TL);

- `Convert T [44h]` — конвертировать температуру. Датчик произведет измерение и запись данных о текущей температуре. Если ведущее устройство будет за этой командой слать тайм-слоты чтения, то пока конвертация не закончена, DS18S20 будет выдавать в линию «0», а после завершения конвертации «1»;
- `Write Scratchpad [4Eh]` — запись в память. Эта команда позволяет записать 3 байта в память датчика. Первый байт запишется в TH, второй — в TL, а третий байт запишется в пятый байт памяти — это байт конфигурации;
- `Read Scratchpad [BEh]` — чтение памяти. Команда позволяет нам считать память датчика. В ответ на эту команду датчик вернет 9 байтов своей памяти, начиная с 0-го байта `TEMPERATURE LSB` и заканчивая восьмым — CRC;
- `Copy Scratchpad [48h]` — копировать память. Датчик скопирует содержимое ОЗУ — TH и TL в EEPROM.

Для получения от датчика данных о температуре необходимо выдать следующую последовательность команд:

1. Произвести `RESET` и поиск устройств на линии 1-Wire.
2. Выдать команду `0x44`, чтобы запустить конвертацию температуры датчиком.
3. Подождать не менее 750 мс.
4. Выдать команду `0xBE`, чтобы считать ОЗУ датчика (данные о температуре будут в первых двух байтах).

Датчик может запитываться двумя способами: внешним питанием (три провода) или паразитным (питание от шины, два провода). Схема подключения к Arduino датчика, запитанного от внешнего источника, представлена на рис. 7.6.

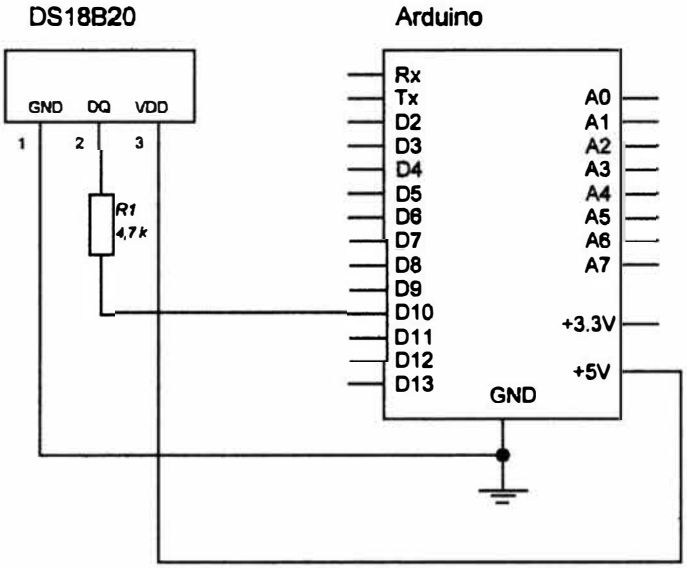


Рис. 7.6. Подключение к Arduino датчика DS18B20

7.4.2. Использование библиотеки OneWire для получения данных температуры с датчика DS18B20

Для работы с датчиками по интерфейсу 1-Wire можно использовать библиотеку OneWire.

> ***ЭЛЕКТРОННЫЙ АРХИВ***
>
> Библиотеку OneWire вы найдете в папке arduino_libraries\OneWire сопровождающего книгу электронного архива.

Скетч получения данных с датчика температуры DS18B20 и вывода данных в последовательный порт с помощью библиотеки OneWire представлен в листинге 7.1.

Листинг 7.1

```
#include <OneWire.h>
OneWire ds(10); // линия 1-Wire будет на pin 10

void setup(void)
{
 Serial.begin(9600);
}

void loop(void)
{
 byte i;
```

```
byte present = 0;
byte data[12];
byte addr[8];

if ( !ds.search(addr)) {
  //Serial.print("No more addresses.\n");
  ds.reset_search();
  return;
}
if ( OneWire::crc8( addr, 7) != addr[7]) {
  Serial.print("CRC is not valid!\n");
  return;
}

if ( addr[0] != 0x28) {
  Serial.print("Device is not a DS18B20 family device.\n");
  return;
}

ds.reset();
ds.select(addr);
ds.write(0x44,1);     // запускаем конвертацию

delay(750);  // ждем 750 мс

present = ds.reset();
ds.select(addr);
ds.write(0xBE);     // считываем ОЗУ датчика

for ( i = 0; i < 9; i++) {      // обрабатываем 9 байтов
 data[i] = ds.read();
}

// высчитываем температуру :)
int HighByte, LowByte, TReading, Tc_100;
LowByte = data[0];
HighByte = data[1];
TReading = (HighByte << 8) + LowByte;
Serial.print(Temp/16);
Serial.print(".");
Serial.print(((Temp%16)*100)/16);
Serial.println();
}
```

Вывод показаний датчика температуры DS18B20 в последовательный порт показан на рис. 7.7.

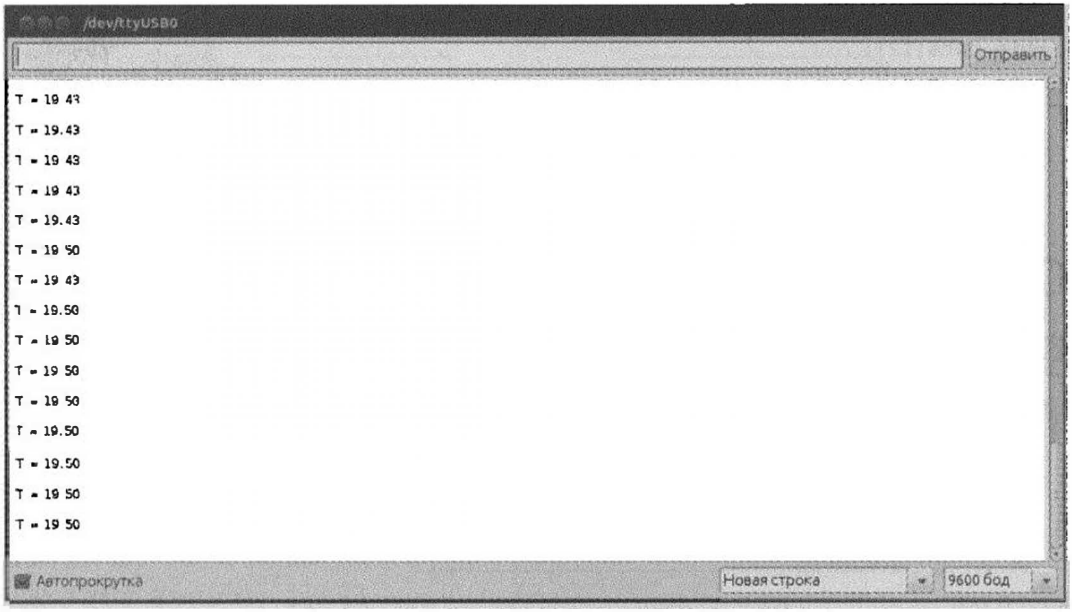
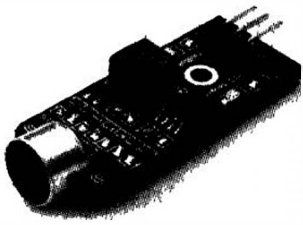


Рис. 7.7. Вывод показаний датчика DS18B20 в последовательный порт

В *главе 8* мы рассмотрим использование датчика DS18B20 в проекте сервера Internet of Thing, собирающего данные с Ethernet-модулей различных датчиков, подключенных к плате Arduino.

ГЛАВА 8



Сервер для сбора данных с Ethernet-модулей датчиков, установленных на Arduino

В этой главе мы создадим веб-сервер для сбора и отображения данных с Ethernet-модулей датчиков, сопряженных с платами Arduino. Каждый модуль представляет собой плату Arduino, состыкованную с платой Ethernet Shield, к которой подключены несколько датчиков. Будут использованы следующие датчики:

- температуры DS18B20;
- влажности DHT11 (или DHT22);
- освещенности BH1750;
- движения HC-SR501;
- звука FC-04.

На каждом модуле установлен веб-сервер, отдающий показания в формате JSON при обращении к нему с центрального сервера (может быть использован любой компьютер). На центральном сервере можно посмотреть отображение собираемых с нескольких модулей данных в графическом и табличном виде.

Датчик измерения освещенности BH1750 мы рассмотрели в *главе 6*, датчик температуры DS18B20 — в *главе 7*. Далее мы кратко познакомимся с другими датчиками, задействованными в этом проекте.

8.1. Датчики влажности DHT11 и DHT22

Датчики DHT11 (рис. 8.1) и DHT22 (рис. 8.2) не отличаются высоким быстродействием и точностью, однако могут найти свое применение в радиолюбительских проектах из-за своей невысокой стоимости. Конструкция этих датчиков основана на емкостном датчике влажности и термисторе. Кроме того, они содержат в себе простенький АЦП для преобразования аналоговых значений влажности и температуры в их цифровое представление.

Датчики DHT имеют по 4 вывода стандарта 2,54 мм:

- 1 — **VCC** (питание 3–5 В);
- 2 — **DATA** (вывод данных);
- 3 — не используется;
- 4 — **GND** («земля»).

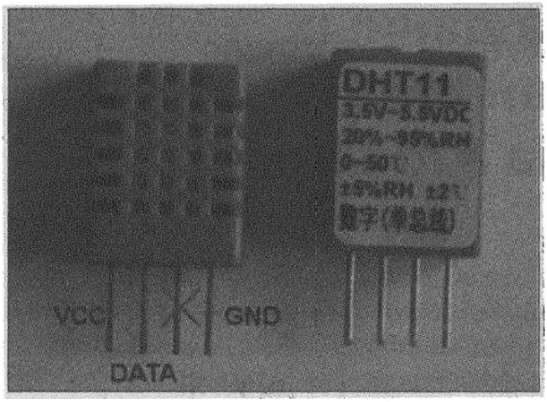


Рис. 8.1. Датчик влажности DHT11

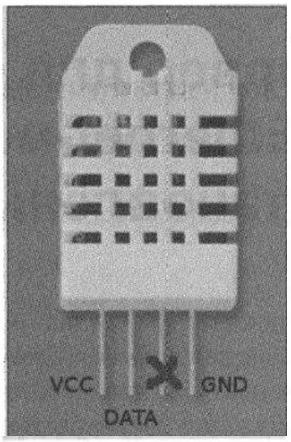


Рис. 8.2. Датчик влажности DHT22

Протокол обмена — однопроводный, по структуре он весьма схож с протоколом датчика температуры DS18B20, но с важными оговорками:

- датчики DHT не умеют работать в «паразитном» режиме (питание по линии данных);
- каждый датчик температуры DS18B20 имеет персональный идентификатор, что дает возможность подключения нескольких таких датчиков к одному выводу Arduino. Однако у датчиков DHT такой возможности нет — один датчик использует строго один цифровой вывод.

Общие характеристики датчиков:

- DHT11:
  - весьма низкая стоимость;
  - питание и I/O: 3–5 В;
  - определение влажности: 20–80% с точностью 5%;
  - определение температуры 0–50 °С с точностью 2%;
  - частота опроса не более 1 Гц (не более одного раза в 1 сек.);
  - размеры 15,5×12×5,5 мм;
- DHT22:
  - низкая стоимость;
  - питание и I/O: 3–5 В;

- определение влажности: 0–100% с точностью 2–5 %;
- определение температуры: от –40 до +125 °С с точностью ±0,5 °С;
- частота опроса не более 0,5 Гц (не более одного раза в 2 сек.);
- размеры 15,1×25×7,7 мм.

Сенсор DHT22 имеет лучшие, чем DHT11 характеристики, но более высокую стоимость.

8.1.1. Подключение датчиков DHT к Arduino

Рекомендуемая схема подключения датчиков DHT к Arduino (рис. 8.3) содержит обязательный для однопроводных линий резистор-подтяжку к VCC и, в качестве опции, рекомендуется конденсатор (фильтр по питанию между VCC и GND).

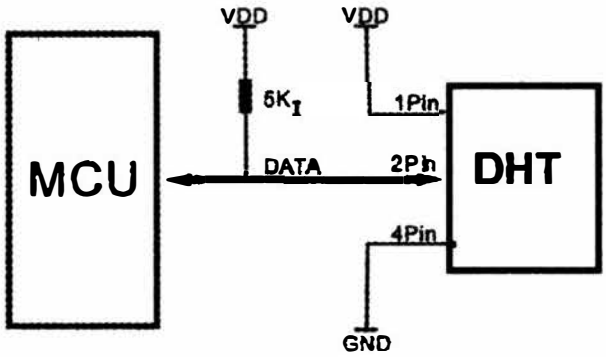


Рис. 8.3. Схема подключения к Arduino

Если к вам в руки попали DHT11 или DHT22 на небольшой плате, можно подключать их к Arduino напрямую — резистор и конденсатор там уже и так есть.

8.1.2. Библиотека DHT

Для работы Arduino с датчиками DH11 и DH22 существует готовая библиотека (DHT), скачать которую можно со страницы: **https://github.com/adafruit/DHT-sensor-library**.

> ***Электронный архив***
> Библиотеку DHT вы найдете в папке arduino_libraries\DHT сопровождающего книгу электронного архива.

Скетч получения данных с датчика температуры и влажности DHT22 и вывода полученных данных в последовательный порт представлен в листинге 8.1.

Для использования в этом скетче датчика DHT11 необходимо закомментировать строку:

```
//#define DHTTYPE DHT22 // DHT 22 (AM2302)
```

и раскомментировать:

```
#define DHTTYPE DHT11 // DHT 11
```

Вывод показаний датчика температуры и влажности DHT22 в последовательный порт показан на рис. 8.4.

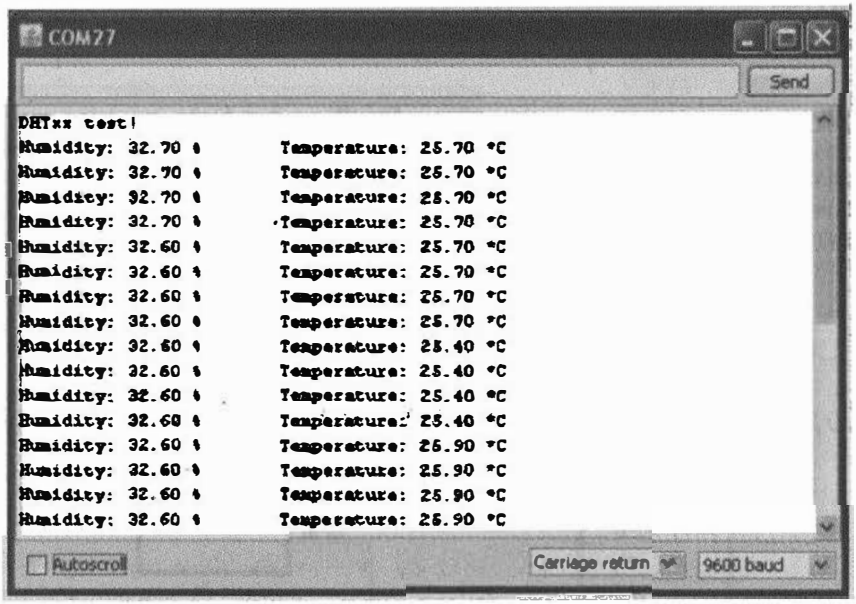


Рис. 8.4. Вывод данных с датчика в монитор последовательного порта

Листинг 8.1

```
#include "DHT.h"
#define DHTPIN 2  // пин подключения
//#define DHTTYPE DHT11 // DHT 11
#define DHTTYPE DHT22 // DHT 22 (AM2302)
//#define DHTTYPE DHT21 // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(9600);
 Serial.println("DHTxx test!");
 dht.begin();
}

void loop() {
 // Reading temperature or humidity takes about 250 milliseconds!
 // Sensor readings may also be up to 2 seconds 'old'
 // (its a very slow sensor)
```

```
 float h = dht.readHumidity();
 float t = dht.readTemperature();

 // check if returns are valid, if they are NaN (not a number)
 // then something went wrong!
 if (isnan(t) || isnan(h)) {
  Serial.println("Failed to read from DHT");
 } else {
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
 }
}
```

8.2. Модуль датчика движения HC-SR501

HC-SR501 (рис. 8.5) — модуль, состоящий из PIR-сенсора и схемы управления.

> ***Пояснение: PIR-sensor***
>
> PIR-sensor переводится с английского как Pyroelectric (Passive) InfraRed sensor — пироэлектрический (пассивный) инфракрасный сенсор. Пироэлектричество — это свойство генерировать определенное электрическое поле при облучении материала инфракрасными (тепловыми) лучами. И поскольку тело человека излучает тепло, PIR-датчики позволяют обнаруживать движение людей в контролируемой зоне. Такие датчики малы по размеру, недороги, имеют низкое энергопотребление, просты в использовании и не изнашиваются.

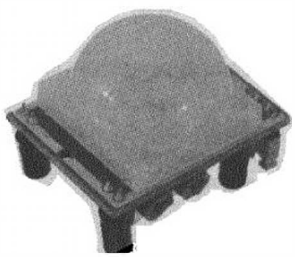


Рис. 8.5. Модуль датчика движения (PIR Motion sensor) HC-SR501

В модуле датчика движения HC-SR501 в качестве органов настройки используются два потенциометра и перемычка (рис. 8.6):

- параметр **Delay Time Adjust**, регулируемый одним из потенциометров, определяет время, в течение которого при обнаружении движения на контакте **OUT** будет удерживаться высокий уровень (5–300 сек.);

- параметр **Distance Adjust** определяет чувствительность датчика — центр контролируемой зоны: от 3 до 7 м;

> ***Совет***
>
> Не стоит располагать PIR-датчики в местах, где температура меняется быстро. Это приведет к тому, что датчик не сможет обнаруживать появление человека в контролируемой зоне, а станет выдавать множество ложных срабатываний.

- режим работы модуля задается перемычкой, обеспечивающей установку одного из двух режимов: режима **H** или режима **L**. На рис. 8.6 в модуле установлен режим **H**. В этом режиме при срабатывании датчика несколько раз подряд на его выходе (на **OUT**) остается высокий логический уровень. В режиме **L** при каждом срабатывании датчика на выходе модуля появляется отдельный импульс.

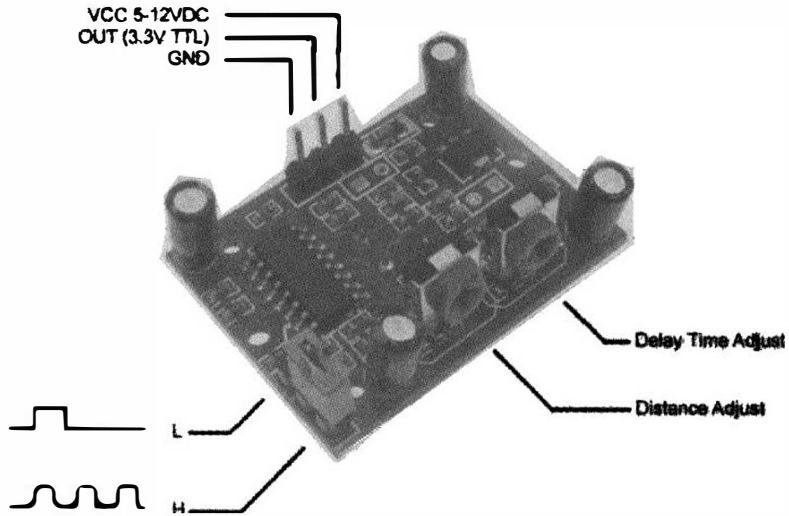


Рис. 8.6. Элементы настройки модуля датчика движения HC-SR501

Основные параметры модуля датчика движения HC-SR501 приведены в табл. 8.1.

***Таблица 8.1.** Основные параметры модуля HC-SR501*

Параметр	Значение
Размеры, см	примерно 3,2×2,4×1,8
Напряжение питания, В	постоянное 4,5–20
Ток на выводе **OUT**, мА	< 60
Напряжение на выходе	Высокие и низкие уровни 3,3 В в TTL-логике
Дистанция обнаружения, м	3–7 (настраивается)
Угол обнаружения	до 120–140° (в зависимости от конкретного датчика и линзы)
Длительность импульса при обнаружении, сек.	5–200 (настраивается)

Таблица 8.1 *(окончание)*

Параметр	Значение
Время блокировки до следующего замера, сек.	2,5 (но можно изменить заменой SMD-резисторов)
Рабочая температура, °C	от –20 до +80
Режим работы	L — одиночный захват, H — повторяемые измерения

Схема подключения модуля датчика движения HC-SR501 к плате Arduino показана на рис. 8.7.

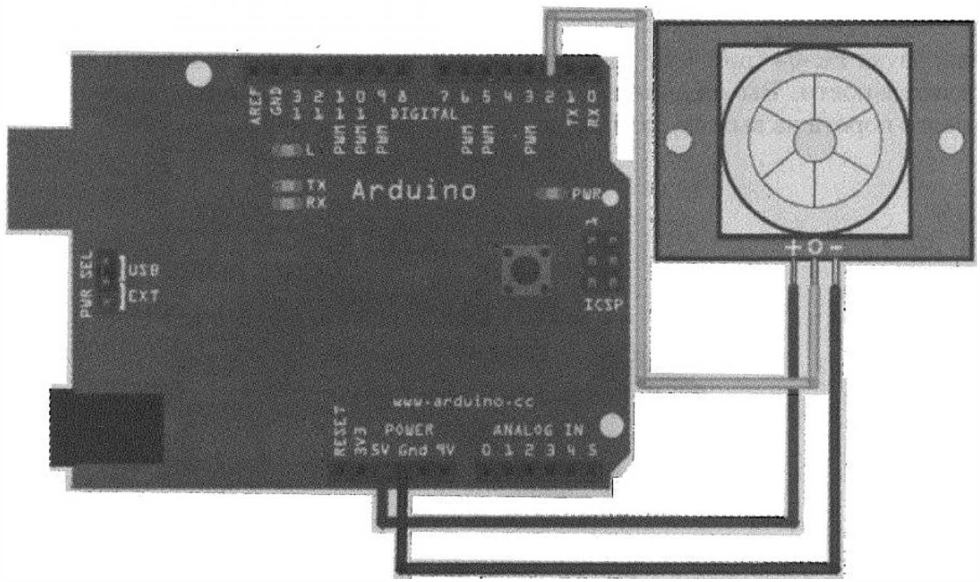


Рис. 8.7. Схема подключения модуля HC-SR501 к плате Arduino

Итак, устанавливаем перемычку в режим работы L и подаем питание. Необходимо подождать примерно 20–40 сек. (для некоторых модулей и до 60 сек.) — в это время датчик калибруется. Теперь, как только датчик зафиксирует движение, на вход Arduino поступит высокий уровень, который будет держаться время, установленное потенциометром **Delay Time Adjust**. В листинге 8.2 приведен скетч для подсчета срабатываний PIR-датчика.

Листинг 8.2

```
int count_motion=0;     // счетчик срабатываний

void setup()
  {
  attachInterrupt(0,motion,RISING);    // запустить прерывания (pin D2)
  }
```

```
void loop ()
    {;}

 // обработка прерывания ИК датчика движения
void motion()
  {
  detachInterrupt(0);
  count_motion++;
  attachInterrupt(0,motion,RISING);
  }
```

8.3. Модуль датчика звука FC-04

Датчик звука FC-04 (рис. 8.8) построен на основе микрофона и имеет регулировку чувствительности. Выход датчика цифровой — при превышении установленного звукового порога на выходе датчика устанавливается логическая единица.

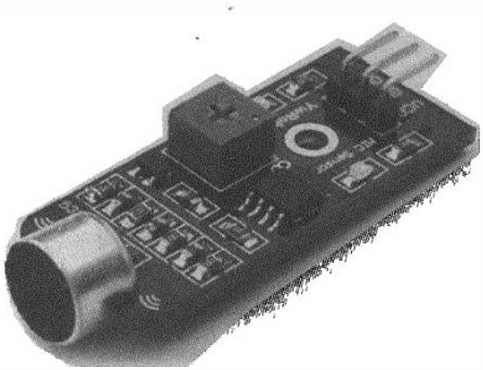


Рис. 8.8. Датчик звука FC-04

Датчик звука FC-04 измеряет интенсивность звука и предназначен для определения факта наличия звука, но он не может определить его уровень и звуковые частоты. Датчик можно применить, например, для голосовых выключателей, в том числе с расшифровкой кодовой последовательности звуков (например, три хлопка в ладоши) определителя лая собаки, если она остается одна в квартире в течение рабочего дня, сигнализации о других посторонних звуках в квартире.

Датчик звука FC-04 представляет собой печатную плату с установленными на ней микрофоном, микросхемой LM393, а также несколькими другими электронными компонентами. Основные характеристики датчика:

- напряжение питания звукового датчика: от +2,7 до +5,5 В;
- ток потребления сенсора звука: 1,4 мА;
- интерфейс или тип выходного сигнала датчика звука: цифровой TTL;
- цифровой выход сигнала: рабочее напряжение на выходе 5 В;
- подключается непосредственно к микроконтроллеру;

- рабочая температура: от –30 до +85 °C;
- размеры модуля звука (длина × ширина × высота): 47×18,4×10 мм;
- вес модуля: 3 грамма;
- диаметр отверстия для монтажа датчика: 3 мм.

Звуковой сенсор снабжен специальным штырьковым разъемом (типа «папа») для подключения к плате микроконтроллера Arduino. Для регулировки чувствительности на плате имеется потенциометр.

В листинге 8.3 представлен скетч для определения срабатывания датчика звука. При срабатывании зажигается светодиод, подключенный к выводу 13 Arduino.

Листинг 8.3

```
const int SensorPin = 9;              // номер pin для датчика
const int ledPin = 13;                // номер led pin
// переменные, изменяемые в процессе работы:
int SensorState = 0;                  // переменная для чтения
                                      // состояния датчика

void setup()
  {
  pinMode(ledPin, OUTPUT);     // начальная установка led pin
                               // как выхода (output)
  pinMode(SensorPin, INPUT);    // установка pin датчика как входа (input)
  }
void loop()
  {
  SensorState = digitalRead(SensorPin);    // Чтение состояния датчика:

  if (SensorState == HIGH) // если на выходе 1
    {
    digitalWrite(ledPin, HIGH);   // включить светодиод
    delay(200);     // Задержка 0.2 сек
    }
  else  // иначе - выключить светодиод
    {
    digitalWrite(ledPin, LOW);
    delay(200);
    }
  }
```

8.4. Ethernet-модуль датчиков на Arduino

Ethernet-модуль датчиков (рис. 8.9) состоит из платы Arduino, платы Ethernet Shield и пяти подключенных к ним датчиков: движения HC-SR501, температуры DS18B20, звука FC-04, освещенности BH1750 и влажности DHT11.

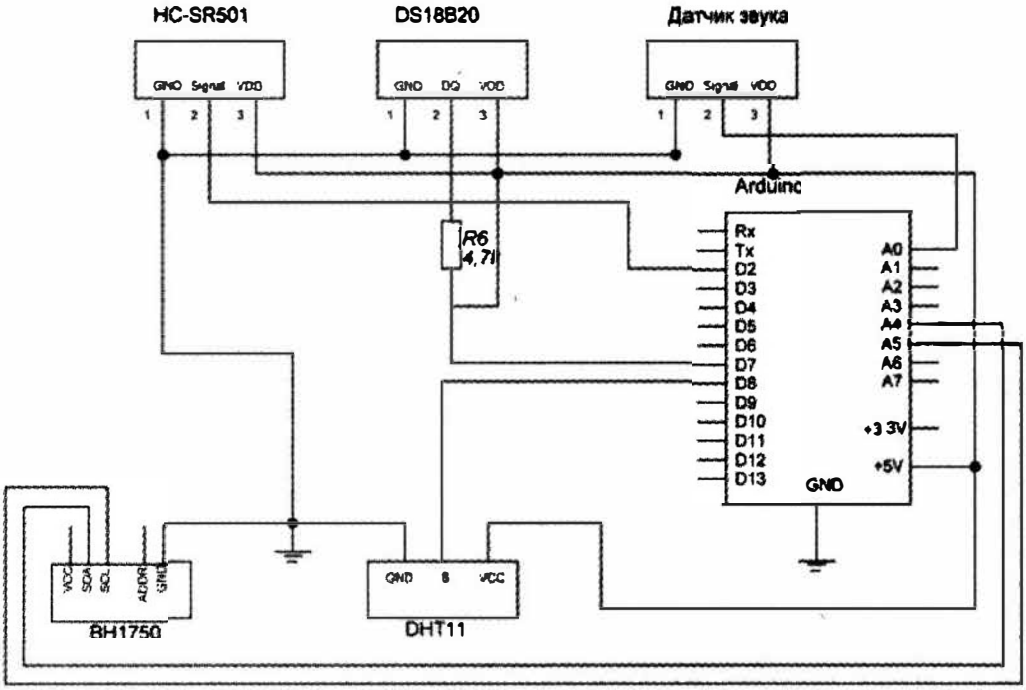


Рис. 8.9. Электрическая схема Ethernet-модуля датчиков

На каждом модуле установлен простейший веб-сервер. Каждый модуль имеет уникальный IP-адрес и свой уникальный порт обращения. При обращении по HTTP к модулю он отдает в формате JSON следующие данные датчиков температуры, влажности, освещенности, количества срабатываний датчика движения и датчика звука после последнего обращения:

```
{«meteo»: {«temp1»:«25.31»,«humidity1»:«35.00»,«lux1»:«10»,«sound1»:«0»,
«motion1»:«0»} }
```

Содержимое скетча для Ethernet-модуля датчиков представлено в листинге 8.4.

Листинг 8. 4

```
// подключение библиотек
#include <SPI.h>
#include <Ethernet.h>
#include <OneWire.h>
#include <Wire.h>
#include <BH1750.h>
// объекты для датчиков BH1750, DHT11, DS18B20
BH1750 light1;
#include "DHT.h"
DHT dht(8, DHTTYPE);
OneWire  ds(7);  // on pin 7
```

```
// датчик (для каждого датчика назначить уникальный ip-адрес и порт)
byte mac[] = { 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xF1 };
byte ip[] = { 192, 168, 1, 121 };
EthernetServer server(10001);
byte my_addr[8]={0x28,0x2A,0x78,0x65,5,0,0,0x10};
// переменные для подсчета срабатываний датчика движения и датчика звука
int count_motion=0;
uint16_t count_sound=0;

void setup() {
  Serial.begin(9600);
  Serial.println("start");
  Ethernet.begin(mac, ip);
  // запуск сервера
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());

  Wire.begin();
  light1.begin();
  // прерывание для PIR-датчика
  attachInterrupt(0,motion,RISING);
}

void loop ()
 {
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        // ожидание конца запроса
        if (c == '\n' && currentLineIsBlank)
          {
          // отправить стандартные header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          // отправка JSON-данных
          client.print('{');
          // данные температуры
          client.print('"');client.print("meteo");client.print('"');
                                                    client.println(":");
          client.print('{');
          int Temp=get_temp();
          client.print('"');client.print("temp1");client.print('"');
                                                      client.print(":");
```

```
          client.print('"');client.print(Temp/16);
          client.print(".");
          client.print(((Temp%16)*100)/16);client.print('"');client.print(',');
          // данные влажности
          float h = dht.readHumidity();
          client.print('"');client.print("humidity1");client.print('"');
                                                          client.print(":");
          client.print('"');client.print(h);client.print('"');
          client.print(',');
          // данные bh1750
          uint16_t lux = light1.readLightLevel();
          client.print('"');client.print("lux1");client.print('"');
                                                          client.print(":");
          client.print('"');client.print(lux);client.print('"');
          client.print(',');
          // sound
          client.print('"');client.print("sound1");client.print('"');
                                                          client.print(":");
          client.print('"');client.print(count_sound);client.print('"');
          client.print(',');
          count_sound=0;
          // датчик движения
          client.print('"');client.print("motion1");client.print('"');
                                                          client.print(":");
          client.print('"');client.print(count_motion);client.print('"');
          client.println("}");
          client.print("}");
          count_motion=0;
          break;
          }
        if (c == '\n')
          {currentLineIsBlank = true;}
        else if (c != '\r')
          {currentLineIsBlank = false;}
        }
      }
    // пауза, чтобы данные ушли
    delay(1);
    // закрыть Ethernet-соединение
    client.stop();
   }
   // sound
   if(analogRead(A0)<500)
    {count_sound++;delay(1000);}
  }
// получение температуры датчика
int get_temp()
```

```
{
byte i;
byte present = 0;
byte data[12];
byte addr[8];
int Temp;

  ds.reset();
  ds.select(my_addr);
  ds.write(0x44,1);
  delay(1000);     // пауза 750 мс
  present = ds.reset();
  ds.select(my_addr);
  ds.write(0xBE);         // команда чтения
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  Temp=(data[1]<<8)+data[0];
  Temp=Temp;
  return Temp;
 }

 // обработка прерывания ИК датчика движения
void motion()
  {
  detachInterrupt(0);
  count_motion++;
  attachInterrupt(0,motion,RISING);
  }
```

Все Ethernet-модули датчиков квартиры/дома соединены в единую локальную сеть. В этой же сети настроен описанный в следующем разделе сервер для сбора данных с устройств и представления данных в виде графиков и таблиц, а также для отображения данных в виде виджетов на плане дома или квартиры.

8.5. Сервер сбора данных

На сервере (Apache, mySQL, php) локальной сети для сбора данных установлен сайт, осуществляющий опрос Ethernet-модулей датчиков по `cron` каждые 5 минут. Данные собираются в базу данных mySQL, где также хранятся настройки всех модулей, задействованных в системе (рис. 8.10).

Сайт написан на HTML5 с элементами canvas, что обеспечивает вывод виджетов на плане объекта. Отправка информации на сервер осуществляется без перезагрузки страницы (использована технология AJAX и библиотека xajax). Вид объектов с виджетами (средние значения температуры, влажности, освещенности) показан на рис. 8.11.

id	id_parent	id_parent1	type	name	description	ip	visible	tek	count_read_errors	coords	plan	icon
1	0	0	1	Дачат	г. Москва ул. Сельская 345		yes	yes	0			house.jpg
20	0	0	1	Квартира	г. Москва ул. Уличная 23 кв. 100		yes	no	0			apartment.jpg
3	13	13	2	Помещение 1	Помещение 1		no	no	0	710,810,710,710,710,120,120,150,165,180		
4	13	13	2	Помещение 2	Помещение 2		no	no	0	410,510,410,410,410,330,390,360,375,390		
5	13	13	2	Помещение 3	Помещение 3		no	no	0	110,100,170,100,170,580,635,640,665,680		
6	14	14	2	Кухня	Кухня		no	no	0	90,80,160,80,160,80,125,140,165,180		
7	14	14	2	Столовая-гостиная	Столовая-гостиная	192.168.1.121:10001	yes	no	0	490,480,550,580,550,130,185,190,215,230		
8	14	14	2	Прихожая	Прихожая		no	no	0	110,110,110,110,110,20,35,50,65,80		
9	14	14	2	Туалет	Туалет		no	no	0	510,510,510,510,510,20,35,50,65,80		
10	14	14	2	Прачечная	Прачечная		no	no	0	110,110,110,110,110,320,335,350,365,380		
11	14	14	2	Холл	Холл	192.168.1.122:10002	yes	no	0	390,390,450,380,450,480,535,540,565,580		
12	14	14	2	Спальня	Спальня		no	no	0	590,580,650,580,650,380,435,440,465,480		
16	14	15	2	Улица	Улица 1 этаж		yes	no	0	790,780,850,880,850,180,235,240,265,280		
17	15	15	2	Спальня 1	Спальня 1 мансарда		no	no	0	450,400,550,480,150,20,75,80,105,120		
15	1	1	1	Мансарда	Мансарда дачи		yes	no	0		plan1mansard.jpg	mansard.jpg
13	1	1	1	Подвал	Подвал дачи		yes	no	0		plan1basement.jpg	basement.jpg
14	1	1	1	1 этаж	1 этаж дачи		yes	yes	0		plan1floor.jpg	1floor.jpg
18	15	15	2	Холл	Холл		no	no	0	510,500,570,500,570,580,635,640,665,680		

Рис. 8.10. Данные модулей в базе mySQL

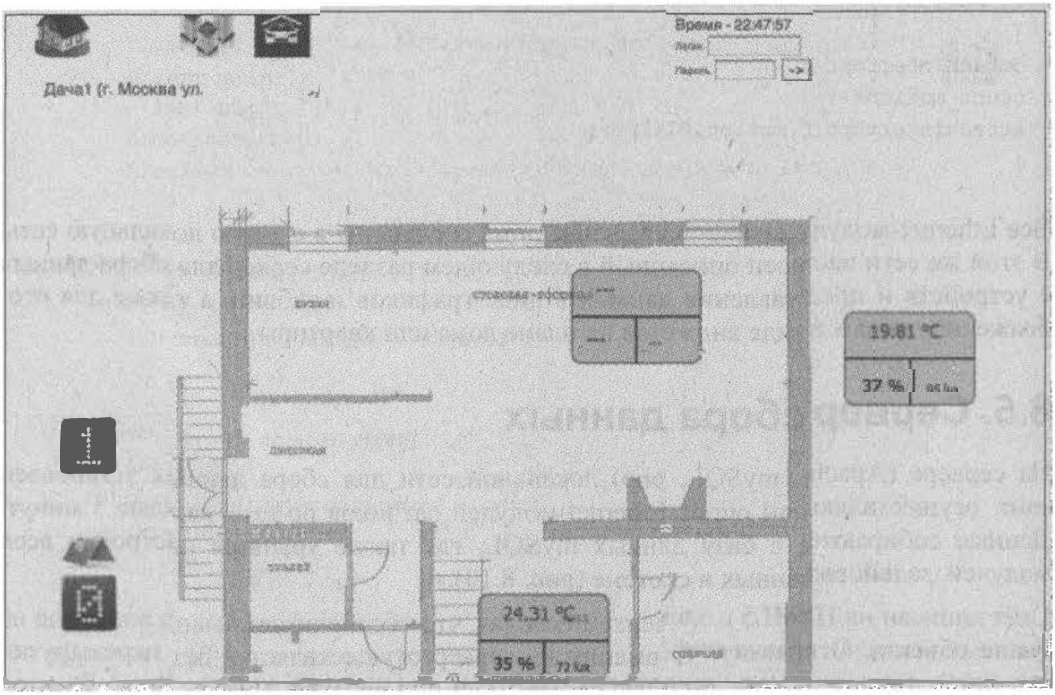


Рис. 8.11. Вид объектов с виджетами на плане дома

После авторизации (рис. 8.12) ссылки с виджетов становятся доступны в режиме администратора, позволяющем осуществлять просмотр данных с возможностью выбора объекта (квартира, дача, гараж), подобъекта (1 этаж ...), комнаты и даты считанных данных (рис. 8.13).

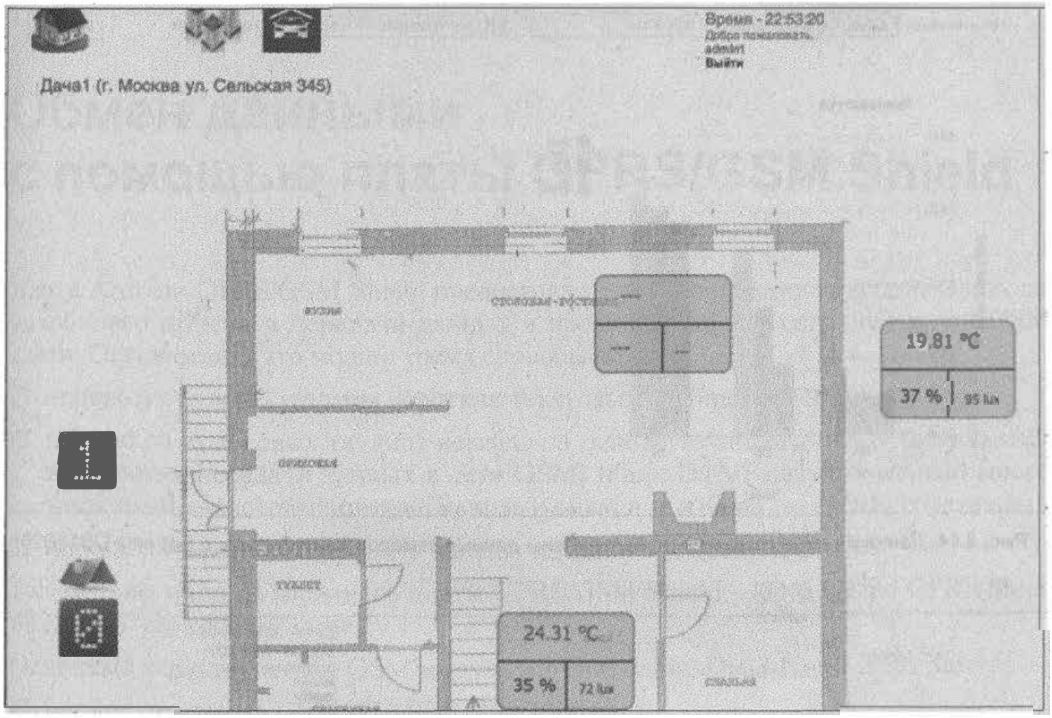


Рис. 8.12. Активация ссылок с виджетов в режиме администратора

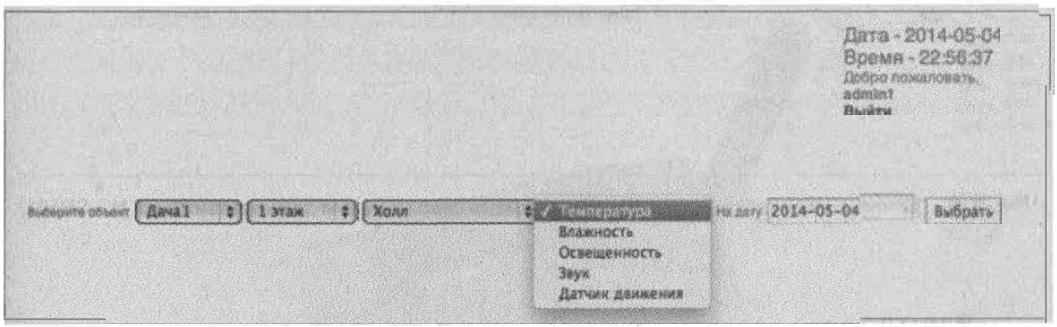


Рис. 8.13. Выбор настроек для просмотра данных, считанных с модулей

Для температуры, влажности, освещенности строятся соответствующие графики (рис. 8.14). Для датчиков звука и датчиков движения — таблицы с количеством срабатываний между запросами (рис. 8.15). При построении графиков и таблиц используется API Google Chart.

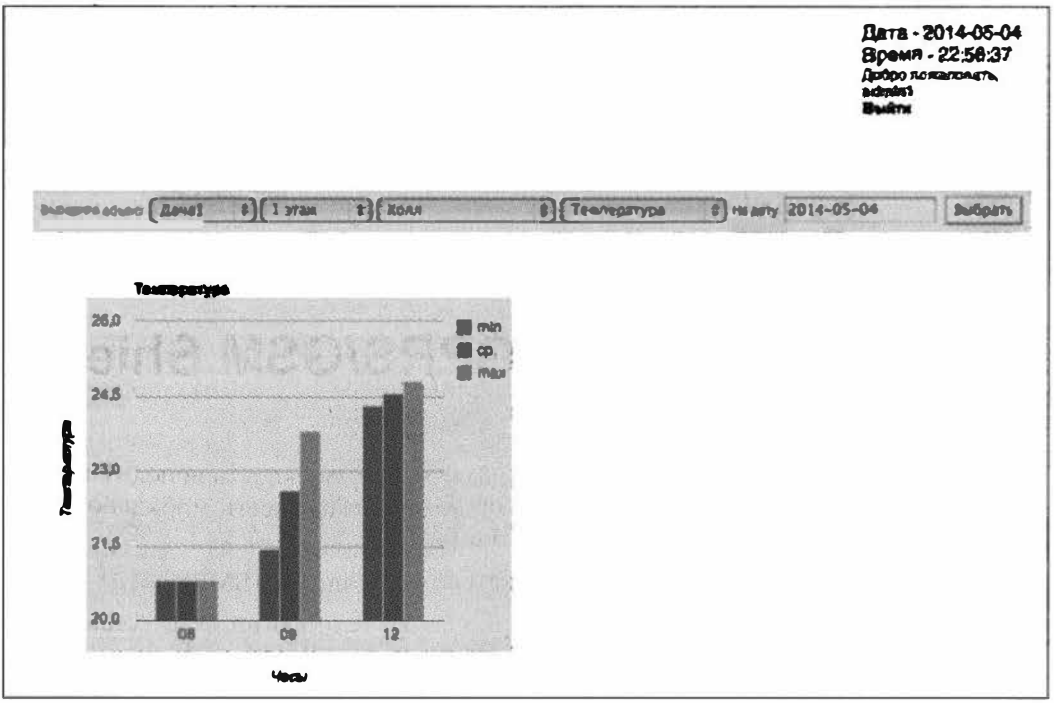


Рис. 8.14. Данные в виде графиков (представлены данные замеров температуры с датчика DS18B20)

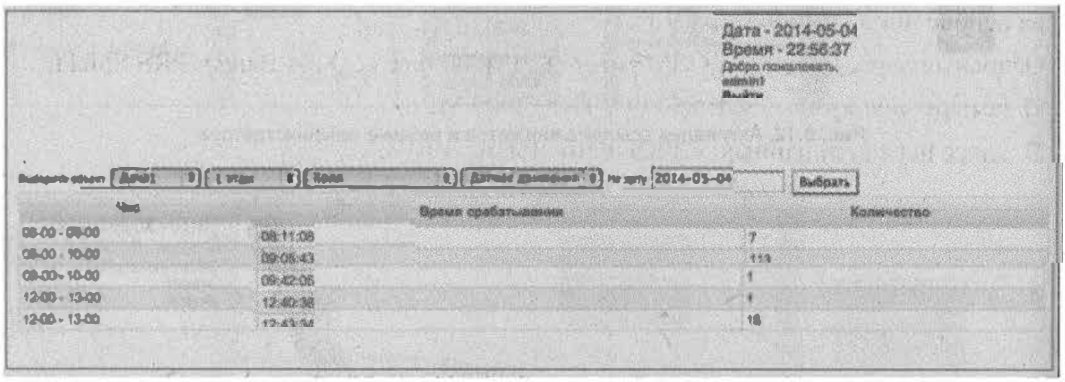
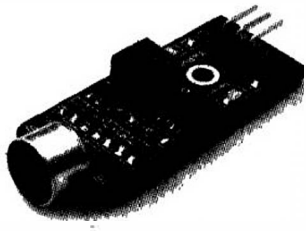


Рис. 8.15. Данные в виде таблиц (представлены данные срабатывания датчика движения HC-SR501)

Электронный архив

Скетч, соответствующий листингу 8.4, можно найти в файле arduino_scetches_08_08_04.ino, а файлы сайта и дамп базы данных сервера — в папке arduino_scetches_08\web сопровождающего книгу электронного архива.

ГЛАВА 9



Обмен данными с помощью платы GPRS/GSM Shield

Плата Arduino GPRS/GSM Shield предоставляет нам возможность использовать для удаленного приема и передачи данных в проектах Arduino сеть мобильной GSM-связи. Осуществить это можно тремя способами:

- используя отправку/прием коротких текстовых сообщений (SMS);
- отправкой голосовых (аудио) команд на основе технологий CSD (стандартная технология передачи данных в сети GSM) и/или DTMF (двухтональный многочастотный аналоговый сигнал, используемый для набора телефонного номера);
- используя пакетную передачу данных на основе технологии GPRS.

Рассмотрим один из вариантов платы GPRS/GSM Shield — Quad-Band GPRS Shield на основе чипа SIM900 (рис. 9.1).

Основные характеристики GSM-модуля SIM900 платы Quad-Band GPRS Shield:

- четыре диапазона GSM: 850/900/1800/1900 МГц;
- класс передачи данных: GPRS multi-slot class 10/8;

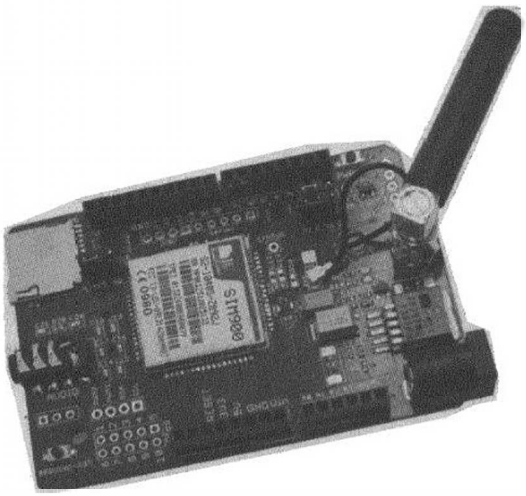


Рис. 9.1. Плата SIM900 Quad-Band GPRS Shield

- соответствие стандарту GSM фазы 2/2+;
- класс мощности 4 (2 Вт в диапазонах 850/900 МГц);
- класс мощности 1 (1 Вт в диапазонах 1800/1900 МГц);
- управление AT-командами (GSM 07.07, 07.05 и фирменные AT-команды SIMCom);
- аудиокодеки HR, FR, EFR, AMR, подавление эха;
- скорость передачи данных по технологии CSD: до 14,4 Кбит/с;
- PPP-стек;
- встроенный стек TCP/IP, UDP/IP;
- протоколы HTTP и FTP;
- протокол защищенных сокетов SSL;
- декодирование DTMF-тонов;
- eMail — формирование и отправка электронных писем посредством AT-команд;
- SMS Autorun — исполнение AT-команд, полученных по SMS от определенного абонента;
- встроенная память для пользовательских данных: 2,5 Мбайт;
- MMS — формирование, дополнение пользовательскими файлами и отправка с помощью AT-команд;
- AMR play — воспроизведение аудиофайлов в динамик или в сторону удаленного абонента;
- функция обнаружения глушения сигнала (Jamming Detection);
- FOTA — обновление прошивки модуля по беспроводному каналу;
- Easy Scan — получение информации об окружающих базовых станциях без подключения SIM-карты;
- PING — проверка доступности адреса в Интернете посредством обмена ICMP-пакетами.

Особенности платы SIM900 Quad-Band GPRS Shield:

- совместимость с Arduino Mega;
- наличие слота для карт SD (включение/отключение с помощью перемычки);
- гнездо наушников «два в одном»;
- программное и аппаратное обеспечение последовательного порта — может общаться с Arduino через последовательный порт программного обеспечения (D2/D3) или последовательный порт (D0/D1);
- интерфейс FTDI;
- наличие слота батарейки для часов реального времени (RTC);
- 10 цифровых входов/выходов GPIO;

- 2 выхода ШИМ;
- интерфейс I^2C.

Плату SIM900 Quad-Band GPRS Shield можно включить двумя способами: аппаратным (кратковременное нажатие кнопки PWRKEY) и программным (используется выход D7 Arduino).

9.1. Отправка и получение SMS-сообщений

В этом примере мы каждые 30 минут будем с помощью платы GSM/GPRS Shield отправлять на определенный телефонный номер показания аналогового датчика температуры LM335, подсоединенного к выводу A0 платы Arduino.

Установим SIM-карту в соответствующий слот платы GSM/GPRS Shield, а саму плату GSM/GPRS Shield состыкуем с Arduino. С помощью джамперов соединим контакты для работы через SoftwareSerial-эмуляцию. Схема соединений представлена на рис. 9.2. Содержимое скетча для отправки SMS приведено в листинге 9.1.

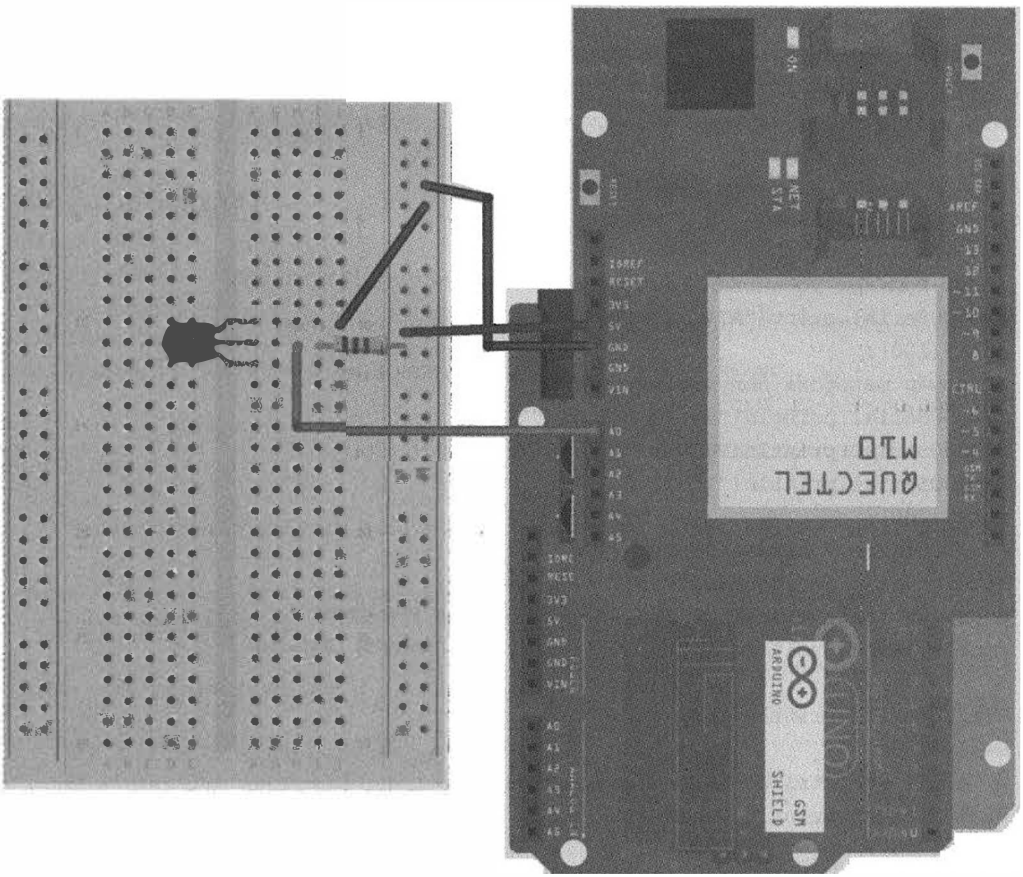


Рис. 9.2. Схема подключения модуля GSM/GPRS Shield и датчика LM335

Листинг 9.1

```
// подключение библиотеки SoftwareSerial
#include <SoftwareSerial.h>
// номер телефона для отправки sms (поменяйте на свой)
#define PHONE "+79034461752"
// Выводы для SoftwareSerial (у вас могут быть 7,8)
SoftwareSerial Sim900Serial(2, 3);
const int lm335=A0;   // для подключения LM335
unsigned long millis1;

void setup()
  {
  Sim900Serial(19200);     // активация последовательного соединения
  }
void loop()
  {
  if (millis()-millis1>30*60*1000)  // прошло 30 минут?
    {
    SendTextMessage();     // отправить sms
    millis1=millis();
    }
  }
// подпрограмма отправки sms
void SendTextMessage()
  {
  // AT-команда установки text mode
  Sim900Serial.print("AT+CMGF=1\r");
  delay(100);
  // номер телефона получателя
  Sim900Serial.println("AT + CMGS = \"");
  Sim900Serial.println(PHONE);
  Sim900Serial.println("\"");
  delay(100);
  // сообщение - данные температуры
  double val = analogRead(lm335);  // чтение
  double voltage = val*5.0/1024;   // перевод в вольты
  double temp = voltage*100 - 273.15;  // в градусы Цельсия

  Sim900Serial.println(temp);
  delay(100);
  // ASCII код ctrl+z - окончание передачи
  Sim900Serial.println((char)26);
  delay(100);
  Sim900Serial.println();
  }
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 9.1, можно найти в файле arduino_scetchès_09_09_01.ino сопровождающего книгу электронного архива.

Проверим работу скетча и, если все в порядке, изменим скетч таким образом, чтобы плата Arduino отправляла SMS-сообщение с данными температуры только при получении приходящего сообщения с текстом temp. Содержимое измененного скетча показано в листинге 9.2.

Листинг 9.2

```
#include <SoftwareSerial.h>
SoftwareSerial Sim900Serial(2, 3);
String currStr = "";    //
String phone = "";      //
// True, если текущая строка является sms-сообщением
boolean isStringMessage = false;

void setup()
  {
  Serial.begin(19200);
  Sim900Serial.begin(19200);
  // Настраиваем прием сообщений с других устройств
  Sim900Serial.print("AT+CMGF=1\r");
  delay(300);
  Sim900Serial.print("AT+IFC=1, 1\r");
  delay(300);
  Sim900Serial.print("AT+CPBS=\"SM\"\r");
  delay(300);
  Sim900Serial.print("AT+CNMI=1,2,2,1,0\r");
  delay(500);
  }

void loop()
  {
  if (!Sim900Serial.available())
    return;
  char currSymb = Sim900Serial.read();
  if ('\r' == currSymb)
    {
    if (isStringMessage)  // текущая строка - sms-сообщение,
      {
      if (!currStr.compareTo("temp"))  // текст sms - temp
        {
        // отправить sms на приходящий номер
        Sim900Serial.print("AT+CMGF=1\r");
        delay(100);
        Sim900Serial.print("AT + CMGS = \"");
```

```
        Sim900Serial.print(phone);
        Sim900Serial.println("\"");
        delay(100);
        double val = analogRead(A0);  // чтение
        double voltage = val*5.0/1024;   // перевод в вольты
        double temp = voltage*100 - 273.15;  // в градусы Цельсия
        Serial.println(temp);
        Sim900Serial.println(temp);
        delay(100);
        Sim900Serial.println((char)26);
        delay(100);
        Sim900Serial.println();
        }
      Serial.println(currStr);
      isStringMessage = false;
      }
    else
      {
      if (currStr.startsWith("+CMT")) {
         Serial.println(currStr);
      // выделить из сообщения номер телефона
      phone=currStr.substring(7,19);
      Serial.println(phone);
      // если текущая строка начинается с "+CMT",
      // то следующая строка является сообщением
      isStringMessage = true;
      }
    }
    currStr = "";
    }
  else if ('\n' != currSymb)
    {
    currStr += String(currSymb);
    }
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 9.2, можно найти в файле arduino_scetches_09_09_02.ino сопровождающего книгу электронного архива.

9.2. Отправка данных на сайт «Народный мониторинг»

С сервисом «Народный мониторинг» мы уже познакомились в *главе 4*, где отправляли ему данные с помощью платы Ethernet Shield. А сейчас мы рассмотрим пример отправки данных на сайт «Народный мониторинг» с использованием платы GPRS/GSM Shield.

Итак, подключаем к Arduino плату GPRS/GSM Shield и датчик температуры LM335. В режиме отправки/получения данных GPRS чип SIM900 потребляет ток до 2 А, поэтому ему необходимо внешнее питание.

Для отправки HTTP-данных по GPRS соединению необходимо выполнить отправку AT-команд в следующей последовательности:

1. Первой отправляется команда `AT` — ответ должен быть `OK`.
2. `AT+SAPBR=1,1` — установка GPRS-связи.
3. `AT+SAPBR=3,1,"CONTYPE","GPRS"` — настройка типа подключения (в данном случае — GPRS).
4. `AT+SAPBR=3,1,"APN","internet.beeline.ru"` — настройка APN (в данном случае — для «Билайн»).
5. `AT+HTTPINIT` — инициализировать HTTP.
6. `AT+HTTPPARA="CID",1` — идентификатор Carrier ID (CID).
7. `AT+HTTPPARA="URL","http://narodmon.ru/post.php"` — собственно URL.
8. Строка данных GET.
9. `AT+HTTPACTION=0` — отправка данных методом GET.
10. Дождаться ответа.
11. `AT+HTTPREAD` — получить ответ.
12. `AT+HTTPTERM` — остановить HTTP.

Теперь пишем скетч отправки данных (листинг 9.3).

Листинг 9.3

```
#define INTERVALSEND 60000
#define LM335 A0

#include <SoftwareSerial.h>
SoftwareSerial GPRS(7, 8);
int onModulePin= 9;
char aux_str[150];
char aux;
char data[512];
int data_size;
uint8_t answer=0;

unsigned long millissend=0;


char apn[]="internet.beeline.ru";
char url[150];
String surl="http://narodmon.ru/post.php/?";
```

```
void setup()
  {
  GPRS.begin(19200);                // скорость для GPRS
  Serial.begin(9600);
  Serial.println("Starting...");
  pinMode(onModulePin,OUTPUT);
  power_on();
  delay(3000);
  // точка доступа APN
  sendATcommand("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"", "OK", 2000);
  snprintf(aux_str, sizeof(aux_str), "AT+SAPBR=3,1,\"APN\",\"%s\"", apn);
  sendATcommand(aux_str, "OK", 2000);
  while (sendATcommand("AT+SAPBR=1,1", "OK", 2000) == 0)
    {
    delay(2000);
    }
  delay(1000);
  }

void loop()
  {
   // отправка раз в 10 минут
   if(millis()-millissend>INTERVALSEND )
     {
      // Initializes HTTP service
     answer = sendATcommand("AT+HTTPINIT", "OK", 10000);
     if (answer == 1)
       {
        // Sets CID parameter
        answer = sendATcommand("AT+HTTPPARA=\"CID\",1", "OK", 5000);
        if (answer == 1)
          {// Sets url
           double val = analogRead(LM335);  // чтение показаний LM335
           double voltage = val*5.0/1024;   // перевод в вольты
           double temp = voltage*100 - 273.15;  // в градусы Цельсия
           String
surl1=surl+"#A0:F3:C1:70:AA:94\n#013950005243291#"+String(temp)+"\n##";

           surl1.toCharArray(url,surl1.length()+1);
           snprintf(aux_str, sizeof(aux_str), "AT+HTTPPARA=\"URL\",\"%s\"", url);
           answer = sendATcommand(aux_str, "OK", 5000);
           if (answer == 1)
           {// Starts GET action
           answer = sendATcommand("AT+HTTPACTION=0", "+HTTPACTION:0,200", 10000);
           if (answer == 1)
             {
             sprintf(aux_str, "AT+HTTPREAD");
```

```
            sendATcommand(aux_str, "OK", 5000);
            }
          else
            {
            Serial.println("Error getting the url");
            }
          }
        else
          {
          Serial.println("Error setting the url");
          }
        }
      else
        {
        Serial.println("Error setting the CID");
        }
      }
    else
      {
      Serial.println("Error initializating");
      }
    sendATcommand("AT+HTTPTERM", "OK", 5000);
    millissend=millis();
    }
  }
// отправка AT-команд
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout)
  {
   uint8_t x=0,  answer=0;
   char response[150];
   unsigned long previous;

   memset(response, '\0', 150);    // Initialize the string
   delay(100);
   while( GPRS.available() > 0) GPRS.read();    // Clean the input buffer
   GPRS.println(ATcommand);    // Отправка AT-команды
   x = 0;
   previous = millis();
   // this loop waits for the answer
   do{
     if(GPRS.available() != 0)
       {
       // if there are data in the UART input buffer, reads it and checks for
                                                                    the asnwer
       response[x] = GPRS.read();
       x++;
```

```
      // check if the desired answer  is in the response of the module
      if (strstr(response, expected_answer) != NULL)
        {
        answer = 1;
        }
      }
    }
    // время ожидания ответа
    while((answer == 0) && ((millis() - previous) < timeout));
    Serial.println(response);
        return answer;
  }
// программное включение питания
void power_on()
  {
  uint8_t answer=0;
  pinMode(onModulePin,OUTPUT);
  // checks if the module is started
  digitalWrite(onModulePin,LOW);
  delay(1000);
  digitalWrite(onModulePin,HIGH);
  delay(2000);
  digitalWrite(onModulePin,LOW);
  delay(3000);
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0)
    {
    digitalWrite(onModulePin,LOW);
    delay(1000);
    digitalWrite(onModulePin,HIGH);
    delay(2000);
    digitalWrite(onModulePin,LOW);
    delay(3000);
    digitalWrite(onModulePin,HIGH);
    delay(3000);
    digitalWrite(onModulePin,LOW);*/
    // время ожидания ответа
    while(answer == 0)
      {
      // Send AT every two seconds and wait for the answer
      answer = sendATcommand("AT", "OK", 2000);
      }
    }
  }
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 9.3, можно найти в файле arduino_scetches_09_09_03.ino сопровождающего книгу электронного архива.

Загружаем скетч в плату Arduino и проверяем отправку данных на сайт. После отправки данных можно добавить новое устройство в список устройств своего профиля (рис. 9.3).

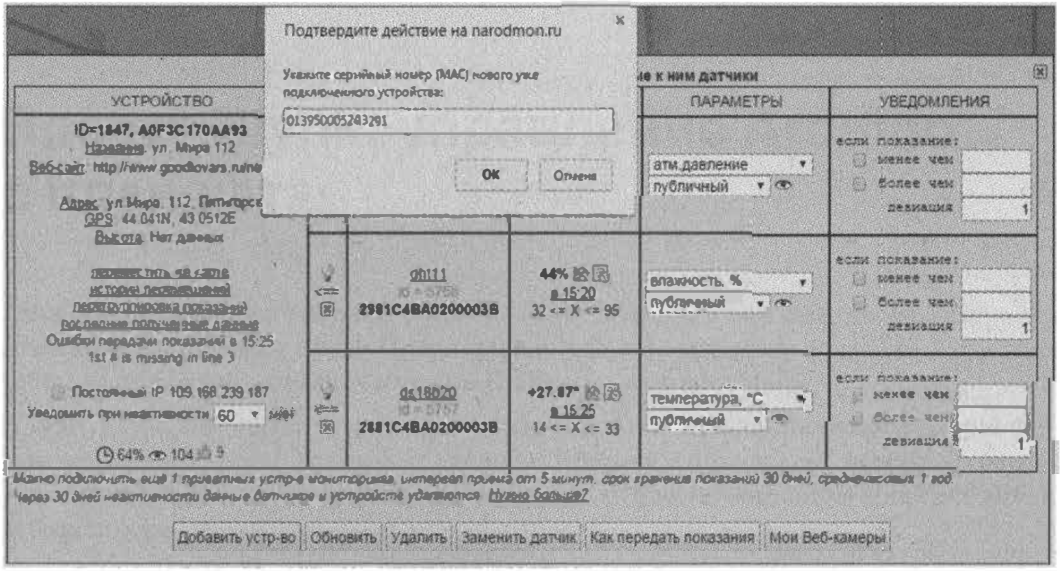
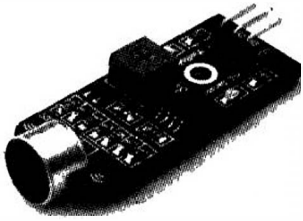


Рис. 9.3. Подключение устройства на сайте «Народный мониторинг»

ГЛАВА 10



Проект Blynk: управление Arduino с планшета

Здесь мы рассмотрим, как управлять Arduino с планшета или смартфона, используя возможности проекта Blynk (**http://www.blynk.cc**), который в начале 2015 года успешно профинансировался на сайте Kickstarter почти на 500%.

Проект Blynk позволяет установить в приложении на планшете/смартфоне различные виджеты (кнопки, слайдеры, дисплеи, графики и пр.) и с их помощью управлять платой Arduino или получать с нее данные. Как можно видеть, проект Blynk работает через Интернет (рис. 10.1).

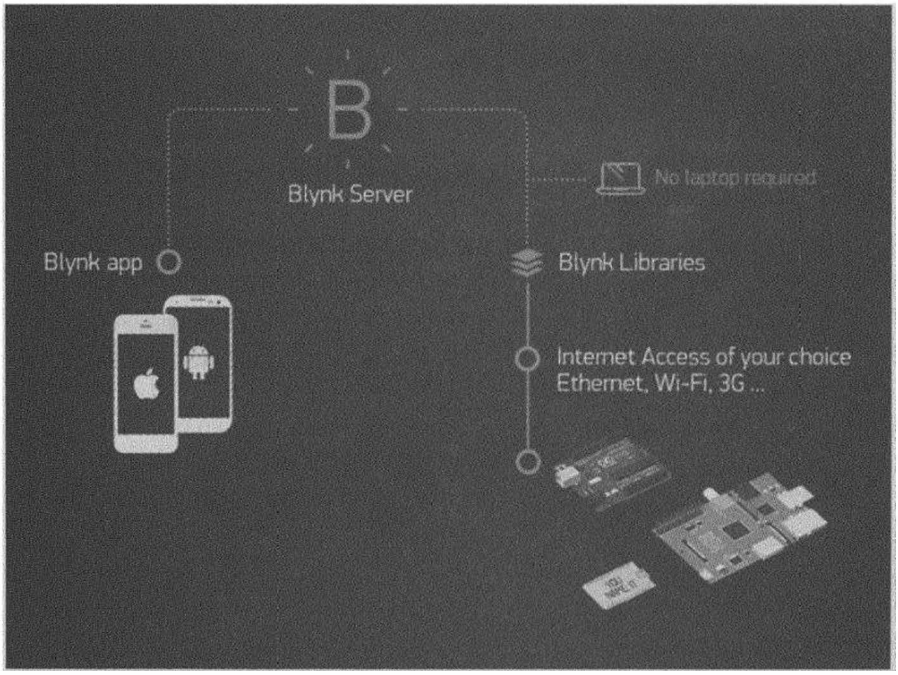


Рис. 10.1. Архитектура проекта Blynk

10.1. Начало работы: тестовый пример

Программное обеспечение Blynk Cloud, написанное на Java, использует простые IP-сокеты TCP и работает на сервере **icloud.blink.cc**. Приложения Blynk для iOS и Android подключаются к Blynk Cloud по умолчанию — каждому пользователю Blynk обеспечивается к нему свободный доступ.

Для начала работы с проектом Blynk необходимо выполнить следующие шаги:

1. Скачать и установить бесплатное приложение Blynk: для Android — из Play Маркет, для iOS — из App Store.
2. Запустить скачанное приложение Blynk и зарегистрироваться в сервисе Blynk.
3. Настроить приложение для своего устройства, добавив на экран необходимые виджеты.
4. Скачать и установить Arduino-библиотеку, загрузить скетч примера на плату Arduino.
5. Запустить приложение на планшете, после чего можно будет управлять выводами Arduino и отображать полученные данные.

Итак, для скачивания приложения для Android заходим в Play Маркет и находим по поиску приложение Blynk (рис. 10.2). Скачиваем его, устанавливаем и создаем себе на сервисе учетную запись (аккаунт), указав адрес своей электронной почты и пароль.

Войдя в аккаунт, для создания нового проекта нажимаем кнопку +, вводим имя проекта и выбираем тип устройства (рис. 10.3). Сервис поддерживает множество



Рис. 10.2. Приложение Blynk в Play Маркет

устройств, включая несколько плат Arduino, Raspberry Pi, ESP8266 и ряд других (список устройств создатели сервиса обещают пополнять). При выборе устройства автоматически генерируется уникальный ключ авторизации (**AUTH TOKEN**), который нам понадобится при написании скетча. В завершение нажимаем на кнопку **Create** — и проект создан.

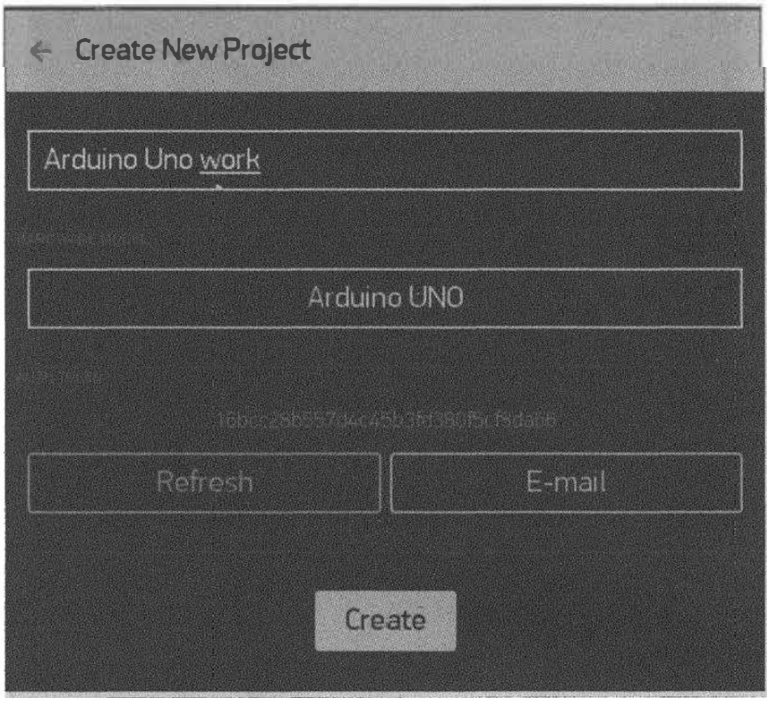


Рис. 10.3. Создание нового проекта в Blynk

Теперь в проект можно добавлять виджеты для управления платой Arduino и получения с нее данных. Для добавления виджетов нажимаем в окне проекта кнопку +. На момент подготовки книги выбор виджетов небольшой — явно меньше заявленного на Kickstarter: камера, джойстики, графики для отображения данных там пока не присутствуют, но что выбрать, все же, есть (рис. 10.4).

Чтобы воспользоваться виджетами, подготовим сначала схему: к плате Arduino с установленной на ней платой Ethernet Shield подключим RGB-светодиод, два обычных светодиода и фоторезистор (рис. 10.5).

После чего согласно этой схеме в проект Blynk на экране планшета мы добавляем виджеты **Button** (рис. 10.6) — задаем переключатель **switch1** состояния вывода D8 Arduino, три слайдера для управления выводами R, G и B RGB-светодиода (рис. 10.7), таймер для включения вывода D12 (рис. 10.8) и виджет **Value Display** для отображения показаний фоторезистора, подключенного к выводу A1 Arduino (рис. 10.9). Общий вид проекта представлен на рис. 10.10.

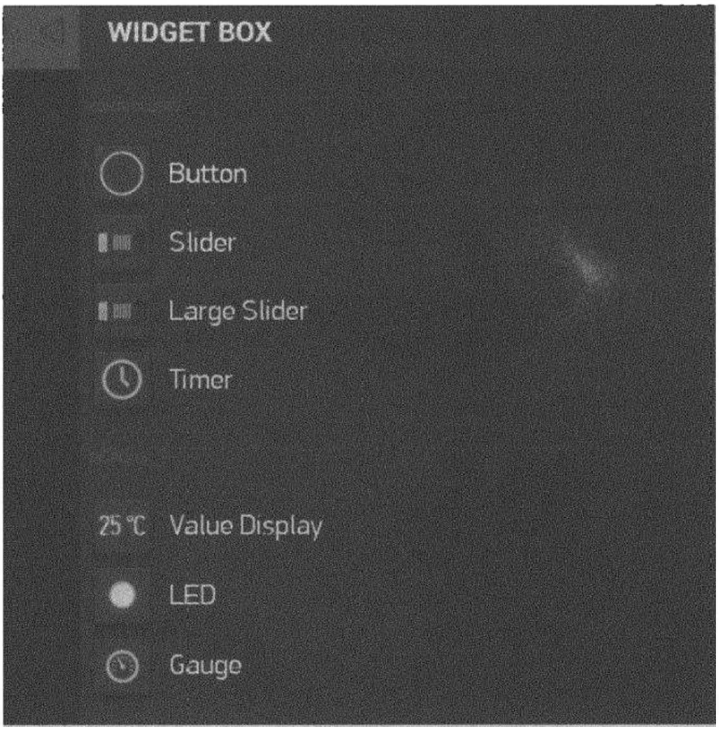


Рис. 10.4. Виджеты для проекта в Blynk

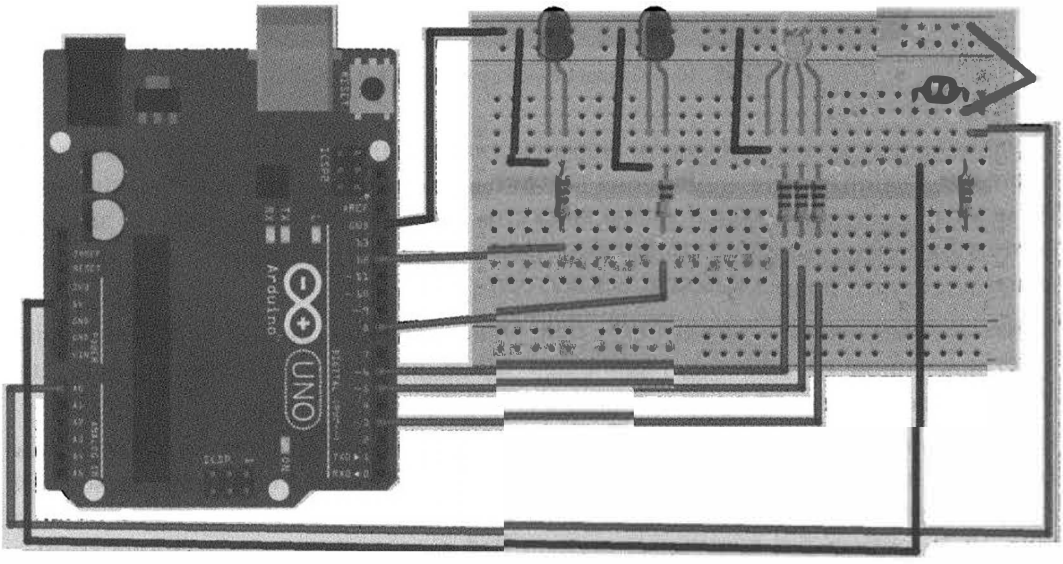
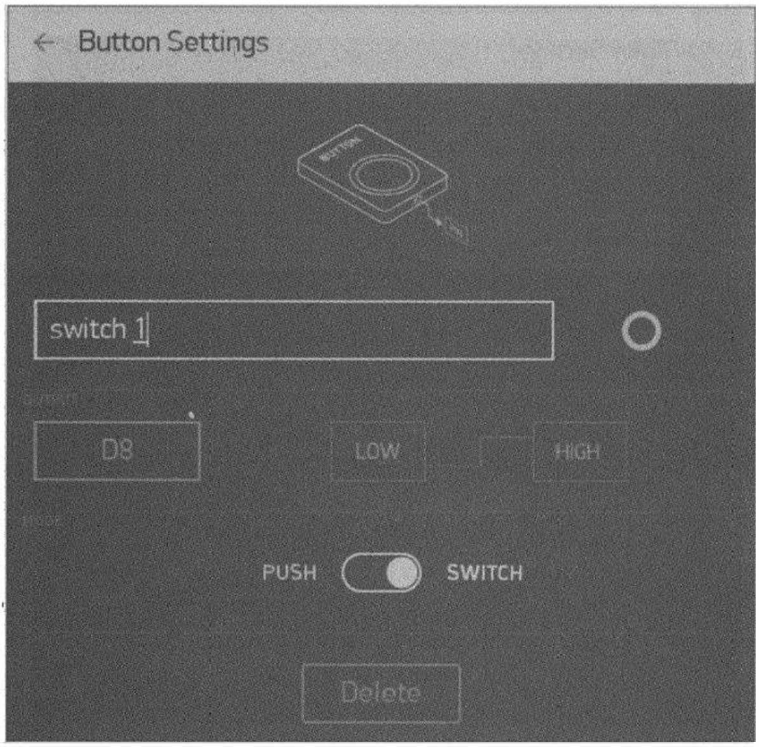


Рис. 10.5. Схема подключения для примера Blynk



, **Рис. 10.6.** Подключение виджета **Button**

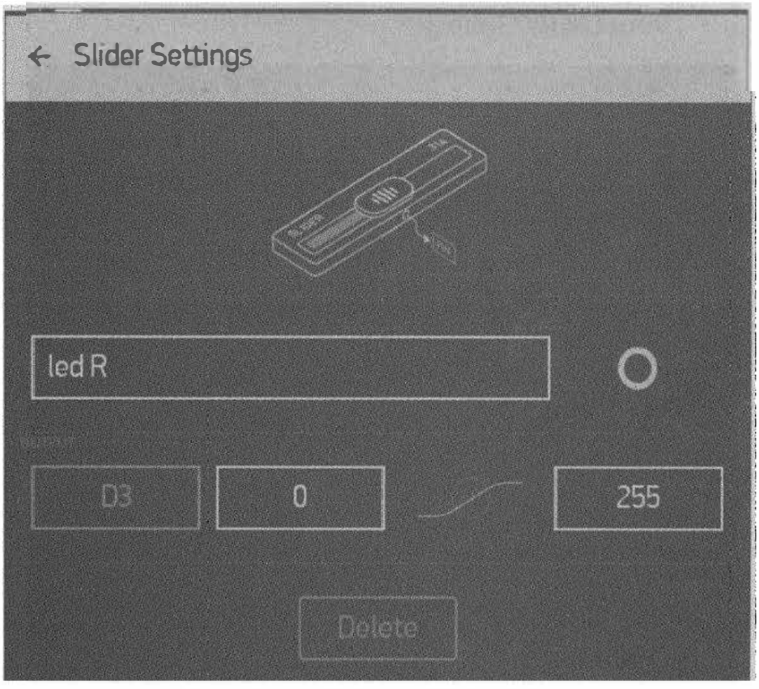


Рис. 10.7. Подключение виджета **Slider**

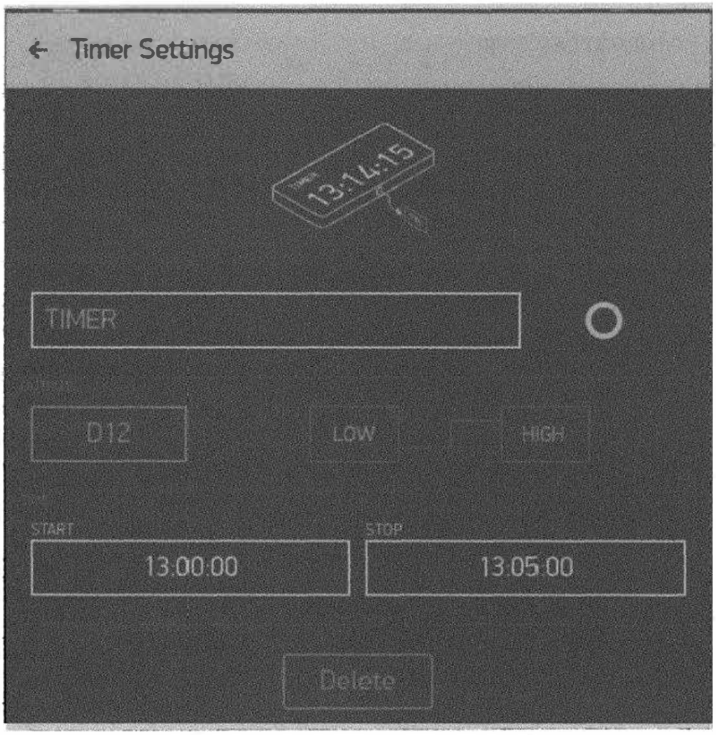


Рис. 10.8. Подключение виджета **Timer**

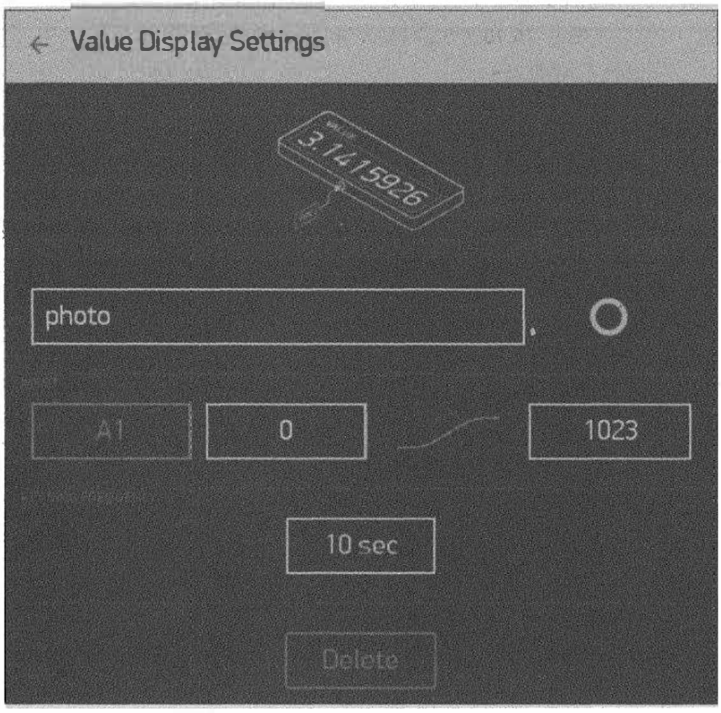


Рис. 10.9. Подключение виджета **Value Display**

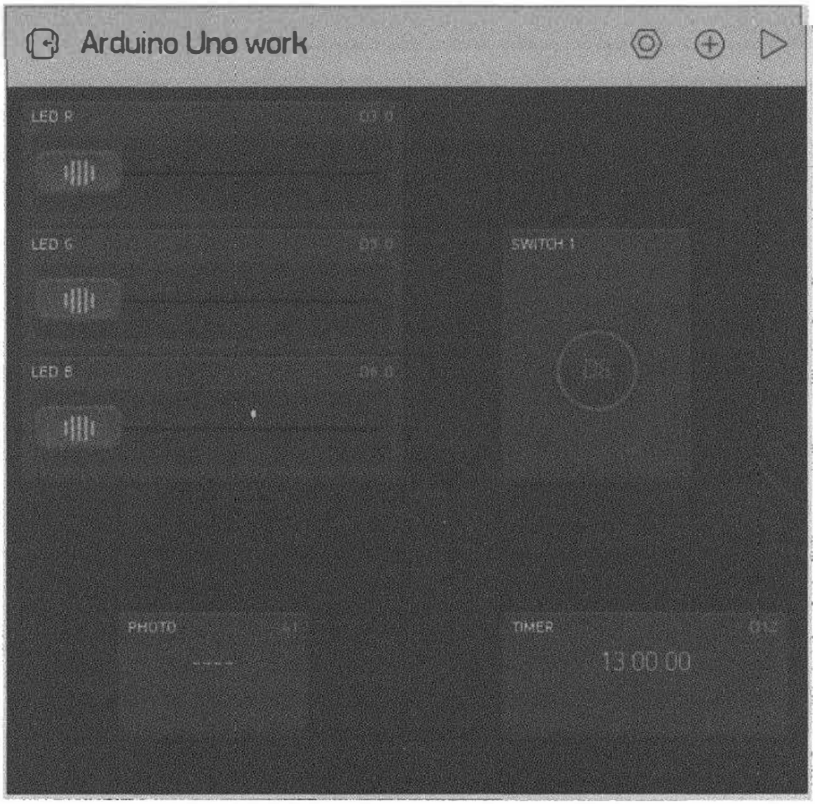


Рис. 10.10. Общий вид нашего проекта Blynk

Вернемся к нашей плате Arduino:

1. Скачиваем библиотеку Blynk по ссылке **https://github.com/blynkkk/blynk-library/archive/v0.2.1.zip.**
2. Распаковываем библиотеку и копируем ее в папку libraries Arduino IDE.
3. Создаем скетч (листинг 10.1) на основе примера Arduino_Ethernet_Manual.ino из библиотеки Blynk:
   - прописываем в нем настройки для нашей платы Arduino (IP, DNS, gateway, subnet, MAC-адрес платы);
   - прописываем ключ авторизации нашего проекта (**AUTH TOKEN**);
   - сервер: `"cloud.blynk.cc"`;
   - порт `8442` (облако **cloud.blynk.cc** использует два порта для обмена данными: 8443 — для мобильных устройств, 8442 — для контроллеров).
4. Загружаем скетч в нашу плату Arduino и открываем монитор последовательного порта.

На рис. 10.11 показан процесс соединения Arduino с сервером **cloud.blynk.cc**.

Листинг 10.1

```
// подключение библиотек
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>

// токен авторизации для проекта на планшете
char auth[] = "16bcc28b557d4c45b3fd380f5cf8da66";
// MAC-адрес и настройки сети для Ethernet shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);
void setup()
  {
  Serial.begin(9600);
  Blynk.begin(auth, "cloud.blynk.cc", 8442, arduino_ip, dns_ip, gateway_ip,
subnet_mask,
        arduino_mac);
  }
void loop()
  {
  Blynk.run();
  }
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 10.1, можно найти в файле arduino_scetches_10_10_01.ino сопровождающего книгу электронного архива.

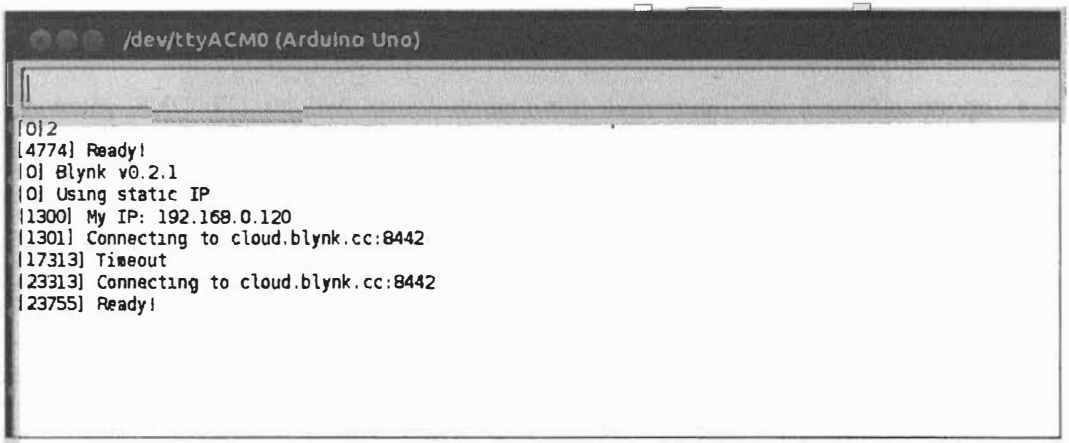


Рис. 10.11. Процесс соединения с сервером cloud.blynk.cc

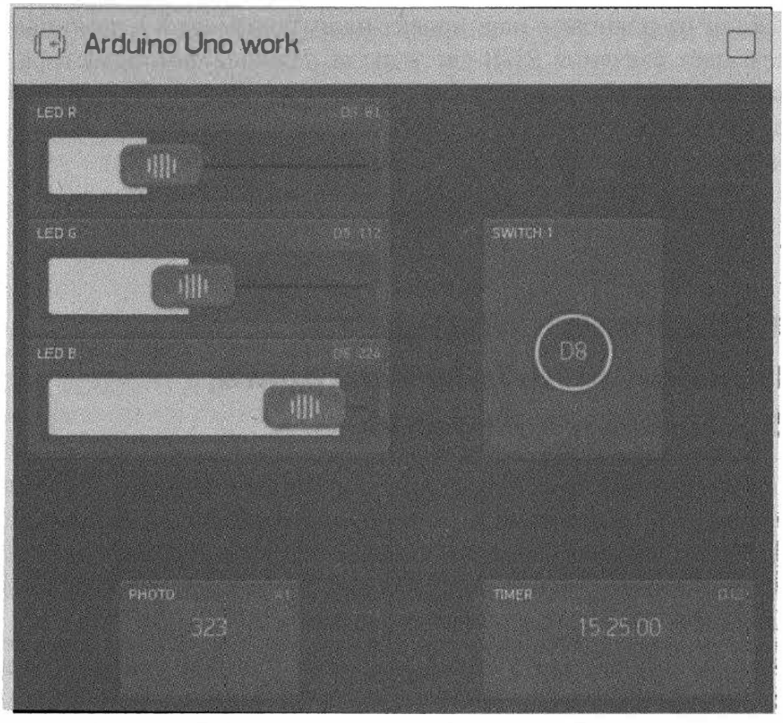


Рис. 10.12. Управление выводами Arduino с экрана планшета

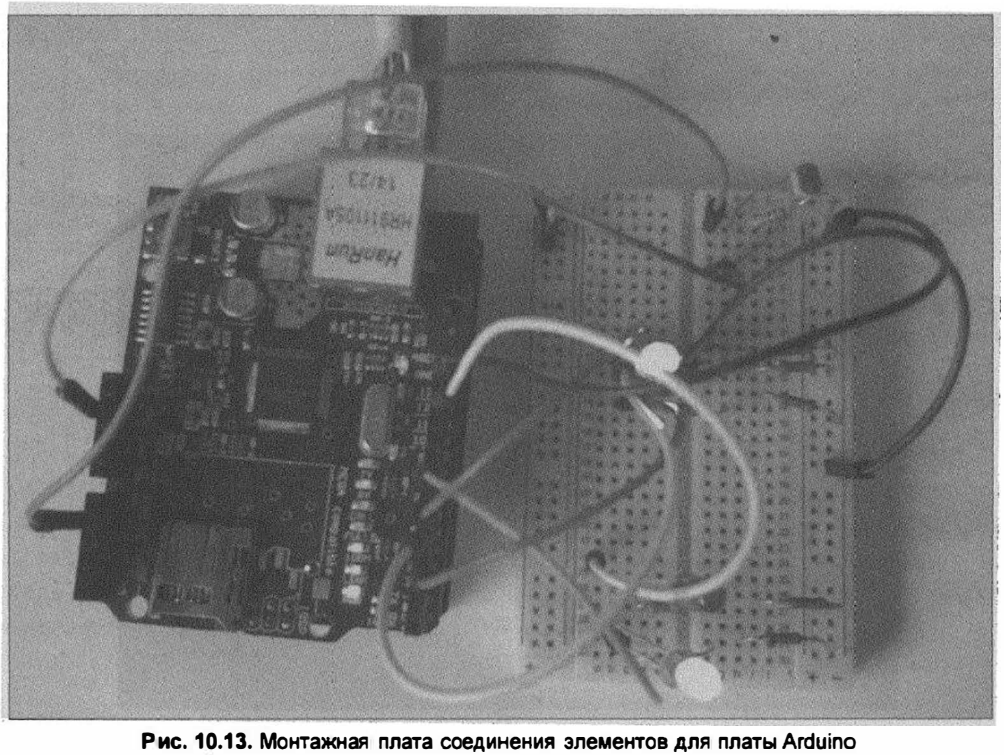


Рис. 10.13. Монтажная плата соединения элементов для платы Arduino

Теперь запускаем на планшете наш проект на выполнение и с помощью слайдеров устанавливаем цвет свечения RGB-светодиода. Данные фоторезистора (вывод **A1**, виджет **Value Display**) обновляются раз в 10 секунд, виджет **Button** (**switch1**) работает как переключатель состояния светодиода, подключенного к выводу **D8**. Timer подает 1 на вывод D12 в 15.00 и 0 — в 15.30 (рис. 10.12).

При этом на плате Arduino наблюдаем изменение состояния светодиодов (рис. 10.13).

10.2. Управление с планшета исполнительными устройствами, подключенными к Arduino

С тестовым примером мы разобрались, надо двигаться дальше. Добавим к нашему проекту возможность управления со смартфона исполнительными устройствами, подключенными к Arduino. Если для управления реле достаточно использования виджета **Button** с опцией **switch**, то управлять сервоприводом подобным образом не получится. Рассмотрим, как это можно сделать.

Чтобы обеспечить угол поворота сервопривода от 0 до 180 градусов, в наш проект на планшете добавляем виджет **Slider**, в качестве виртуального порта (**PIN**) для него выбираем **Virtual V0** и задаем область значений в промежутке **0** до **180** (рис. 10.14).

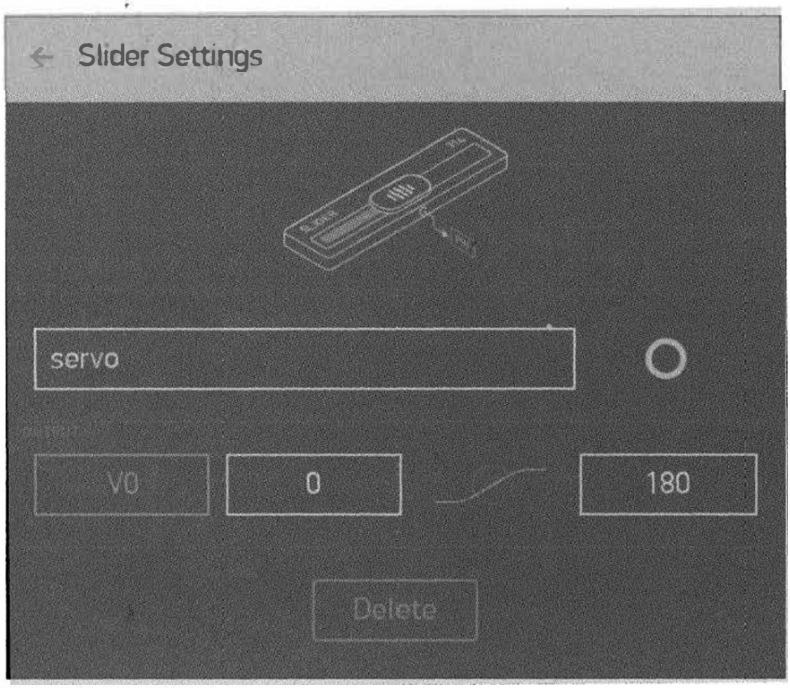


Рис. 10.14. Создание виджета **Slider** для виртуального порта **V0**

Для получения данных из облака с виртуального порта **V0** (при изменении значения на выводе **V0**) необходимо добавить функцию `BLYNK_WRITE(0)`. В качестве параметра в эту функцию как раз и приходит значение виртуального порта **V0**. Выведем его в последовательный порт:

```
BLYNK_WRITE(0)
  {
  int valV0=param.asInt();
  Serial.print("virtual pin0=");Serial.println(valV0);
  }
```

Добавим этот фрагмент в скетч из листинга 10.1 и загрузим в плату Arduino. Теперь, меняя значение на слайдере в проекте на планшете, мы видим вывод этого значения в монитор последовательного порта (рис. 10.15).

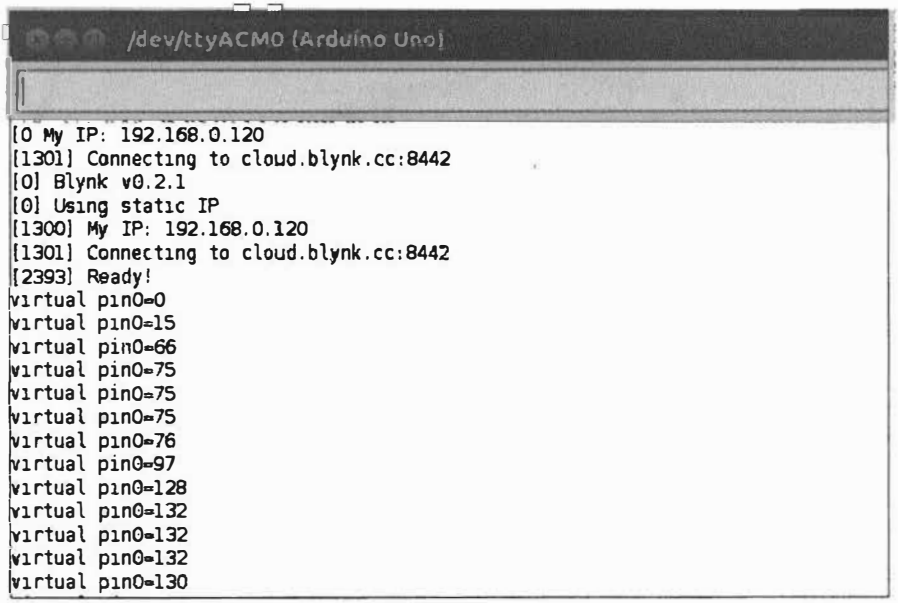


Рис. 10.15. Получение значений виртуального порта V0 на Arduino

Соответственно, получаемое Arduino значение положения слайдера мы можем использовать для задания угла поворота сервопривода — подсоединяем сервопривод в выводу D9 платы Arduino (рис. 10.16) и загружаем в плату скетч из листинга 10.2. Теперь с помощью слайдера на планшете мы можем управлять поворотом подключенного к плате Arduino сервопривода (см. рис. 10.16).

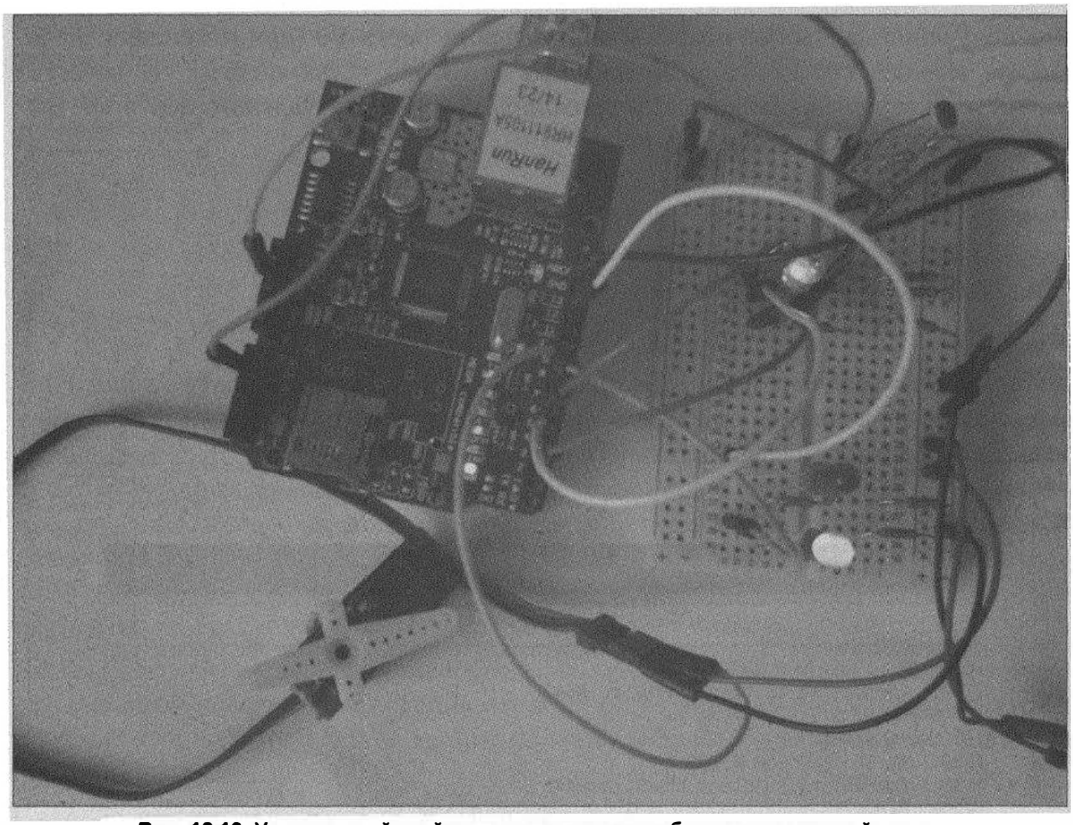


Рис. 10.16. Управляемый слайдером сервопривод добавлен к монтажной плате

Листинг 10.2

```
// подключение библиотек
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>
#include <Servo.h>
// создание экземпляра объекта Servo
Servo servo1;
// токен авторизации для проекта на смартфоне
char auth[] = "16bcc28b557d4c45b3fd380f5cf8da66";
// MAC-адрес и настройки сети для Ethernet shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);
```

```
BLYNK_WRITE(0)
 {
int valV0=param.asInt();
Serial.print("virtual pin0=");Serial.println(valV0);
servo1.write(valV0);    // поворот сервопривода
}

void setup()
  {
  Serial.begin(9600);
  Blynk.begin(auth, "cloud.blynk.cc", 8442, arduino_ip, dns_ip, gateway_ip,
                                                                  subnet_mask,
        arduino_mac);
  servo1.attach(9);    // подключает переменную servo к выходу 9
  }
void loop()
  {
  Blynk.run();
  }
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 10.2, можно найти в файле arduino_scetches_10_10_02.ino сопровождающего книгу электронного архива.

10.3. Отправка данных из Arduino на экран планшета

Дополним наш проект возможностью отображения на экране планшета данных с любых датчиков, подключенных к плате Arduino. Воспользуемся для этого датчиком освещенности BH1750, и станем отправлять с него данные в облако каждые 10 секунд. Чтобы организовать этот процесс, добавим в наш проект на планшете виджет **Value Display**. В качестве виртуального порта (**PIN**) выберем **Virtual V1** (рис. 10.17).

Создадим новый скетч (листинг 10.3) на основе скетча из листинга 10.1, добавив в него процедуру получения Arduino данных с датчика BH1750 и отправку этих данных в проект на планшете через облако **cloud.blynk.cc**. Для задания периодичности опроса датчика и отправки данных в облако мы воспользуемся библиотекой SimpleTimer. Отправку данных в виртуальный порт обеспечивает функция:

```
Blynk.virtualWrite(pin, value);
```

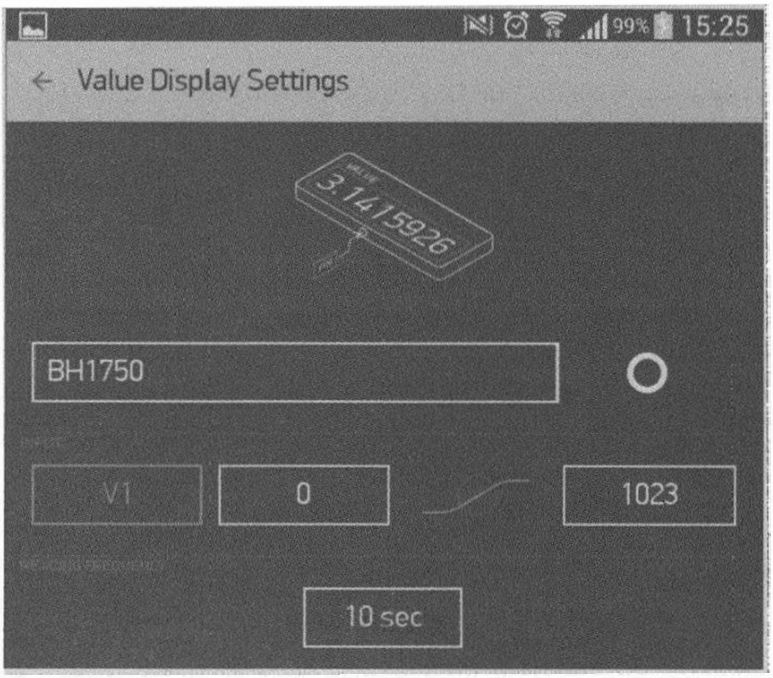


Рис. 10.17. Создание виджета **Value Display** для виртуального порта **V1**

Листинг 10.3

```
// подключение библиотек
#define BLYNK_PRINT Serial

#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>
#include <Servo.h>
#include <SimpleTimer.h>
#include <BH1750.h>
#include <Wire.h>

// создание экземпляра объекта BH1750
BH1750 lightl;
// создание экземпляра объекта Servo
Servo servo1;
// создание экземпляра объекта SimpleTimer
SimpleTimer timer;

// токен авторизации для проекта на смартфоне
char auth[] = "16bcc28b557d4c45b3fd380f5cf8da66";

// MAC-адрес и настройки сети для Ethernet shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
```

```
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);IPAddress subnet_mask(255, 255, 255,
0);
// получение значения с виртуального порта V0
BLYNK_WRITE(0)
  {
  int valV0=param.asInt();
  Serial.print("virtual pin0=");Serial.println(valV0);
  servo1.write(valV0); // поворот сервопривода
  }

void setup()
  {
  Serial.begin(9600);
  Blynk.begin(auth, "cloud.blynk.cc", 8442, arduino_ip, dns_ip, gateway_ip,
subnet_mask,arduino_mac);
  servo1.attach(9); // подключает переменную servo к выходу 9
  light1.begin();   // запуск датчика BH1750
  timer.setInterval(10000, sendBH1750);
  }

void loop()
  {
  Blynk.run();
  timer.run();
  }

void sendBH1750()
  {
  // получить значение освещенности в lx
  uint16_t lux = light1.readLightLevel();
  Serial.println(lux);
  // отправка значения освещенности
  Blynk.virtualWrite(1, lux);
  }
```

Электронный архив

Скетч, соответствующий листингу 10.3, можно найти в файле arduino_scetches_10_10_03.ino, а библиотеку SimpleTimer — в папке arduino_libraries\SimpleTimer сопровождающего книгу электронного архива.

Теперь на планшете в виджете **Value Display** каждые 10 секунд мы видим изменение данных с датчика **BH1750** (рис. 10.18).

Для отображения данных с датчика, подключенного к плате Arduino, необязательно писать код отправки данных по таймеру — можно воспользоваться возможностями библиотеки Blynk (листинг 10.4).

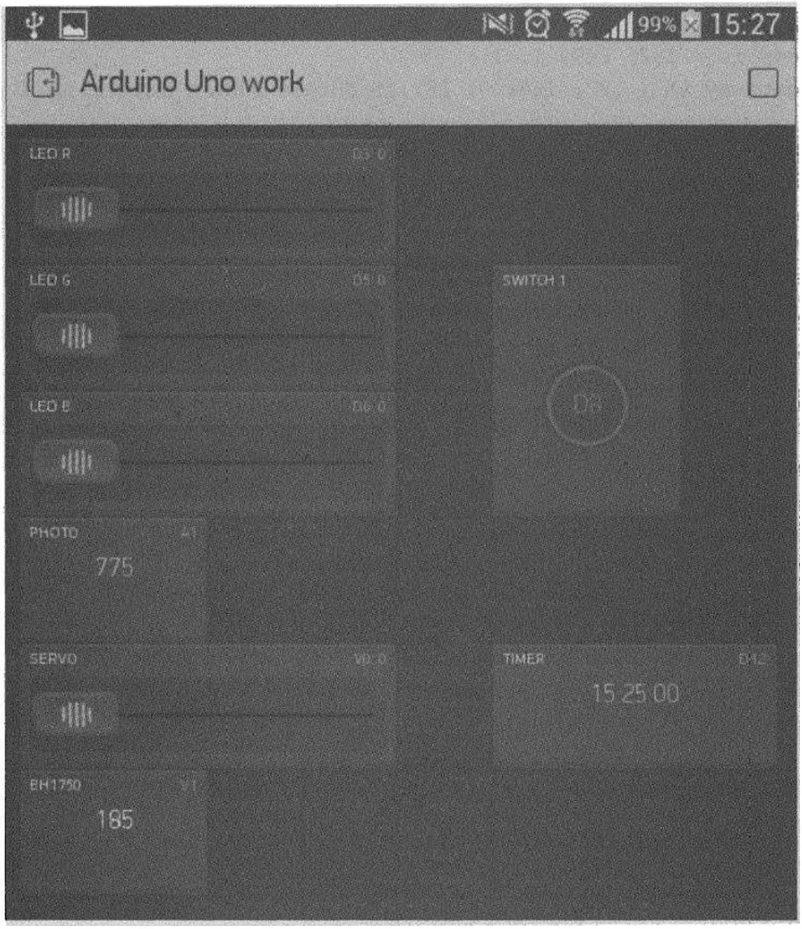


Рис. 10.18. Получение на планшете данных с датчика BH1750, подключенного к Arduino

При создании виджета **Value Display (BH1750)** мы указывали время опроса 10 секунд (см. рис. 10.17). С помощью библиотеки Blynk мы также каждые 10 секунд отправляем данные в виртуальный порт **V1** при получении запроса из виджета:

```
BLYNK_READ(1)
  {
  // получить значение освещенности в lx
  uint16_t lux = light1.readLightLevel();
  Serial.println(lux);
  // отправка значения освещенности
  Blynk.virtualWrite(1, lux);
  }
```

Листинг 10.4

```
// подключение библиотек
#define BLYNK_PRINT Serial

#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>
#include <Servo.h>
#include <BH1750.h>
++++++-#include <Wire.h>

// создание экземпляра объекта BH1750
BH1750 lightl;
// создание экземпляра объекта Servo
Servo servo1;

// токен авторизации для проекта на планшете
char auth[] = "16bcc28b557d4c45b3fd380f5cf8da66";

// MAC-адрес и настройки сети для Ethernet shield
byte arduino_mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
IPAddress arduino_ip ( 192, 168, 0, 120);
IPAddress dns_ip ( 192, 168, 1, 1);
IPAddress gateway_ip ( 192, 168, 0, 28);
IPAddress subnet_mask(255, 255, 255, 0);

BLYNK_WRITE(0)
  {
  int valV0=param.asInt();
  Serial.print("virtual pin0=");Serial.println(valV0);
  servo1.write(valV0); // поворот сервопривода
  }
// функция обработки запроса с выхода V1
BLYNK_READ(1)
  {
  // получить значение освещенности в lx
  uint16_t lux = lightl.readLightLevel();
  Serial.println(lux);
  // отправка значения освещенности
  Blynk.virtualWrite(1, lux);
  }

void setup()
  {
  Serial.begin(9600);
```

```
  Blynk.begin(auth, "cloud.blynk.cc", 8442, arduino_ip, dns_ip, gateway_ip,
                                                    subnet_mask,arduino_mac);
  servo1.attach(9); // подключает переменную servo к выходу 9
  lightl.begin();
  }

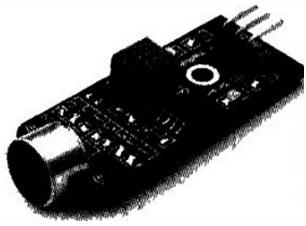
void loop()
  {
  Blynk.run();
  }
```

Электронный архив

Скетч, соответствующий листингу 10.4, можно найти в файле arduino_scetches_10_10_04.ino сопровождающего книгу электронного архива.

Мы рассмотрели здесь базовые возможности сервиса Blynk. Разработчики сервиса обещают добавлять в него новые возможности, поэтому периодически заходим на сайт **http://blynk.cc** и следим за новостями.

ГЛАВА 11



Микрокомпьютер Raspberry Pi

Raspberry Pi — это миниатюрный, размером с кредитную карту, компьютер стоимостью порядка 25 долларов за базовую модель и 35 — за более продвинутую, снискавший огромную популярность и разошедшийся по миру в количестве более 4,5 миллионов штук (на начало 2015 года).

Одним из основных преимуществ Raspberry Pi является соотношение качества продукта и его стоимости. Конечно, этот мини-ПК не столь мощный, как современные компьютеры, но он прекрасно подходит для изучения компьютера и основ работы с ним, для решения многих практических задач, для интернет-серфинга, а также для проигрывания видео- и прослушивания аудиофайлов.

В первую очередь, Raspberry Pi отводится роль компьютера, предназначенного для изучения с его помощью базовых информационных технологий в школе. Позиционируется Raspberry Pi и как дешевое решение для начинающих разработчиков. С учетом удовлетворительной мощности этого устройства, низкого энергопотребления и малой себестоимости его можно использовать для создания личного мини-сервера.

Raspberry Pi — идеальный компьютер для постановки различных экспериментов. А применительно к статистическому управлению процессами (от англ. Statistical Process Control, SPC) — методу мониторинга производственных процессов с целью управления качеством продукции непосредственно в процессе производства, Raspberry Pi является первым доступным техническим решением такого размера, которое можно использовать повсеместно для программирования на многих языках и в качестве микроконтроллеров для управления роботизированными устройствами.

Одна из главных и привлекательных особенностей Raspberry Pi — наличие на плате аппаратных портов ввода/вывода GPIO (General Purpose Input/Output, интерфейс ввода/вывода общего назначения), что открывает перспективы использования его в робототехнических проектах и устройствах «умного дома».

11.1. Технические характеристики и возможности Raspberry Pi

Первые версии Raspberry Pi: модели «A», «B» (рис. 11.1), «A+» и «B+» — оснащены процессором Broadcom BCM2835 архитектуры ARM11 с тактовой частотой 700 МГц и модулем оперативной памяти на 256 (или 512) Мбайт, размещенным по технологии package-on-package непосредственно на процессоре. Модель «A» оснащается одним портом USB 2.0, модель «B» — двумя такими портами, модель «B+» — четырьмя. У моделей «B» и «B+» также наличествует порт Ethernet. Помимо основного ядра, процессор BCM2835 включает графическое ядро с поддержкой OpenGL ES 2.0, аппаратного ускорения и видео Full HD, а также ядро DSP (Digital Signal Processor, цифровой сигнальный процессор).

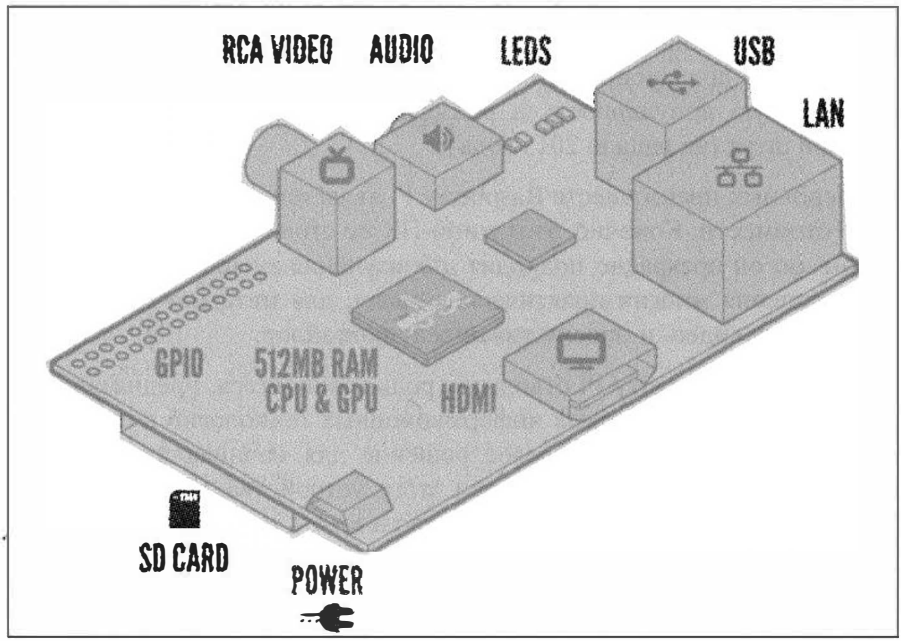


Рис. 11.1. Схема Raspberry Pi, модель «B»

Питание компьютера осуществляется через разъем MicroUSB, при этом сила тока должна составлять минимум 0,5–0,7 А. При меньших значениях компьютер все еще может включиться, но будет уходить в перезагрузку при запуске ресурсоемких задач. Следовательно, подключать плату лучше не через хаб, а напрямую к USB-порту компьютера или в розетку через специальный переходник.

Никаких кнопок включения/выключения на плате Raspberry Pi нет. Если необходимо устройство запустить — подключаете USB-питание, для выключения — выдергиваете шнур. Остается надеяться, что в будущих ревизиях Raspberry Pi, возможно, добавят питание по Ethernet, поскольку это один из самых частых запросов от пользователей.

Версии «A» и «B» устройства оснащены слотом для карт памяти формата SD, версии «A+» и «B+» — слотом для карт формата MicroSD. Без карты памяти Raspberry Pi не включается, поскольку именно на ней должна быть записана операционная система, — это все равно что пробовать запустить компьютер без жесткого диска. Поскольку собственной ОС во внутренней памяти (как, например, в телефонах) у Raspberry Pi не имеется, то из этого следует один положительный момент — устройство практически невозможно превратить в «кирпич»: после любого неудачного эксперимента достаточно перезаписать дистрибутив на карте памяти, и Raspberry Pi снова заработает как новенький.

Для подключения дисплея на плате Raspberry Pi имеются сразу два интерфейса: RCA Video (композитный) и HDMI. Применяя соответствующие переходники, можно выйти и на более традиционные: VGA и DVI. HDMI поддерживает передачу как видео, так и звука, но если потребуется отдельный аудиоканал, то и он присутствует на плате в виде стандартного мини-джека 3,5 мм. Подключение микрофона также возможно, но для этого понадобится найти USB-устройство, совместимое с Raspberry Pi.

Существующие сейчас модели Raspberry Pi не имеют модуля Wi-Fi, и для работы с ними в Интернете понадобится задействовать порт Ethernet. Поскольку физически на плате он скоммутирован через USB 2.0, то обеспечивает не гигабитную, а всего лишь 100-мегабитную скорость.

Чипы процессора и графического ускорителя не оснащены даже простейшими радиаторами, и после нескольких часов работы компьютера становится очевидным, почему его разработчики остановились именно на этом решении: плата нагревается при работе совсем незначительно — она скорее теплая, чем горячая.

Частота процессора, как уже упоминалось, составляет 700 МГц, и, в зависимости от дистрибьютора, его можно разогнать до 1000 МГц без потери гарантии (возможен выбор и более щадящих режимов). Чип памяти производства Samsung или Hynix напаян прямо поверх основного чипсета, так что увеличить RAM самостоятельно не получится. При покупке стоит обратить внимание на маркировку этой SoC (от англ. System-on-a-Chip, системы на кристалле): номер партии для «старых» версий модели «B» с 256 Мбайт RAM начинается с K4P2G, а у выпуска с 512 Мбайт памяти — с K4P4G.

Видеоускоритель Broadcom VideoCore IV позволяет даже при таком слабом процессоре декодировать видео 1080p h.264 с битрейтом вплоть до 40 Мбит/с. Для аппаратного ускорения MPEG-2 и VC-1 лицензии придется докупать отдельно.

Плата оснащена индикаторами — пятью светодиодами: три из них демонстрируют активность и режим работы Ethernet, а еще два сигнализируют о наличии питания и работе с SD-картой.

На рынке можно найти несколько корпусов как официальных производителей устройства, так и сторонних, которые применяются для повышения защищенности компьютера и более удобной его транспортировки.

А теперь — самое интересное: уточним набор низкоуровневых интерфейсов, которые позволяют подключать к Raspberry Pi платы расширения, внешние контролле-

ры, датчики и прочие аксессуары. Во-первых, на плате имеются 15-штырьковые слоты CSI-2 — для подключения камеры и DSI — для подключения дисплея. Во-вторых, присутствует колодка на 26 (40 — для версии «B+») линий ввода/вывода общего назначения GPIO. На них же реализованы интерфейсы UART, консольный порт, шина SPI (Serial Peripheral Interface, последовательный периферийный интерфейс), I²C (Inter-Integrated Circuit, последовательная шина данных для связи интегральных схем) и I²S (Integrated Inter-chip Sound, последовательная шина данных, служащая для соединения цифровых аудиоустройств). Использование GPIO — это как раз самое интересное и творческое применение Raspberry Pi.

Впрочем, недостатков у Raspberry Pi тоже хватает. В нем, к примеру, нет собственных часов реального времени (Real Time Clock, RTC), поэтому единственный способ получения времени — это синхронизация с NTP-серверами. SoC, как уже упоминалось, содержит в себе цифровой сигнальный процессор (DSP), но полного доступа к его API до сих пор нет. Выводы GPIO никак не защищены от короткого замыкания, поэтому ошибка в монтаже может сгубить весь мини-ПК. Кроме того, Raspberry Pi способен обрабатывать только цифровые сигналы. Видеовыходы не могут одновременно выводить картинку. Аудиовхода вообще нет...

В 2015 году выпущена новая версия — Raspberry Pi 2 Model B (рис. 11.2). Вместо процессора BCM2835 в ней установлен чип BCM2836. Отличается он от предшественника наличием четырех ядер ARM Cortex-A7 с набором инструкций ARMv7 — против ARMv6k у BCM2835. Тактовая частота этого процессора составляет 900 МГц — не слишком много по меркам современных решений Qualcomm и даже MediaTek, но для подавляющего большинства DIY-проектов (от англ. Do It Yourself, «Сделай сам») ее хватит с лихвой, тем более что, напомним, ядер на этот раз целых четыре. Объем оперативной памяти нового устройства увеличен с 512 Мбайт до 1 Гбайт. Видеоядро изменений не претерпело, и это по-прежнему Broadcom

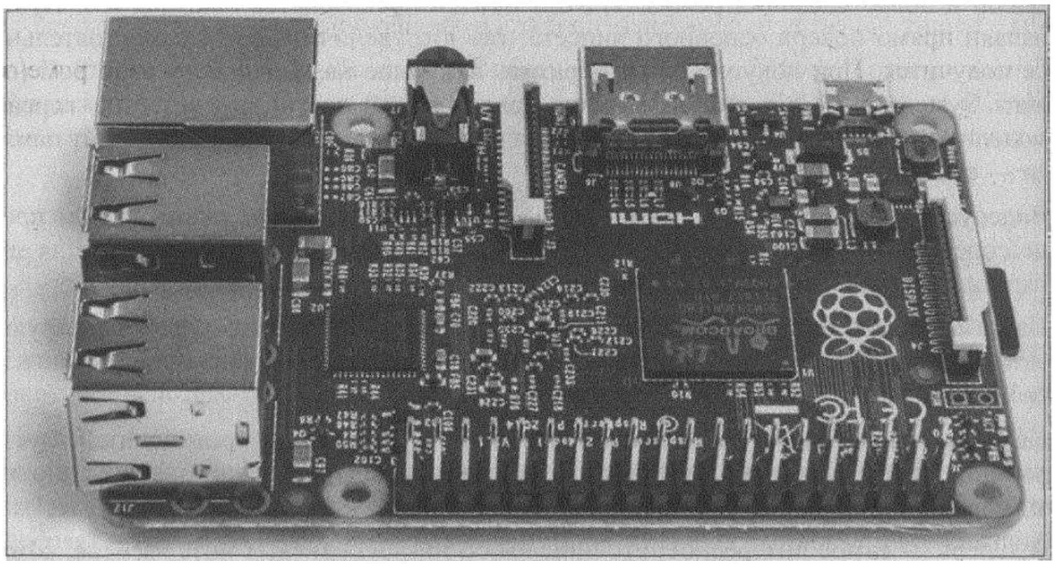


Рис. 11.2. Микрокомпьютер Raspberry Pi 2

VideoCore IV. Разработчики говорят о шестикратном приросте производительности в многопоточных тестах и о трехкратном — в однопоточных. Новая мощь может пригодиться во встроенных системах при обработке изображений — есть много энтузиастов, которым хочется создать компьютерное зрение на основе OpenCV. Примечательно, что на новую модель сохранилась демократичная цена в 35 долларов США.

Raspberry Pi может стать в ваших руках и медиацентром, и управляющим центром «умного дома», и мозгом робота — тут уж все зависит от вашей фантазии и желания. В Сети есть немало примеров, готовых проектов, сообществ пользователей и целых магазинов, посвященных Raspberry Pi. Есть даже официальный очень-очень скромный интернет-магазин The Pi Store (**http://store.raspberrypi.com/projects**) с небольшим количеством ПО, игр, руководств и собственным журналом.

В этой книге мы рассмотрим использование микрокомпьютеров Raspberry Pi в проектах IoT (Интернет вещей).

11.2. Установка операционной системы

Для того чтобы запустить Raspberry Pi, необходимо установить на него операционную систему. В качестве таковой нам доступны три официальных дистрибутива Linux:

- Pidora — основанный на Fedora;
- Archlinux — установка этого дистрибутива происходит практически вручную;
- Raspbian — основанный на Debian.

Кроме этих трех операционных систем, на Raspberry Pi портировано очень много других. Поскольку операционная система устанавливается на SD-карту, чтобы запустить другую систему, достаточно вставить в устройство карту с этой системой. Мы в наших проектах будем ориентироваться на операционную систему Raspbian.

Самый простой способ установить дистрибутив на Raspberry Pi — инструмент NOOBS от создателей Raspberry Pi, который позволяет при первой загрузке выбрать для установки одну операционную систему из следующих:

- ArchLinux;
- OpenElec;
- Pidora;
- RaspBMC;
- Raspbian;
- Risc Os.

Для установки дистрибутива на Raspberry Pi с помощью NOOBS потребуется карта формата SD (для моделей «A» и «B») или MicroSD емкостью 4 Гбайт или более. SD-карту необходимо отформатировать, в чем нам поможет программа SD Formatter 4.0, архив которой доступен для закачки по ссылке **https://www.sdcard.org/downloads/formatter_4/eula_windows/** (рис. 11.3).

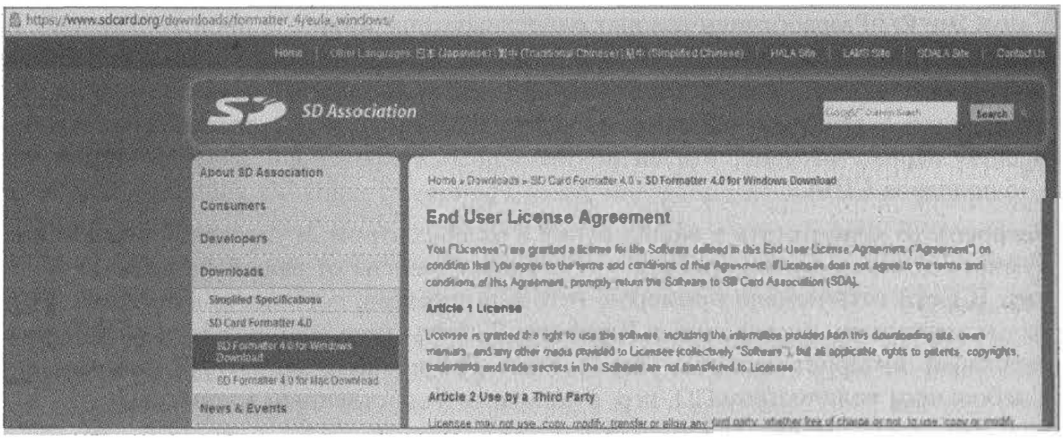


Рис. 11.3. Страница загрузки программы SD Formatter 4.0

Итак, помещаем нашу карту в картридер компьютера, скачиваем программу, устанавливаем ее на компьютер и запускаем. Выбираем нашу SD-карту, нажав кнопку **Options**, задаем параметр **FORMAT SIZE ADJUSTMENT** равным **ON** и форматируем (рис. 11.4).

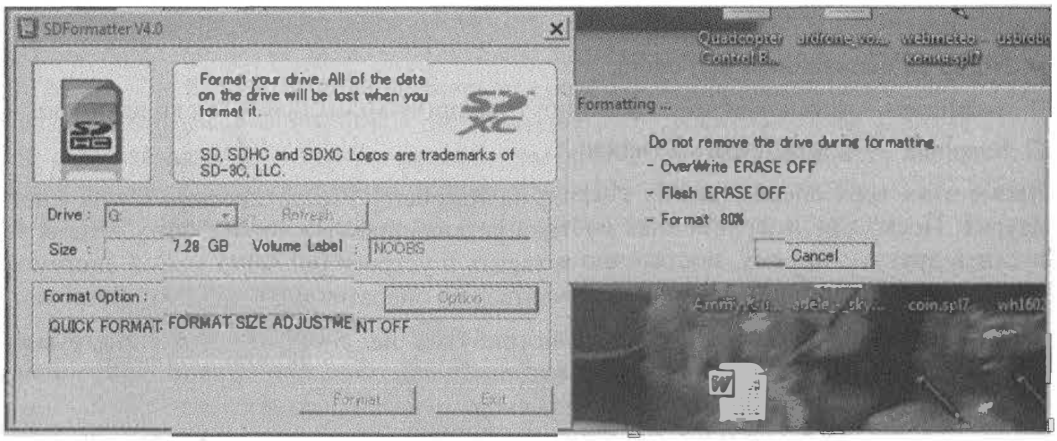


Рис. 11.4. Окно программы SD Formatter 4.0

Далее скачиваем программное обеспечение NOOBS (**http://downloads.raspberrypi.org/noobs**) и распаковываем ZIP-файл на нашу SD-карту.

По завершении записи вынимаем SD-карту из картридера компьютера и вставляем ее в Raspberry Pi. Подключаем монитор (по HDMI или VGA), клавиатуру, а также — опционально — мышь и кабель Ethernet. Затем подаем питание через порт MicroUSB. При первой загрузке устройства нам будет продемонстрировано меню, предлагающее установить одну из нескольких операционных систем в свободное пространство на карте памяти (рис. 11.5).

После установки операционной системы Raspberry Pi станет загружаться в обычном режиме. Тем не менее, NOOBS остается на карте, поэтому, удерживая клавишу

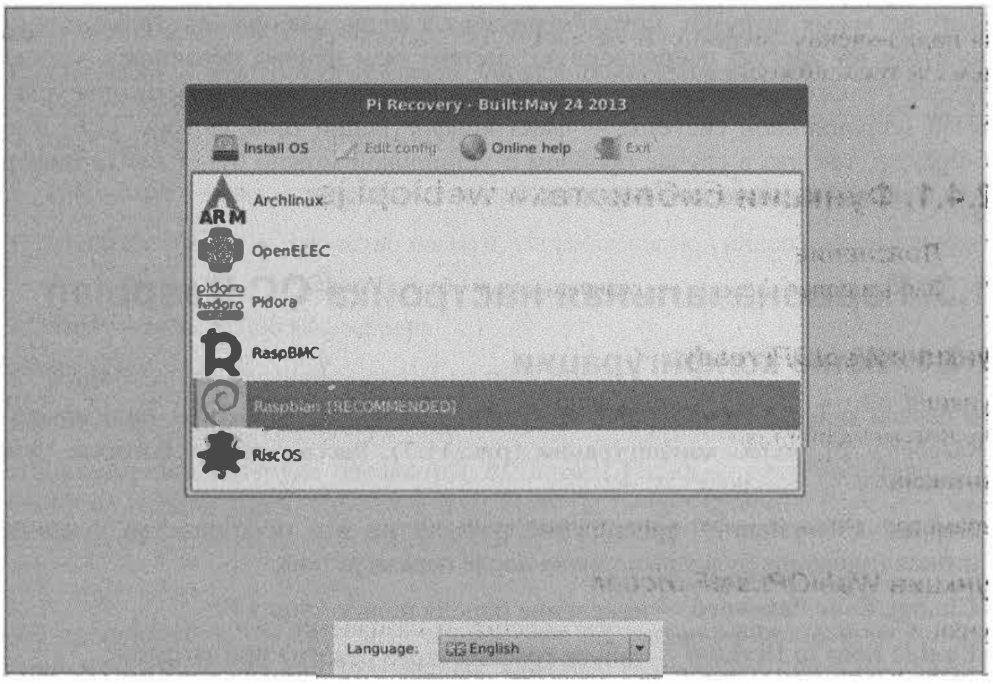


Рис. 11.5. Меню NOOBS при первой загрузке

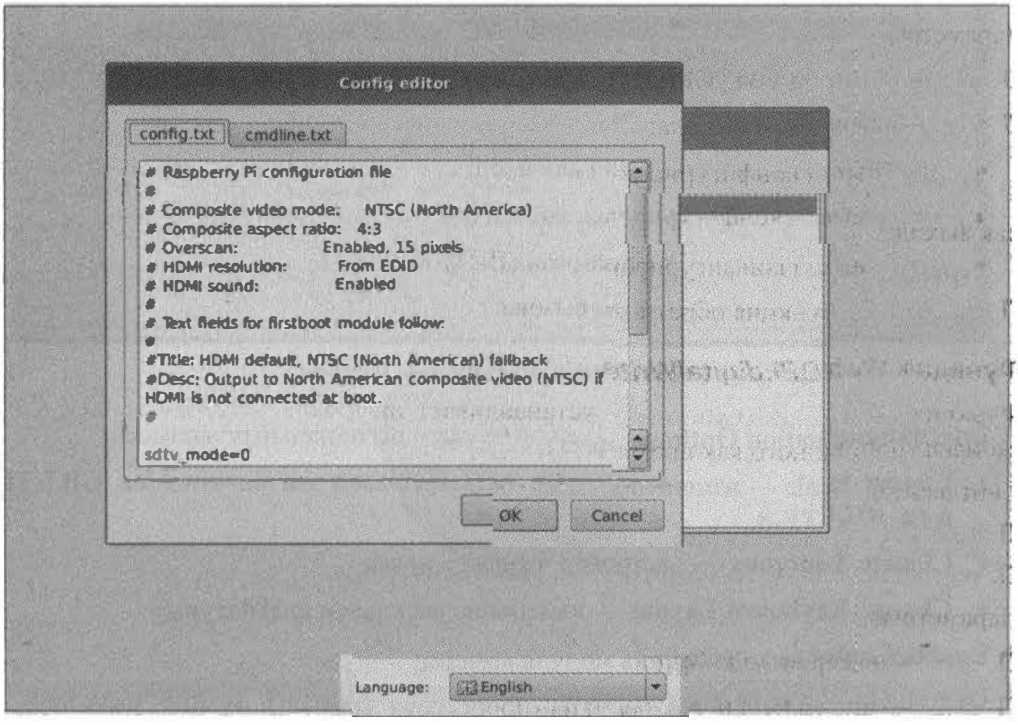


Рис. 11.6. Редактирование файла config.txt

<Shift> во время загрузки, можно вернуться к меню выбора. Это позволяет переключиться на другую операционную систему или заново переписать установку текущей, а также предоставляет удобный инструмент для редактирования установленной операционной системы — файл конфигурации config.txt (рис. 11.6) и даже веб-браузер, так что в случае проблем с установкой можно будет найти информацию в Сети на тематических форумах.

11.3. Первоначальная настройка ОС Raspbian

11.3.1. Меню конфигурации

При первом запуске ОС Raspbian мы увидим на экране монитора, подключенного к Raspberry Pi, меню конфигурации (рис. 11.7). Рассмотрим некоторые пункты этого меню:

- **Expand Filesystem** — расширение раздела на все пространство флеш-накопителя (операция будет выполнена после перезагрузки);
- **Change User Password** — изменение пароля пользователя Pi;
- **Enable Boot to Desktop** — запуск графического режима при загрузке;

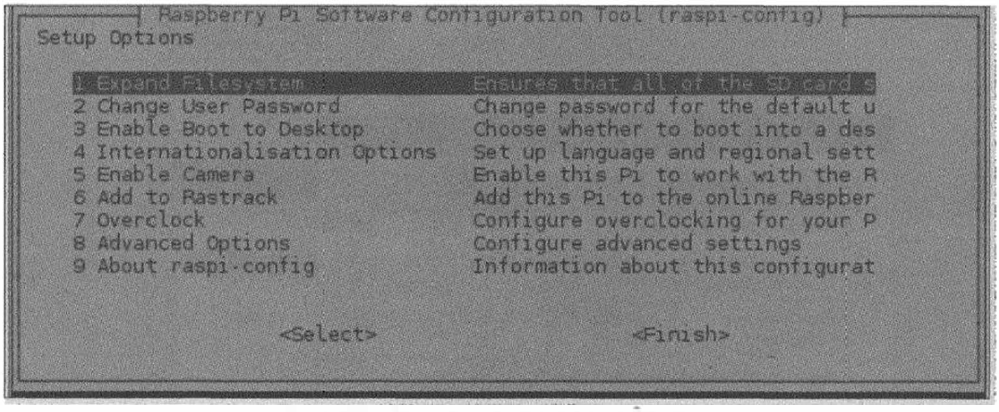


Рис. 11.7. Меню конфигурации Raspbian

- **Internationalisation Options** — выбор языка и региональных настроек:
  - **Change locale** — изменение языка: устанавливаем два значения: **en_GB.UTF-8** и **ru_RU.UTF-8**;
  - **Change Timezone** — настройка часового пояса;
  - **Change Keyboard Layout** — изменение раскладки клавиатуры;
- **Enable Camera** — поддержка модуля камеры;
- **Overclock** — увеличение частоты процессора (разгон), можно разогнать до 1 ГГц;

- **Advanced Options** — дополнительные параметры:
  - **Overscan** — настройка режима overscan (вылета развертки): если у вас по краю изображения имеется широкая черная полоса, то необходимо выключить этот режим (опция **Disable**);
  - **Hostname** — имя компьютера в сети (по умолчанию raspberrypi);
  - **Memory split** — количество памяти, выделяемое под видео (по умолчанию — 64 Мбайт);
  - **SSH** — включение ssh-сервера;
  - **Update** — обновление программы raspi-config;
- **About raspi-config** — о программе.

Настроив все необходимые параметры, выбираем **Finish** — система запросит разрешение на перезагрузку. Соглашаемся. Если вы позже захотите поменять какие-либо настройки, необходимо в консоли набрать команду:

```
sudo raspi-config
```

После продолжительной перезагрузки система выйдет на запрос логина и пароля: вводим логин пользователя Pi и пароль, который был введен при настройке. Для запуска графического режима набираем в консоли команду:

```
startx
```

и попадаем на рабочий стол Raspbian (рис. 11.8).

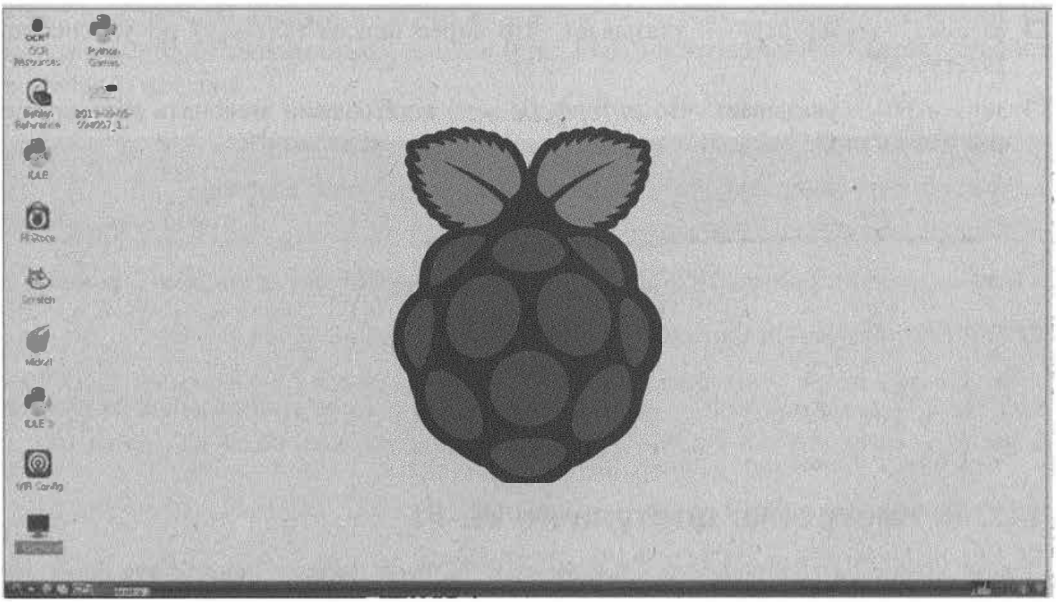


Рис. 11.8. Рабочий стол Raspbian

11.3.2. Настройка сетевых параметров

Скорее всего, вы не будете использовать Raspberry Pi для вывода информации на монитор — вам наверняка понадобится удаленный доступ к нему по ssh, поэтому сейчас следует настроить его сетевые параметры.

Установку сетевого соединения для Raspberry Pi обеспечивает файл конфигурации /etc/network/interfaces, и если IP-адрес Raspberry Pi получает по DHCP, то ничего править в этом файле вам не придется. В случае же назначения Raspberry Pi статического IP-адреса, выполняем команду:

```
sudo nano /etc/network/interfaces
```

и дописываем в файл конфигурации /etc/network/interfaces следующие строки:

```
iface eth0 inet static
address 192.168.0.117
netmask 255.255.255.0
gateway 192.168.0.28
auto eth0
```

где:

- `iface eth0 inet static` — указывает системе, что интерфейс (`iface eth0`) находится в диапазоне адресов IPv4 (`inet`) со статическим IP-адресом (`static`);
- `address 192.168.0.117` — указывает, что IP-адрес (`address`) нашей сетевой карты `192.168.0.117`;
- `netmask 255.255.255.0` — указывает, что наша маска подсети (`netmask`) имеет значение `255.255.255.0`;
- `gateway 192.168.0.28` — указывает, что адрес шлюза (`gateway`) по умолчанию `192.168.0.1`;
- `auto eth0` — указывает, что интерфейс `eth0` необходимо включать автоматически при загрузке системы с указанными здесь параметрами.

Если требуется изменить адреса серверов DNS, выполняем команду:

```
sudo gedit /etc/resolv.conf
```

и вписываем в открывшийся файл конфигурации resolv.conf следующие строки:

```
nameserver 192.168.1.1
nameserver 8.8.8.8
```

где: `192.168.1.1` и `8.8.8.8` — адреса серверов DNS. Если нужно добавить больше адресов — каждый адрес следует начинать с новой строки и со слова `nameserver`.

11.3.3. Настройка доступа по Wi-Fi

Далеко не всегда удобно вести к Raspberry Pi сетевой провод. Решить эту проблему можно с помощью адаптера Wi-Fi, подключаемого через USB. У меня в наличии имеется соответствующий адаптер (рис. 11.9), купленный в Китае за 7 долларов.

Позиционируется он в интернет-магазине как устройство, специально предназначенное для Raspberry Pi.

Еще раз напомню, что при подключении адаптера Wi-Fi, как и любого другого внешнего устройства, запитывать Raspberry Pi следует от достаточно мощного блока питания.



Рис. 11.9. USB-адаптер Wi-Fi

Итак, подключаем адаптер в USB-порт и проверяем, определит ли его система. Результат выполнения команды `iwconfig` (рис. 11.10) показывает, что наше устройство (**wlan0**) найдено.

```
pi@raspberrypi2: ~
Файл Правка Вид Поиск Терминал Справка
pi@raspberrypi2 ~ $ iwconfig
wlan0     unassociated  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

lo        no wireless extensions.

eth0      no wireless extensions.

pi@raspberrypi2 ~ $
```

Рис. 11.10. Результат выполнения команды `iwconfig`

Теперь сканируем пространство на поиск доступных беспроводных сетей командой:

```
sudo iwlist wlan0 scan
```

и видим (рис. 11.11), что найдена точка доступа **AndroidAP** — в нашем случае это планшет Samsung GalaxyTab 2, настроенный как точка доступа Wi-Fi.

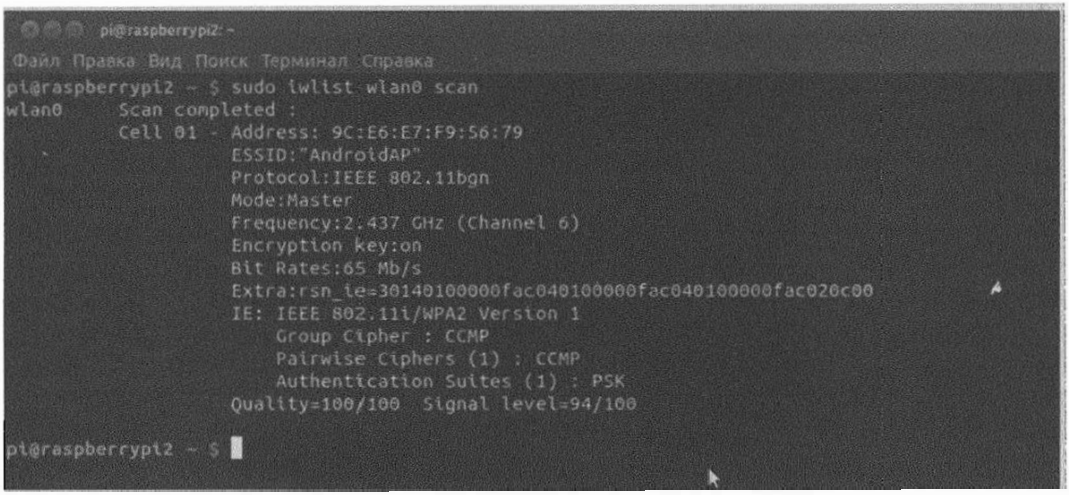


Рис. 11.11. Поиск доступных беспроводных сетей

Теперь соединимся с точкой доступа по WPA-шифрованию. Соединение с таким шифрованием поддерживает утилита wpa_passphrase. С помощью этой утилиты, которая входит в состав пакета wpa_supplicant, генерируем пароль на основе ключа доступа (для моей точки доступа **AndroidAP** — lcwt9634):

```
wpa_passphrase AndroidAP lcwt9634
```

Утилита выдает сгенерированную строку **psk** (рис. 11.12).

Далее блок, содержащий **ssid** сети (**AndroidAP**) и строку **psk**, вставляем в конец конфигурационного файла /etc/wpa_supplicant/wpa_supplicant.conf (рис. 11.13).



Рис. 11.12. Генерирование строки **psk**

В завершение перезагружаем наш Raspberry Pi и наблюдаем на планшете Samsung подключение Raspberry Pi (устройство **raspberrypi2**) к нашей точке доступа (рис. 11.14).

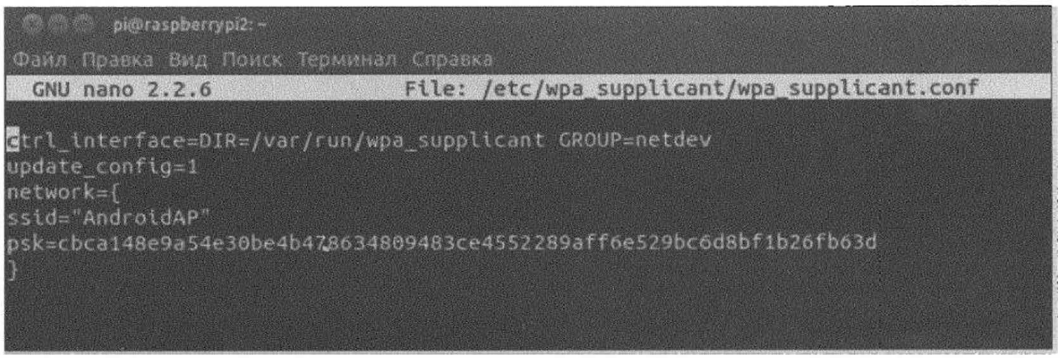


Рис. 11.13. Добавление данных в конфигурационный файл wpa_supplicant conf

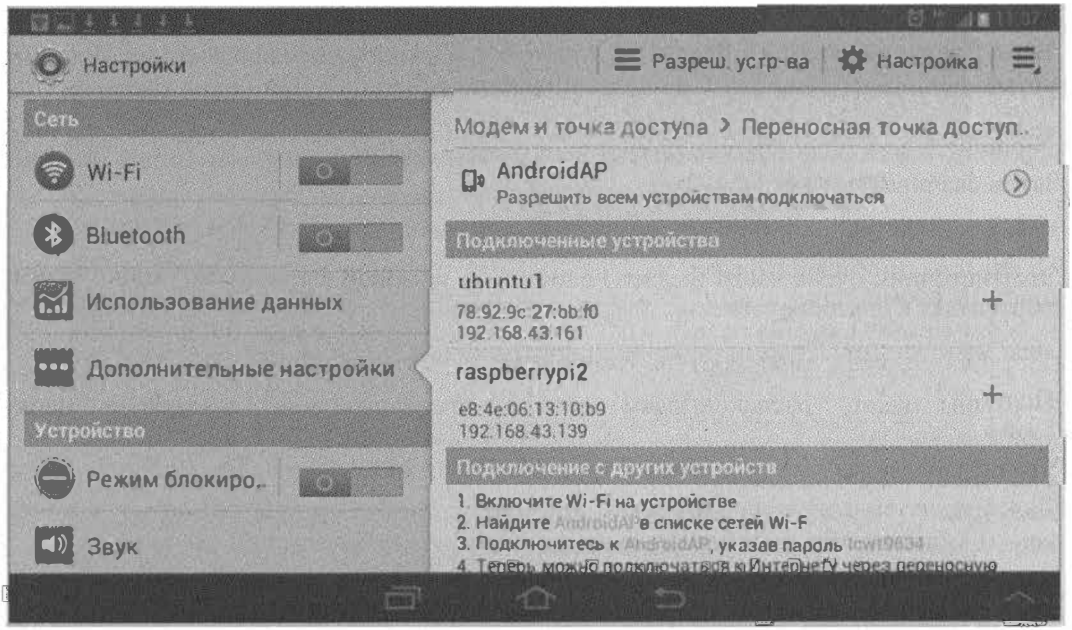


Рис. 11.14. Подключение к точке доступа Wi-Fi

11.3.4. Подключение 3G-модема

Поскольку сотовая связь доступна практически на всей населенной территории РФ, бесспорное достоинство 3G-модема состоит в том, что в любой точке зоны охвата можно соединиться с Интернетом, пусть и на относительно малых скоростях.

У меня как раз оказался в наличии 3G-модем ZTE MF180 от «Билайн» (рис. 11.15), и хотя я и не нашел его в списке совместимого с Raspberry Pi оборудования



Рис. 11.15. 3G-модем ZTE MF180 от «Билайн»

(**http://elinux.org/RPi_VerifiedPeripherals**), однако подключить его и получить Интернет на скорости 3G у меня получилось. Рассмотрим, как это удалось сделать.

Итак, для подключения к Raspberry Pi по USB 3G-модема прежде всего необходимо установить пакеты `ppp` и `sakis3g`. Для установки пакета `ppp` выполним команду:

```
sudo apt-get install ppp
```

Затем скачиваем пакет `sakis3g`:

```
sudo wget "http://www.sakis3g.org/versions/latest/armv4t/sakis3g.gz"
```

Этот источник очень часто бывает недоступен, и, попав в такую ситуацию, можно обратиться к альтернативному:

```
sudo wget "http://raspberry-at-home.com/files/sakis3g.tar.gz"
```

Получив пакет, распаковываем его в предварительно созданную папку **/usr/bin/modem3g**:

```
sudo mkdir /usr/bin/modem3g
sudo chmod 777 /usr/bin/modem3g
sudo cp sakis3g.tar.gz /usr/bin/modem3g
cd /usr/bin/modem3g
sudo tar -zxvf sakis3g.tar.gz
sudo chmod +x sakis3g
```

Теперь можно запустить скрипт `sakis3g`, работающий в терминале:

```
sudo ./sakis3g -interactive
```

В открывшемся главном меню программы (рис. 11.16) выбираем подключение к сети 3G. Скрипт `sakis3g` считывает с модема настройки и предлагает выбор точки доступа для подключения (рис. 11.17).

Для моего тарифа приходится вводить свои параметры точки доступа: `home.beeline.ru` (рис. 11.18).

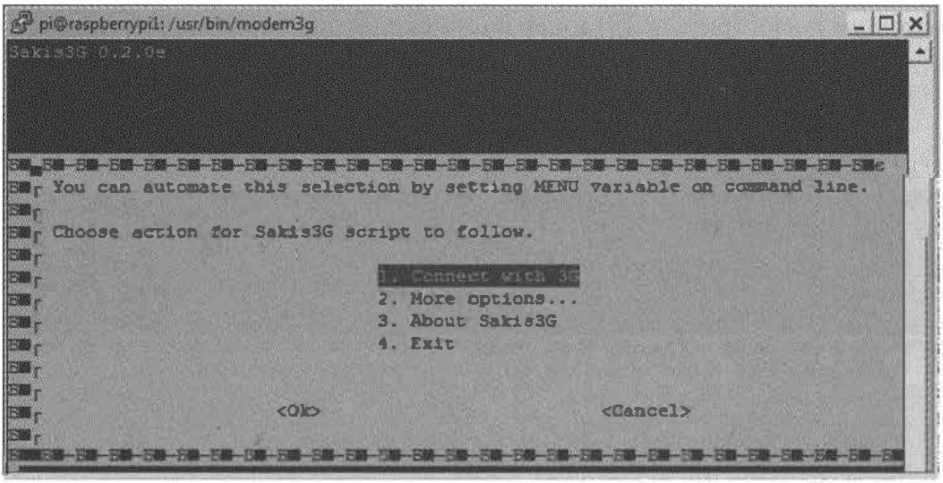


Рис. 11.16. Главное меню программы **Sakis3G**

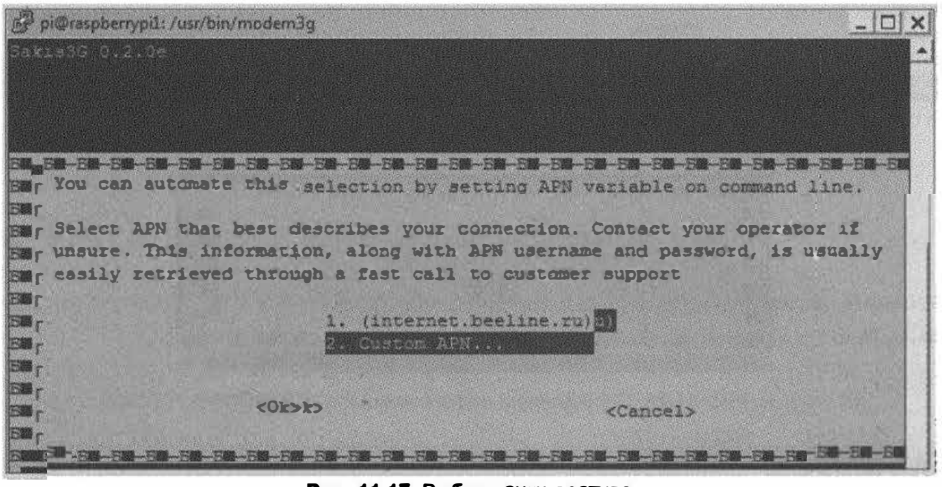


Рис. 11.17. Выбор точки доступа

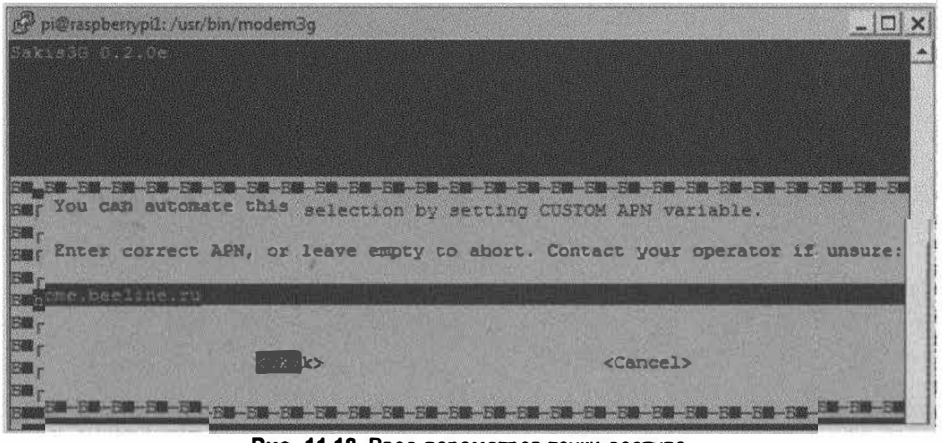


Рис. 11.18. Ввод параметров точки доступа

Затем вводим логин (рис. 11.19) и пароль — скрипт пытается подключиться к сети и выдает нам об этом соответствующее сообщение. Как можно видеть, подключиться к сети у нас получилось (рис. 11.20).

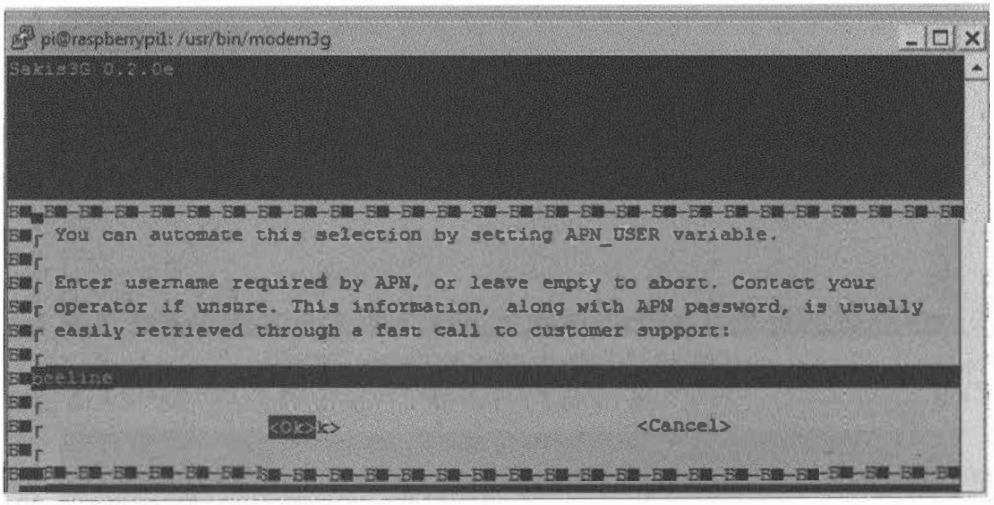


Рис. 11.19. Ввод логина/пароля для точки доступа

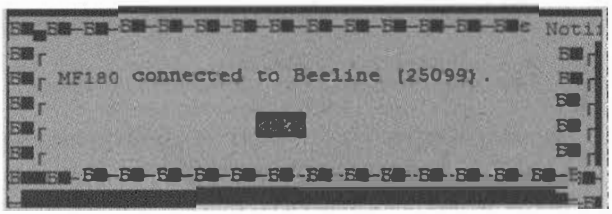


Рис. 11.20. Сообщение о подключении к точке доступа

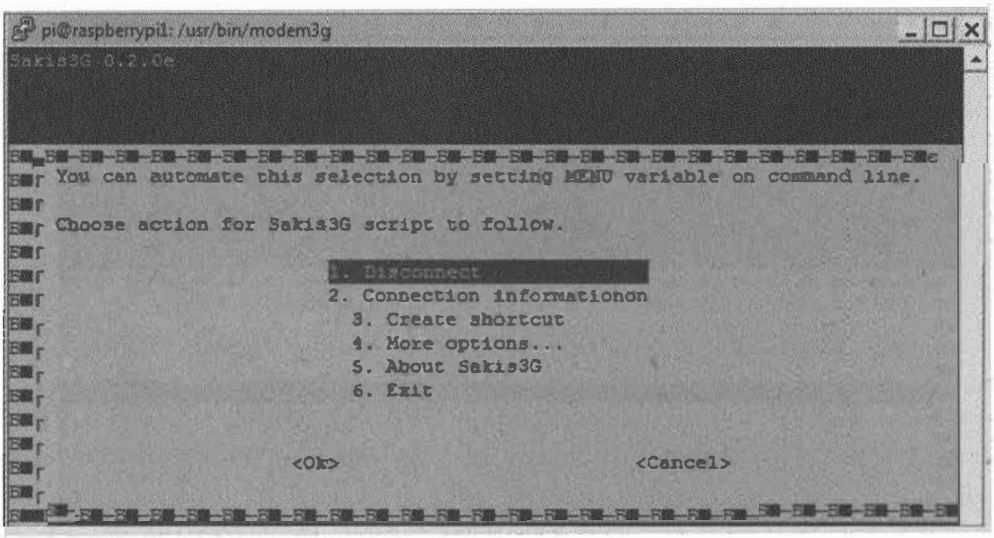


Рис. 11.21. Меню программы **Sakis3G**

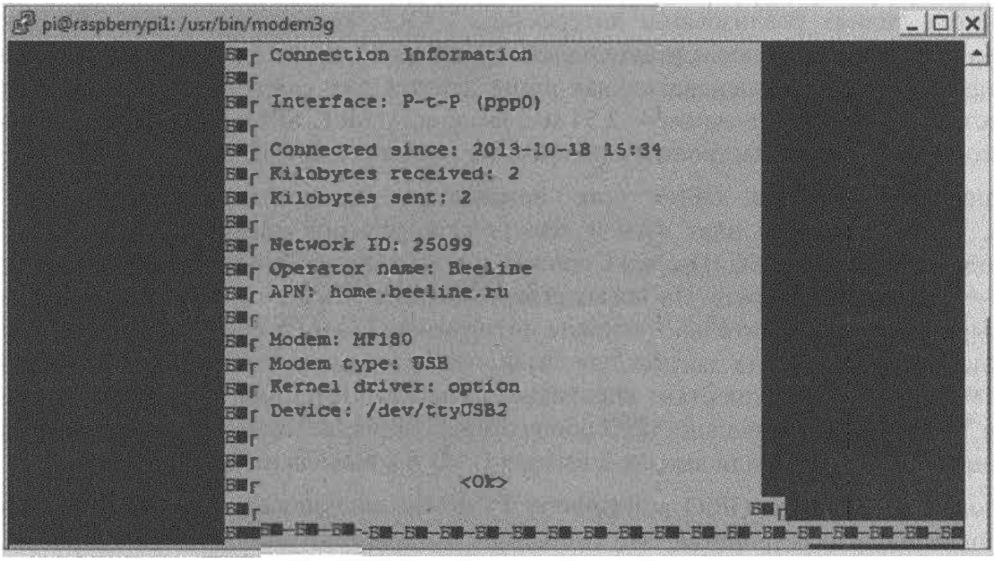


Рис. 11.22. Просмотр данных соединения

Далее мы попадаем в меню скрипта (рис. 11.21), где можно посмотреть настройки соединения (рис. 11.22) или разорвать его.

11.4. Интерфейс GPIO

Как уже отмечалось ранее, Raspberry Pi несет на борту интерфейс, называемый GPIO (рис. 11.23), — набор портов ввода/вывода общего назначения (General Purpose Input/Output). В моделях «A» и «B» — это 26 линий, из которых для управления доступны 17, в моделях «B+» — 40 линий, из которых доступны 26.

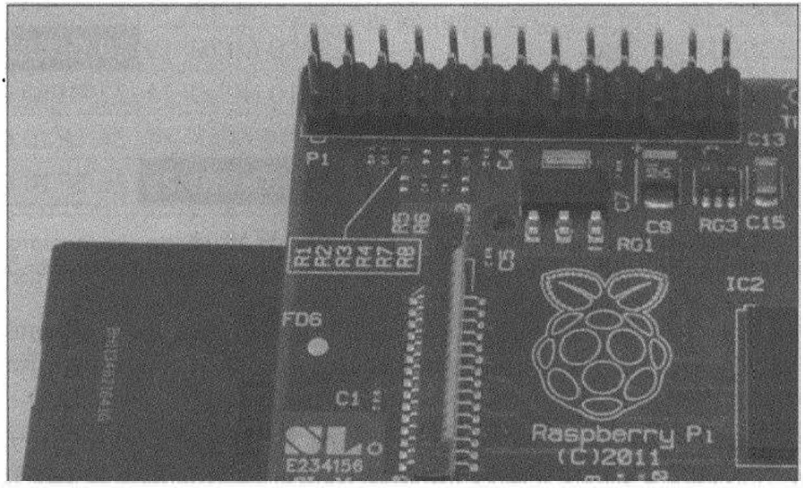


Рис. 11.23. Порты GPIO на плате Raspberry Pi

На этих линиях реализованы интерфейсы UART, консольный порт, SPI (Serial Peripheral Interface, последовательный периферийный интерфейс) и I²C (Inter-Integrated Circuit, последовательная шина данных для связи интегральных схем). Расстояние между выводами — 2,54 мм. Выводы UART, SPI и I²C в случае необходимости могут быть настроены как обычные порты ввода/вывода.

Назначение выводов GPIO (так называемая *распиновка*) представлено на рис. 11.24–11.26. Для плат «A» и «B» первой ревизии выводы 4, 9, 14, 17, 20, 25 обозначены как DNC (Do Not Connect), и подсоединять к ним что-либо не следует — плата может сгореть. На новых ревизиях платы разведены, но не распаяны, еще четыре GPIO, дополнительно дающие интерфейсы I²C и I²S (Inter-Integrated Circuit, последовательная шина данных для связи интегральных схем). Первые 26 выводов GPIO платы «B+» полностью идентичны разъемам GPIO плат «A» и «B», остальные 14 содержат 9 выводов GPIO общего назначения, которые могут быть сконфигурированы как вход или выход, 3 вывода GND и 2 вывода не задействованы.

С помощью выводов GPIO к Raspberry Pi можно подключать датчики и внешние исполнительные устройства, что открывает интересные возможности творческого использования Raspberry Pi.

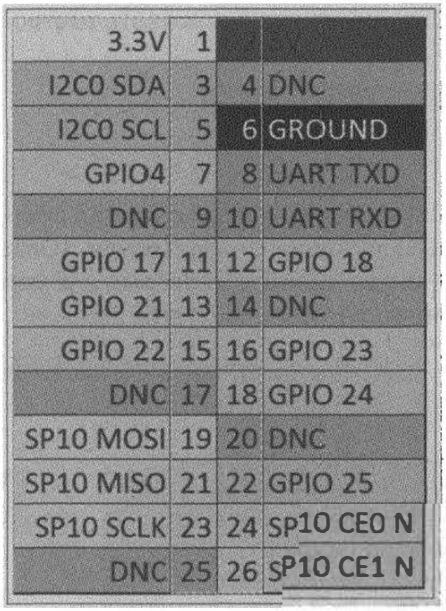


Рис. 11.24. «Распиновка» портов GPIO плат «A» и «B» Revision 1

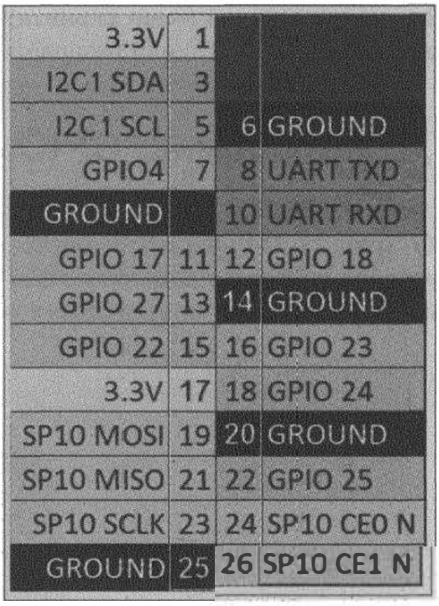
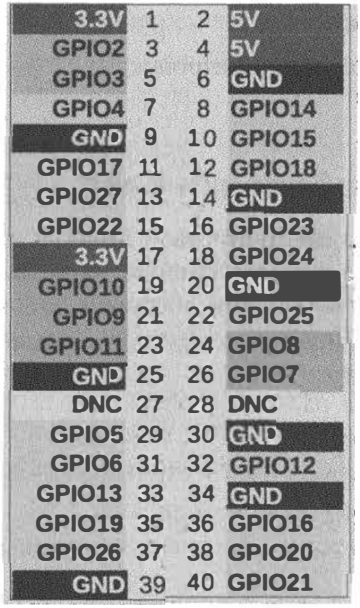


Рис. 11.25. «Распиновка» портов GPIO плат «A» и «B» Revision 2

При работе с портами GPIO следует помнить о некоторых их особенностях и соблюдать определенные меры предосторожности, чтобы не повредить Raspberry Pi. Вот основные из них:

- максимальный суммарный ток обоих выводов 3,3 В равен 50 мА, и эти выводы могут использоваться для питания внешних устройств только в том случае, если их потребляемый ток меньше 50 мА;



3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Рис. 11.26. «Распиновка» портов GPIO плат «B+» и «B 2»

- максимальный суммарный ток обоих выводов 5 В равен 300 мА, и эти выводы также могут использоваться для питания внешних устройств только в том случае, если их потребляемый ток меньше 300 мА;
- на GPIO нельзя подавать напряжение больше 3,3 В! Цифровые выводы GPIO имеют уровни напряжения 0–3,3 В и не совместимы с традиционными уровнями напряжения 0–5 В. Если подать на вывод GPIO логическую единицу, представляющую собой 5 В (а не 3,3 В), — этот вывод может выйти из строя;
- выводы GPIO 14 и GPIO 15 по умолчанию выполняют альтернативную функцию и являются выводами UART (RXD и TXD), поэтому после включения на них присутствует высокий уровень 3,3 В, однако программно их можно переконфигурировать в обычные выводы. Все остальные выводы GPIO после включения Raspberry Pi выполняют основную функцию и работают как обычные цифровые;
- все настраиваемые выводы GPIO — кроме GPIO 0 (SDA) и GPIO 1 (SCL) — по умолчанию являются входами и поэтому имеют высокое входное сопротивление, при этом подтяжка логического уровня у них не включена, так что после включения Raspberry Pi напряжение на них может «плавать»;
- выводы GPIO 0 (SDA) и GPIO 1 (SCL) по умолчанию «подтянуты» к питанию, поэтому после включения Raspberry Pi на них присутствует напряжение логической единицы (3,3 В);
- сигнал на любом из цифровых выводов может служить источником внешнего прерывания.

Нужно помнить, что GPIO — это выводы, непосредственно подключенные к процессору Raspberry Pi, они являются инструментом для взаимодействия с ним. Поэтому неосторожное обращение с GPIO может привести к необратимым последствиям для процессора.

Работать с GPIO можно двумя способами:

- используя оболочку bash и файловую систему Raspbian;
- используя языки программирования.

11.4.1. Управление GPIO из оболочки bash

ОС Raspbian представляет собой один из дистрибутивов Linux, а концепция Linux предполагает, что любой объект является файлом. Именно это позволяет выводить и считывать сигналы с GPIO обычными командами оболочки bash прямо в терминале. Вывод логической единицы при этом выглядит как команда записи "1" в файл, соответствующий нужному выводу:

```
sudo su -
echo "25" > /sys/class/gpio/export
echo "25" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio25/direction
echo "1" > /sys/class/gpio/gpio25/value
echo "0" > /sys/class/gpio/gpio25/value
```

Для чтения входов надо использовать команду cat и путь к файлу:

```
echo "24" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio0/direction
cat /sys/class/gpio/gpio24/value
```

Все операции должны выполняться от имени пользователя root.

11.4.2. Управление GPIO командами языка Python

Для работы с GPIO на языке Python требуется специальная библиотека RPi.GPIO. В новом дистрибутиве Raspbian она уже установлена, а если у вас дистрибутив старый, то для установки библиотеки RPi.GPIO выполните команду:

```
sudo apt-get install python-rpi.gpio
```

Чтобы использовать эту библиотеку, необходимо в программу на Python добавить строку импорта библиотеки RPi.GPIO:

```
Import RPi.GPIO as GPIO
```

При написании программы можно выбрать один из двух способов нумерации портов GPIO: первый — GPIO.BOARD — использует систему нумерации портов на плате Raspberry Pi. Преимущество этой системы нумерации в том, что ваше оборудование будет работать всегда, независимо от номера ревизии — вам не придется перемонтировать свой разъем или изменить имеющийся код. Вторая система нумерации — GPIO.BCM (номера BCM). Это более низкий уровень работы — с прямым обращением к номерам каналов на процессоре Broadcom. Выбор способа нумерации определяется соответствующими командами:

```
GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)
```

Следующие команды устанавливают режим работы контакта на вход или выход:

```
GPIO.setup(channel, GPIO.IN)
GPIO.setup(channel, GPIO.OUT)
```

Если входной канал ни к чему не подключен, его значение может «плыть». Следующие команды устанавливают начальную «подтяжку» вывода к питанию или к «земле»:

```
GPIO.setup(channel, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(channel, GPIO.IN, GPIO.PUD_DOWN)
```

Для выходов OUT можно установить начальное значение 0 или 1:

```
GPIO.setup(channel, GPIO.OUT, GPIO.LOW)
GPIO.setup(channel, GPIO.OUT, GPIO.HIGH)
```

Для чтения значения контакта GPIO, настроенного как вход IN, служит следующая команда:

```
GPIO.input(channel)
```

Значение контакта, настроенного как выход OUT, устанавливается следующей командой:

```
GPIO.output(channel, state)
```

В листинге 11.1 приведен пример использования команд работы с GPIO.

Листинг 11.1

```
import RPi.GPIO as GPIO          #подключаем библиотеку
     GPIO.setmode(GPIO.BCM)      #устанавливаем режим нумерации
     GPIO.setup(7, GPIO.OUT)     #конфигурируем GPIO 7 как выход
     GPIO.setup(8, GPIO.IN)      #конфигурируем GPIO 8 как вход
     GPIO.output(7, True)        #выводим на GPIO 7 логическую "1" (3.3 V)
     GPIO.output(7, False)       #выводим на GPIO 7 логический "0"
     signal = GPIO.input(8)      #считываем с GPIO 8 в переменную signal
     GPIO.cleanup()              #завершаем работу с GPIO
```

Библиотека RPi.GPIO позволяет использовать контакты GPIO в качестве выходов ШИМ (сигналов широтно-импульсной модуляции).

Для создания экземпляра ШИМ служит команда:

```
p = GPIO.PWM(channel, frequency)
```

где frequency — частота, Гц.

Для старта ШИМ на контакте:

```
p.start(dc)
```

где dc — рабочий цикл ШИМ (0,0–100,0).

Для изменения частоты сигнала:

```
p.ChangeFrequency(freq)
```

где `freq` — частота сигнала, Гц.

Для изменения рабочего цикла ШИМ:

```
p.ChangeDutyCycle(dc)  # where 0.0 <= dc <= 100.0
```

Останов выдачи сигнала ШИМ на контакте:

```
p.stop()
```

В листинге 11.2 представлен пример плавного включения/выключения светодиода, подключенного к контакту.

Листинг 11.2

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 50)  # контакт 12 частота 50 Гц
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```

Функция ожидания события — изменения состояния на входе `IN` — выглядит следующим образом:

```
GPIO.wait_for_edge(channel, GPIO.RISING)
GPIO.wait_for_edge(channel, GPIO.FALLING)
GPIO.wait_for_edge(channel, GPIO.BOTH)
```

Эта функция прерывает выполнение программы до изменения состояния на входе GPIO. В отличие от нее функция `add_event_detected()` выполняется в цикле программы, и для того, чтобы узнать об изменении состояния на контакте GPIO, необходимо в цикле проверять наступление события `event_detected()`:

```
GPIO.add_event_detect(channel, GPIO.RISING
.................
if GPIO.event_detected(channel):
    print('Button pressed')
```

Каждый контакт GPIO может быть настроен на работу в режиме прерывания — в этом случае при наступлении события на контакте, управление передается функции обработки прерывания. Функция обработки прерывания работает в отдельном потоке, не прерывая выполнения основной программы:

```
def my_callback(channel):
    print('обработка прерывания!')
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback)
```

В конце любой программы рекомендуется очистить все ресурсы, которые могли использоваться. Для такой очистки в конце скрипта надо предусмотреть команду:

```
GPIO.cleanup()
```

Для сброса одного контакта служит команда:

```
GPIO.cleanup(channel)
```

11.5. Raspberry Pi и датчик температуры DS18B20 на шине 1-Wire

11.5.1. Подключение датчика DS18B20 к Raspberry Pi

В *главе 7* мы уже рассматривали датчик температуры DS18B20, работающий по однопроводному интерфейсу 1-Wire.

Схема подключения датчика DS18B20 к контактам GPIO Raspberry Pi по протоколу 1-Wire приведена на рис. 11.27. Протокол 1-Wire позволяет подключить к одной шине и несколько датчиков DS18B20 (рис. 11.28).

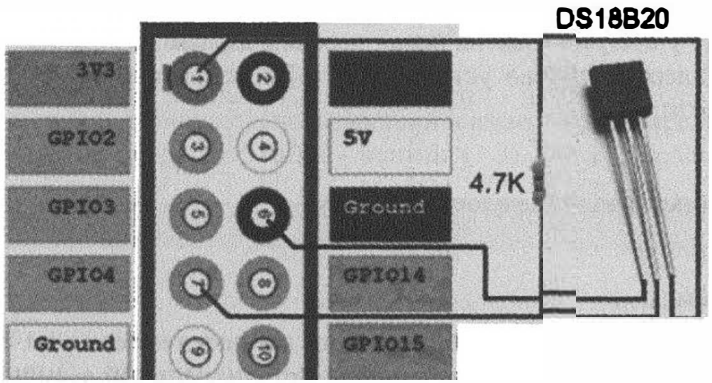


Рис. 11.27. Схема подключения датчика DS18B20 к Raspberry Pi по шине 1-Wire

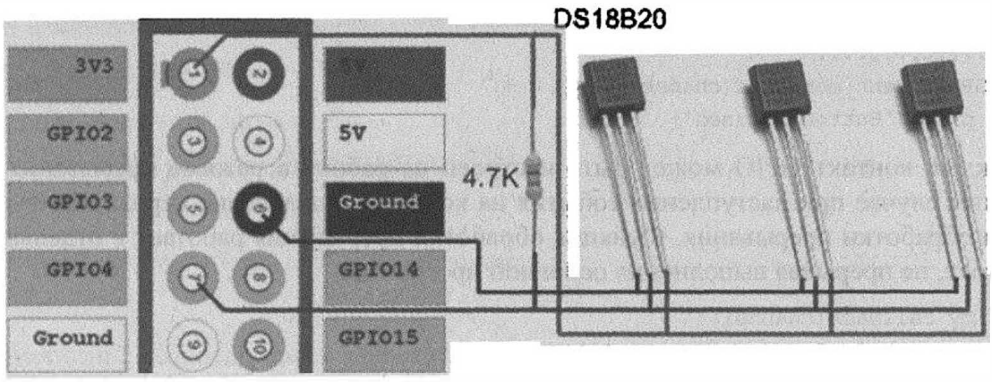


Рис. 11.28. Схема подключения к Raspberry Pi нескольких датчиков DS18B20

Для использования с Raspberry Pi датчиков DS18B20 на шине 1-Wire необходимо загрузить два специальных модуля: w1-gpio и w1-term. Для этого набираем в терминале:

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
```

Первая строка здесь активирует протокол модуля 1-Wire на GPIO4, а вторая — загружает модуль, который, собственно, и читает температуру с шины 1-Wire. Чтобы модули автоматически загружались после перезагрузки или выключения Raspberry Pi, две эти строки нужно добавить в файл /etc/modules, отвечающий за автозагрузку модулей, для чего набираем в терминале:

```
sudo nano /etc/modules
```

и добавляем в конец открывшегося для редактирования файла две следующие строчки:

```
w1-gpio
w1-therm
```

Через несколько секунд после подключения датчика в каталоге /sys/bus/w1/devices появится вложенный каталог вида 28-xxxxxxxxxxxx (рис. 11.29). Если этого не произошло, значит у вас установлена новая версия Raspbian, в которой по умолчанию отключена поддержка дерева устройств. Чтобы исправить ситуацию, открываем файл /boot/config.txt:

```
sudo nano /boot/config.txt
```

вписываем в конец файла следующую строку:

```
dtoverlay=w1-gpio
```

и перезагружаемся.

Для каждого подключенного по шине 1-Wire датчика температуры создается свой такой каталог. Название каждого каталога — это уникальный серийный номер датчика DS18B20.

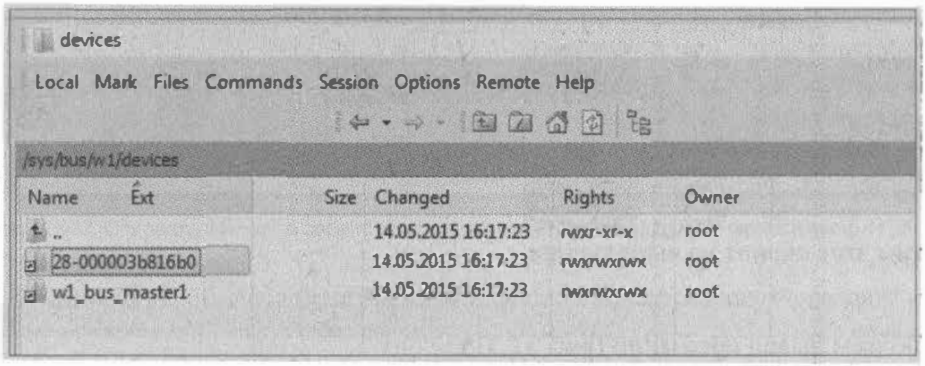


Рис. 11.29. Каталог для датчика DS18B20 в папке /sys/bus/w1/devices

Данные температуры хранятся в файле 28-000003b816b0/w1-slave, содержимое которого можно считать по команде:

```
cat /sys/bus/w1/devices/28-000003b816b0/w1_slave
```

Результат считывания показаний датчика DS18B20 приведен на рис. 11.30.

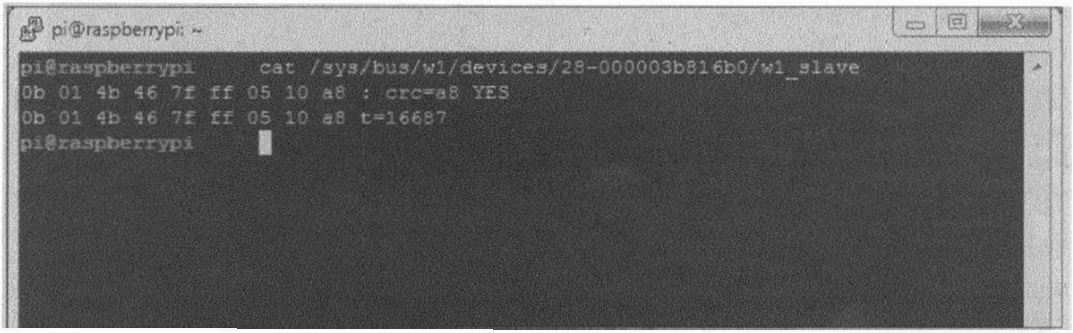


Рис. 11.30. Содержимое файла с показаниями датчика DS18B20

Обратите внимание на присутствующий в считанной информации фрагмент: **t = 16687**. Это температура по шкале Цельсия, умноженная на 1000. То есть, реально замеренная температура составляет 16687/1000 = 16,687 °C. Не надо обманываться относительно трех знаков после запятой — абсолютная точность датчика всего 0,5 °C.

Напишем на языке Python скрипт, считывающий и выводящий каждые три секунды данные температуры с датчика DS18B20 (листинг 11.3).

Листинг 11.3

```
#-*-coding:utf-8 -*-
from time import sleep

while 1:
  tfile=open("/sys/bus/w1/devices/28-000003b816b0/w1_slave")
```

```
ttext=tfile.read()
tfile.close()
temp=ttext.split("\n")[1].split(" ")[9]
temperature=float(temp[2:])/1000
print "temperature="+str(temperature)
sleep(3)
```

Запустим этот скрипт на выполнение:

```
python listing_11_03.py
```

и наблюдаем вывод показаний (рис. 11.31).

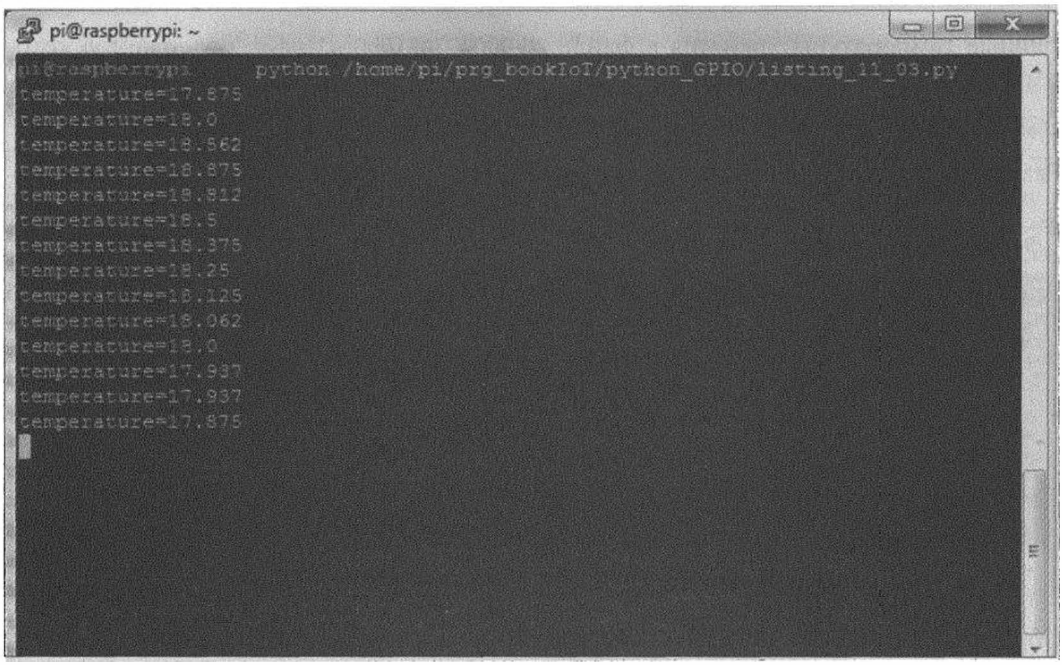


Рис. 11.31. Работа скрипта, читающего показания датчика DS18B20

11.5.2. Отправка данных с датчика DA18B20 в сервис «Народный мониторинг»

Как известно, для регистрации какого-либо датчика в сервисе «Народный мониторинг» сначала необходимо отправить туда данные его замеров. Так что, прежде всего напишем скрипт, отправляющий данные с датчика в этот сервис. Для этого создаем файл listing_11_04.py, заносим в него код из листинга 11.4, запускаем наш скрипт на выполнение:

```
python listing_11_04.py
```

и наблюдаем отправку данных (рис. 11.32).

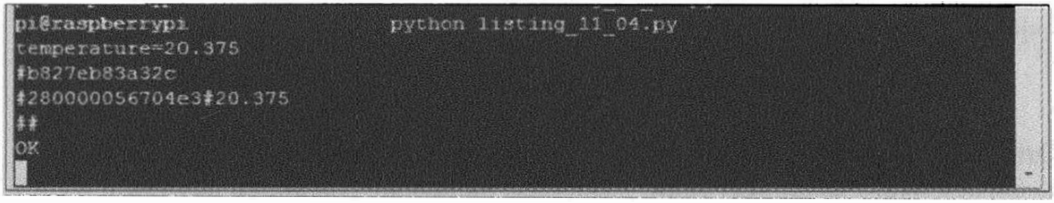


Рис. 11.32. Отправка данных в сервис «Народный мониторинг» из python-скрипта listing_11_04.py

Листинг 11.4

```
#-*-coding:utf-8 -*-

import socket
from time import sleep

# MAC-адрес устройства. Заменить на свой!
DEVICE_MAC = 'b827eb83a32c'
# идентификатор устройства
SENSOR_ID_1 = '28000003b816b0'
# значения датчика, тип float/integer
sensor_value_1 = 0

while 1:
   tfile=open("/sys/bus/w1/devices/28-000003b816b0/w1_slave")
   ttext=tfile.read()
   tfile.close()
   temp=ttext.split("\n")[1].split(" ")[9]
   temperature=float(temp[2:])/1000
   print "temperature="+str(temperature)
   sensor_value_1=temperature

   # создание сокета
   sock = socket.socket()
   # обработчик исключений
   try:
    # подключаемся к сокету
    sock.connect(('narodmon.ru', 8283))
    # пишем в сокет единичное значение датчика
    sock.send("#{}\n#{}#{}\n##".format(DEVICE_MAC, SENSOR_ID_1,
sensor_value_1))
    # пишем в сокет множественные значения датчиков
    # sock.send("#{}\n#{}#{}\n#{}#{}\n##".format(DEVICE_MAC, SENSOR_ID_1,
                                  sensor_value_1, SENSOR_ID_2, sensor_value_2))
    # читаем ответ
    data = sock.recv(1024)
    sock.close()
    print data
```

```
except socket.error, e:
 print('ERROR! Exception {}'.format(e))
sleep(600)
```

Отправив данные, заходим в свой профиль на сайте **http://www.narodmon.ru**, выбираем пункт меню **Датчики**, нажимаем на кнопку **Добавить датчик** и в соответствующее поле открывшегося окна вносим MAC-адрес нашего устройства (используем MAC-адрес сетевой платы Raspberry Pi). Если данные были переданы успешно, наша плата появится в списке устройств (рис. 11.33).

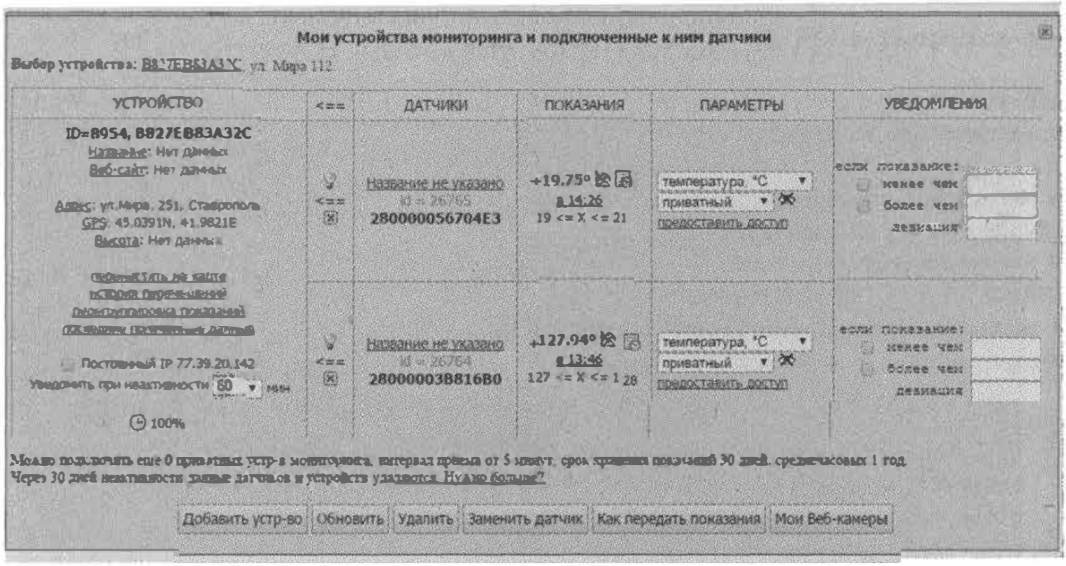


Рис. 11.33. Добавление датчика на сайт сервиса «Народный мониторинг»

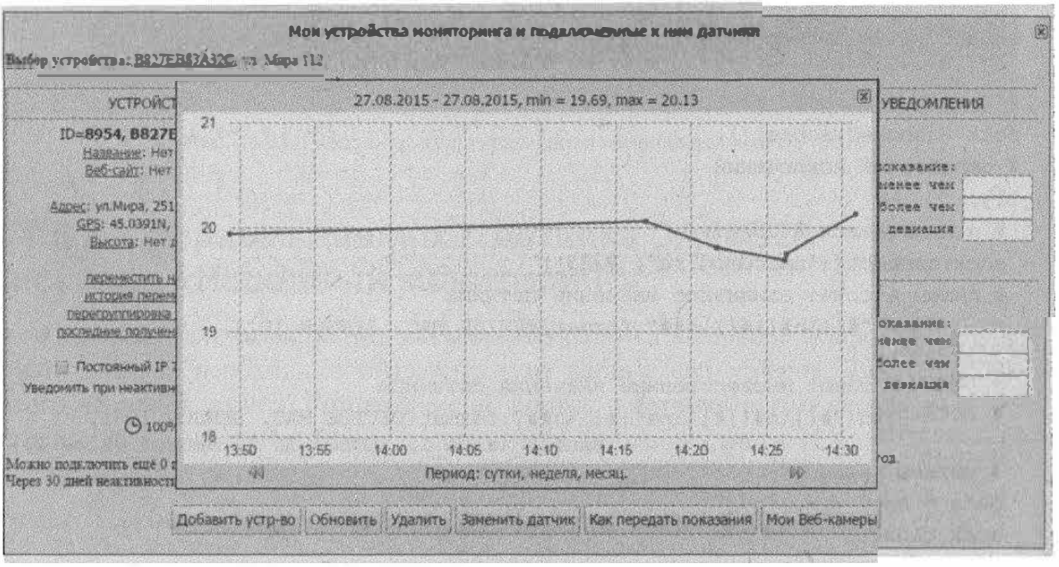


Рис. 11.34. График передачи показаний датчика температуры

Осталось настроить параметры датчика нашего устройства, и через некоторое время можно будет увидеть график его показаний (рис. 11.34).

ЭЛЕКТРОННЫЙ АРХИВ

Скрипт, соответствующий листингу 11.4, можно найти в файле python\listing_11_04.py сопровождающего книгу электронного архива.

11.6. Raspberry Pi и датчик освещенности BH1750 на шине I²C

11.6.1. Подключение датчика BH1750 к Raspberry Pi

В *главе 6* мы уже рассматривали датчик освещенности BH1750 на шине I²C — подключим его теперь к Raspberry Pi.

Для работы с датчиками I²C на Raspberry Pi необходимо сначала настроить поддержку интерфейса I²C, который в нем отключен по умолчанию. Поэтому запускаем утилиту конфигурации raspi-config:

```
sudo raspi-config
```

и выбираем в главном меню пункт **Advanced options**, а в открывшемся окне (рис. 11.35) — вариант **A7 I2C**, чем и включаем поддержку интерфейса I²C.

Затем добавляем в файл **/etc/modules**, отвечающий за автозагрузку модулей, следующие строки:

```
i2c-bcm2708
i2c-dev
```

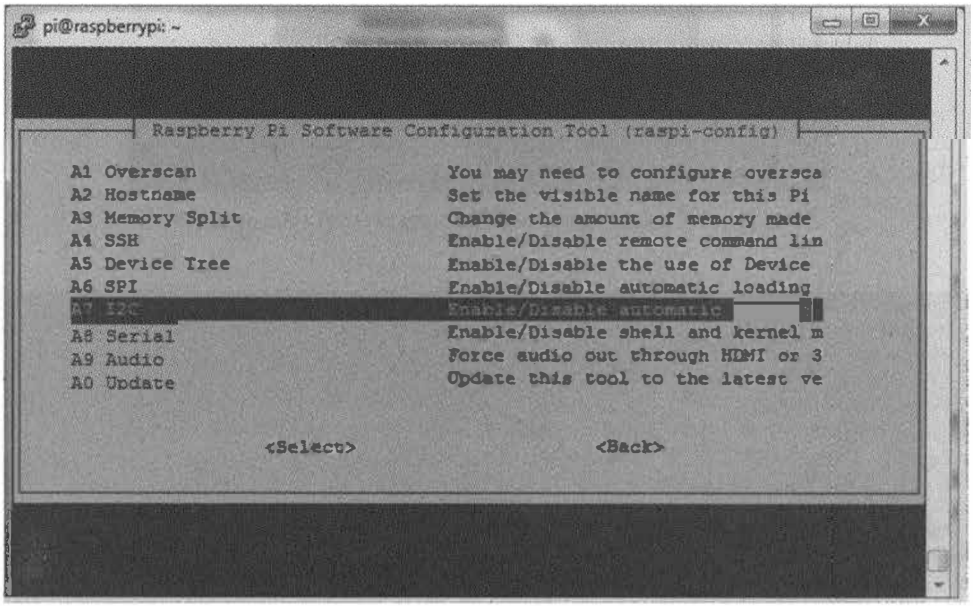


Рис. 11.35. Настройка поддержки протокола I²C в меню конфигурации raspi-config

и устанавливаем утилиты python-smbus и i2c-tools:

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Теперь подключаем к Raspberry Pi датчик освещенности BH1750 (рис. 11.36) и проверяем, видит ли Raspberry Pi наш датчик, — отдаем в терминале команду `i2cdetect`, которая содержится в утилите i2c-tools:

```
i2cdetect -y 1
```

и видим «решетку адресов» (рис. 11.37) — адрес нашего датчика здесь: **0x23**.

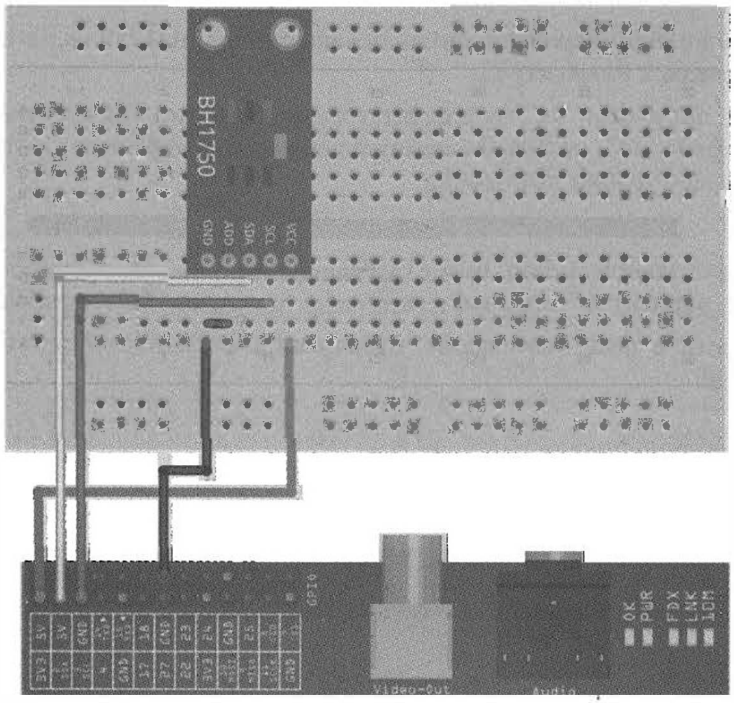


Рис. 11.36. Схема подключения датчика BH1750 к Raspberry Pi

```
pi@raspberrypi      sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- 23 -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi
```

Рис. 11.37. «Решетка адресов» I^2C-датчиков, подключенных к Raspberry Pi

11.6.2. Получение на Raspberry Pi данных с датчика BH1750

Напишем на языке Python скрипт получения Raspberry Pi данных с датчика BH1750 (листинг 11.5).

Листинг 11.5

```
#!/usr/bin/python
import smbus
import time

# Установка констант по технической документации
DEVICE     = 0x23   # I2C адрес датчика

POWER_DOWN = 0x00   # Неактивное состояние
POWER_ON   = 0x01   # Включение
RESET      = 0x07   # Сбросить значение регистра Не POWER_DOWN = 0x00 #

# Запуск измерения с разрешением 4 lx. Время, как правило, 16 мс.
CONTINUOUS_LOW_RES_MODE = 0x13
# Запуск измерения с разрешением 1 lx. Время, как правило, 120 мс
CONTINUOUS_HIGH_RES_MODE_1 = 0x10
# Запуск измерения с разрешением 0.5 lx. Время, как правило, 120 мс
CONTINUOUS_HIGH_RES_MODE_2 = 0x11
# Запуск измерения с разрешением 1 lx. Время, как правило, 120 мс
# Устройство автоматически устанавливается на Power Down после измерения.
ONE_TIME_HIGH_RES_MODE_1 = 0x20
# Запуск измерения с разрешением 0.5 lx. Время, как правило, 120 мс
# Устройство автоматически устанавливается на Power Down после измерения.
ONE_TIME_HIGH_RES_MODE_2 = 0x21
# Запуск измерения с разрешением 1 lx. Время, как правило, 120 мс
# Устройство автоматически устанавливается на Power Down после измерения.
ONE_TIME_LOW_RES_MODE = 0x23

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0
bus = smbus.SMBus(1)  # Rev 2 Pi uses 1

def convertToNumber(data):
  # перевод данных датчика в числовое значение
  return ((data[1] + (256 * data[0])) / 1.2)

def readLight(addr=DEVICE):
  data = bus.read_i2c_block_data(addr,ONE_TIME_HIGH_RES_MODE_1)
  return convertToNumber(data)
```

```
def main():

  while True:
    print "Light Level : " + str(readLight()) + " lx"
    time.sleep(0.5)

if __name__=="__main__":
   main()
```

Запустим этот скрипт в терминале:

```
python listing_11_05.py
```

и увидим там вывод показаний датчика BH1750 (рис. 11.38).

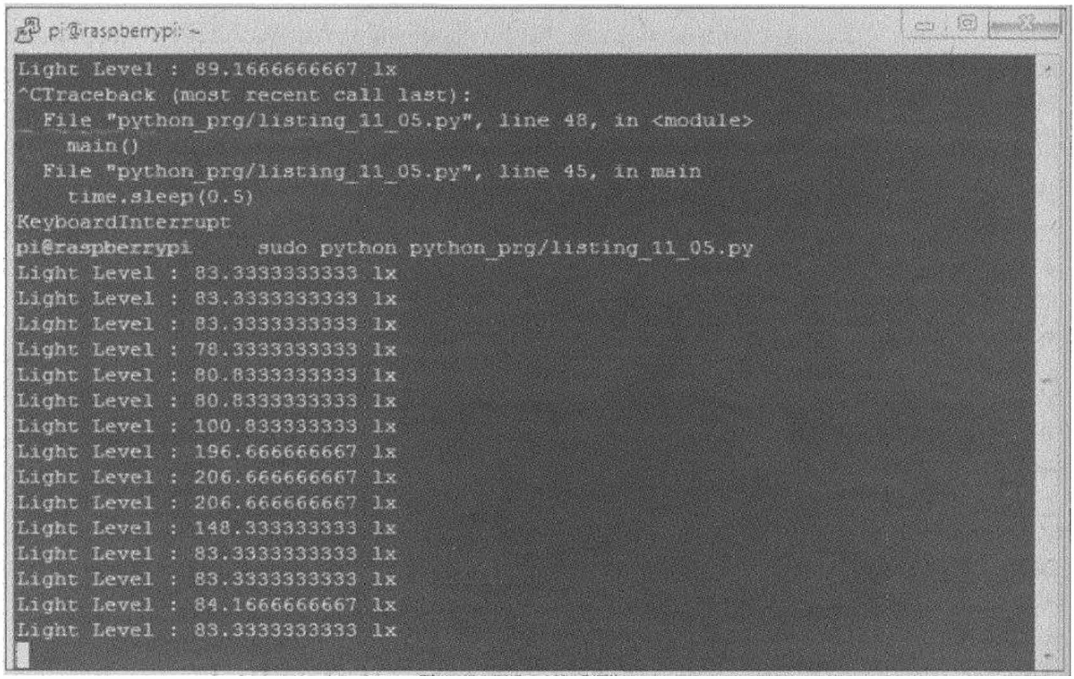


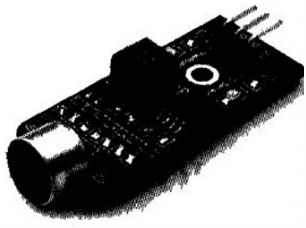
Рис. 11.38. Вывод показаний датчика BH1750

ЭЛЕКТРОННЫЙ АРХИВ

Скрипт, соответствующий листингу 11.5, можно найти в файле python\listing_11_05.py сопровождающего книгу электронного архива.

Теперь данные освещенности с датчика BH1750 можно отправлять в облачный сервис — например, в тот же «Народный мониторинг», добавив в этот скрипт код отправки данных из листинга 11.4.

ГЛАВА 12



WebIOPi — веб-интерфейс и облако для Raspberry Pi

Воспользуемся для доступа к портам GPIO фреймворком WebIOPi — сервисом Internet of Things, позволяющим контролировать состояние и управлять всеми портами GPIO локально или удаленно, из браузера или любого приложения.

Возможности WebIOPi:

- доступ к функционалу сервиса через основанный на HTTP интерфейс REST API (Representational State Transfer, передача состояния представления) и протокол ограниченного применения CoAP (Constrained Application Protocol) с поддержкой мультикаста (многоадресного вещания);
- работа с портами GPIO, Serial, I^2C, SPI, 1-Wire;
- встроенная поддержка более чем 30 устройств, включая ЦАП, АЦП, датчики;
- возможность работы с облачным сервисом Weaved;
- совместимость с Python 2 и 3;
- защита логином/паролем;
- множество примеров.

12.1. Установка WebIOPi на ОС Raspbian

Для установки фреймворка WebIOPi на ОС Raspbian необходимо скачать соответствующий архив, извлечь содержащиеся в нем файлы и запустить сценарий установки, который автоматически загрузит и инсталлирует необходимые зависимости:

```
wget http://webiopi.googlecode.com/files/WebIOPi-0.7.1.tar.gz
tar xvzf WebIOPi-0.7.1.tar.gz
cd WebIOPi-0.7.1
sudo ./setup.sh
```

В процессе достаточно продолжительной установки вам будет предложено создать профиль в облачном сервисе Weaved (**https://www.weaved.com**) для доступа

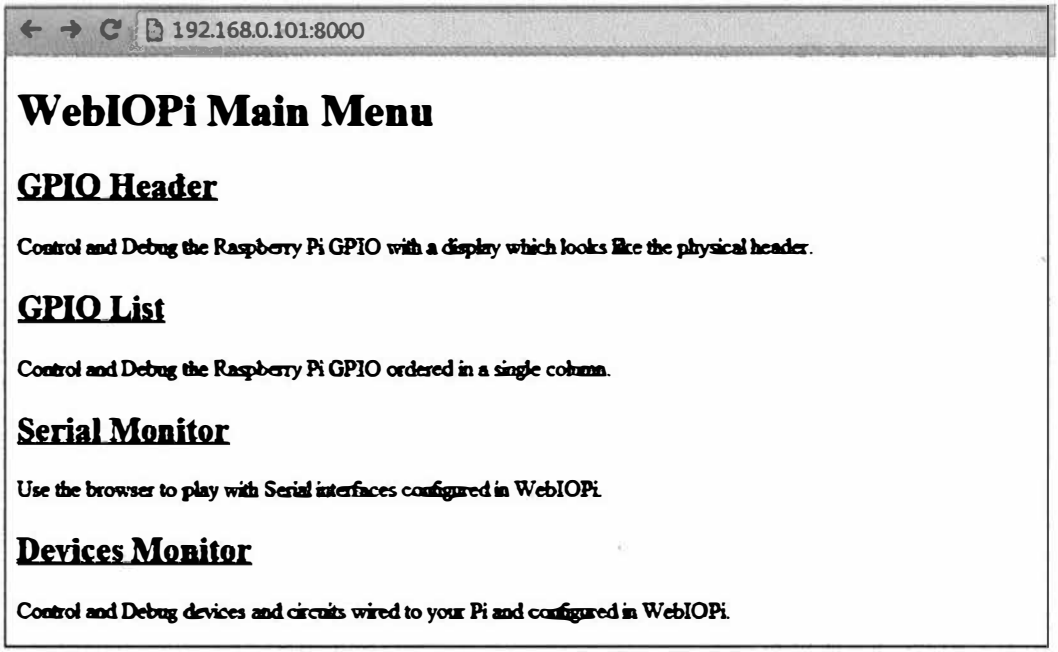


Рис. 12.1. Стартовая страница WebIOPi

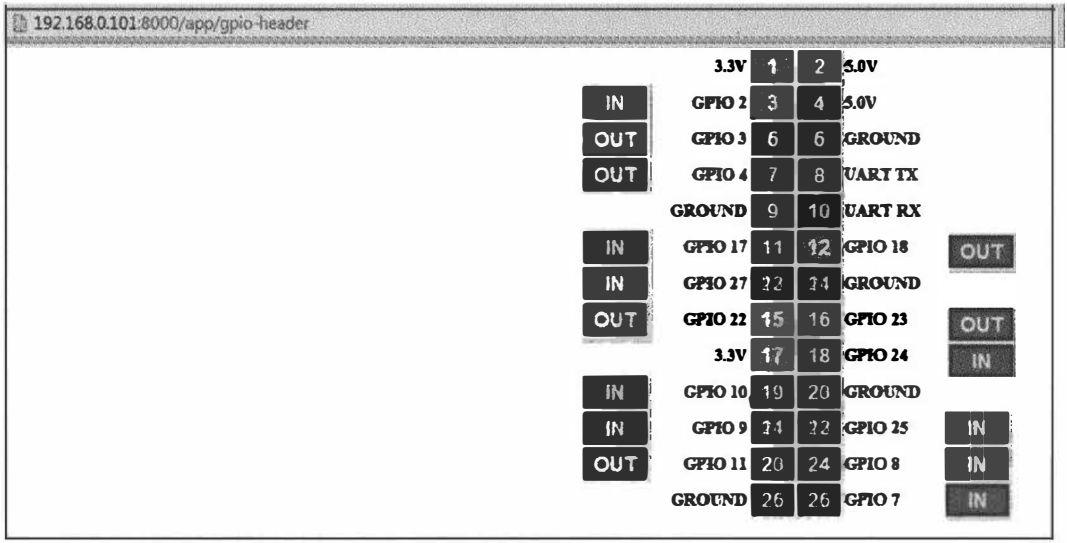


Рис. 12.2. Страница управления выводами GPIO

к WebIOPi из сети Интернет (установка сервиса Weaved рассмотрена в *разд. 12.7*). Установленный WebIOPi лучше запускать как сервис:

```
sudo /etc/init.d/webiopi start
```

Если нужно, что бы WebIOPi стартовал автоматически при загрузке системы, выполните следующую команду:

```
sudo update-rc.d webiopi defaults
```

Теперь можно открыть веб-браузер на любом компьютере домашней сети, набрать адрес: `http://iprasp:8000` (где `iprasp` — IP-адрес Raspberry Pi) и авторизоваться — имя пользователя (логин): `webiopi`, пароль: `raspberry`. В результате в браузере откроется стартовая страница WebIOPi (рис. 12.1).

На странице управления выводами GPIO (рис. 12.2), открывающейся по ссылке **GPIO Header**, можно задать режим работы любой ножки и установить значение на выходе.

12.2. Задание пользовательского пароля WebIOPi

Сервер WebIOPi использует зашифрованный файл etc/webiopi/passwd, содержащий логин и пароль (по умолчанию логин: `webiopi`, пароль: `raspberry`). Чтобы изменить пароль, необходимо выполнить команду:

```
webiopi-passwd
```

и далее следовать инструкциям (рис. 12.3).

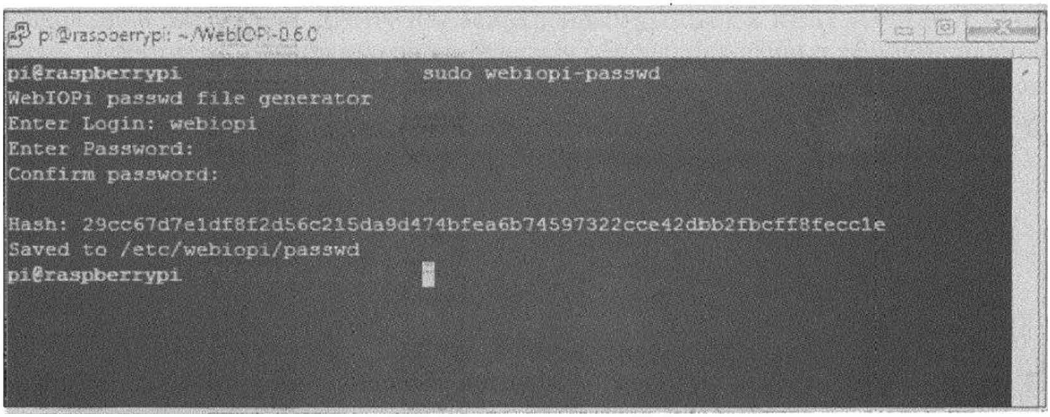


Рис. 12.3. Изменение пароля WebIOPi

После изменения пароля сервер необходимо перезагрузить:

```
sudo /etc/init.d/webiopi restart
```

Для снятия защиты при входе в WebIOPi необходимо либо ввести пустой пароль, либо удалить файл /etc/webiopi/passwd.

12.3. Настройка сервера WebIOPi

Настраивается сервер WebIOPi внесением изменений в файл его конфигурации: /etc/webiopi/config. Синтаксис этого файла такой же, как и у прочих INI-файлов, — он содержит несколько разделов, содержащих пары «ключ = значение».

- Блок [HTTP] позволяет включить или отключить HTTP, а также изменить значение порта. В этом блоке можно изменить местоположение файла passwd, домашней папки и название индексного файла HTML:

```
[HTTP]
enabled = true
port = 8000
passwd-file = /etc/webiopi/passwd
doc-root = /home/pi/webiopi/examples/scripts/macros
welcome-file = index.html
```

- Блок [COAP] позволяет включить или отключить сервер CoAP, а также изменить значение порта:

```
[COAP]
enabled = true
port = 5683
multicast = true
```

- Блок [GPIO] позволяет установить пользовательские настройки и значения для портов GPIO при запуске WebIOPi. Например:

```
[GPIO]
21 = IN
23 = OUT 0
24 = OUT 0
25 = OUT 1
```

- Блок [~GPIO] позволяет установить пользовательские настройки и значения для портов GPIO при перезагрузке WebIOPi. Например:

```
[~GPIO]
21 = IN
23 = IN
24 = IN
25 = OUT 0
```

- Блок [SCRIPTS] определяет список скриптов, выполняемых при запуске WebIOPi. Например:

```
[SCRIPTS]
myscript = /home/pi/webiopi/examples/scripts/macros/script.py
```

- Блок [REST] позволяет с помощью REST API (см. *разд. 6.2.3*) ограничить доступ GET/POST к некоторым портам:

```
[REST]
gpio-export = 21, 23, 24, 25
```

```
gpio-post-value = false
gpio-post-function = false
device-mapping = false
```

- Блок `[DEVICES]` позволяет подключить устройства, поддерживаемые WebIOPi, к конкретным портам GPIO (список поддерживаемых WebIOPi устройств см. на странице: **https://code.google.com/p/webiopi/wiki/DEVICES**):

```
usb0 = Serial device:ttyUSB0 baudrate:9600
adc = MCP3008
dac = MCP4922 chip:1
gpio0 = MCP23017
gpio1 = MCP23017 slave:0x21
gpio2 = MCP23017 slave:0x22
pwm0 = PCA9685
pwm1 = PCA9685 slave:0x41
```

- Блок `[ROUTES]` определяет список маршрутов переадресации. Это позволяет при использовании REST API скрыть в адресной строке значение порта доступа или другое назначение, т. е. придать адресам удобный вид:

```
/bedroom/light = /GPIO/25/value
/bedroom/temperature = /devices/temp2/sensor/temperature/c
```

12.4. Javascript-библиотека webiopi.js

WebIOPi включает в себя HTTP-сервер, обеспечивающий как HTML-ресурсы, так и интерфейс REST API для управления выводами GPIO. При запуске сервера браузер сначала загружает HTML-файл с включенной Javascript-библиотекой webiopi.js, содержащей библиотеку jQuery для асинхронных вызовов к REST API. Этот метод очень эффективен, потому что не требует для обновления данных обновления страницы. Расширить возможности WebIOPi можно путем загрузки пользовательского сценария Python, содержащего функции настройки выводов GPIO и созданного с использованием Arduino-подобного синтаксиса (рис. 12.4).

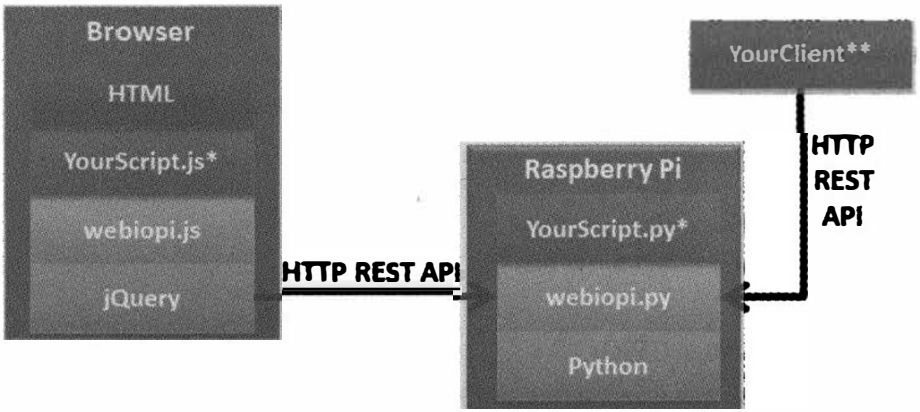


Рис. 12.4. Схема WebIOPi

Для подключения библиотеки webiopi.js на странице HTML в блок [HEADER] помещаем следующий код:

```
<script type="text/javascript" src="/webiopi.js"></script>
```

12.4.1. Функции библиотеки webiopi.js

Пояснение

Здесь и далее `webiopi()` **— возвращаемый объект WebIOPi.**

Функция *WebIOPi.ready*

Функция `WebIOPi.ready()` **регистрирует функцию обратного вызова при загрузке библиотеки webiopi.js.**

Синтаксис: `WebIOPi.ready(callback)`

Параметр: `callback` **— функция обратного вызова.**

Функция *WebIOPi.setFunction*

Функция `WebIOPi.setFunction()` **устанавливает назначение вывода GPIO.**

Синтаксис:

- `WebIOPi.setFunction(gpio, func)`
- `WebIOPi.setFunction(gpio, func, callback)`

Параметры:

- `gpio` **— номер вывода GPIO;**
- `func` **— назначение вывода:**
  - `IN` **— вывод конфигурирован как вход;**
  - `OUT` **— вывод конфигурирован как выход;**
  - `PWM` **— вывод конфигурирован как ШИМ-выход.**
- `callback` **— функция обратного вызова.**

Функция *WebIOPi.digitalWrite*

Функция `WebIOPi.digitalWrite()` **устанавливает цифровое значение вывода GPIO, сконфигурированного как выход (**`OUT`**).**

Синтаксис:

- `WebIOPi.digitalWrite(gpio, value)`
- `WebIOPi.digitalWrite(gpio, value, callback)`

Параметры:

- `gpio` **— номер вывода GPIO;**
- `value` **— значение для вывода (0 или 1);**
- `callback` **— функция обратного вызова.**

Функция *WebIOPi.digitalRead*

Функция `WebIOPi.digitalRead()` получает значение с вывода GPIO, сконфигурированного как вход (`IN`).

Синтаксис:

- `WebIOPi.digitalRead(gpio)`
- `WebIOPi.digitalRead(gpio, callback)`

Параметры:

- `gpio` — номер вывода GPIO;
- `callback` — функция обратного вызова.

Возвращаемое значение: `value` (0 или 1).

Функция *WebIOPi.toggleValue*

Функция `WebIOPi.toggleValue()` переключает значение на выводе GPIO на противоположное.

Синтаксис: `WebIOPi. toggleValue(gpio)`

Параметр: `gpio` — номер вывода GPIO.

Функция *WebIOPi.callMacro*

Функция `WebIOPi.callMacro()` выполняет макрофункцию на сервере WebIOPi. Макрос может объявляться в скрипте Python, запускаемом при старте WebIOPi.

Синтаксис:

- `WebIOPi.callMacro(macro)`
- `WebIOPi.callMacro(macro, args)`
- `WebIOPi.callMacro(macro, args, callback)`

Параметры:

- `macro` — наименование макрофункции;
- `args` — массив передаваемых аргументов для макрофункции;
- `callback` — функция обратного вызова.

Функция *WebIOPi.outputSequence*

Функция `WebIOPi.outputSequence()` отправляет последовательность битов на выход GPIO.

Синтаксис:

- `WebIOPi.outputSequence(gpio, period, sequence)`
- `WebIOPi.outputSequence(gpio, period, sequence, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `period` — время отправки (миллисекунд) одного бита;

- `sequence` — последовательность битов;
- `callback` — функция обратного вызова.

Пример:

```
var sequence = "01010100110011001100101010";
// отправка последовательности на gpio 7 с периодом 100 мсек
webiopi().outputSequence(7, 100, sequence, sequenceCallback);
```

Функция *WebIOPi.pulse*

Функция `WebIOPi.pulse()` отправляет импульс на выход GPIO.

Синтаксис:

- `WebIOPi.pulse(gpio)`
- `WebIOPi.pulse(gpio, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `callback` — функция обратного вызова.

Функция *WebIOPi.pulseRatio*

Функция `WebIOPi.pulseRatio()` отправляет ШИМ-сигналы на выход GPIO.

Синтаксис:

- `WebIOPi.pulseRatio (gpio, ratio)`
- `WebIOPi.pulseRatio (gpio, ratio, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `ratio` — значение для сигнала ШИМ (0,0–1,0);
- `callback` — функция обратного вызова.

Функция *WebIOPi.pulseAngle*

Функция `WebIOPi.pulseAngle()` отправляет ШИМ-сигналы на выход GPIO. Эта функция удобна при управлении сервоприводами для установки угла поворота рабочего органа от –45 до +45 градусов.

Синтаксис:

- `WebIOPi.pulseAngle (gpio, angle)`
- `WebIOPi.pulseAngle(gpio, angle, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `angle` — значение сигнала ШИМ (от –45 до +45);
- `callback` — функция обратного вызова.

Функция WebIOPi.createButton

Функция `WebIOPi.createButton()` возвращает объект — простую кнопку.

Синтаксис:

- `WebIOPi.createButton(id, label)`
- `WebIOPi.createButton(id, label, mousedown)`
- `WebIOPi.createButton(id, label, mousedown, mouseup)`

Параметры:

- `id` — идентификатор кнопки;
- `label` — надпись на кнопке;
- `mousedown` — функция, вызываемая при событии `mousedown` (нажатие по кнопке);
- `mouseup` — функция, вызываемая при событии `mouseup` (отпускание кнопки).

> **Примечание**
>
> События `mousedown` и `mouseup` в основном используются, когда идет нажатие на кнопку, перемещение ее, а потом мышь отпускается.

Чтобы кнопка появилась на странице, ее необходимо добавить — например, так:

```
button = webiopi().createButton("bt1", "button1", fun_down, fun_up);
$("#div1").append(button);
```

Функция WebIOPi.createFunctionButton

Функция `WebIOPi.createFunctionButton()` создает объект — кнопку, при нажатии на которую изменяется назначение вывода GPIO.

Синтаксис: `WebIOPi.createFunctionButton(gpio)`

Параметр: `gpio` — вывод, назначение которого изменяется (`IN`, `OUT`), при этом на кнопке выводится соответствующая надпись (**IN, OUT**).

Функция WebIOPi.createGPIOButton

Функция `WebIOPi.createGPIOButton()` создает объект — кнопку, при нажатии на которую изменяется значение (0 или 1) вывода GPIO.

Синтаксис: `WebIOPi.createGPIOButton(label, gpio)`

Параметры:

- `label` — надпись на кнопке;
- `gpio` — вывод, назначение которого изменяется (`0`, `1`), при этом на кнопке выводится соответствующая надпись (**0, 1**).

Функция WebIOPi.createMacroButton

Функция `WebIOPi.createMacroButton()` создает объект — кнопку, при нажатии на которую выполняется макрофункция на сервере (макрофункции прописываются в Python-скрипте, запускаемом при старте `webiopi`).

Синтаксис: `WebIOPi.createMacroButton(id, label, macro, args)`

Параметры:

- `id` — идентификатор кнопки;
- `label` — надпись на кнопке;
- `macro` — название макрофункции на сервере;
- `args` — список параметров, передаваемых макрофункции.

Функция *WebIOPi.createSequenceButton*

Функция `WebIOPi.createSequenceButton()` создает объект — кнопку, при нажатии на которую на вывод GPIO отправляется последовательность битов.

Синтаксис: `WebIOPi.createSequenceButton(id, label, gpio, period, sequence)`

Параметры:

- `id` — идентификатор кнопки;
- `label` — надпись на кнопке;
- `gpio` — вывод для отправки последовательности битов;
- `period` — время отправки одного бита (мсек);
- `sequence` — последовательность битов в виде строки.

Пример:

```
button = webiopi().createSequenceButton("but1", "label1", 25, 100,
"01010100110011001100101010");
$("#div1").append(button);
```

Функция *WebIOPi.createRatioSlider*

Функция `WebIOPi.createRatioSlider()` создает объект — шкалу (см. рис. 6.15), положением указателя на которой регулируется значение ШИМ-сигнала на выводе GPIO.

Синтаксис: `WebIOPi.createRatioSlider(gpio, ratio)`

Параметры:

- `gpio` — вывод для отправки последовательности битов;
- `ratio` — начальное значение для ШИМ (0,0–1,0).

Пример:

```
button = webiopi().createRatioSlider(24, 1.0);
$("#div1").append(button);
```

Функция *WebIOPi.createAngleSlider*

Функция `WebIOPi.createAngleSlider()` создает объект — шкалу, положением указателя которой регулируется значение ШИМ-сигнала, подаваемого на вывод GPIO при управлении сервоприводом для значений поворота угла рабочего органа от –45 до +45 градусов.

Синтаксис: `WebIOPi.createAngleSlider(gpio, angle)`

Параметры:

- `gpio` — вывод для отправки последовательности битов;
- `angle` — начальное значение угла для сигнала ШИМ (от –45 до +45).

Функция *WebIOPi.setLabel*

Функция `WebIOPi.setLabel()` изменяет надпись на кнопке.

Синтаксис: `WebIOPi.setLabel(id, label)`

Параметры:

- `id` — идентификатор кнопки;
- `lable` — новая надпись для кнопки.

В листинге 12.1 представлен пример кода страницы для формирования элементов управления `webiopi`, а на рис. 12.5 — вид этой HTML-страницы.

Листинг 12.1

```
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content = "height = device-height, width = 420,
                                                    user-scalable = no" />
        <title>WebIOPi | Demo</title>
        <script type="text/javascript" src="/webiopi.js"></script>
        <script type="text/javascript">
        webiopi().ready(function() {
        var content, button;
        content = $("#content");

        button = webiopi().createFunctionButton(25);
        content.append(button);

        button = webiopi().createGPIOButton(7, "SWITCH");
        content.append(button);

        button = webiopi().createSequenceButton("sos", "S.O.S 1", 25, 100,
                                                "01010100110011001100101010");
        content.append(button);

        button = webiopi().createMacroButton("macro", "Print Time", "PrintTime");
        content.append(button);

        button = webiopi().createAngleSlider(23, 30);
        content.append(button);
```

```
        button = webiopi().createRatioSlider(24, 0.5);
        content.append(button);

        webiopi().refreshGPIO(true);
        });

</head>
<body>
        <div id="content" align="center"></div>
</body>
</html>
```

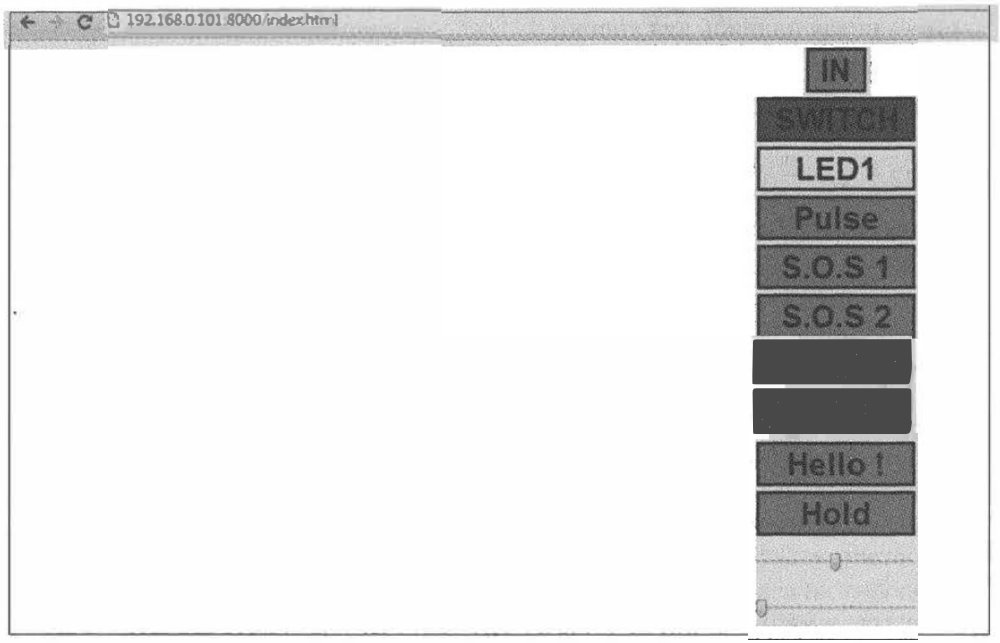


Рис. 12.5. Вид HTML-страницы с элементами управления `webiopi`

12.5. Проект управления веб-камерой на сервоприводах

Для создания своего проекта с использованием фреймворка WebIOPi необходимо реализовать следующие его компоненты:

- веб-интерфейс — HTML-страницу с использованием библиотеки webiopi.js;
- серверный скрипт на языке Python, запускаемый при старте `webiopi` для реализации функционала веб-интерфейса;
- файл конфигурации с установкой начальных значений портов.

Подготовим проект управления из веб-интерфейса камерой, укрепленной на подвесе, снабженном сервоприводами. Из веб-интерфейса можно будет поворачивать камеру, а также делать снимки и отправлять их на электронную почту.

Для проекта понадобятся следующие детали:

- подвес для камеры (рис. 12.6);
- сервоприводы TG9 (рис. 12.7) — 2 шт.;
- дополнительный блок питания 5 В;
- камера Raspberry Pi Camera Board (рис. 12.8).

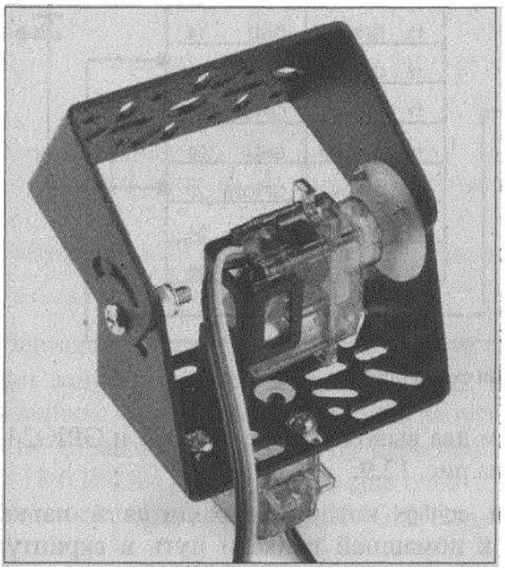


Рис. 12.6. Подвес для камеры с сереоприводами



Рис. 12.7. Сервопривод TG9

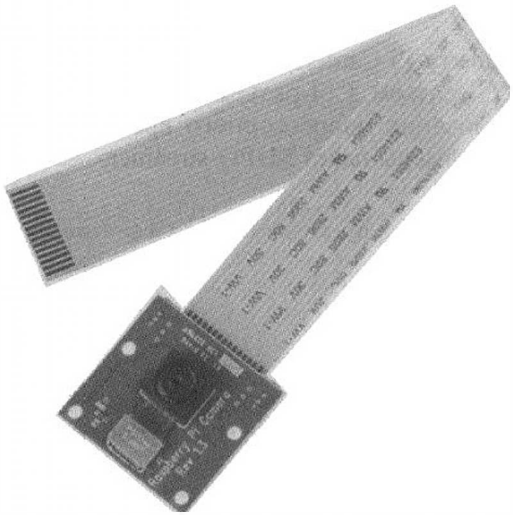


Рис. 12.8. Камера Raspberry Pi Camera Board

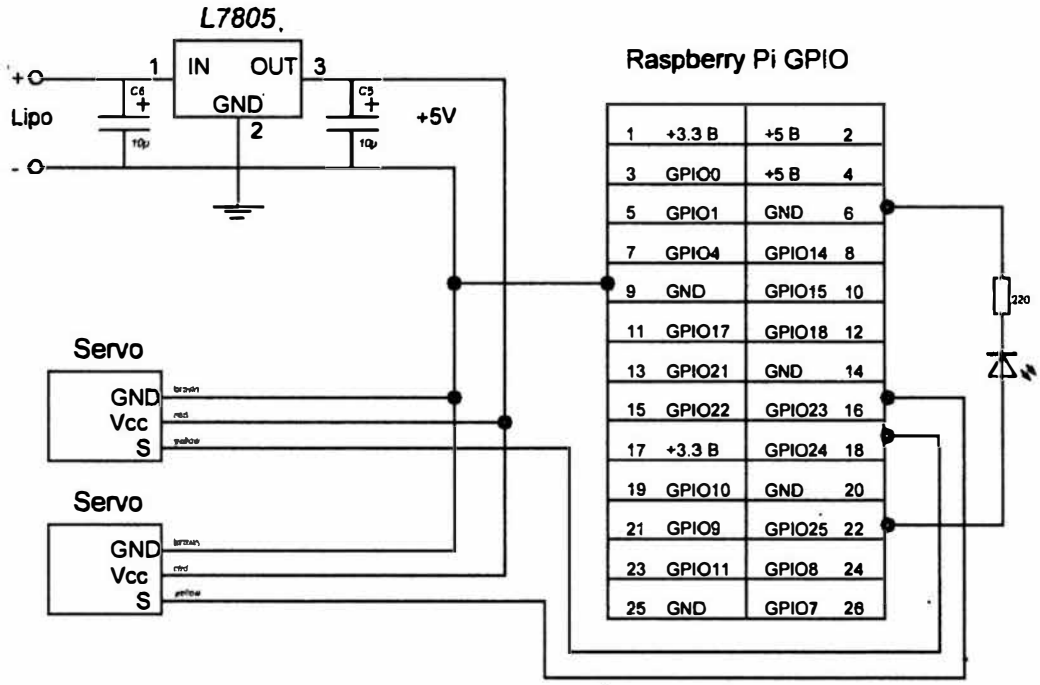


Рис. 12.9. Электрическая схема проекта

Для управления сервоприводами задействуем два вывода GPIO: GPIO23 и GPIO24. Электрическая схема проекта представлена на рис. 12.9.

Сначала настроим в файле конфигурации config, который находится в папке **/etc/webiopi/**, необходимые параметры: путь к домашней папке и путь к скрипту, выполняемому при запуске `webiopi`:

```
doc-root = /home/pi/webiopi/examples/servo-camera
script = /home/pi/webiopi/examples/servo-camera/camera.py
```

Поскольку камера Raspberry Pi Camera Board подключена напрямую к графическому процессору через CSI-разъем, CSI-разъем на плате, запись и кодирование видео происходят без использования процессорного времени. Установим для передачи потокового видео с этой камеры пакет MJPG-streamer:

```
su pi
https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer/ mjpg-streamer
cd mjpg-streamer/mjpg-streamer
make USE_LIBV4L2=true clean all
sudo su
make DESTDIR=/usr install
cp -R www /var/
```

Напишем запускающий камеру sh-скрипт **stream_start.sh** (листинг 12.2), создав для него предварительно файл:

```
sudo nano /usr/local/bin/stream_start.sh
sudo chmod +x /usr/local/bin/stream_start.sh
```

Листинг 12.2

```
#!/bin/bash
if [ -d /tmp/stream ];then
        echo "/tmp/stream already created"
else
        mkdir /tmp/stream
fi

if [ -f /tmp/stream/pic.jpg ];then
        echo "raspistill already running"
else
        raspistill -w 320 -h 240 -q 5 -o /home/pi/webiopi/examples/
                                        servo-camera/pic.jpg -tl 100 -t 9999999&
fi
mjpg_streamer -i "input_file.so -f /home/pi/webiopi/examples/servo-camera"
-o "output_http.so -w /home/pi/webiopi/examples/servo-camera"
```

Запуск скрипта осуществляется командой:

```
/usr/local/bin/stream_start.sh
```

Напишем на языке Python серверный скрипт camera.py (листинг 12.3), который будет запускаться при запуске `webiopi`. **В скрипте пропишем назначение контактов, макросы и выполним начальный запуск камеры потокового видео. Учтем при этом, что управляющие контакты сервоприводов подключены к выводам 16 (GPIO23) и 18 (GPIO24) GPIO. Плюс еще одна дополнительная опция — мигание светодиода, подключенного к выводу 22 (GPIO25).**

Листинг 12.3

```
import webiopi
import sys
import time
from subprocess import call

GPIO = webiopi.GPIO

SERVO1  = 23
SERVO2  = 24
LED1    = 25

def camera_start():
    return_code = call("/usr/share/webiopi/examples/servo-camera/
                                                stream_start.sh", shell=True)

def setup():
    webiopi.debug("Blink script - Setup")
```

```
    # Установка выводов GPIO
    GPIO.setFunction(SERVO1, GPIO.PWM)
    GPIO.setFunction(SERVO1, GPIO.PWM)
    GPIO.setFunction(LED1, GPIO.OUT)
    GPIO.pwmWriteAngle(SERVO1, 0)    # set to 0 (neutral)
    GPIO.pwmWriteAngle(SERVO2, 0)    # set to 0 (neutral)
    GPIO.digitalWrite(LED1, GPIO.HIGH)
    camera_start()

# Цикл loop()
def loop():
    # Toggle LED each 5 seconds
    value = not GPIO.digitalRead(LED1)
    GPIO.digitalWrite(LED1, value)
    webiopi.sleep(5)

# Возврат выводов в начальное состояние
def destroy():
    webiopi.debug("Blink script - Destroy")
    # Reset GPIO functions
    GPIO.setFunction(SWITCH, GPIO.IN)
    GPIO.setFunction(SERVO, GPIO.IN)
    GPIO.setFunction(LED0, GPIO.IN)
    GPIO.setFunction(LED1, GPIO.IN)
```

Веб-интерфейс у нас будет достаточно простой: два слайдера управления подвесом камеры, кнопка для отправки фото на e-mail и картинка изображения с камеры (рис. 12.10). При изменении позиции слайдера подвес поворачивается в направле-

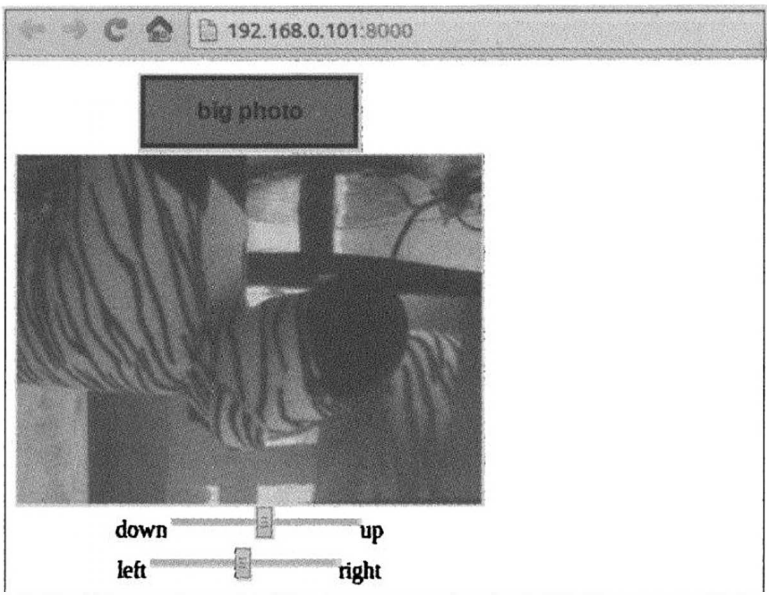


Рис. 12.10. HTML-страница проекта управления подвесом для камеры

нии вверх-вниз и влево-вправо. Изображение с камеры постоянно обновляется запуском функции set_Timeout(). Необходимо также запретить кэширование страницы. Код HTML-страницы camera.html представлен в листинге 12.4.

Листинг 12.4

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta http-equiv = "cache-control" content = "max-age=0">
        <meta http-equiv = "cache-control" content = "no-cache">
        <meta http-equiv = "expires" content = "0">
        <meta http-equiv = "expires" content = "Tue, 01 Jan 1970 1:00:00 GMT">
        <meta http-equiv = "pragma" content = "no-cache">
        <meta name="viewport" content = "height = device-height, width = 420,
                                                    user-scalable = no" />
        <title>WebIOPi camera</title>
        <script type="text/javascript" src="/webiopi.js"></script>
        <script type="text/javascript">
        function init() {
                var button;

                button = webiopi().createButton("photo", "big photo", photo);
                $("#cam").append(button);

                button = webiopi().createAngleSlider(23, 30);
                $("#updown").append(button);


                button = webiopi().createAngleSlider(24, 0);
                $("#leftright").append(button);
                }

        function photo() {
                $("#ph").html('');
                $("#ph").html('<img src="pic.jpg">');
                }
        function get_img()
          {
          document.getElementById("img1").src="http://192.168.0.101:8080/pic.jpg";
          }
        webiopi().ready(init);

        </script>
        <style type="text/css">
```

```
            button {
                    margin: 5px 5px 5px 5px;
                    width: 150px;
                    height: 50px;
                    font-size: 12pt;
                    font-weight: bold;
                    color: black;
            }
        </style>
</head>
<body onload='setInterval("get_img();",1000);'>
        <div style="float:left">
        <div id="content" align="center">
                <div id="cam"></div>
                <div id="vid"><img id="imgl" width="320" height="240"
                                src="http://192.168.0.101:8080/pic.jpg"></div>
                <div>down<span id="updown"></span>up</div>

                <div>left<span id="leftright"></span>right</div>
        </div>
        </div>
        <div style="float:right" id="ph"></div>
</body>
</html>
```

Электронный архив

Коды листингов этого проекта вы найдете в папке python\servo-camera сопровождающего книгу электронного архива.

12.6. WebIOPi — подключение устройств

Фреймворк WebIOPi позволяет использовать устройства типов Serial, I^2C, SPI и 1-Wire прямо из интерфейса REST API, без написания макросов (список поддерживаемых датчиков имеется на странице: **https://code.google.com/p/webiopi/wiki/DEVICES**).

Включить поддержку устройства можно в блоке `[DEVICE]` файла конфигурации /etc/webiopi/config, раскомментировав строку с выбранным датчиком, который следует подключить к выходам GPIO:

```
[DEVICES]
...
#temp0 = TMP102
#temp1 = TMP102 slave:0x49
temp2 = DS18B20
#temp3 = DS18B20 slave:28-0000049bc218
```

Пример схемы подключения датчика температуры DS18B20 представлен на рис. 12.11.

Выполнив необходимые монтажные манипуляции, открываем стартовую страницу WebIOPi (см. рис. 12.1), выбираем пункт **Device Monitor** и попадаем на страницу подключенных устройств (рис. 12.12). В нашем случае здесь отображен только датчик температуры DS18B20. Страница демонстрирует данные, получаемые с датчика, с обновлением их каждые 5 секунд.

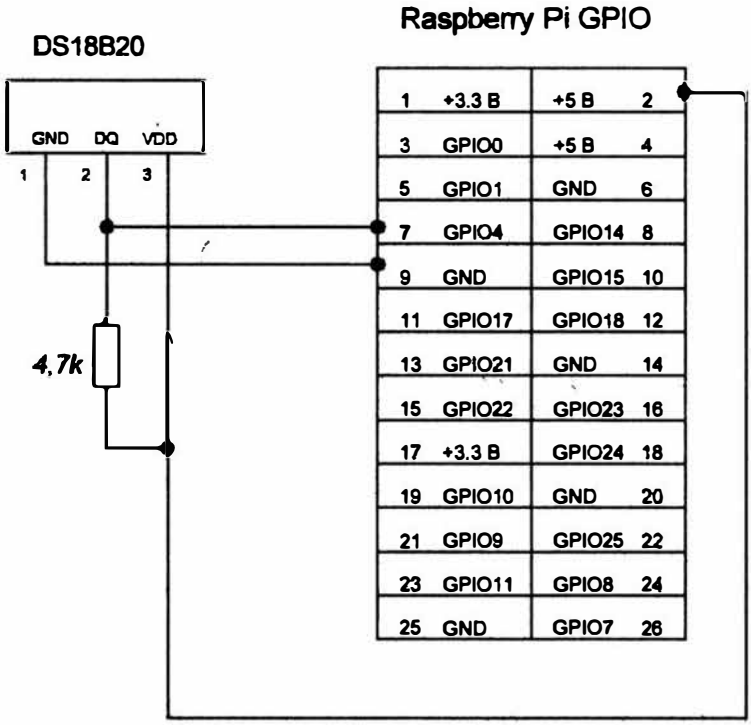


Рис. 12.11. Схема подключения к GPIO датчика DS18B20

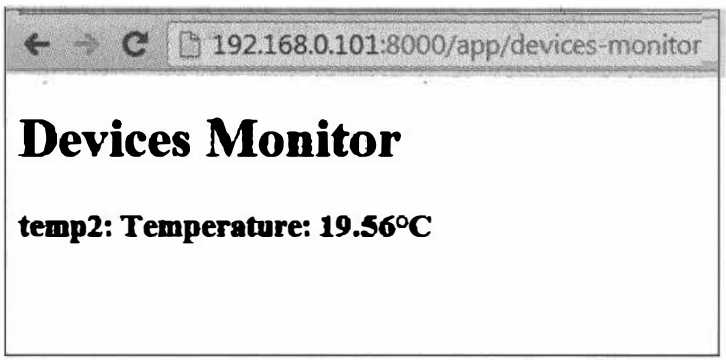


Рис. 12.12. Монитор подключенных устройств

12.7. Доступ к устройству из сервиса Weaved

Сервис Weaved (**https://www.weaved.com**) позволяет подключиться к Raspberry Pi (и не только) из браузера, находясь в любой точке мира. Если вы не установили программное обеспечение для сервиса Weaved при установке WebIOPi, сейчас для этого самое время.

12.7.1. Установка сервиса Weaved

Для установки сервиса сначала необходимо в нем зарегистрироваться. Сделать это можно на странице **https://developer.weaved.com/portal/** (рис. 12.13), открываемой по нажатию кнопки **Sign Up** на главной странице сервиса.

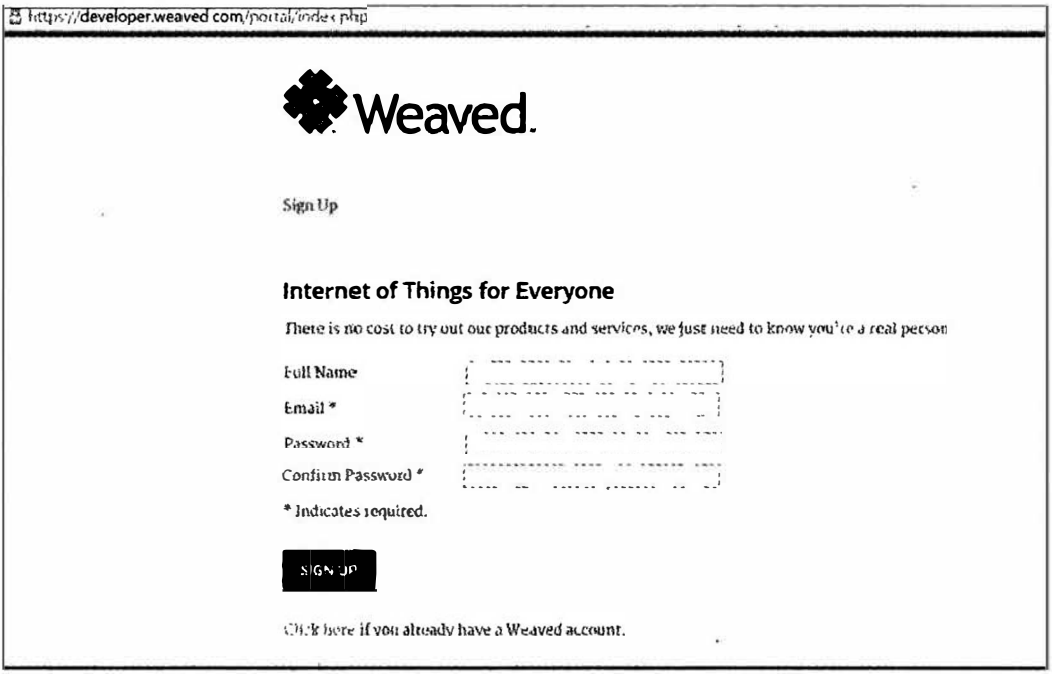


Рис. 12.13. Регистрация в сервисе Weaved

Зарегистрировавшись в сервисе, запускаем терминал нашего Raspberry Pi и скачиваем установщик программного обеспечения для сервиса Weaved:

```
wget https://github.com/weaved/installer/raw/master/binaries/
weaved-nixinstaller_1.2.13.bin
```

Делаем файл установщика исполняемым:

```
chmod +x weaved-nixinstaller_1.2.13.bin
```

Запускаем установку:

```
./weaved-nixinstaller_1.2.13.bin
```

Во время установки вам будет предложено выбрать вид доступа для связи с Raspberry Pi (рис. 12.14):

- SSH на порту 22;
- Web (HTTP) на порту 80;
- WebIOPi на порту 8000;
- VNC на порту 5901 (tested with tightvncserver);
- пользовательский сервис TCP.

Выбрать можно только один из предложенных. Если вы хотите использовать несколько видов доступа, необходимо запускать установку несколько раз, выбирая каждый раз новый вид доступа.

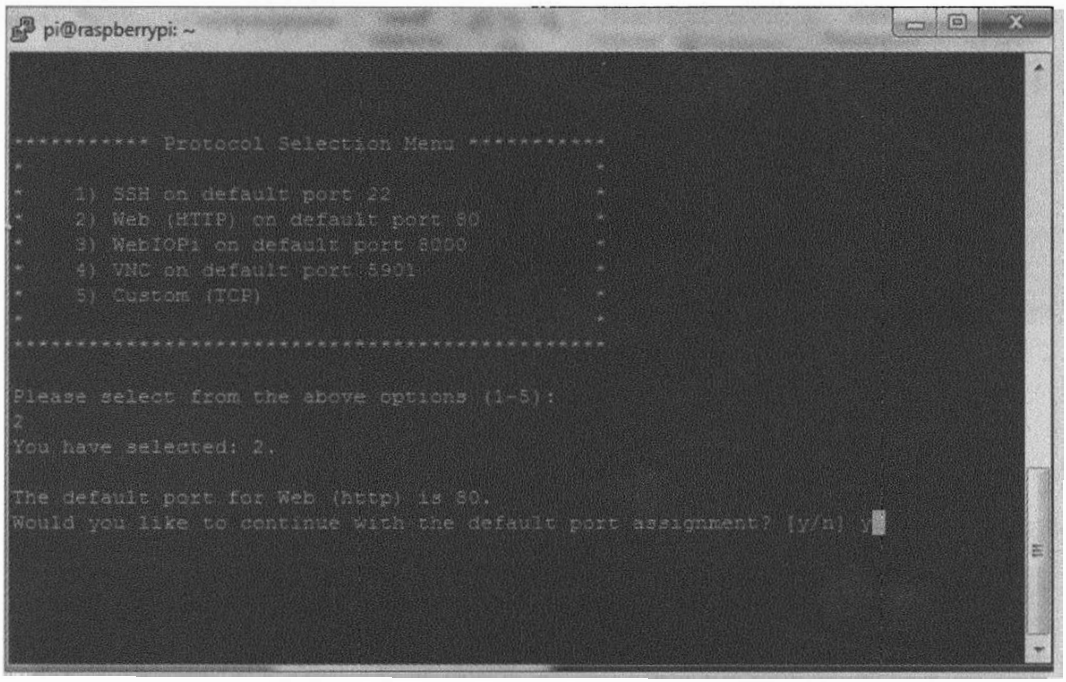


Рис. 12.14. Выбор вида доступа к плате из сервиса Weaved: здесь выбрано подключение по Web (HTTP)

Далее необходимо ввести данные для авторизации в сервисе Weaved: логин (адрес электронной почты) и пароль (рис. 12.15).

После чего создать имя для устройства в сервисе Weaved (рис. 12.16).

В завершение установки выдается информация по командам для запуска/остановки сервиса (рис. 12.17).

Завершив установку, мы можем зайти в свой профиль в сервисе Weaved и увидеть список наших устройств (рис. 12.18).

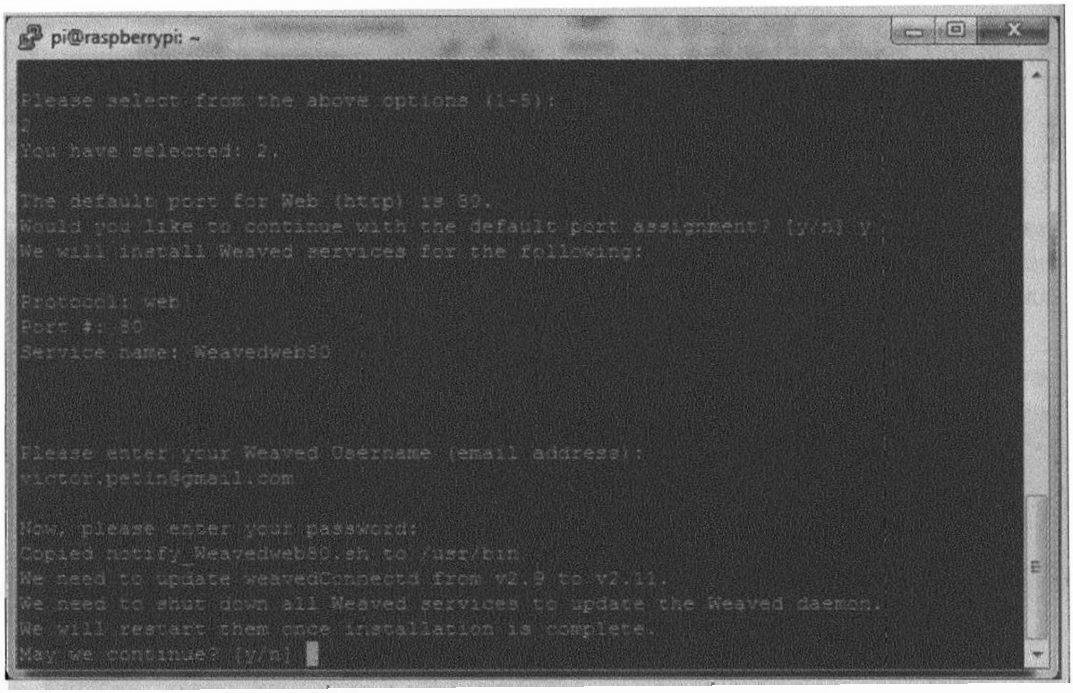


Рис. 12.15. Ввод данных для авторизации в сервисе Weaved

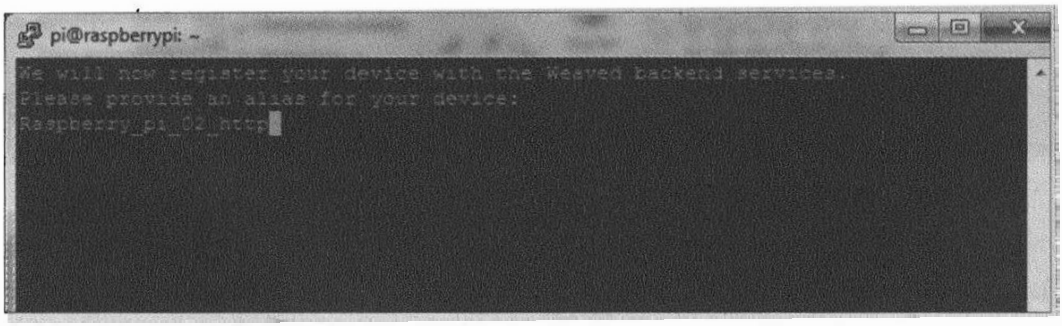


Рис. 12.16. Создание имени для устройства в сервисе Weaved

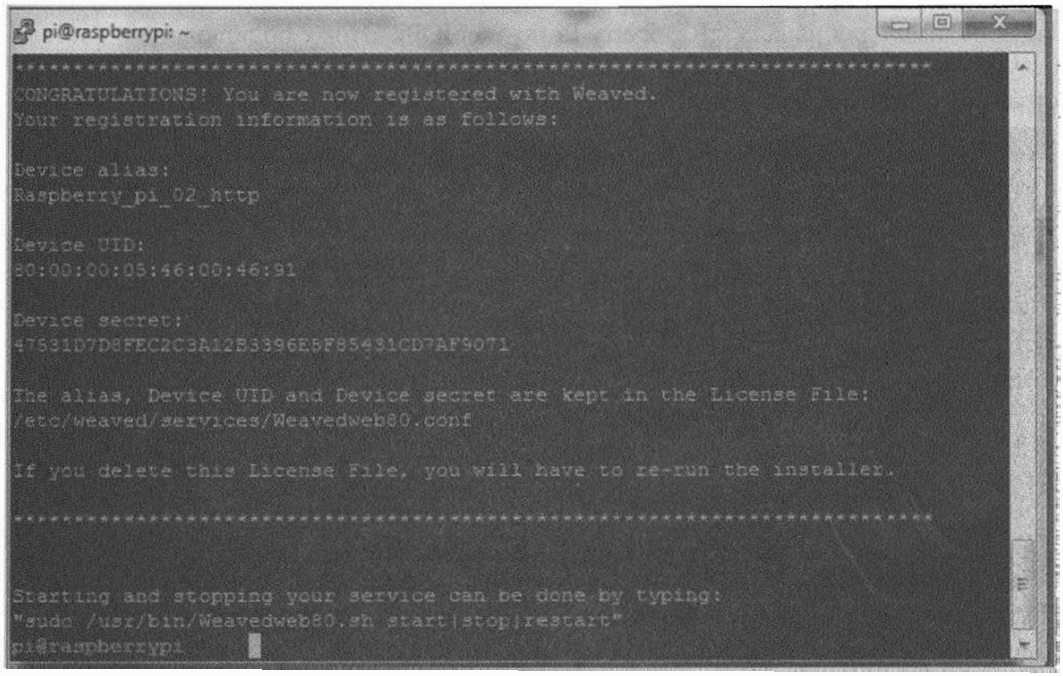


Рис. 12.17. Информация по командам запуска сервисов для доступа из Weaved

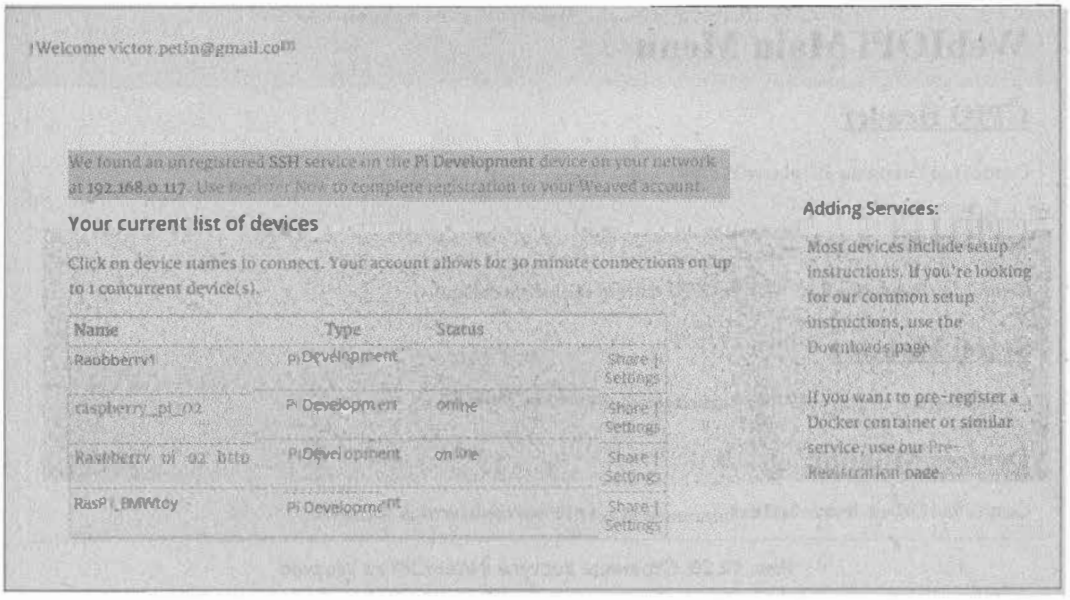


Рис. 12.18. Список устройств в сервисе Weaved

12.7.2. Подключение к Raspberry Pi в сервисе Weaved

Подключимся для начала к Raspberry Pi по WebIOPi. Этому подключению соответствует устройство **raspberry_pi_02** (см. рис. 12.18) — выбираем его и попадаем на страницу авторизации сервиса WebIOPi нашего Raspberry Pi (рис. 12.19).

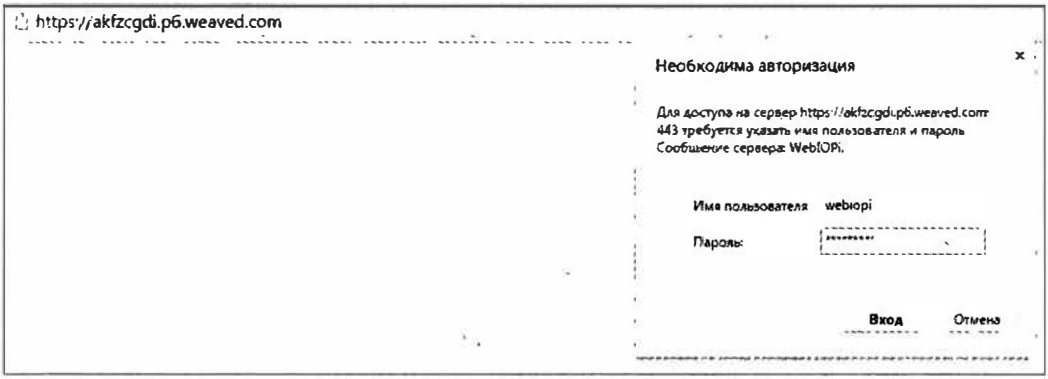


Рис. 12.19. Страница доступа к WebIOPi из Weaved

После авторизации — мы на главной странице WebIOPi нашего Raspberry Pi (рис. 12.20).

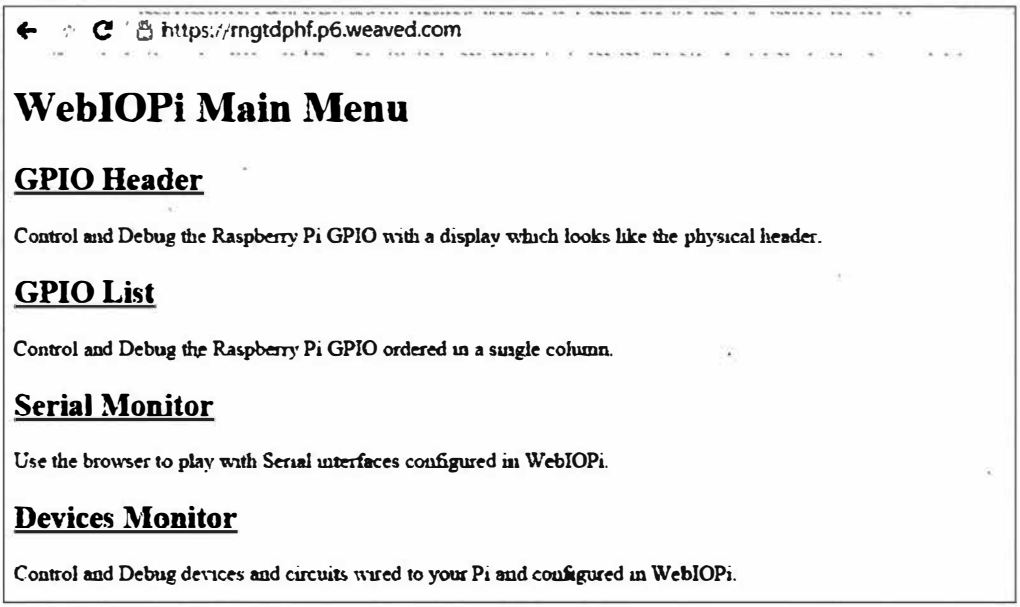


Рис. 12.20. Страница доступа к WebIOPi из Weaved

Подключимся теперь к Raspberry Pi по Web. Этому подключению соответствует устройство **raspberry_pi_02_http** (см. рис. 12.18). Для такого подключения на Raspberry Pi необходимо установить веб-сервер:

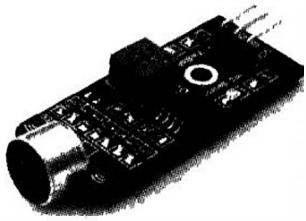
```
sudo apt-get update
sudo apt-get install apache2
```

Страница доступа из Weaved к веб-серверу на Raspberry Pi показана на рис. 12.21.



Рис. 12.21. Страница доступа к веб-серверу на Raspberry Pi из Weaved

ГЛАВА 13



Проект Wyliodrin: управление удаленными устройствами из браузера

Румынский онлайн-сервис Wyliodrin (**https://www.wyliodrin.com/**), основанный на графической среде программирования Blockly, предоставляет возможность графического программирования сенсоров/актуаторов на Intel Galileo, Intel Edison, Beagle Bone Black, Raspberry Pi, Arduino и еще целом ряде устройств (рис. 13.1). С помощью Wyliodrin можно программировать устройства, загружать в них программы и управлять этими устройствами прямо из браузера.

Для программированиия встраиваемых устройств необходимо, как правило, чтобы они были подключены к компьютеру. Однако при работе через Wyliodrin устройство должно быть подключено только к Интернету. А раз так, то и находиться оно может в любом месте, — вы можете запрограммировать его, просто войдя в свою учетную запись (аккаунт) Wyliodrin. Таким образом, если у вас есть, например, метеостанция или удаленные устройства автоматизации, вы можете программировать их и управлять ими непосредственно из своего дома или офиса. Сервис позволяет также строить графики для отображения данных, получаемых с устройств.

Программирование встраиваемых устройств обычно требует знания языков C или C++. Работая же через Wyliodrin, вы можете выбрать язык, который хорошо знаете и любите: от тех же C/C++ до Java, Pascal, Shell Script, Perl, PHP, Objective-C, C#, Python или JavaScript. И даже если вы не знаете никакого языка программирования, то сможете воспользоваться визуальной средой программирования, где построение программы осуществляется простым перетаскиванием блоков.

Чтобы начать пользоваться сервисом, необходимо зарегистрироваться, после чего вы сможете выбрать тип учетной записи: бесплатный (использовать одно устройство и создать три приложения) или платный (от 3,9 до 50 долларов в месяц). Для владельцев плат Intel Galileo действует специальное предложение — годовая подписка, позволяющая бесплатно использовать три платы и создать 15 приложений.

Итак, регистрируемся в сервисе и попадаем в свой профиль. Сейчас наш профиль пуст — какая-либо информация о подключенных устройствах и созданных прило-



Рис. 13.1. Главная страница проекта Wyliodrin

жениях в нем отсутствует. Так что, прежде всего нам надо добавить в него устройство.

13.1. Добавление устройства в профиль

Для добавления устройства нажимаем в окне профиля кнопку **Add New Board** и в открывшемся окне **New Board** (рис. 13.2) вводим название устройства (в поле

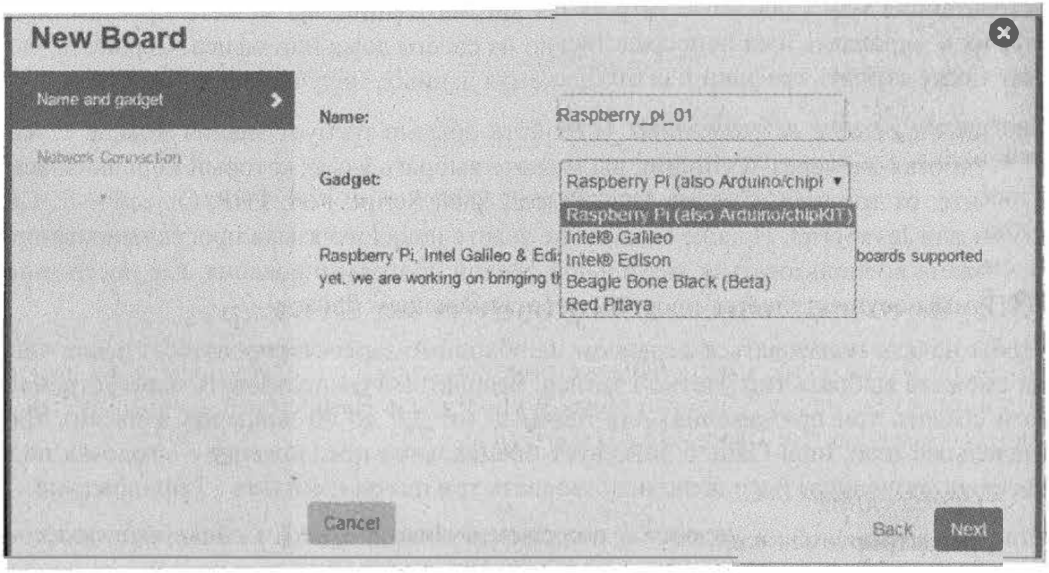


Рис. 13.2. Добавление устройства

Name), выбираем плату, с которой собираемся работать (сейчас это Raspberry Pi), и нажимаем кнопку **Next**. В следующем окне (рис. 13.3) выбираем тип подключения устройства к сети Интернет и нажимаем кнопку **Submit** — все, устройство добавлено в профиль (рис. 13.4).

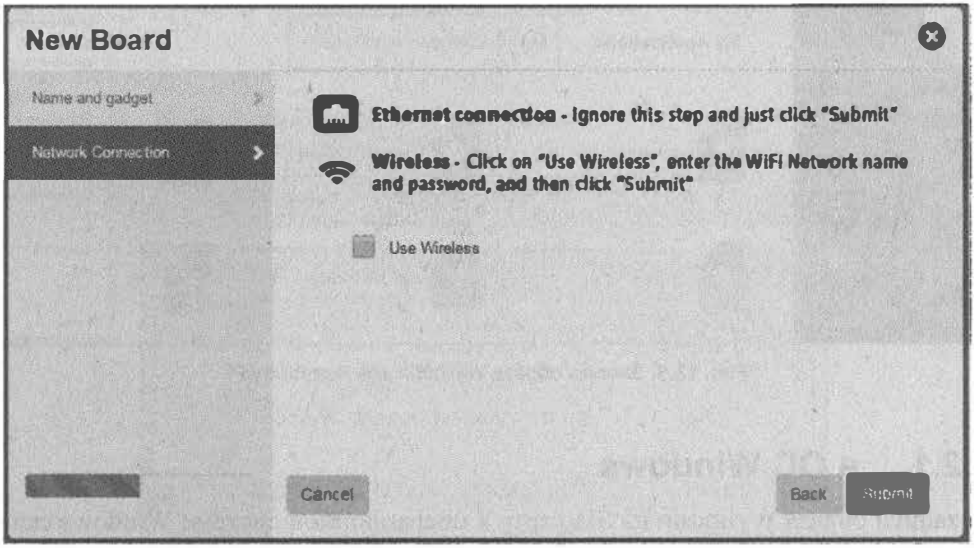


Рис. 13.3. Выбор настроек подключения устройства к Интернету



Рис. 13.4. Профиль пользователя с добавленными устройствами

13.2. Запись образа Wyliodrin на SD-карту...

Теперь необходимо настроить наше устройство, для чего следует прежде всего скачать и загрузить на карту памяти образ Wyliodrin для Raspberry Pi, выбрав из выпадающего меню опцию **Download SD Card** (рис. 13.5).

Примечание

Нам понадобится карта MicroSD емкостью минимум 4 Гбайт (рекомендуемый класс — не ниже 10).

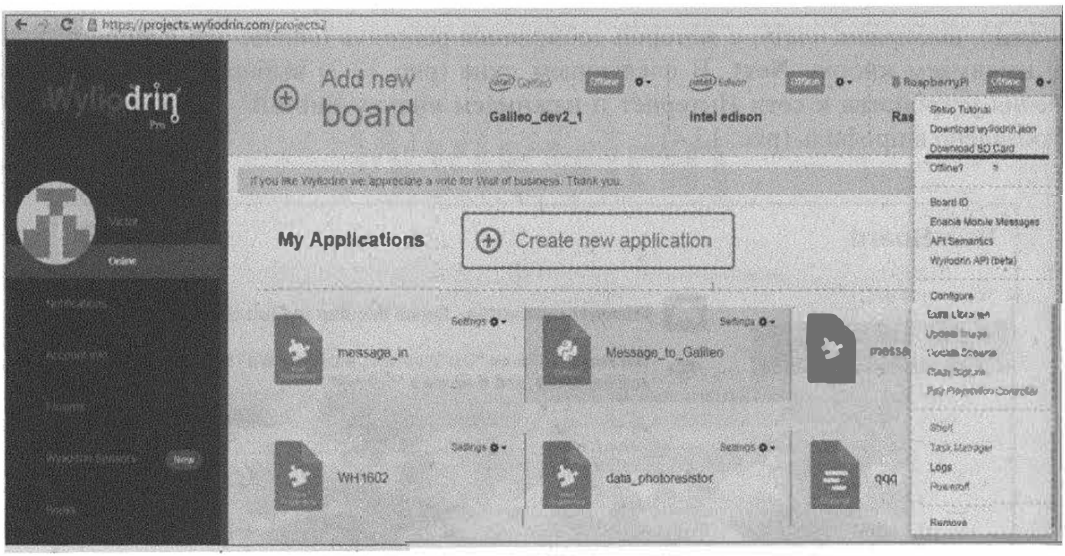


Рис. 13.5. Закачка образа Wyliodrin для Raspberry Pi

13.2.1. ...в ОС Windows

Для записи образа Wyliodrin на SD-карту в операционной системе Windows скачайте и установите на компьютер программу Win32 Disk Imager (инсталлятор программы находится в свободном доступе по ссылке **http://win32-disk-imager.ru.uptodown.com/**).

Запустив установленную программу, выберите устройство (картридер), где находится ваша SD-карта (рис. 13.6), и, нажав на значок папки у поля **Image File**, укажите путь к файлу со скачанным и распакованным образом Wyliodrin (рис. 13.7).

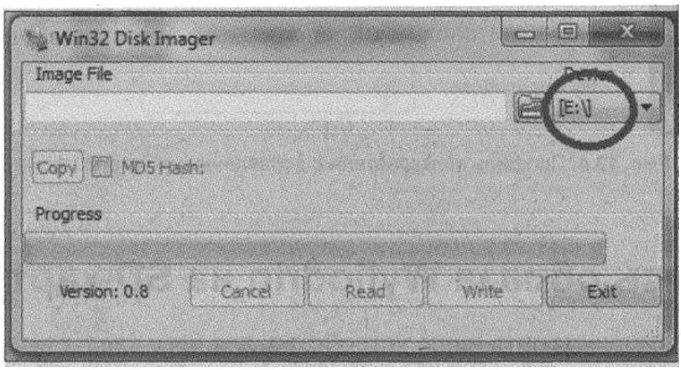


Рис. 13.6. Выбор местонахождения SD-карты

Вернувшись после этого в окно программы Win32 Disk Imager, нажмите кнопку **Write** — начнется процесс записи образа Wyliodrin на карту (рис. 13.8), по завершении которого нажмите кнопку **Exit** и закройте приложение. Теперь ваша карта MicroSD готова для использования на Raspberry Pi.

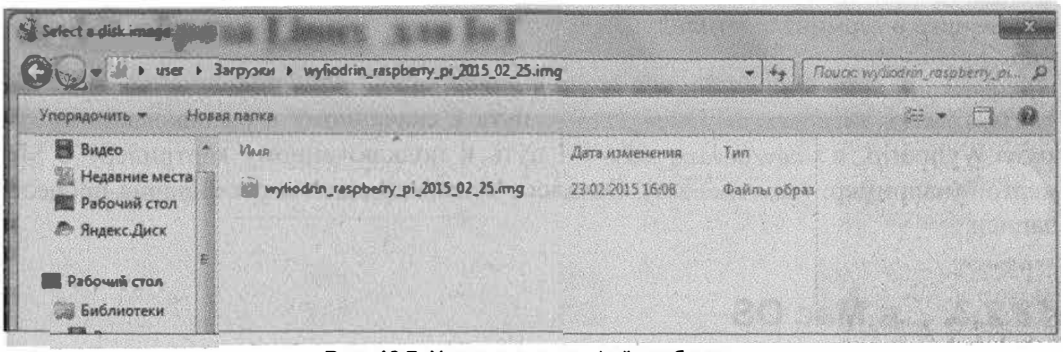


Рис. 13.7. Указание пути к файлу образа

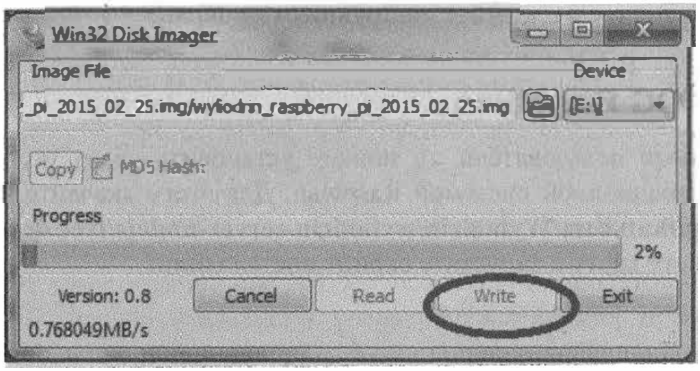


Рис. 13.8. Запись образа на SD-карту

13.2.2. ...в ОС Linux

Для записи образа Wyliodrin на SD-карту в операционной системе Linux вставьте SD-карту в картридер — карта должна определиться в системе (в данном случае — под именем **mmcblk0**) (рис. 13.9).

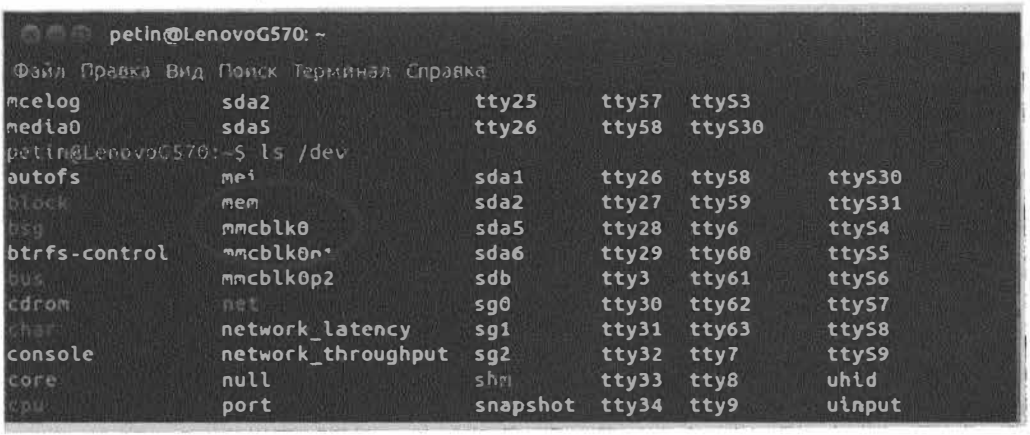


Рис. 13.9. Определение SD-карты в списке устройств /dev

Выполните в терминале команду:

```
dd if=wyliodrin_raspberrypi_image_file of=/dev/device_name
```

где *wyliodrin_raspberrypi_image_file* — путь к скачанному и распакованному образу Wyliodrin, а */dev/device_name* — путь к подключенному картридеру с SD-картой (например: `/dev/mmcblk0`). Осталось только дождаться завершения процесса записи.

13.2.3. ...в Mac OS

Для записи образа на SD-карту в операционной системе Mac OS X скачайте утилиту PiWriter (**http://sourceforge.net/projects/piwriter/**), вставьте SD-карту в картридер, запустите утилиту и следуйте инструкциям на экране.

13.2.4. ...в ОС Raspbian

Если вы опытный пользователь, то можете установить образ Wyliodrin сразу на SD-карту с операционной системой Raspbian. Для этого скачайте из репозитория архив **https://github.com/Wyliodrin/wyliodrin-server-nodejs/tree/development**, распакуйте его на эту карту и запустите скрипт инсталляции:

```
./wyliodrin-server-raspberry-pi.sh
```

Если используется беспроводное соединение, необходимо изменить файл **/etc/network/interfaces**, записав в него код из листинга 13.1.

Листинг 13.1

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
wpa-roam /home/pi/wyliodrin-server-nodejs/conf/wireless/wireless.conf
iface default inet dhcp
```

13.3. Запись на SD-карту настроек Wyliodrin

Для каждой новой платы система генерирует уникальный файл настроек wyliodrin.json — скачайте из своего профиля этот файл, выбрав из выпадающего меню опцию **Download wyliodrin.json** (рис. 13.10), и запишите его на ту же карту MicroSD, куда уже был записан образ Wyliodrin (рис. 13.11).

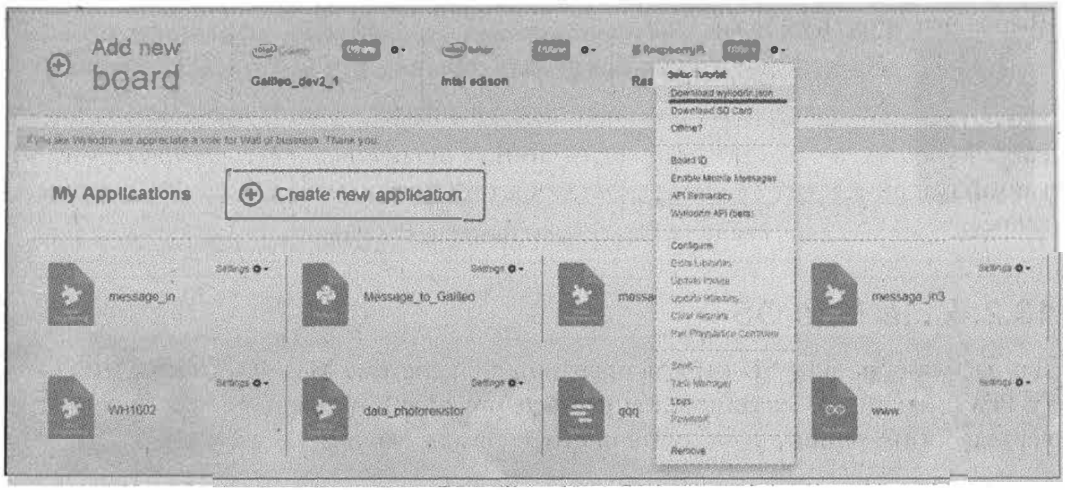


Рис. 13.10. Закачка файла настроек wyliodrin.json

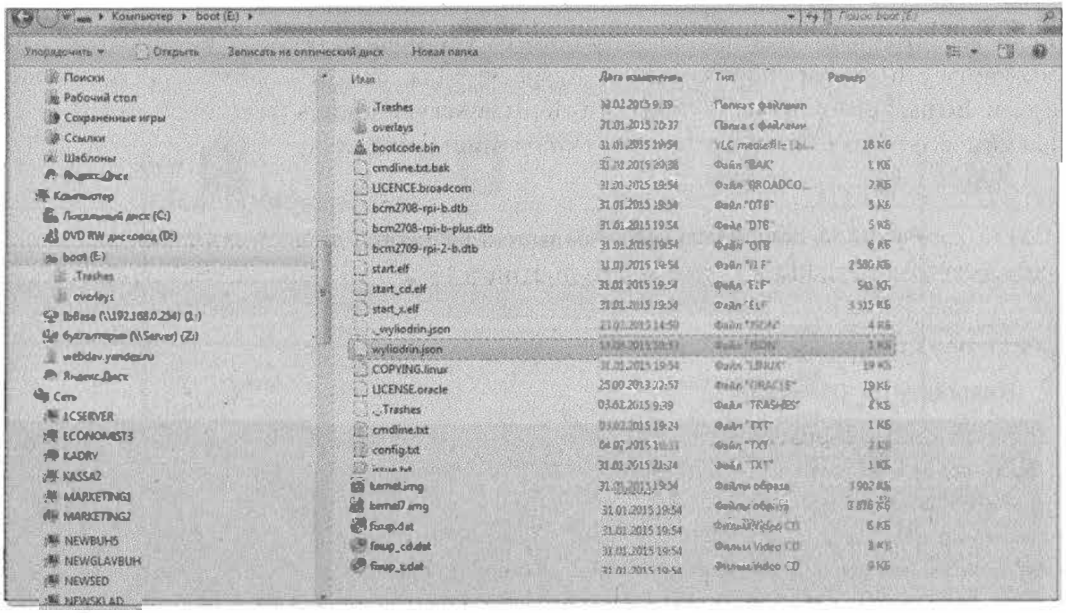


Рис. 13.11. Запись файла настроек на карту MicroSD

13.4. Подключение Raspberry Pi к Wyliodrin

Вставьте подготовленную SD-карту в Raspberry Pi и загрузитесь с нее. Если плата получила доступ в Интернет, то после загрузки системы в профиле Wyliodrin ее статус изменится на **Online** (рис. 13.12).

Из веб-интерфейса Wyliodrin можно произвести удаленную загрузку библиотек (опция **Extra Libraries**) и обновление (опция **Update Image**) системы (рис. 13.13), причем протекание этих процессов вам будет наглядно показано (рис. 13.14).

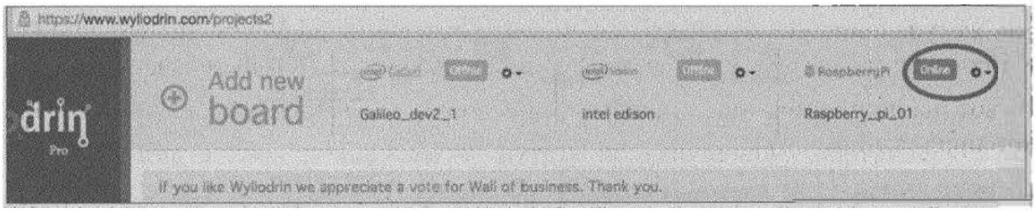


Рис 13.12. Подключение Raspberry Pi к Wyliodrin

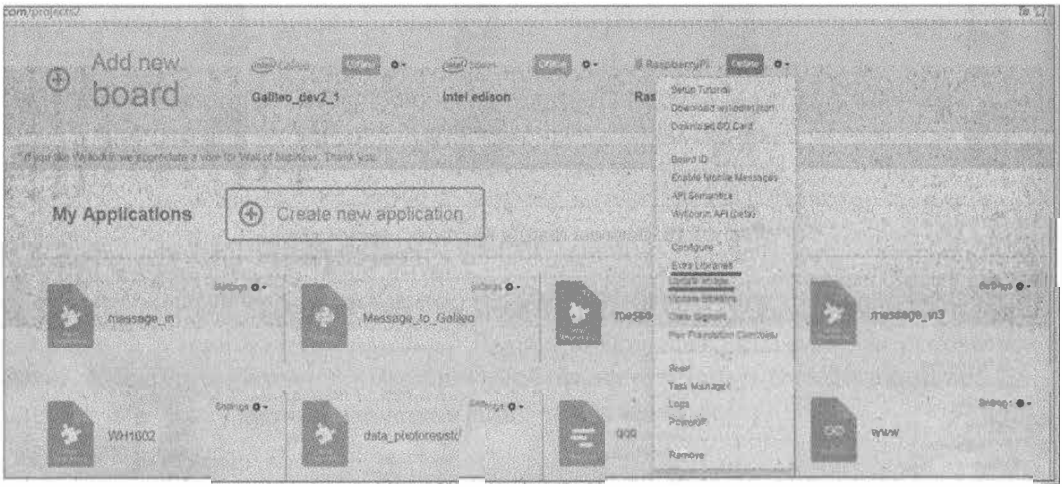


Рис. 13.13. Возможность удаленной загрузки библиотек и обновления системы

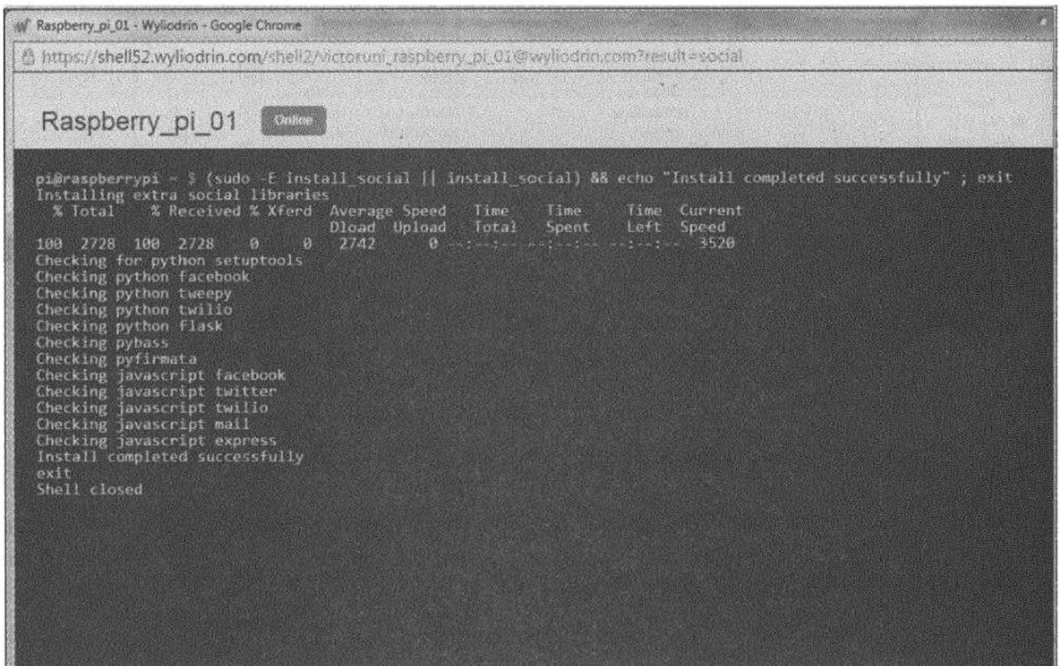


Рис. 13.14. Процесс удаленной загрузки библиотек на плату Raspberry Pi

Так же удаленно (из браузера) вы можете запустить оболочку shell (рис. 13.15) и диспетчер задач (рис. 13.16), а также просто выключить Raspberry Pi.

Ну вот, наша плата готова к программированию, и мы можем приступить к созданию приложений.

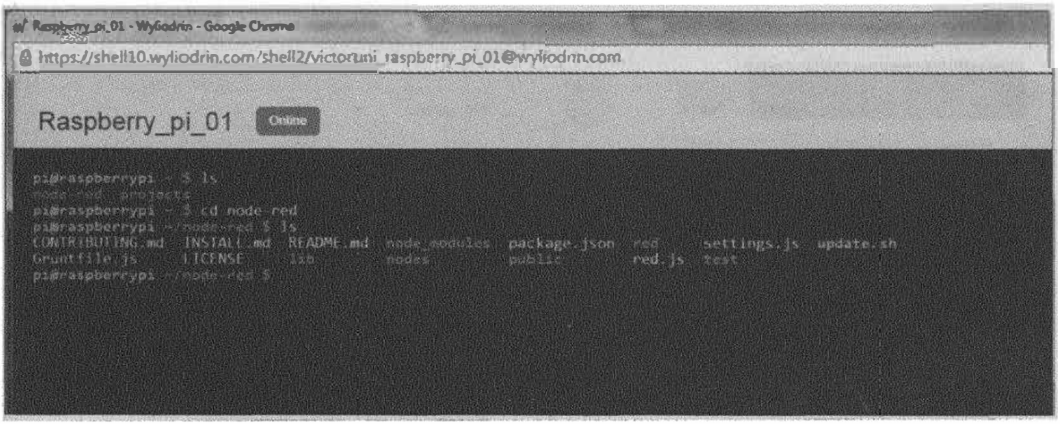


Рис. 13.15. Удаленный запуск оболочки shell

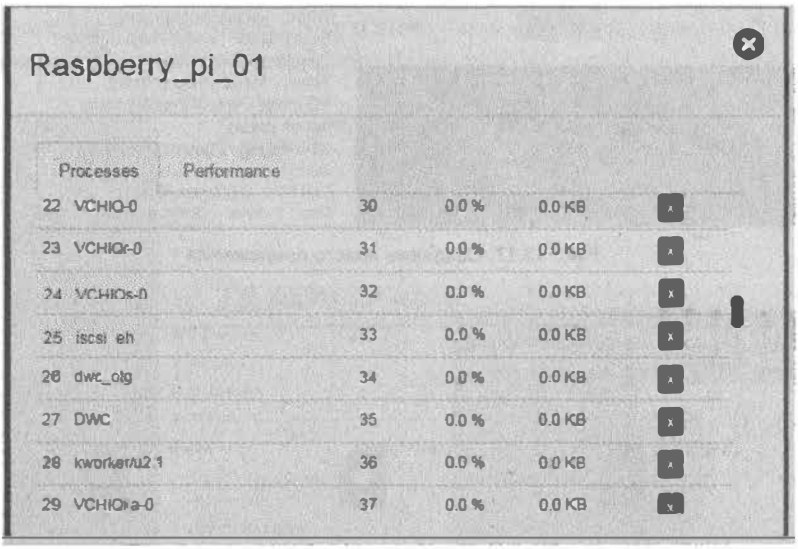


Рис. 13.16. Удаленный запуск диспетчера задач

13.5. Создание приложения в графической среде программирования

Для создания нового приложения в окне профиля (см. рис. 13.13) нажмите кнопку **Create new application**, после чего в открывшемся окне **New Project** введите название нового приложения, его описание и выберите язык программирования.

В списке языков программирования **Programming Language** уже имеется несколько готовых шаблонов приложений от Wyliodrin — выберите в качестве первого приложения: **Led Blink - Visual Programming** (рис. 13.17). Подтвердите свой выбор, и наше новое приложение создано (рис. 13.18).

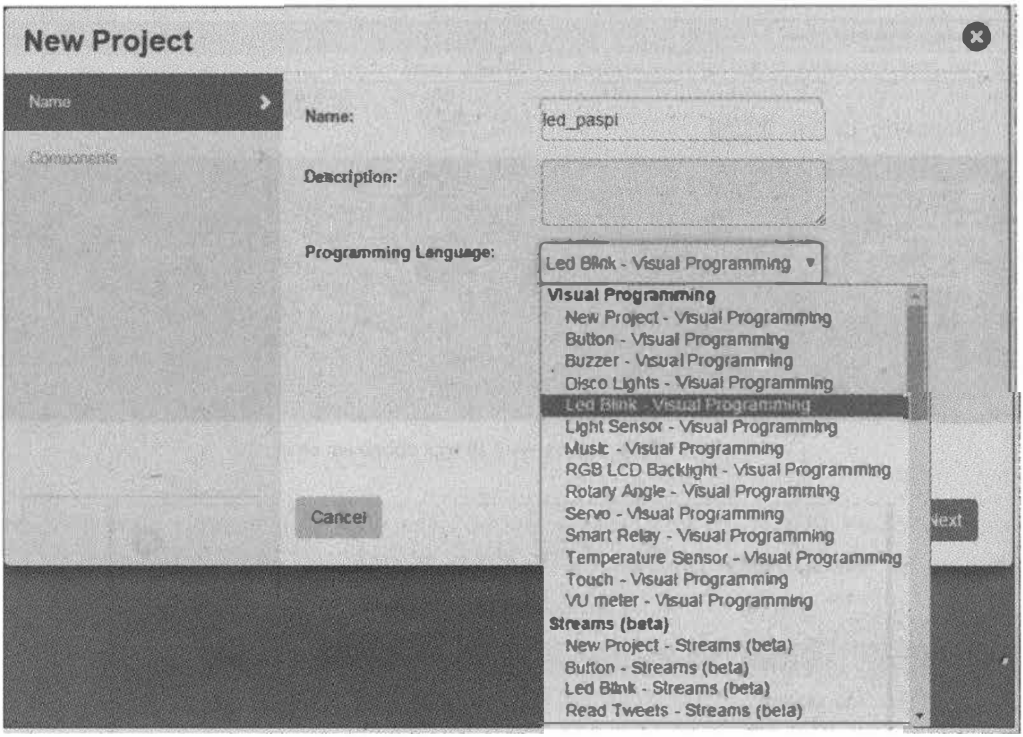


Рис. 13.17. Создание нового приложения

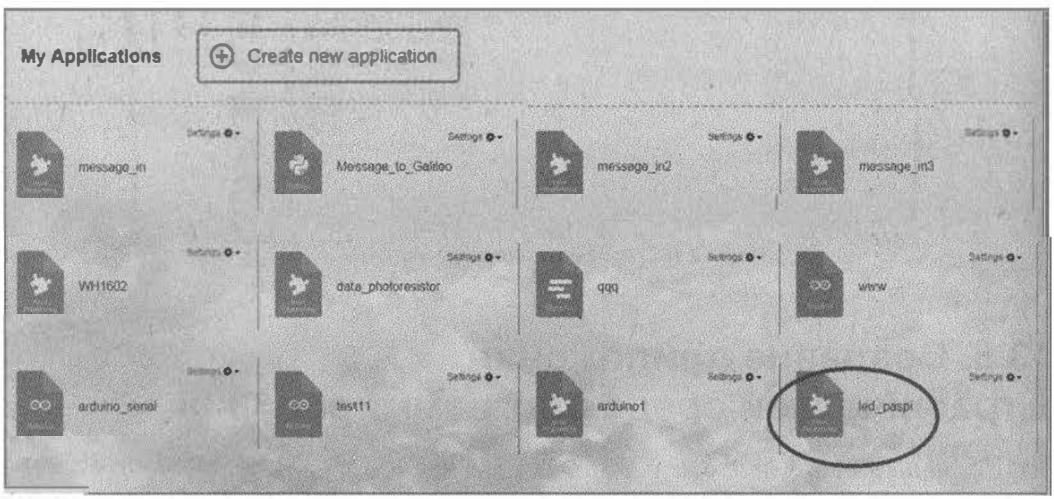


Рис. 13.18. Список приложений профиля

Выбрав приложение в списке приложений (см. рис. 13.18), мы попадем в окно его редактирования (рис. 13.19) — здесь можно посмотреть, как выглядит код приложения на языках Python и JavaScript, внести в этот код какие-либо изменения, а также загрузить приложение на плату и запустить на выполнение.

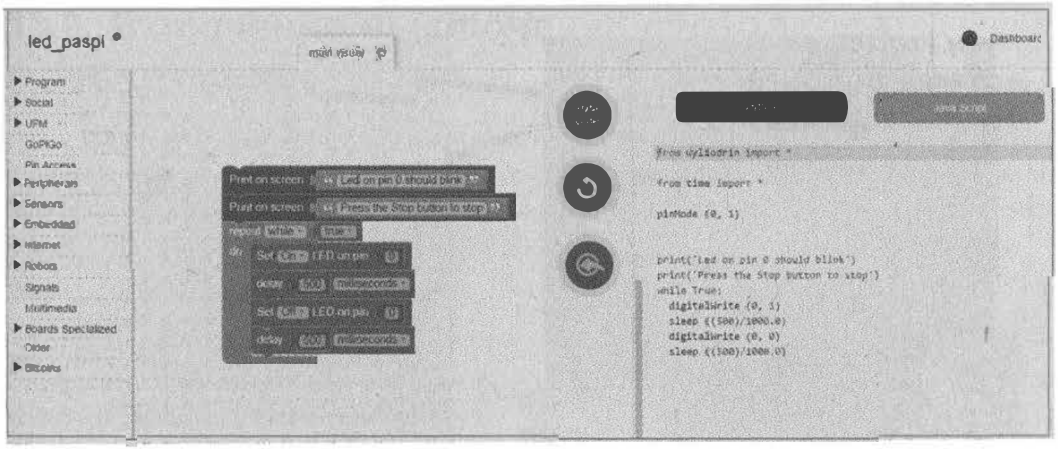


Рис. 13.19. Окно редактирования приложения

Немного подредактируем наше приложение: изменим в визуальных блоках значения **pin** на вывод 7 и увеличим задержки **delay** до 2000 мс (рис. 13.20).

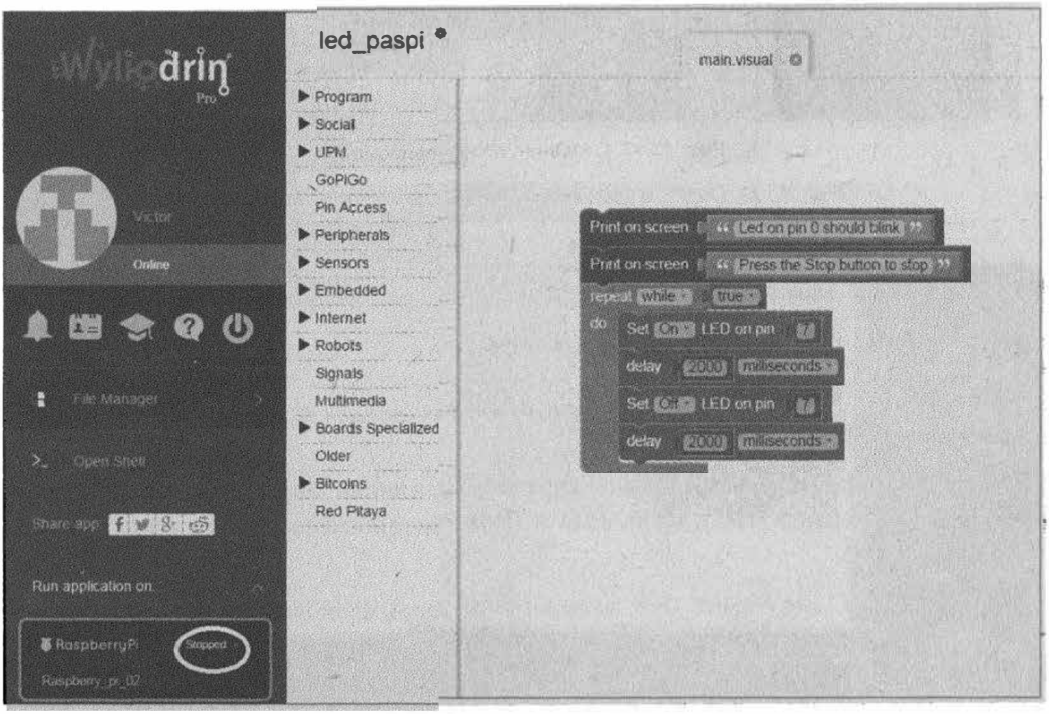


Рис. 13.20. Редактирование приложения

Теперь подключим к Raspberry Pi светодиод согласно схеме, приведенной на рис. 13.21, и нажмем в окне приложения на надпись **Stopped** (выделена кружком на рис. 13.20). Система проверит код и, если все в порядке, осуществит загрузку приложения на плату и запуск его на выполнение (рис. 13.22).

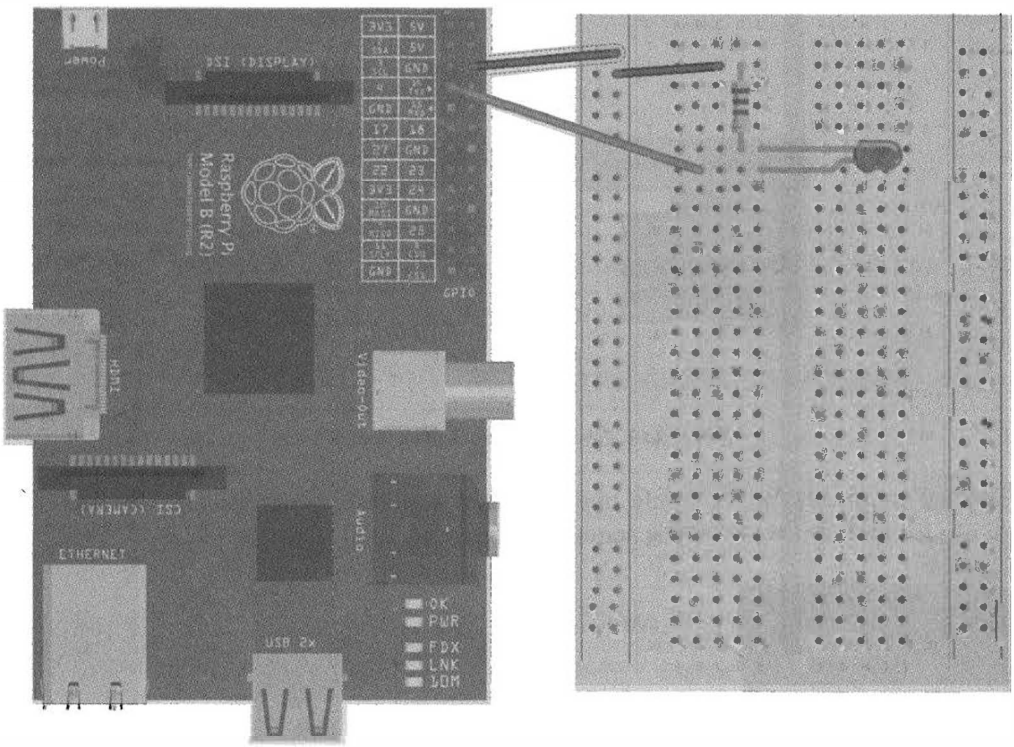


Рис. 13.21. Схема подключения светодиода к плате Raspberry Pi



Рис. 13.22. Запуск приложения

В это время на плате Raspberry Pi мы можем наблюдать мигание светодиода, подключенного к выводу 7. Нажатие кнопки **Stop** в правом верхнем углу окна приложения (рис. 13.22) приведет к остановке приложения на плате.

13.6. Включение/выключение светодиода с веб-страницы

Для управления светодиодом с веб-страницы создадим новый проект **web_raspi** и в визуальном редакторе впишем код, представленный на рис. 13.23. Запустим на Raspberry Pi веб-сервер (порт 5000). При попадании на главную страницу браузер выдает код:

```
<br><a href='/on'>On</a>
<br><a href='/off'>Off</a>
```

По нажатию на ссылку **On** светодиод, подключенный к выводу 7, зажигается, по нажатию на ссылку **Off** — гаснет.

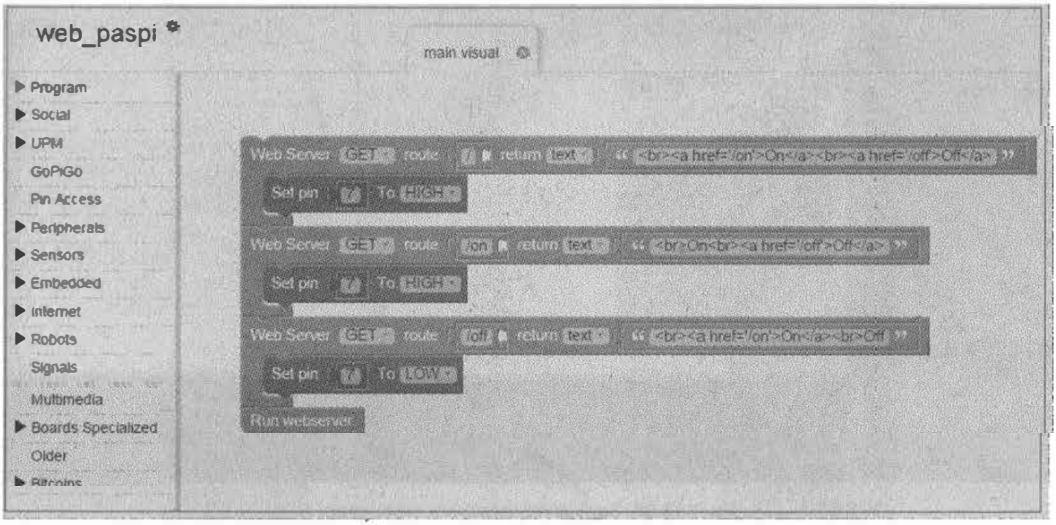


Рис. 13.23. Код приложения **web_raspi** в визуальном редакторе

Теперь надо узнать адрес нашего веб-сервера, т. е. IP-адрес Raspberry Pi, — запустим для этого из браузера оболочку shell и выполним в ней команду `ip addr show` (рис. 13.24).

Далее запускаем в Wyliodrin наше приложение **web_raspi** (рис. 13.25), набираем в браузере адрес **http://ip_raspberry:5000** и управляем включением/выключением светодиода переходом по ссылкам (рис. 13.26).

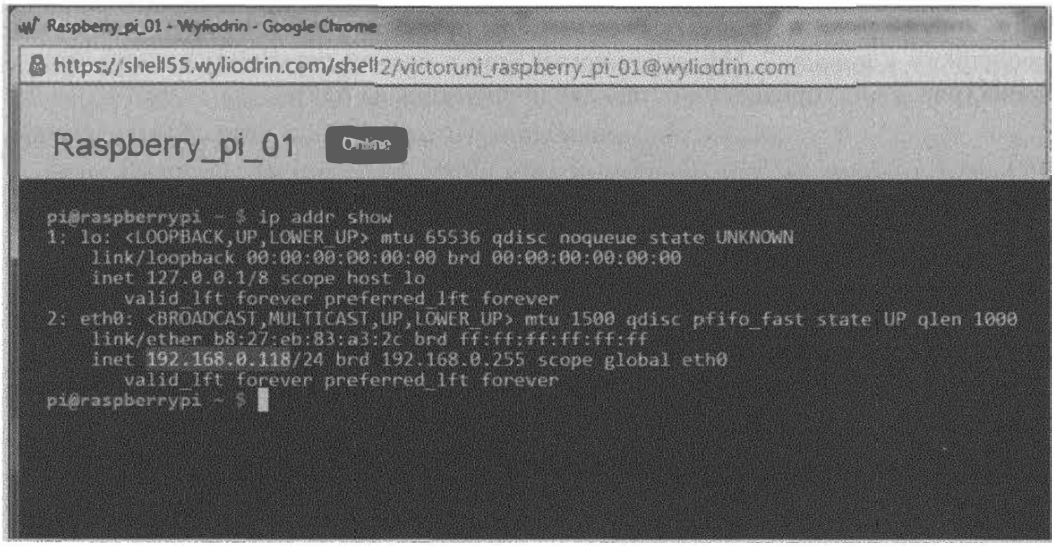


Рис. 13.24. Получение IP-адреса Raspberry Pi из командной оболочки shell

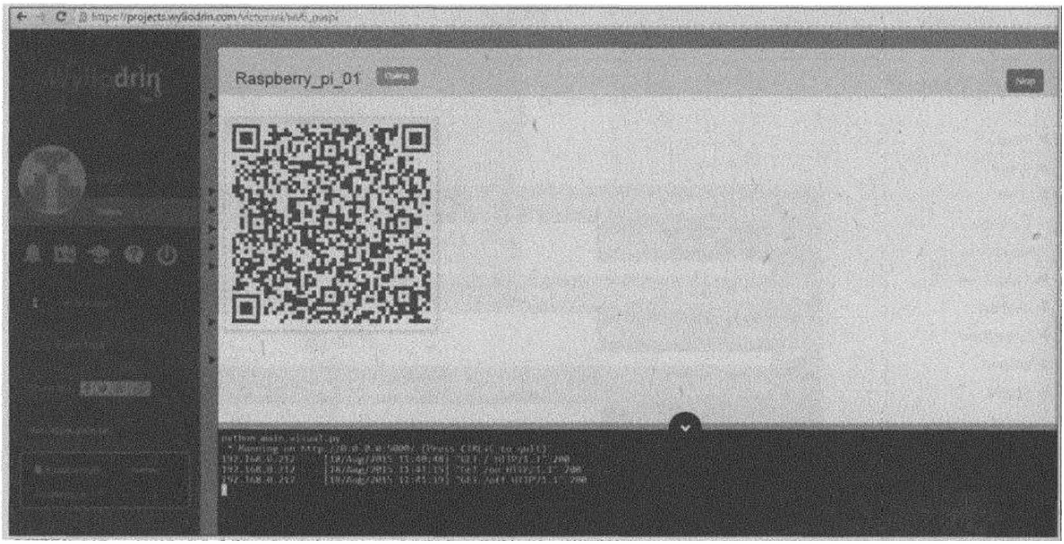


Рис. 13.25. Запуск приложения **web_raspi**

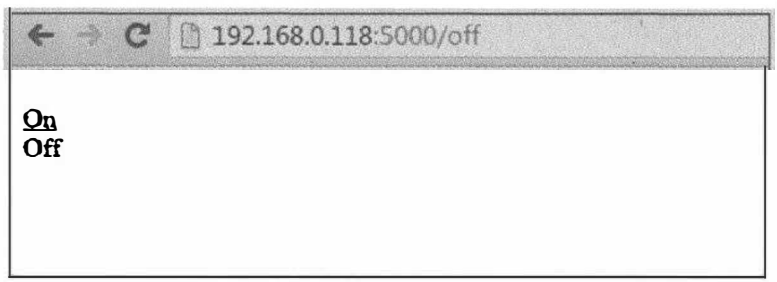


Рис. 13.26. Веб-страница управления включением/выключением светодиода

13.7. Подключение платы Arduino к сервису Wyliodrin...

Плату Arduino невозможно напрямую использовать для подключения к сервису Wyliodrin. Но программное обеспечение Wyliodrin позволяет использовать возможности платы Arduino, подключенной к микрокомпьютеру Raspberry Pi. Рассмотрим здесь соответствующие примеры.

13.7.1. ...с помощью библиотеки Firmata

Сначала загрузим на плату Arduino скетч StandartFirmata из Arduino-библиотеки Firmata (рис. 13.27), входящей в состав Arduino IDE. Библиотека Firmata реализует протокол Firmata, упрощающий общение с программами на компьютере.

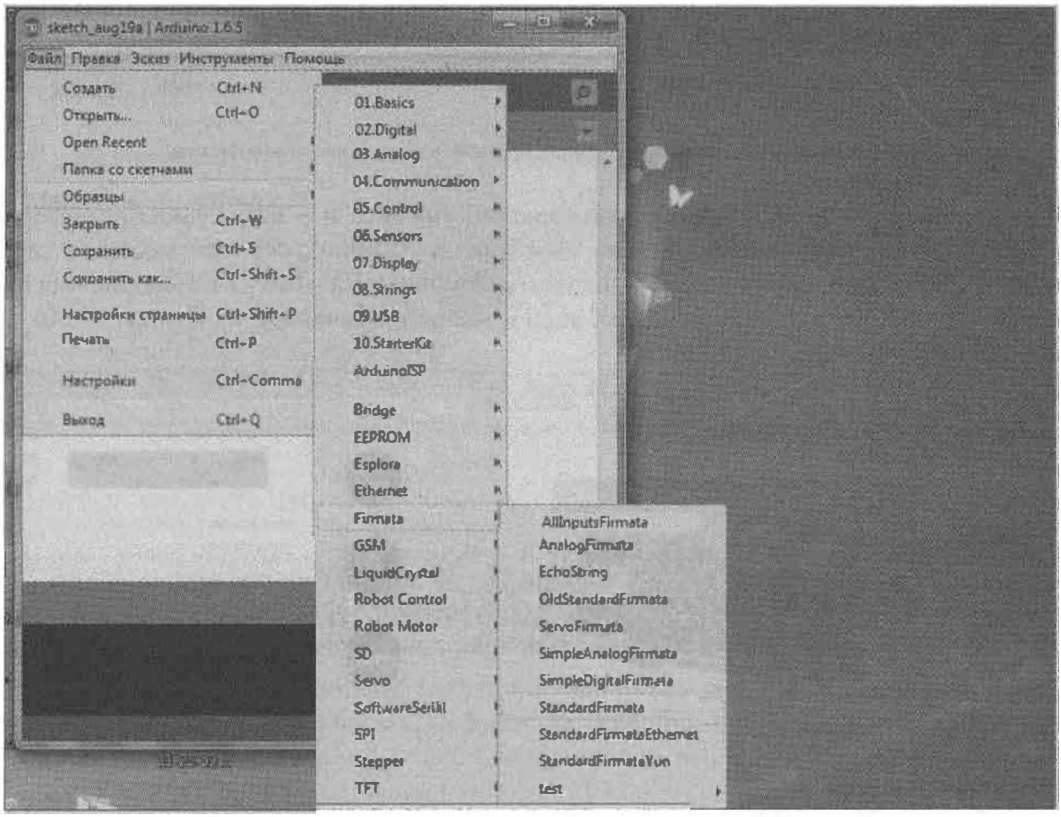


Рис. 13.27. Загрузка на плату Arduino скетча StandartFirmata из библиотеки Firmata

Подсоединим теперь плату Arduino к нашему Raspberry Pi, создадим в своем профиле Wyliodrin новый проект **arduino1** и выберем в качестве языка программирования **VisualProgramming**. На следующем шаге выберем подключение платы Arduino: **Connect an Arduino to the board (Raspberry Pi only)** — как показано на рис. 13.28.

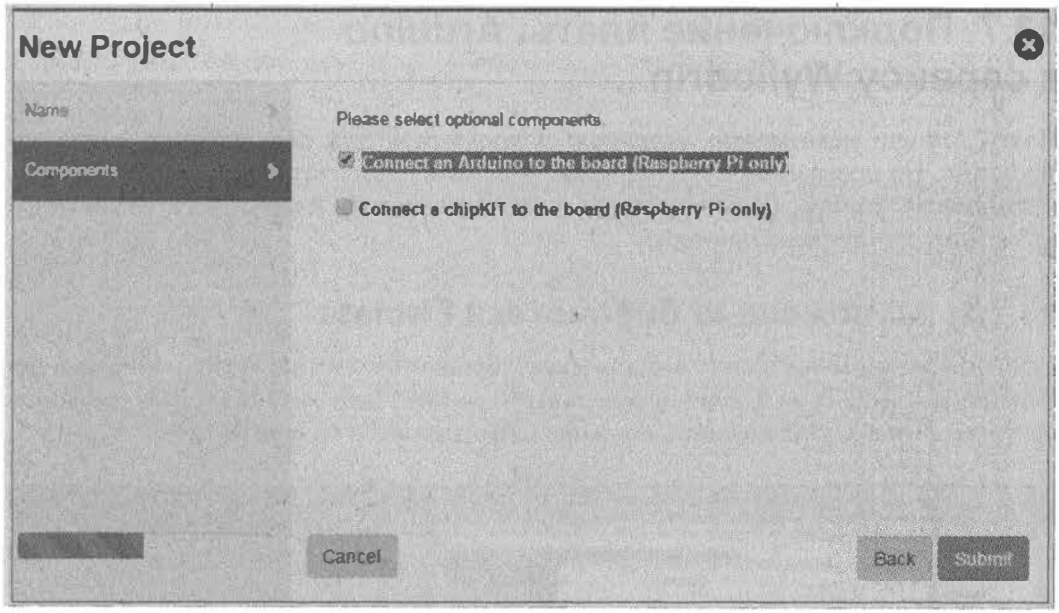


Рис. 13.28. Выбираем при создании проекта подключение платы Arduino

Создав проект, зайдем в окно редактирования проекта и в визуальном редакторе впишем код мигания светодиодом на 13-м выводе Arduino с периодичностью 2 сек. (рис. 13.29). Узнать порт подключения платы Arduino к Raspberry Pi можно прямо из сервиса Wyliodrin, запустив оболочку shell и выполнив команду `dmesg` (рис. 13.30).

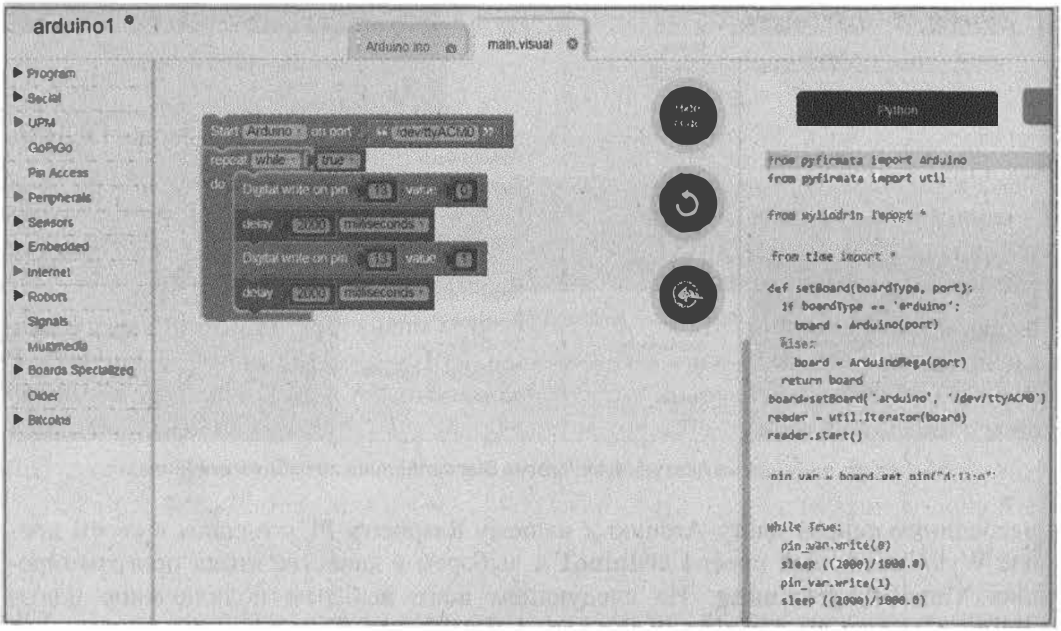


Рис. 13.29. Программа в VisualProgramming

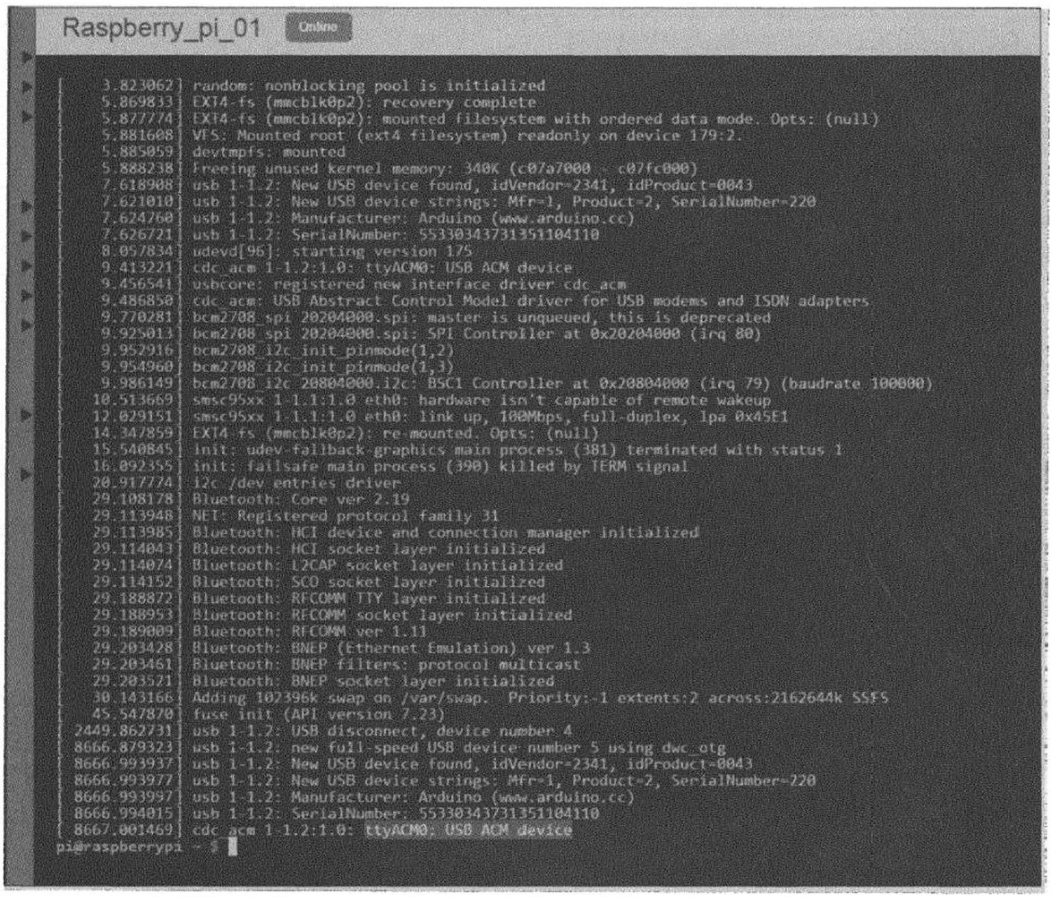


Рис. 13.30. Определение порта подключения Arduino с помощью оболочки shell

Осталось запустить проект и наблюдать мигание светодиода на выводе 13 платы Arduino.

Создадим новый проект (**arduino2**), для которого подключим к аналоговым входам Arduino аналоговый датчик температуры LM35 и фоторезистор (рис. 13.31), и рассмотрим отображение получаемых данных в виде виджетов и графиков.

Выберем для этого проекта тот же язык программирования — **VisualProgramming** и подключение платы Arduino: **Connect an Arduino to the board (Raspberry Pi only)** — как показано на рис. 13.28. Создав проект, заходим в режим редактирования и создаем в визуальном редакторе код, приведенный на рис. 13.32.

Запускаем проект на выполнение и наблюдаем вывод в монитор программы на сайте Wyliodrin показаний датчика температуры LM35 и фоторезистора (рис. 13,33).

Теперь добавим в браузере отображение температуры на виджете и показаний фоторезистора на графике. Для этого в окне редактирования приложения **arduino2** открываем вкладку **Signals** (рис. 13.34), выбираем элемент **Send signal "..." with value 0** и добавляем его в цикл опроса данных аналоговых входов. Добавить следует

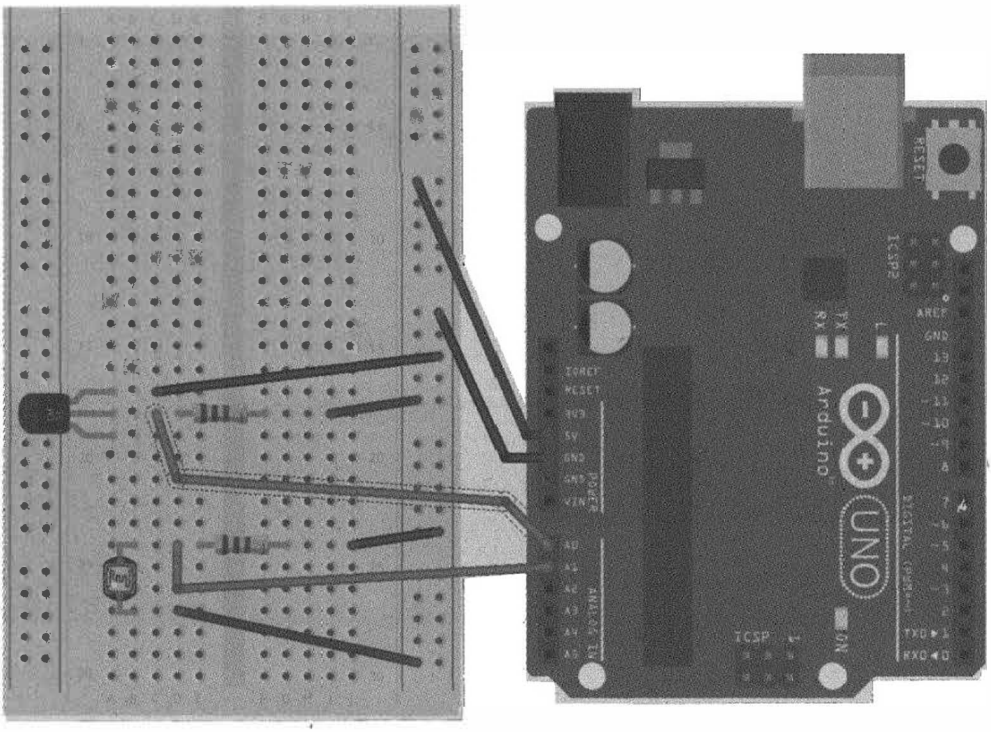


Рис. 13.31. Схема подключения к плате Arduino аналогового датчика температуры и фоторезистора

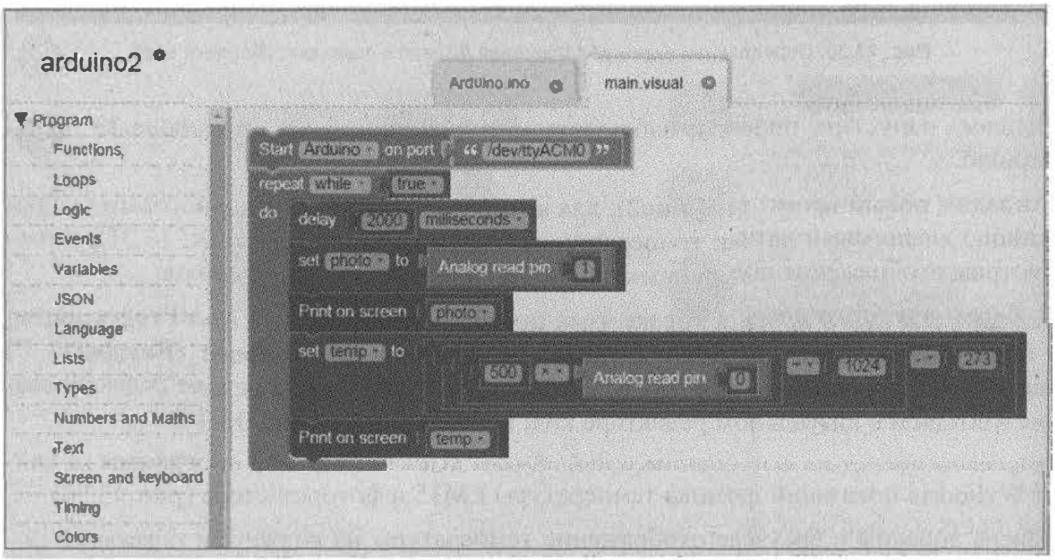


Рис. 13.32. Программа в VisualProgramming для получения показаний с датчиков

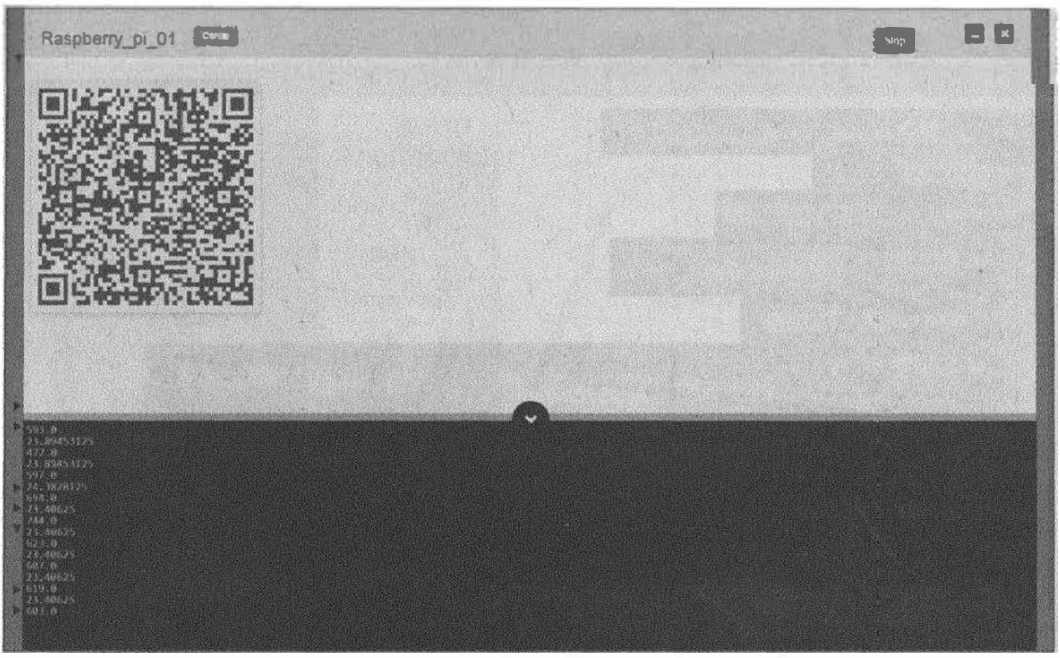


Рис. 13.33. Получение показаний с датчиков температуры и освещенности

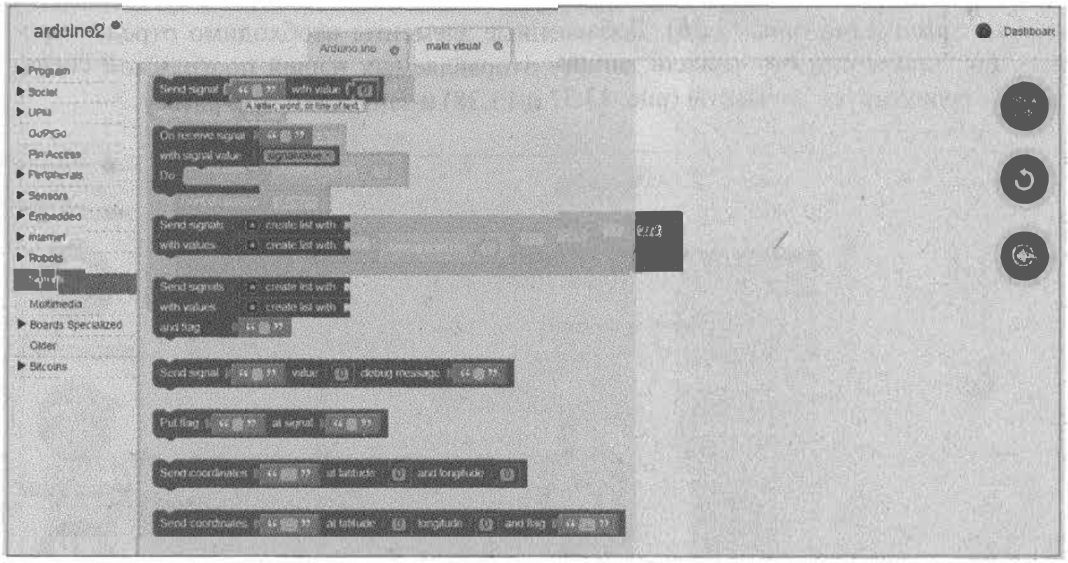


Рис. 13.34. Вкладка **Signal**

два таких элемента: один — для фоторезистора, второй — для датчика температуры (рис. 13.35). При этом присваиваем каждому сигналу соответствующее имя: **GraphPhoto** — для отправляемых значений фоторезистора, **VidgetTemp** — для отправляемых значений датчика температуры.

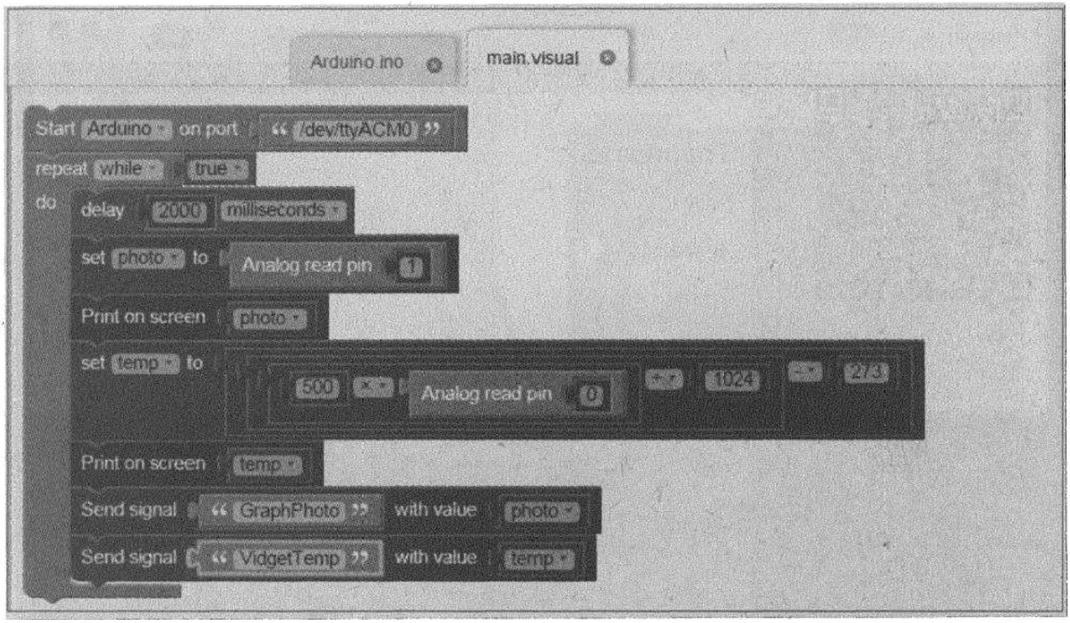


Рис. 13.35. Элементы вкладки **Signal** (**Send signal...**) добавлены в программу

Далее нажимаем в окне программы (см. рис. 13.34) на ссылку **Dashboard** и добавляем в окно из правого списка два элемента: виджет температуры **Thermometer** и график **Spline Line** (рис. 13.36). Добавленные элементы необходимо отредактировать так, чтобы они отображали данные отправляемых нашей программой сигналов, — редактируем элементы (рис. 13.37 и 13.38) и запускаем программу.

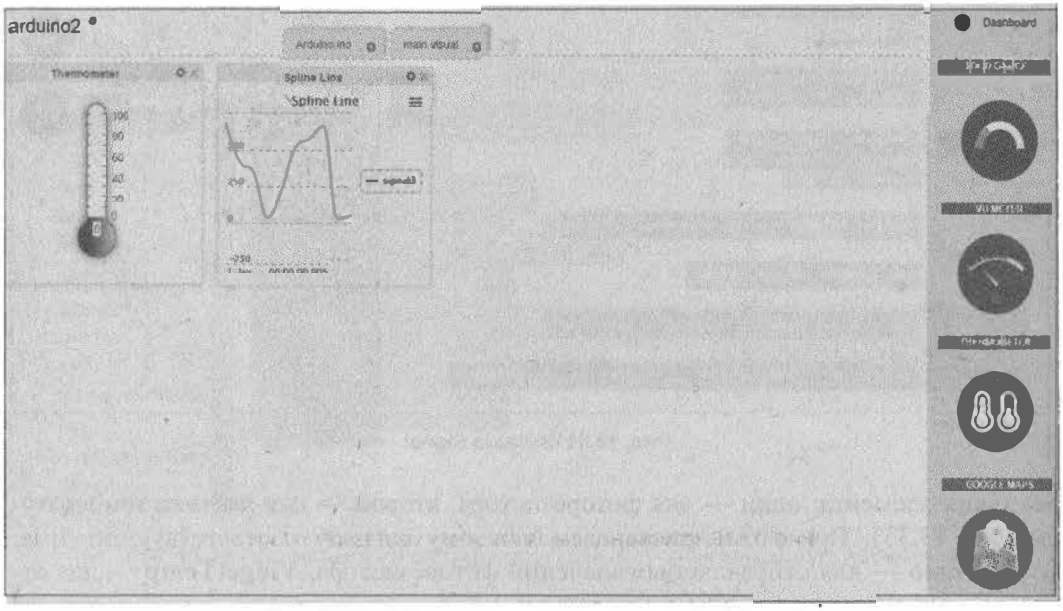


Рис. 13.36. Добавление элементов в окне **Dashboard**

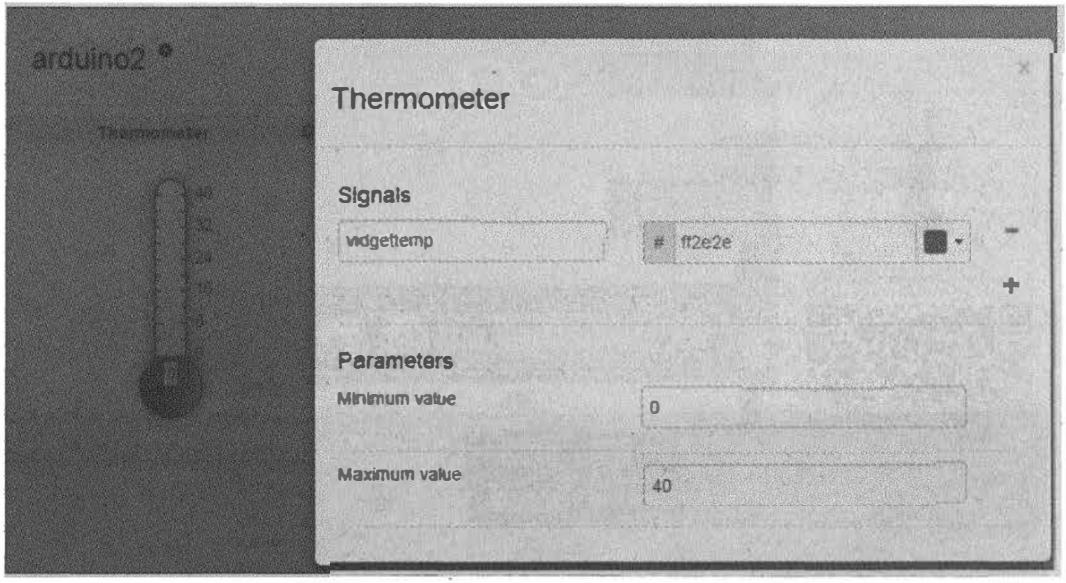


Рис. 13.37. Изменение параметров элемента Thermometer

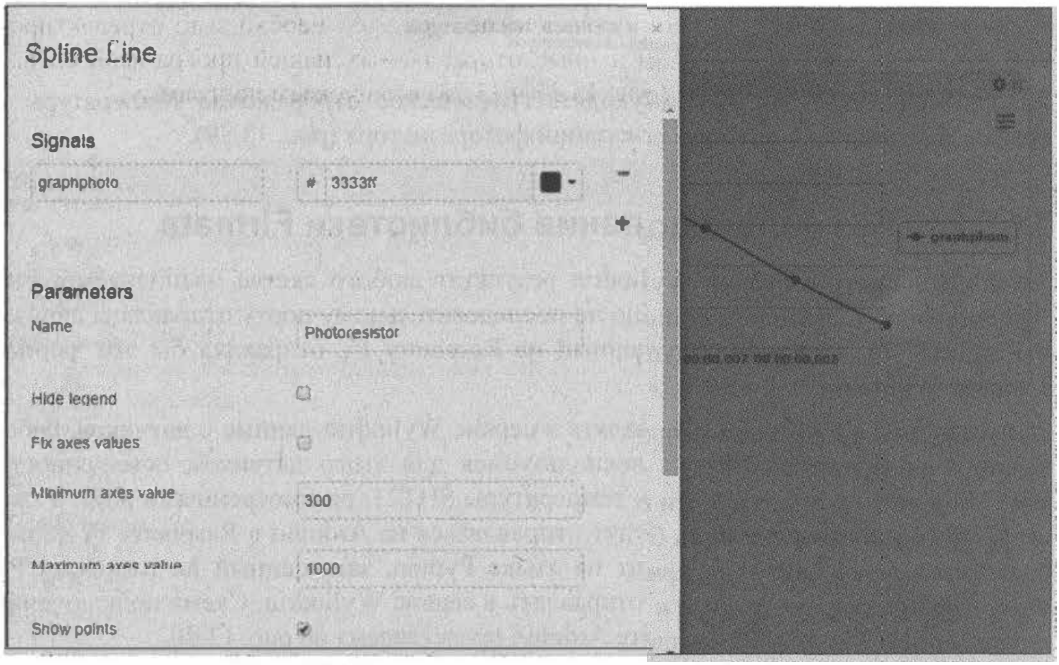


Рис. 13.38. Изменение параметров элемента Spline Line

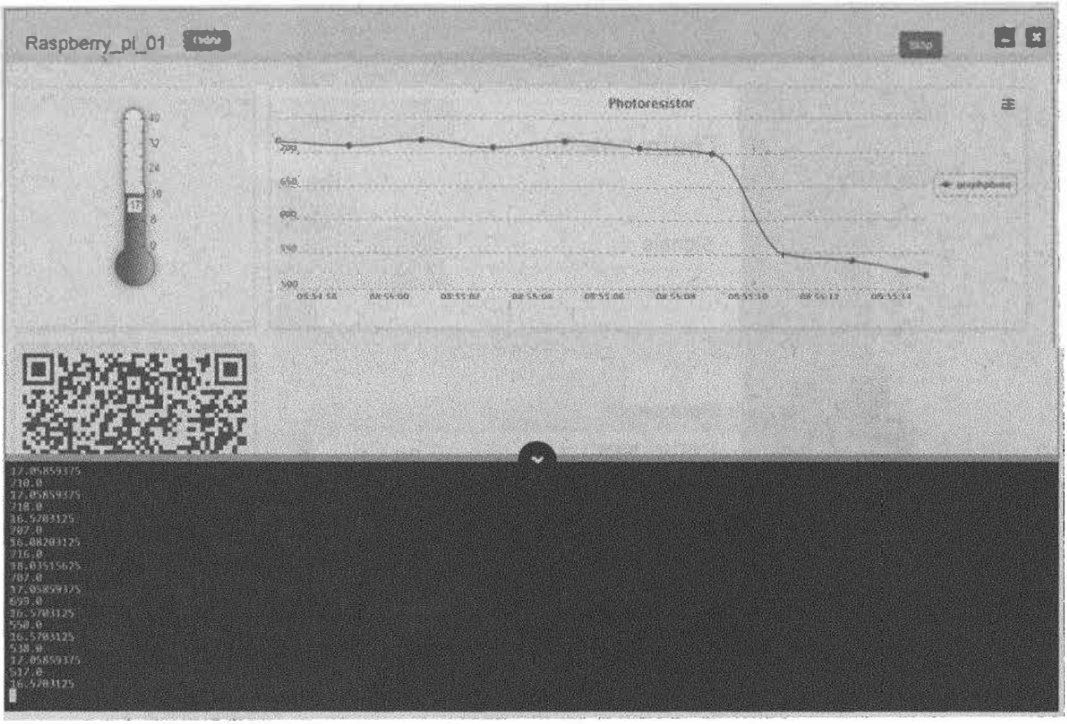


Рис. 13.39. Графическое отображение в браузере данных фоторезистора и датчика температуры

Теперь в браузере мы можем увидеть графическое отображение температуры и график изменения во времени показаний фоторезистора (рис. 13.39).

13.7.2. ...без использования библиотеки Firmata

Чтобы отправить в сервис Wyliodrin результат любого скетча, выполняемого на Arduino, надо чтобы плата Arduino по последовательному порту отправляла данные в Raspberry Pi, а скрипт, запущенный на Raspberry Pi, отправлял бы эти данные в сервис Wyliodrin.

Допустим, мы собираемся отправлять в сервис Wyliodrin данные с датчиков, работающих по протоколу I^2C, — воспользуемся для этого датчиком освещенности BH1750 и датчиком влажности и температуры SHT21, рассмотренными нами в *главе 6*. Данные с этих датчиков будут отправляться из Arduino в Raspberry Pi по последовательному порту, а скрипт на языке Python, запущенный на Raspberry Pi, станет принимать эти данные и отправлять в сервис Wyliodrin. Схема подключения датчиков BH1750 и SHT21 к плате Arduino представлена на рис. 13.40.

Загрузим на плату Arduino скетч из листинга 13.2. Arduino здесь получает данные с датчиков и отправляет эти данные в формате JSON по последовательному порту в Raspberry Pi.

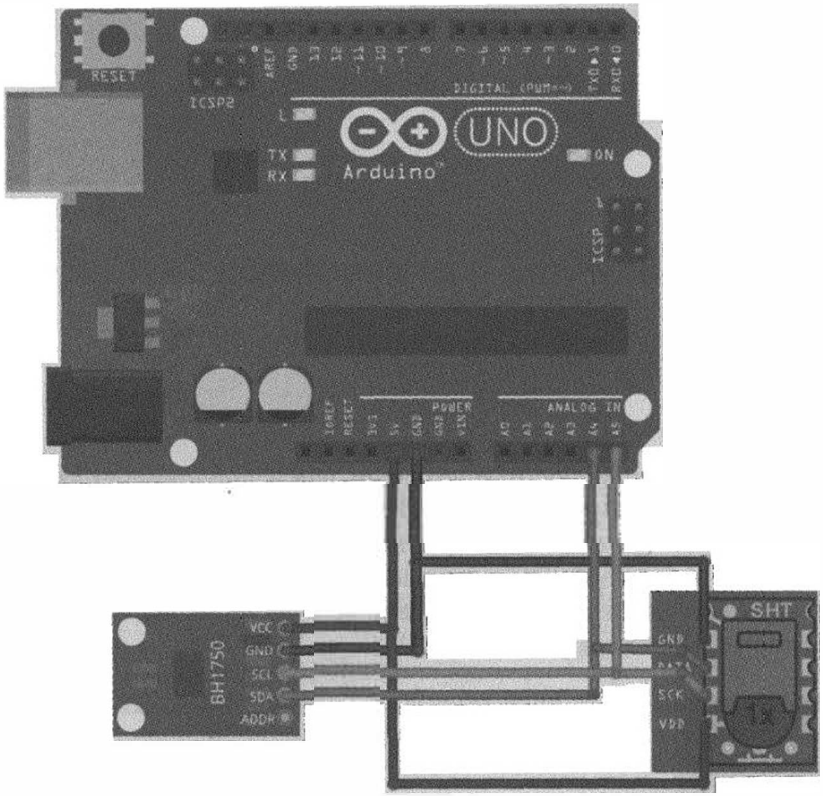


Рис. 13.40. Схема подключения датчиков BH1750 и SHT21 к плате Arduino

Листинг 13.2

```
// Подключение библиотек для датчика BH1750
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;
// Подключение библиотеки для датчика SHT21
#include <SHT2x.h>

uint16_t lux;
float temp;
float humidity;

void setup()
  {
  // подключение последовательного порта
  Serial.begin(9600);
  // запуск датчика BH1750
  lightMeter.begin();
  }
```

```
void loop()
  {
  // получение данных с BH1750
  lux = lightMeter.readLightLevel();
  humidity=SHT2x.GetHumidity();
  temp=SHT2x.GetTemperature();
  Serial.print("{lux:");Serial.print(lux);
  Serial.print(",temp:");Serial.print(temp);
  Serial.print(",humidity:");Serial.print(humidity);
  Serial.println("}");
  // задержка 10 сек
  delay(10000);
  }
```

Зайдем теперь в сервис Wyliodrin и создадим новое приложение: **arduino3**. Выберём для него язык программирования **Python** и впишем в окне редактирования программы (рис. 13.41) код, приведенный в листинге 13.3. Здесь мы считываем по последовательному порту из Arduino данные с датчиков BH1750 и SHT21 в формате JSON и отправляем эти значения в виджеты **Lux** (Освещенность), **Temp** (Температура) и **Humidity** (Влажность).

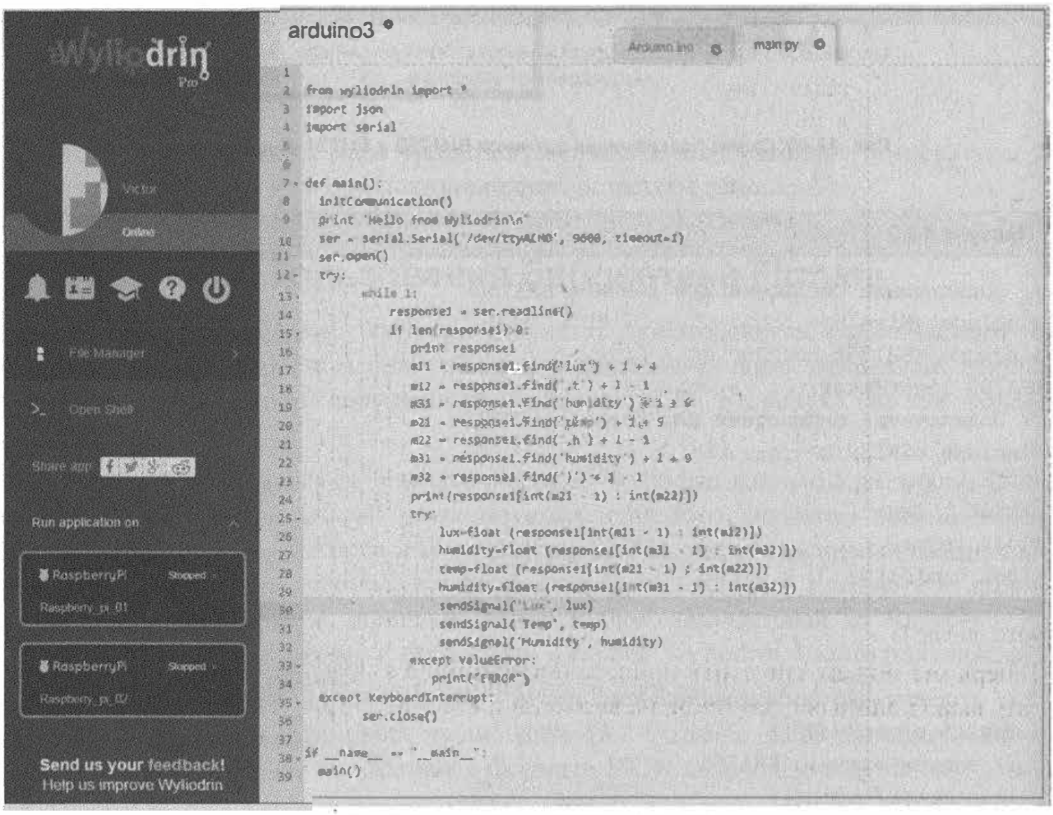


Рис. 13.41. Код приложения **arduino3**

Листинг 13.3

```
from wyliodrin import
import json
import serial

def main():
  initCommunication()
  print 'Hello from Wyliodrin\n'
  ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
  ser.open()
  try:
        while 1:
            response1 = ser.readline()
            if len(response1)>0:
               print response1
               m11 = response1.find('lux') + 1 + 4
               m12 = response1.find(',t') + 1 - 1
               m31 = response1.find('humidity') + 1 + 9
               m21 = response1.find('temp') + 1 + 5
               m22 = response1.find(',h') + 1 - 1
               m31 = response1.find('humidity') + 1 + 9
               m32 = response1.find('}') + 1 - 1
               print(response1[int(m21 - 1) : int(m22)])
               try:
                   lux=float (response1[int(m11 - 1) : int(m12)])
                   humidity=float (response1[int(m31 - 1) : int(m32)])
                   temp=float (response1[int(m21 - 1) : int(m22)])
                   humidity=float (response1[int(m31 - 1) : int(m32)])
                   sendSignal('Lux', lux)
                   sendSignal('Temp', temp)
                   sendSignal('Humidity', humidity)
               except ValueError:
                  print("ERROR")
  except KeyboardInterrupt:
        ser.close()

if __name__ == "__main__":
  main()
```

Теперь мы можем запустить приложение **arduino3** в браузере в Wyliodrin и наблюдать вывод данных с датчиков на виджеты и в монитор (рис. 13.42).

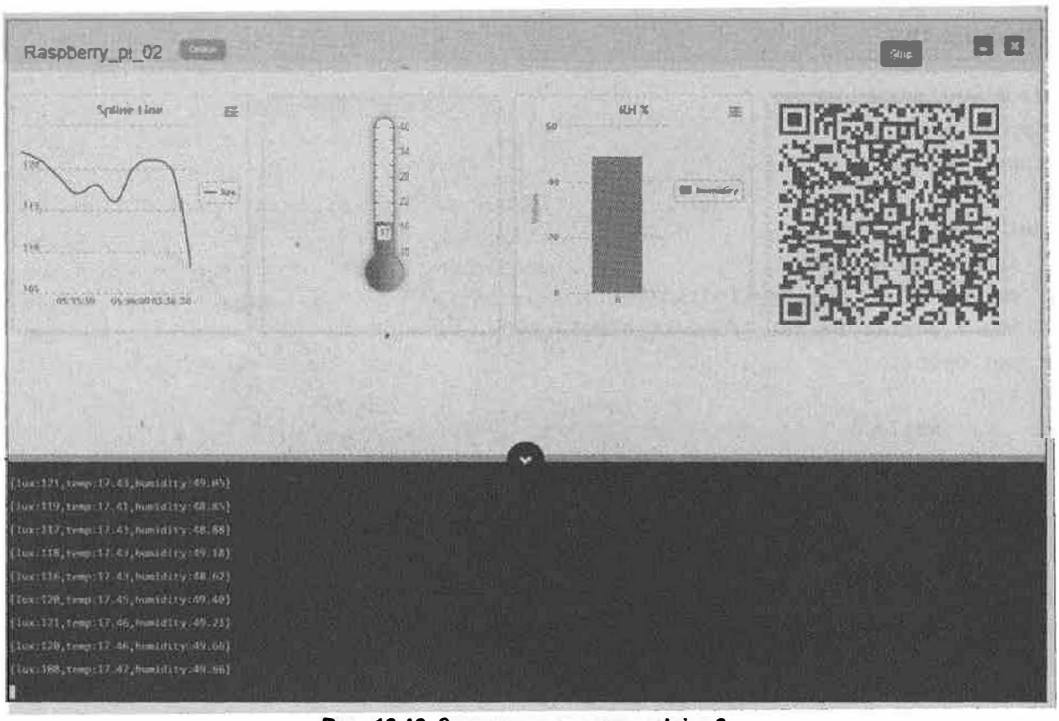


Рис. 13.42. Запуск приложения **arduino3**

13.8. Совместная работа Raspberry Pi и платы GrovePi

Система Grove представляет собой набор готовых к использованию электронных устройств в виде модулей. Она включает базовую плату и различные модули со стандартными разъемами для подключения к ней. Базовая плата позволяет легко связать выводы популярных микроконтроллеров с контактами модулей Grove. Существуют базовые платы для Arduino, Raspberry Pi, Intel Galileo, Intel Edison, mbed, LaunchPad и других микроконтроллеров и микрокомпьютеров. Большой выбор базовых плат и модулей Grove представлен в интернет-магазине Seeedstudio: **https://www.seeedstudio.com/depot/s/grove.html?search_in_description=0**.

Для подключения к Raspberry Pi (рис. 13.43) используются платы расширения GrovePi и GrovePi+ (плата для подключения плат Raspberry Pi версий «A+» и «B+»).

Плата GrovePi+ поддерживает подключение семи цифровых и трех аналоговых выводов, трех портов I^2C, одного последовательного порта подключения к GrovePi и последовательного порта подключения к Raspberry Pi.

Для работы Raspberry Pi с платой GrovePi необходимо установить соответствующее программное обеспечение. Для этого подключаемся по ssh к нашему Raspberry Pi и клонируем репозиторий GrovePi:

```
git clone https://github.com/DexterInd/GrovePi
```

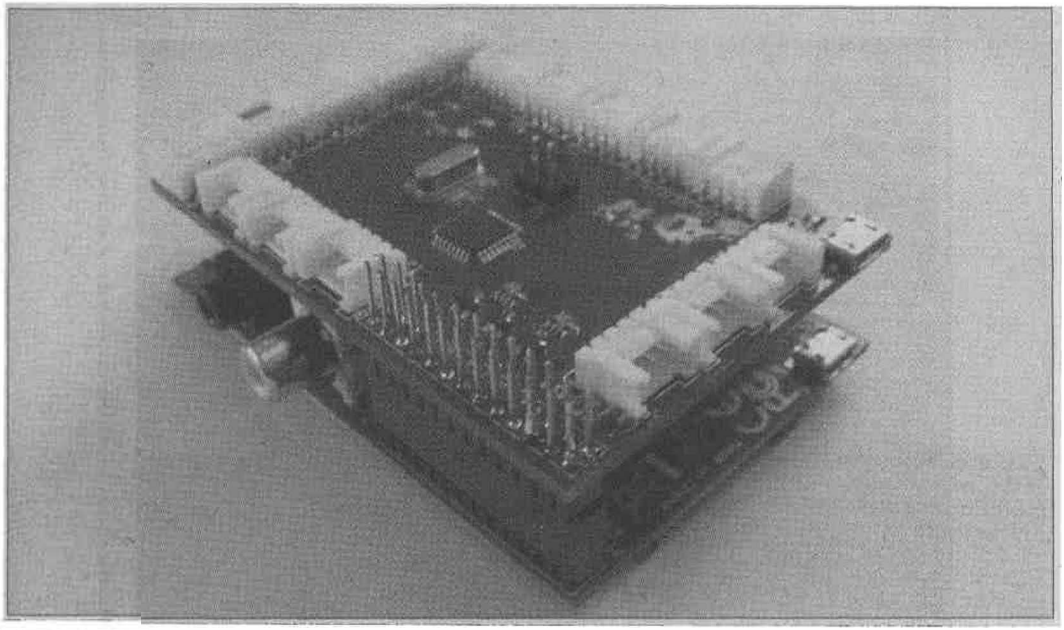


Рис. 13.43. Подключение платы GrovePi+ к Raspberry Pi

Переходим в папку GrovePi/Script:

```
cd GrovePi/Script
```

Предоставлям файлу install.sh права на выполнение:

```
sudo chmod +x install.sh
```

И запускаем его:

```
sudo ./install.sh
```

Скрипт будет работать достаточно продолжительное время, связанное с закачкой из Интернета и установкой дополнительных пакетов (рис. 13.44). По завершении работы скрипта Raspberry Pi необходимо перезагрузить:

```
sudo reboot
```

После перезагрузки можно подключить GrovePi к Raspberry и использовать с ним устройства Grove. На странице **http://www.seeedstudio.com/wiki/GrovePi+** можно найти ссылки на проекты с кодом по подключению некоторых датчиков системы Grove (рис. 13.45).

Работать с устройствами Grove через плату GrovePi в Wyliodrin весьма просто: подсоединим к контактам A0 платы GrovePi датчик света Grove light (рис. 13.46), создадим в Wyliodrin новое приложение **arduino4** и выберем для него язык **VisualProgramming**, напишем программу вывода получаемых значений на график (рис. 13.47), запустим программу и увидим график получаемых с датчика Grove light значений (рис. 13.48).

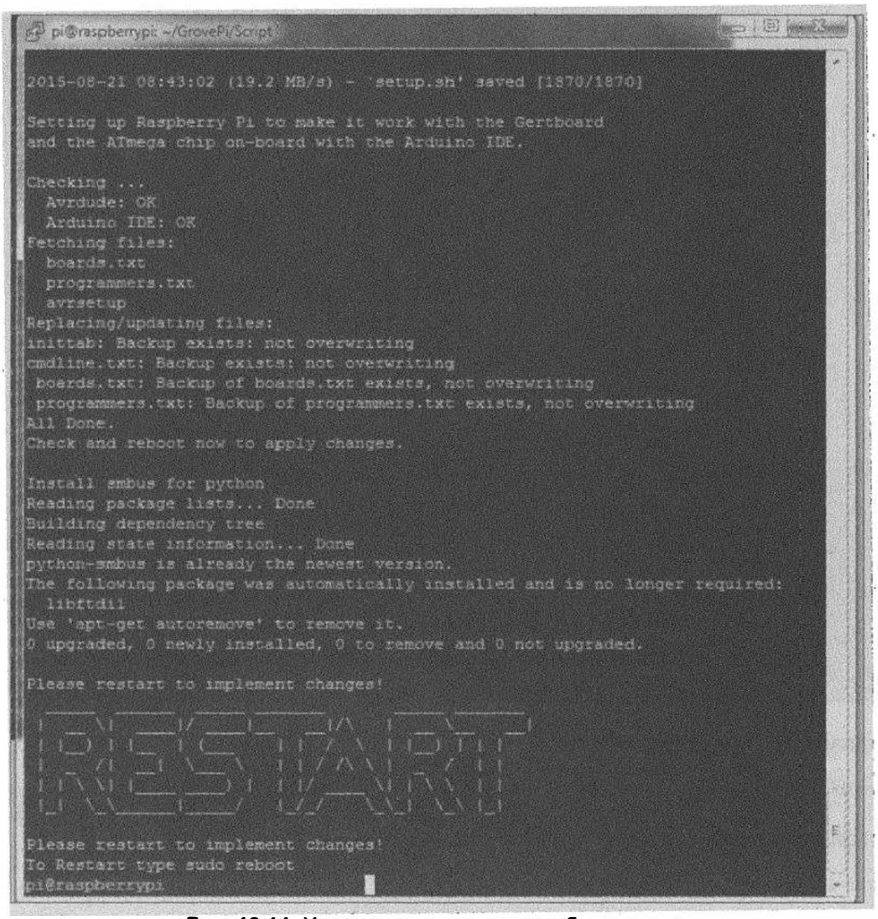


Рис. 13.44. Установка программного обеспечения

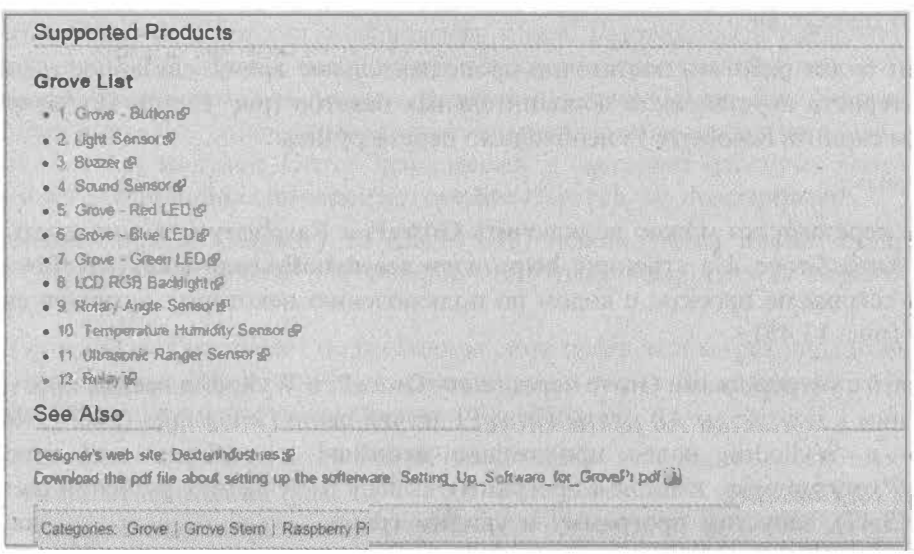


Рис. 13.45. Ссылки на проекты с подключением датчиков Grove к плате GrovePi

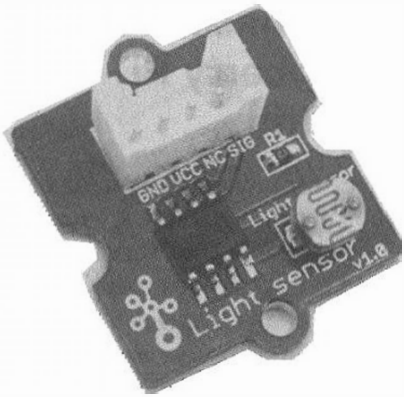


Рис. 13.46. Датчик Grove light

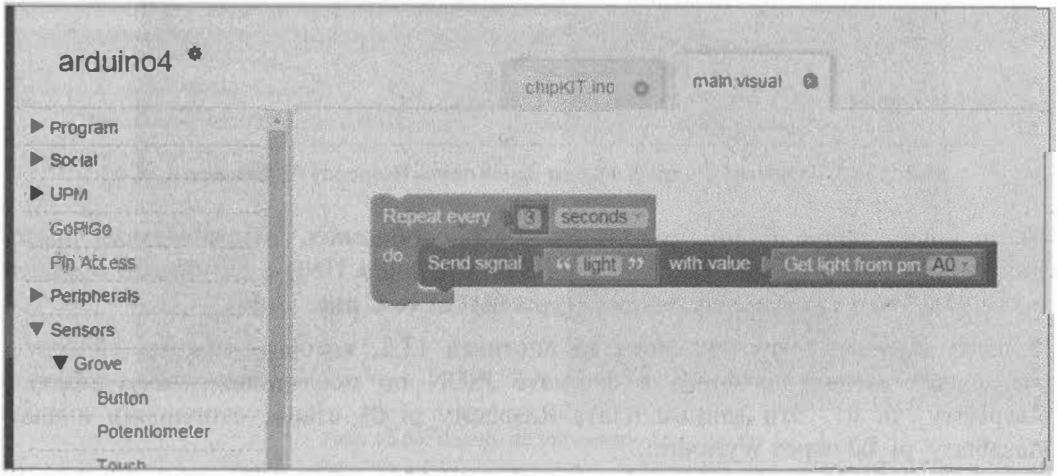


Рис. 13.47. Программа для вывода данных датчика Grove light на график

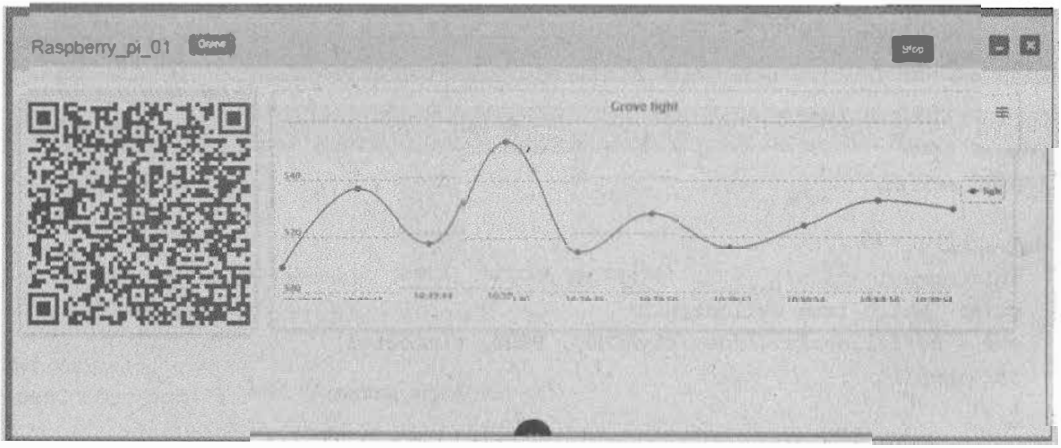


Рис. 13.48. График получаемых с датчика Grove light значений

13.9. Обмен сообщениями между платами Raspberry Pi через сервис Wyliodrin

В сервисе Wyliodrin существует возможность обмена сообщениями между платами. Чтобы исследовать эту возможность, добавим к нашему профилю еще одну плату Raspberry Pi (см. *разд. 13.1*) — и назовем ее **Raspberry_pi_02**. Запишем на нее образ, файл настроек wyliodrin.json и подключим плату к сети Интернет. Откроем свой профиль Wyliodrin и увидим, что она находится в статусе **Online** (рис. 13.49).

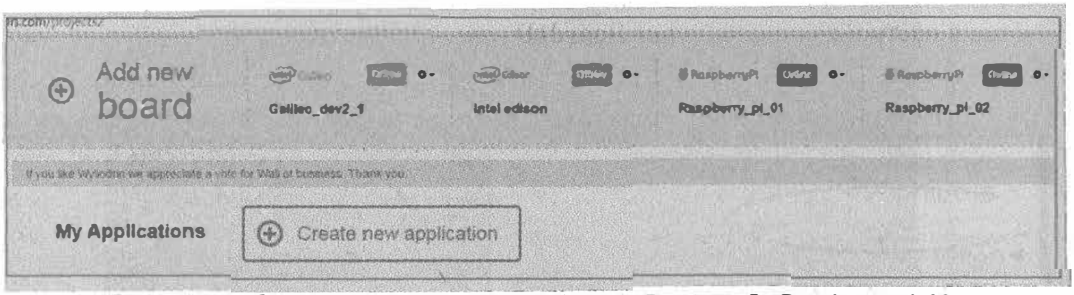


Рис. 13.49. Добавление в профиль еще одной платы Raspberry Pi: **Raspberry_pi_02**

Наша новая плата будет принимать JSON-сообщения, отправляемые платой Raspberry_pi_01, к которой подключена плата Arduino UNO с датчиками освещенности BH1750 и влажности и температуры SHT21 (см. рис. 13.40).

В плату Arduino загружен скетч из листинга 13.2, который каждые 10 секунд отправляет данные датчиков в формате JSON по последовательному порту в Raspberry_ pi_01. Эти данные плата Raspberry_pi_01 станет отправлять в плату Raspberry_pi_02 через Wyliodrin.

Создадим новое приложение: **arduino5_send_data**, выберем для него язык программирования **Python** и запишем в него код из листинга 13.4.

Листинг 13.4

```
from wyliodrin import *
import json
import serial

def main():
  initCommunication()
  print 'Hello from Wyliodrin\n'
  ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
  ser.open()
  try:
        while 1:
            response1 = ser.readline()
```

```
            if len(response1)>0:
               print response1
               sendMessage('victoruni_raspberry_pi_02@wyliodrin.com', 'label1',
json.dumps(response1))
  except KeyboardInterrupt:
        ser.close()

if __name__ == "__main__":
  main()
```

Функция `sendMessage('boardid', 'label', json.dumps(message))` отправляет сообщение `message` с меткой `label` в плату `boardid`. Параметр `boardid` можно посмотреть, выбрав в настройках платы пункт **BoardID** (рис. 13.50).

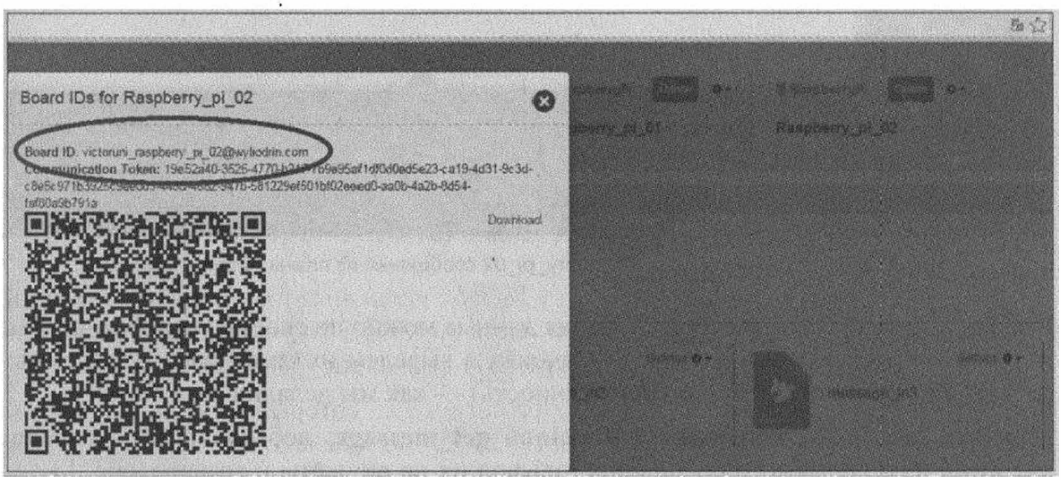


Рис. 13.50. Значение параметра `boardid`

Теперь напишем приложение для платы Raspberry_pi_02, обеспечивающее получение сообщений от платы Raspberry_pi_01: создаем новое приложение **arduino6_get_message**, выбираем для него язык программирования **Python** и заносим в приложение код из листинга 13.5.

Листинг 13.5

```
from wyliodrin import *
import json

def messagesReceiver(sender, label, error, message):
 print(message)

def main():
openConnection('label1', messagesReceiver)

if __name__ == "__main__":
main()
```

Теперь при запуске на плате Raspberry_pi_02 этого приложения мы увидим получение сообщений, отправленных приложением из платы Raspberry_pi_01 (рис. 13.51).

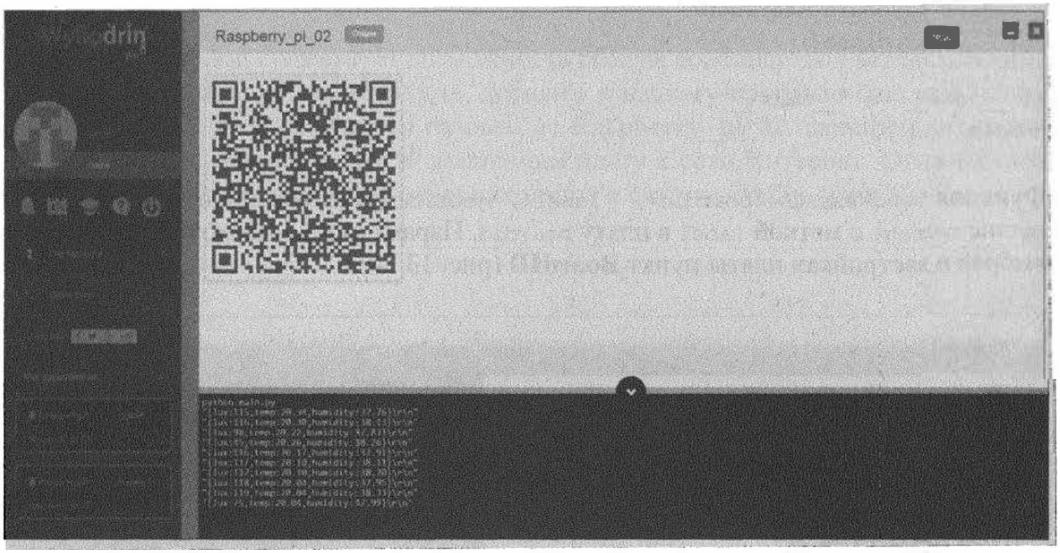


Рис. 13.51. Получение платой Raspberry_pi_02 сообщений из платы Raspberry_pi_01

Использовать получаемые в сообщениях данные можно по своему усмотрению. Мы же просто выделим из них нужные значения и выведем их на виджеты (для температуры и влажности) и график (освещенность) — как мы делали в *разд. 13.7*.

Изменим для этого приложение **arduino6_get_message**, добавив в него парсинг значений из получаемого сообщения и вывод их на виджеты и график (рис. 13.52). Код приложения приведен в листинге 13.6.

Листинг 13.6

```
from wyliodrin import *
import json

def messagesReceiver(sender, label, error, message):
 print(message)
 m11 = message.find('lux') + 1 + 4
 m12 = message.find(',t') + 1 - 1
 m31 = message.find('humidity') + 1 + 9
 m21 = message.find('temp') + 1 + 5
 m22 = message.find(',h') + 1 - 1
 m31 = message.find('humidity') + 1 + 9
 m32 = message.find('}') + 1 - 1
 try:
  lux=float (message[int(m11 - 1) : int(m12)])
  humidity=float (message[int(m31 - 1) : int(m32)])
```

```
    temp=float (message[int(m21 - 1) : int(m22)])
    humidity=float (message[int(m31 - 1) : int(m32)])
    print(lux)
    print(temp)
    print(humidity)
    sendSignal('Lux', lux)
    sendSignal('Temp', temp)
    sendSignal('Humidity', humidity)
  except ValueError:
        print("ERROR")

def main():
openConnection('label1', messagesReceiver)

if __name__ == "__main__":
main()
```

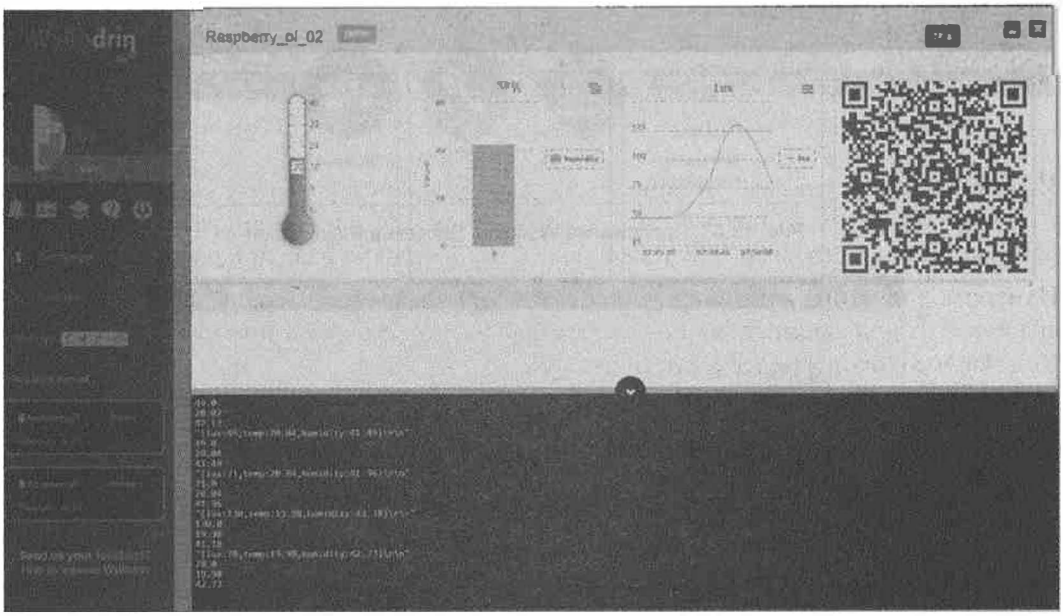


Рис. 13.52. Вывод значений, получаемых платой Raspberry_pi_02 из платы Raspberry_pi_01, на виджеты и график

13.10. Отправка данных в сервис Wyliodrin с мобильного устройства

Сервис Wyliodrin позволяет плате Raspberry Pi получать данные с датчиков мобильного устройства (планшета или смартфона). Для этого необходимо скачать и установить на него из Play Маркет приложение Wyliodrin Sensors (рис. 13.53).

В приложении следует задать настройки: ID устройства и интервал передачи данных (рис. 13.54) и выбрать список датчиков устройства для передачи данных (рис. 13.55).



Рис. 13.53. Приложение Wyliodrin Sensors в Play Маркет

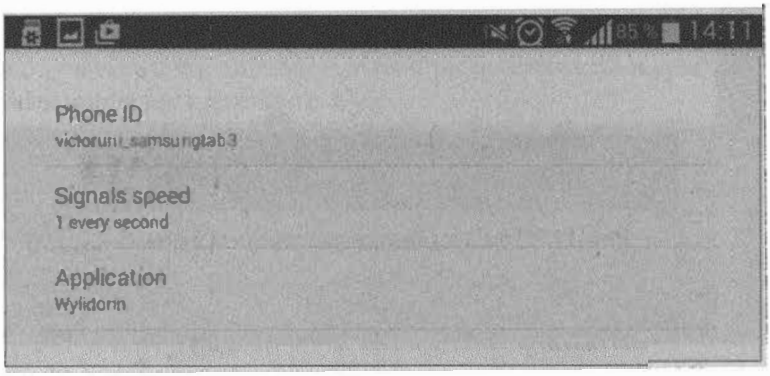


Рис. 13.54. Задание ID устройства и интервала передачи данных

В окне профиля платы Raspberry Pi из соответствующего выпадающего меню надо выбрать возможность получения мобильных сообщений: **Enable Mobile Messages** (рис. 13.56). Сделав это, создайте соединение между платой и мобильным приложением — просто нажмите кнопку **Board ID** в выпадающем меню, и QR-код соединения отобразится на экране. Чтобы сканировать этот код, нажмите на большую красную кнопку **Scan Token** в мобильном приложении. Завершив сканирование QR-кода (рис. 13.57), мобильное устройство начнет посылать в сервис Wyliodrin данные, поступающие из выбранных датчиков.

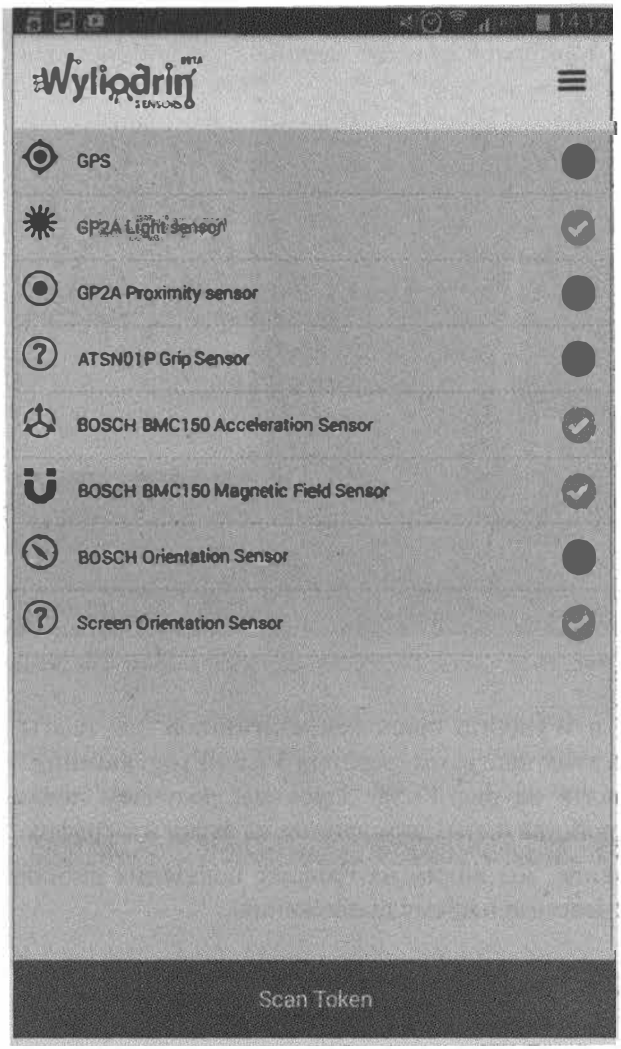


Рис. 13.55. Выбор датчиков для передачи данных

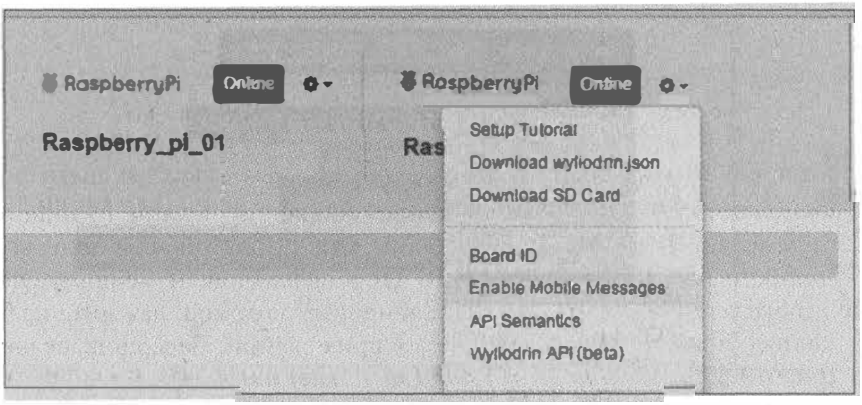


Рис. 13.56. Включение мобильных сообщений для платы Raspberry Pi

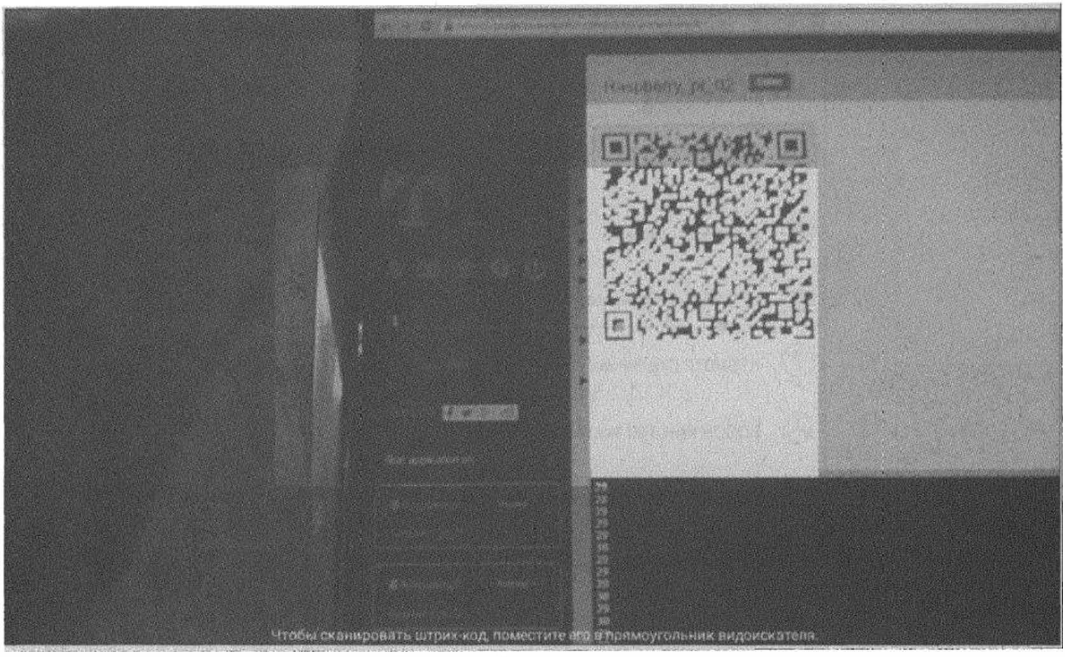


Рис. 13.57. Сканирование QR-кода в мобильном приложении

Теперь создадим в Wyliodrin приложение **arduino8** для платы Raspberry_pi_02 и выберем для него язык программирования **VisualProgramming**. Код для этого приложения представлен на рис. 13.58. Здесь мы получаем показания сенсора **light** с мобильного устройства и выводим данные на экран и в график.

Запустив приложение, мы видим на графике показания датчика **light** мобильного устройства, передаваемые нашему приложению.

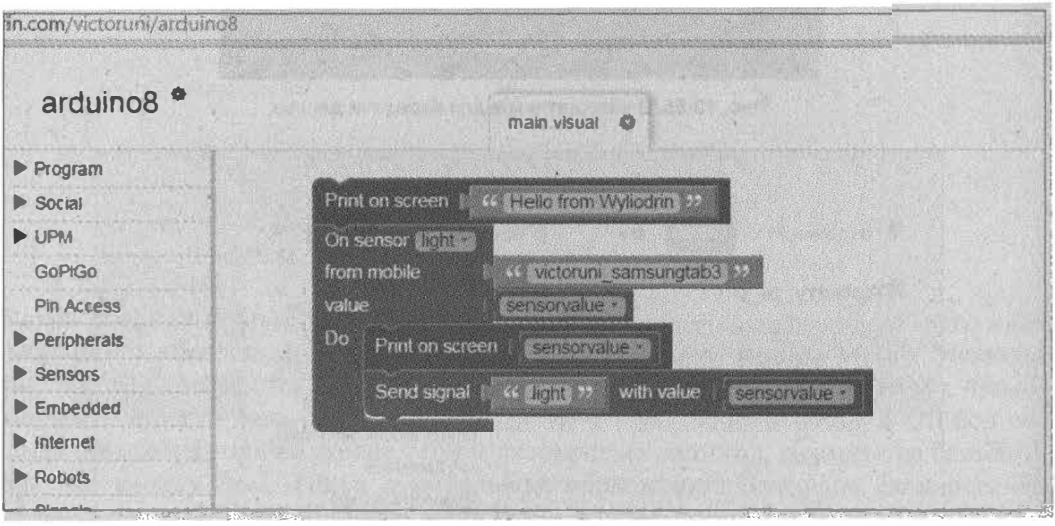


Рис. 13.58. Код приложения **arduino8**

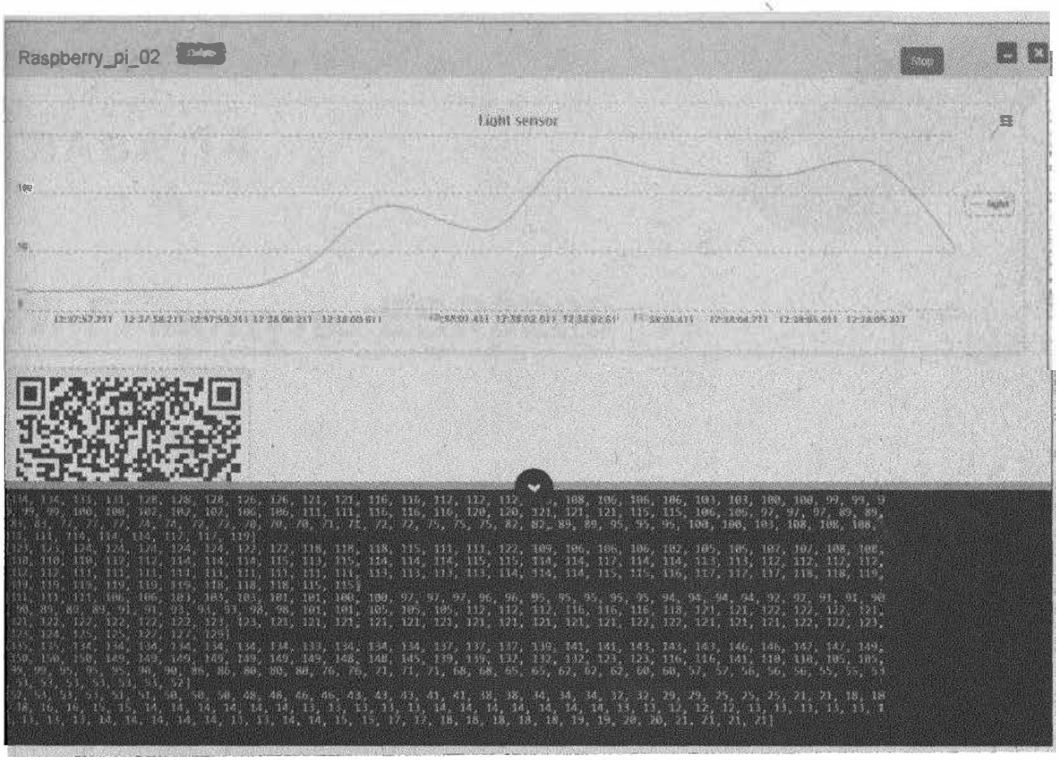
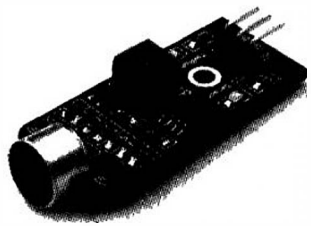


Рис. 13.59. График и показания датчика **light** мобильного устройства, передаваемые нашему приложению

ГЛАВА 14



Wi-Fi модуль ESP8266

С конца 2014 года на китайских торговых площадках появились Wi-Fi модули ESP8266. Причем, как выяснилось, это не просто модули Wi-Fi, а полноценные 32-битные микроконтроллеры со своими наборами GPIO, в том числе поддерживающими шины SPI, UART и I^2C. При этом сами модули состоят из минимального количества деталей: собственно чипа ESP8266, Flash-памяти и кварцевого генератора. Характеристики модулей представлены в табл. 14.1.

***Таблица 14.1.** Характеристики модулей ESP8266*

Частота	Wi-Fi 2412–2484 МГц
Стандарт	802.11 b/g/n
Мощность	+ 20 дБ
Поддерживаемые типы шифрования	WEP, WPA, WPA2
Поддерживаемые режимы работы	Клиент (STA), точка доступа (AP), клиент+точка доступа (STA+AP)
Напряжение питания	1,7–3,6 В
Потребление тока	70 мА (пиковое значение 240 мА)
Количество доступных выводов GPIO	4–10
Внешняя Flash-память	512 Кбайт
RAM данных	80 Кбайт
RAM инструкций	32 Кбайт
Температурный режим	От –40 до +70 °С

В настоящее время выпускается 12 модификаций плат модулей ESP8266, различающихся количеством выводов и вариантами исполнения (рис. 14.1). Модули продаются с загруженной прошивкой, которая образует мост Wi-Fi → UART для подключения к другому микроконтроллеру, в том числе и к Arduino. Настройка соединения и обмен данными осуществляются с помощью АТ-команд.

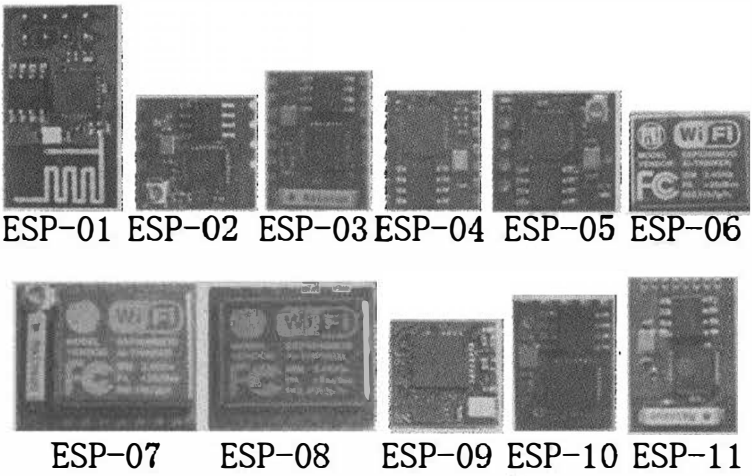


Рис. 14.1. Модули ESP8266

Возможно два варианта работы с модулем:

- использование его совместно с микроконтроллером, который будет управлять модулем по UART;
- создание собственной прошивки для чипа ESP8266 и его применение как самодостаточного устройства.

14.1. Режим AT-команд

Рассмотрим работы с модулем ESP8266 в режиме AT-команд, для чего подключим его к компьютеру через переходник USB-to-RS232. Назначение выводов модуля представлено на рис. 14.2. Для работы модуля требуется внешнее питание 3,3 В. Схема подключения весьма простая:

- вывод **VCC** — питание платы (+3,3 В);
- вывод **GND** — общий;
- выводы **RX** и **TX** — подключаем к конвертеру USB-to-RS232 (в режиме 3,3 В);
- вывод **CH_PD** (Chip enable) — подключаем к питанию платы (+3,3 В).

Для отправки в модуль AT-команд я использовал в Mac OS X программу CoolTerm, а в ОС Windows — программу Termite. Узнать скорость COM-порта для соедине-

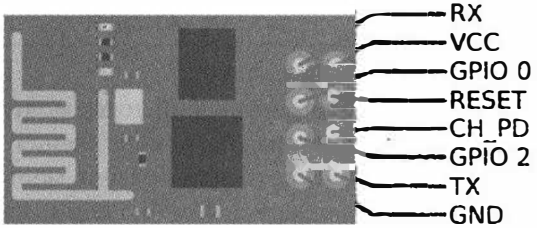


Рис. 14.2. Назначение выводов модуля ESP8266 исполнения ESP-01

ния с модулем можно только экспериментально, поскольку для различных прошивок она может быть разной. Так, для моего модуля эта скорость оказалась равной 9600 бод. Кроме того, установить обмен удалось только после отключения и повторного подключения к питанию вывода **CH_PD**.

После удачного подключения модуля к компьютеру набираем в терминале AT и должны получить в ответ от модуля OK. Команда AT+GMR выдает номер версии прошивки модуля, команда AT+RST — перезагружает модуль (рис. 14.3).

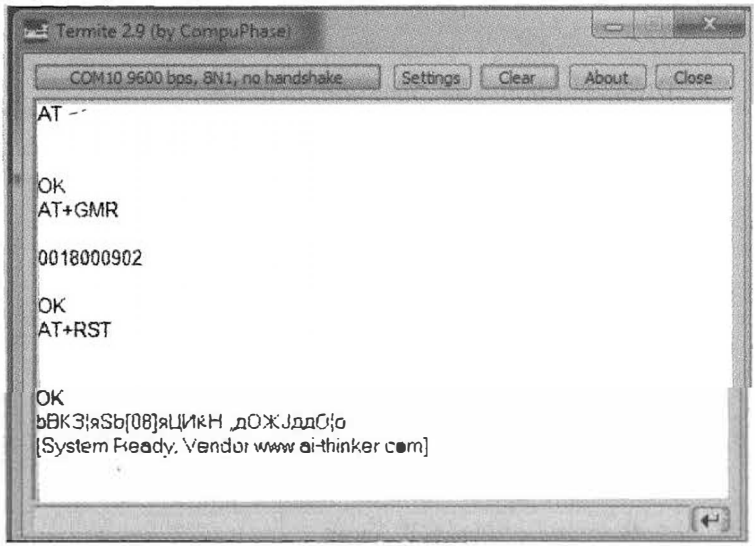


Рис. 14.3. Отправка AT-команд в модуль из программы Termite

Набор доступных AT-команд в новых версиях прошивки модулей ESP8266 постоянно пополняется. В табл. 14.2 приведен полный список AT-команд для комплекта средств разработки (SDK) 0.9.5 версии 021.

***Таблица 14.2.** Полный список AT-команд для SDK 0.9.5 версии 021*

Команда	Описание	Выполнение
AT	Проверка модуля. Если модуль успешно стартовал, то он отвечает OK	AT
AT+RST	Перезапуск модуля	AT+RST
AT+GMR	Отобразить версию прошивки	AT+GMR
AT+GSLP	Переход в режим пониженного энергопотребления	AT+GSLP=<время мс>
ATE	Включить/выключить эхо	AT0 AT1
AT+RESTORE	Сбросить на заводские настройки	AT+RESTORE
AT+UART	Настройка последовательного интерфейса	AT+UART= baudrate, databits, stopbits, parity, flow control

Таблица 14.2 (продолжение)

Команда	Описание	Выполнение
AT+CWMODE	Переключение режима Wi-Fi. Для вступления в силу требуется перезапуск модуля командой AT+RST	AT+CWMODE? AT+CWMODE=1 (station) AT+CWMODE=2 (AP) AT+CWMODE=3 (station+AP)
AT+CWJAP	Подключение к AP (точке доступа)	AT+CWJAP =<идентификатор сети>,<пароль> AT+CWJAP?
AT+CWLAP	Отобразить список доступных AP	AT+CWLAP
AT+CWQAP	Отключение от AP	AT+CWQAP
AT+CWSAP	Установить параметры для режима AP	AT+CWSAP= <идентификатор сети>, <пароль>, <канал>, <тип шифрования>
AT+CWLIF	Отобразить IP-адреса подключенных клиентов	AT+CWLIF
AT+CWDHCP	Установить режим DHCP	AT+CWDHCP=<режим>, <вкл>
AT+CIPSTAMAC	Посмотреть/установить MAC-адрес в режиме station	AT+CIPSTAMAC=<mac> AT+CIPSTAMAC?
AT+CIPAPMAC	Посмотреть/установить MAC-адрес в режиме softAP	AT+CIPAPMAC=<mac> AT+CIPAPMAC?
AT+ CIPSTA	Посмотреть/установить IP-адрес в режиме station	AT+ CIPSTA=<ip> AT+ CIPSTA?
AT+ CIPAP	Посмотреть/установить IP-адрес в режиме softAP	AT+ CIPAP=<ip> AT+ CIPAP?
AT+CIPSTATUS	Отобразить статус подключения. Возвращает ID соединения, тип соединения (TCP или UDP), IP-адрес, порт, тип связи (клиент, сервер)	AT+CIPSTATUS
AT+CIPSTART	Установить подключение TCP или UDP	
AT+CIPSEND	Отправить данные	AT+CIPSEND=? AT+CIPSEND=<длина> AT+CIPSEND=<идентификатор><длина>
AT+CIPCLOSE	Закрыть подключение TCP или UDP	
AT+CIFSR	Отобразить IP-адрес, который получили от AP, и адрес softAP	
AT+CIPMUX	Выбрать режим одиночных или множественных подключений	
AT+CIPSERVER	Запустить (перезапустить) сервер	

***Таблица 14.2** (окончание)*

Команда	Описание	Выполнение
AT+CIPSTO	Установить тайм-аут сервера	
AT+CIPMODE	Установить сквозной режим	
AT+CIUPDATE	Обновление прошивки через облако. Модуль должен находиться в режиме 1 или 3 и быть подключенным к AP с доступом к Интернету	
AT+PING	Пинг по имени хоста или IP-адресу	
+IPD	Получить данные из сети	

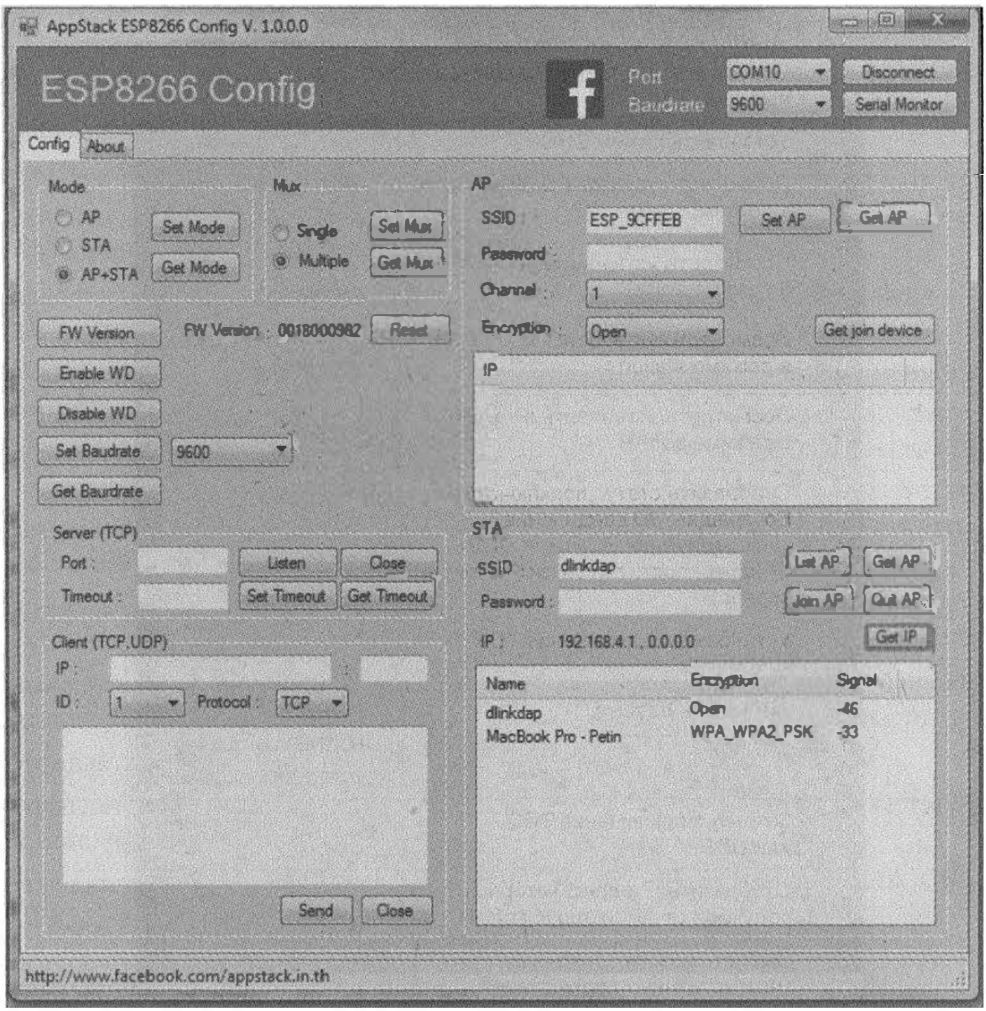


Рис. 14.4. Программа AppStack ESP8266 Config

Легко настроить модуль ESP8266, не заморачиваясь с AT-командами, можно с помощью программы AppStack ESP8266 Config, свободно доступной для закачки, например, с российского сайта поддержки платы по ссылке: **http://esp8266.ru/download/esp8266-utils/ESP8266_Config.zip**.

Внешний вид программы AppStack ESP8266 Config представлен на рис. 14.4. Настройка модуля осуществляется с помощью графического интерфейса, при этом выполнение команд можно видеть в мониторе программы (рис. 14.5). Из командной строки монитора также можно посылать в модуль и AT-команды.

Обновление прошивок модуля ESP8266 возможно осуществлять через облако. Модуль при этом должен находиться в режиме 1 или 3 и быть подключен к точке доступа с выходом в Интернет. Для обновления прошивки надо переключить модуль в режим прошивки, «подтянув» GPIO0 на «минус» и выполнить команду:

```
AT+CIUPDATE
```

После прошивки эту «подтяжку» следует отключить.

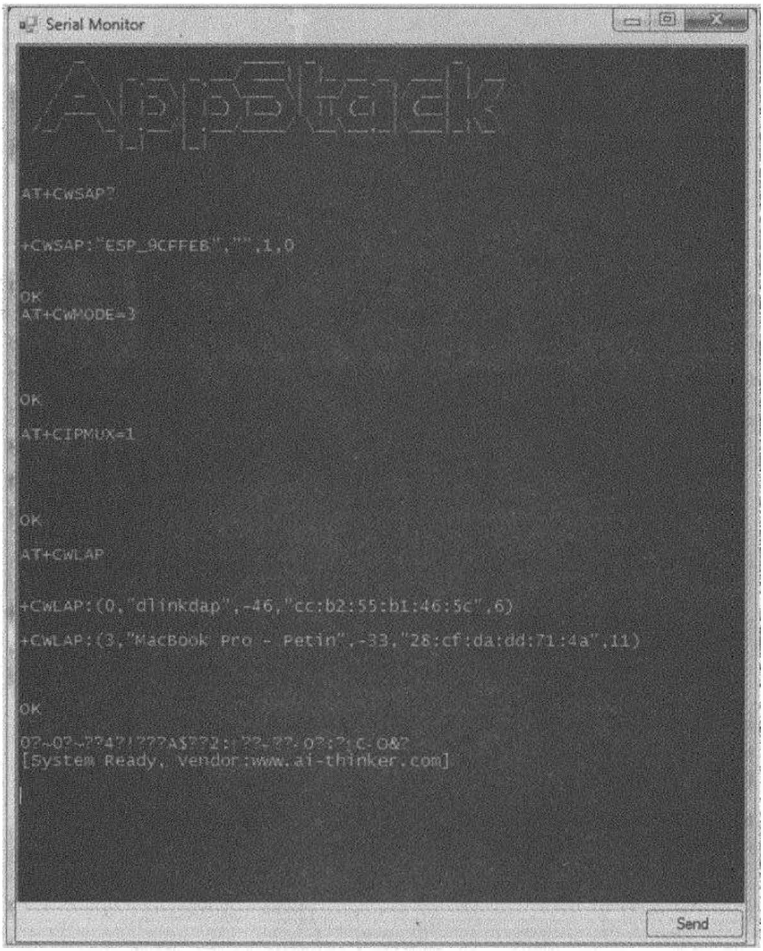


Рис. 14.5. Serial Monitor программы AppStack ESP8266 Config

Следуя этой инструкции, можно обновиться только до официальных версий. Есть и другой вариант обновления прошивки — вручную. Для этого можно воспользоваться программами XTCOM UTIL, ESP8266 Flasher, NodeMCU Flasher и рядом других, доступных для свободной закачки с российского сайта поддержки платы ESP8266 (**http://esp8266.ru**).

14.2. Прошивка NodeMCU

Прошивка NodeMCU способна интерпретировать команды языка Lua — скриптового языка программирования, разработанного подразделением Tecgraf Католического университета Рио-де-Жанейро. Интерпретатор этого языка распространяется свободно, с открытыми исходными текстами на языке Си.

Прошивка NodeMCU, кроме интерпретирования команд языка Lua, может также создавать файлы во Flash-памяти модуля ESP8266 и выполнять их. Роль запускающего файла прошивки (autorun) исполняет при этом файл init.lua. С помощью lua-команд можно:

- подключаться к точке доступа Wi-Fi;
- выступать в роли точки доступа Wi-Fi;
- уходить в глубокий сон для снижения энергопотребления (недоступно для AT-команд);
- привязать lua-функцию к кнопке на GPIO16 (недоступно для AT-команд в стандартной прошивке, частично доступно в некоторых кастомных);
- включать/выключать светодиод на GPIO16 (недоступно для AT-команд в стандартной прошивке, доступно в некоторых кастомных);
- перенаправлять вывод — в примерах использования прошивки есть Telnet-сервер (недоступно для AT-команд);
- создавать, записывать, читать, выполнять, искать, удалять, выводить списком файлы Flash-памяти (недоступно для AT-команд);
- в режиме startsmart автоматически находить открытую сеть Wi-Fi и подключаться к ней (недоступно для AT-команд);
- выводить свой MAC-адрес (недоступно для AT-команд в стандартной прошивке, доступно в некоторых кастомных);
- управлять пользовательским таймером (недоступно для AT-команд);
- управлять таймером WatchDog (недоступно для AT-команд в стандартной прошивке, частично доступно в некоторых кастомных);
- управлять (запись, чтение, триггер) выводами GPIO1–GPIO5, GPIO10, GPIO12–GPIO15 (недоступно для AT-команд);
- обеспечивать PWM (ШИМ) на выводах GPIO1–GPIO5, GPIO10, GPIO12–GPIO15 (недоступно для AT-команд);
- использовать TCP/IP-сокеты;

- обеспечивать режим веб-сервера;
- осуществлять адресацию, запись, чтение по шине I^2C — (недоступно для AT-команд);
- выполнять 10-битное преобразование АЦП на выводе TOUT (недоступно для AT-команд).

Чтобы воспользоваться прошивкой NodeMCU, надо скачать ее последнюю версию с сайта **https://github.com/nodemcu/nodemcu-firmware/tree/master/pre_build** и прошить ее на плату, например, с помощью утилиты ESP8266 Flasher (рис. 14.6), которую можно скачать со страницы **http://esp8266.ru/downloads/esp8266-utils/#wpfb-cat-3**.

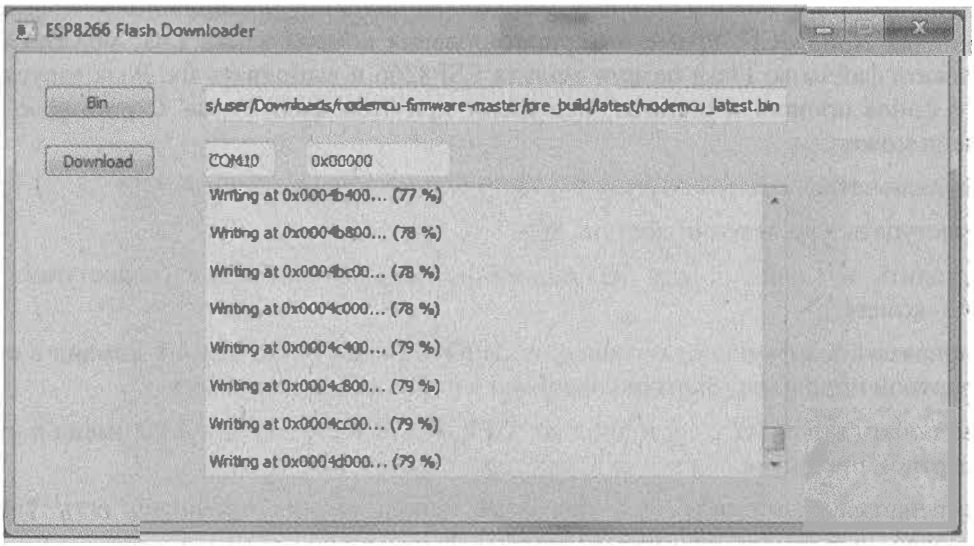


Рис. 14.6. Прошивка модуля программой esp8266_flasher_win

Затем со страницы **http://esp8266.ru/esplorer-ide-esp8266** скачиваем комплект средств разработки (SDK) и запускаем ESPlorer (рис. 14.7). Средство ESPlorer отличается от других программ для ESP8266 тем, что:

- работает на множестве платформ;
- поддерживает несколько открытых файлов;
- обеспечивает подсветку кода языков Lua и Python;
- имеет режимы Undo/Redo;
- поддерживает цветовые темы редакторов: dark, Eclipse, IDEA, Visual Studio;
- осуществляет автозавершение кода по нажатию комбинации клавиш <Ctrl>+<Space>;
- обеспечивает «умную» отправку файлов с ожиданием ответа;
- поддерживает несколько прошивок одновременно.

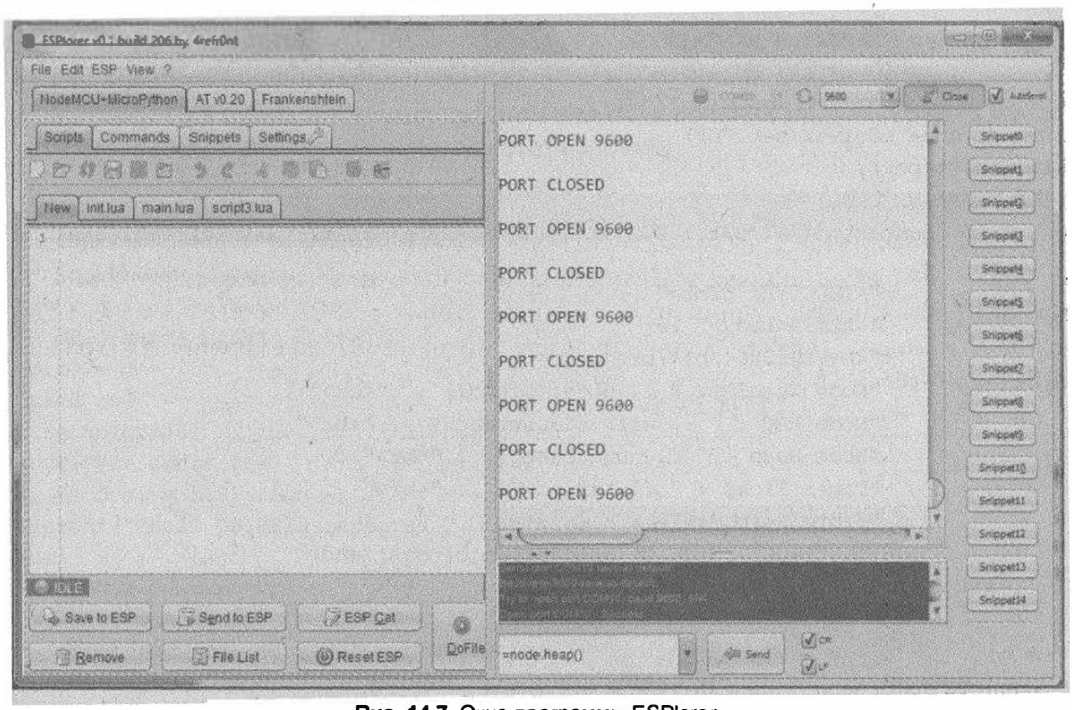


Рис. 14.7. Окно программы ESPlorer

Как уже было отмечено, обеспечивающий autorun скрипт init.lua автоматически стартует при перезагрузке. Создадим этот скрипт и внесем в него код для автоматического подключения модуля в качестве клиента к беспроводной сети (листинг 14.1), а для модуля установим режим `client+AP`.

Листинг 14.1

```
wifi.setmode(3)
print('set mode=STATION (mode=',wifi.getmode(),')')
print('MAC=',wifi.sta.getmac())
print('set wifi')
wifi config start
wifi.sta.config("dlinkdap","")
wifi config end
```

Загрузим скрипт в модуль, нажав на кнопку **Save to ESP** (см. рис. 14.7), перезагрузим модуль и убедимся, что он подключен в качестве клиента к беспроводной сети.

14.2.1. Запуск веб-сервера

Теперь напишем скрипт создания простейшего веб-сервера, чтобы при обращении к модулю по HTTP с него выдавалась информация. Создадим для этого файл server1.lua и запишем в него код, представленный в листинге 14.2.

Листинг 14.2

```
port = 80
srv=net.createServer(net.TCP)
srv:listen(port,
     function(conn)
  \       conn:send("HTTP/1.1 200 OK\nContent-Type: text/html\nRefresh: 5\n\n"
..
             "<!DOCTYPE HTML>" ..
             "<html><body>" ..
             "<b>ESP8266</b></br>" ..
             "Node ChipID : " .. node.chipid() .. "<br>" ..
             "Node MAC : " .. wifi.sta.getmac() .. "<br>" ..
             "Node Heap : " .. node.heap() .. "<br>" ..
             "Timer Ticks : " .. tmr.now() .. "<br>" ..
             "</html></body>")
         conn:on("sent",function(conn) conn:close() end)
     end
)
```

Сохраним файл server1.lua в модуле и запустим.

> ***Внимание!***
>
> Для запуска сервера при загрузке модуля необходимо в конце нашего autorun-файла init.lua добавить строку:
>
> `dofile(server1.lua)`

А для проверки работы сервера подключимся к точке доступа модуля и наберем в браузере ее адрес: **http://192.168.4.1** (рис. 14.8).

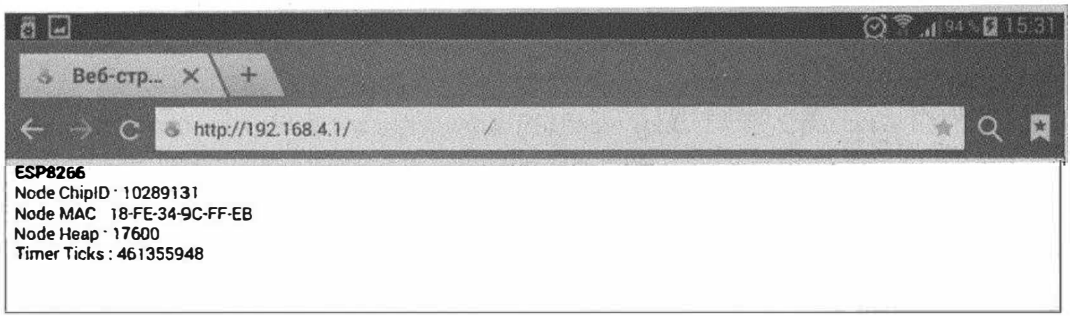


Рис. 14.8. Обращение к серверу на ESP8266

14.2.2. Подключение к ESP8266 модулей датчиков средствами языка Lua

Важный момент — в прошивке NodeMCU присутствуют модули датчиков, которые можно подключать к скриптам командой `require()`. Количество таких модулей по-

стоянно увеличивается, и они доступны для просмотра и закачки на странице **https://github.com/nodemcu/nodemcu-firmware/tree/master/lua_modules**.

В листинге 14.3 приведен пример скрипта для подключения модуля датчика температуры DS18B20, получения и вывода его значений.

Листинг 14.3

```
-- подключение модуля
t = require("ds18b20")
-- GPIO карта модуля ESP-01
gpio0 = 3
gpio2 = 4
t.setup(gpio0)
addrs = t.addrs()
if (addrs ~= nil) then
  print("Total DS18B20 sensors: "..table.getn(addrs))
end
-- чтение температуры
print("Temperature: "..t.read().."'C")
-- освободить память после использования
t = nil
ds18b20 = nil
package.loaded["ds18b20"]=nil
```

Результат выполнения этого скрипта при подключении датчика температуры к выводу GPIO0 приведен на рис. 14.9.

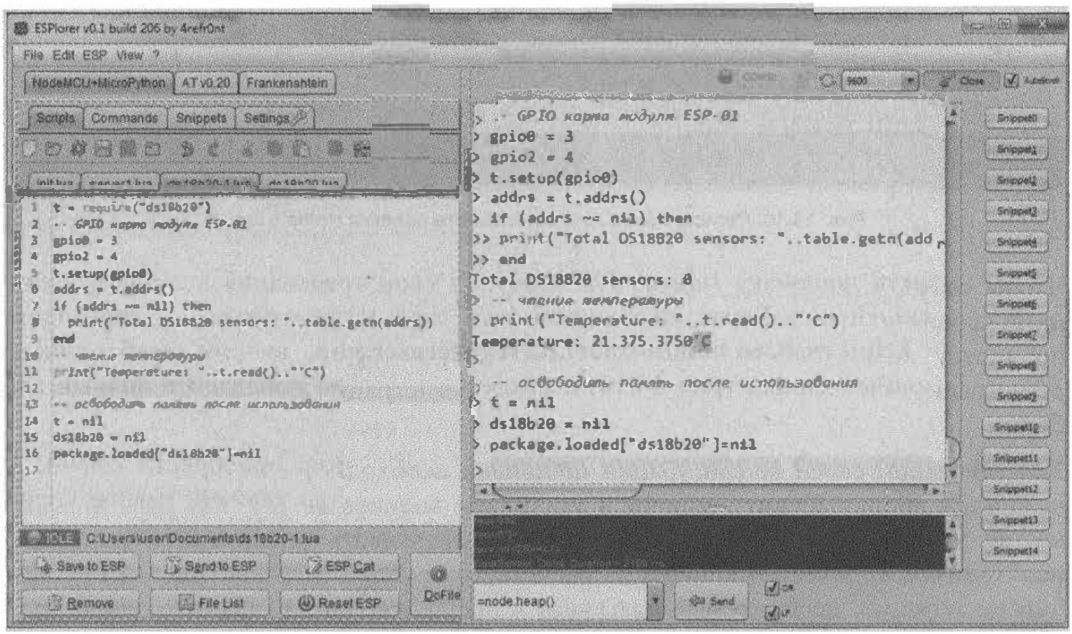


Рис. 14.9. Получение и вывод значений датчика температуры DS18B20 с помощью модуля ESP8266

14.3. Проект Home`s Smart

Мы рассмотрим здесь один из проектов, в котором наиболее полно реализованы функции Интернета вещей, — российский проект Home`s Smart («Умный дом») (**http://homes-smart.ru**), обеспечивающий, в том числе, сопряжение различных датчиков с беспроводным Wi-Fi модулем ESP8266 для отправки данных на удаленные серверы.

Последняя на момент написания книги прошивка модуля ESP8266 (0.1.1 Конструктор) поддерживает чтение датчиков DHT11/22, BMP085/180, BH1750, DS18B20, AM2321 и др. и умеет передавать их показания на сайт «Народный мониторинг», на сервис учета метрик ThingSpeak.com, на MQTT-сервер, в систему «умного дома» MajorDoMo, в систему компьютерного контроля «Бенукс». Имеется в этой прошивке и возможность управлять состоянием свободных GPIO для передачи команд нагрузкам (реле), читать состояние входов GPIO, выводить данные на LCD- или OLED-экран, поддерживается также множество и других функций.

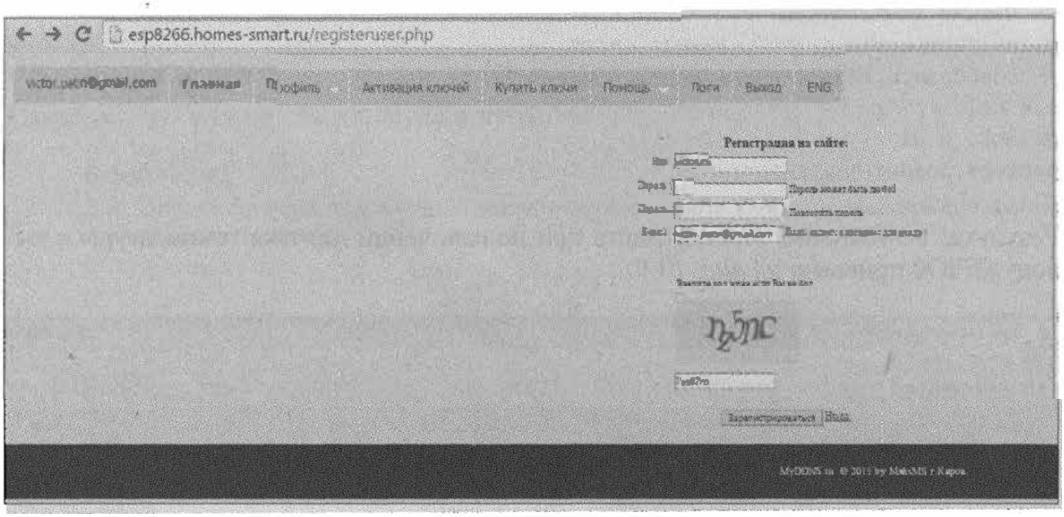


Рис. 14.10. Регистрация в личном кабинете проекта Home`s Smart

Чтобы собрать прошивку модуля ESP8266 под свои требования в личном кабинете проекта Home`s Smart, необходимо там зарегистрироваться, — переходим по адресу **http://esp8266.homes-smart.ru/registeruser.php**, вводим необходимые регистрационные данные (рис. 14.10) и после регистрации попадаем в личный кабинет.

Для генерации своей конфигурации прошивки необходимо приобрести ключи — нажимаем кнопку **Купить ключи** и указываем количество ключей, равное количеству устройств, которое должна будет поддерживать прошивка. Оплачиваем необходимое количество ключей из расчета 100 рублей за ключ (рис. 14.11), выбираем позиции, необходимые для учета в прошивке (рис. 14.12), и создаем прошивку.

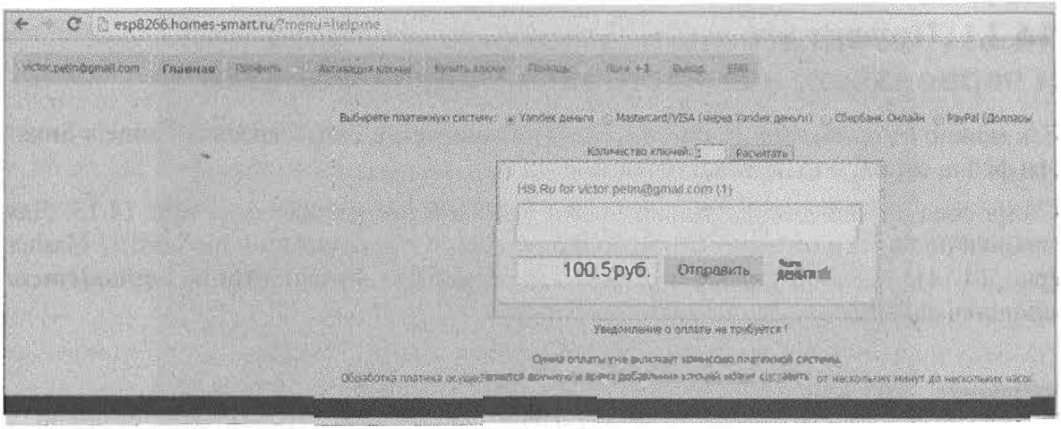


Рис. 14.11. Приобретение ключа для генерации собственной прошивки

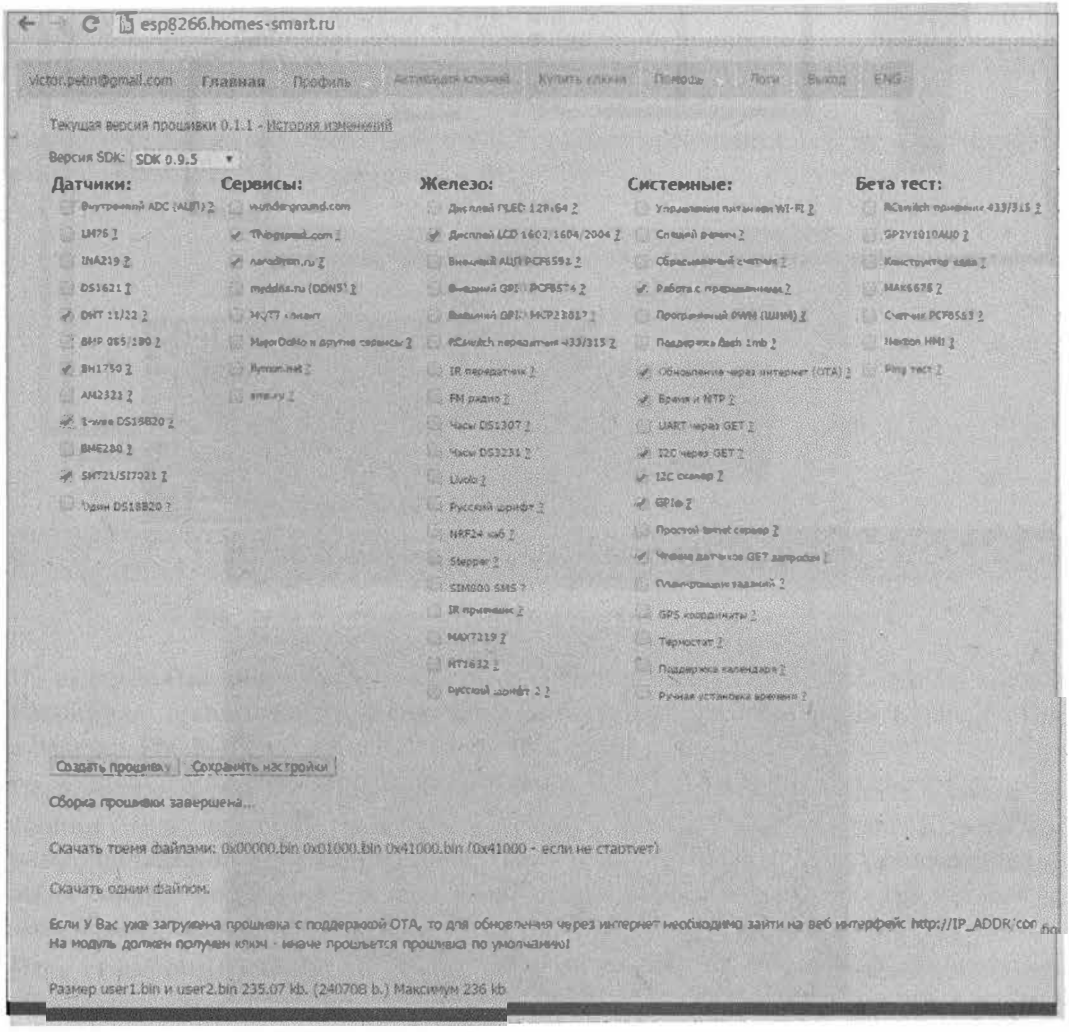


Рис. 14.12. Выбор опций для генерации и генерация прошивки

14.3.1. Прошивка и первоначальная настройка модуля ESP8266

Как можно было видеть на рис. 14.12, мы скачиваем с сайта проекта Home`s Smart три файла собранной нами прошивки.

Схема соединений для прошивки платы ESP8266 представлена на рис. 14.13. Для заливки прошивки на плату ESP8266 мы воспользуемся утилитой NodeMCU Flasher (рис. 14.14), скачать которую можно со страницы **https://github.com/nodemcu/nodemcu-flasher**.

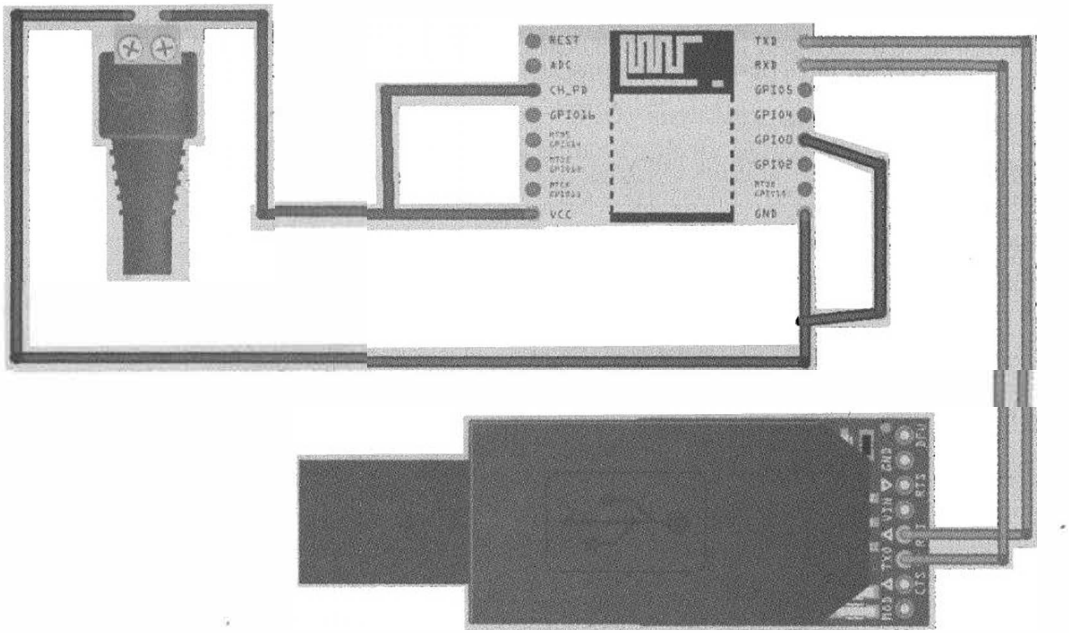


Рис. 14.13. Схема соединений для прошивки модуля ESP8266

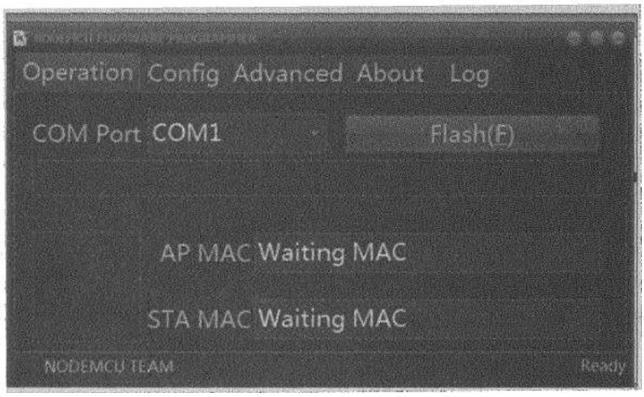


Рис. 14.14. Утилита NodeMCU Flasher

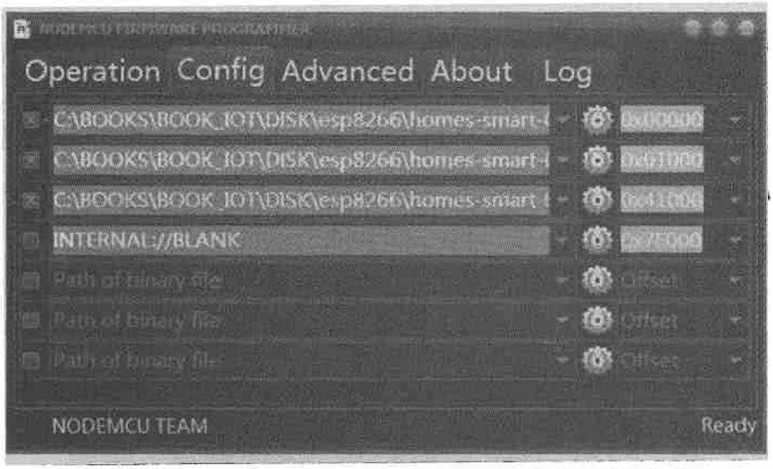


Рис. 14.15. Настройки вкладки **Config** утилиты NodeMCU Flasher

На вкладке **Config** утилиты NodeMCU Flasher указываем файлы прошивки и адреса смещения (рис. 14.15).

На вкладке **Advanced** утилиты NodeMCU Flasher необходимо всегда устанавливать размер: **Flash size 512 kByte** (рис. 14.16).



Рис. 14.16. Настройки вкладки **Advanced** утилиты NodeMCU Flasher

На вкладке **Operation** выбираем порт подключения платы и нажимаем на кнопку **Flash(F)** — процесс загрузки прошивки на плату ESP8266 начнется (рис. 14.17), а надпись **Flash(F)** на кнопке сменится на **Stop(S)**.

Первоначальную настройку модуля ESP8266 лучше всего производить в так называемом *безопасном режиме* (Safe mode). Необходим такой режим и если по каким-либо причинам был потерян пароль к настройкам. Для активации безопасного режима следует перед включением модуля соединить вместе выводы RX и TX, при этом к ним не должно быть ничего подключено.

Итак, переводим модуль ESP8266 в безопасный режим, подключаемся к точке доступа проекта Home's Smart и набираем в браузере: `http://192.168.4.1`. На странице настроек выбираем пункт **Main** и вводим данные для подключения модуля к сети

Wi-Fi (рис. 14.18). Далее обновляем страницу и видим внизу IP-адрес, на который уже можно будет заходить внутри вашей локальной сети. Там же вы можете установить свои логин и пароль на страницу настроек веб-интерфейса — длина логина и пароля не должна превышать 8 символов (по умолчанию логин: **esp8266**,

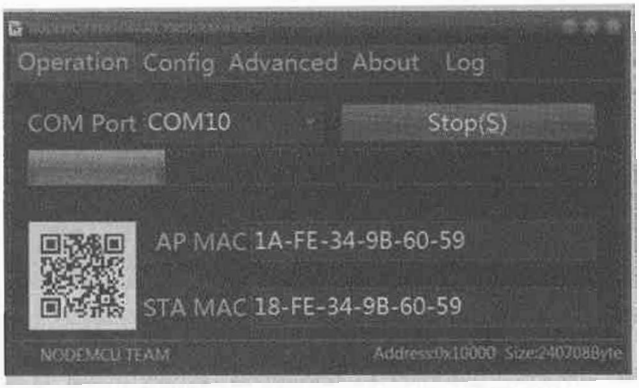


Рис. 14.17. Загрузка прошивки на плату ESP8266

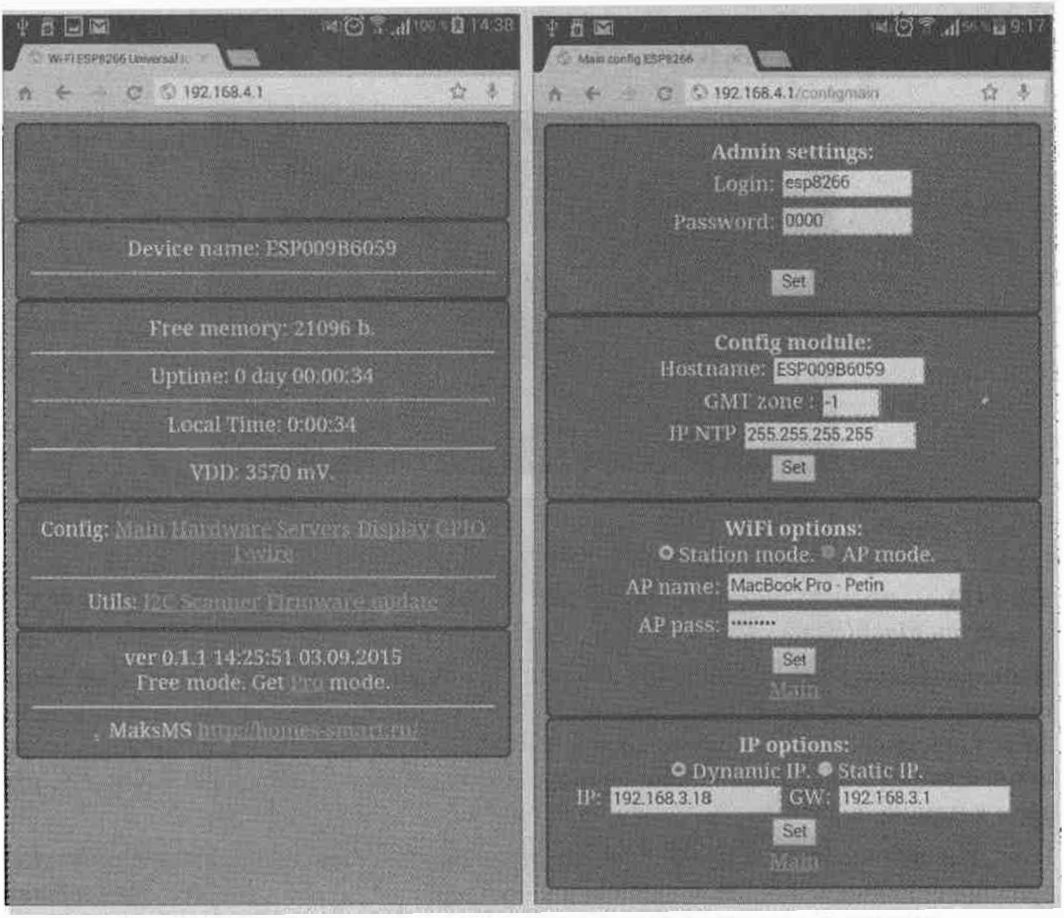


Рис. 14.18. Первоначальная настройка модуля в безопасном режиме (Safe mode)

пароль: **0000**). Здесь же можно задать имя модуля, которое будет отображаться на главной странице и в системе flymon, а так же в пути на сервере MQTT.

Теперь активируем версию прошивки Pro (она позволяет подключить к модулю ESP8266 до 10 датчиков DS18B20, работающих по протоколу 1-Wire, два датчика DHT11 (DHT22) и т. п.) — для этого необходимо приобрести специальный ключ: заходим в личный кабинет на сайте **http://esp8266.homes-smart.ru/**, выбираем раздел **Активация ключей** и нажимаем в нем кнопку **Добавить ключ**. В соответствующее поле открывшейся формы вводим ID нашего модуля и нажимаем кнопку **OK** (рис. 14.19) — будет сформирован ключ для выбранного модуля (рис. 14.20), который необходимо ввести в веб-интерфейсе модуля (рис. 14.21), чего достаточно для активации версии Pro.

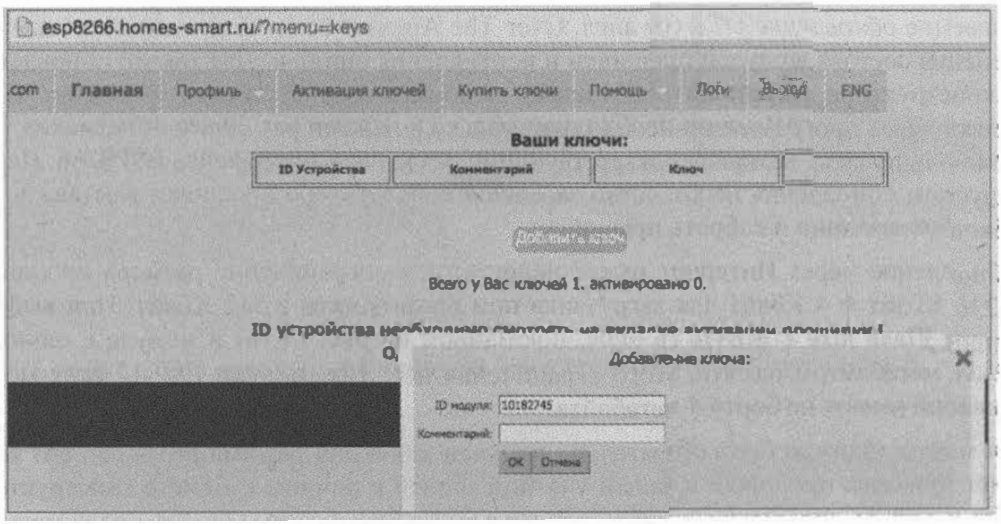


Рис. 14.19. Добавление ключа: ввод ID модуля

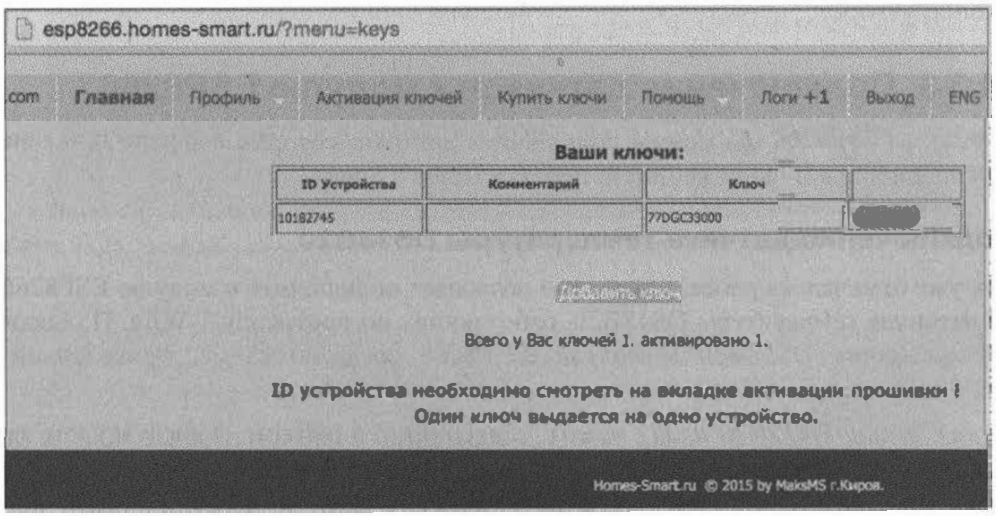


Рис. 14.20. Генерация ключа для модуля ESP8266

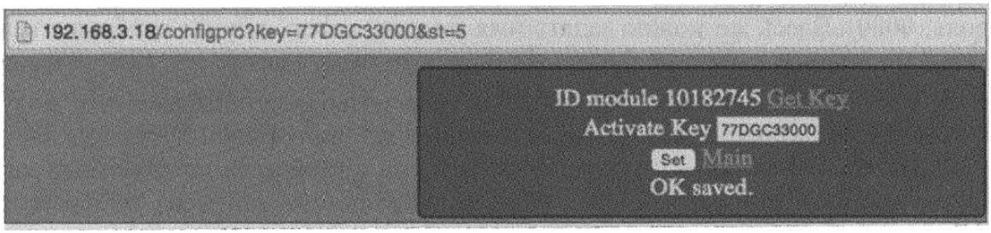


Рис. 14.21. Активация версии Pro: ввод полученного ключа

14.3.2. Обновление прошивки через Интернет

Модуль ESP8266 может получать обновления прошивки через Интернет — так называемое *обновление OTA* (от англ. Over The Air, «по воздуху»), например, для изменения состава модулей прошивки и исправления ошибок. Для такого обновления в конструкторе прошивки необходимо включить соответствующую опцию. Прошивка через программатор необходима только в первый раз, далее обновления уже можно получать, заходя в соответствующий раздел веб-интерфейса ESP8266. Перед запуском обновления необходимо заранее в конструкторе прошивки выставить необходимые опции и собрать прошивку.

Обновление через Интернет имеет недостаток — ограничение размера прошивки в 236 Кбайт + 4 Кбайт для загрузчика при Flash-памяти в 512 Кбайт. При выборе опции **Flash size 1 mByte** (в окне, показанном на рис. 14.16) и модуле с одним и более мегабайтом памяти, этого ограничения нет. Так, модули ESP-12 всех модификаций имеют на борту 4 мегабайта.

На модуль должен быть обязательно получен ключ для режима Pro — по его коду идет привязка прошивки к вашей учетной записи в личном кабинете конструктора (без ключа будет загружена облегченная версия прошивки). Однако после получения ключа из облегченной версии можно обновиться до полной версии, созданной в конструкторе.

14.3.3. Подключение датчиков к модулю ESP8266

К модулю ESP8266 мы можем подключить датчики, которые выбрали при генерации прошивки в личном кабинете проекта Home`s Smart.

Подключение датчика температуры DS18B20

Как уже отмечалось ранее, версия Pro позволяет подключить к модулю ESP8266 до 10 датчиков температуры DS18B20, работающих по протоколу 1-Wire. Подключим сейчас датчик DS18B20 к модулю ESP8266 согласно схеме, приведенной на рис. 14.22.

Чтобы датчик DS18B20 начал работу, необходимо в веб-интерфейсе модуля зайти на вкладку **Hardware** и активировать подключение датчика к выбранному выводу GPIO (рис. 14.23), а затем зайти на вкладку **1-Wire** и просканировать адреса (рис. 14.24).

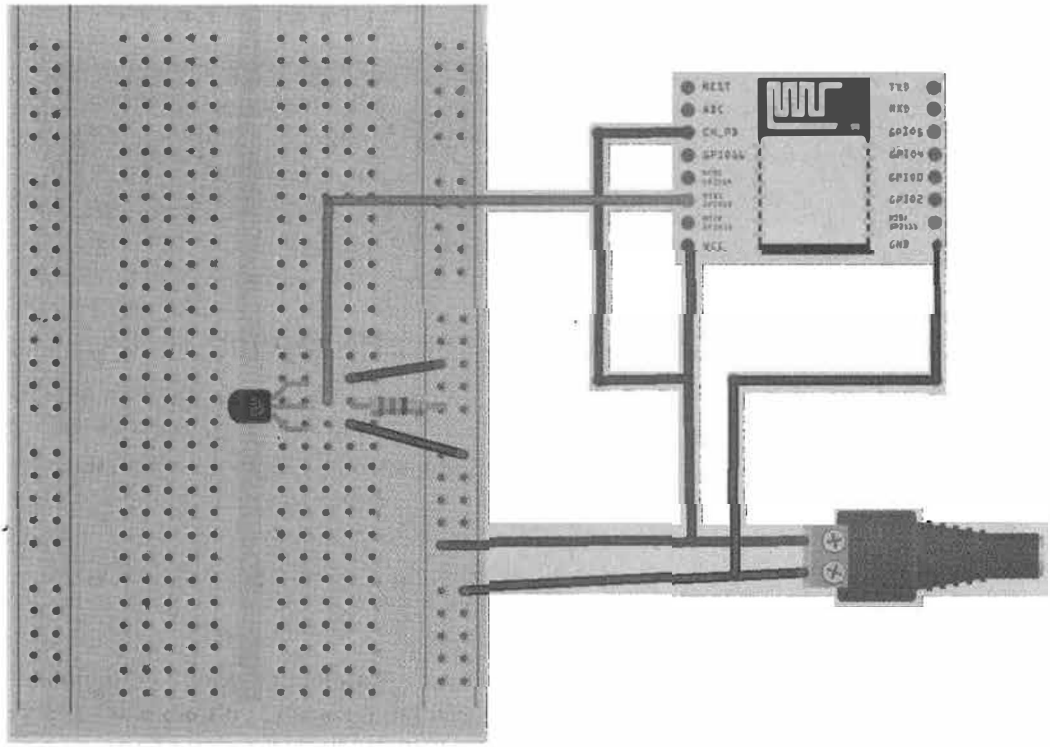


Рис. 14.22. Схема подключения датчика температуры DS18B20 к модулю ESP8266

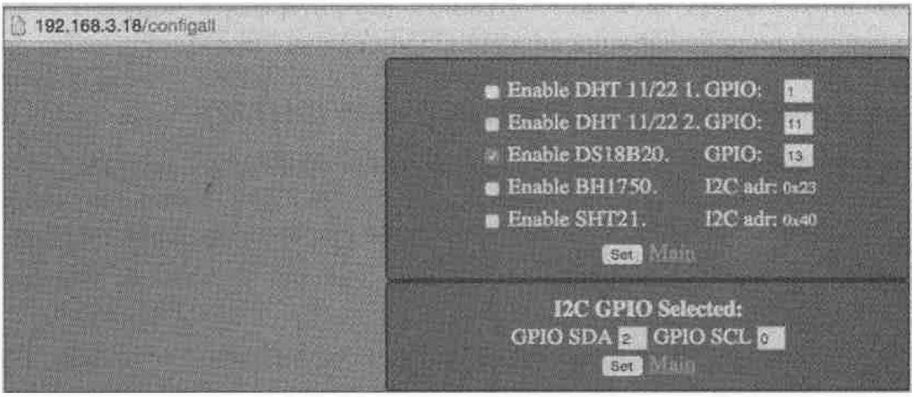


Рис. 14.23. Активация датчика DS18B20 на вкладке **Hardware** веб-интерфейса модуля ESP8266

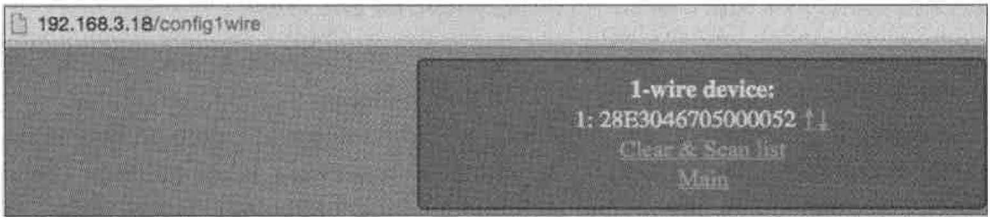


Рис. 14.24. Сканирование адресов 1-Wire

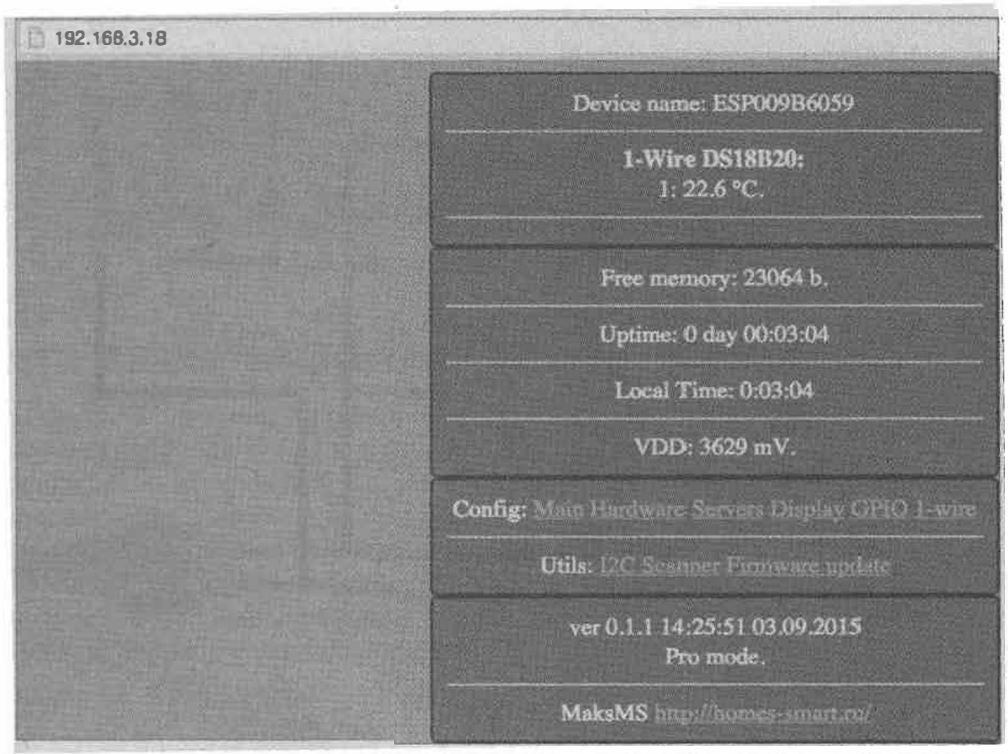


Рис. 14.25. Показания датчика DS18B20 на главной странице веб-интерфейса модуля ESP8266

Через некоторое время на главной странице веб-интерфейса модуля ESP8266 мы можем увидеть вывод показаний датчика DS18B20 (рис. 14.25).

Подключение датчика влажности DHT11 (DHT22)

Как уже отмечалось ранее, версия Pro позволяет подключить к модулю ESP8266 до двух датчиков температуры и влажности DHT11 (DHT22). Подключим сейчас датчик DHT11 (DHT22) к модулю ESP8266 согласно схеме, приведенной на рис. 14.26.

Выводы модуля для подключения датчика указываются на вкладке **Hardware** его веб-интерфейса (рис. 14.27).

Через некоторое время на главной странице веб-интерфейса модуля ESP8266 (рис. 14.28) мы можем увидеть вывод показаний датчика DHT11 (DHT22).

Подключение I²C-датчика освещенности BH1750

В *главе 6* мы уже знакомились с I²C-датчиками. Подключим сейчас один из них — датчик освещенности BH1750 — к модулю ESP8266.

Чтобы датчик BH1750 начал работу, необходимо в веб-интерфейсе модуля зайти на вкладку **Hardware** и активировать подключение датчика к выбранным выводам GPIO модуля ESP8266 (рис. 14.29). Для I²C-интерфейса мы задействуем здесь выводы GPIO0 (SCL) и GPIO2 (SDA).

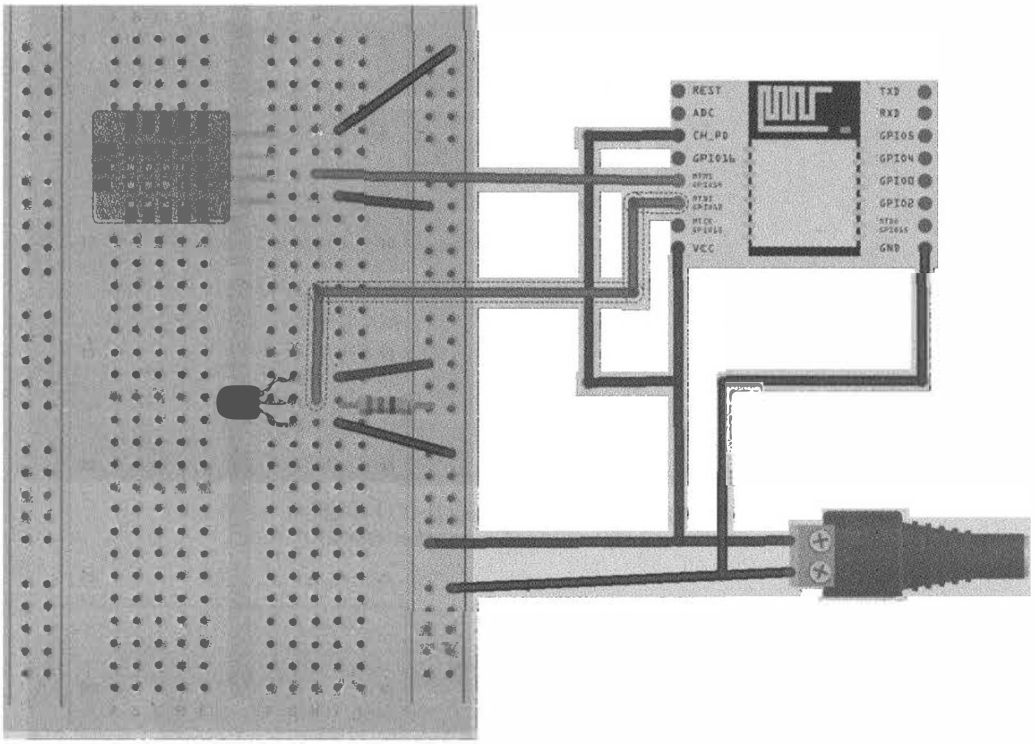


Рис. 14.26. Схема подключения датчика температуры и влажности DHT11 (DHT22) к модулю ESP8266

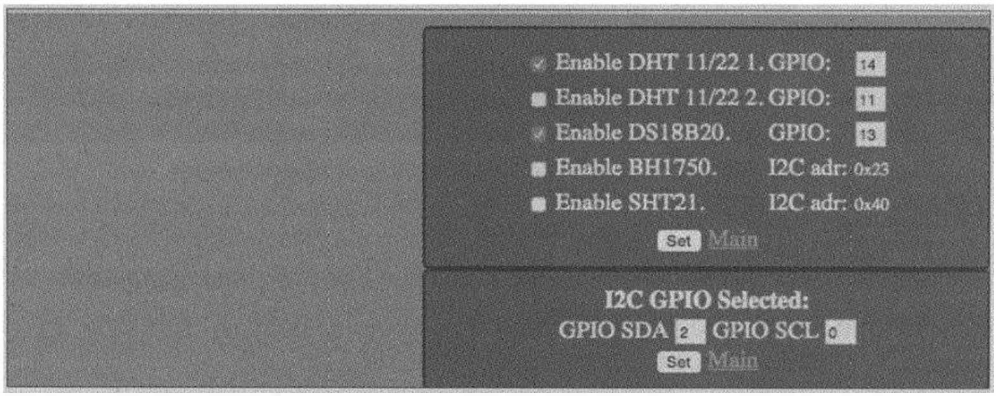


Рис. 14.27. Активация датчика DHT11 (DHT22) на вкладке **Hardware**

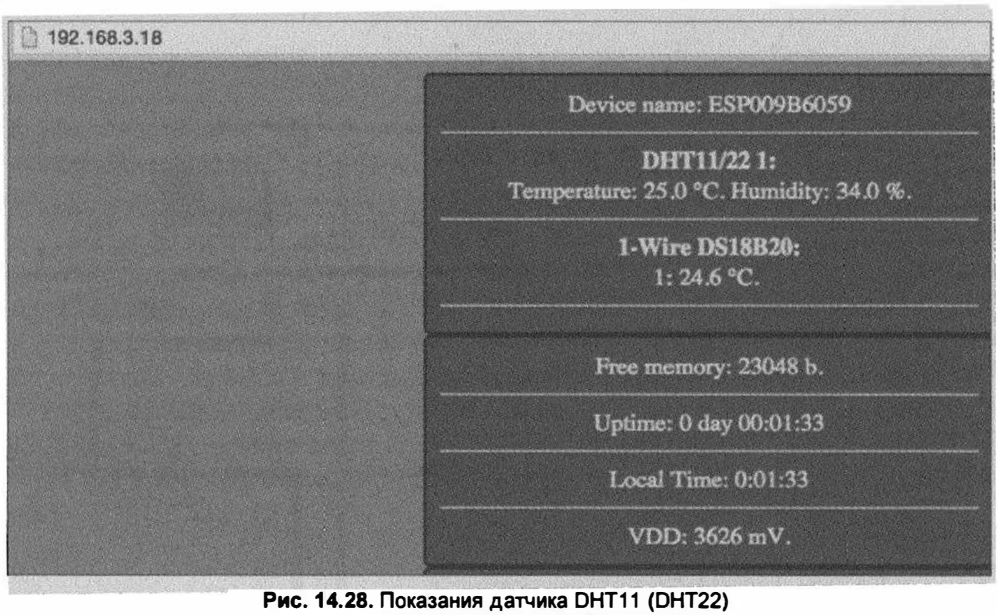


Рис. 14.28. Показания датчика DHT11 (DHT22) на главной странице веб-интерфейса модуля

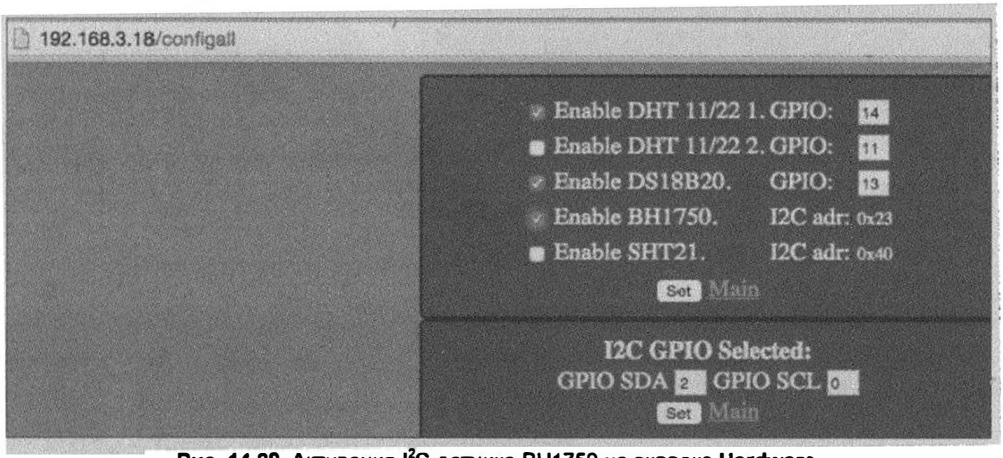


Рис. 14.29. Активация I²C-датчика BH1750 на вкладке **Hardware**

Схема подключения I²C-датчика BH1750 к модулю ESP8266 показана на рис. 14.30.

Подключив датчик, просканируем подключенные к модулю ESP8266 I²C-устройства с помощью опции I²C-сканирования (рис. 14.31).

Через некоторое время показания датчика освещенности BH1750 появятся на главной странице веб-интерфейса нашего модуля (рис. 14.32).

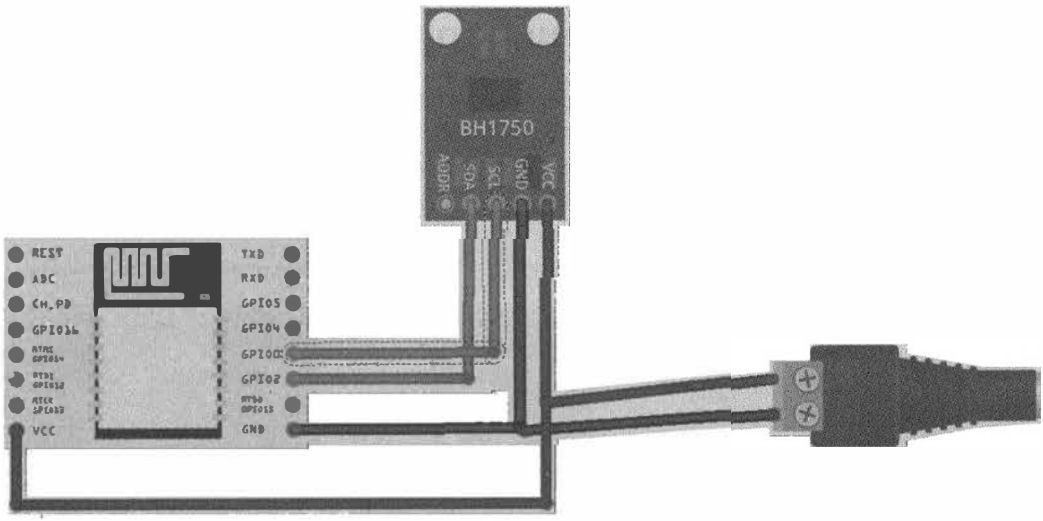


Рис. 14.30. Схема подключения I²C-датчика BH1750 к модулю ESP8266

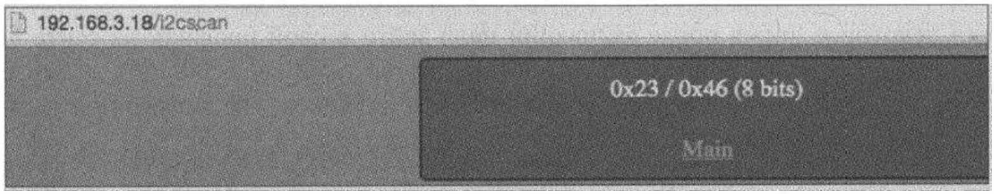


Рис. 14.31. Вывод списка адресов подключенных к модулю ESP8266 I²C-устройств опцией I²C-сканирования

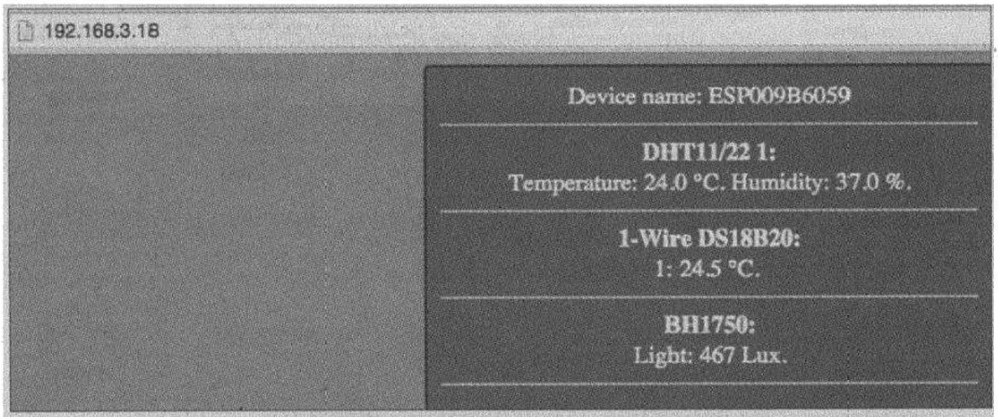


Рис. 14.32. Показания датчика BH1750 на главной странице веб-интерфейса модуля ESP8266

14.4. Отправка данных с модуля ESP8266 на сайт «Народный мониторинг»

В предыдущих главах книги мы уже рассматривали отправку различных данных на сайт «Народный мониторинг» (**http:www.narodmon.ru**). Для отправки туда данных с модуля ESP8266 следует открыть вкладку **Servers** веб-интерфейса модуля и активировать там соответствующую позицию (рис. 14.33).

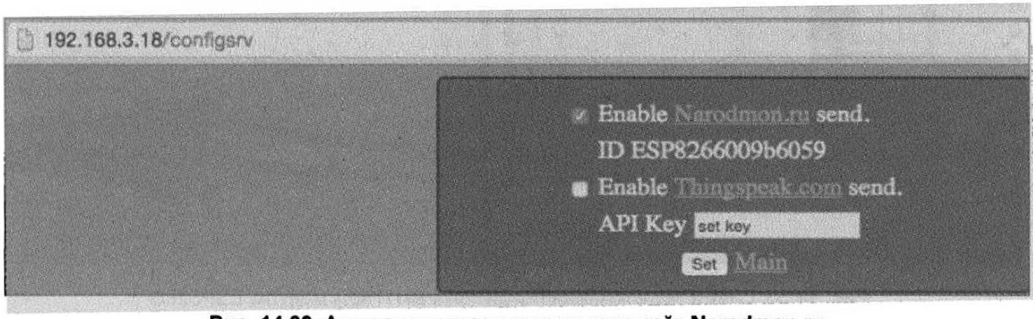


Рис. 14.33. Активация отправки данных на сайт **Narodmon.ru**

Через некоторое время после активации надо зайти в свой профиль на сайте «Народный мониторинг» и добавить наше устройство (модуль ESP8266): вводим ID устройства (рис. 14.34) и попадаем в список его датчиков, где можно при необходимости подредактировать имеющиеся там данные (рис. 14.35).

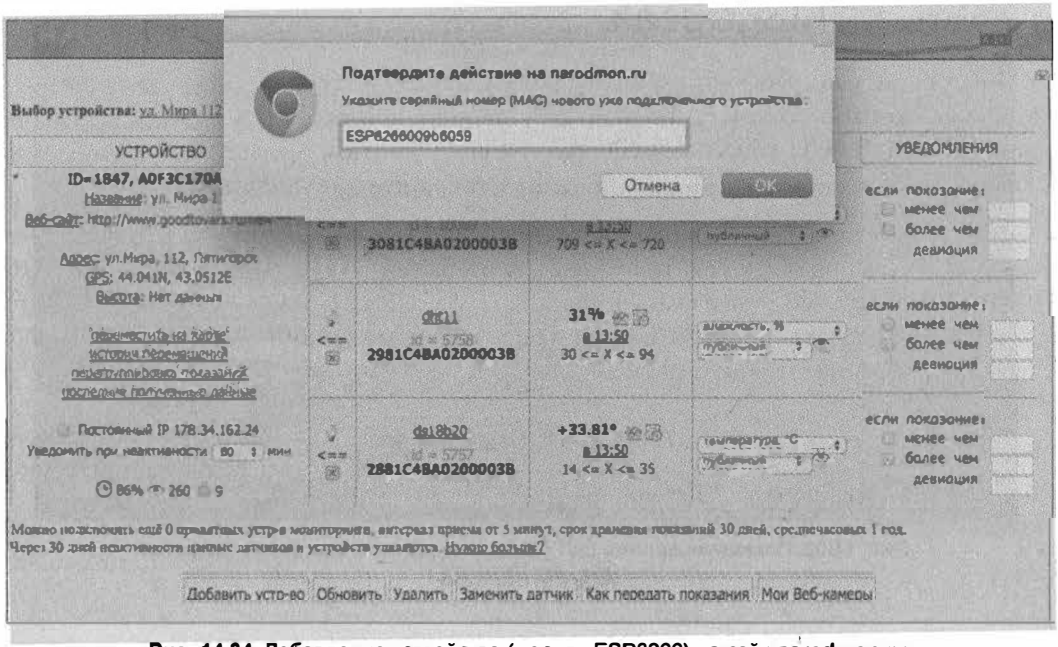


Рис. 14.34. Добавление устройства (модуля ESP8266) на сайт **narodmon.ru**

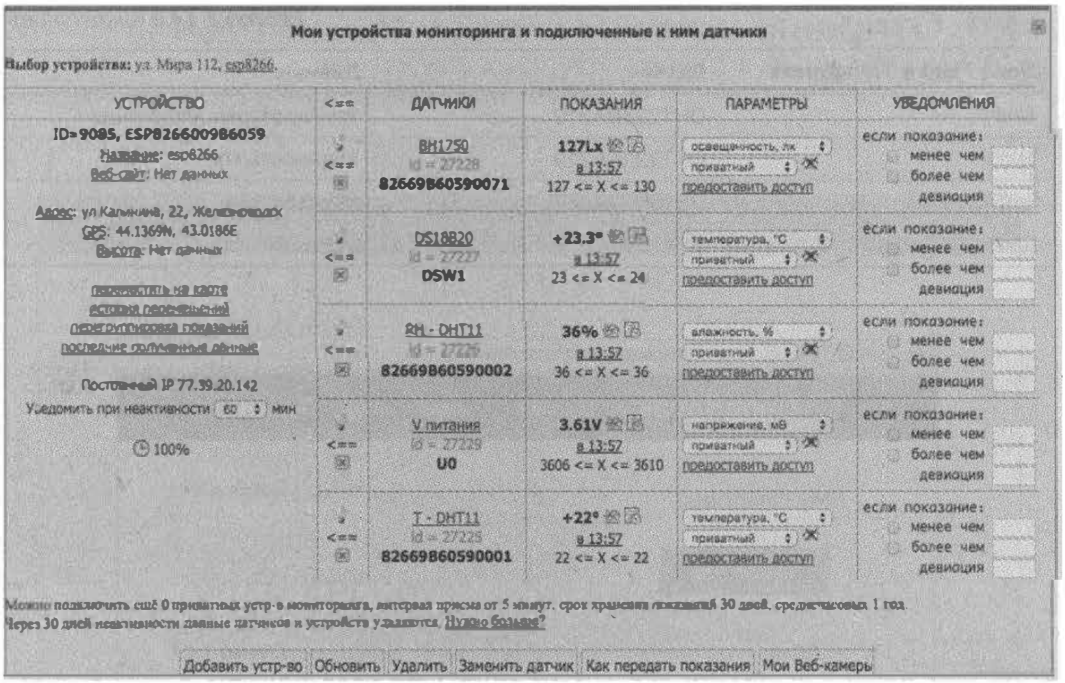


Рис. 14.35. Список датчиков модуля ESP8266, передающих показания на сайт **narodmon.ru**

14.5. Отправка данных с модуля ESP8266 в сервис ThingSpeak

В *главе 4* мы уже рассматривали отправку данных в сервис ThingSpeak (**https://thingspeak.com**) — открытую платформу данных для проектов Internet of Things, включающую в себя сбор данных с датчиков в реальном времени, обработку этих данных, их визуализацию и использование в приложениях и плагинах.

Напомню, что при создании прошивки ESP8266 мы предусмотрели отправку данных в сервис ThingSpeak (см. рис. 14.12), и чтобы это стало возможным, необходимо создать в сервисе новый канал для хранения данных: на вкладке **My Channels** нажимаем на кнопку **New Channel** (рис. 14.36) и вводим данные для канала (рис. 14.37) в соответствии с табл. 14.3.

Таблица 14.3. *Данные, отправляемые в канал ThingSpeak*

Поле Field в ThingSpeak	Датчик	Данные
field1	DHT11/AM2321	температура
field2	DHT11/AM2321	влажность
field3	BMP	температура
field4	BMP	давление

Таблица 14.3 (окончание)

Поле Field в ThingSpeak	Датчик	Данные
field5	DHT22/INA219	температура/напряжение
field6	DHT22/INA219	влажность/ток
field7	DS18B20/LM75	температура
field8	BH1750	освещенность

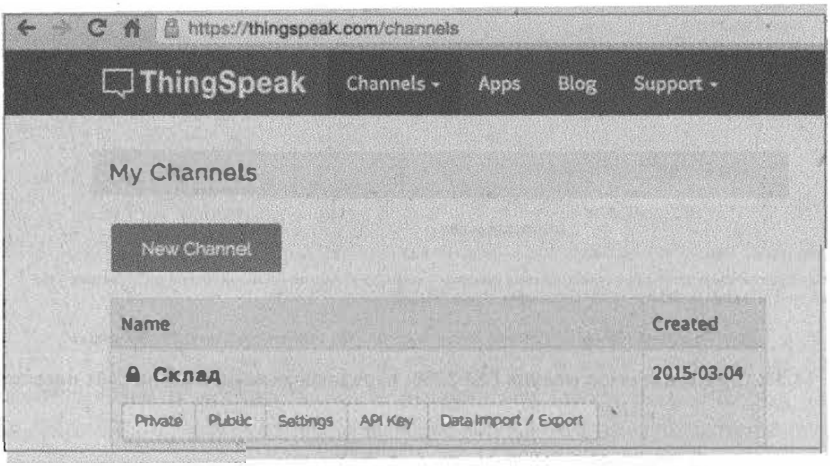


Рис. 14.36. Создание нового канала в сервисе ThingSpeak

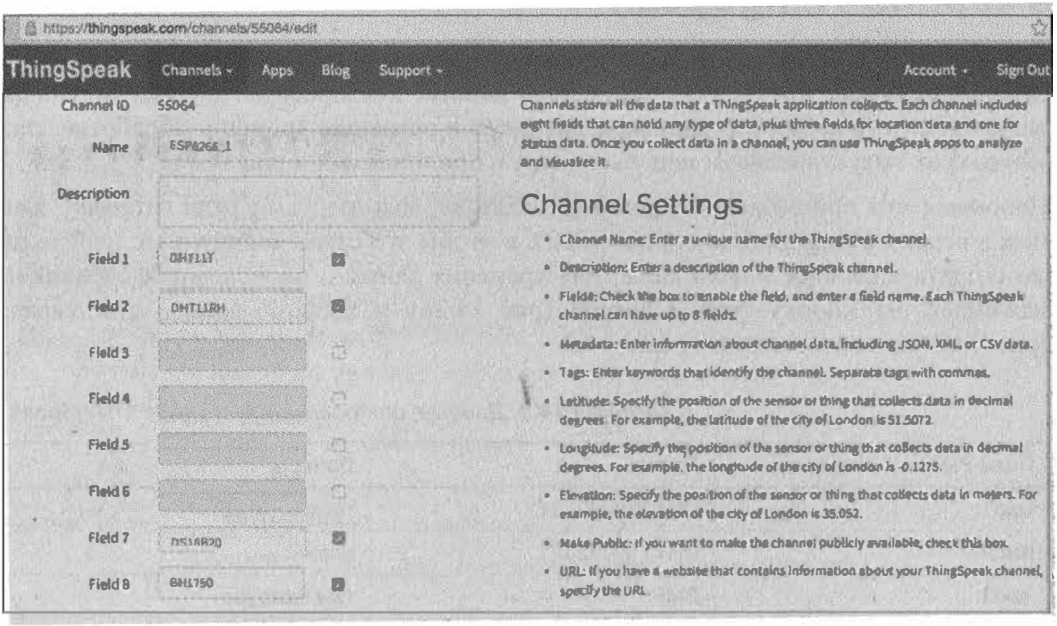


Рис. 14.37. Ввод данных для нового канала в сервисе ThingSpeak

Затем по ссылке **API Keys** сервиса ThingSpeak (см. рис. 14.36) находим ключ **Write API Key** (рис. 14.38) и заносим его в соответствующее поле модуля ESP8266 на вкладке **Servers** его веб-интерфейса (рис. 14.39).

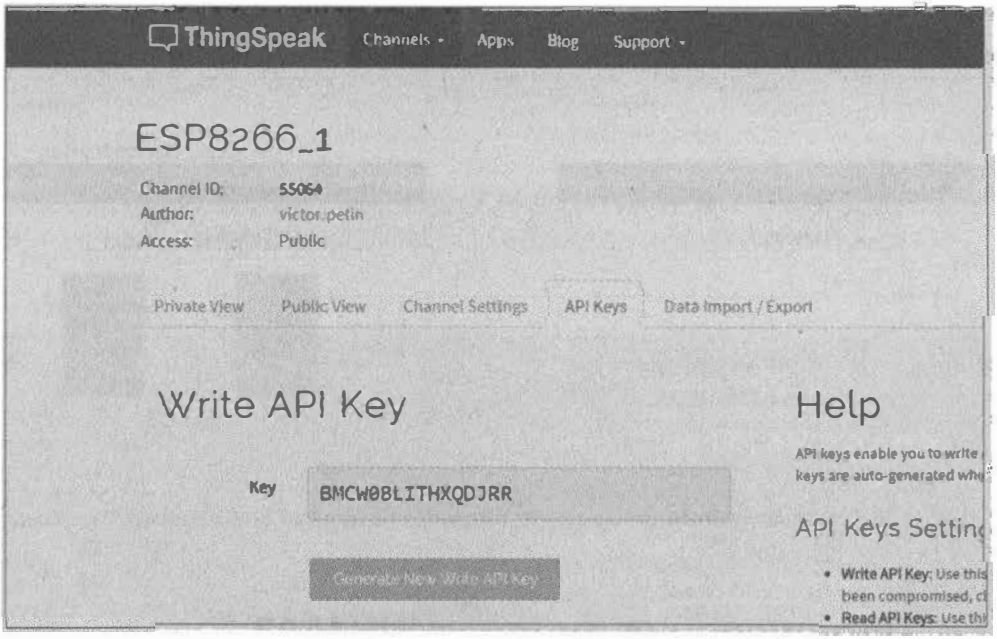


Рис. 14.38. Значение **Write API Key** для канала

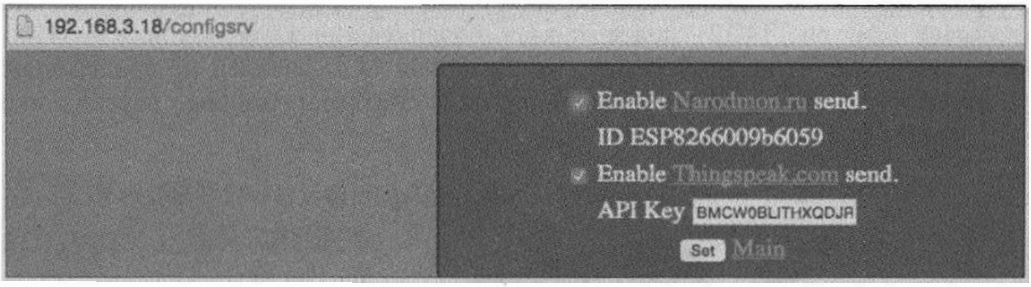


Рис. 14.39. Ввод ключа для сервиса ThingSpeak

Осталось некоторое время подождать, и мы увидим на графиках сервиса ThingSpeak отображение поступающих с датчиков данных в канале **ESP8266_1** (рис. 14.40).

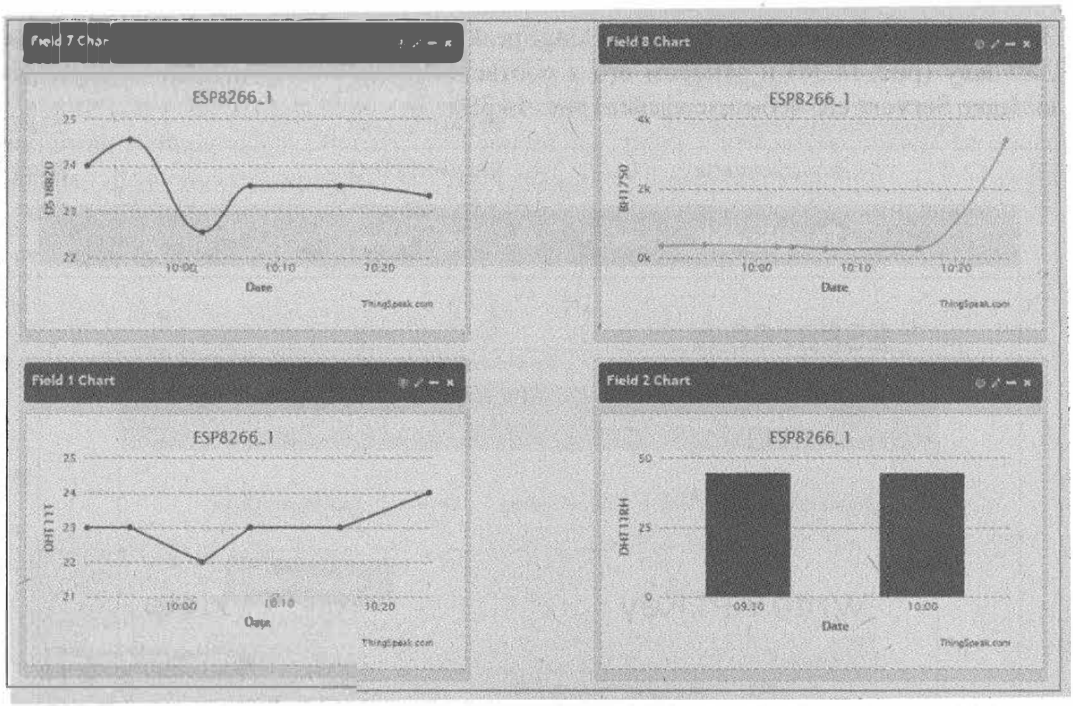


Рис. 14.40. Графическое отображение данных с датчиков модуля ESP8266 в сервисе ThingSpeak

14.6. Подключение дисплея WH1602 к плате модуля ESP8266

К плате модуля ESP8266 можно подсоединить и LCD-дисплей. Созданная нами прошивка (см. рис. 14.12) поддерживает подключение LCD-дисплеев WH1602, 1604, 2004 и OLED по интерфейсу I^2C. Если ваш LCD-дисплей не поддерживает I^2C-интерфейс, вы можете воспользоваться соответствующим переходником (рис. 14.41).

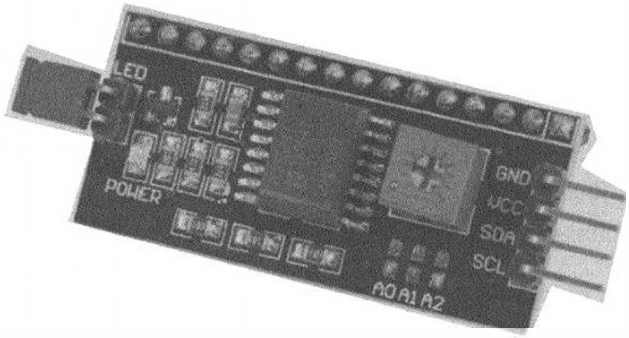


Рис. 14.41. Переходник для I^2C-интерфейса для LCD 1602, 1604, 2004

Адрес у LCD-переходника I^2C по умолчанию 0x27, но его можно поменять в настройках дисплея. У OLED-дисплеев адрес фиксирован: 0x3C. Активация поддержки дисплея осуществляется на вкладке **Display** веб-интерфейса модуля ESP8266, здесь же можно настроить и вывод на дисплей показаний подключенных датчиков (рис. 14.42). При этом, выбрав опцию **Out msg**, мы получим возможность отправлять сообщения на экран дисплея через веб-интерфейс:

```
http://IP_ADRESS/lcdmsg?st=X&txt=text&font=N
```

где:

- `X` — номер строки;
- `text` — сообщение (пробелы заменяем на нижнее подчеркивание);
- `N` — шрифт (для OLED).

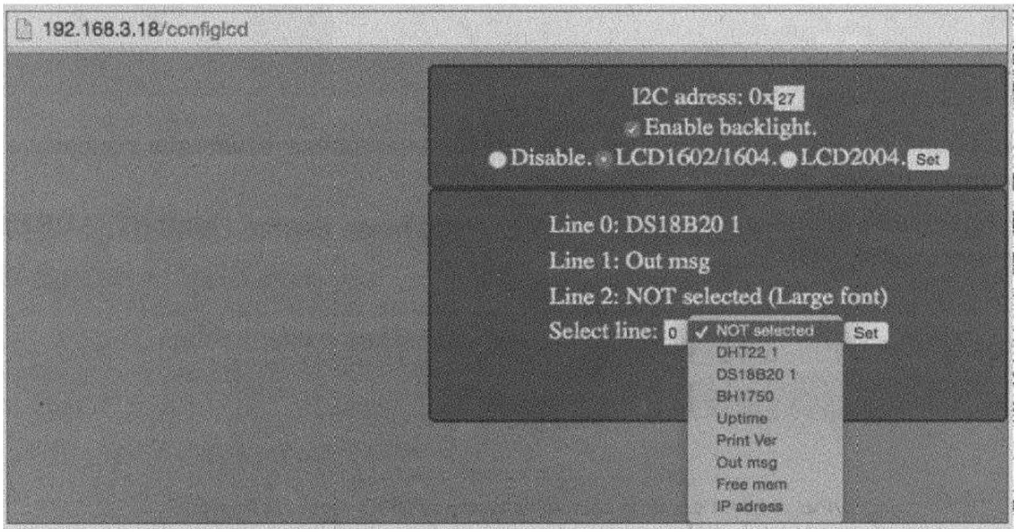


Рис. 14.42. Настройка поддержки дисплея в веб-интерфейсе модуля ESP8266

14.7. Управление выводами GPIO модуля ESP8266

Модуль ESP8266 имеет 16 выводов GPIO, причем на различных версиях плат модуля количество доступных для управления выводов GPIO колеблется от 4 до 10. К выводу GPIO можно подсоединить, например, реле и удаленно управлять его состоянием, или датчик открытия двери или окна (геркон) и наблюдать его состояние через веб-интерфейс.

Чтобы получить доступ к управлению выводами GPIO, необходимо прежде всего в веб-интерфейсе ESP8266 зайти в настройки GPIO и указать, какие выводы вы хотите задействовать, для чего ввести номер GPIO в соответствующее поле ввода, выбрать нужный режим и нажать кнопку **Set** (рис. 14.43).

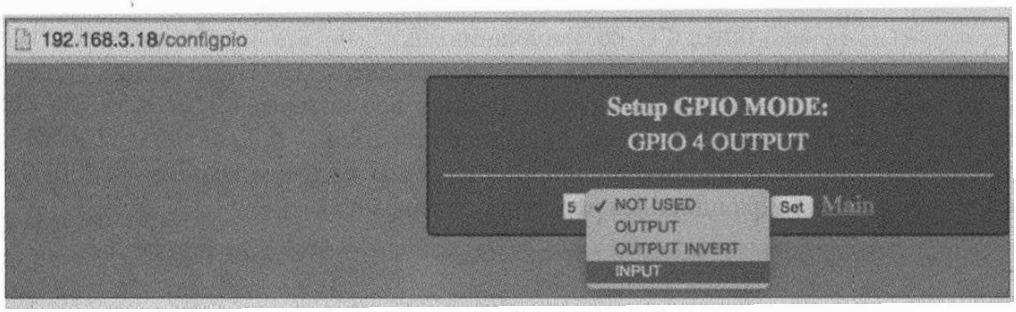


Рис. 14.43. Настройка выводов GPIO

Управлять состоянием GPIO можно отправкой GET-запроса:

http://IP_ADRES/gpio?st=DATA&pin=N

где:

- **N** — номер вывода GPIO;
- **DATA** — данные для установки: 0 или 1.

Если дописать к команде фрагмент **&flash=1**, то настройка запишется в энергонезависимую память модуля.

Просмотреть состояние выводов GPIO можно по адресу: **http://IP_ADRESS/gpioprint** (рис. 14.44).

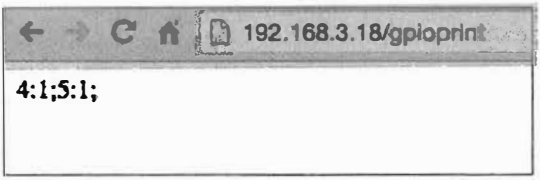


Рис. 14.44. Просмотр состояния выводов GPIO через GET-запрос

Выбрав для вывода GPIO режим **INPUT** (см. рис. 14.43), состояние вывода можно получить GET-запросом: **http://IP_ADRESS/gpioprintinput**.

14.8. Работа с прерываниями модуля ESP8266

При создании прошивки модуля ESP8266 можно выбрать и работу с прерываниями. Модуль умеет считать импульсы (например, для водо- и электросчетчиков) и отправлять показания на удаленные серверы — для этого надо активировать соответствующий пункт в меню его веб-интерфейса (рис. 14.45). Так, при включении режима **Active send** модуль будет немедленно отсылать событие на сервер при изменении состояния на входе прерывания на выбранном GPIO (функция эта актуальна и для датчиков движения и датчиков открытия двери). При включении режима **Key** модуль может управлять другим GPIO (выключение/включение), указанным в поле **GPIO For Key**.

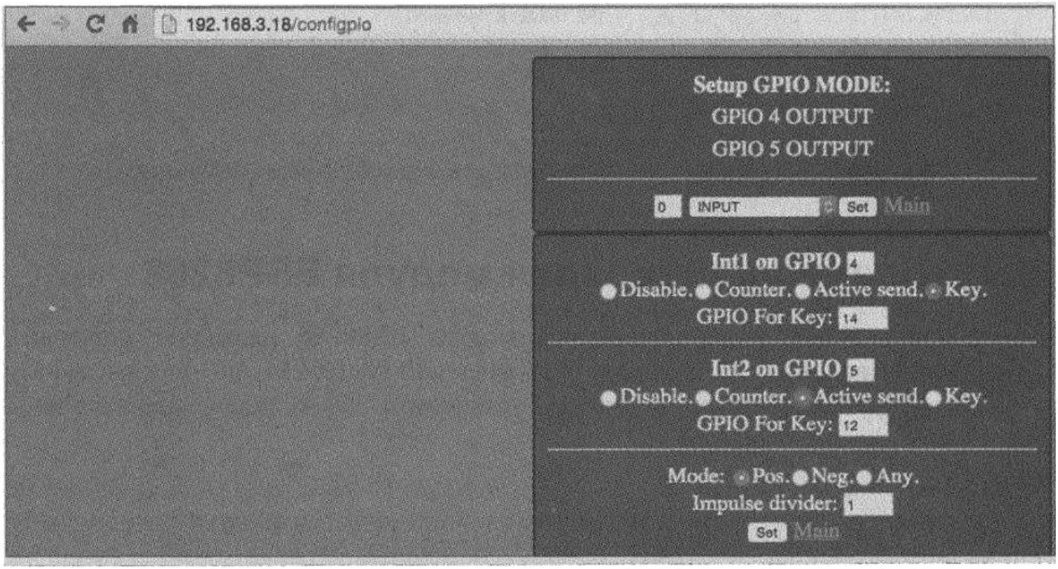


Рис. 14.45. Установка настроек прерываний в веб-интерфейсе модуля ESP8266

14.9. Управление каналами ШИМ модуля ESP8266

Модуль ESP8266 имеет три канала ШИМ (PWM). Для работы с ШИМ необходимо указать количество каналов и задать выводы GPIO, на которых необходимо запустить ШИМ (рис. 14.46). Управление ШИМ осуществляется GET-запросом:

http://IP_ADRESS/pwm?ch=Y&set=XXX

где:

- **Y** — номер канала (0, 1, 2);
- **XXX** — уровень ШИМ 0–255.

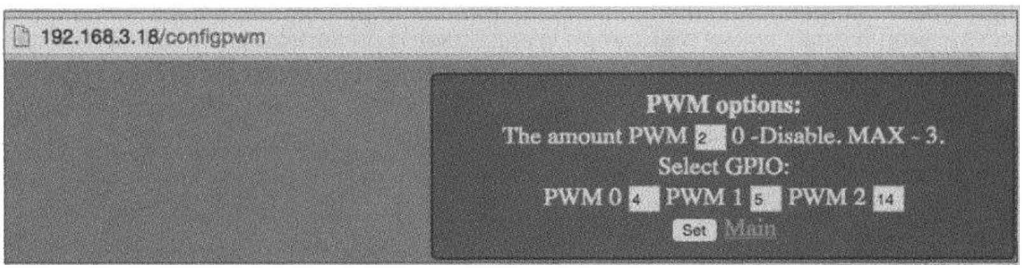


Рис. 14.46. Установка настроек ШИМ (PWM) в веб-интерфейсе модуля ESP8266

Посмотреть установленные уровни ШИМ можно по GET-запросу **http://IP_ADRESS/pwmprint** (рис. 14.47). Для сохранения состояния ШИМ в энергонезависимую память необходимо дописать к этому запросу фрагмент **&flash=1**.

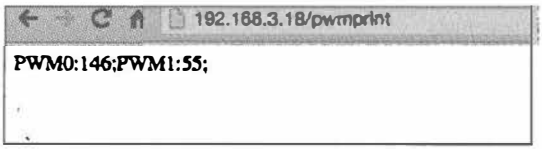


Рис. 14.47. Просмотр значений ШИМ (PWM) на выводах GPIO через GET-запрос

14.10. Планировщик задач модуля ESP8266

Планировщик **Sheduler** веб-интерфейса модуля ESP8266 позволяет управлять состоянием выводов GPIO, а также и уровнем ШИМ (PWM) по определенному времени. Прошивка модуля позволяет установить до 8 задач планировщика (рис. 14.48).

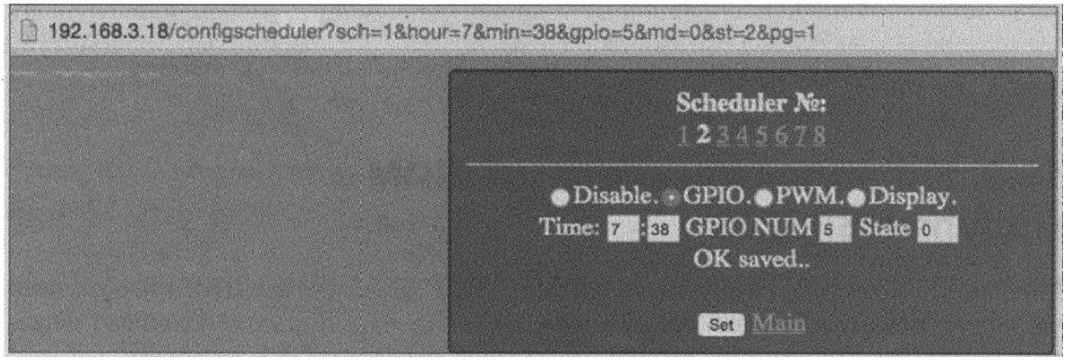


Рис. 14.48. Настройка задач планировщика **Sheduler**

Заключение

Интернет вещей (Internet of Things, IoT) — это новое направление, которое сейчас стремительно развивается. Связанные с Интернетом вещи призваны сделать нашу жизнь еще более функциональной и удобной. Интернет вещей трансформирует привычные для нас объекты в совершенно новые инструменты, позволяющие собирать, агрегировать и анализировать значительные объемы информации об окружающей нас действительности. Устройства с постоянным подключением к Интернету могут следить практически за всеми измеримыми параметрами. Таким образом, Интернет вещей предоставляет нам возможности наблюдения за любыми показателями окружающей среды в реальном времени. И, что самое важное, — возможность реализации функций анализа получаемых показателей и управления устройствами, их изменяющими.

По оценкам аналитиков Интернет вещей в ближайшем будущем охватит огромное число подключений: 1,9 миллиардов устройств к концу этого года и 9 миллиардов — к 2018-му. Ценность Интернета вещей будет расти в геометрической прогрессии.

В этой книге мы рассмотрели вопросы реализации идей Интернета вещей на самых известных и популярных платформах — Arduino, Raspberry Pi, а также на ставшем весьма популярным в последнее время Wi-Fi модуле ESP8266.

А практическое воплощение полученных при прочтении книги знаний поможет вам поучаствовать в построении мира Интернета вещей не только в качестве потребителя, но и разработчика.

Желаю вам творческих успехов!

ПРИЛОЖЕНИЕ

Описание электронного архива

Электронный архив с материалами, сопровождающими книгу, можно скачать с FTP-сервера издательства по ссылке **ftp://ftp.bhv.ru/9785977536462.zip**, а также со страницы книги на сайте **www.bhv.ru**.

В архиве находятся следующие папки:

- \arduino_scetches — исходники примеров и проектов *глав 4–10* для Arduino IDE;
- \python — исходники примеров для *глав 11* и *12*;
- \arduino_libraries — библиотеки Arduino, используемые в примерах и проектах книги и не включенные в среду разработки Arduino IDE.

Предметный указатель

M

MAC-адрес 30, 34, 35, 37, 45
Master-устройство 90

P

PIR-sensor 145

R

Raspberry Pi 187

S

Slave-устройство 90
Start-условие 91
Stop-условие 91

W

Wi-Fi модуль ESP8266 283

А

Автоматическая центровка и масштабирование схемы 21
Активация версии прошивки Pro модуля ESP8266 299
Аналоговые датчики 25
◊ оптические 49
Аналоговый датчик
◊ освещенности (фоторезистор) 39
◊ температуры LM335 27
◊ температуры LM35 261
Аналого-цифровые преобразователи (АЦП) 26
Аппаратные порты ввода/вывода GPIO 187

Б

Библиотека
◊ TinyWebServer 77
◊ Wire 93

В

Видеоускоритель Broadcom VideoCore IV 189
Виджет температуры в сервисе Wyliodrin 261
Виджеты для проекта в Blynk 171
Включение транзистора в схему управления реле 65
Вращающий момент сервопривода 73

Г

График показаний фоторезистора в сервисе Wyliodrin 261

Д

Датчик 25
◊ влажности DHT11 141
◊ влажности DHT22 141
◊ влажности и температуры SHT21 в сервисе Wyliodrin 266
◊ влажности и температуры SHT21 на шине I^2C 107
◊ движения HC-SR501 145
◊ звука FC-04 148
◊ освещенности BH1750 181, 215, 302
▫ в сервисе Wyliodrin 266
▫ на шине I2C 96
◊ света Grove light 271
◊ температуры DS18B20 209, 237, 300
◊ температуры LM335 163
◊ температуры LM35 в сервисе Wiliodrin 261
◊ температуры и влажности DHT11 (DHT22) 302
Диммер 68
Дисплей WH1602 310
Добавление
◊ в библиотеку Fritzing новых компонентов 22
◊ устройства в сервис Wyliodrin 246

З

Загрузка скетча 14
Загрузчик (bootloader) Arduino 14
Запись образа Wyliodrin на SD-карту 247

И

Изменение пароля WebIOPi 221
Интернет вещей 29
Интерфейс GPIO (General Purpose Input/Output) 203
Инфракрасные датчики расстояния Sharp 49
◊ GP2Y0A02YK 25
Исполнительные устройства 63
Использование
◊ в сервисе Wyliodrin платы Arduino, подключенной к Raspberry Pi 259
◊ внешних носителей 121
◊ технологии 1-Wire 128

К

Камера Raspberry Pi Camera Board 231
Карта памяти
◊ формата MicroSD 189
◊ формата SD 189
Концепция Internet of Things 29

М

Меню конфигурации Raspberry Pi 194
Методы библиотеки Wire 94
Микросхема часов реального времени Dallas DS1307 116
Монитор последовательного порта 14

Н

Набор стандартных библиотек 14
Назначение выводов GPIO 204
Накопитель на SD-карте 121
Настройка
◊ доступа к Raspberry Pi по Wi-Fi 196
◊ сервера WebIOPi 222
◊ сетевых параметров Raspberry Pi 196
Нейтральное положение сервопривода 73

О

Облачный сервис Xively 99
Обмен
◊ информацией по шине 1-Wire 130
◊ сообщениями между платами Raspberry Pi через сервис Wyliodrin 274
Обновление прошивок модуля ESP8266 288
Онлайн-сервис Wyliodrin 245
Оптрон 67
Отправка данных
◊ в несколько каналов 111
◊ в сервис ThingSpeak 42
◊ в сервис Wyliodrin с мобильного устройства 277
◊ из Arduino в Xively 102
◊ на сайт «Народный мониторинг» с использованием платы GPRS/GS Shield 162
◊ с Raspberry Pi в сервис «Народный мониторинг» 212
◊ с модуля ESP8266 на сайт «Народный мониторинг» 306
◊ с модуля ESP8266 в сервис ThingSpeak 307
◊ с платы Arduino в облачные сервисы 33
Отправка и получение SMS-сообщений с помощью платы GSM/GPRS Shield 159

П

Первоначальная настройка модуля ESP8266 297
Пироэлектрический инфракрасный сенсор 145
Питание от паразитного источника 125
Планировщик задач модуля ESP8266 314
Плата
◊ Arduino GPRS/GSM Shield 157
◊ Ethernet Shield 29, 63
◊ SIM900 Quad-Band GPRS Shield 157
Платы расширения GrovePi и GrovePi+ 270
Подключение
◊ 3G-модема к Raspberry Pi 199, 200
◊ Raspberry Pi к Wyliodrin 251
◊ датчика BH1750 к плате Arduino 98
◊ датчика DS18B20
▫ к контактам GPIO Raspberry Pi 209
▫ к модулю ESP8266 300
◊ датчика освещенности BH1750 к модулю ESP8266 302
◊ датчика температуры и влажности DHT11 (DHT22) к модулю ESP8266 302
◊ датчиков DHT к Arduino 143
◊ датчиков расстояния Sharp к Arduino 51
◊ диммера к Arduino 69
◊ к Raspberry Pi датчика BH1750 215
◊ модуля SD Card 121
◊ твердотельного реле к Arduino 67
◊ электромагнитного реле к Arduino 65
Получение данных
◊ из сервиса Xively 105
◊ из нескольких каналов 113
Приложение ThingTweet сервиса ThingSpeak 55
Пример отправки данных в Twitter 53

Принцип действия датчиков расстояния Sharp 49
Проект
◊ Blynk 169
◊ Home's Smart («Умный дом») 294
Протокол 1-Wire 125
Процессор
◊ BCM2836 190
◊ Broadcom BCM2835 188
Прошивка NodeMCU модуля ESP8266 289

Р

Работа
◊ с GPIO на языке Python 206
◊ с прерываниями модуля ESP8266 312
◊ с ШИМ модуля ESP8266 313
Регистрация в сервисе Wyliodrin 245
Режим AT-команд модуля ESP8266 284

С

Сайт «Народный мониторинг» 37
Сборка прошивки модуля ESP8266 294
Сглаживание значений, получаемых с датчика расстояния 52
Сенсор 25
Сервис
◊ ThingSpeak 41
◊ Twitter 53
◊ Weaved 238
Сервопривод 72
Симистор 67, 68
Система Grove 270
Скетч 13
Скорость вращения вала сервопривода 75
Слот для флеш-карты MicroSD 78
Создание
◊ нового проекта в Blynk 170
◊ приложения в среде программирования Wyliodrin 253
◊ принципиальной схемы 21
Среда программирования
◊ Arduino IDE 9
◊ Blockly 245
Среда разработки Fritzing 17
Стабилитрон 27
Стандартные сервоприводы 73
Стартовая страница WebIOPi 221
Статистическое управление процессами 187
Страница управления выводами GPIO на WebIOPi 221
Схема
◊ подключения к GPIO датчика температуры DS18B20 237
◊ соединения элементов 17

Т

Твердотельное реле 67
Техническая документация (Datasheet) 27
Технология
◊ «Умный дом» 128
◊ AJAX 80
Тиристоры 67
Топология общей шины 126
Точка калибровки датчика 28

У

Угол поворота сервопривода 73
Управление
◊ выводами GPIO модуля ESP8266 311
◊ светодиодом с веб-страницы Wyliodrin 257
◊ сервоприводом 73
◊ сервоприводом из проекта Blynk 178
Установка
◊ Arduino IDE 10
◊ WebIOPi на ОС Raspbian 219
◊ дистрибутива ОС на Raspberry Pi 191
◊ драйверов Arduino IDE 11
◊ среды разработки Fritzing 17
Устройство электромагнитного реле 64
Учетная запись Wyliodrin 245

Ф

Файл настроек Wyliodrin 250
Фоторезистор 39, 96
Фреймворк WebIOPi 219

Ч

Часы реального времени (RTC) 190

Ш

Шина I^2C 89
Широтно-импульсная модуляция (ШИМ) 73

Э

Электромагнитное реле 64
◊ SRD-05VDC фирмы SONGLE 64

Я

Язык программирования Lua 289