ACM WI07C WIFI MODULE - LOW COST

Specification

- Module power 3.3V, regular current consumption at 70ma, peak current at 240mA (300mA must be able to provided)
- +20Dbm power, 100M max transmitting distance on ideal circumstance.
- It is common and correct to see some random error data when module is power up, and end up with "ready" (Turn baud rate to 115200 can see this actual debug data, this is used for firmware updating)

IC Features

- 802.11 b / g / n
- WIFI @ 2.4 GHz, supports WPA / WPA2 security mode
- Ultra-small size module 11.5mm * 11.5mm
- Built-in 10 bit precision ADC
- Built-in TCP / IP protocol stack
- Built-in TR switch, balun, LNA, power amplifier and matching network
- Built-in PLL, voltage regulator and power management components
- 802.11b mode + 19.5dBm output power
- Supports antenna diversity
- Off leakage current is less than 10uA
- Built-in low-power 32-bit CPU: can double as an application processor
- SDIO 2.0, SPI, UART
- STBC, 1x1 MIMO, 2x1 MIMO
- The guard interval A-MPDU, the polymerization of the A-MSDU and 0.4 s of
- Within 2ms of the wake, connect and transfer data packets
- Standby power consumption is less than 1.0mW (DTIM3)
- Operating temperature range -40 ~ 125 ℃

AT Commands

Format

- Baud rate at 57600, 115200 (new line) use option "send new line" or 'carriage return' for each command
- x is the commands

Set	Inquiry	Test	Execute
AT+<x>=<...>	AT+<x>?	AT+<x>=?	AT+<x>
AT+CWMODE=<mode>	AT+CWMODE?	AT+CWMODE=?	-
Set the network mode	Check current mode	Return which modes supported	-

Commands

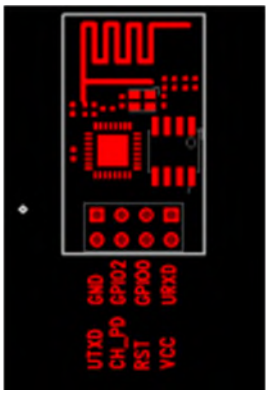
- carefully there are must be no any spaces between the " and IP address or port

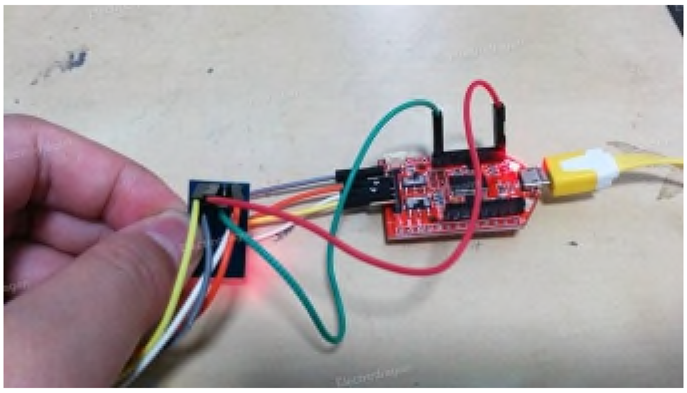
Commands	Description	Type	Set/Execute	Inquiry	test	Parameters and Examples
AT	general test	basic	-	-	-	-
AT+RST	restart the module	basic	-	-	-	-
AT+GMR	check firmware version	basic	-	-	-	-
AT+CWMODE	wifi mode	wifi	AT+CWMODE=<mode>	AT+CWMODE?	AT+CWMODE=?	1= Sta, 2= AP, 3=both, Sta is the default mode of router, AP is a normal mode for devices
AT+CWJAP	join the AP	wifi	AT+ CWJAP =<ssid>,< pwd >	AT+ CWJAP?	-	ssid = ssid, pwd = wifi password
AT+CWLAP	list the AP	wifi	AT+CWLAP			
AT+CWQAP	quit the AP	wifi	AT+CWQAP	-	AT+CWQAP=?	
AT+ CWSAP	set the parameters of AP	wifi	AT+ CWSAP= <ssid>,<pwd>,<chl>, <ecn>	AT+ CWSAP?		ssid, pwd, chl = channel, ecn = encryption; eg. Connect to your router: AT+CWJAP="www.electrodragon.com","helloworld"; and check if connected: AT+CWJAP?
AT+CWLIF	check join devices' IP	wifi	AT+CWLIF	-	-	
AT+ CIPSTATUS	get the connection status	TCP/IP	AT+ CIPSTATUS			<id>,<type>,<addr>,<port>,<tetype>= client or server mode
AT+CIPST	set up TCP or UDP	TCP/IP	1)single connection	-	AT+CIPSTAR	id = 0-4, type = TCP/UDP, addr = IP address, port= port; eg. Connect to

Commands	Description	Type	Set/Execute	Inquiry	test	Parameters and Examples
ART	connection		(+CIPMUX=0) AT+CIPSTART= <type>,<addr>,<port>; 2) multiple connection (+CIPMUX=1) AT+CIPSTART= <id><type>,<addr>, <port>		T=?	another TCP server, set multiple connection first: AT+CIPMUX=1; connect: AT+CIPSTART=4,"TCP","X1.X2.X3.X4",9999
AT+CIPMODE	set data transmission mode	TCP/IP	AT+CIPMODE=<mode>	AT+CIPSEND?		0 not data mode, 1 data mode; return "Link is builded"
AT+CIPSEND	send data	TCP/IP	1)single connection(+CIPMUX=0) AT+CIPSEND=<length>; 2) multiple connection (+CIPMUX=1) AT+CIPSEND= <id>,<length>		AT+CIPSEND=?	eg. send data: AT+CIPSEND=4,15 and then enter the data.
AT+CIPCLOSE	close TCP or UDP connection	TCP/IP	AT+CIPCLOSE=<id> or AT+CIPCLOSE		AT+CIPCLOSE=?	
AT+CIFSR	Get IP address	TCP/IP	AT+CIFSR		AT+ CIFSR=?	
AT+ CIPMUX	set mutiple connection	TCP/IP	AT+ CIPMUX=<mode>	AT+ CIPMUX?		0 for single connection 1 for multiple connection
AT+ CIPSERVER	set as server	TCP/IP	AT+ CIPSERVER= <mode>[,<port>]			mode 0 to close server mode, mode 1 to open; port = port; eg. turn on as a TCP server: AT+CIPSERVER=1,8888, check the self server IP address: AT+CIFSR=?

Commands	Description	Type	Set/Execute	Inquiry	test	Parameters and Examples
AT+ CIPSTO	Set the server timeout	AT+CIPSTO=	AT+CIPSTO?		<time>0~28800 in second	
+IPD	received data					For Single Connection mode(CIPMUX=0): + IPD, <len>: For Multi Connection mode(CIPMUX=1): + IPD, <id>, <len>: <data>

Pin Wiring (V090)





- Use FT232RL can supply enough power, connect 3V3, GND, TXD, RXD, (Swap these two pins if no data come up), also CH_PD to 3V3 (red), GPIO0 to GND (green, **ONLY** connect GPIO0 when update firmware)
- There are two LEDs on the board, one is power led (RED), another one is status LED (BLUE), when power up, pwr led keeps on and status led will blink once.
- baud rate may work at 9600 (**seems the latest correct one**), 115200 or 57600

Pin description

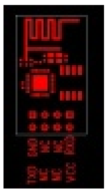
Pin	High Status	Low Status	Note
VCC, GND			Use standalone power source, or large capacitor, all power of this module

Pin	High Status	Low Status	Note
			from external
TXD, RXD			The serial port, swap these two pins if no data come up. this is very easily go wrong. (TX to RX, and RX to TX, not TX to TX and RX to RX)
RST	-	restart	
CH_PD	Flash boot and Update Mode	-	chip enable, so always connect to high status with VCC
GPIO0	-	Update mode	
GPIO2	-	-	
GPIO 15 (when avaialble)	NC	Flash boot Mode	Only for a few version, Wi07-3
GPIO 12			
GPIO 13			
GPIO 14			
GPIO 16	Hardware RST		

- No need any pull-up

Old version (V080)

The old version



Setup Verification

- check two LEDS status when boot up, if this is not working, double check your wiring first then continue, don't forget CHPD to VCC
- check if your devices (phone) can find a wifi spot named like "ESP_98529F" or similar, the later number part is the mac ID, if you can see this wifi spot, it means your module boot up successfully
- swap RX and TX pins if you cannot get AT commands response
- Tick "new line" option on SSCOM32 serial port monitor tool
- Try baud rate 9600 or 115200 normally should be these two, old version is 115200
-

"ESP_990B15" is the module wifi spot

See the final "ready" when boot up successfully

- Don't forget to connect GPIO15 to GND if you are using the SMD model

First time use guide

Using Arduino as serial port Montior

- Connect VCC and GND of module to 3v3 and GND of Arduino, RXD to TXD of Arduino, and TXD to RXD of Arduino (should add resistors or logic level shifter for logic level and protect IOs)
- Simply upload blink sketch to Arduino, to ensure MCU won't use serial port
- Use any other serial port monitor like SSCOM32 we used here, available here UART shown below :

UART

Introduction

Check here to see the definition of UART in Wikipedia.

Tools

- Terminal Terminal,
- SSCOM32 English [see - sscom32E (zip file)]
- SSCOM42 version 4.2 [see - Sscom42 (zip file)]

- Putty
- www.vandyke.com/products/securecrt/ SecurecRT

…………………………………………………………………………………

- In the serial port, you should see "ready" in the end of the random data after powered up.
- Send AT (commands, with "newline option") will receive OK in return.

Steps and note

- **AT+RST** restart the module, received some strange data, and "ready"
- **AT+CWMODe=3** change the working mode to 3, AP+STA, only use the most versatile mode 3 (AT+RST may be necessary when this is done.)

Join Router

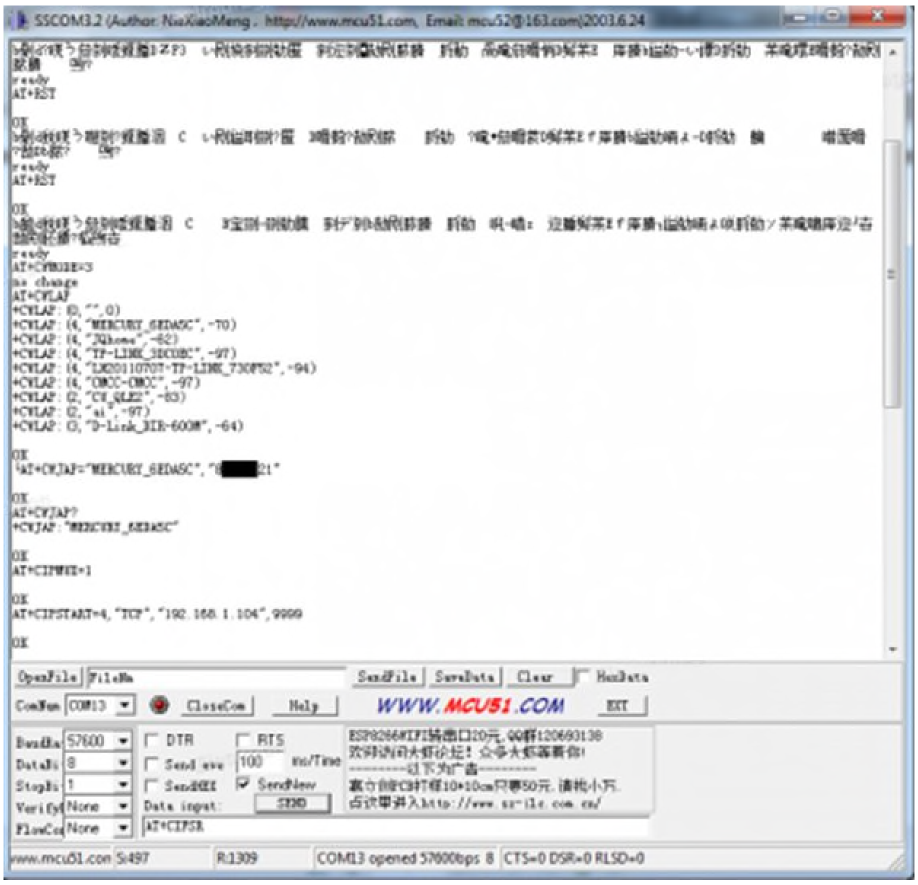
- **AT+CWLAP** search available wifi spot
- **AT+CWJAP="you ssid", "password"** join my mercury router spot (ops, the wifi password is here :))
- **AT+CWJAP=?** check if connected successfully, or use AT+CWJAP?

TCP Client

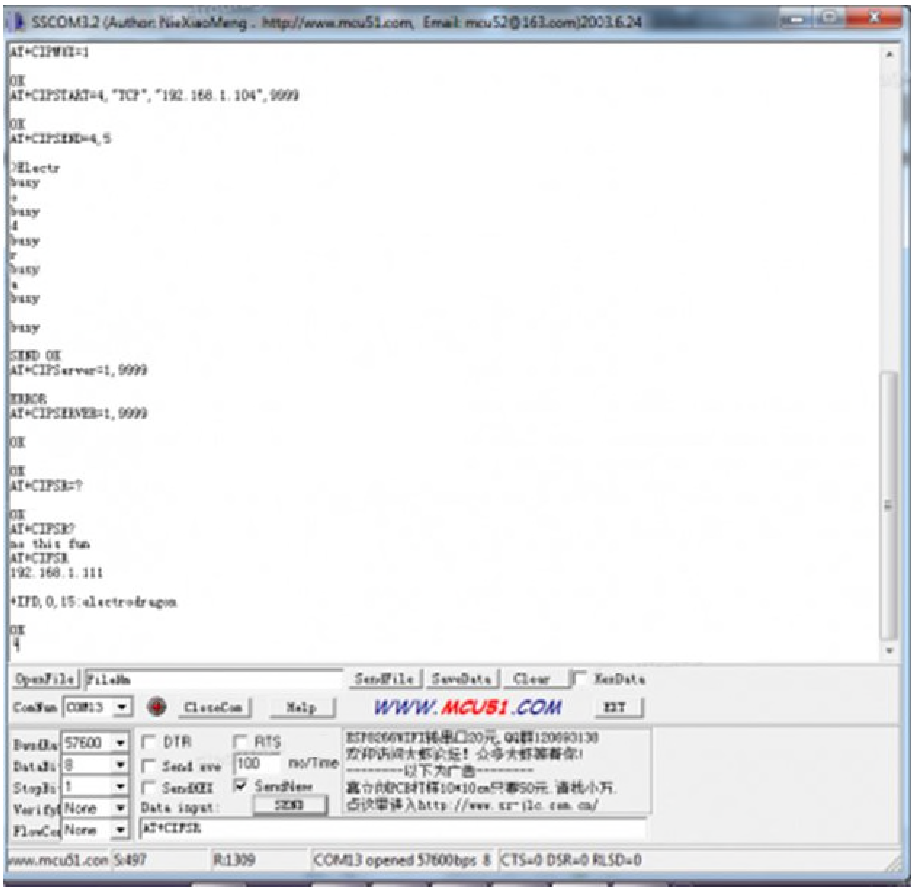
- **AT+CIPMUX=1** turn on multiple connection
- **AT+CIPSTART=4,"TCP","192,168.1.104",9999** connect to remote TCP server 192.168.1.104 (the PC)
- **AT+CIPMODE=1** optionally enter into data transmission mode
- **AT+CIPSEND=4,5** send data via channel 4, 5 bytes length (see socket test result below, only "elect" received), link will be "unlink" when no data go through

TCP Server

- **AT+CIPSERVER=1,9999** setup TCP server, on port 9999, 1 means enable
- **AT+CIFSR** check module IP address
- PC as a TCP client connect to module using socket test, send data



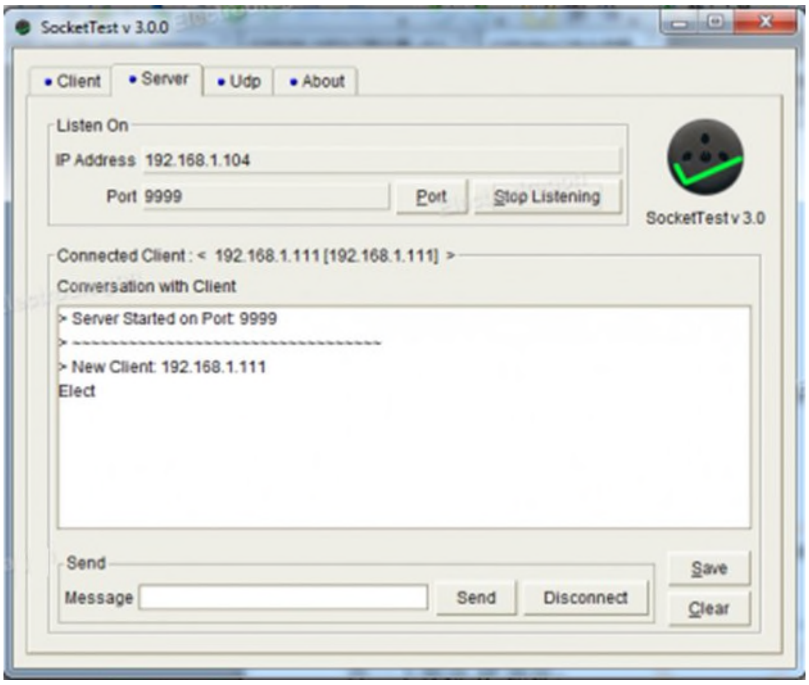
```
[illegible]
ready
AT+RST

OK
[illegible]
ready
AT+RST

OK
[illegible]
ready
AT+CWMODE=3
no change
AT+CWLAP
+CWLAP:(0,"",0)
+CWLAP:(4,"MERCURY_6EDA5C",-70)
+CWLAP:(4,"JQhome",-82)
+CWLAP:(4,"TP-LINK_3DCUBC",-97)
+CWLAP:(4,"LK20110707-TP-LINK_730F52",-94)
+CWLAP:(4,"CMCC-CMCC",-97)
+CWLAP:(2,"CY_QLEZ",-83)
+CWLAP:(2,"ai",-97)
+CWLAP:(3,"D-Link_DIR-600M",-64)

OK
AT+CWJAP="MERCURY_6EDA5C","[redacted]21"

OK
AT+CWJAP?
+CWJAP:"MERCURY_6EDA5C"

OK
AT+CIPMUX=1

OK
AT+CIPSTART=4,"TCP","192.168.1.104",9999

OK
```



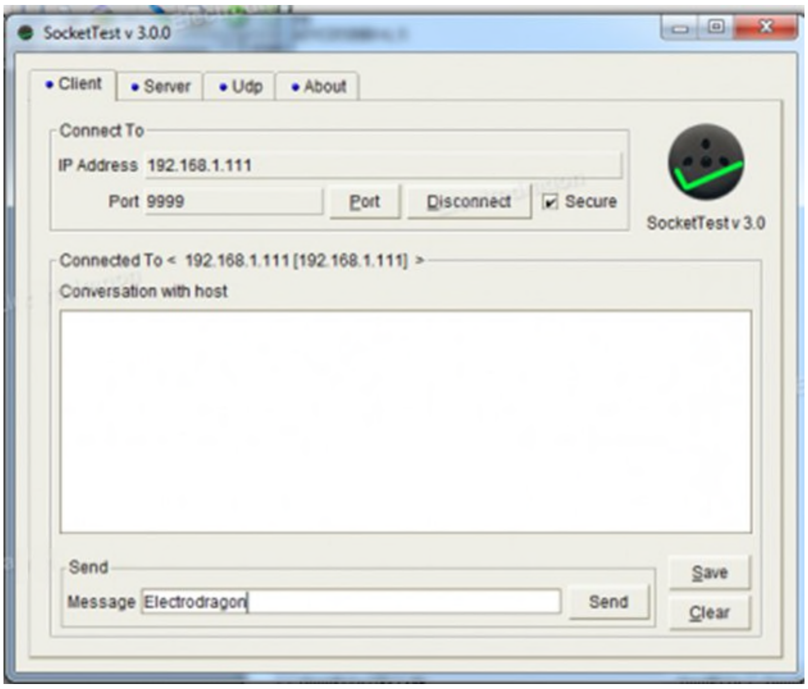
```
AT+CIPMUX=1

OK
AT+CIPSTART=4,"TCP","192.168.1.104",9999

OK
AT+CIPSEND=4,5

>Electr
busy
o
busy
d
busy
r
busy
a
busy

busy

SEND OK
AT+CIPServer=1,9999

ERROR
AT+CIPSERVER=1,9999

OK

OK
AT+CIPSR=?

OK
AT+CIPSR?
no this fun
AT+CIPSR
192.168.1.111

+IPD,0,15:electrodragon

OK
```

Socket test running result

- In the socket test, do not tick the "secure" in TCP client, it causes unstability



IP Address 192.168.1.104
Port 9999
Connected Client : < 192.168.1.111 [192.168.1.111] >
Conversation with Client
```
> Server Started on Port: 9999
> ------------------------------------
> New Client: 192.168.1.111
Elect
```
Message



IP Address 192.168.1.111
Port 9999
☑ Secure
Connected To < 192.168.1.111 [192.168.1.111] >
Conversation with host
Message Electrodragon

Other Arduino Demo Code

- In this case, the wifi module still connect to hardware serial (software serial port cannot higher than 19200 baud rate), and another software serial port should be created on Arduino and print out via another serial port
- So the connection should be

```
Wifi's uart to arduino hardware uart;
```

```
arduino's sotware UART to another serial port device, for example like FTDI basic, CP2102
breakout, etc, and this serial port device can connect to PC to read data
```

- change the SSID and password in code for your wifi router

```
#include <SoftwareSerial.h>
#define SSID        "xxxxxxxx"
#define PASS        "xxxxxxxx"
#define DST_IP      "220.181.111.85"    //baidu.com
SoftwareSerial dbgSerial(10, 11); // RX, TX
void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  Serial.setTimeout(5000);
  dbgSerial.begin(9600);  //can't be faster than 19200 for softserial
  dbgSerial.println("ESP8266 Demo");
  //test if the module is ready
  Serial.println("AT+RST");
  delay(1000);
  if (Serial.find("ready"))
  {
    dbgSerial.println("Module is ready");
  }
  else
  {
    dbgSerial.println("Module have no response.");
    while (1);
  }
  delay(1000);
  //connect to the wifi
  boolean connected = false;
  for (int i = 0; i < 5; i++)
  {
    if (connectWiFi())
    {
      connected = true;
      break;
    }
  }
  if (!connected) {
    while (1);
  }
  delay(5000);
  //print the ip addr
  /*Serial.println("AT+CIFSR");
  dbgSerial.println("ip address:");
  while (Serial.available())
    dbgSerial.write(Serial.read());*/
  //set the single connection mode
  Serial.println("AT+CIPMUX=0");
```

```
}
void loop()
{
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += DST_IP;
  cmd += "\",80";
  Serial.println(cmd);
  dbgSerial.println(cmd);
  if (Serial.find("Error")) return;
  cmd = "GET / HTTP/1.0\r\n\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
  if (Serial.find(">"))
  {
    dbgSerial.print(">");
  } else
  {
    Serial.println("AT+CIPCLOSE");
    dbgSerial.println("connect timeout");
    delay(1000);
    return;
  }
  Serial.print(cmd);
  delay(2000);
  //Serial.find("+IPD");
  while (Serial.available())
  {
    char c = Serial.read();
    dbgSerial.write(c);
    if (c == '\r') dbgSerial.print('\n');
  }
  dbgSerial.println("====");
  delay(1000);
}
boolean connectWiFi()
{
  Serial.println("AT+CWMODE=1");
  String cmd = "AT+CWJAP=\"";
  cmd += SSID;
  cmd += "\",\"";
  cmd += PASS;
  cmd += "\"";
  dbgSerial.println(cmd);
  Serial.println(cmd);
  delay(2000);
  if (Serial.find("OK"))
  {
    dbgSerial.println("OK, Connected to WiFi.");
    return true;
  } else
  {
    dbgSerial.println("Can not connect to the WiFi.");
    return false;
  }
}
```

Debug and Note

- Better use standalone power source, not using power from USB-TTL module, it may not able to provide sufficient current.
- Module will disconnect "unlink" TCP/UDP when no data go through
- Wait AT commands feedback and continue, otherwise will return "busy"
- Potential cause for "error" : password length must be more than 8 bytes, use multiple connection and mode three, try disconnect current connection before try "AT+CWLAP" (module will reconnect after restart), re-flash firmware.
- mac address please check in your router page or use arp to check.
- (1st Oct.) change CR to CR/LF (\r\n in coding), which means "carriage return and line feed" for new firmware version 0.92

Firmware Log

All version files:

- 0.8 early version not available anymore
  - [see - Firmware log - ESP8266 9.0 AT BIN.bin (zip file)]
  - 0.91: one-click-to-done tool in Chinese [see - Firmware log - ESP8266 updating tool V091 (zip file)] Choose right serial port, click button on the right (00160901)
  - 0.92 by cloud update : see - [Cloud updating your Wi07C ESP8266 now (pdf file)] (00170901 and 00180902)
  - [see - Firmware Log - V0.9.2.2 AT Firmware.bin (zip file)]   (0018000902 modified 0.922)

Other modified firmware:

- [see - Firmware Log - GPIO ESP8266.(zip file)] (Password electrodragon).

Old firmware SDK

- Old firmware : [see - Firmware Log - Esp iot SDK v0.6 (zip file)]
- [see - Firmware Log - ESP8266 SDK v0.9.1 (zip file)]

Firmware uploading tool

- [see - Firmware Log - XTCOM UTIL (zip file)]
- How to use (see below)

```
Download the bin file
Set the module to update mode, connect the module : choose "tools" - "configure device"
Upload bin file: API Test - Flash image download, upload the bin file at 0x0000
```

Upload log

V0.922

- More stable version than the cloud updated version.
- Support to change baud rate, default baud rate is 9600, AT command is

```
inquiry range: AT+CIOBAUD=?
check current baudrate AT+CIOBAUD?
set: ：AT+CIOBAUD=9600, supported 9600, 19200, 38400, 57600, 74880, 115200, 230400,460800,
921600
```

- Support watchdog, auto restart when program have errors occurred, AT command: turn on watchdog AT+CSYSWDTENABLE; turn off watchgod: AT+CSYSWDTDISABLE