【國產 IC 開發套件】

型號：HUB 8735

(RTL8735)

AT User Guide

Getting Started with HUB 8735

指導單位：經濟部工業局

主辦單位：財團法人資訊工業策進會

執行單位：物聯網智造基地

合作單位：振邦科技股份有限公司

目錄

圖目錄

一、 Get Started

(一) What is HUB 8735-AT

HUB 8735-AT is based on HUB 8735 develop board. It makes HUB 8735 board as slave, and MCU is a host. The host can sends AT commands to HUB 8735 board and received AT response back. HUB 8735 provides a wide range of AT commands with different function.

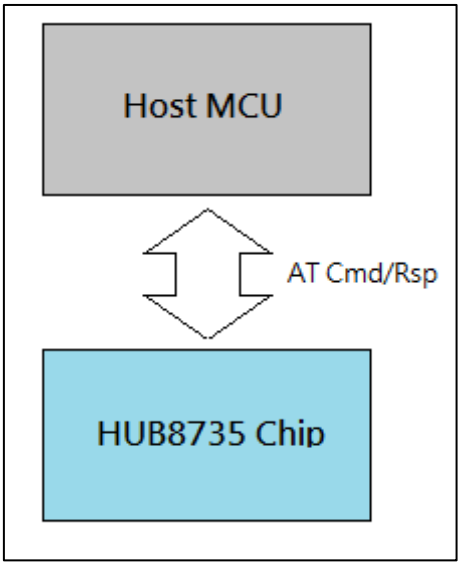

圖 1、HUB 8735-AT 運作示意圖

資料來源：本計畫整理

AT commands start with "AT", and end with a new line (CR LF). Every commands will return OK or ERR means the command result. Please be noted that all commands are executed serially, which means only one AT command can be executed at a time.

(二) Download Firmware by UART

1. Introduction Image Tool

This chapter introduces how to use Image Tool to download image firmware.

If the user has developed the HUB 8735 in the Arduino system and download the firmware through the Arduino project, the UART must be used to upgrade the HUB 8735-AT firmware.

The image tool - PGTool can be found in Tools folder and it has two functions:

(1) Download image firmware to a HUB 8735 device through UART.

(2) Generate composited image from multiple image files. (Not necessary)

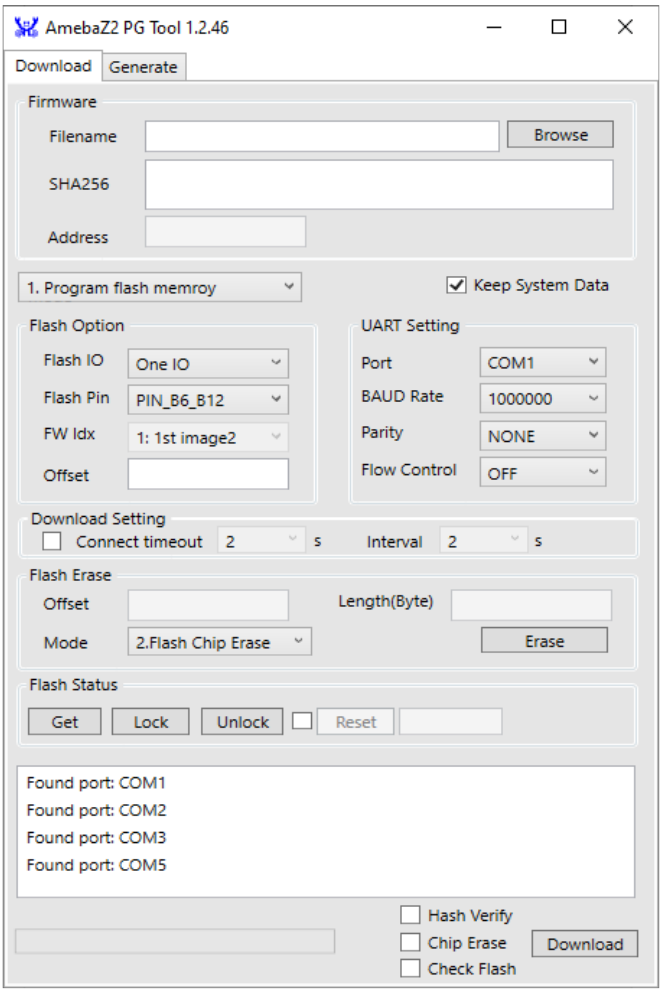

圖 2、PGTool

資料來源：本計畫整理

2. Download Environment Setup – Hardware

To download a firmware image, the device must be booted in download mode. User needs to set UART download mode first by connecting pin 3.3V and pin A5, and then connect U1R/U1T to TX/RX of PC console tool. Then press the reset button to enter the download mode.

3. Download Environment Setup – Software

- PC environment requirements: Windows 7 above with FT232 driver.
- PGTool

4. Image Download

User can download the image to EVB board by following steps:

(1) Boot HUB 8735 into download mode

You can check whether your board is in download mode by UART message:

```
== Rtl8735b IoT Platform ==
Chip VID: 0, Ver: 0
ROM Version: v3.0
Test Mode: boot_cfg1=0x0

[test mode PG]
test_mode_img_download
Download Image over UART1[tx=4,rx=3] baud=115200
```

CAUTION

Remember to disconnect log UART console before downloading image

(2) Run image tool

(3) Find your image: hub8735.bin from "Browse" button.

(4) Choose "1. Program flash memory" and select the correct COM port

(5) Disable "Keep System Data" to prevent some errors

(6) Press the Download button and the image will start to be downloaded to HUB 8735 device

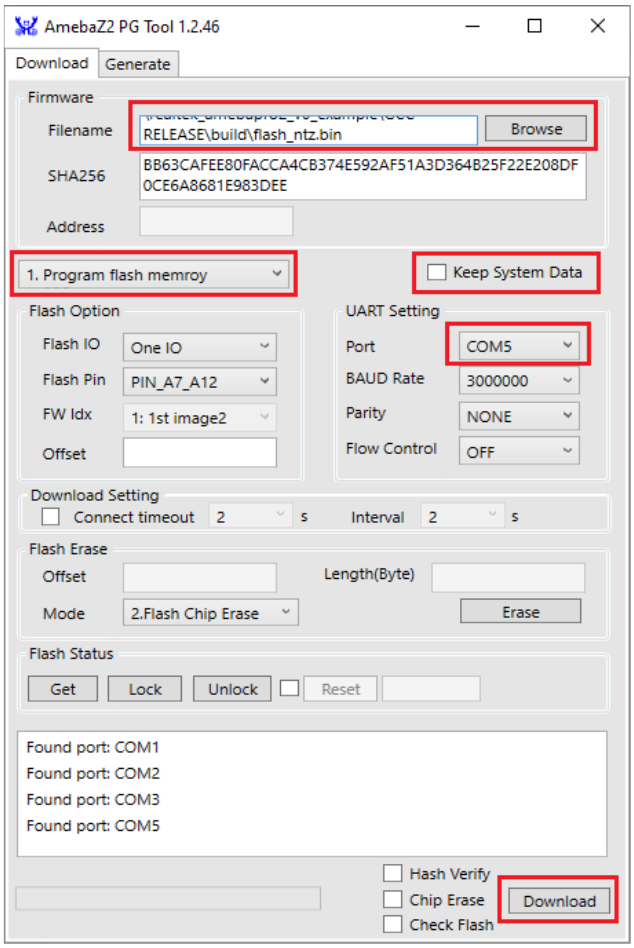

圖 3、Download Image

資料來源：本計畫整理

(三) Download Firmware by SD

1. Introduction

This chapter introduces how to use SD card to download images. If the user has developed the HUB 8735 in the Arduino system and download the firmware through the Arduino project, the SD download function must be failed.

2. Download Environment Setup

Use SD card download function, the device should not enter the download mode. Users only need to put firmware image to SD card and put into SD slot. Then press reset button to reset device.

3. SD Image Download

User can download the SD image to EVB board by following steps:

(1) Put hub8735.bin file into SD card. (Transcend SD card is recommended)

(2) Put SD card into SD slot in HUB 8735 device.

(3) Press reset button if device has power or power on the device.

You can check whether your board for SD download by UART message:

```
sd_upgrade enter
[sd_upgrade] hub8735.bin size 6729728
[sd_upgrade] Erasing... 0%
...
[sd_upgrade] Erasing... 100%
...
[sd_upgrade] Upgrading.... 100%

[sd_upgrade] Upgrade done
 [sd_upgrade] Delete file & Hold system, please reset board
```

(4) When upgrade done, system will automatically delete the download file.

(SDK Version "v94b-1108-1" not delete download file automatically.)*

CAUTION

If power supply is unstable in downloading stage, user need use UART upgrade method to re-download.

二、 AT Command Set

(一) Basic AT Commands

- AT: Test AT Startup.
- AT+RST: Restart a module.
- AT+GMR: Check version information.
- AT+CMD: List all commands.
- AT+GSLP: Enter deep sleep mode.
- ATE: Configure AT commands echoing.
- AT+SYSTIMESTAMP: Query/Set local time stamp.

1. AT: Test AT Startup

Execute command

Command:

```
AT
```

Response:

```
OK
```

2. AT+RESET: Restart a Module

Execute command

Command:

```
AT+RST
```

Response:

```
OK
```

3. AT+GMR: Check version information

Execute command

Command:

```
AT+GMR
```

Response:

```
<AT version info>

<SDK version info>

<Compiler time>

<Bin version>



OK
```

Parameter:

- <AT version info>:
- <SDK version info>:

- <Compiler time>:
- <Bin version>:

Example

```
AT+GMR

AT version:1.0.1122.0

SDK version:v94b-1122

Compile time:2022-11-21 10:37:33 [UTC]

Bin version:2022-11-21



OK
```

4. AT+CMD: List all AT commands and types supported

Query command

Command:

```
AT+CMD?
```

Response:

```
+CMD:<index>,<AT command name>,<support test

command>,<support query command>,<support set

command>,<support execute command>
```

```
OK
```

Parameters

- <index>: AT command sequence number.
- <AT command name>: AT command name.
- <support test command>: 0 means not supported, 1 means supported.
- <support query command>: 0 means not supported, 1 means supported.
- <support set command>: 0 means not supported, 1 means supported.
- <support execute command>: 0 means not supported, 1 means supported.

5. AT+GSLP: Enter Deep-sleep Mode

Set command

Command:

```
AT+GSLP=<time>
```

Response:

```
<time>
```

```
OK
```

Parameters

- <time>: the duration when the device stays in Deep-sleep. Unit: millisecond

6. ATE: Configure AT Commands Echoing

Execute command

Command:

```
ATE0
```

or

```
ATE1
```

Response:

```
OK
```

Parameters

- ATE0: Switch echo off
- ATE1: Switch echo on

7. AT+SYSTIMESTAMP: Query/Set Local Time Stamp

Query command

Function:

Query the time stamp

Command:

```
AT+SYSTIMESTAMP?
```

Response:

```
+SYSTIMESTAMP=<Unix_timestamp>

<ctime string>



OK
```

Set command

Command:

```
AT+SYSTIMESTAMP=<Unix_timestamp>
```

Response:

```
OK
```

Parameters

- < **Unix_timestamp** >: Unix timestamp. Unit:second.

- **<ctime string>**: Transfer Unix timestamp to string by using ctime function.

Example

```
AT+SYSTIMESTAMP=1565853509 //2019-08-15 15:18:29
```

(二) Wi-Fi AT Commands

- AT+CWMODE: Set the Wi-Fi mode (Station/SoftAP/Station+SoftAP).
- AT+CWSTATE: Query the Wi-Fi State and Wi-Fi infomation.
- AT+CWJAP: Connect to an AP.
- AT+CWRECONNCFG: Query/Set the Wi-Fi reconnecting configuration.
- AT+CWLAP: List available APs.
- AT+CWQAP: Disconnect from am AP.
- AT+CWSAP: Query/Set the configuration of SoftAP.
- AT+CWLIF: Obtain IP address of the station that connects to a SoftAP.
- AT+CWDHCP: Enable/Disable DHCP.
- AT+CIPSTAMAC: Query/Set the MAC address of a station.
- AT+CIPAPMAC: Query/Set the MAC address of a SoftAP.

- AT+CIPSTA: Query/Set the IP address of a station.
- AT+CIPAP: Query/Set the IP address of a SoftAP.

1. AT+CWMODE

Query command

Function:

Query the Wi-Fi mode.

Command:

```
AT+CWMODE?
```

Response:

```
+CWMODE:<mode>

OK
```

Set command

Function:

Set the Wi-Fi mode

Command:

```
AT+CWMODE=<mode> [,<auto_connect>]
```

Response:

```
OK
```

Parameters

- < **mode**>:
  - 0: Null mode. Wi-Fi RF will be disabled.
  - 1: Station mode.
  - 2: SoftAP mode.
  - 3. SoftAP + Station mode.
- <**auto_connect**>: Enable or disable automatic connection to AP.
  - 0: Disable automatically connect to an AP.
  - 1: Enable automatically connect to an AP.

Example

```
AT+CWMODE=3
```

2. AT+CWSTATE

Query command

Function:

Query the Wi-Fi state and Wi-Fi information of HUB 8735.

.

Command:

```
AT+CWSTATE?
```

Response:

```
+CWSTATE:<state>,<" ssid" >


OK
```

Parameters

- < **state**>:current Wi-Fi state
  - 0: HUB 8735 station has no started any Wi-Fi connection.
  - 1: HUB 8735 station has connected to an AP, but does not get an IP.
  - 2: HUB 8735 station has connected to an AP, and got an IP.
- <" **ssid**" >: the SSID of the target AP.

3. AT+CWJAP

Query command

Function:

Query the AP to which the HUB 8735 Station is already connected.

Command:

```
AT+CWJAP?
```

Response:

```
+CWJAP: <ssid>,<bssid>,<channel>,<rssi>

OK
```

Set command

Function:

Set the Wi-Fi mode

Command:

```
AT+CWJAP=<ssid> [,<pwd>]
```

Response:

```
WIFI CONNECT

WIFI GOT IP



OK
```

Execute command

Function:

Connect an HUB 8735 station to a targeted AP with last Wi-Fi

configuration.

Command:

```
AT+CWJAP
```

Response:

```
WIFI CONNECT

WIFI GOT IP



OK
```

Or

```
+CWJAP:<error code>

ERROR
```

Parameters

- < **ssid**>: the SSID of the target AP.
- <**pwd** >: password, MAX" 63-byte ASCII.
- <**bssid**>: the MAC address of the target AP.
- <**channel**>: channel
- <**rssi**>: signal strength.

- **<error code>**: (for reference only)
  - -2: bad argument.

Example

```
// If the target AP' s SSID is "abcde" and password is

 "0123456789" , the command should be

AT+CWJAP=abcde,0123456789
```

4. AT+CWRECONNCFG

Query command

Function:

Query the Wi-Fi configuration of Wi-Fi reconnect.

Command:

```
AT+CWRECONNCFG?
```

Response:

```
+CWRECONNCFG: <interval_second>,<repeat_count>

OK
```

Set command

Function:

Set the configuration of Wi-Fi reconnect

Command:

```
AT+ CWRECONNCFG =<interval_second>,<repeat_count>
```

Response:

```
OK
```

Parameters

- < **interval_second**>: the interval between Wi-Fi reconnections. Uint: Second.
  - **0**: The HUB 8735 station will not reconnect to the AP.
  - **1**: The HUB 8735 station will reconnect to the AP at the specified interval when disconnected
- <**repeat_count** >: the number of attempts the HUB 8735 makes to reconnect to the AP.

Example

```
// The HUB 8735 station tries to reconnect to AP at the interval of one second for 100 times
AT+CWRECONNCFG=1,100
```

```
// The HUB 8735 station will not reconnect to AP when disconnected

AT+CWRECONNCFG=0,0
```

5. AT+CWLAP

Execute command

Function:

List all available APs

Command:

```
AT+ CWLAP
```

Response:

```
+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<channel>
```

Parameters

- < **ecn**>: encryption method.
  - **0**:OPEN.
  - **1**:WEP
  - **2**:WPA_PSK
  - **3**:WPA2_PSK
  - **4**:WPA_WPA2_PSK

- **5**:WPA2_ENTERPRISE
- **6**:WPA3_PSK
- **7**:WPA_WPA2_ENTERRPRISE
- **8**:WAPI_PSK
- **9**:WPA_TKIP_PSK
- **10**:WPA2_TKIP_PSK
- **11**:WPA2_MIXED_PSK

- **<ssid>**:string parameter showing SSID of the AP.
- **<rssi>:**signal strength
- **<mac>:** string parameter showing MAC address of the AP.
- **<channel>:** channel

6. AT+CWQAP: Disconnect from an AP

Execute command

Command:

AT+ CWQAP

Response:

OK

7. AT+CWSAP

Set command

Function:

Set the configuration of a HUB 8735 SoftAP.

Command:

```
AT+ CWSAP =<ssid>,<pwd>,<channel>,<ecn>[,<max conn>,<ssid hidden>]
```

Response:

```
OK
```

Parameters

- **< ssid>**: string parameter showing SSID of the AP.
- **<pwd>**: string parameter showing the password. Length:8~63 bytes ASCII.
- **<channel>:** channel ID.
- **<ecn>**:encryption method; WEP is not support
  - **0**:OPEN.
  - **2**:WPA_PSK
  - **3**:WPA2_PSK

- **4**:WPA_WPA2_PSK

- **[<max conn>]:** maximum number of stations that HUB 8735 SoftAP.

- **[<ssid hidden>]:**
  - **0**: broadcasting SSID(default).
  - **1**: not broadcasting SSID.

Example

```
AT+CWSAP=HUB8735AP,1234567890,5,3
```

8. AT+CWLIF: Obtain IP Address of the Station

Execute command

Command:

```
AT+ CWLIF
```

Response:

```
+CWLIF:<ip addr>,<mac>


OK
```

Parameters

- < **ip addr**>: IP address of the station.
- <**mac**>:MAC address of the station.

Note

- This command cannot get a static IP, It works only when DHCP of both the HUB 8735 SoftAP and the connected station are enabled.

9. AT+CWDHCP: Enable/Disable DHCP

Query command

Command:

```
AT+CWDHCP?
```

Response:

```
+CWDHCP:<state>

OK
```

Set command

Function:

Enable/disable DHCP.

Command:

```
AT+CWDHCP=<operate>, <mode>
```

Response:

```
OK
```

Parameters

- **< operate>**:
  - 0: disable.
  - 1: enable.
- **<mode>**:
  - Bit0: Station DHCP.
  - Bit1: SoftAP DHCP
- **<state>**: the status of DHCP
  - Bit0:

    * 0: Station DHCP is disabled.

    * 1: Stat ion DHCP is enabled.
  - Bit1: SoftAP DHCP

    * 0: SoftAP DHCP is disabled.

    * 1: SoftAP ion DHCP is enabled.

Example

```
//Enable Station DHCP

AT+CWDHCP=1,1

//Disable SoftAP DHCP

AT+CWDHCP=0,2
```

10. AT+CIPSTAMAC: Query/Set the MAC Address

Query command

Function:

Query the MAC address of the HUB 8735 Station.

Command:

```
AT+CIPSTAMAC?
```

Response:

```
+CIPSTAMAC:<mac>

OK
```

Set command

Function:

Set the MAC address of the HUB 8735 Station.

Command:

```
AT+ CIPSTAMAC =<mac>
```

Response:

```
OK
```

Parameters

- < **mac**>:string parameter showing MAC address of a HUB 8735 Station.

Note

- This mac will also be change by setting HUB 8735 SoftAP.

Example

```
AT+CIPSTAMAC=11:22:33:44:55:66
```

11. AT+CIPAPMAC: Query/Set the MAC Address

Query command

Function:

Query the MAC address of the HUB 8735 SoftAP.

Command:

```
AT+CIPAPMAC?
```

Response:

```
+CIPAPMAC:<mac>

OK
```

Set command

Function:

Set the MAC address of the HUB 8735 SoftAP.

Command:

```
AT+ CIPAPMAC =<mac>
```

Response:

```
OK
```

Parameters

- **< mac>**:string parameter showing MAC address of a HUB 8735 SoftAP.

Note

- This mac will also be change by setting HUB 8735 Station.

Example

```
AT+CIPAPMAC=11:22:33:44:55:66
```

12. AT+CIPSTA: Query/Set the IP Address

Query command

Function:

Query the IP address of the HUB 8735 Station.

Command:

```
AT+CIPSTA?
```

Response:

```
+CIPSTA:ip:<mac>

+CIPSTA:gateway:<gateway>

+CIPSTA:netmask:<netmask>



OK
```

Set command

Function:

Set the IPv4 address of the HUB 8735 station.

Command:

```
AT+ CIPSTA =<ip>[,<gateway>,<netmask>]
```

Response:

```
OK
```

Parameters

- < **ip**>:string parameter showing IPv4 address of a HUB 8735 Station
- <**gateway**>: gateway.
- <**netmask**>: netmask.

Note

- This IP will also be change by setting HUB 8735 SoftAP.

Example

```
AT+CIPSTA=192.168.2.100,192.168.2.1,255.255.255.0
```

13. AT+CIPAP: Query/Set the IP Address

Query command

Function:

Query the IP address of the HUB 8735 SoftAP.

Command:

```
AT+CIPAP?
```

Response:

```
+CIPAP:ip:<mac>

+CIPAP:gateway:<gateway>

+CIPAP:netmask:<netmask>



OK
```

Set command

Function:

Set the IPv4 address of the HUB 8735 SoftAP.

Command:

```
AT+ CIPAP =<ip>[,<gateway>,<netmask>]
```

Response:

```
OK
```

Parameters

- **< ip>**:string parameter showing IPv4 address of a HUB 8735 Station.
- **<gateway>**: gateway.
- **<netmask>**: netmask.

Note

- This IP will also be change by setting HUB 8735 Station.

Example

```
AT+CIPAP=192.168.2.100,192.168.2.1,255.255.255.0
```

(三) TCP/IP AT Commands

- AT+PING: Ping the remote host

1. AT+PING: Ping the remote host.

Set command

Function:

Ping the remote host.

Command:

```
AT+PING=<host>
```

Response:

```
+PING:<time>


OK
```

or

```
+PING:TIMEOUT


ERROR
```

Parameters

- < **host**>:string parameter showing IPv4 address or domain name.
- <**time**>: the response time of ping. Unit: millisecond.

Example

```
AT+PING=192.168.1.1

AT+PING=www.google.com
```

(四) Driver AT Commands

- AT+DRVADC: Read ADC channel value.
- AT+DRVPWMINIT: Initialize PWM driver.
- AT+DRVPWMDUTY: Set PWM duty.
- AT+DRVI2CINIT: Initialize I2c master driver.
- AT+DRVI2CRD: Read I2C data.

- AT+DRVI2CWRBYTES: Write I2C data.
- AT+DRVGPIO: Configure GPIO pin and read/write GPIO pin.

1. AT+DRVADC: Read ADC channel value

Set command

Command:

```
AT+DRVADC=<channel>,<atten>
```

Response:

```
+DRVADC:<raw data>

OK
```

Parameters

- < **channel**>:ADC1 channel.

  For HUB 8735 devices, the range is [0,6].

CHANNEL	GPIO
0	GPIOF_0

1	GPIOF_1
2	GPIOF_2
3	GPIOA_0
4	GPIOA_1
5	GPIOA_2
6	GPIOA_3

- **<atten>**: Reserved.
- **<raw data>**: ADC channel value

Example

```
//The return 4095 means the voltage is 4095/4095 * 3300 = 3300 mV

AT+DRVADC=0,0

+DRVADC:4095



OK
```

2. AT+DRVPWMINIT: Initialize PWM driver.

Set command

Command:

```
AT+DRVPWMINIT=<freq>,<duty_res>,<ch0_gpio>[,...,<ch3_gpio>]
```

Response:

```
OK
```

Parameters

- < **freq**>: timer frequency. Unit: Hz.
- < **duty_res** >: duty resolution. Range: 0~20 bits.
- <**chx_gpio**>: output of channel x. For example, if you want to use GPIOF_6 as channel 0, set <ch0_gpio> to 0.

chx_gpio	GPIO
0	GPIOF_6
1	GPIOF_7
2	GPIOF_8
3	GPIOF_10

Example

```
// set 4 channels; frequency: 5 kHz; duty resolution: 13 bits

AT+DRVPWMINIT=5000,13,0,1,2,3

// only use channel 0, frequency: 10 kHz; duty resolution: 10 bits; other

PMW //commands can only set one channel

AT+DRVPWMINIT=10000,10,0
```

3. AT+DRVPWMDUTY: Set PWM duty.

Set command

Command:

```
AT+DRVPWMDUTY=<ch0_duty>[,...,<ch3_duty>]
```

Response:

```
OK
```

Parameters

- **< chx_duty>**: channel x duty. Range: [0,2^{duty_res}].

Example

```
// set channel 0 to duty 255, set channel 1 to duty 512

AT+DRVPWMDUTY=255,512
```

```
// set channel 2 to duty 0

AT+DRVPWMDUTY=,,0
```

4. AT+DRVI2CINIT: Initialize I2c master driver

Set command

Command:

```
AT+DRVI2CINIT=<num>,<scl_io>,<sda_io>,<clock>
```

Response:

```
OK
```

Parameters

- **< num >**: I2C port number. Range: 0~1.

num	SCL	SDA
0	GPIOA_0	GPIOA_1
1	GPIOF_1	GPIOF_2

- **<scl_io >**: GPIO pin for I2C SCL signal. Reserved in HUB 8735.
- **<sda_io>**: GPIO pin for I2C SDA signal. Reserved in HUB 8735.
- **<clock>**: I2C clock frequency for master mode. Unit: Hz. Maximum:

1MHz.

Example

```
// Initialize I2C1; SCL reserved; SDA reserved; I2C clock is 100 kHz

AT+DRVI2CINIT=1,0,0,100000

// Deinitialize I2C0

AT+DRVI2CINIT=0 // deinitialize I2C0

// Deinitialize I2C1

AT+DRVI2CINIT=1 // deinitialize I2C1
```

5. AT+DRVI2CRD: Read I2C Data

Set command

Command:

```
AT+DRVI2CRD=<num>,<address>,<length>
```

Response:

```
+DRVI2CRD:<read data>

OK
```

Parameters

- **< num >**: I2C port number. Range: 0~1.

num	SCL	SDA
0	GPIOA_0	GPIOA_1
1	GPIOF_1	GPIOF_2

- **<address>**: I2C slave device address
  - 7-bit address: 0 ~ 0x7F.
- **<length>**: I2C data length.
- **<read data>**: I2C data.

Example

```
//I2C1 reads address 0x6B, register address 0x0F, read 1 bytes

AT+DRVI2CWRBYTES=1,0x6B,1,0x0F

AT+DRVI2CRD=1,0x6B,1
```

6. AT+DRVI2CWRBYTES: Write I2C Data

Set command

Command:

```
AT+DRVI2CWRBYTES=<num>,<address>,<length>,<data>
```

Response:

```
OK
```

Parameters

- < **num** >: I2C port number. Range: 0~1.

num	SCL	SDA
0	GPIOA_0	GPIOA_1
1	GPIOF_1	GPIOF_2

- <**address**>: I2C slave device address
  - 7-bit address: 0 ~ 0x7F.
- <**length**>: the length of I2C data you want to write. Range:1~3 byte;
- <**data**>: the data of <**length**> long. Range: 0~0xFFFFFF.

Example

```
// I2C1 writes address 0x6B; register address: 0x0D; data: 0xFF

AT+DRVI2CWRBYTES=1,0x6B,2,0x0DFF

// I2C1 writes address 0x6B; register address: 0x0D; data: 0xFFFF

AT+DRVI2CWRBYTES=1,0x6B,3,0x0DFFFF
```

7. AT+DRVGPIO: Configure GPIO pin

Set command

Command:

```
AT+DRVGPIO=<gpio_ch>,<gpio_dir>,<gpio_mode/set_value>
```

Response:

```
OK
```

Or

```
+PINSTATE:<state>

OK
```

Parameters

- < **gpio_ch** >: GPIO channel. Range: 0~14.

gpio_ch	GPIO
0	GPIOA_0
1	GPIOA_1
2	GPIOA_2
3	GPIOA_3
4	GPIOA_5
5	GPIOE_1

6	GPIOE_2
7	GPIOF_0
8	GPIOF_1
9	GPIOF_2
10	GPIOF_5
11	GPIOF_6
12	GPIOF_7
13	GPIOF_8
14	GPIOF_10

- <**gpio_dir**>: gpio direction
  - 0: input.
  - 1: output.
- <**gpio mode**>: this parameter is only used by input
  - 0: None pull.
  - 1: Pull up.
  - 2: Pull down.
  - 3. Open drain
- <**set value**>: this parameter is only used by output.
  - 0: Output Low.

- 1: Output High.

Example

```
//setting GPIOA_5 for input pull up.

AT+DRVGPIO=4,0,1

//setting GPIOA_5 for output high

AT+DRVGPIO=4,1,1
```

(五) User AT Commands

- AT+USERAM: Get user’s free RAM.
- AT+REC: record video to SD card.
- AT+SUVC: start USB video class.
- AT+SMSC: start USB Mass storage.
- AT+AISTART: start AI function.
- AT+SETTONE: initialize tone parameter.
- AT+PLAYTONE: play/ pause tone.

1. AT+USERRAM: Get free RAM

Query command

Command:

```
AT+USERRAM?
```

Response:

```
+USERRAM:< size >

OK
```

Parameters

- < **size** >: free ram size.

2. AT+REC: record video to SD card

Query command

Command:

```
AT+REC?
```

Response:

```
+REC: <file name>,<recording time>,<recording

resolution>,<recording fps > , <recording file number>

OK
```

Set command

Command:

```
AT+REC=<start/stop recording>,[<filename>,<recording
```

```
time>,<recording resolution>,<recording fps>,<recording file
number>]
```

Response:

```
OK
```

Or

```
+RECEND
```

+RECEND: Recording stop.

Parameters

- < **start/stop recording** >: Start or Stop recording.
  - 0: stop recording.
  - 1: start recording.
- <**file name**>: Recording file name: default: AmebaPro_recording
  - 0: input.
  - 1: output.
- <**recording time**>: Recording time. Unit: seconds.
- <**recording resolution**>: Recording resolution.
  - 0: 1920 x 1080.
  - 1: 1280 x 720
  - 2: 640 x 480.

- **<recording fps>**: Recording frame rate.
- **<recording file number>**: Recording file number. Set 0 is means loop setting.

Example

```
//get record information

AT+REC?

//name: AmebaPro_recording, time 30 seconds, resolution 1920x1080,

30 fps, recording 1 file

+REC:AmebaPro_recording,30,0,30,1

//fast record.

AT+REC=1

//recording file name: test, 10 seconds, 640x480, 30fps, continues 5 files.

AT+REC=1,test,10,2,30,5
```

3. AT+SUVC

Execute command

Function:

Start USB video class.

Command:

AT+SUVC

Response:

OK

Note

The USB video function needs to correctly connect the DP/DN pin to PC or use HUB 8735 I/O board.

4. AT+SMSC

Execute command

Function:

Start USB mess storage.

Command:

AT+SMSC

Response:

OK

Note

The USB mess storage function needs to correctly connect the DP/DN pin to PC or use HUB 8735 I/O board.

5. AT+AISTART

Execute command

Function:

Start AI function. The AT command about AI function will enable when

AT+AISTART OK.

Command:

```
AT+AISTART
```

Response:

```
+AISTART:OK
```

6. AT+ SETTONE

Query command

Command:

```
AT+SETTONE?
```

Response:

```
+SETTONE: <audio_tone_rate>,<audio_tone_db>

OK
```

If user needs to change the tone setting, please paused tone first.

Set command

Command:

```
AT+SETTONE =<audio_tone_rate>,<audio_tone_db>
```

Response:

```
OK
```

Parameters

- < **audio_tone_rate** >: audio tone rate. Unit: Hz. Default :1KHz.Range: [0~23999]
- **<audio_tone_db>**: audio tone volume. Default: 0 dB. Maximum: 0 dB

Example

```
AT+SETTONE=2000,2    //set tone rate 2K, tone gain -2 dB.

AT+SETTONE=20000,5    //set tone rate 20K, tone gain -5 dB.
```

7. AT+PLAYTONE

Set command

Command:

```
AT+PLAYTONE =<start/pause>
```

Response:

```
OK
```

Parameters

- < **start/pause** >: start/ pause the tone voice.

Example

```
AT+PLAYTONE=1     //start to play tone

AT+PLAYTONE=0     //pause to play tone
```

(六) AICAM AT Commands

AICAM AT commands takes effect after "AT+AISTART" command is completed.

- AT+DETECTNUM: Set yolov4 AI detects maximum number.
- AT+DETECTTYPE: Query yolov4 AI detects types.
- AT+DETECTDELAY: Set yolov4 detect interval.
- AT+DETECTFILTER: Set Detect filter in yolov4.
- AT+DETECTINTR: Set Interrupt response type.
- AT+NEWFACE: Add new face id.
- AT+LISTFACE: List the specified face id name or all face id.
- AT+DELFACE: Delete the specified face id or all face id.

- AT+RENAMEFACE: Rename the specified face id name.
- AT+FACEINTR: Interrupt response when face detect.
- AT+SOUNDINTR: Enable interrupt response when sound detect.
- AT+AIPAUSE: Pause AI model.
- AT+AIRESUME: Resume AI model.

1. AT+DETECTNUM

Set command

Function:

Set yolov4 AI detects maximum number.

Command:

AT+DETECTNUM= <detect item max number>

Response:

OK

Parameters

- **< detect item max number >**: Maximum detect number.

2. AT+DETECTTYPE

Query command

Function:

Query yolov4 AI detects types.

Command:

```
AT+ DETECTTYPE?
```

Response:

```
<index>: < object name>
```

Parameters

- **< index >**: show the index type of yolov4 detect.
- **< object name**>: show the type name of yolov4 detect

3. AT+DETECTDELAY

Set command

Command:

```
AT+DETECTDELAY= <AI type>,<detect item interval>
```

Response:

```
OK
```

Parameters

- **< AI types >**:
  - 0:yolo4t

- 1:yamnet
- 2:retina face

- < **Delay item interval** >: Delay item interval. Range 0~1000, Unit: million seconds. Default: 50.

4. AT+ DETECTFILTER

Set command

Function:

Only detect those filter item by user setting.

Command:

```
AT+DETECTFILTER=<filter type id>,[<filter type id2>,...,<filter type id10>]
```

Or

```
AT+DETECTFILTER=ALL
```

Filter all type.

Response:

```
OK
```

Parameters

- **<filter type idx>**: Filter type id x. Type id can use

"AT+DETECTTYPE?" to get.

5. AT+DETECTINTR

Set command

Function:

Set AT command interrupt when AI detection the object.

Command:

```
AT+DETECTINTR=<detect type id>,[<detect type id2>,...,<detect type id10>]
```

Or

```
AT+DETECTINTR=ALL
```

Interrupt all type.

Response:

```
OK
```

Interrupt response

```
+DETECTINTR=<detect type name>:<posi_x>,<posi_y>,<pois_x_len>,<pos_y_len>
```

Parameters

- **< detect type idx>**: detect type id x. Type id can use

"AT+DETECTTYPE?" to get

- < **detect type name**>: detect type name.
- < **posi_x**>: detect object position start in horizon.
- < **posi_y**>: detect object position start in vertical
- < **posi_x_len**>: detect object position end in horizon.
- < **posi_y_len**>: detect object position end in vertical.

6. AT+NEWFACE: Add new face id

Execute command

Command:

```
AT+NEWFACE
```

Response:

```
+NEWFACE:DETECTING
```

This command will detect 10 seconds

Or

```
+NEWFACE:<id>,<id name>
```

If command success, return id index and name.

Or

```
+NEWFACE:<err>
```

If command failed, return error number.

Parameters

- **< err>**:
  - -1: 10 seconds timeout.
  - -2: the face already exists.

7. AT+LISTFACE

Set command

Command:

```
AT+LISTFACE=<face id>
```

Response:

```
+FACE:<face id>,<face name>

OK
```

Or

```
+FACE:<err>
```

Parameters

- **<face id>**: face id.
- **<face name>**: face name.
- **< err>**:

- -1: not found face name.

Execute command

Command:

```
AT+LISTFACE
```

Response:

```
+FACE:<face id0>,<face name>

+FACE:<face id1>,<face name>

OK
```

Or

```
+FACE:<err>
```

Parameters

- **<face id x>**: face id x.
- **<face name>**: face id x name.
- **< err>**:
  - -1: not found face name.

8. AT+DELFACE

Set command

Function:

Delete the special face id.

Command:

```
AT+DELFACE=<face id>
```

Response:

```
OK
```

Execute command

Function:

Delete all face lists.

Command:

```
AT+DELFACE
```

Response:

```
OK
```

9. AT+ RENAMEFACE

Set command

Command:

```
AT+RENAMEFACE=<face id>,<face name>
```

Response:

```
OK
```

Parameters

- **<face id x>**: face id.
- **<face name>**: face id name.
-

10. AT+ FACEINTR

Set command

Function:

Enable interrupt log when face detect.

Command:

```
AT+FACEINTR=<enable/disable>[,<score>][,<count>]
```

Response:

```
OK
```

Interrupt response

```
+FACEINTR=<face name>
```

Parameters

- **< enable/disable >**:
  - 0: enable interrupt response log.
  - 1: disable interrupt response log.
- **< score >**: set the interrupt condition "face recognition

accuracy" . Default: 0.70

- < **count** >: set compare times to improve accuracy. Default: 3 times
- <**face name**>: The name of the face in the face list, if the detected face is not in the list, it should be "STRANGER" .

Example

```
//enable interrupt response, accuracy 90%, compare 5 times.

AT+FACEINTR=1,0.9,5
```

11. AT+SOUNDINTR

Set command

Function:

Enable interrupt log when sound detect.

Command:

```
AT+SOUNDINTR=<enable/disable>
```

Response:

```
OK
```

Interrupt response

```
+SOUNDINTR=<sound_type>,<prob>
```

Parameters

- < **enable/disable** >:
  - 0: enable sound interrupt response log.
  - 1: disable sound interrupt response log.
- < **sound_type** >: detect sound event type.
- < **prob** >: sound detect accuracy.

Example

```
//enable sound interrupt response.

AT+SOUNDINTR=1
```

12. AT+AIPAUSE

Set command

Function:

Pause the AT Demo type.

Command:

```
AT+AIPAUSE=< AI types>
```

Response:

```
OK
```

Parameters

- **< AI types>**:
  - 0: yolo4t.
  - 1: yamnet.
  - 2: retina.
  - 3. mfn.
  - 4.objtracking.

13. AT+AIRESUME

Set command

Function:

Resume the AT Demo type.

Command:

AT+AIRESUME=<AI types>

Response:

OK

Parameters

- **< AI types>**:
  - 0: yolov4.
  - 1: yamnet.

- 2: retina.
- 3. mfn.
- 4.objtracking.

(七) BLE AT Commands

- AT+BLEINIT: Bluetooth LE initialization.
- AT+BLEADDR: Query/Set Bluetooth LE device address.
- AT+BLENAME: Query/Set Bluetooth LE device name.
- AT+BLESCANPARAM: Query/Set parameters of Bluetooth LE scanning
- AT+BLESCAN: Enable Bluetooth LE scanning.
- AT+BLESCANRSPDATA: Set Bluetooth LE scan response.
- AT+BLEADVPARAM: Query/Set parameters of Bluetooth LE advertising.
- AT+BLEADVDATA: Set Bluetooth LE advertising data.
- AT+BLEADVSTART: Start Bluetooth LE advertising.
- AT+BLEADVSTOP: Stop Bluetooth LE advertising.

1. AT+BLEINIT

Query command

Function:

Check the initialization status of Bluetooth LE.

Command:

```
AT+BLEINIT?
```

Response:

If Bluetooth LE is initialized, AT will return:

```
+BLEINIT:<role>

OK
```

If Bluetooth LE is not initialized, AT will return:

```
+BLEINIT:0

OK
```

Set command

Function:

Initialize the role of Bluetooth LE.

Command:

```
AT+BLEINIT=<init>
```

Response:

```
OK
```

Parameters

- **< role>**:
  - 1: client role.
  - 2: server role.
- **<init>**:
  - 0: deinit Bluetooth LE.
  - 1: client role.
  - 2: server role.

Example

```
AT+BLEINIT=1
```

2. AT+BLEADDR

Query command

Function:

Query the Bluetooth LE Public Address.

Command:

```
AT+BLEADDR?
```

Response:

```
+BLEADDR:<BLE public address>

OK
```

Parameters

- < BLE public address >: BLE public address.

3. AT+BLENAME

Query command

Function:

Query the Bluetooth LE device name.

Command:

```
AT+BLENAME?
```

Response:

```
+BLENAME:<device_name>

OK
```

Set command

Function:

Initialize the role of Bluetooth LE.

Command:

```
AT+BLENAME=<device_name>
```

Response:

```
OK
```

Parameters

- < **device_name**>:the Bluetooth LE device name. The maximum length is 32. Default: "HUB8735_AT"

Example

```
AT+BLENAME=HUB8735_DEMO
```

Note

After setting the device name with this command, it is recommended that you execute the AT+BLEADVDATA command to add the device name into the advertising data.

4. AT+BLESCANPARAM

Query command

Function:

Query the parameters of Bluetooth LE scanning.

Command:

```
AT+BLESCANPARAM?
```

Response:

```
+BLESCANPARAM:

<scan_type>,<own_addr_type>,<filter_policy>,<scan_interval>,

<scan_window>

OK
```

Set command

Function:

Set the parameters of advertising.

Command:

```
AT+BLESCANPARAM=<scan_type>,<own_addr_type>,<filter_policy>,<

scan_interval>,

<scan_window>
```

Response:

```
OK
```

Parameters

- **< scan_type >**
  - 0: passive scan

- 1: active scan.

- < **own_addr_type** >:
  - 0: Public address
  - 1: Random address.
- < **filter_policy**>:
  - 0: BLE_SCAN_FILTER_ALLOW_ALL
  - 1: BLE_SCAN_FILTER_ALLOW_ONLY_WLST.
  - 2. BLE_SCAN_FILTER_ALLOW_UND_RPA_DIR
  - 3. BLE_SCAN_FILTER_ALLOW_WLIST_PRA_DIR
- < **scan_interval**>: scan interval.
- < **scan_window** >: scan window..

Example

```
AT+BLEINIT=2 // Role: server

AT+BLEADVPARAM=50,50,0,0,4,0,1
```

5. AT+BLESCAN

Set command

Function:

Enable/ disable scanning.

Command:

```
AT+BLESCAN=<enable>[,<interval>]
```

Response:

```
+BLESCAN:<addr>,<rssi>,<adv_data>,<scan_rsp_data>,<addr_type>

OK
```

Parameters

- **<enable>**: scan response data is a HEX string.
  - 0: ADV_TYPE_IND.
  - 1: ADV_TYPE_DIRECT_IND_HIGH.
- **<Interval>**: optional parameter. Unit: second.
- **<addr>**: Bluetooth LE address.
- **<rssi>**: signal strength.
- **<adv_data>: advertising data.**
- <scan_rsp_data>: scan response data.
- <addr_type>: the address type of broadcasters.

Example

```
AT+BLEINIT=1

AT+BLESCAN=1

AT+BLESCAN=0
```

```
AT+BLESCAN=1,5        //start scanning for 5 second
```

6. AT+BLESCANRSPDATA

Set command

Function:

Set scan response.

Command:

```
AT+BLESCANRSPDATA=<scan_rsp_data>
```

Response:

```
OK
```

Parameters

- **< scan_rsp_data>**: scan response data is a HEX string.

Example

```
AT+BLEINIT=2

AT+BLEADVDATA=1122334455
```

7. AT+ BLEADVPARAM

Query command

Function:

Query the parameters of advertising.

Command:

```
AT+BLEADVPARAM?
```

Response:

```
+BLEADVPARAM:

<adv_int_min>,<adv_int_max>,<adv_type>,<own_addr_type>,

<channel_map>,<adv_filter_policy>,<peer_adv_type>

OK
```

Set command

Function:

Set the parameters of advertising.

Command:

```
AT+BLEADVPARAM

=<adv_int_min>,<adv_int_max>,<adv_type>,<own_addr_type>,

<channel_map>,<adv_filter_policy>,<peer_adv_type>
```

Response:

```
OK
```

Parameters

- < **adv_int_min** >:minimum advertising interval.
- < **adv_int_max** >:maximum advertising interval.
- < **adv_type**>:
  - 0: ADV_TYPE_IND.
  - 1: ADV_TYPE_DIRECT_IND_HIGH.
  - 2. ADV_TYPE_SCAN_IND
  - 3. ADV_TYPE_NONCONN_IND
  - 4. ADV_TYPE_DIRECT_IND_LOW
- < **own_addr_type**>: own Bluetooth LE address type
  - 0: BLE_ADDR_TYPE_PUBLIC
  - 1: BLE_ADDR_TYPE_RANDOM
- < **channel_map** >: channel of advertising.
  - 1:ADV_CHNL_37
  - 2:ADV_CHNL_38
  - 4:ADV_CHNL_39
  - 7:ADV_CHNL_ALL
- < **adv_filter_policy** >:
  - 0: ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY
  - 1: ADV_FILTER_ALLOW_SCAN_WLST_CON_ANY

- 2: ADV_FILTER_ALLOW_SCAN_ANY_CON_WLST
- 3: ADV_FILTER_ALLOW_SCAN_WLST_CON_WLST

- < **peer_adv_type** >:
  - 0: PUBLIC
  - 1: RANDOM

Example

```
AT+BLEINIT=2 // Role: server

AT+BLEADVPARAM=50,50,0,0,4,0,1
```

8. AT+BLEADVDATA

Set command

Function:

Set advertising data.

Command:

```
AT+BLEADVDATA=<adv data>
```

Response:

```
OK
```

Parameters

- < **adv data**>: advertising data in HEX string.

Example

```
AT+BLEINIT=2

AT+BLEADVDATA=1122334455
```

9. AT+BLEADVSTART

Execute command

Function:

Start advertising.

Command:

```
AT+BLEADVSTART
```

Response:

```
OK
```

Example

```
AT+BLEINIT=2      //Role Server

AT+BLEADVSTART
```

10. AT+BLEADVSTOP

Execute command

Function:

Stop advertising.

Command:

```
AT+BLEADVSTOP
```

Response:

```
OK
```

Example

```
AT+BLEINIT=2        //Role Server

AT+BLEADVSTART

AT+BLEADVSTOP
```

三、 How to use AI Module

(一) Hardware Requirement

- HUB 8735 develop board
- Transcend micro SD card (4G 以上)
- PC UART tool.
- Dupont Line.

(二) Hardware Environment

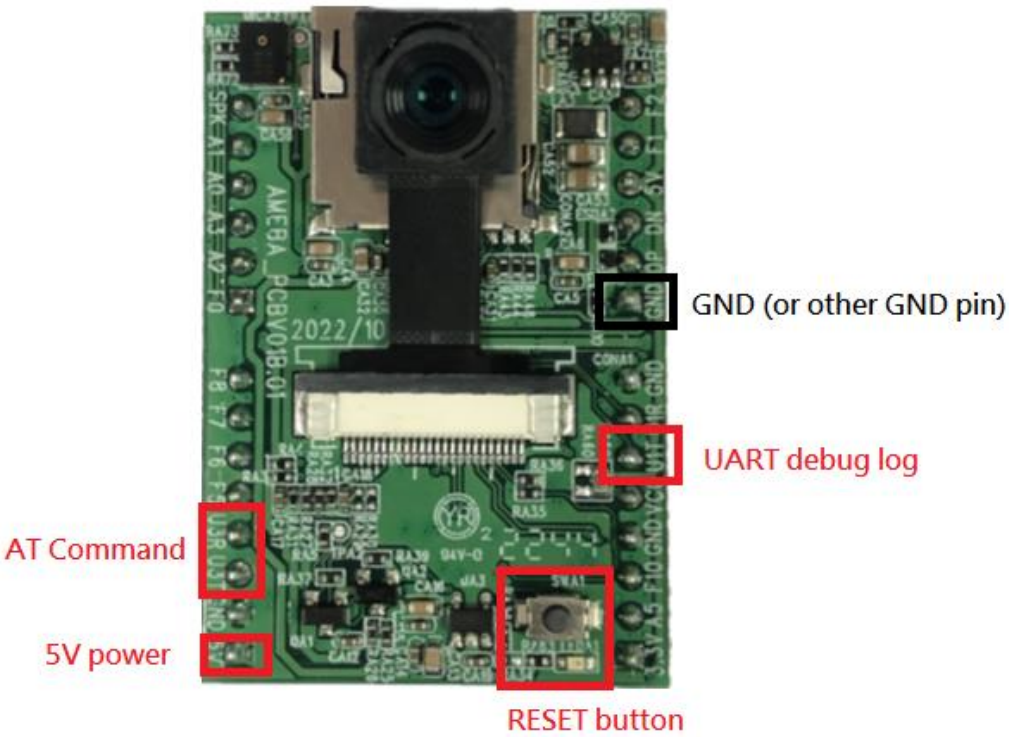

圖 4、Hardware Environment

資料來源：本計畫整理

- 5V power：HUB 8735 Main power。
- GND：GND。

- UART debug log：Log output，must use this to confirm the upgrading status。
- RESET button：Reset HUB 8735。
- AT Command：AT command input/output。

(三) SDK Version check

User need check SDK version and Compiler time and Bin Version by AT command “AT+GMR”.

```
AT version:0.0.0.0
SDK version:v94b-1108-1
Compile time:2022-11-07 11:16:34 [UTC]
Bin version:2022-11-07
```

圖 5、SDK version “v94b-1108-1”
資料來源：本計畫整理

SDK version “v94b-1108-1” only support AI types from SD card.

```
AT version:0.0.0.0
SDK version:v94b-1108
Compile time:2022-11-07 11:16:34 [UTC]
Bin version:2022-11-07
```

圖 6、SDK version “v94b-1108”
資料來源：本計畫整理

SDK version “v94b-1108” support facial recognition, face tracking and other AI types from SD card.

(四) AI Demo file in SD card

Users need to put these files into the SD card to fully enable all the functions of AI DEMO.

HUB 8735 AI DEMO file list:

htdocs.bin	2022/9/22 上午 1...	BIN 檔案	412 KB
HUB8735_AI_DEMO.json	2022/9/22 上午 1...	JSON 檔案	2 KB
yamnet_10_u8.nb	2022/9/22 上午 1...	NB 檔案	2,662 KB
yolov4_tiny_hybrid_SD9.nb	2022/9/22 上午 1...	NB 檔案	8,531 KB

圖 7、HUB 8735 AI DEMO file list

資料來源：本計畫整理

It is recommended to use a Transcend SD card.

Download file name: sdcard_for_HUB8735_AI_DEMO.rar

(五) How to use AI DEMO

Before SDK version "v94b-1108", AI DEMO will automatically start when HUB 8735 device power on. After this version user need use AT command "AT+ATSTART=DEMO" to Start AI DEMO function with AP mode and HTTP.

1. Connect Wi-Fi AP

User can find "HUB8735_AI_DEMO" in wireless manager. Connect it and enter password "12345678". Then wait for the connection.

圖 8、Connect Wi-Fi

資料來源：本計畫整理

2. Connect Ameba PRO2 AI webpage

Open browser, and enter "//192.168.1.1:80", user will see the Ameba PRO2 AI webpage.

3. AI DEMO function list

Object Detection (YOLO4-Tiny):

Object recognition and sorting of objects and displaying object types. 80 kinds of objects can be identified, and the supported types can be viewed from Category.

This function need put yolov4_tiny_hybrid_SD9.nb in SD card.

Facial Recognition:

Face recognition and list and display name, can add, edit, delete face

records. Face data will save in SD card.

Single Tracking:

Single face tracking.

Multi Tracking:

Multiple face tracking.

Sound Event:

Sound recognition. It can recognize 521 kinds of sound types, and you can see the supported types from Category.

This function need put yamnet_10_u8.nb in SD card.

四、 AT Command Examples

(一) Simple Access Control System

User can use HUB 8735 AT command to create a simple access control system. In this example, PIN "GPIOA_2" connect with the Door Controller.

1. Create User Face

(1) Start AI function.

Command:

```
AT+AISTART
```

Response:

```
+AISTART:OK
```

(2) Add new face

Command:

```
AT+NEWFACE
```

Response:

```
+NEWFACE:DETECTING...

+NEWFACE:0,Guest_0
```

(3) Change name

Command:

```
AT+RENAMEFACE=0,Peter
```

Response:

```
OK
```

2. Check/Change/Delete Face List

(1) Check current face list

Command:

```
AT+LISTFACE
```

Response:

```
+FACE:0,Peter

+FACE:1,David

+FACE:2,Nick
```

(2) Change name or Delete

Command:

```
AT+DELFACE=2
```

Response:

```
OK
```

3. Get Face detect response

(1) Start face detect interrupt response

Command:

```
AT+FACEINTR=1,0.95,5
```

Response:

```
OK
```

(2) Analyze response data and do corresponding processing.

Waiting response

```
+FACEINTR=Peter
```

4. Do Other Activities (ex: GPIOA_2 to control door)

(1) Initial GPIOA_2 for output low for close the door.

Command:

```
AT+DRVGPIO=2,1,0
```

Response:

```
OK
```

(2) When detect correct face name. Set GPIOA_2 output high to opening the door.

Command:

```
AT+DRVGPIO=2,1,1
```

Response:

```
OK
```

(二) Simple Bird Repellent

User can use HUB 8735 AT command to create a simple bird repellent. In this example, Pin "Audio_OUT" and pin "GND" connect with the Speaker.

1. Set repellent condition

(1) Start AI function.

Command:

```
AT+AISTART
```

Response:

```
+AISTART:OK
```

(2) Find the bird detect id

Command:

```
AT+DETECTTYPE?
```

Response:

```
....

13:bench

14:bird
```

```
15:cat

...
```

(3) Set Interrupt condition

Command:

```
AT+DETECTINTR=14
```

Response:

```
OK
```

(4) Set sound Interrupt condition [Not necessary]

Command:

```
AT+SOUNDINTR=1
```

Response:

```
OK
```

2. Get response

(1) Analyze response data and do corresponding processing.

Waiting response

```
+DETECTINTR=bird:77,72,245,218
```

Or

```
+SOUNDTYPE:Bird,0.88
```

```
+SOUNDTYPE:Animal,0.84
```

3. Set tone and playing

(1) When get response of bird detecting or Bird sound detecting. Setting 20K tone to repelling birds

Command:

```
AT+SETTONE=20000,0
```

Response:

```
OK
```

(2) Playing tone

Command:

```
AT+PLAYTONE=1
```

Response:

```
OK
```

(3) Stop tone.

Command:

```
AT+PLAYTONE=0
```

Response:

```
OK
```

五、 參考資料

- **物聯網智造基地 - 國產 IC 智造工具包**

https://www.ideas-hatch.com/mem_evb.jsp