AI7688H User Manual

V.1 160407

An IOT Solution Company

Copyright@2016 AcSiP

Revision History

Revision	Date	Author	Description
V.1	160407	Kevin	New Create

Copyright@2016 AcSiP

Contents:

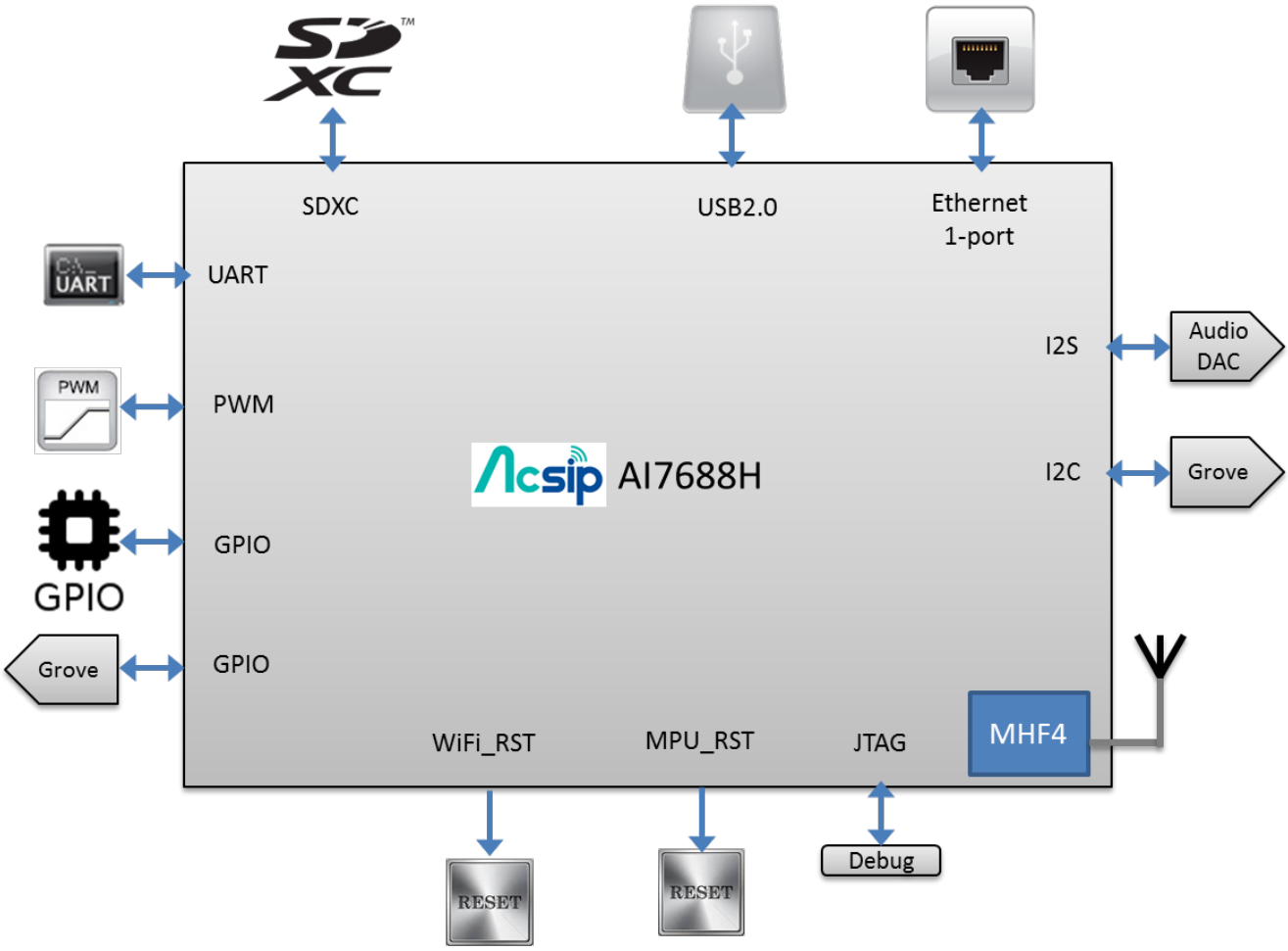
1. Introduction.....4
2. Start AI7688H development 5
3. Firmware and Bootloader.....9
4. File Storage.....19
5. Wi-Fi LED State.....22
6. Network.....24
7. AWS IoT.....40
8. Peripheral.....43
9. C/C++ Programming.....51
10. Using USB Webcam.....58
11. Audio Playback and Recording.....59
12. Federal Communication Commission Interference Statement.....61

Copyright@2016 AcSiP

1. Introduction

AI7688H integrates a 1T1R 802.11n Wi-Fi radio, a 575/580 MHz MIPSR 24KEc™ CPU, 1-port fast Ethemet PHY,USB2.0 host,PCIe,SD-XC,I2S/PCM and multiple slow IOs.

AI7688H provides two operation modes – IoT gateway mode and IoT device mode.In IoT gateway Mode, the PCIe Express interface can connect to 802.11ac chipset for 11ac dual-band concurrent Gateway. The high performance USB 2.0 allows AI7688H to add 3G/LTE modem support or add a H.264 ISP for wireless IP camera. For the IoT device mode, AI7688Hsupports eMMC , SD-XC and USB2.0. AI7688H can support the WiFi high quality audio via 192kbps/24bits I2S interface and VoIP application through PCM. In IoT device mode, it further supports PWM,SPI slave,3rd UART and more GPIOs. For IoT gateway, it can connect to touch panel and BLE,Zigbee/Z-Wave and sub-1G RF for smart home control.



Copyright@2016 AcSiP

2. Start AI7688H development

2.1 Get S/W

 # Download an SSH client (Windows only)

PuTTY provides Secure Socket Shell (SSH) access to the development board operating system.

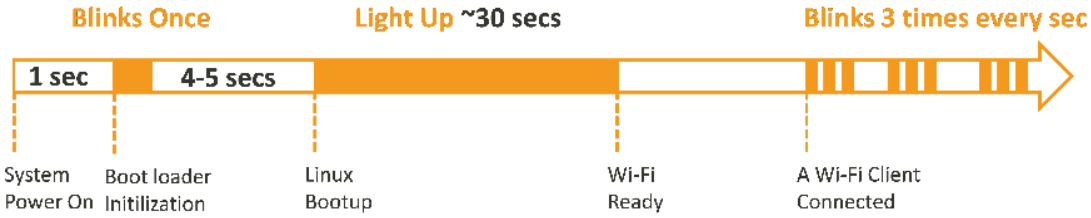
 # Install Bonjour Print Service (For Windows 7 only)

The AI7688H development board uses mylinkit.local as its local domain. In Windows7, you'll need to installBonjour print service because mDNS is not support.

This helps your computer discover the LinkIt Smart AI7688H's IP address with the local domain name. For Windows 8 and later, Mac OS X and Linux, mDNS is supported and you can use mylinkit.local.

2.2 Power up AI7688H board

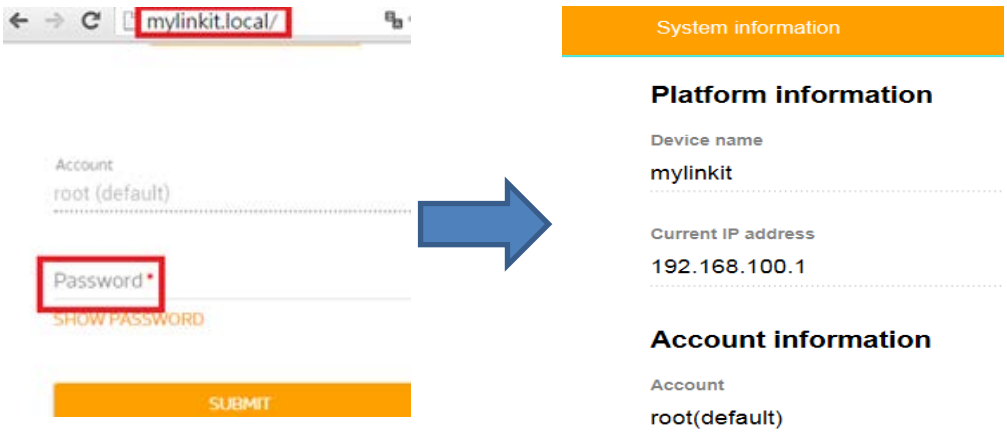
After bootup and Wi-Fi initialization completes, which takes about 30 seconds, the Wi-Fi LED goes off.
This means the system is ready to accept Wi-Fi connection – now we can connect to it. Following Figure shows how the Wi-Fi LED status matches the system state.



Copyright@2016 AcSiP

2.3 Find the AI7688H AP and connect

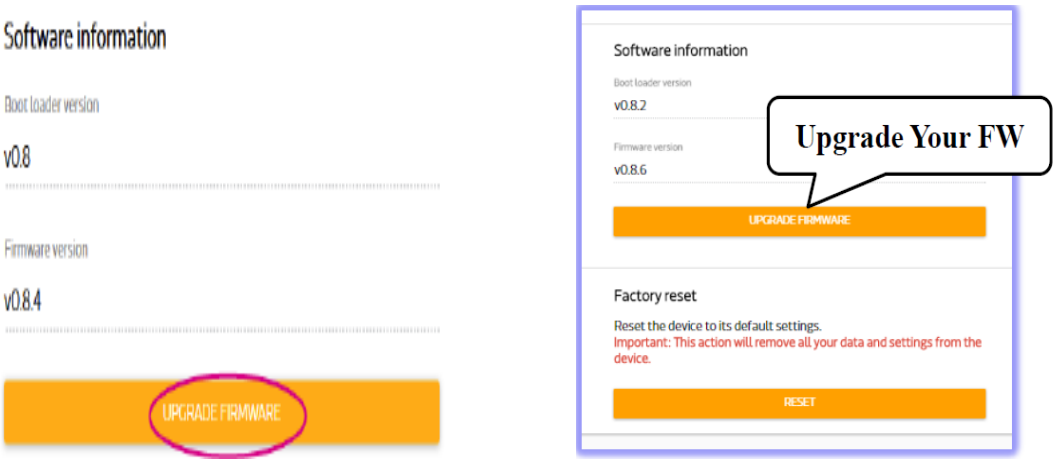
2.4 Open ***http://mylinkit.local***



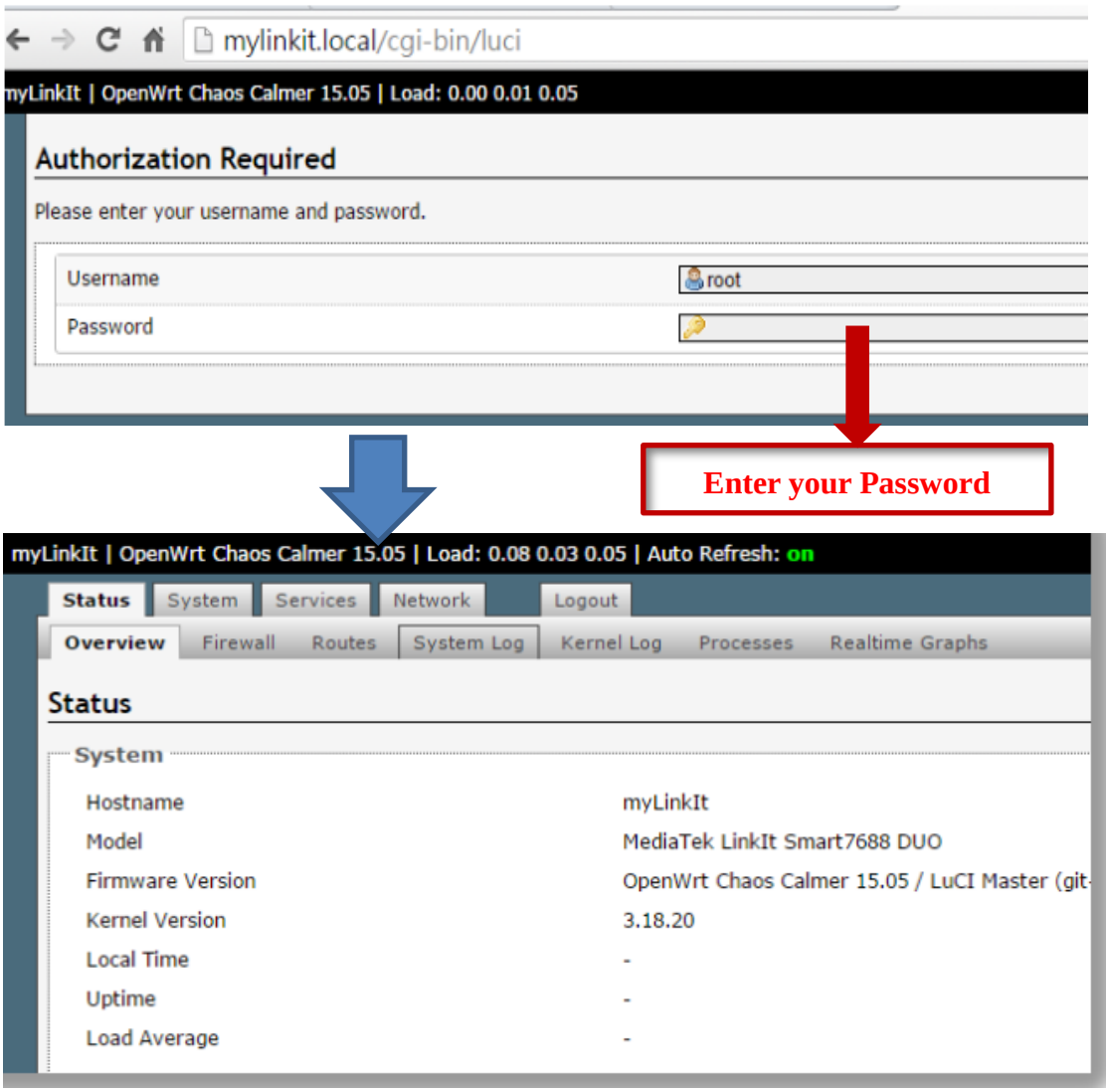
If the board already has a password and you don't have or lost it,
You have to press "Wi-Fi Reset button" for at least 20 seconds then Release.
Then you can configure your password again.

Copyright@2016 AcSiP

2.5 Get system Info and Upgrade F/W



2.6 Go to ***OpwnWrt*** for advanced configuration



Copyright@2016 AcSiP

2.7 Access system console

For OS X and Linux:

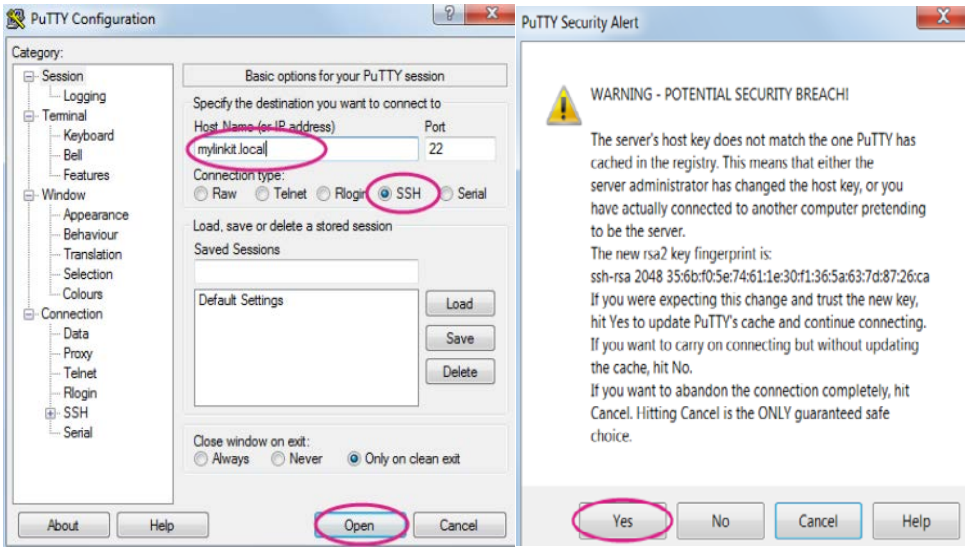
Open Terminal application

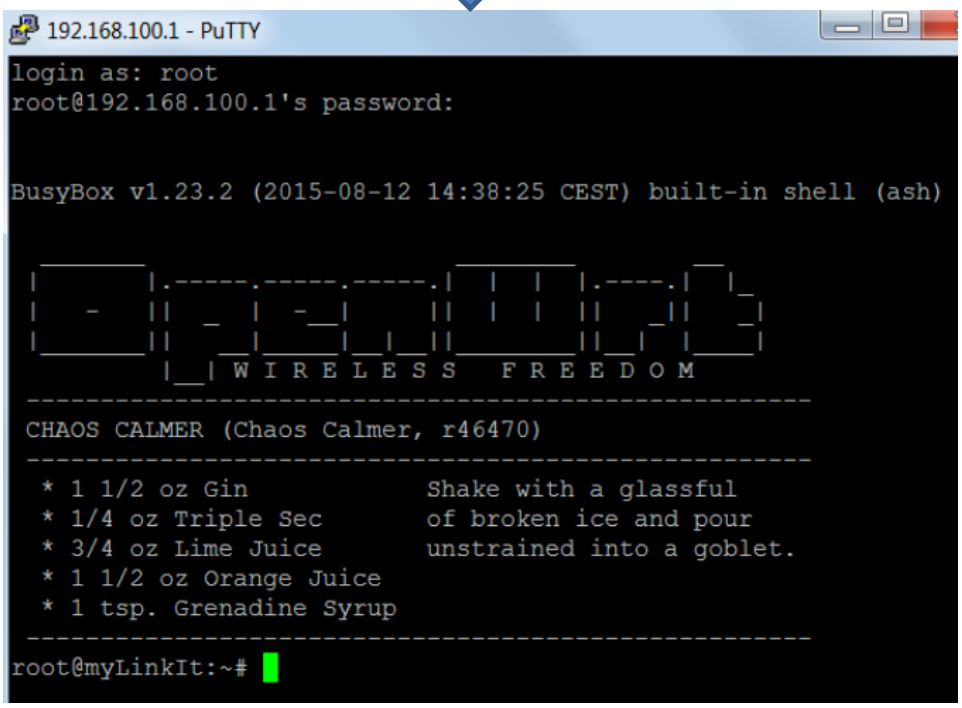
At the command prompt type

ssh root@mylinkit.local

Press return and enter the password you set previously in the Web UI

For Windows:





Copyright@2016 AcSiP

3 Firmware and Bootloader

3.1 Flash F/W from USB drive

Download lates Firmware

Unzip it and copy the unzipped lks7688.img file to the **root** directory of a **FAT32** USB Drive

Attach the USB drive to the HOST port of the board with an OTG cable.

Hold the WiFi button

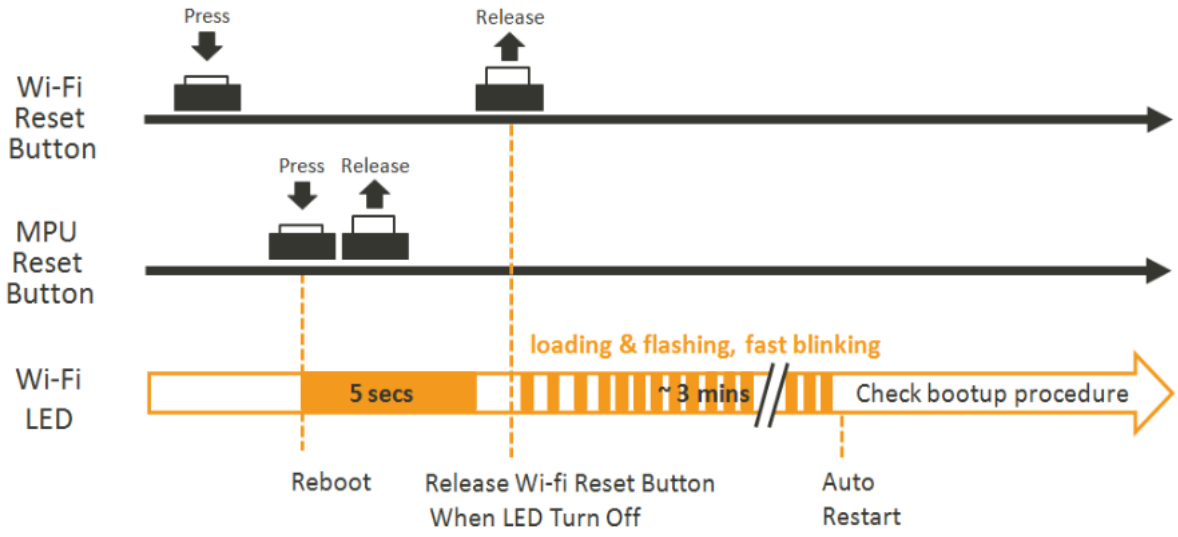
Press the MPU reset button once while holding the WiFi button

Keep holding WiFi button about 5 seconds. Release WiFi button until the WiFi orange LED becomes OFF.

Do not press the Wi-Fi button for longer than 20s or it will upgrade the bootloader.

Wait while the WiFi LED blinks fast. This takes about 3 minutes.

The device will automatically reboot after firmware update completed.



Copyright@2016 AcSiP

3.2 Update bootloader

USB drive must be in FAT file system or the file cannot be recognized by the AI7688H development platform.

Save the bootloader file (lks7688.ldr) in the root directory of a USB drive and name it lks7688.ldr.

Plug the USB drive to AI7688H.

Press the WiFi and MPU (Reset) button at the same time, then release the MPU Reset button but hold the WiFi button for at least *20* seconds.

After 20 seconds Wi-Fi LED will turn on. Release WiFi button.

The board will start to read the bootloader (WiFi LED blinks fast) and perform the bootloader upgrade process (Wi-Fi LED blinks slowly). It takes about 2 seconds to finish the bootloader upgrade process

3.3 Build F/W from Source

The following operations are performed under a Ubuntu LTS 14.04.3 environment. For a Windows or a Mac OS X host computer, you can install a VM for having the same environment:

- Install prerequisite packages for building the firmware:

```
$ sudo apt-get install git g++ libncurses5-dev subversion libssl-dev gawk
  libxml-parser-perl unzip
```

- Download OpenWrt CC source codes:

```
$ git clone git://git.openwrt.org/15.05/openwrt.git
```

- Prepare the default configuration file for feeds:

```
$ cd openwrt
$ cp feeds.conf.default feeds.conf
```


Copyright@2016 AcSiP

- Add the AI7688H feed:

```
$ echo src-git linkit https://github.com/MediaTek-Labs/linkit-smart-7688-feed.git >>
    feeds.conf
```

- Update the feed information of all available packages for building the firmware:

```
$ ./scripts/feeds update
```

- Install all packages:

```
$ ./scripts/feeds install -a
```

- Prepare the kernel configuration to inform OpenWrt that we want to build an firmware for AI7688H:

```
$ make menuconfig
```

  - Select the options as below:
    - Target System: Ralink RT288x/RT3xxx
    - Subtarget: AI7688H based boards
  - Save and exit (**use the deafult config file name without changing it**)
- Start the compilation process:

```
$ make V=99
```

- After the build process completes, the resulted firmware file will be under bin/ramips/openwrt-ramips-7688-LinkIt7688-squashfs-sysupgrade.bin. Depending on the H/W resources of the host environment, the build process may **take more than 2 hours**.
- You can use this file to do the firmware upgrade through the Web UI. Or rename it to lks7688.img for upgrading through a USB drive

Copyright@2016 AcSiP

3.4 Build Bootloader from Source

The following operations are performed under a Ubuntu LTS 14.04.3 environment. For a Windows or a Mac OS X host computer, you can install a VM for having the same environment:

- Refer to Build_F/W_Source for installing all prerequisite packages.
- Download the bootloader source codes:

```
$ git clone https://github.com/MediaTek-Labs/linkit-smart-uboot.git
```

- Change to the source folder:

```
$ cd linkit-smart-uboot
```

- Install the toolchain needed for building the bootloader:

```
$ sudo tar xjf buildroot-gcc342.tar.bz2 -C /opt/
```

- Since the toolchain is provided in 32-bit executables, you need to install additional packages for execution on a 64-bit machine:

```
$ sudo dpkg --add-architecture i386
$ sudo apt-get update
$ sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386
```

- Start the compilation process:

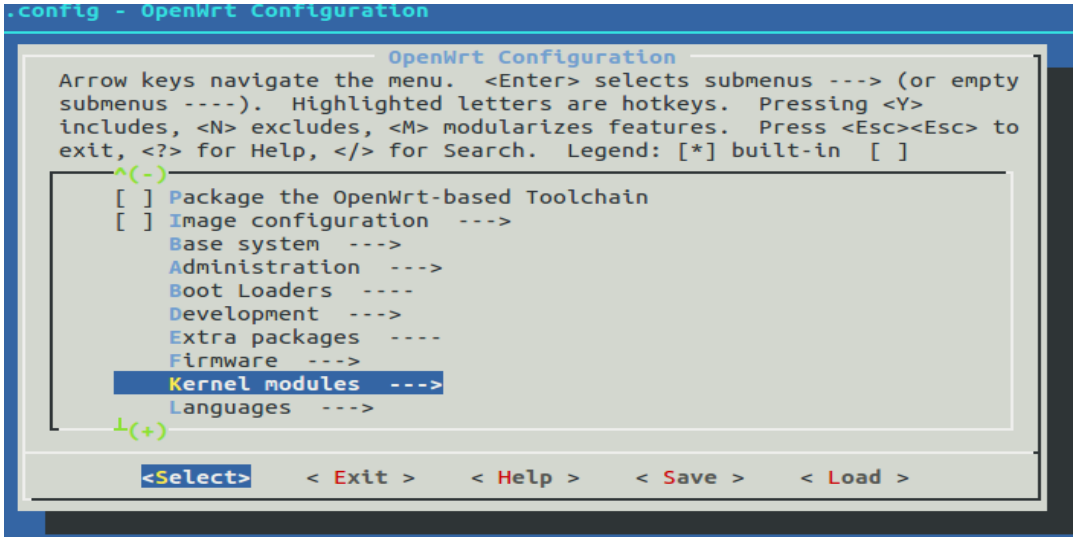
```
$ make
```

- The resulted bootloader file is uboot.bin.
- You can rename it to lks7688.ldr for upgrading the system bootloader through a USB drive.

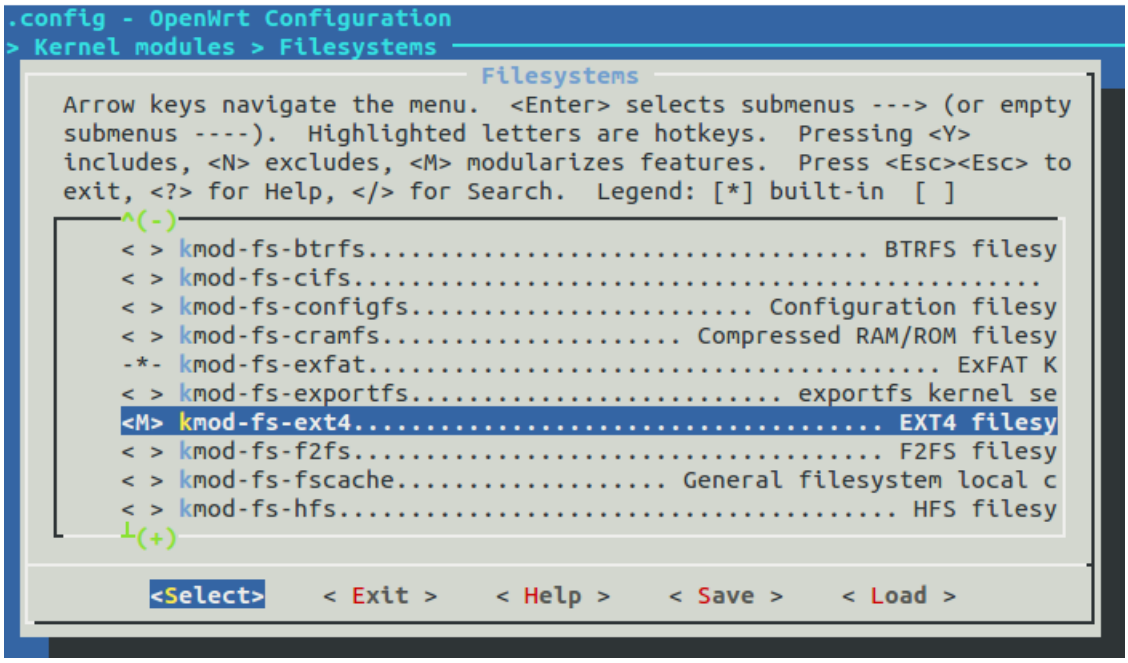
Copyright@2016 AcSiP

3.5 Rebuild Existing Kernel Packages

In the config menu, select the **Kernel modules**.



Go to **Filesystems** and select the **kmod-fs-ext4** as **M** (modularizes features). **Note**: other kernel packages which **kmod-fs-ext4** depends on will also be selected automatically.



Follow the original build flow to continue the setup and save the configuration for the firmware building process.

Copyright@2016 AcSiP

After the building process is complete, you can find the kernel packages

under bin/ramips/packages/base/ (if the package you built is not there, you can also use find command to locate where it is)

After the building process is complete, you can find the kernel packages
under bin/ramips/packages/base/ (if the package you built is not there, you can also use find command to locate where it is).

Copy **all** related kernel module packages to a USB drive.

Plug the USB drive to AI7688H and perform the *opkg install* command to install **all** necessary kernel packages from the USB drive.

```
root@mylinkit:/tmp/run/mountd/sda1# opkg install kmod*.ipk
Installing kmod-crypto-core (3.18.23-1) to root...
Installing kmod-crypto-hash (3.18.23-1) to root...
Installing kmod-fs-ext4 (3.18.23-1) to root...
Installing kmod-lib-crc16 (3.18.23-1) to root..
Package kmod-lib-crc16 (3.18.23-1) installed in root is up to date.
Configuring kmod-crypto-core.
Configuring kmod-crypto-hash.
Configuring kmod-lib-crc16.
Configuring kmod-fs-ext4.
```

Note: if you simply install the *kmod-fs-ext4*, an error will occur like

```
root@mylinkit:/tmp/run/mountd/sda1# opkg install
kmod-fs-ext4_3.18.23-1_ramips_24kec.ipk
Installing kmod-fs-ext4 (3.18.23-1) to root...
Collected errors:
* satisfy_dependencies_for: Cannot satisfy the following dependencies for kmod-fs-ext4:
*        kmod-lib-crc16 *        kmod-crypto-hash *
* opkg_install_cmd: Cannot install package kmod-fs-ext4.
```

From these information, we can know what additional kernel packages are also needed for *kmod-fs-ext4* under bin/ramips/packages/base/ in **Step 5**

Copyright@2016 AcSiP

3.6 Bootloader and Kernel Console

The Bootloader console and Linux kernel console are configured to serial port UART2(UART_TXD2 and UART_RXD2) at baudrate 57600

```
[04060C0F][04060C0C]
DDR Calibration DQS reg = 0000898A
U-Boot 1.1.3 (Sep 10 2015 - 05:56:31)
Board: Ralink APSoC DRAM:  128 MB
relocate_code Pointer at: 87f68000
flash manufacture id: c2, device id 20 19
find flash: MX25L25635E
*** Warning - bad CRC, using default environment
  ============================================
Ralink UBoot Version: 4.3.0.0
--------------------------------------------
ASIC 7628_MP (Port5<->None)
DRAM component: 1024 Mbits DDR, width 16
DRAM bus: 16 bit
Total memory: 128 MBytes
Flash component: SPI Flash
Date:Sep 10 2015  Time:05:56:31
============================================
icache: sets:512, ways:4, linesz:32 ,total:65536
dcache: sets:256, ways:4, linesz:32 ,total:32768

##### The CPU freq = 580 MHZ ####
estimate memory size =128 Mbytes
RESET MT7628 PHY!!!!!!
GPIOMODE --> 50054404
GPIOMODE2 --> 5540551
Please choose the operation:
1: Load system code to SDRAM via TFTP.
2: Load system code then write to Flash via TFTP.
3: Boot system code via Flash (default).
4: Entr boot command line interface.
7: Load Boot Loader code then write to Flash via Serial.
9: Load Boot Loader code then write to Flash via TFTP.
```


Copyright@2016 AcSiP

There is about 1 second for you to choose from the bootloader menu. If there is no user input, the bootloader continues into Linux kernel bootup:

3: System Boot system code via Flash.

Booting image at bc050000 ...

Image Name: MIPS OpenWrt Linux-3.18.21

Image Type: MIPS Linux Kernel Image (lzma compressed)

Data Size: 1118412 Bytes = 1.1 MB

Load Address: 80000000

Entry Point: 80000000

Verifying Checksum ... OK

Uncompressing Kernel Image ... OK

No initrd

Transferring control to Linux (at address 80000000) ...

Giving linux memsize in MB, 128

Starting kernel ...

[0.000000] Linux version 3.18.21 (root@builder1) (gcc version 4.8.3 (OpenWrt/Linaro GCC 4.8-2014.04 r47269)) #7 Sat Nov 7 14:50:53 CET 2015

[0.000000] Board has DDR2

[0.000000] Analog PMU set to hw control

[0.000000] Digital PMU set to hw control

[0.000000] SoC Type:

[0.000000] bootconsole [early0] enabled

[0.000000] CPU0 revision is: 00019655 (MIPS 24KEc)

[0.000000] MIPS:

[0.000000] Determined physical RAM map:

[0.000000] memory: 08000000 @ 00000000 (usable)

At this point you can press ENTER to gain access to the Linux kernel console

```
BusyBox v1.23.2 (2015-10-27 13:23:36 CET) built-in shell (ash)


  ______                    _______        __
 |      |.------.-----.------.|  |  |  |.-----.|  |_
 |  -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |______||   __|_____|__|__||_______||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 CHAOS CALMER (15.05+linkit, r47390)
 -----------------------------------------------------
  * 1 1/2 oz Gin            Shake with a glassful
  * 1/4 oz Triple Sec       of broken ice and pour
  * 3/4 oz Lime Juice       unstrained into a goblet.
  * 1 1/2 oz Orange Juice
  * 1 tsp. Grenadine Syrup
 -----------------------------------------------------
root@mylinkit:/#
```


Copyright@2016 AcSiP

3.7 Change bootloader Console

If you want to use another UART port as the bootloader console, you need to modify the bootloader.

```
# Get bootloader source code
# modify the file board/rt2880/serial.h
/*
 * UART registers
*/
#if defined (MT7621_FPGA_BOARD) || defined (MT7621_ASIC_BOARD) || defined
(MT7628_FPGA_BOARD) || defined (MT7628_ASIC_BOARD)
#define RT2880_UART1        0x0C00    /* UART Lite */
#define RT2880_UART2        0x0D00    /* UART Lite */
#define RT2880_UART3        0x0E00    /* UART Lite */
//#define CFG_RT2880_CONSOLE        RT2880_UART1
#define CFG_RT2880_CONSOLE        RT2880_UART3
#else
#define RT2880_UART1        0x0500
#define RT2880_UART2        0x0C00    /* UART Lite */
#define CFG_RT2880_CONSOLE        RT2880_UART2
#endif
```

and modify the CFG_RT2880_CONSOLE configuration.

Note that in bootloader code, RT2880_UART1 refers to UART0(UART_TXD0/UART_TXR0) and so on. So modify the following modification:

Change this:

```
//#define CFG_RT2880_CONSOLE        RT2880_UART1
#define CFG_RT2880_CONSOLE        RT2880_UART3
```

to this:

```
#define CFG_RT2880_CONSOLE        RT2880_UART1
//#define CFG_RT2880_CONSOLE        RT2880_UART3
```

This changes the bootloader console from UART2 to UART0.

if you also want to change the baudrate, please modify the file include/configs/rt2880.h

Copyright@2016 AcSiP

```
#define SERIAL_CLOCK_DIVISOR 16
#define CONFIG_BOOTDELAY        1       /* autoboot after 5 seconds     */
#define CONFIG_BAUDRATE         57600
#define CONFIG_SERVERIP 10.10.10.3
#define CONFIG_IPADDR 10.10.10.123
```

Change this:

```
#define CONFIG_BAUDRATE         57600
```

to this (change the baudrate from 57600 to 115200 as an example):

```
#define CONFIG_BAUDRATE         115200
```

This will then change the bootloader console baudrate from 57600 to 115200

 # Now build the bootloader and upload it to the board

3.8 Change Kernel Console

If you want to use another UART port as the kernel console, you can follow the steps below to adjust the configuration.

modify the file target/linux/ramips/dts/LINKIT7688.dts and chang

```
chosen {
bootargs = "console=ttyS2,57600";
  };
```

to this (e.g. change to UART0 and 115200 baudrate):

```
chosen {
bootargs = "console=ttyS0,115200";
  };
```

This will then change the kernel console from UART2 to UART0 and its baudrate from 57600 to 115200.

Now build the firmware and upload it to the board through the Web UI or the USB drive.

Copyright@2016 AcSiP

4 File Storage

4.1 USB Drive and SD Card

When a USB drive or SD card is inserted into AI7688H, they can be accessed under /Media/SD* or /Media/USB* (The device name displayed varies depending on the number of drives you use and the number of partitions available on the USB drive or SD card).

You can use the following commands to check the contents of the USB drive and SD card:

- In this example, a USB drive named USB-A1 is used:
- # ls /Media/USB-A1
- In this example, a SD card named SD-P1 is used:
- # ls /Media/SD-P1

4.2 How to mount the root FS on a SD card

The on-board flash is a raw flash. It is with limited write cycles (about 100,000 times) and without wear leveling mechanism and atomic write operation. So it is not recommended to write user data to the on-board flash frequently. Besides, its storage is also limited to 32MB, the spare space might not be sufficient for storing lots of user data and software packages. As a result, it is recommended to mount the root FS on a SD card for getting more and reliable storage.

Steps:

Below are the steps for making this work. For details, please refer to the OpenWrt WiKi for extroot. **Note: the procedure below assumes there is only one partition on the SD card.** If there are multiple partitions on the SD card, you can use fdisk command to manage partition settings.

1. Insert a microSD card into the AI7688H device. **Warning: all the data on the SD card will be erased in the following steps.**
2. Make sure the device is under Station mode for accessing internet.
3. Open the system console of the AI7688H.
4. Type the following commands for installing related packages:

```
# opkg update
# opkg install block-mount kmod-fs-ext4 kmod-usb-storage-extras e2fsprogs fdisk
```


Copyright@2016 AcSiP

5. Format the SD card. **ext4** file system will be used in this example:

```
# mkfs.ext4 /dev/mmcblk0p1
```

During the formatting process, it'll prompt a confirmation: Proceed anyway? (y,n). Press "y" to continue.+

6. Duplicate current root FS and move it to the SD card:

```
# mount /dev/mmcblk0p1 /mnt
# tar -C /overlay -cvf - . | tar -C /mnt -xf -
# umount /mnt
```

7. Create a *fstab* template:

```
# block detect > /etc/config/fstab
```

8. Open the *fstab* configuration (use vi as the editor in this example):

```
# vi /etc/config/fstab
```

9. In the **'mount'** config section, change+
   - the target option to '**/overlay**'
   - the enabled option to '**1**' +

   Then the config file will look like:

```
config 'mount'
  option  target  '/overlay'
  ...
  option  enabled '1'
```

   (In vi, by pressing the "**i**" key for entering the editing mode)+

10. Save and exit the configuration file (press "**Esc**" key and type "**:x**". Then press "**Enter**" in vi for saving the file).

Copyright@2016 AcSiP

11. Reboot the platform. Done.

Check if the setup takes effect:

We can use the "df -h" command to check if the root FS is mounted successfully.

```
root@mylinkit:/# df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                   11.0M      1.2M      9.8M  11% /
/dev/root                19.8M     19.8M         0 100% /rom
```

When the root FS is on the on-board flash, we can see rootfs only has 11MB storage space left:

```
root@mylinkit:~# df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                    7.2G     24.6M      6.7G   0% /
/dev/root                19.8M     19.8M         0 100% /rom
```

After mounting the root FS on the SD card, we can see the spare space of rootfs increases:

```
root@mylinkit:~# df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                    7.2G     24.6M      6.7G   0% /
/dev/root                19.8M     19.8M         0 100% /rom
```

In this example, a 8GB SD card is used and the rootfs now becomes 7.2GB in size.

Copyright@2016 AcSiP

5 Wi-Fi LED State

After powerring up or resetting the device, the Wi-Fi LED will first blink once to indicate the bootloader is in initialization process. Following the bootloader initialization, which takes about 4 seconds, the Linux kernel is loaded and starts the bootup process. When the Linux is booting up, the Wi-Fi LED will light on steadily.It takes about 30 seconds for the Linux kernel to initialize subsystems.

Once the Linux completes the boot-up process and the Wi-Fi subsystem is ready, the LED switches off.The LED will have different behaviors according to different Wi-Fi configuration. AI7688H can operate under 2 different modes, the **AP (Access Point)** mode and the **Station** mode. The AP mode forms a Wi-Fi network that allows your computer to join, and the station mode allows AI7688H to join other Wi-Fi network formed by other Wi-Fi access points.+

In AP mode

There are 2 Wi-Fi LED states in AP mode:

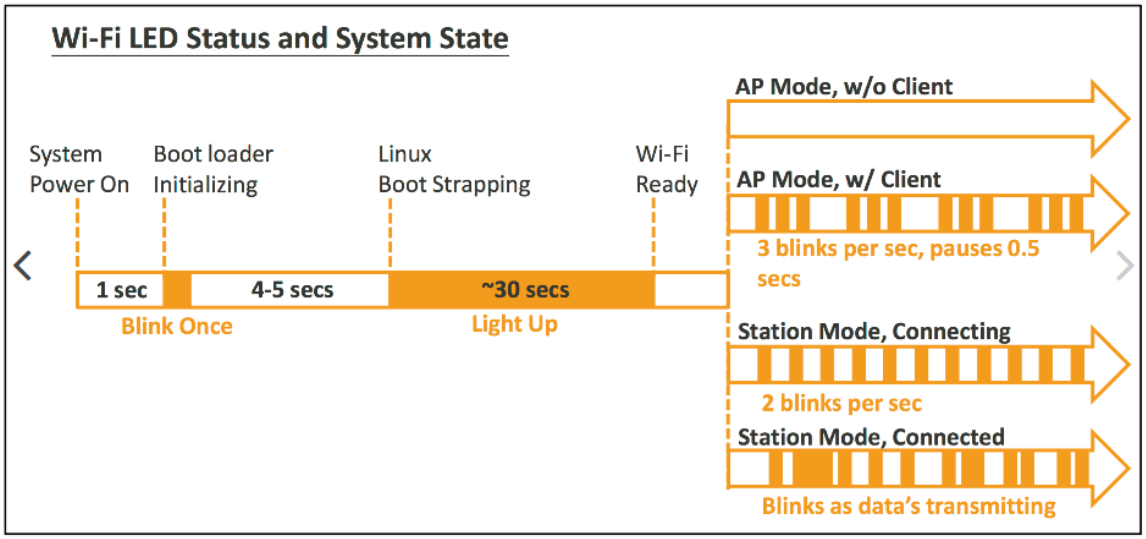
1. LED is off. It means no client device is connected to AI7688H.
2. LED blinks 3 times per second and pauses 0.5 seconds, and repeats the pattern. This indicates there is at least 1 client device connected to AI7688H.

In Station mode

There are 3 Wi-Fi LED states in Station mode:

1. LED is off. AI7688H failed to connect to a wireless router and is timed-out.
2. LED blinks twice per second continuously. It indicates AI7688H is connecting to a wireless router.
3. LED blinks according to data transmission. AI7688H has connected to a wireless router and the Wi-Fi LED will blink as data is transmitting.

Copyright@2016 AcSiP



The Wi-Fi LED state is an important indicator for troubleshooting network configuration issues - here's a list of common errors and how to fix them.

LED state	Problem	Fixes
After switching to Station mode, the Wi-Fi LED keeps blinking twice per second, then switches off	It has failed to join the Wi-Fi network you assigned	Press the Wi-Fi button for at least 5 seconds and release to restore the Wi-Fi mode to AP. Re-connect and re-configure the Wi-Fi settings accordingly.
After resetting device, the Wi-Fi LED doesn't blink at all	The bootloader is corrupted or the on-board flash is damaged	The device is bricked. You need to re-program the flash using an external hardware flash programmer.
Wi-Fi LED lights up (doesn't switch off)	Linux has failed to initialize properly	This is usually caused by a damaged system image or corrupted Wi-Fi driver. You can resolve this by upgrading the firmware usinga USB drive. Please
Wi-Fi LED blinks once after reset, but does not light up	It failed to load Linux kernel	This is usually caused by a damaged system image. You can resolve this by upgrading the firmware usinga USB drive.

Copyright@2016 AcSiP

6 Network

6.1 Reset Wi-Fi Configuration

You can reset Wi-Fi configuration back to AP mode by press and hold WiFi button for at least 5 seconds. Note that you must release the button before 20 seconds, otherwise Factory Reset will be performed instead This can be useful when you set an incorrect password to the Station mode, or when you need to re-configure Wi-Fi settings.

Note: if AI7688H is already in AP mode, this operation takes no effect

6.2 Switch to Station mode

Step1: Type UCI commands to assign SSID, key, and encryption information for running Station mode

Assume the wireless router to be connected is with the following properties:

- SSID: **SampleAP**
- Password: **12345678**
- Encryption: **WPA2 Personal**

In the system console of AI7688H, type the following commands:

```
# uci set wireless.sta.ssid=SampleAP
# uci set wireless.sta.key=12345678
# uci set wireless.sta.encryption=psk2
# uci set wireless.sta.disabled=0
# uci commit
```

Step 2: Restart the Wi-Fi driver for activating the configuration

Type the command in the system console:

```
# wifi
```


Copyright@2016 AcSiP

Notice: if you use SSH to connect to the system console of AI7688H, you'll lose the connection once the wifi command is set. Because this command will restart the Wi-Fi driver. After the Wi-Fi restarts, connect your PC to the same wireless router as AI7688H did. Then you can connect to AI7688H with mylinkit.local by SSH again.

You can also refer to the LED behavior to check the states of the connection in Station mode.

Step 3: Check for Internet connection

Now we can check if you've established Internet connection by typing the following command in the system console:

ping –c 5 www.mediatek.com

6.3 Switch to AP mode

Step1: type UCI command to disable Station mode

type the following commands in the system console

uci set wireless.sta.disabled=1

uci commit

Step 2: restart the Wi-Fi driver

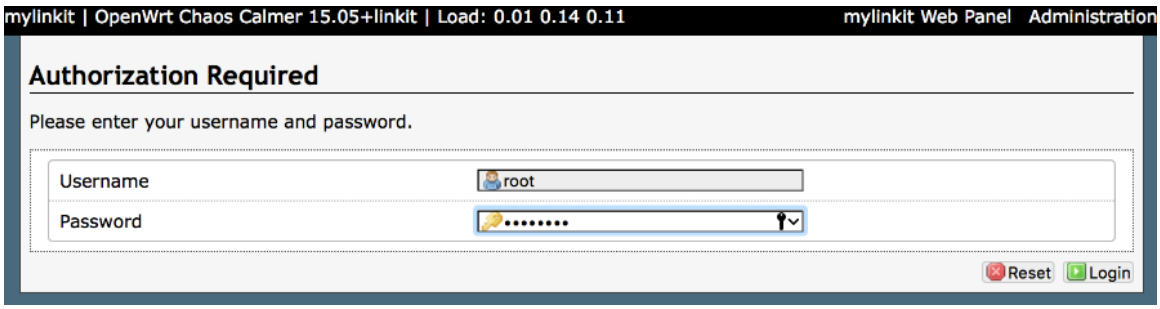
type the command in the console:

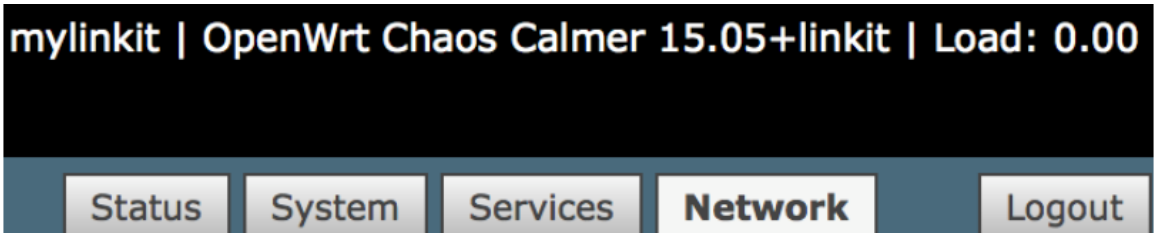
wifi

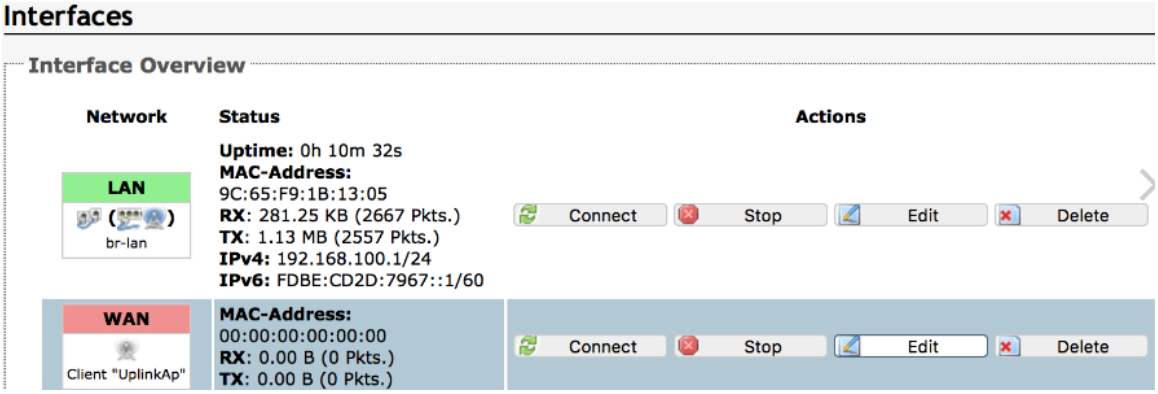
After the Wi-Fi driver restarts, the PC should be able to scan the AI7688H AP as shown in the video of the Reset Wi-Fi Configuration section. Also the Wi-Fi status can be checked through the LED behavior.

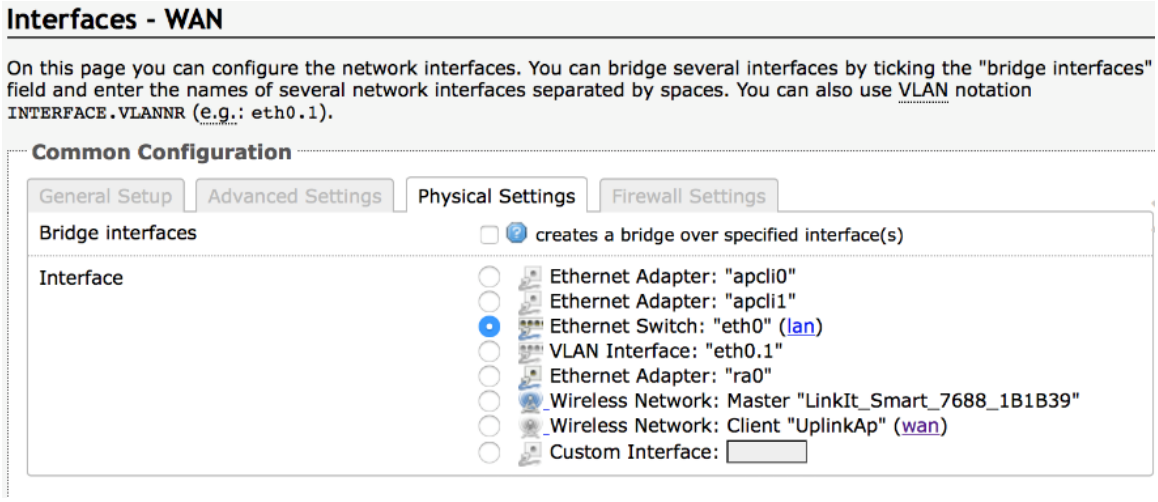
Copyright@2016 AcSiP

6.4 Setup Wireless Router









Copyright@2016 AcSiP

Interfaces

Interface Overview

Network	Status	Actions
LAN (br-lan)	**Uptime:** 0h 22m 20s **MAC-Address:** 9C:65:F9:1B:13:05 **RX:** 632.88 KB (5903 Pkts.) **TX:** 1.57 MB (5329 Pkts.) **IPv4:** 192.168.100.1/24 **IPv6:** FDBE:CD2D:7967::1/60	Connect Stop Edit Delete
WAN (eth0)	**Uptime:** 0h 0m 0s **MAC-Address:** 00:00:00:00:00:00 **RX:** 0.00 B (0 Pkts.) **TX:** 50.95 KB (305 Pkts.)	Connect Stop Edit Delete

Add new interface...

Interfaces - LAN

On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of several network interfaces separated by spaces. You can also use VLAN notation `INTERFACE.VLANNR` (e.g.: `eth0.1`).

Common Configuration

General Setup | Advanced Settings | Physical Settings | Firewall Settings

Bridge interfaces	☐ creates a bridge over specified interface(s)
Interface	○ Ethernet Adapter: "apcli0" ○ Ethernet Adapter: "apcli1" ○ Ethernet Switch: "eth0" (lan, wan) ○ VLAN Interface: "eth0.1" ○ Ethernet Adapter: "ra0" ◉ Wireless Network: Master "LinkIt_Smart_7688_1B1B39" ○ Wireless Network: Client "UplinkAp" (wan) ○ Custom Interface:

DHCP Server

General Setup | Advanced Settings | IPv6 Settings

Ignore interface	☐ Disable DHCP for this interface.
Start	100 Lowest leased address as offset from the network address.
Limit	150 Maximum number of leased addresses.
Leasetime	12h Expiry time of leased addresses, minimum is 2 minutes (2m).

Interfaces

Interface Overview

Network	Status	Actions
LAN (ra0)	**MAC-Address:** 00:00:00:00:00:00 **RX:** 3.26 MB (15052 Pkts.) **TX:** 2.48 MB (28124 Pkts.)	Connect Stop Edit Delete
WAN (eth0)	**Uptime:** 0h 0m 0s **MAC-Address:** 00:00:00:00:00:00 **RX:** 0.00 B (0 Pkts.) **TX:** 138.73 KB (599 Pkts.)	Connect Stop Edit Delete

Copyright@2016 AcSiP

6.5 Change Wi-Fi AP SSID with USB Drive

Step 1: Create lks7688.cfg file

Create an ASCII text file named lks7688.cfg with following content:

wifi_ssid=THE_AP_SSID
wifi_key=THE_AP_PASSWORD

Change THE_AP_SSID to the AP name you want, and change THE_AP_PASSWORD to the password you want. Save this file to the root directory of a USB drive formatted in FAT32 file system format.+

Step 2: Attach the USB Drive to the board

Plug an OTG cable to the HOST port, and attach the USB drive to the OTG cable. Reboot and the AP name will change according to the file. Use THE_AP_PASSWORD you assigned to log-in into the AP.

Step 3: Reset and press WiFi button

- Hold the WiFi button
- Press MPU reset button once
- Release the WiFi button after the WiFi LED light up

Note that if you hold the WiFi button longer than 5 seconds, it registers as firmware update instead. So make sure you release the WiFi button shortly after Wi-Fi LED light up.

After those 3 steps, the system will then boot up with the SSID and password specified for bringing up the Wi-Fi.

Copyright@2016 AcSiP

6.6 Copying Files To the Board

 # Using SCP

SCP in OS X and Linux

scp command line tool should be already installed and ready to use. If it is not installed, you can use package managers such as MacPorts or Homebrew to install it. To use SCP, open the Terminal and issue following command:

scp ./helloworld root@mylinkit.local:/example/helloworld

In the above example, a file named helloworld from the current directory is copied to the path /example/helloworld in AI7688H. The SCP tool will instruct you to enter the Password of root account.

Using SCP to copy files in Windows

You can download tools that support SCP protocol. WinSCP is used in this guide. It provides both GUI and command line interfaces.

To use command line interface, type the following command in the Windows command line console:

winscp.com –hostkey="*" scp://root@myLinkt.local helloworld.py

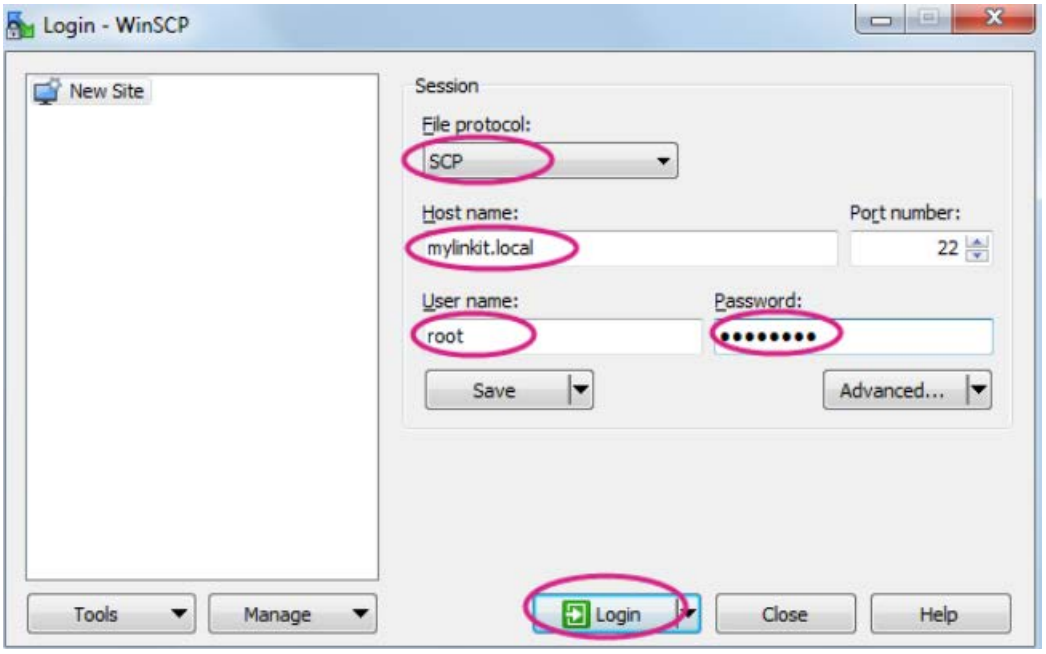
This copies the helloworld.py file to the home directory of the root account.

Note that by default it requires user to explicitly designate the host key of remote server, so we need to explicitly allow unknown host key by adding the -hostkey="*" parameter.

Copyright@2016 AcSiP

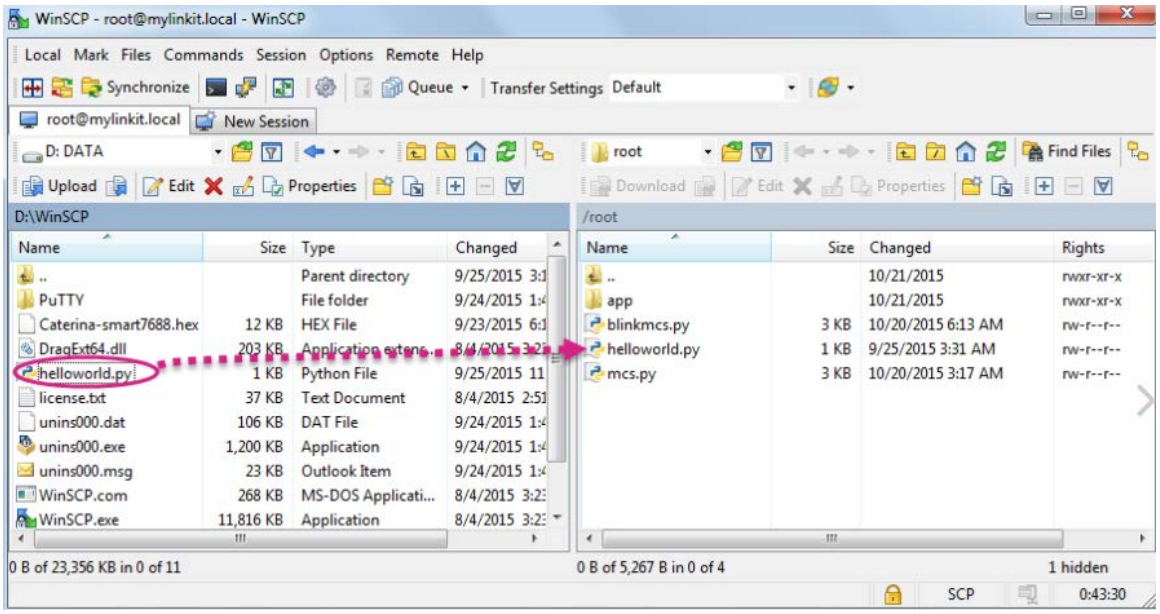
To use GUI,

start WinSCP program, select SCP for file protocol, enter mylinkit.local in the host name box, root for user name and the password you set in Web UI:



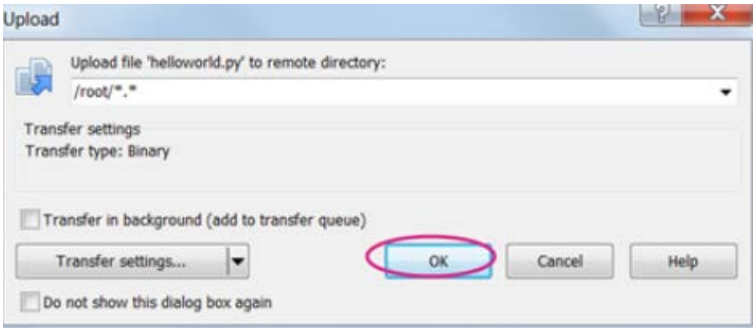
Click **Yes** when you see a warning window about *Continue connecting to an unknown server and add its host key to a cache?.+*

Locate the file you want to transfer on the left pane (your computer) and drag it to the right , as shown below:



Copyright@2016 AcSiP

After you've dragged the file, an Upload window will appear asking you to confirm the upload. Click **OK** to copy the file.



Using Samba

Samba is a networking tool that is built-in AI7688H and provides a file sharing service on the device for file transfer. In the example below you'll learn how to use UCI command to share a directory and set appropriate access permission for this directory.

UCI Configuration

This example shares a /IoT directory.

Change the shared folder path to /IoT. In AI7688H console, type the following command:

uci set samba.media.path='/IoT'

Name the shared folder MySharedFolder:

uci set samba.media.name='MySharedFolder'

Change the permission of the shared folder to make it readable and writable:

chmod o+rwx /IoT

Save the UCI settings and reboot the AI7688H platform:

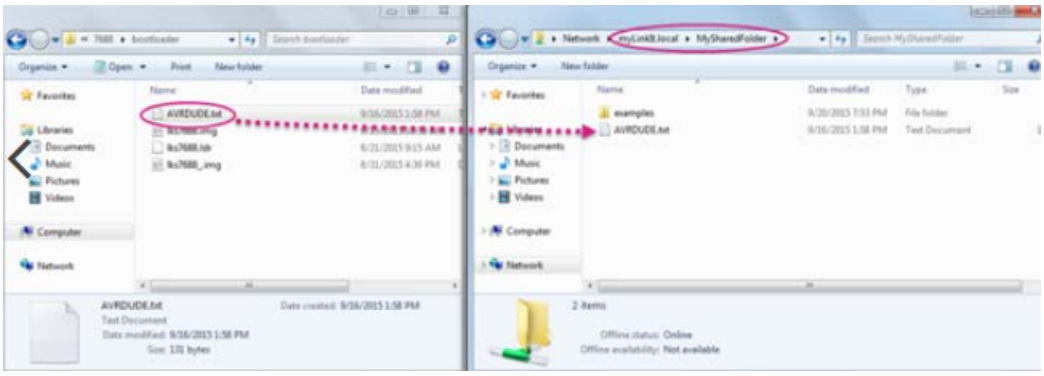
uci commit
reboot

After AI7688H reboots and is connected to the same local network as your computer, you are ready to use the Samba transfer tool. Check the steps according to your operating system below.

Copyright@2016 AcSiP

Access Samba in Windows

Open file explorer and type \\mylinkit.local, you should see the MySharedFolder. Open another file explorer, drag and drop your file to MySharedFolder, as shown below:



Access Samba in OS X

The steps to transfer files using Samba in Mac are as follows:

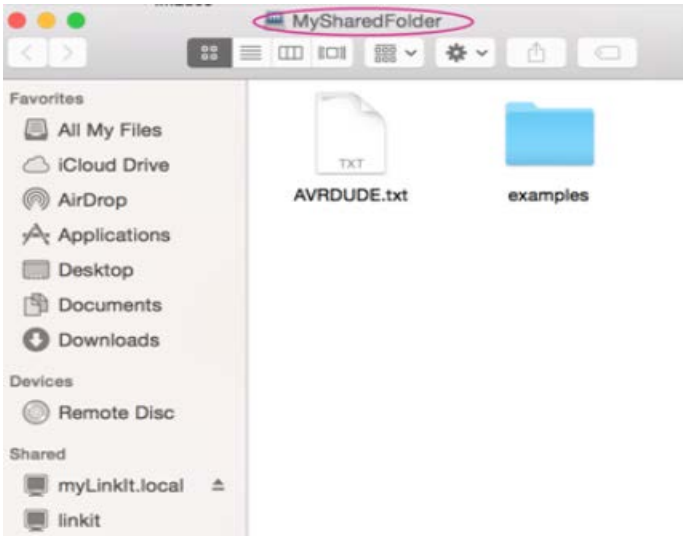
Open Finder and in the menu click Go > Connect to server

In the server address field, type smb://mylinkit.local and click **Connect**

Copyright@2016 AcSiP

Connect as guest:

Check Finder and you'll see MySharedFoloder. You can now transfer files in this folder.



6.7 iwpriv Commands

Access Point Scanning

-> Use set SiteSurvey = 1 to enable access point scanning. Note that it takes a while to scan nearby APs.

-> Use get_site_survey ra0 to collect scan results

Example

root@myLinkIt:/# iwpriv ra0 set SiteSurvey=1
root@myLinkIt:/# sleep 5
root@myLinkIt:/# iwpriv ra0 get_site_survey ra0

Example output:

```
get_site_survey:
Ch  SSID                                    BSSID              Security        Siganl(%)W-Mode  ExtCH  NT WPS DPID
1   router_66                       90:72:40:25:65:1c   WPA2PSK/AES                 91        11b/g/n NONE   In  NO
1   WILHELM.TEL-1XU533RM            34:31:c4:b3:ca:e8   WPA1PSKWPA2PSK/TKIPAES       7        11b/g/n NONE   In YES
1   Pluto                           36:31:c4:b3:ca:e8   WPA2PSK/AES                  5        11b/g/n NONE   In YES
11  OpenWrt                         18:aa:45:0b:85:88   WPA2PSK/TKIP               100        11b/g   NONE   In  NO
11  OpenWrt2                        1a:aa:45:00:85:88   WPA2PSK/TKIP               100        11b/g   NONE   In  NO
```


Copyright@2016 AcSiP

6.8 MAC Address Rules for MBSSID

The rules of MAC address adjustment for those virtual interfaces are:

1. Modify the 2nd bit of the most significant byte in the MAC address to **1** to make it as a locally administered MAC address.
2. Apply a MAC mask and add number for enumerating each interface.

The above two rules can be described as:

virtual_mac[i] = ((original_mac | 0x02 00 00 00 00 00) & mac_mask) + (i << 20)

And the mac_mask is defined as

* If BssidNum <= 2, mac_mask = 0xff ff ff ef ff ff
* If BssidNum <= 4, mac_mask = 0xff ff ff cf ff ff
* If BssidNum <= 8, mac_mask = 0xff ff ff 8f ff ff
* If BssidNum <= 16, mac_mask = 0xff ff ff 0f ff ff

For example, if the original MAC is **9C:65:F9:1B:13:62** and the current BssidNum is set to 4, then the newly virtualized MAC address for Station mode will be:

1. Apply the locally administered bit => 9E:65:F9:1B:13:62.
2. Apply mask => 9E:65:F9:0B:13:62.
3. We only need 1 interface for Station mode, so *i = 0*, and this makes 9E:65:F9:0B:13:62 + (0 << 20) = 9E:65:F9:0B:13:62.

As a result, the burned-in MAC address for AP mode (9C:65:F9:1B:13:62) will then become **9E**:65:F9:**0B**:13:62 when it's in Station mode.

6.9 Using the Wi-Fi Dongle

AI7688H can be used as a wireless Wi-Fi router – not just for providing an Ethernet port for Internet connection (by using a breakout board), but to provide a second Wi-Fi interface – by using a Wi-Fi dongle (the original on-board Wi-Fi is used for the AP connection between devices and AI7688H).This section walks you through how to install related packages and set up system configurations for using a Wi-Fi dongle for providing Internet connectivity on the AI7688H in AP mode. An EDIMAX Wi-Fi dongle (with Ralink

Copyright@2016 AcSiP

Wi-Fi chipset) is used in the following example, but the process is similar if you're using a different Wi-Fi dongle.

Before you start to install related packages to enable the Wi-Fi dongle, it's recommended to install a package called **usbutils** that provides information on the Wi-Fi dongle. To install **usbutils**, you need to configure the AI7688H to the Station mode first and then execute the **opkg update**:

```
root@mylinkit:/# opkg update
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/Packages.gz.
Updated list of available packages in /var/opkg-lists/chaos_calmer_base.
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/Packages.sig.
Signature check passed.
Downloading http://mirror2.openwrt.org/mt7688_v0.9/luci/Packages.gz.
Updated list of available packages in /var/opkg-lists/chaos_calmer_luci.
Downloading http://mirror2.openwrt.org/mt7688_v0.9/luci/Packages.sig.
Signature check passed.
Downloading http://mirror2.openwrt.org/mt7688_v0.9/packages/Packages.gz.
Updated list of available packages in /var/opkg-lists/chaos_calmer_packages.
Downloading http://mirror2.openwrt.org/mt7688_v0.9/packages/Packages.sig.
Signature check passed.
```

and **opkg install usbutils** commands to install the utility:

```
root@mylinkit:/# opkg install usbutils
Installing usbutils (007-1) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/usbutils_007-1_ramips_24kec.ipk
Configuring usbutils.
```

After the utility is installed, you'll need to install the driver package for the Wi-Fi dongle. Insert the Wi-Fi dongle into the USB port of the AI7688H and type the command **lsusb** to see the product information of the dongle:

```
root@mylinkit:/# lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 7392:7711 Edimax Technology Co., Ltd EW-7711UTn nLite Wireless Adapter [Ralink RT2870]
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

You'll see the Wi-Fi dongle is based on Ralink RT2870 Wi-Fi chipset from the output of the **lsusb** command. This information is important to identify the correct driver to be installed. Next, type the **opkg list | grep Ralink** command and you'll see all available packages related to Ralink, as shown below:

Copyright@2016 AcSiP

```
root@mylinkit:/# opkg list | grep Ralink
kmod-mt7628 - 3.18.21+4.0.1.3-1 - Ralink mt7628 wifi AP driver
kmod-rt2400-pci - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT2400 PCI)
kmod-rt2500-pci - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT2500 PCI)
kmod-rt2500-usb - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT2500 USB)
kmod-rt2800-lib - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (rt2800 LIB)
kmod-rt2800-mmio - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT28xx/RT3xxx MMI
kmod-rt2800-pci - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT2860 PCI)
kmod-rt2800-usb - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT2870 USB)
kmod-rt2x00-lib - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (LIB)
kmod-rt2x00-mmio - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (MMIO)
kmod-rt2x00-pci - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (PCI)
kmod-rt2x00-usb - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (USB)
kmod-rt61-pci - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT2x61 PCI)
kmod-rt73-usb - 3.18.21+2015-03-09-3 - Ralink Drivers for RT2x00 cards (RT73 USB)
```

From the available packages you're able to identify the driver needed which is the **kmod-rt2800-usb** package (One that applies to RT2870 USB device). So type the **opkg install kmod-rt2800-usb** command to install the related driver:

```
root@mylinkit:/# opkg install kmod-rt2800-usb
Installing kmod-rt2800-usb (3.18.21+2015-03-09-3) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-rt2800-usb_3.18.21+2015-03-09-3_ramips_24kec.ipk
Installing kmod-rt2x00-usb (3.18.21+2015-03-09-3) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-rt2x00-usb_3.18.21+2015-03-09-3_ramips_24kec.ipk
Installing kmod-rt2x00-lib (3.18.21+2015-03-09-3) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-rt2x00-lib_3.18.21+2015-03-09-3_ramips_24kec.ipk
Installing kmod-mac80211 (3.18.21+2015-03-09-3) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-mac80211_3.18.21+2015-03-09-3_ramips_24kec.ipk.
Installing kmod-crypto-core (3.18.21-1) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-crypto-core_3.18.21-1_ramips_24kec.ipk.
Installing kmod-crypto-arc4 (3.18.21-1) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-crypto-arc4_3.18.21-1_ramips_24kec.ipk.
Installing kmod-crypto-aes (3.18.21-1) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-crypto-aes_3.18.21-1_ramips_24kec.ipk.
Installing kmod-cfg80211 (3.18.21+2015-03-09-3) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-cfg80211_3.18.21+2015-03-09-3_ramips_24kec.ipk.
Installing iw (3.17-1) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/iw_3.17-1_ramips_24kec.ipk.
Installing kmod-lib-crc-itu-t (3.18.21-1) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-lib-crc-itu-t_3.18.21-1_ramips_24kec.ipk.
Installing kmod-rt2800-lib (3.18.21+2015-03-09-3) to root...
Downloading http://mirror2.openwrt.org/mt7688_v0.9/base/kmod-rt2800-lib_3.18.21+2015-03-09-3_ramips_24kec.ipk
Configuring iw.
Configuring kmod-crypto-core.
Configuring kmod-crypto-arc4.
```

After the installation process is done, switch AI7688H back to the AP mode and execute the **reboot** command. After the reboot process is done, open and edit the /etc/config/wireless file to configure the network setup. At the end of the file, you'll see configuration for **radio1** (the Wi-Fi dongle), as shown below:

Copyright@2016 AcSiP

```
config wifi-device radio1
    option type       mac80211
    option channel    11
    option hwmode     11g
    option path       '101c1000.ohci/usb2/2-1/2-1:1.0'
    option htmode     HT20
    # REMOVE THIS LINE TO ENABLE WIFI:
    option disabled   1

config wifi-iface
    option device     radio1
    option network    lan
    option mode       ap
    option ssid       OpenWrt
    option encryption none
```

Next, you'll do the network setup for the Wi-Fi dongle by applying the modifications as shown below to make AI7688H connect to an existing AP which is connected to the Internet (assuming the AP name is **the_test_ap**, and its encryption mode is **psk2** with password **12345678**):

```
config wifi-device radio1
    option type       mac80211
    option channel    11
    option hwmode     11g
    option path       '101c1000.ohci/usb2/2-1/2-1:1.0'
    option htmode     HT20

config wifi-iface
    option device     radio1
    option network    wan
    option mode       sta
    option ssid       the_test_ap
    option encryption psk2
    option key        12345678
```

Save this file and exit, then type the **wifi** command to restart the Wi-Fi interfaces. During the Wi-Fi interface restarting process, you might see errors similar to the one shown:

```
[ 375.390000] ieee80211 phy0: rt2x00usb_vendor_request: Error - Vendor Request 0x06 failed for offset 0x0404 with error -19
[ 377.460000] ieee80211 phy0: rt2800_wait_wpdma_ready: Error - WPDMA TX/RX busy [0xffffffff]
[ 377.470000] ieee80211 phy0: rt2800usb_set_device_state: Error - Device failed to enter state 4 (-5)
```


Copyright@2016 AcSiP

If you see the error, please type the **wifi** command again to re-activate the W-Fi interfaces until there is no such error and you see a messages similar to below:

```
[ 424.490000] wlan1: authenticate with f0:56:76:88:9e:e8
[ 425.030000] wlan1: send auth to f0:56:76:88:9e:e8 (try 1/3)
[ 425.040000] wlan1: authenticated
[ 425.060000] wlan1: associate with f0:56:76:88:9e:e8 (try 1/3)
[ 425.070000] wlan1: RX AssocResp from f0:56:76:88:9e:e8 (capab=0x411 status=0 aid=2)
[ 425.190000] wlan1: associated [ 425.200000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan1: link becomes ready
```

If you see the above message, it means the interface is activated successfully. Note that at this point, you've not yet connected to the **the_test_ap** AP for the Internet connection. From the above message, you can see the MAC address of the AP to connect (it's f0:56:76:88:9e:e8). Open the /etc/config/wireless file again and add one line for the **radio1** for the **BSSID** property as below:

```
config wifi-iface
    option device     radio1
    option network    wan
    option mode       sta
    option ssid       the_test_ap
    option encryption psk2
    option key        12345678
    option bssid      f0:56:76:88:9e:e8
```

Save and exit the file. Type the **wifi** command to connect to the **the_test_ap** again. At this point, you can test for Internet connection by using the **ping -c 5 www.mediatek.com** command. If you see a message similar to below:

```
root@mylinkit:/# ping -c 5 www.mediatek.com
PING www.mediatek.com (175.98.146.37): 56 data bytes
64 bytes from 175.98.146.37: seq=0 ttl=243 time=239.289 ms
64 bytes from 175.98.146.37: seq=1 ttl=243 time=138.998 ms
64 bytes from 175.98.146.37: seq=2 ttl=243 time=137.700 ms
64 bytes from 175.98.146.37: seq=3 ttl=243 time=137.466 ms
64 bytes from 175.98.146.37: seq=4 ttl=243 time=154.235 ms

--- www.mediatek.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 137.466/161.537/239.289 ms
```

You have now successfully configured the Wi-Fi dongle and established the Internet connection for the AI7688H AP.

Copyright@2016 AcSiP

6.10 Query IP / MAC with the Wi-Fi button

With firmware v0.9.3 and above, you can use the Wi-Fi botton to query the IP / MAC address through the UART0 port.

Note: the baudrate of the UART0 is 9,600. Please configure the serial monitor software correctly to see the output from this UART port.+

The behavior of this function is: when AI7688H is under **AP mode**, by pressing the button will output the MAC address information to the UART0 port. And when the platform is under **Station mode**, its IP address will be outputted to the UART port when the Wi-Fi button is pressed. If you watch the output on the UART0, you will see the output as below.

Under **AP mode**, the MAC address is printed:

AP MAC address: f0:56:76:88:9e:e8

Under **Station mode**, the IP address is printed:+

Device IP address: 192.168.2.3

For **AI7688H** users, you may need to do some wiring connections to listen to the data on the UART0 port. This sketch can help pass through the UART0 message to the Arduino serial output.

```
long linuxBaud = 9600;        // the baudrate for 7688 UART0

void setup() {
  Serial.begin(115200);       // open serial connection via USB-Serial
  Serial1.begin(linuxBaud);   // open serial connection to 7688 UART0
}

void loop() {
  int c = Serial.read();      // read from the USB-Serial

  if (c != -1) {
    Serial1.write(c);         // pass it to the 7688 UART0
  }

  c = Serial1.read();         // read from the 7688 UART0
  if (c != -1) {
    Serial.write(c);          // pass it to the USB-Serial
  }
}
```

After uploading this sketch to the board, open **Tools/Serial Monitor** in the Arduino IDE and set the baudrate to 115,200. Then you should be able to see the messages while pressing the Wi-Fi button.

Copyright@2016 AcSiP

7 AWS IoT

AWS IoT is a managed cloud service on AWS cloud.You can use AWS IoT Device SDK to write programs running on AI7688H connects to AWS IoT.

It provides 2 different SDKs, and their compatibility with AI7688H is shown below:

AWS IoT SDK for JavaScript , Embedded C

Environment Setup

To run the SDK examples you'll need an AWS account and optionally AWS CLI(Command Line Interface)

Use an SD card to expand the root file system. This is because AWS CLI and device SDK takes considerable amount of disk spaces.

Sign-up an AWS account

Install AWS CLI. You can install the CLI on AI7688H directly with pip install awscli or use your PC to install it.

Generates certificates required by the examples by following the instructions in AWS IoT Device SDK.

Copy the **certificates, private key, and root-CA** file to AI7688H to ~/certs directory. You can follow the instructions here to copy files.

AWS IoT SDK for JavaScript

Simply follow the npm installation steps:

```
npm install aws-iot-device-sdk
```

Note that there will be two missing optional dependencies utf-8-validate@1.2.1 and bufferutil@1.2.1 - this can be ignored.

Make sure you have copied proper certification files to ~/certs directory before running the examples.

Then you should be able to run the examples from the SDK:

Copyright@2016 AcSiP

```
node examples/thing-example.js -f ~/certs --test-mode=1
```

```
node examples/thing-example.js -f ~/certs --test-mode=2
```

AWS IoT Embedded-C SDK

Requirements

You'll need a computer running OS X or Ubuntu Linux. This is because you need the cross compilation toolchain to use the AWS IoT C SDK.

Building and Running SDK Examples

Follow the steps below to know how to build the example applications.

1. Download and setup the cross-compilation tools (**OpenWrt SDK**) . Make sure you can build and deploy a helloworld example without problem.
2. Locate the path to the cross compilation toolchain. If you extract the OpenWrt SDK in path WRT_SDK, the toolchain is in

   ```
   WRT_SDK/staging_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin
   ```
3. Download the AWS IoT Embedded-C SDK. There are 2 versions. In this example we download the **mbedTLS from ARM** version.
4. Extract the downloaded AWS IoT SDK, in this example we use the path AWS_SDK.
5. Use your AWS IoT Console, choose **"Connect a device"** to generate certification files. Download and store the certification files to AWS_SDK/certs directory.
6. Navigate the the **subscribe_publish_sample** sample application directory:

   ```
   cd AWS_SDK/sample_apps/subscribe_publish_sample
   ```
7. Copy the resulting configuration text to aws_iot_config.h, it will look similar to this:

```
// Get from console
// ================================================
#define AWS_IOT_MQTT_HOST              "A1XY4MZO1406ZZ.iot.us-west-2.amazonaws.com"
#define AWS_IOT_MQTT_PORT              8883
#define AWS_IOT_MQTT_CLIENT_ID         "linkit_device"
#define AWS_IOT_MY_THING_NAME          "linkit_device"
#define AWS_IOT_ROOT_CA_FILENAME      "root-CA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME   "b77f46f13c-certificate.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "b77f46f13c-private.pem.key"
// ================================================
```


Copyright@2016 AcSiP

8. Modify the Makefile of the sample application. Open the Makefile and find the following line in the beginning of the file: `CC = gcc`
   remove the line and modify it to use the cross compilation toolchain:

```
export STAGING_DIR = WRT_SDK/staging_dir/
CROSS_COMPILE = WRT_SDK/staging_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin/mipsel-openwrt-linux-
export CC = $(CROSS_COMPILE)gcc
export AR = $(CROSS_COMPILE)ar
export LD = $(CROSS_COMPILE)ld
export RANLIB = $(CROSS_COMPILE)ranlib
```

9. Type make to start building the sample application. It will build an executable file subscribe_publish_sample.

10. Make sure the board is connected to the internet and is in the same network as your computer. Now we copy the certificationfiles and the sample application to the board:

```
ssh root@mylinkit.local 'mkdir -p ~/aws_sdk/certs'
ssh root@mylinkit.local 'mkdir -p ~/aws_sdk/sample_apps/subscribe_publish_sample'
scp -r AWS_SDK/certs/* root@mylinkit.local:~/aws_sdk/certs
scp -r AWS_SDK/sample_apps/subscribe_publish_sample/subscribe_publish_sample root@mylinkit.local:~/aws_sdk/sample_apps/s
```

11. Then we can execute the sample application on the board. First we log into the console:

```
ssh root@mylinkit.local
```

   after login to the console, run the example:

```
cd ~/aws_sdk/sample_apps/subscribe_publish_sample
./subscribe_publish_sample
```

12. You should see the following output, which is repeatedly printed:

```
Subscribing...
-->sleep
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 0
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 1
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 2
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 3
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 4
-->sleep
```

   You can press ctrl-c to stop the execution.

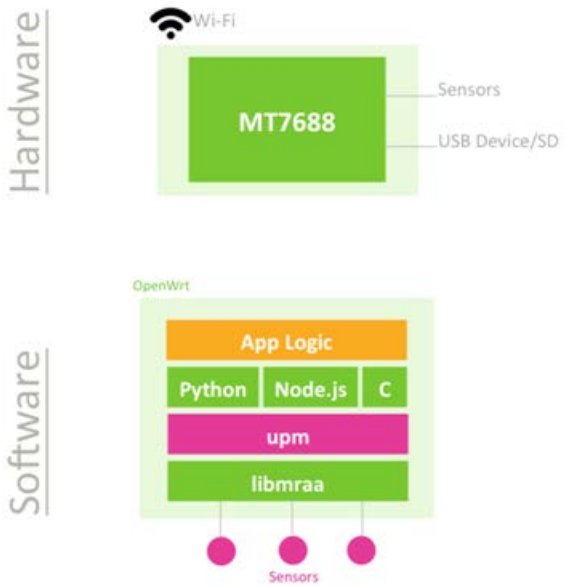
   Now you can build other sample applications as well. You need to modify the Makefile and aws_iot_config.h for each sample application as described in **step 7 and 8** above.

Copyright@2016 AcSiP

8 Peripheral

8.1 How to switch the Pin Mux

From the hardware aspect, AI7688H board handles all the Wi-Fi communication, USB device control, SD card access and sensor connection.



Related software stacks are provided for developers to access the sensors attached to the AI7688H.UPM is the repository for sensor drivers written in libmraa and it provides API bindings for Python, Node.js and C languages. So it's convenient for developers to use UPM to access peripheral sensors and modules with the programming language that they preferred.

AI7688H has built-in UPM support. The detailed support list of sensors in UPM can be found in the UPM project page.

Copyright@2016 AcSiP

8.2 Basic Concepts of MRAA

Libmraa is a C/C++ library to interface with the peripheral on AI7688H. Libmraa is pre-installed in the system image of AI7688H and supports C++, Python and Node.js bindings.

Installing MRAA:

Libmraa is already installed in the system image of AI7688H, so you don't need to install it again.

Basic Concepts :

The majority of hardware modules such as GPIO, UART, SPI, and PWM are represented as objects created by mraa's factory function.

These modules are initialized on certain pins that are identified by **pin numbers**.

The pin numbers in the libmraa on AI7688H are identical to the **GPIO number** in the data sheet and in the Linux GPIO subsystem. The following Python example creates GPIO object on GPIO 2:

```
import mraa
pin = mraa.Gpio(2)    # Initialize GPIO2 (P10 on LinkIt Smart 7688 board)
```

This maps to P10 of AI7688H and it's the IS2_WS pin in data sheet, as shown in table below.

GPIO Number	Datasheet	Silk print on LinkIt Smart 7688
2	I2S_WS	P10

8.3 Using MRAA in Python

To use libmraa in Python, you need to **import** it. In the below example, libmraa is imported and the output is the build version of the mraa:

```
import mraa
print (mraa.getVersion())
```

GPIO and Interupts

To control GPIO pins, initialize the pin as GPIO pin and set its *mode*. The simplest operation mode is OUTPUT - set the pin to **HIGH** or **LOW** to enable and disable external switches or to form signal patterns.

Copyright@2016 AcSiP

```
import mraa
pin = mraa.Gpio(2)    # Initialize GPIO2 (P10 on LinkIt Smart 7688 board)
pin.dir(mraa.DIR_OUT) # set as OUTPUT pin
```

Then, call pin.write(0) to set the pin state to **LOW** or call pin.write(1) to set the pin state to **HIGH**. To make the Wi-Fi LED blink periodically, set pin 44 (WLED_N) to GPIO mode and execute the following code

```
import mraa
import time

# Refer to the pinout digram for the GPIO number to silk print mapping
# in this example the number 44 maps to Wi-Fi LED.
pin = mraa.Gpio(44)
pin.dir(mraa.DIR_OUT)

while True:
    pin.write(1)
    time.sleep(1)
    pin.write(0)
    time.sleep(1)
```

There's another GPIO mode which is INPUT. It takes the digital signal input from the pin and interprets it into 1 and 0. This example will continuously print out the value received from P10 on the board. You can short 3V3 and P10 to observe the change in values

```
import mraa
import time

# Refer to the pinout digram for the GPIO number to silk print mapping
# in this example the number 2 maps to P10 on LinkIt Smart 7688 board
pin = mraa.Gpio(2)
pin.dir(mraa.DIR_IN)

while True:
    print "P10 state:", pin.read()
    time.sleep(0.3)
```

Finally, an interrupt service routine can be installed to the pin and invoked when the values of the input pin P10 (GPIO2) has changed. Call **isr API** with the trigger type you want to register and the function to be called. Note that the function runs in a **different thread**.

```
import mraa
import time

def callback(userdata):
    print "interrupt triggered with userdata=", userdata

pin = mraa.Gpio(2)
pin.dir(mraa.DIR_IN)
pin.isr(mraa.EDGE_BOTH, callback, None)

while(True):
    time.sleep(1)
    # simply wait for interrupt
```


Copyright@2016 AcSiP

PWM

Use PWM module to generate a pulse width modulated signal pattern.

This is useful to control actuator peripherals such as servo motors. To use PWM, initialize it on a certain pin such as GPIO. Note that only **GPIO18**, **GPIO19**, **GPIO20**, **GPIO21** supports PWM on AI7688H. To control the PWM pattern, several parameters are required, including:

Period

This defines the carrier frequency of the modulation. It's controlled by period, period_ms and period_us APIs.

Duty Cycle or Pulse Width

These two parameters are related to each other and usually you only need to set one of them.Duty cycle is controlled by write API with a value range between 0.0 to 1.0, where 0.0 is 0% of duty cycle and 1.0 is 100% of duty cycle.

Pulse width also defines the pattern in a different unit: the "uptime" of the signal in time units.

This is defined by pulsewidth, pulsewidth_ms, and pulsewidth_us APIs.

The following example generates a 500Hz PWM signal with 25% duty cycle on pin P26.

```
import mraa

pin = mraa.Pwm(18)  # initialize on GPIO18 (pin P26)
pin.period_ms(2)    # set PWM frequency to 500Hz (2ms period)
pin.enable(True)    # enable PWM output
pin.write(0.25)     # set duty cycle to 25%
```

I2C

I2C (Inter-Integrated Circuit) is a widely used protocol among peripherals. It consists of **2 signal pins** - usually named SDA and SCL. AI7688H comes with 1 set of I2C on GPIO4(P21) and GPIO5(P20) as SCL and SDA respectively. The way I2Cs are initialized is slightly different from GPIO modules - instead of using pins, I2Cs are initialized according to its **device index**. Since there is only 1 set of I2C master device on AI7688H, you can simply pass 0 - and it is always on pin GPIO4 and GPIO5.

```
import mraa
i2c = mraa.I2c(0)
```

I2C is capable of connecting multiple slave devices to a single I2C master. Each slave device is identified by a 7-bit address.

```
import mraa

i2c = mraa.I2c(0)
# Grove - 3-Axis Digital Accelerometer(+-16g)
# is a ADXL345 configured to I2C address 0x53.
i2c.address(0x53)
# The device ID should be
if 0xE5 == i2c.readReg(0x00):
    print "Grove - 3-Axis Digital Accelerometer found on I2C Bus"
else:
    print "Grove - 3-Axis Digital Accelerometer not found"
```


Copyright@2016 AcSiP

SPI

SPI (Serial Peripheral Interface) can also be used to control peripheral devices.

AI7688H it consists of 4 pins: SPI_MOSI(P22), SPI_MISO(P23), SPI_CLK(P24), SPI_CS1(P25).

It's important to note that the SPI device is also used for communicating with the internal flash storage on the board.

Therefore, developers should access the SPI functionality through SPI modules only and avoid treating these SPI pins as general GPIO. Otherwise, the flash storage may work incorrectly.

The SPI module in libmraa is initialized by device index, instead of pin number:

```
import mraa
spi = mraa.Spi(0)
```

8.4 Using UPM in Python

UPM is an open source sensor and peripheral driver repository based on **libmraa** APIs. Among the popular sensor drivers such as I2C accelerometers and many others are available from this repository.

AI7688H system image is pre-installed with UPM and you can start programming on existing sensors immediately – but if the default implementation is not available, you can use opkg package manager to update the UPM library.

UPM comes with bindings in C++, Python and Node.js. Let's get started with an example where you'll learn how to use UPM and Python to receive values from an I2C accelerometer – a Grove 3-Axis Digital Accelerometer (±16g).

#Connect the accelerometer to your board. If you have the breakout board, you can attach it to the I2C grove interface. If not, you can also connect the pins from the accelerometer to the corresponding pins GND, 3V3, SDA(P20) and SCL(P21) on AI7688H.

#Import pyupm_adxl345 module from the UPM repository in your program, you'll do this in the next step. This module is used because the Grove 3-Axis Digital Accelerometer (±16g) uses the ADXL345 chipset.

#Create a Python script adxl.py with following content

```
import pyupm_adxl345 as adxl
import time
device = adxl.Adxl345(0)
while True:
    device.update()
    a = device.getAcceleration()
    print "(x,y,z)=%5.1f, %5.1f, %5.1f" % (a[0], a[1], a[2])
    time.sleep(0.3)
```

Execute the Python script in system console by typing the following command:

```
python adxl.py
```


Copyright@2016 AcSiP

8.5 Using MRAA in node.js

libmraa comes with node.js bindings - so you can use it in node.js applications.

This example shows how to control the LED light with libmraa and node.js.

Installing MRAA

Libmraa and its Nodejs bindings are already installed in the system image of AI7688H, so you don't need to install it again.

Controlling LED with MRAA in Node.js

Our first tutorial is controlling the LED with a simple node.js program.

Steps

- Gain access to system console
- Create a nodejs app:

```
mkdir app
cd app
npm init
```

- Create a app.js file with following content:

```
var m = require('mraa');
var ledState = true;
var myLed = new m.Gpio(44);

myLed.dir(m.DIR_OUT);

function periodicActivity() {
  myLed.write(ledState ? 1 : 0);
  ledState = !ledState;
  setTimeout(periodicActivity, 1000);
}
periodicActivity();
```

- Execute the application:

```
node app
```

The WiFi LED (orange) should now start to blink every second.

Copyright@2016 AcSiP

8.6 Controlling PWM with MRAA on AI7688H

Configure PWM pin:

To use PWM, initialize it on a certain pin such as GPIO. Note that only **GPIO18**, **GPIO19**, **GPIO20**, **GPIO21** supports PWM on AI7688H. So the first step is to create a Pwm object by assigning the desired GPIO number.

```
var mraa = require('mraa');
var pwmPin = new mraa.Pwm(19);
```

Period

This defines the carrier frequency of the modulation. It's controlled by period, period_ms and period_us APIs, for example:

```
var mraa = require('mraa');
var pwmPin = new mraa.Pwm(19);
pwmPin.period_ms(20);          // 20ms period ==> 500Hz
```

Duty Cycle and Pulse Width

These two parameters are related to each other and usually you only need to set one of them. Duty cycle is controlled by write API with a value range between 0.0 to 1.0, where 0.0 is 0% of duty cycle and 1.0 is 100% of duty cycle. Pulse width also defines the pattern in a different unit: the "uptime" of the signal in time units. This is defined by pulsewidth, pulsewidth_ms, and pulsewidth_us APIs.

The following example generates a 500Hz PWM signal with 25% duty cycle on pin **P27**(GPIO19)

```
var mraa = require('mraa');
var pwmPin = new mraa.Pwm(19);  // GPIO19 (P27)
pwmPin.period_ms(20);           // 20ms period ==> 500Hz
pwmPin.pulsewidth_ms(10);       // 10ms pulse width over 20ms period ==> 50% duty cycle
```

You can also use config_percent to directly assign period and duty cycle:

```
var mraa = require('mraa');
var pwmPin = new mraa.Pwm(19);  // GPIO19 (P27)
pwmPin.config_percent(20, 0.5); // 20ms period with 50% duty cycle
```


Copyright@2016 AcSiP

Enable Output

After configuration you can enable the PWM output by calling enable(true):

```
var mraa = require('mraa');
var pwmPin = new mraa.Pwm(19);  // GPIO19 (P27)
pwmPin.period_ms(20);           // 20ms period ==> 500Hz
pwmPin.pulsewidth_ms(10);       // 10ms pulse width over 20ms period ==> 50% duty cycle
pwmPin.enable(true);            // Start sending PWM signal
```

Control a Servo

You can use PWM signal to control a servo. A pulse width of 1500us means the neutral position, while usually 1000ms and 2000ms being the minimum and maximum position. So the following code resets the servo motor to neutral position:

```
var mraa = require('mraa');
var pwmPin = new mraa.Pwm(19);  // GPIO19 (P27)
pwmPin.period_ms(20);
pwmPin.pulsewidth_us(1500);     // change to 1000 / 2000 for min/max positions
pwmPin.enable(true);
```


Copyright@2016 AcSiP

9 C/C++ Programming

9.1 Building C/C++ Programs with OpenWrt SDK

Building C/C++ binaries for AI7688H requires cross-compilation. We provide an OpenWrt SDK. Follow these steps to build an example C/C++ ipk file that can be installed with opkg command.

Requirement

Currently only Ubuntu Linux and OS X are supported. Windows with Cygwin is not supported. The following steps assume an Ubuntu Linux environment.

Step

First, download the SDK zip file

Unzip the package The name is quite long and we'll use SDK to denote its name.

```
sudo tar -xvjf SDK.tar.bz2
```

Note that sudo is mandatory – without it the file won't properly unpacked.

```
cd SDK
```

Copy the example helloworld directory to SDK/package folder. The folder structure should look like this:

```
SDK/package
   +helloworld              # Name of the package
        -Makefile           # This Makefile describes the package
        +src
            -Makefile       # This Makefile builds the binary
            -helloworld.c   # C/C++ source code
```

In the SDK directory, type make package/helloworld/compile to build the package. Once it is built:

- Navigate to SDK/bin/ramips/packages/base
- There should be a package file named helloworld_1.0.0-1_ramips_24kec.ipk
- Copy the .ipk file to the AI7688H
- In the system console of the board, navigate to the location of the .ipk file and type opkg install

```
helloworld_1.0.0-1_ramips_24kec.ipk
```


Copyright@2016 AcSiP

- There should be some installation messages. After installation completes, type helloworld and you'll see the string Hell! O'

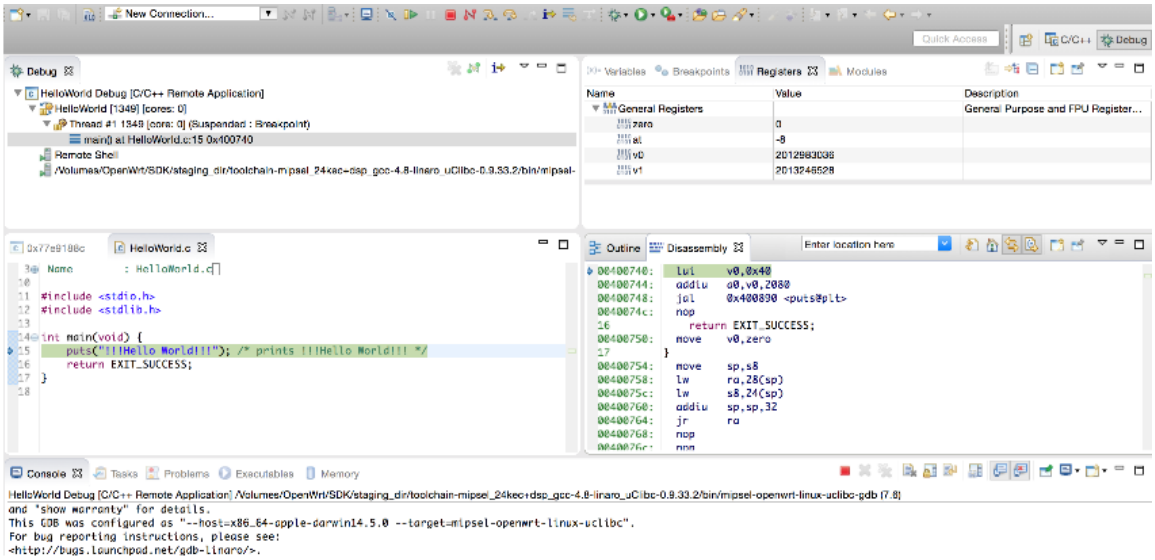
```
world, why won't my code compile?
```

If you want to cross-compile some known program or libraries, do this in the SDK directory:

```
./scripts/feeds update
./scripts/feeds list            # This gives you all the available pacakges
./scripts/feeds install curl    # for example we want to build curl
make package/curl/compile
```

9.2 Building and Debugging with Eclipse IDE

This article describes how to setup **Eclipse IDE for C/C++ Developers** and **OpenWrt SDK** to build and debug programs running on AI7688H.After proper setup, Eclipse will be able to build, upload and attach debugger to your C/C++ program that runs on AI7688H. This can be pretty helpful if you are writing your own C/C++ programs:



Copyright@2016 AcSiP

Step-by-step

Download and setup OpenWrt SDK

Enable SFTP on AI7688H

First we need to prepare AI7688H to enable SFTP service. This allows Eclipse IDE to upload binaries and debug files to the board.Make sure your board is connected to the internet and you have access to the system console. Use following commands to install and enable SFTP service.

```
opkg update
opkg install vsftpd openssh-sftp-server
/etc/init.d/vsftpd enable
/etc/init.d/vsftpd start
```

Install and Setup Eclipse IDE for C/C++ Developers

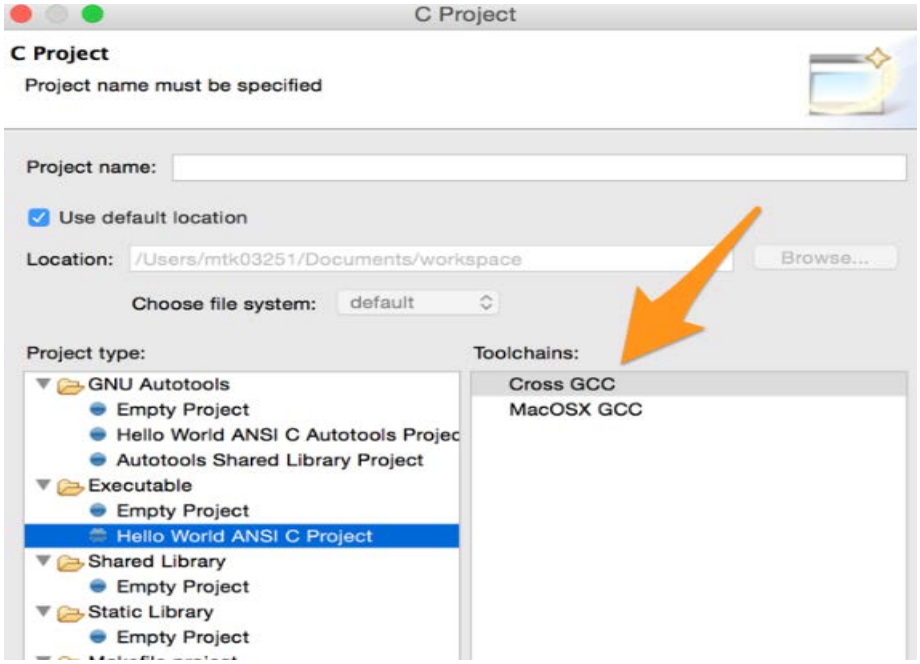
Create and configure C project

Now we create a *Hello World* project that uses cross compilation toolchain from the AI7688H OpenWrt SDK.

#Launch the IDE, choose File > New > Project...

#In the New Project wizard, choose C project and click Next to the project setting page.

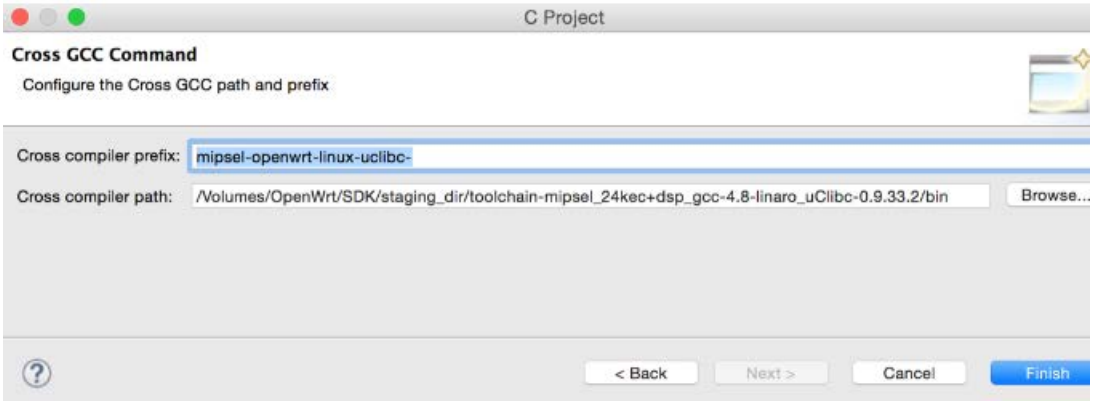
#Choose Hello World ANSI C Project and Cross GCC toolchain, as following screenshot.



- If you have no Cross GCC option, you need to install **C/C++ GCC Cross Compiler Support**.
- To install Cross Compiler Support, choose Help > Install new Software... and look for C/C++ GCC Cross Compiler Support(org.eclipse.cdt.build.crossgcc) and install it.

Copyright@2016 AcSiP

Continue the steps in the wizard until you see the **Cross GCC Command** setting page as following:



Fill the fields:

*Cross compiler prefix: mipsel-openwrt-linux-uclibc-

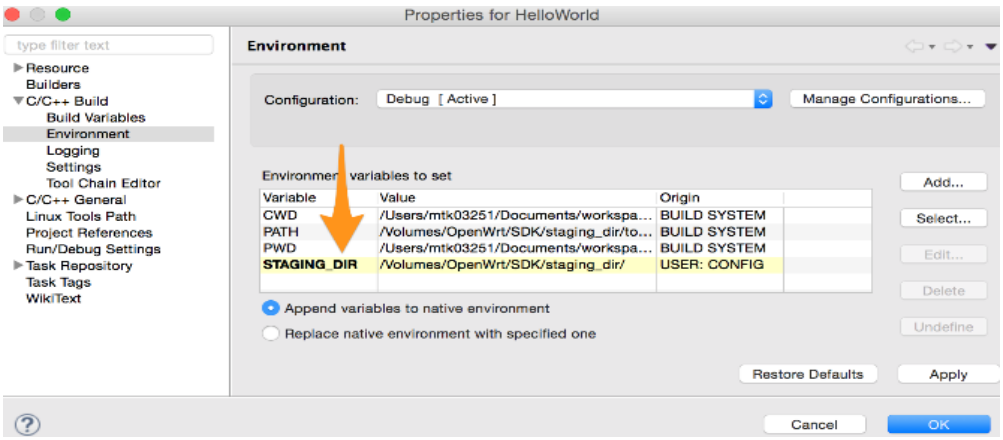
*Cross compiler path: OpenWrt_SDK/staging_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0,9.33.2/bin, where OpenWrt_SDK is the path to your OpenWrt SDK directory.

Click Finish and the wizard will create the project.

Now we setup the STAGING_DIR environment variable required by OpenWrt SDK. From the menu select Project > Properties. This brings up the Properties diaglog.

From the dialog, choose C/C++ Build > Environment. This page allows you to add additional environment variables required by the build systems. In this case we're going to add a new environment variable:

- Variable: STAGING_DIR
- Value: OpenWrt_SDK/staging_dir/ where OpenWrt_SDK is the path to your OpenWrt SDK directory See the following screenshot for example:



Copyright@2016 AcSiP

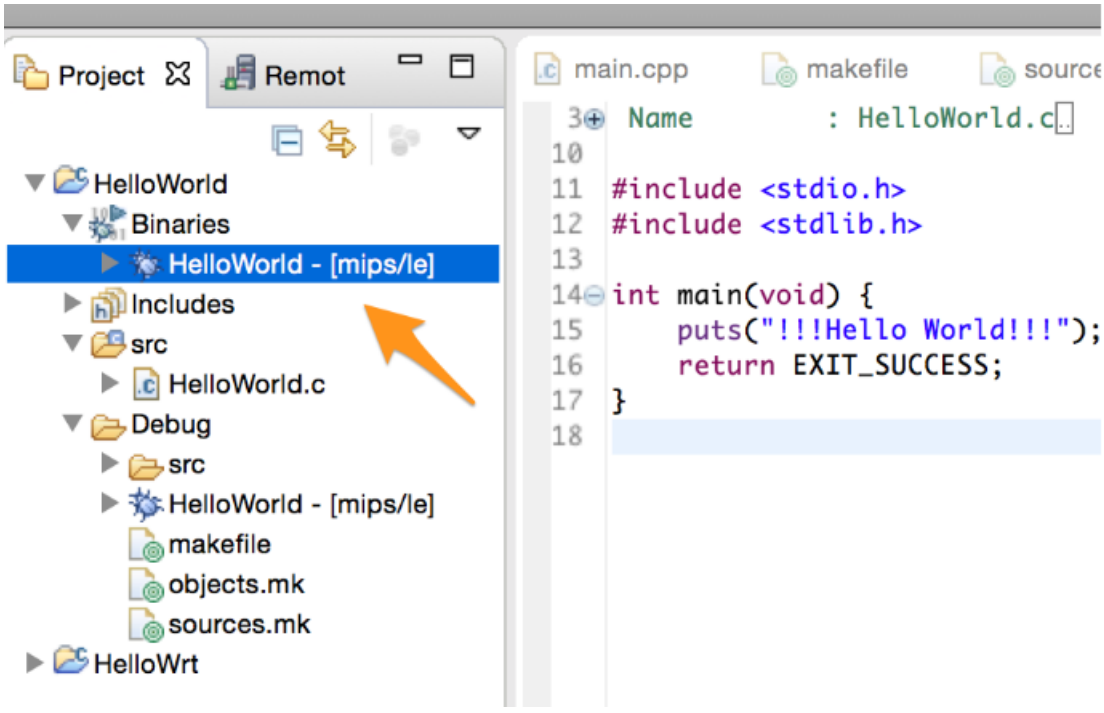
Build the Hello World program

Since the code is already generated by the wizard, we can start build directly. Choose Project > Build All from the menu. If the setup in previous steps are correct, you should see a build log similar to this:

```
15:41:52 **** Incremental Build of configuration Debug for project HelloWorld ****
make all
Building file: ../src/HelloWorld.c
Invoking: Cross GCC Compiler
mipsel-openwrt-linux-uclibc-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/HelloWorld.d" -MT"src/HelloWorld.o" -o "
Finished building: ../src/HelloWorld.c

Building target: HelloWorld
Invoking: Cross GCC Linker
mipsel-openwrt-linux-uclibc-gcc  -o "HelloWorld"  ./src/HelloWorld.o
Finished building target: HelloWorld
```

It will also generate a **mips/le** binary - this is the executable that we're going to debug in the next step.



Copyright@2016 AcSiP

Debug the Hello World program

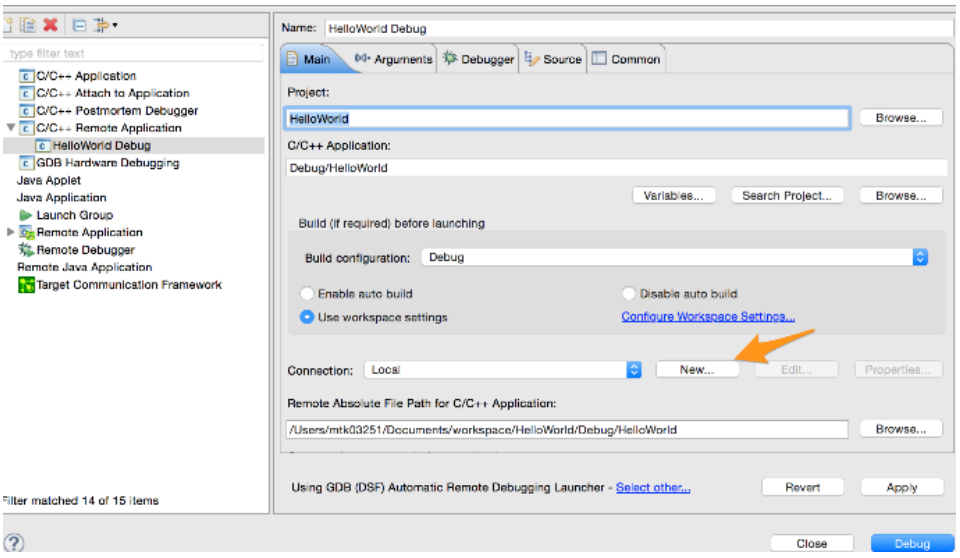
Now we'll configure the IDE to make it upload the executable file automatically and remote debug with gdb.

Make sure your computer is in the same Wi-Fi network as AI7688H. In this example we'll use the default domain name mylinkit.local.
You'll need to change to your settings accordingly. Make sure you can access the system console with ssh.

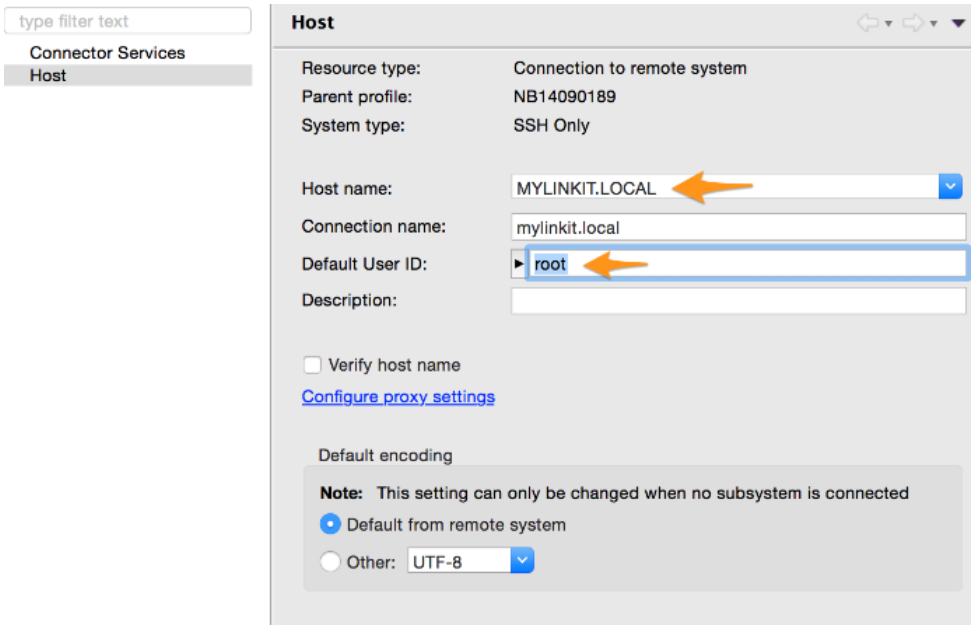
Choose Run > Debug Configurations... from the menu.

In the dialog, select C/C++ Remote Application and then click New button.

Choose New... connection:



and set **Host name** to mylinkit.local. Also set **Default User Id** to root



Copyright@2016 AcSiP

Change the setting **Remote Absolute File Path for C/C++ Application** to /tmp/HelloWorld. You may also use other path if you wan to.

Now select the **Debug** tab. Set the **GDB Debugger** path to OpenWrt_SDK/staging_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin/mipsel-openwrt-linux-uclibc-gdb where OpenWrt_SDK is the path to the OpenWrt SDK.

Click Apply then Debug. The IDE will now upload the built executable to AI7688H through SFTP service. Enter to root password of AI7688H if the IDE prompts you to do so.

The IDE now starts a gdbserver on the board, then connect Eclipse debug view to it. By default it will break at the first line of main function.

And that's it - now you should be able to extend the program and can debug the program with Eclipse.

Copyright@2016 AcSiP

10 Using USB Webcam

By connecting a USB webcam to the USB host port on AI7688H, users can easily setup the video streaming service according to the following steps.

Supported cameras

AI7688H has installed Linux UVC (USB Video Class) drivers to provide USB webcam suppport. Webcams follow the UVC standard can be supported on AI7688H. The real capabilities and supported resolution depend on the current UVC driver implementation.

Step-by-step

we use *mjpg-streamer* as the streaming application and it's already installed in the system firmware by default. For the webcam, Logitech C310 is used in this example.

Step 1: plug the webcam into the USB host port on AI7688H

Step 2: type the following command in the system console

```
# mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 25" -o "output_http.so -p 8080 -w /www/webcam" &
```

Step 3: connect the PC to the same local network as AI7688H

If AI7688H is in AP mode, connect the PC to AI7688H AP directly.

If AI7688H is in Station mode, connect the PC to the same AP that AI7688H connects to.

Step 4: open a brower and check the video

Copyright@2016 AcSiP

11 Audio Playback and Recording

AI7688H has an I2S interface for audio playback and recording. You'll need an audio DAC to convert I2S to analog sound data.

Note: the recording function is only supported with firmware **v0.9.3** and above.

Audio Playback

MP3 playback

To play a MP3 file, use madplay:

```
# madplay "path_to_your_mp3_file"
```

WAV playback

To play a WAV file, use **aplay** as below:

```
# aplay -M "path_to_your_wav_file"
```

Audio recording

WAV recording

Note: for a high bit-rate WAV recording (like 16bit/44.1k format as the below example), please record the file to a destination with **high I/O speed** (e.g. USB drive, SD card, or RAM) instead of the on-board flash. Due to the low writing speed of the on-board flash, users will experience sound jittering and buffer overrun if the recorded file is written to the on-board flash.

To record an audio file, use **arecord** as below:

```
# arecord -f cd -t wav -M /Media/USB-A1/my_recording.wav
```


Copyright@2016 AcSiP

12 Federal Communication Commission Interference Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FCC Caution:

- Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.
- This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Radiation Exposure Statement:

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

This device is intended only for OEM integrators under the following conditions:

1) The antenna must be installed such that 20 cm is maintained between the antenna and users, and the maximum antenna gain allowed for use with this device is 2 dBi.

2) The transmitter module may not be co-located with any other transmitter or antenna.

As long as 2 conditions above are met, further transmitter test will not be required. However, the OEM

Copyright@2016 AcSiP

integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed

IMPORTANT NOTE: In the event that these conditions can not be met (for example certain laptop configurations or co-location with another transmitter), then the FCC authorization is no longer considered valid and the FCC ID can not be used on the final product. In these circumstances, the OEM integrator will be responsible for re-evaluating the end product (including the transmitter) and obtaining a separate FCC authorization.

End Product Labeling

This transmitter module is authorized only for use in device where the antenna may be installed such that 20 cm may be maintained between the antenna and users. The final end product must be labeled in a visible area with the following: "**Contains FCC ID: 2ADWC-AI7688H**". The grantee's FCC ID can be used only when all FCC compliance requirements are met.

Manual Information To the End User

The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user's manual of the end product which integrates this module. The end user manual shall include all required regulatory information/warning as show in this manual.

Figure 1 below details the standard product marking for all AcSiP Corp. products. Cross reference to the applicable line number and table for a full detail of all the variables.

Figure 1 Standard Product Marking Diagram- TOP VIEW

Copyright@2016 AcSiP