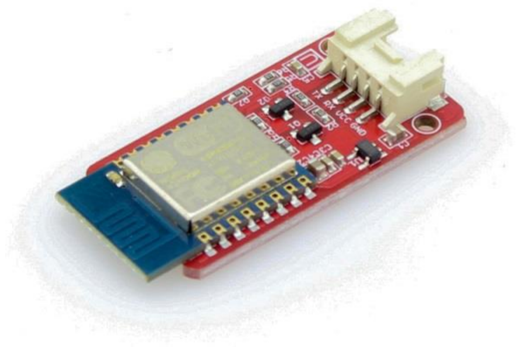
Crowtail Serial WiFi – V1.0

ESP8266MOD based

Arduino WiFi Module



Written by Palmieri Niccolò

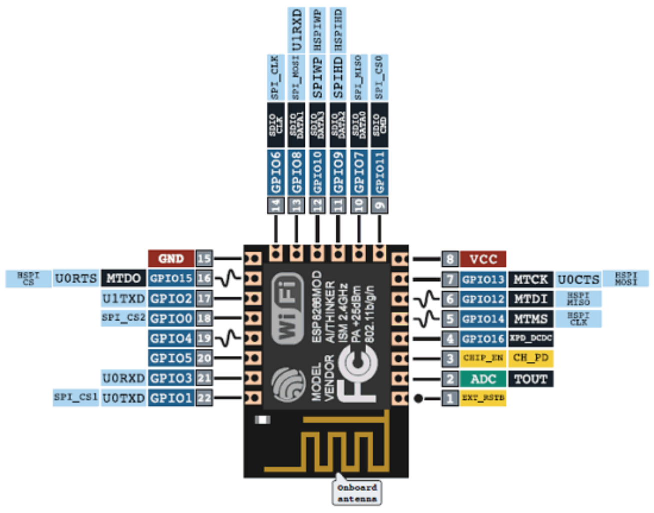
02.06.2018

Description

The Crowtail serial wifi module, based on ESP8266MOD, is a ultra-low power UART-WiFi module. It can be used in serial communication between arduino and other devices thanks to its features:

- UART Connection Mode;
- High speed serial port;
- 5V working voltage;
- 802.11 b/g/n protocol;
- WiFi Direct (P2P), soft-AP;
- Integrated TCP/IP protocol;
- Integrated TR switch, balun, LNA, power amplifier and matching network;
- Integrated PLL, regulators and power management units;
- +19,5 dBm output power in 802.11b mode.

The WiFi unit is ESP8266: it is a low-cost microchip with full TCP/IP stack and microcontroller capability.

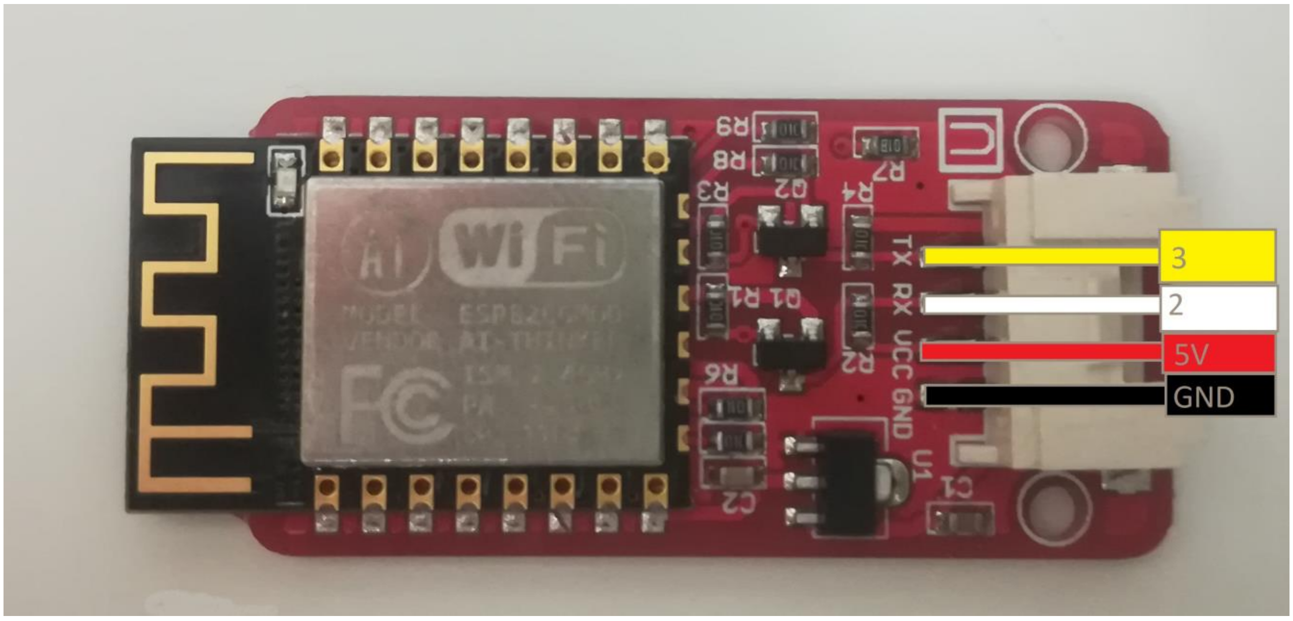


ESP8266MOD

It works at baud rate of 9600 but it requires a particular set up every time you start the chip.

Setup

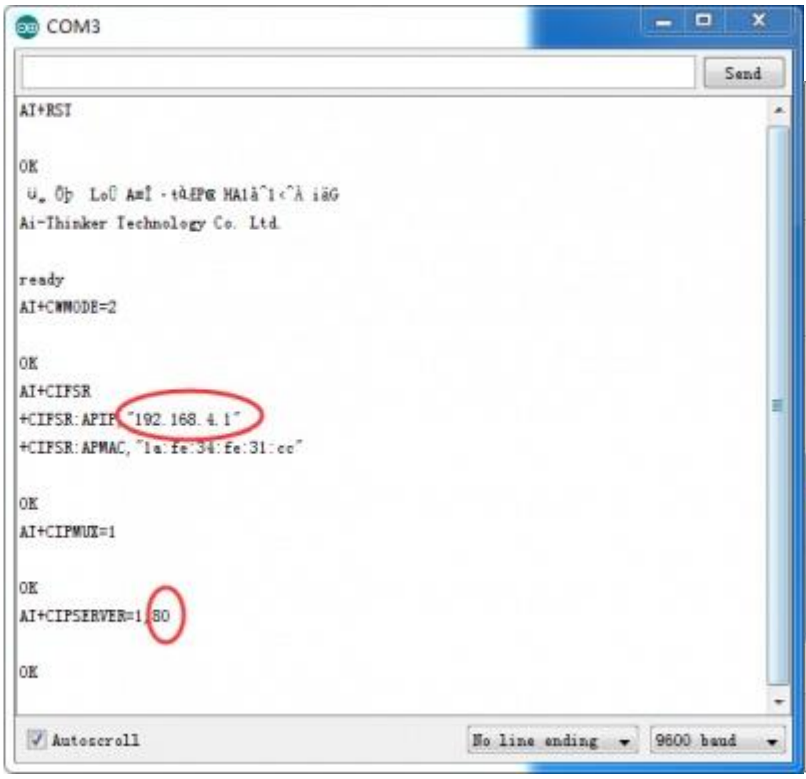
Before starting the module, you must perform certain setup operations. First of all, you need to correctly connect the crowtail wifi module pins to your Arduino controller. The RX and TX pins of the module will go to Arduino pins 2 and 3 respectively, instead the VCC and GND pins will be connected respectively to the 5V and GND pins of Arduino. You can see the connections in the image.



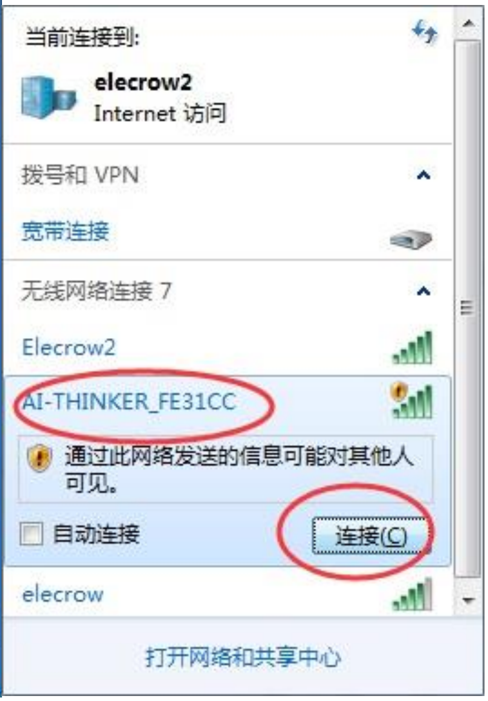
Now we can continue with the writing of the program, which will be like that in example:

```
#include <SoftwareSerial.h>
#define DEBUG true
 SoftwareSerial esp8266(2,3); // make RX Arduino line is pin 2, make TX Arduino line is pin 3.
                             // This means that you need to connect the TX line from the esp to the Arduino's pin 2
                             // and the RX line from the esp to the Arduino's pin 3
void setup(){
  Serial.begin(115200);
  esp8266.begin(115200); // esp8266 baud rate
  esp8266.print("AT+CIOBAUD=9600\r\n"); // change baud rate to 9600
  delay(100);
  esp8266.begin(9600); //change communication baud rate to 9600
  delay(100);
  sendData("AT+GMR\r\n",2000,DEBUG);// firmware info
  Serial.println("");
  sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point
  Serial.println("");
  sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address
  Serial.println("");
  sendData("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple connections
  Serial.println("");
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80*/
}
 void loop(){
  delay(10000);
  if(esp8266.available()) // check if the esp is sending a message{
   if(esp8266.find("+IPD,")){
     delay(1000);
     int connectionId = esp8266.read()-48; // subtract 48 because the read() function returns
                                         // the ASCII decimal value and 0 (the first decimal number) starts at 48
     String webpage = "<h1>Hello World! </h1>"; //print string on the webpage
     String cipSend = "AT+CIPSEND=";
     cipSend += connectionId;
     cipSend += ",";
     cipSend +=webpage.length();
     cipSend +="\r\n";
     sendData(cipSend,1000,DEBUG);
     sendData(webpage,1000,DEBUG);
     String closeCommand = "AT+CIPCLOSE=";
     closeCommand+=5; // append connection id
     closeCommand+="\r\n";
     sendData(closeCommand,3000,DEBUG);
    }
  }
}

String sendData(String command, const int timeout, boolean debug){
    String response = "";
    esp8266.print(command); // send the read character to the esp8266
     long int time = millis();
    char ch;
    while( (time+timeout) > millis()){
     while(esp8266.available()){
      // The esp has data so display its output to the serial window
      ch = esp8266.read();
      //delay(1);// read the next character.
      response += ch;
     }
    }

    if(debug){
     Serial.print(response);
    }
    return response;
}
```

In the setup function, we order the wifi chip to change its refresh rate from 115200 to 9600 to allow proper communication between Arduino and wifi chips. then we execute some commands, that I will list later, that allow us to start, for example, a small server which will generate a new access point.



```
AT+RST


OK
 ü, Ōþ  Lo  A≡Î -tàEP¤ HA1â^1<^À iäG
Ai-Thinker Technology Co. Ltd.


ready
AT+CWMODE=2


OK
AT+CIFSR
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"1a:fe:34:fe:31:cc"


OK
AT+CIPMUX=1


OK
AT+CIPSERVER=1,80


OK
```



If we connect to the new access point, typing the IP address, communicated via serial monitor as image, we can view the text "Hello world!" on our web page.

In addition to this example, you can create custom pages in which you can enter particular data such as the detection of some sensors or things like that.

Commands

This is a list of command you can use to communicate with the Crowtail serial WiFi module.

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+CIOBAUD	set baud rate	set	AT+CIOBAUD=<value>			9600, 115200
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:

Commands	Description	Type	Set/Execute	Inquiry	test	Parameters and Examples
						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+CIPSTART	set up TCP or UDP connection	TCP/IP	1)single connection (+CIPMUX=0) AT+CIPSTART= <type>,<addr>,<port>; 2) multiple connection (+CIPMUX=1) AT+CIPSTART= <id><type>,<addr>, <port>	-	AT+CIPSTART=?	id = 0-4, type = TCP/UDP, addr = IP address, port= port; eg. Connect to 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=		AT+CIPSEND=?	eg. send data: AT+CIPSEND=4,15 and then enter the data.

Commands	Description	Type	Set/Execute	Inquiry	test	Parameters and Examples
			<id>,<length>			
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=?
AT+ CIPSTO	Set the server timeout	AT+CIPSTO=<time>	AT+CIPSTO?		<time>0~28800 in second	
+IPD	received data					

esp8266.begin(value) → set communication baud rate between Arduino and wifi module;

esp8266.print(“command”) → sending command to the module;

esp8266.find(“command”) → data reception function;