AI7688H/
EK-AI7688H
Reference Manual



Document Name	AI7688H Reference Manual
Version	V 2.0
Date	2019/08/08

Document Name	AI7688H Reference Manual
Version	V 2.0
Date	2019/08/08

Document History

Revision	Date	Author	Description
V.1.0	**160407**	**PW**	**New Create**
V.2.0	**190808**	**Ken**	**Update Image**

Index

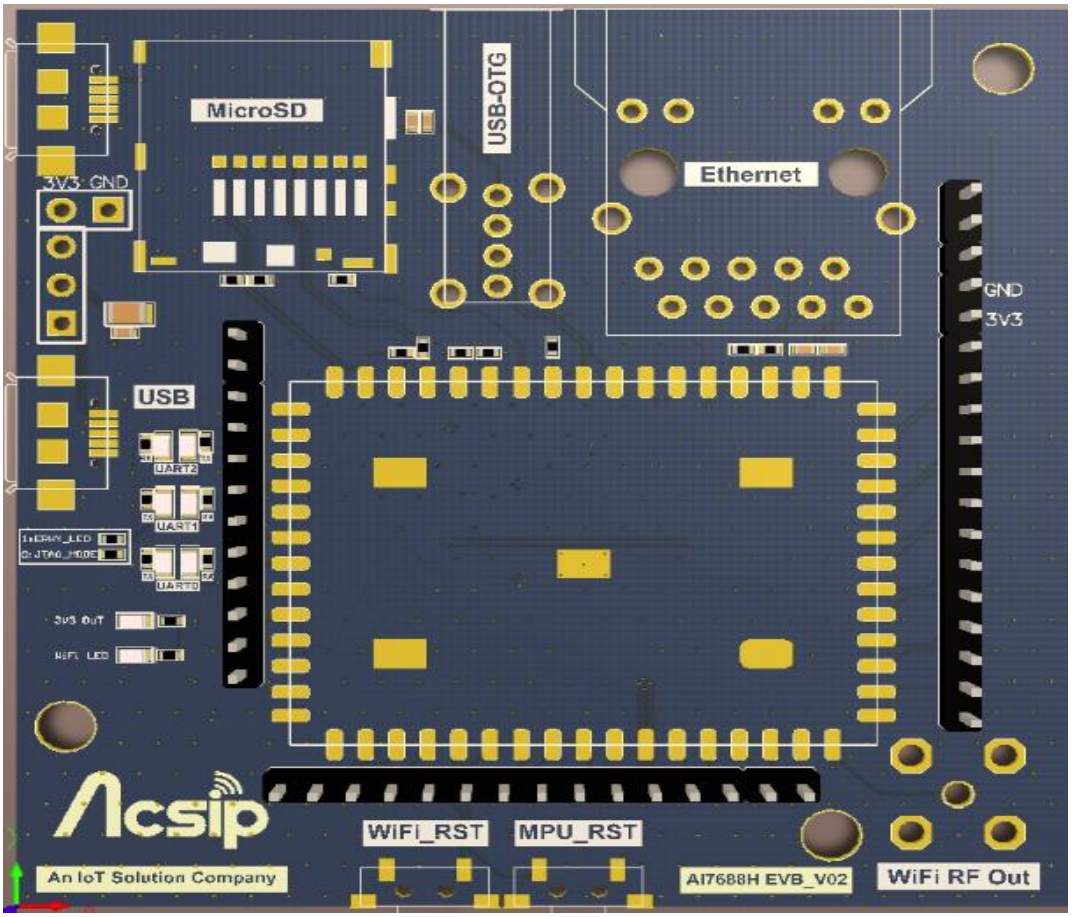
1 Introduction
2 Start AI7688H development
2.1 Software setup
2.2 Power up AI7688H board
2.3 Find the AI7688H AP and connect
2.4 System Information and Upgrade Firmware
2.5 OpenWrt for advanced configuration
2.6 Access system console
3 Firmware and Bootloader
3.1 Update Firmware with USB drive
3.2 Update bootloader with USB drive
3.3 Factory reset
3.4 Build Firmware from Source
3.5 Build Bootloader from Source
3.6 Rebuild Existing Kernel Packages
3.7 Bootloader and Kernel Console
3.8 Change bootloader console
3.9 Change Kernel Console
4 File Storage
4.1 USB Drive and SD Card
4.2 How to mount the SD card
5 Wi-Fi LED State
6 Network
6.1 Reset Wi-Fi Configuration
6.2 Switch to Station mode
6.3 Switch to AP mode
6.4 Setup Wireless Router
6.5 Change Wi-Fi AP SSID with USB Drive
7 Copying Files To the Board
7.1 Using SCP
7.2 Using Samba

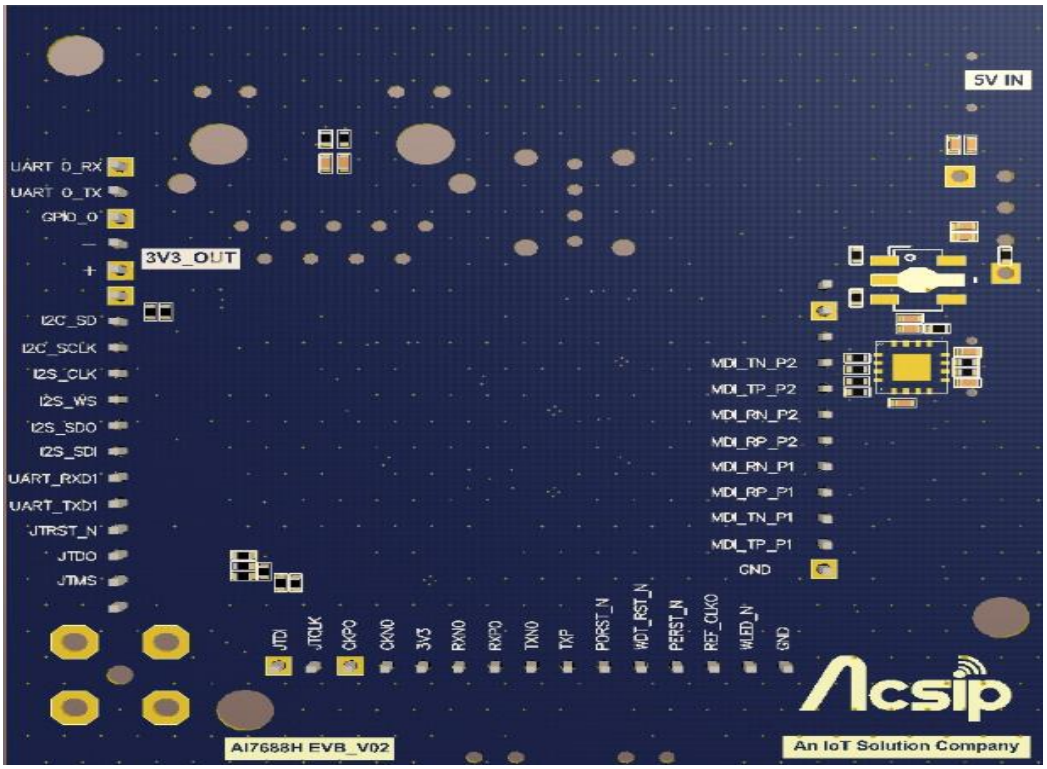
8 iwpriv Commands
9 AWS IoT
10 Peripheral
10.1 How to switch the Pin Mux
10.2 Basic Concepts of MRAA
10.3 Using MRAA in Python
10.4 Using UPM in Python
10.5 Using MRAA in node.js
10.6 Controlling PWM with MRAA on AI7688H
11 C/C++ Programming
11.1 Building C/C++ Programs with OpenWrt SDK
12 Using webcam and audio
12.1 Supported cameras
12.2 Audio Playback and Recording

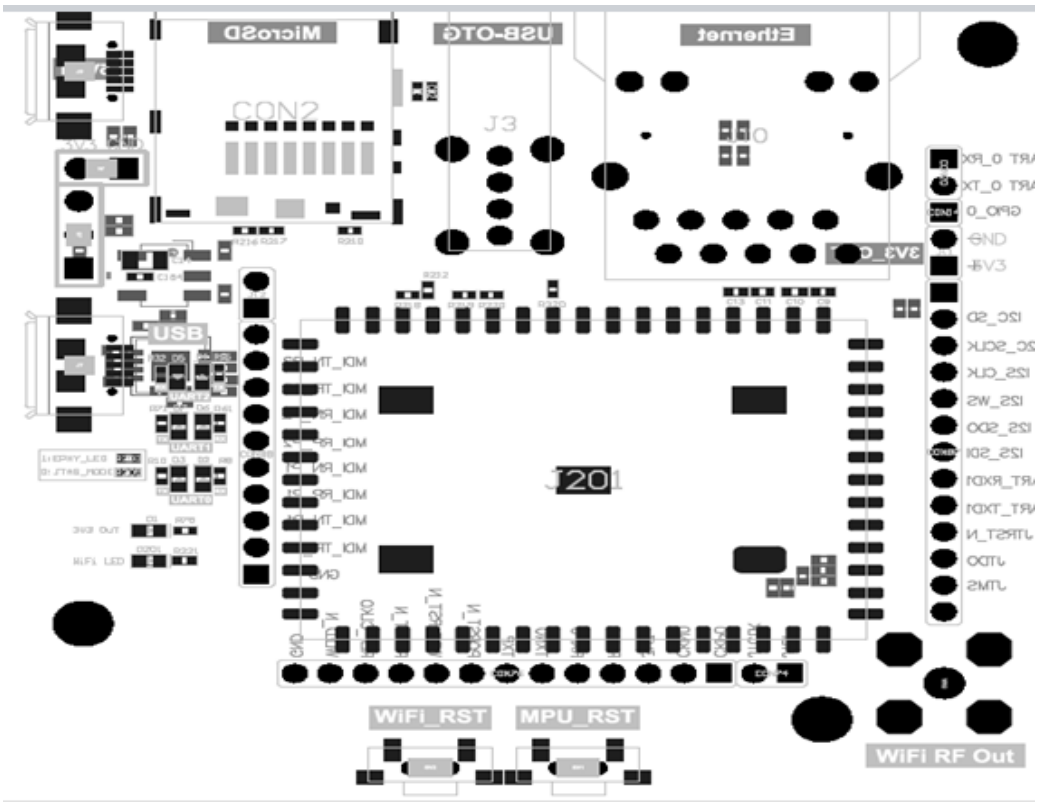
1 Introduction

AI7688H integrates a 1T1R 802.11n Wi-Fi radio, a 575/580 MHz MIPSR 24KEc™ CPU, 1-port Fast Ethernet PHY, USB2.0 host, PCIe, SD-XC, I2S/PCM and 3 UARTs. 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, AI7688H supports SD-XC and USB2.0. AI7688H can support the I2S interface at 192kbps@24bits. In IoT device mode, it further supports PWM, SPI slave, 3rd UART and more GPIOs.

EVK Pinout




5V IN
UART_0_RX
UART_0_TX
GPIO_0
3V3_OUT
I2C_SD
I2C_SCLK
I2S_CLK
I2S_WS
I2S_SDO
I2S_SDI
UART_RXD1
UART_TXD1
JTRST_N
JTDO
JTMS
MDI_TN_P2
MDI_TP_P2
MDI_RN_P2
MDI_RP_P2
MDI_RN_P1
MDI_RP_P1
MDI_TN_P1
MDI_TP_P1
GND
JTDI
JTCLK
GPIO
GND
3V3
RXN0
RXP0
TXN0
TXP
PORST_N
WDT_RST_N
PERST_N
REF_CLK0
WLED_N
GND
AI7688H EVB_V02
An IoT Solution Company

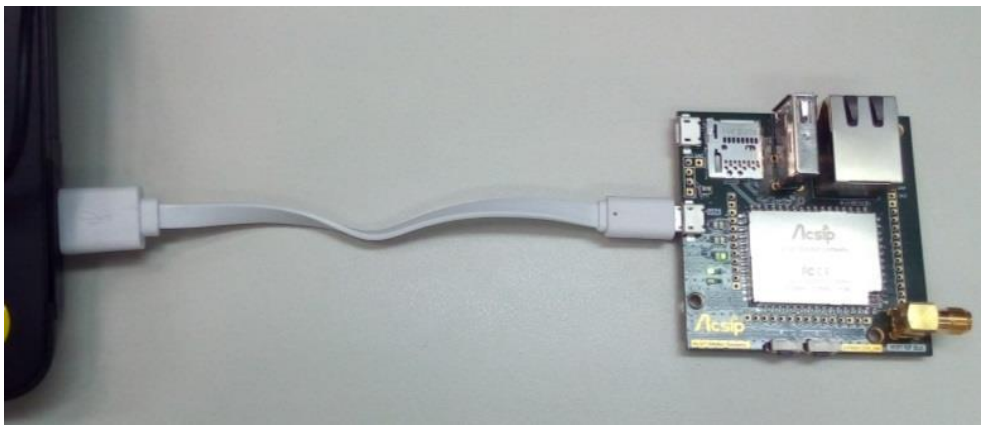

MicroSD
USB-OTG
Ethernet
CON2
J3
USB
J201
WiFi_RST
MPU_RST
WiFi RF Out

2 Start AI7688H development

2.1 Software setup

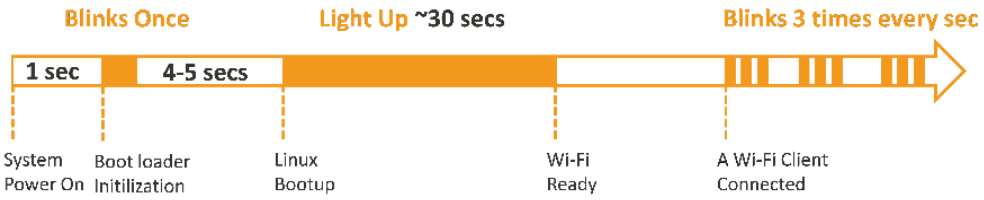
- Download an SSH client (Windows only)
  PuTTY provides Secure Socket Shell (SSH) access to the development board operating system.
- Install Bonjour Print Service (For Windows 7 only)
  The AI7688H development board uses mylinkit.local as its local domain. In Windows7, you'll need to install Bonjour print service for mDNS.
  This helps your computer discover the EK-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.
- PuTTY for Microsoft Windows.
  PuTTY provides you with the system console environment using Secure Socket Shell (SSH) access to the development board's operating system.
- WinSCP for windows
  WinSCP provides you with the file transfer support under SSH protocol. You can transfer file into AI7688H board.

2.2 Power up AI7688H board



After boot up and Wi-Fi initialization completes, which takes about 30 seconds, the Wi-Fi LED goes off.

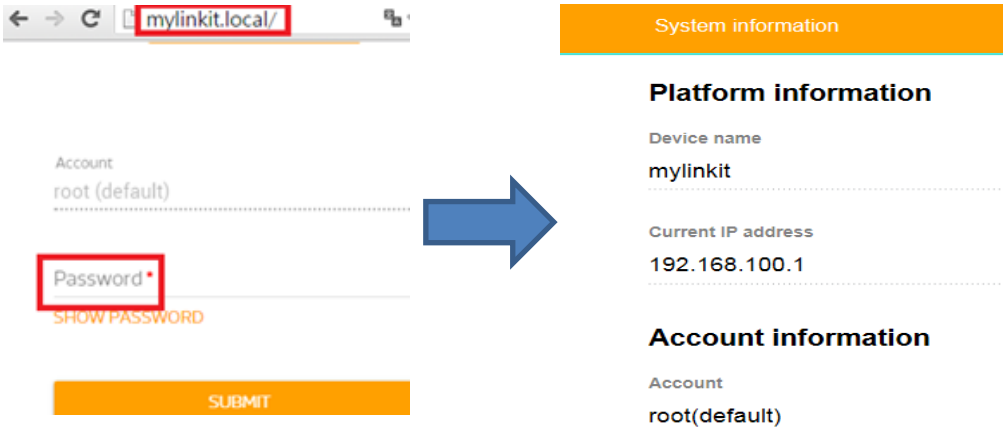
This means the system is ready to accept Wi-Fi connection – now we can connect to it. Following Figure shows how the Wi-Fi LED status matches the system state.



2.3 Find the AI7688H AP and connect



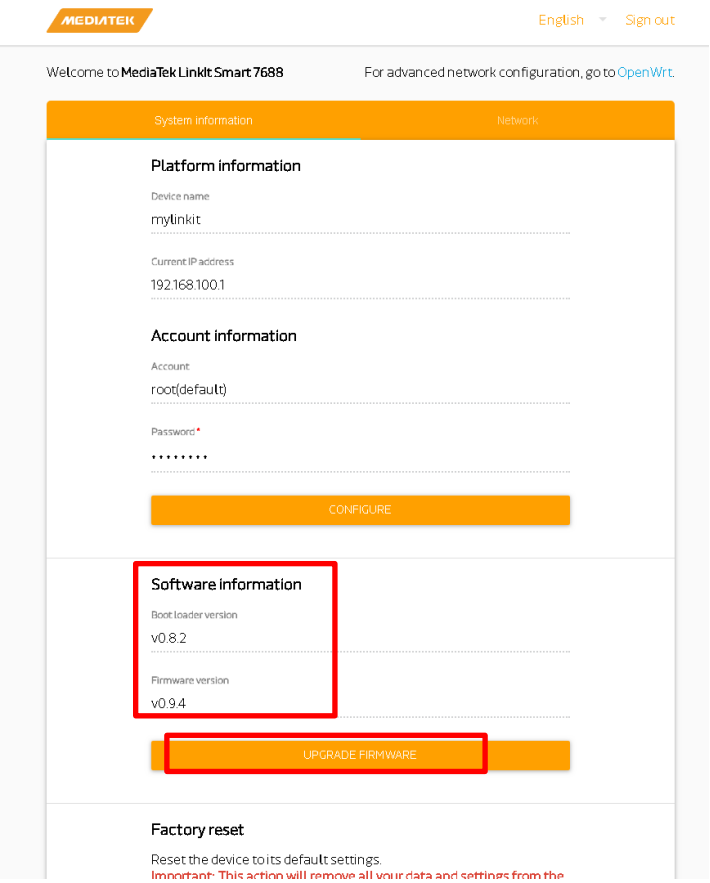
Open browser http://mylinkit.local or http://192.168.100.1



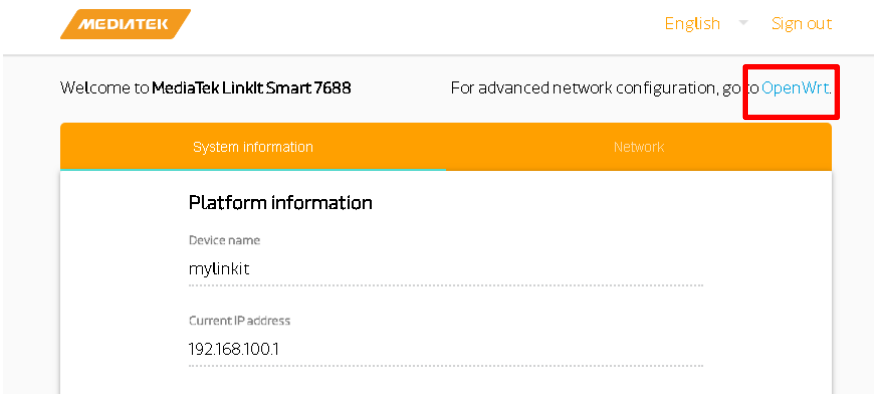
If the board already has a password, you can press “Wi-Fi Reset button” for at least 20 seconds then Release. Then you can configure your password again.

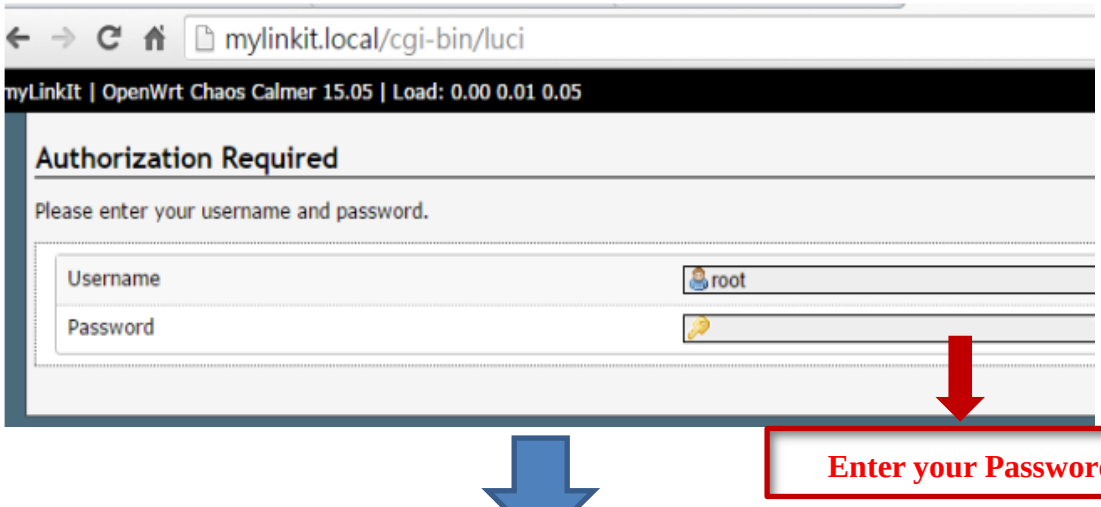
2.4 System Information and Upgrade Firmware

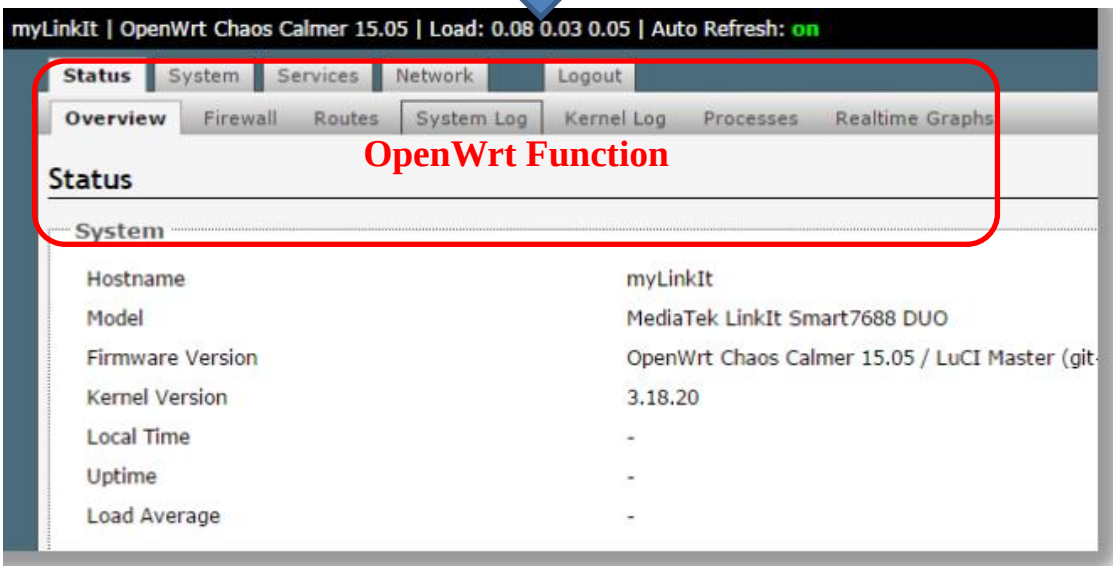
Download lks7688.img(version 0.9.4) from here.



2.5 OpenWrt for advanced configuration







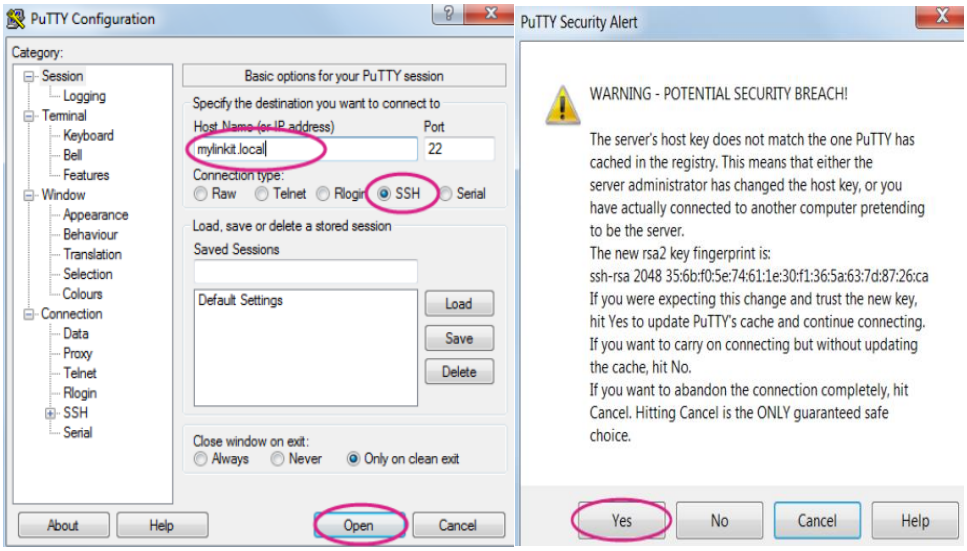
2.6 Access system console

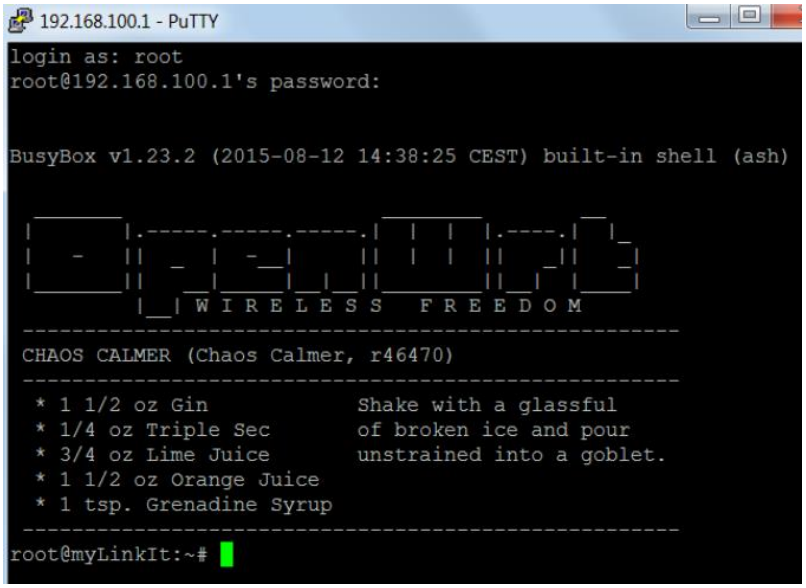
For OS X and Linux:

- 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:

- **Open PuTTY**





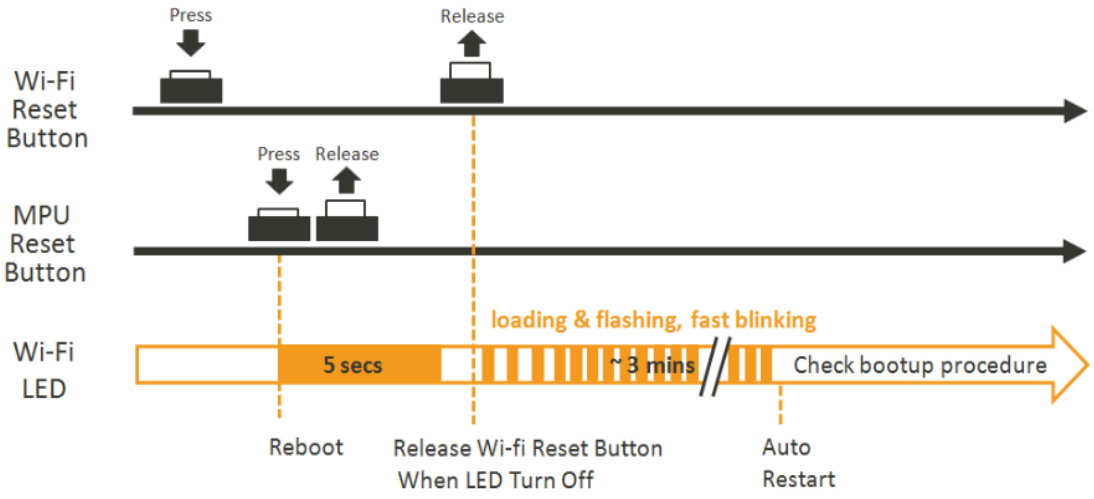
3 Firmware and Bootloader

3.1 Update Firmware with USB drive

- copy 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 Wi-Fi button
- Press the MPU Reset button once while holding the Wi-Fi button
- Keep holding Wi-Fi button about 5 seconds.
  Release Wi-Fi button until the Wi-Fi orange LED becomes OFF.

Do not press the Wi-Fi button for more than 20 seconds or it will upgrade the bootloader. Wait while the Wi-Fi LED blinks fast. This takes about 3 minutes.

The device will automatically reboot after firmware update completed.



3.2 Update bootloader with USB drive

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

- Download bootloader image file, Please refer to ”09_Firmware”.
- Save the bootloader file (uboot_<version>.bin) in the root directory of a USB drive and rename it “lks7688.ldr”.
- Plug the USB drive to AI7688H.
- Press the Wi-Fi and MPU (Reset) button at the same time, then release the MPU Reset button but hold the Wi-Fi button for at least 20 seconds.
- After 20 seconds Wi-Fi LED will turn on. Release Wi-Fi button.
- The board will start to read the bootloader (Wi-Fi 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.
- If bootloader is missing or damaged, it is no way to recovery by TFTP, UART, or USB.

3.3 Factory reset

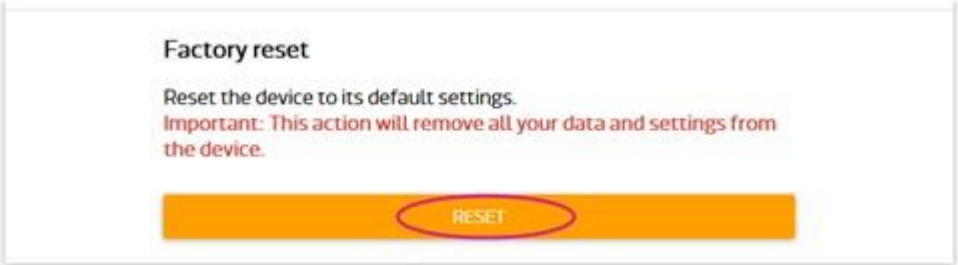
You can reset the firmware to its original state with a factory reset.

WARNING

Performing a factory reset erases all data and system changes you've made to the firmware, such as root password, Wi-Fi configuration and opkg installation.

Using the Web UI

Login to WebUI, under Factory reset, click RESET.



Using the Wi-Fi reset button

Boot up the board and wait for the Wi-Fi LED to go off.

Press and hold the Wi-Fi button for at least 20 seconds then release.

The Wi-Fi LED will blink fast for 1 second, and then reboot to perform factory reset.

3.4 Build Firmware from Source

The following operations are performed under a Ubuntu LTS 14.04.3 environment and 60 GB free disk space. For a Windows or a Mac OS X host computer, you can install a VM for having the same environment:

- Download to user's home directory "10_SDK\10_2_OpenWRT_SDK\ AI7688H_OpenWRT_Build_FW_Setting_Tool_V1.0.tar.gz".
- tar xvfz AI7688H_OpenWRT_Build_FW_Setting_Tool_V1.0.tar.gz (to extract file)
- cd AI7688H_OpenWRT_Build_FW_Setting_Tool_V1.0 (change to the folder.)
- cat Readme.txt (install additional package to build code.)
- make download (download code from internet)
- make menuconfig

```
    Target System (Ralink RT288x/RT3xxx)  --->
    Subtarget (MT7688 based boards)  --->
    Target Profile (LinkIt7688)  --->
    Target Images  --->
    Global build settings  --->
[ ] Advanced configuration options (for developers) (NEW)  ----
[ ] Build the OpenWrt Image Builder (NEW)
[ ] Build the OpenWrt SDK (NEW)
[ ] Package the OpenWrt-based Toolchain (NEW)
[ ] Image configuration (NEW)  --->
    Base system  --->
```

- Save and exit (use default configuration filename without change it)
- Start the compilation process: make -j1 V=s
- 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

3.5 Build Bootloader from Source

The following operations are performed under an 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:

- Download the bootloader source codes:

  git clone https://github.com/AcSiP/linkit-smart-7688-uboot.git

- Change to the source folder:

  cd linkit-smart-7688-uboot

- Install the toolchain (install to path /opt) for building the bootloader:

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

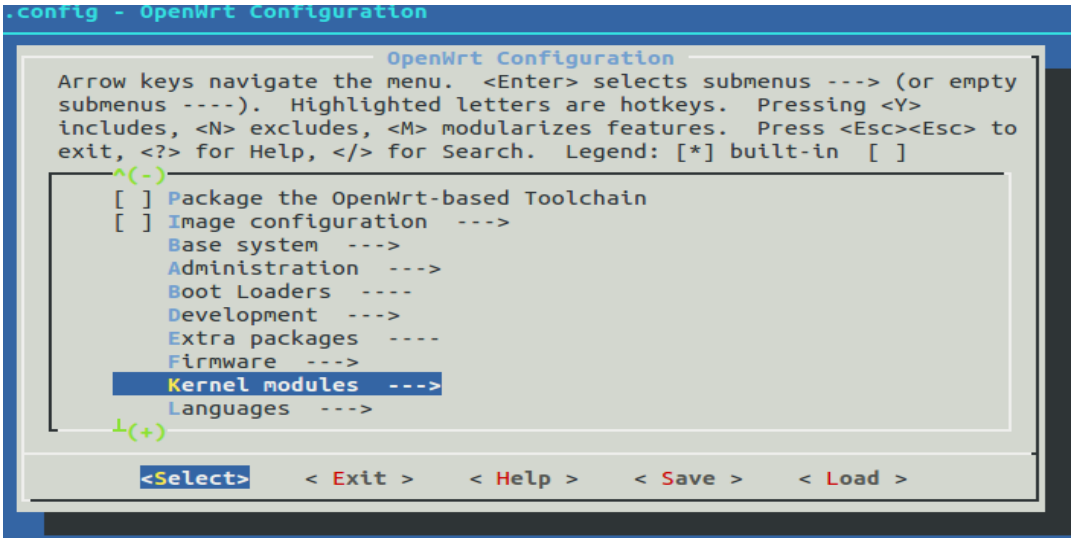
- Since the toolchain is provided in 32-bit executables, you need to install additional packages for execution on a 64-bit machine:

  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
- Bootloader filename is uboot.bin.
- You can rename it to “lks7688.ldr” for upgrading the system bootloader through a USB drive.

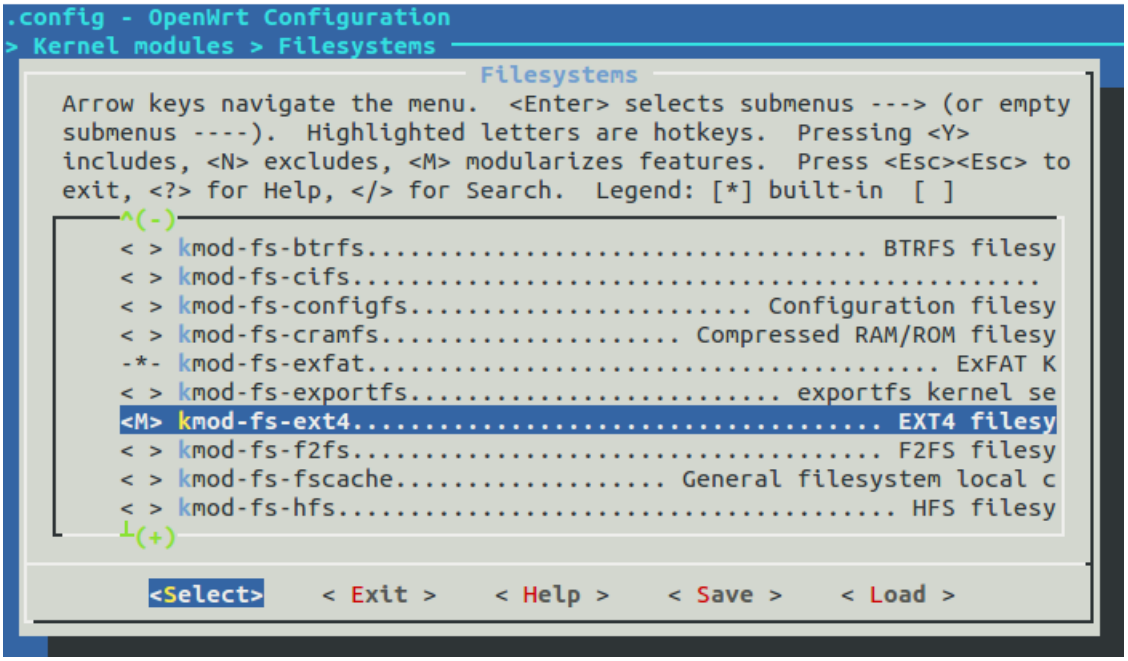
3.6 Rebuild Existing Kernel Packages

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.

- 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 below.
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/

3.7 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.
```

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



3.8 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

```
#define SERIAL_CLOCK_DIVISOR 16
#define CONFIG_BOOTDELAY       1
#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**

3.9 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.

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 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
```

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 editing mode)+

10. Save and exit the configuration file
    (press "**Esc**" key and type "**:x**". Then press "**Enter**" in vi for saving the file).
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. Remember you can’t boot up AI7688 from SD card.

5 Wi-Fi LED State

After powering 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 boot up 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

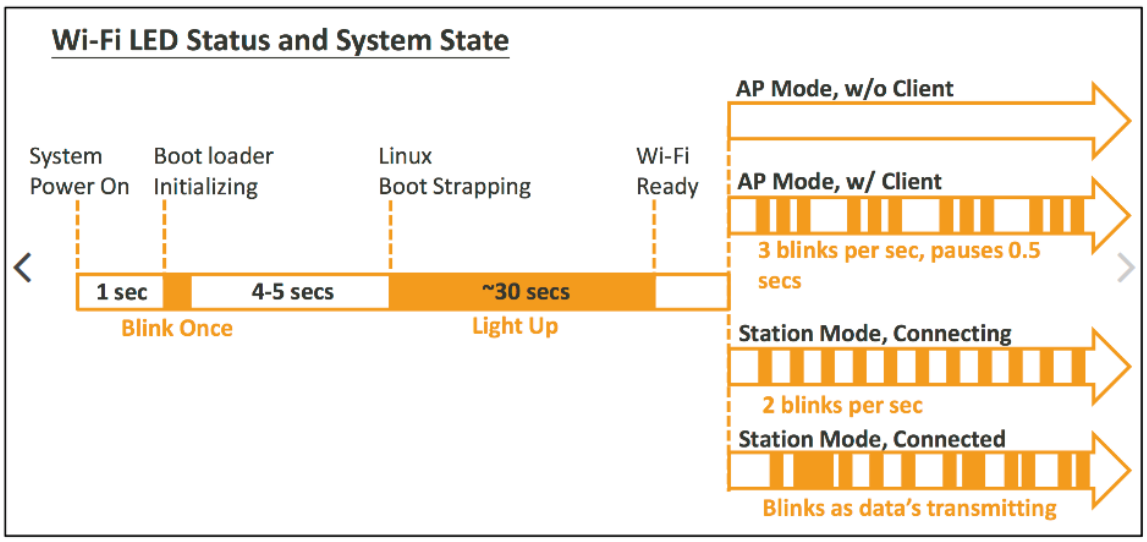
There are 2 Wi-Fi LED states in AP mode:

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.



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 Reset 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 using a USB drive.
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 using a USB drive.

6 Network

6.1 Reset Wi-Fi Configuration

You can reset Wi-Fi configuration back to AP mode by press and hold Wi-Fi 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 commit

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

Type the command in the system console:

wifi_mode sta

Notice: if you use SSH to connect to the system console of AI7688H, you'll lose the connection once the Wi-Fi command is set. 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

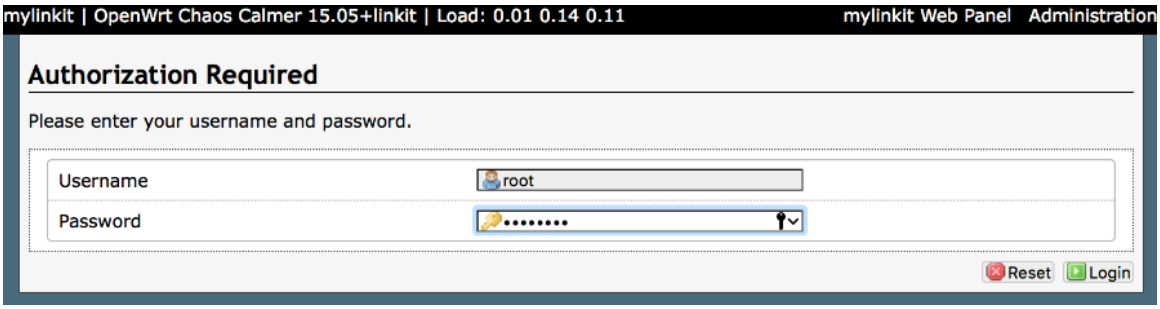
wifi_mode ap
uci set wireless.sta.disabled=1
uci commit

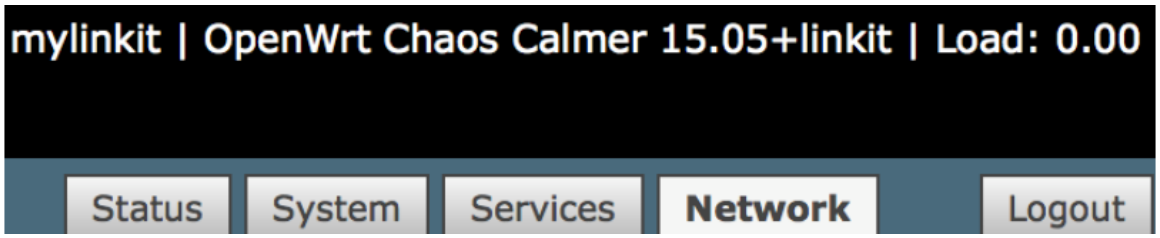
Step 2: Restart the Wi-Fi driver

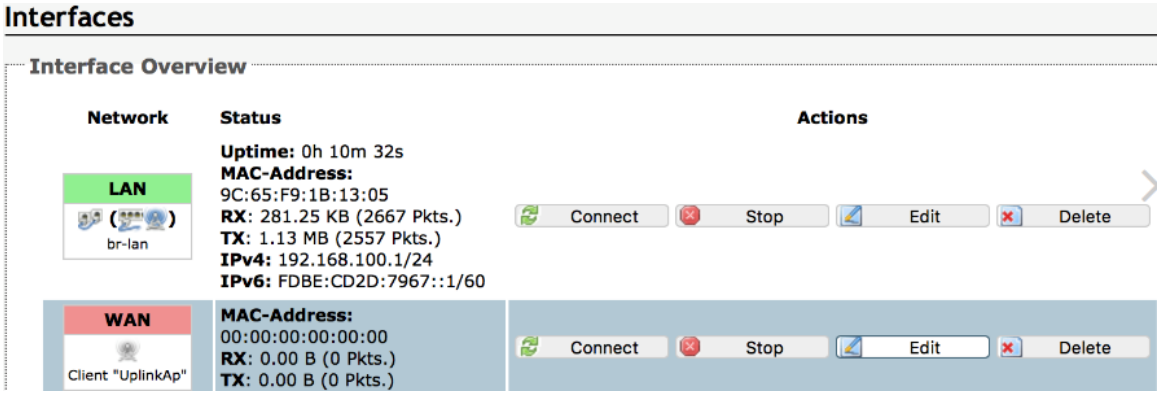
wifi

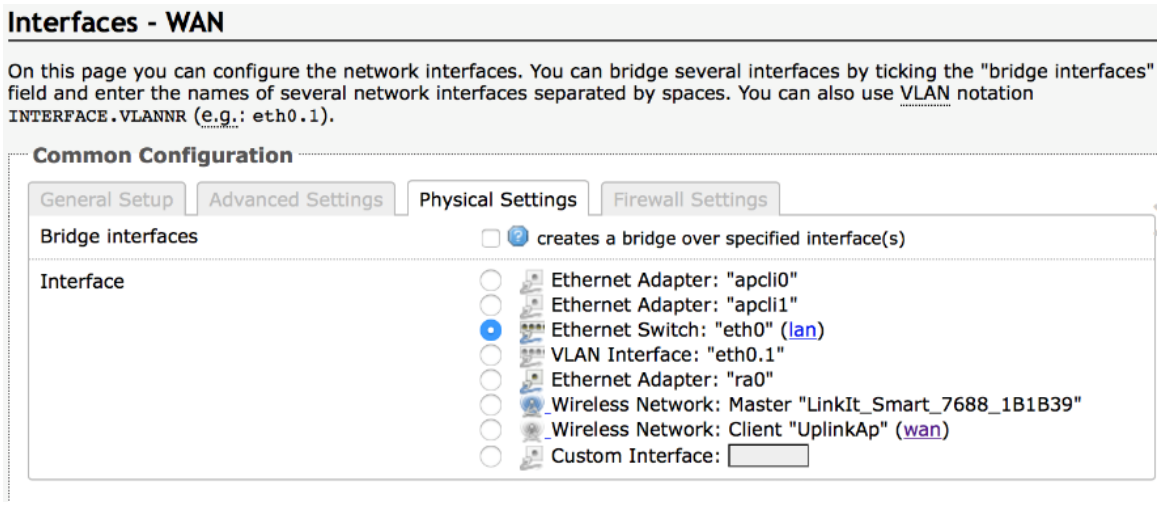
After the Wi-Fi driver restarts, the PC should be able to scan the AI7688H AP as shown in the video of the Reset Wi-Fi Configuration section.
Also the Wi-Fi status can be checked through the LED behavior.

6.4 Setup Wireless Router









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

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 Wi-Fi button

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

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

7 Copying Files To the Board

7.1 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:

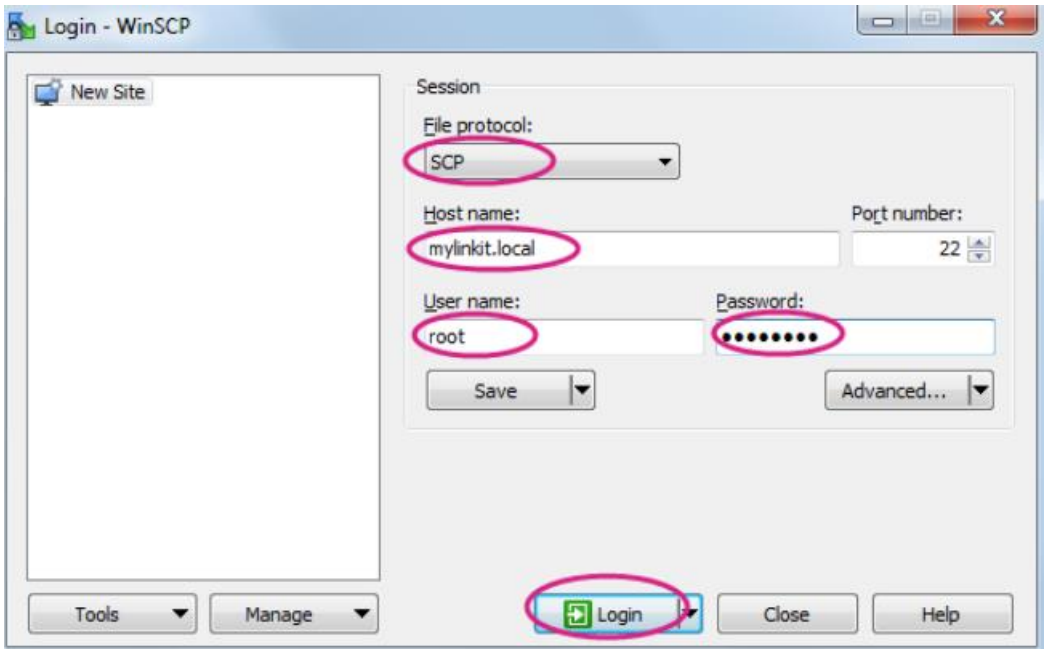
winscp.com –hostkey="*" scp://root@myLinkt.local helloworld.py

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.

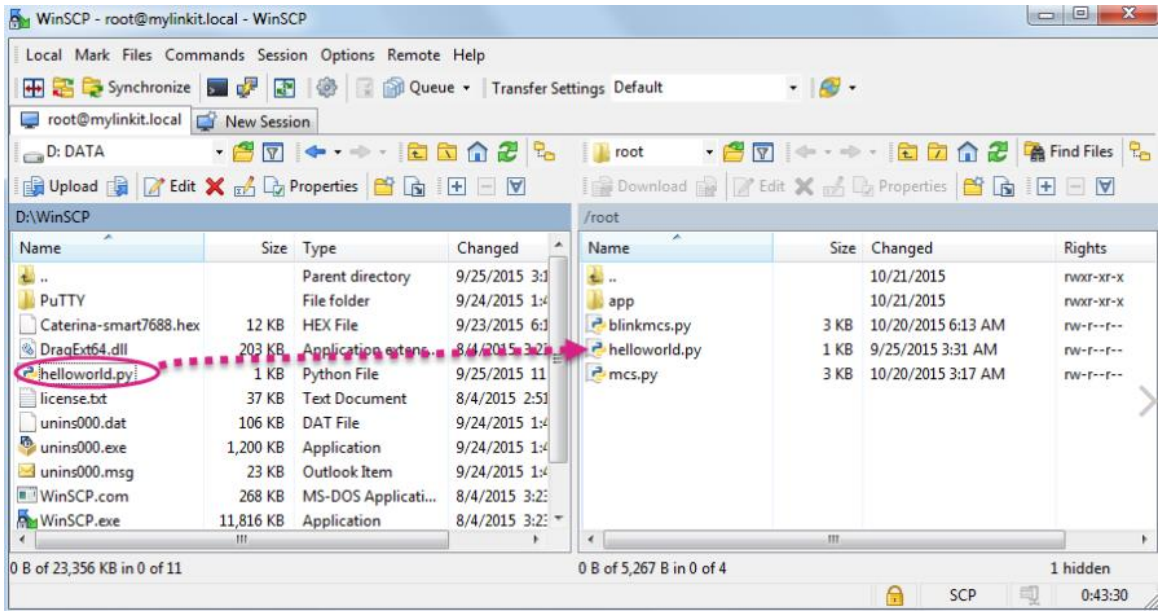
To use GUI

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:

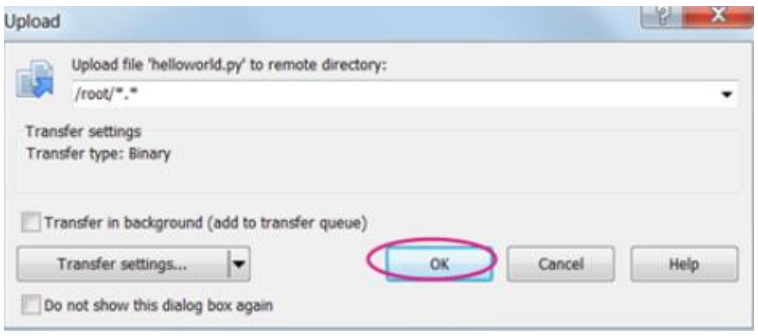


Click **Yes** when you see a warning window about *Continue connecting to an unknown server and add its host key to a cache*.

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



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



7.2 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
```

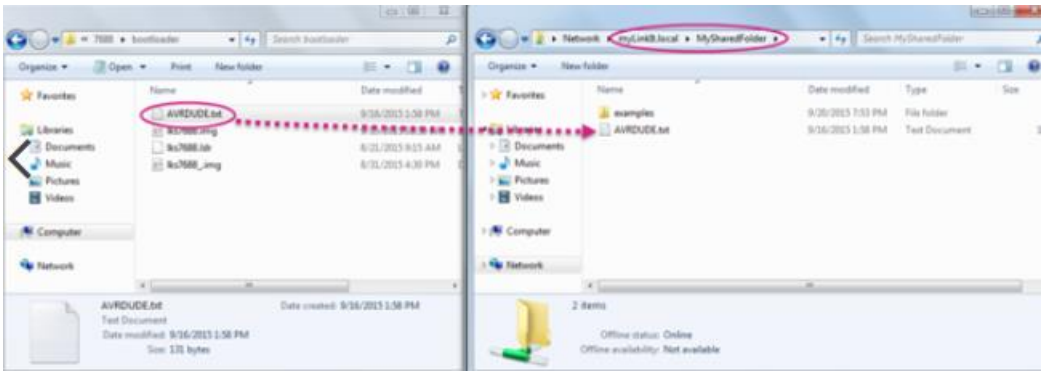
Save the UCI settings and reboot the AI7688H platform:

```
# 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.

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:



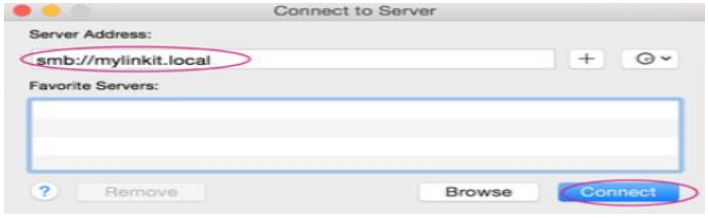
Access Samba in OS X

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

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

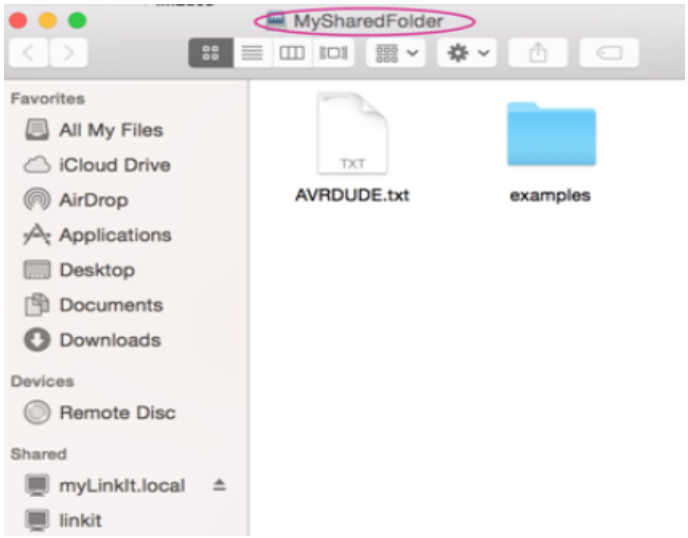


In the server address field, type smb://mylinkit.local or smb://192.168.100.1 and click **Connect**



Connect as guest:

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



8 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
```

#Get Statistic Data

root@myLinkIt:/# iwpriv ra0 stat

```
root@mylinkit:/# iwpriv ra0 stat
ra0       stat:
CurrentTemperature              = 49
Tx success                      = 16
Tx fail count                   = 1, PER=5.8%
Rx success                      = 735
Rx ICV Error                    = 0
Rx with CRC                     = 674, PER=47.8%
Rx with PhyErr                  = 0
Rx with PlcpErr                 = 0
Rx drop due to out of resource  = 0
Rx duplicate frame              = 0
False CCA                       = 0
BaSnResetCnt         = 0
RSSI                            = -49 0 0
TX AGG Range 1 (1)              = 13
TX AGG Range 2 (2~5)            = 0
TX AGG Range 3 (6~15)           = 0
TX AGG Range 4 (>15)            = 0
AMPDU Tx success                = 13
AMPDU Tx fail count             = 0, PER=0.0%
WPS Information:
Enrollee PinCode(ra0)           21580949

Enrollee PinCode(ApCli0)        00609425
Ap Client WPS Profile Count     = 0
```

9 AWS IoT

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

It provides 2 different SDKs and compatible 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.

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:

```
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 "**subscribe_publish_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"
// ================================================
```

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 certification files 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
```

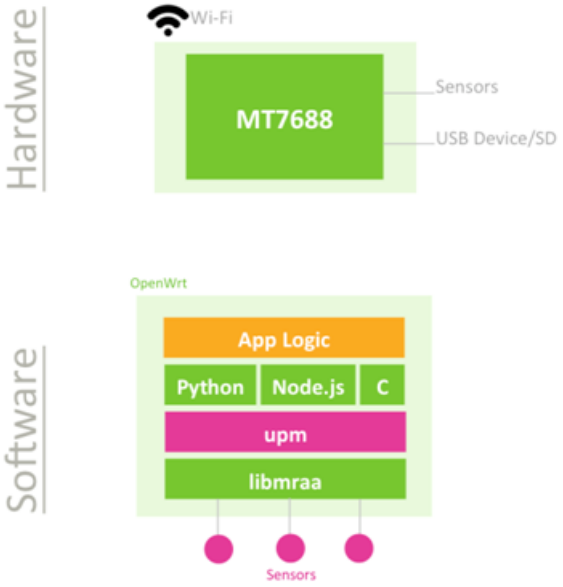
You can press ctrl-c to stop the execution.

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.

10 Peripheral

10.1 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.

Switch the PinMux

Please refer to the "02_ProductSpec" Page 11. You can see each pin takes multiple functions. For example, the EPHY_LED pin can act as an Ethernet LED or a GPIO.

Module_pinout	Pin NAME	DIGITAL Pin	SERIAL Pin	Other
1	GND			
2	JTAG_TMS			
3	JTAG_TDO	GPIO#43	EPHY LED	
4	JTAG_RST_N			
5	UART_TXD1	GPIO#45	UART_TXD1	
6	UART_RXD1	GPIO#46	UART_RXD1	

In this section, you'll learn how to use the mt7688_pinmux command to assign a different functionality to the pin.

Synopsis

mt7688_pinmux get

mt7688_pinmux set <group> <function>

Description

In the system console of the AI7688H development board, enter mt7688_pinmux get to query the current pinmux status:

```
root@mylinkit:/# mt7688_pinmux get
Group i2c - [i2c] gpio
Group uart0 - [uart] gpio
Group uart1 - [uart] gpio
Group uart2 - [uart] gpio pwm
Group pwm0 - [pwm] gpio
Group pwm1 - [pwm] gpio
Group refclk - refclk [gpio]
Group spi_s - spi_s [gpio]
Group spi_cs1 - [spi_cs1] gpio refclk
Group i2s - i2s [gpio] pcm
Group ephy - [ephy] gpio
Group wled - [wled] gpio
```

Let's consider the Group ephy mapping to the EPHY_LED0_N pin as an example. This group supports both ephy and GPIO pins. In this example, it's set to ephy with [ ].

To change the group functionality, use mt7688_pinmux set command. The following example changes the function of the Group ephy from ephy to gpio:

root@mylinkit:/# mt7688_pinmux set ephy gpio

set pinmux ephy -> gpio

You can also use the get parameter to verify the current pinmux status:

root@mylinkit:/# mt7688_pinmux get
Group i2c - [i2c] gpio
Group uart0 - [uart] gpio
Group uart1 - [uart] gpio
Group uart2 - [uart] gpio pwm
Group pwm0 - [pwm] gpio
Group pwm1 - [pwm] gpio
Group refclk - refclk [gpio]
Group spi_s - spi_s [gpio]
Group spi_cs1 - [spi_cs1] gpio refclk
Group i2s - i2s [gpio] pcm
Group ephy - ephy [gpio]
Group wled - [wled] gpio

As you can see the Group ephy functionality changed from ephy to gpio.

10.2 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 AI7688H's GPIO#2 (IS2_WS) pin in data sheet.

10.3 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 Interrupts

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.

```
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 GPIO#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 GPIO#2 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
```

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"
```

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)
```

10.4Using 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. You can connect the pins from the accelerometer to the corresponding pins GND, 3V3, SDA and SCL 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.Adxl1345(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
```

10.5Using 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 Wi-Fi LED (orange) should now start to blink every second.

10.6Controlling PWM with MRAA on AI7688H

Configure PWM pin:

To use PWM, initialize it on a certain pin such as GPIO. Note that only **GPIO#18**, **GPIO#19**, **GPIO#20**, **GPIO#21** 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 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
```

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);
```

11 C/C++ Programming

11.1 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
```

- 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
```

12 Using webcam and audio

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.

12.1 Supported cameras

AI7688H has installed Linux UVC (USB Video Class) drivers to provide USB webcam support. 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

Use *mjpg-streamer* for streaming application. 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" &
```

Note: Image resolution is 640x480, 25 frame per second, output to http stream on port 8080.

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.

Step 4: open a browser and check the video

connect to http://mylinkit.local:8080

12.2 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.

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
```