Embedded System Design & Application

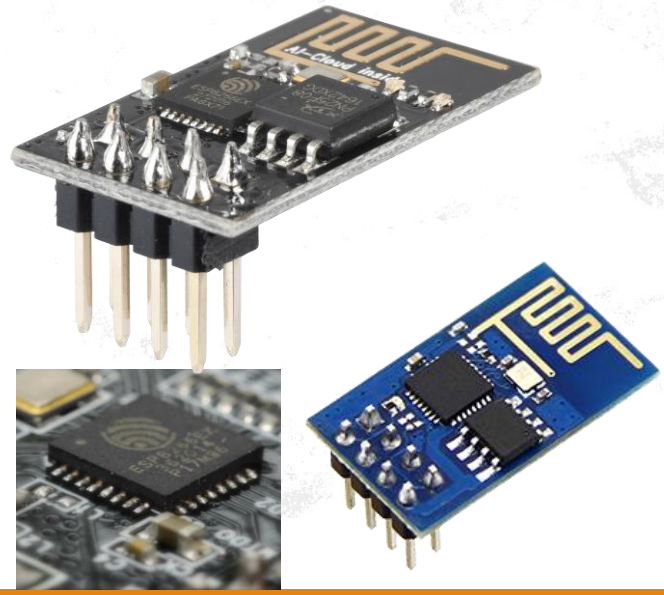
NodeMCU

Dr. Sarwan Singh

NIELIT Ropar

Agenda

- Types, Usage
- Interfacing with Arduino
- AT Commands
- Connecting to wifi network, posting data
- Uploading code in ESP8266
- ESP8266 as Web Server



ESP 8266

- System on Chip (SoC)
- Low cost
- Full TCP/IP stack (!!!!)
- Can be flashed with different firmwares
- Can also be programmed with Arduino IDE
- Many models

Features

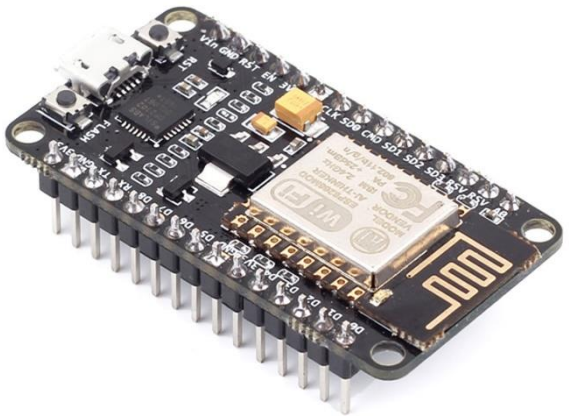
- Serial UART Interface
- It run LWIP
- 802.11 bgn
- WIFI Direct (P2P),SOFT-AP
- Built-in TCP/IP
- The AT command is perfect,efficient,concise
- Support three modes: AP, STA and AP+STA coexistence mode
- Onboard PCB Antenna

Specifications

- WIFI Direct (P2P), SOFT-AP
- Controlled via UART interface at 115200 baud with a simple set of AT commands
- Only 2 microcontroller pins needed for communication (RXD/TXD)
- Support three modes: AP, STA and AP+STA coexistence mode the TCP/IP protocol suit
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <10uA
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1/2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- ESP-01 PCB Antenna , after matching the distance to achieve open 400Meters.

ESP8266 Types

- NodeMCU



ESP01

ESP02

ESP03

ESP04

ESP05

ESP06

ESP07

ESP08

ESP09

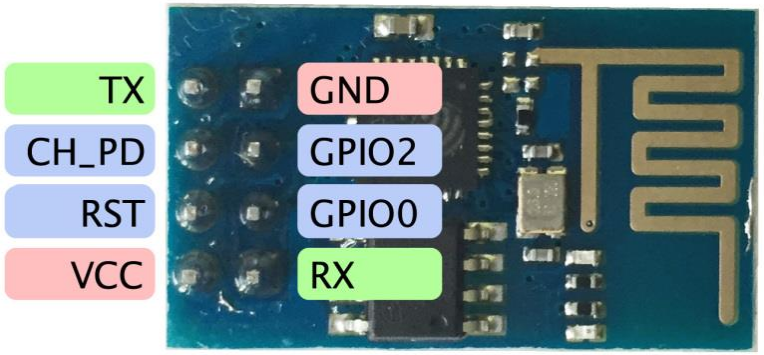
ESP10

ESP11

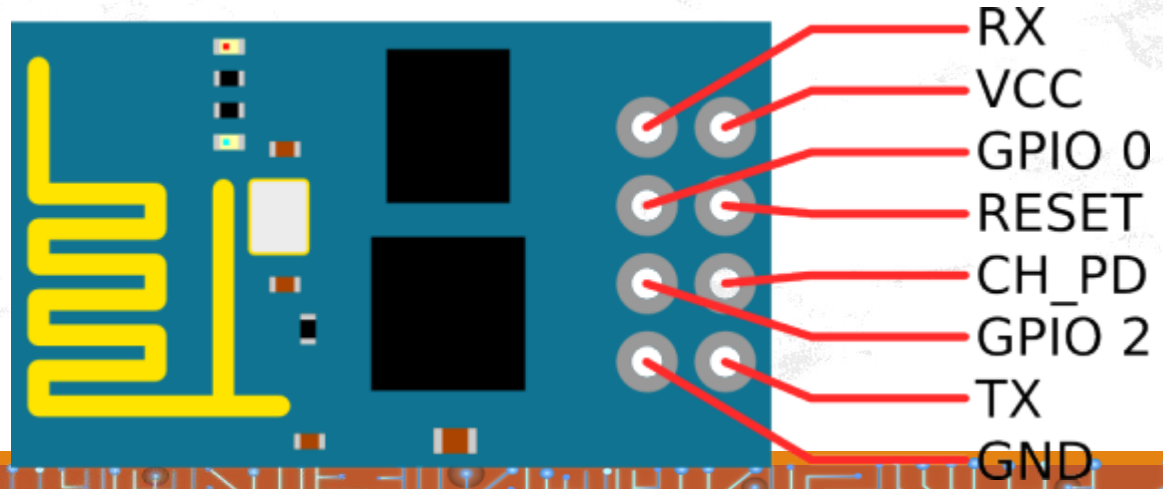
ESP12

ESP 8266 Pinout

- **ESP8266 is 3.3v device & cannot tolerate 5v levels**



Name	Type	Description
VCC	-	Voltage DC input (3.3 V)
RST	-	Reset
CH-PD	Chip Enable	High: On, chip works properly Low: Off, small current
TXD	Output	Serial Port Transmit Data (TTL level)
GND	-	Ground
GPIO 2		UART Tx during flash progamming
GPIO 0		SPICS2
RXD	Input	Serial Port Receive Data (TTL level)



ESP8266-01

- 802.11 b/g/n
- Input power: 3.3V
- I/O voltage tolerance: 3.6V Max
- Regular operation current draw: ~70mA
- Peak operating current draw: ~300mA

Highly Integrated

- ESP8266EX is among the most integrated WiFi chips in the industry with the size of 5mm x 5mm
  - it integrates the antenna switches,
  - RF balun, power amplifier,
  - low noise receive amplifier, filters,
  - power management modules while requires minimal external circuitry.

http://espressif.com/products/hardware/esp8266ex/overview/

32-bit MCU

- ESP8266EX integrates Tensilica L106 32-bit micro controller (MCU) which features extra low power consumption and 16-bit RSIC.
- The CPU clock speed is 80 MHz.
- It can also reach a maximum value of 160 MHz.
- Real Time Operation System (RTOS) is enabled.
- Currently, only 20% of MIPS has been occupied by the Wi-Fi stack, the rest can all be used for user application programming and development.

Low Power

- ESP8266EX has been designed for mobile, wearable electronics and Internet of Things applications with the aim of achieving the lowest power consumption with a combination of several proprietary technologies.
- The power saving architecture operates in 3 modes:
  - active mode,
  - sleep mode and
  - deep sleep mode.

Robustness

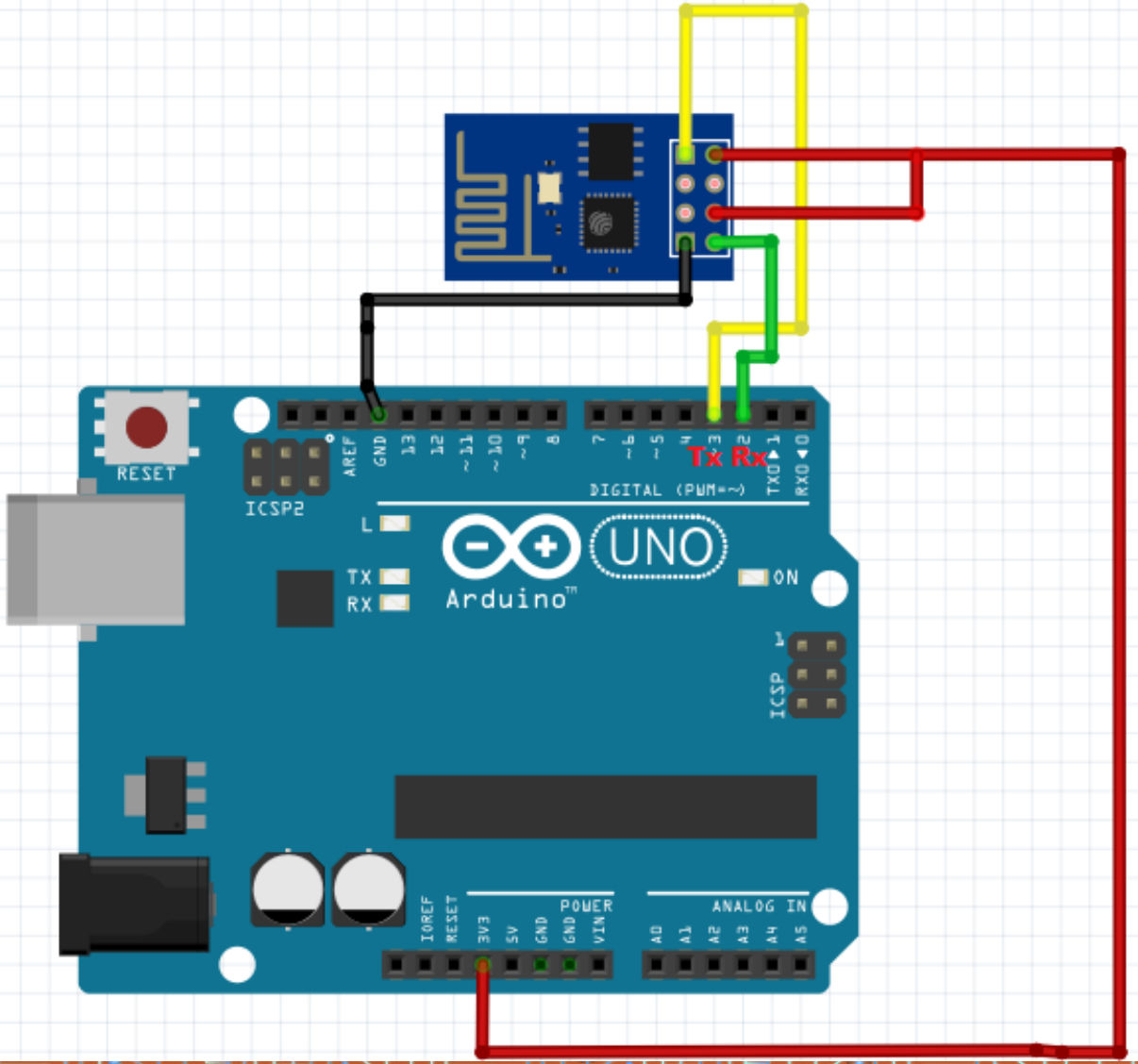
- By integrating more components on-chip, we have made the solution to be the most robust and manufacturable.
- Our solutions also feature the widest operating temperature range, from -40°C to +125°C.

Coding

INTERFACING ESP8266 WITH ARDUINO

Arduino interfacing with ESP8266

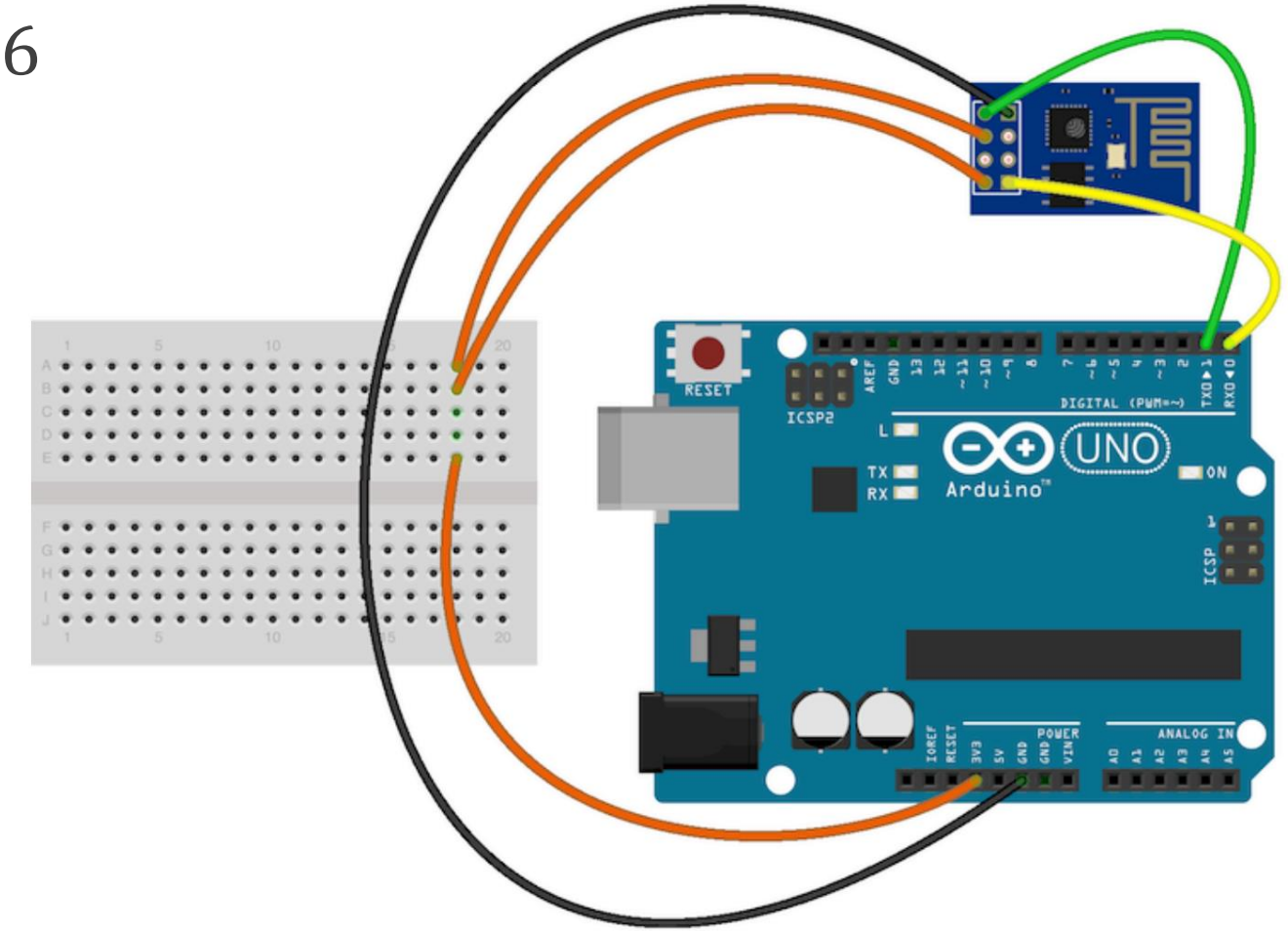
- Vcc
- Gnd
- Rx
- Tx



Arduino interfacing with ESP8266

Connections

Arduino	ESP8266
TX	RX
RX	TX
3.3V	VCC
3.3V	CH_PD
GND	GND



Software Serial

- The Arduino Uno has built-in support for serial communication on pins 0 and 1 (which also goes to the computer via the USB connection).
- The native serial support happens via a piece of hardware (built into the chip) called a UART.
- This hardware allows the Atmega chip to receive serial communication even while working on other tasks, as long as there room in the 64 byte serial buffer.
- The SoftwareSerial library has been developed to allow serial communication on other digital pins of the Arduino, using software to replicate the functionality (hence the name "SoftwareSerial").

Reference: https://www.arduino.cc/en/Reference/SoftwareSerial

SoftwareSerial

- It is possible to have multiple software serial ports with speeds up to 115200 bps.

Limitations

- If using multiple software serial ports, only one can receive data at a time.
- Not all pins on the Mega and Mega 2560 support SoftwareSerial

ESP8266 - AT Command

- ESP8266, in it's default configuration, boots up into the serial modem mode.
- In serial mode communication is possible using a set of **AT commands**.

AT Commands

Basic	WiFI layer	TCPIP Layer
AT	AT+CWMODE	AT+CIPSTATUS
AT+RST	AT+CWJAP	AT+CIPSTART
AT+GMR	AT+CWLAP	AT+CIPSEND
AT+GSLP	AT+CWQAP	AT+CIPCLOSE
ATE	AT+CWSAP	AT+CIFSR
	AT+CWLIF	AT+CIPMUX
	AT+CWDHCP	AT+CIPSERVER
	AT+CIPSTAMAC	AT+CIPMODE
	AT+CIPAPMAC	AT+CIPSTO
	AT+CIPSTA	AT+CIUPDATE
	AT+CIPAP	+IPD

Arduino Code

```
void setup()
{    Serial.begin(115200);     // Begin serial monitor to receive
115200                                    // bits per second (BAUD
RATE)
     WiFi_Serial.println("AT+UART_DEF=9600,8,1,0,0");
    // set WiFi Send/Receive at 115200 bits per second
    // (BAUD RATE) to 9600 bits per second
    WiFi_Serial.begin(9600);
    // Begin SoftwareSerial with ESP at 9600 bps (BAUD RATE)
    WiFi_Serial.println("ATE0"); // turn echo off
    WiFi_Serial.println("AT+CWQAP");
   // Disconnect from previous network connections

}
```

https://github.com/sarwansingh/Arduino/tree/master/ESP_TestPostData

```
void WIFI_Check()
{
 WiFi_Serial.println("AT"); // send a Attention command
 if (WiFi_Serial.available())
 {
     if (WiFi_Serial.find("OK")) // check with expected output
     {
      Serial.println("WIFI PLUGGED ON TO THE BOARD..!");
       WiFi_Serial.println("AT+CWMODE=1");
                                        //set mode to client mode
      isESPonBoard = true;
     }
 } else {
     Serial.println("WIFI NOT PLUGGED..!   ");    Serial.println();
     Serial.println("PLUG IN YOUR WIFI CHIP");    Serial.println();

 }
}
```

AT+CWMODE - WIFI mode（station, AP, station + AP）

Variant	Command	Response	Function
Test	AT+CWMODE=?	+CWMODE:(1-3) OK	List valid modes
Query	AT+CWMODE?	+CWMODE:modeOK	Query AP's info which is connect by ESP8266.
Execute	AT+CWMODE=mode	OK	Set AP's info which will be connect by ESP8266.

mode： An integer designating the mode of operation either 1, 2, or 3.
1 = Station mode (client)
2 = AP mode (host)
3 = AP + Station mode (Yes, ESP8266 has a dual mode!

void connectWiFi()

```
{
 WiFi_Serial.println("AT+CWJAP?");
        //check if WIFI connected to any WiFi network
 if (WiFi_Serial.available())
 {   if (WiFi_Serial.find("No AP"))
        //we receive response "No AP" when not connected to any
network
    {       Serial.println("NOT CONNECTED TO WIFI NETWORK");
        Serial.println("Trying to Connect to WiFi Network");
    }
    String cmd = "AT+CWJAP=\"";     // connected to specified
                //network passing mentioned WiFi username and
password
```

void connectWiFi() .... Contd...

```
Serial.println("-->"+cmd);
if (WiFi_Serial.available())
{    String RES_input = "";
      while (WiFi_Serial.available()) // read data into a variable
     {       RES_input += (char)WiFi_Serial.read();      }
      Serial.println(RES_input);
     if (WiFi_Serial.find("WIFI CONNECTED"))
      {
        Serial.println("CONNECTED TO WIFI NETWORK");
        isESPconnectedtoAP = true;
      }
   }
}
```

void loop()

```
{
 Serial.println("Welcome to ESP8266 interfacing");
  while (1)
  {
       WIFI_Check();
      if (isESPonBoard == true) {
            connectWiFi(); postData();
      }
      delay(4000);
  }
}
```

void post()

```
{  //form the JSON string with the available sensor data
    String data;
    data += "{\"username\":\"" + String(username)
  ;
    data +=  "\",\"name\":\"";
    data += String(Device_No);
    data += "\",\"sample1\":\"";
    data += String( CELSIUS); //(unsigned char)
    data += "\",\"sample2\":\"";
    data += String(HUM);
    data += "\",\"sample5\":\"";
    data += String(CO2);
    data += "\"}";
```

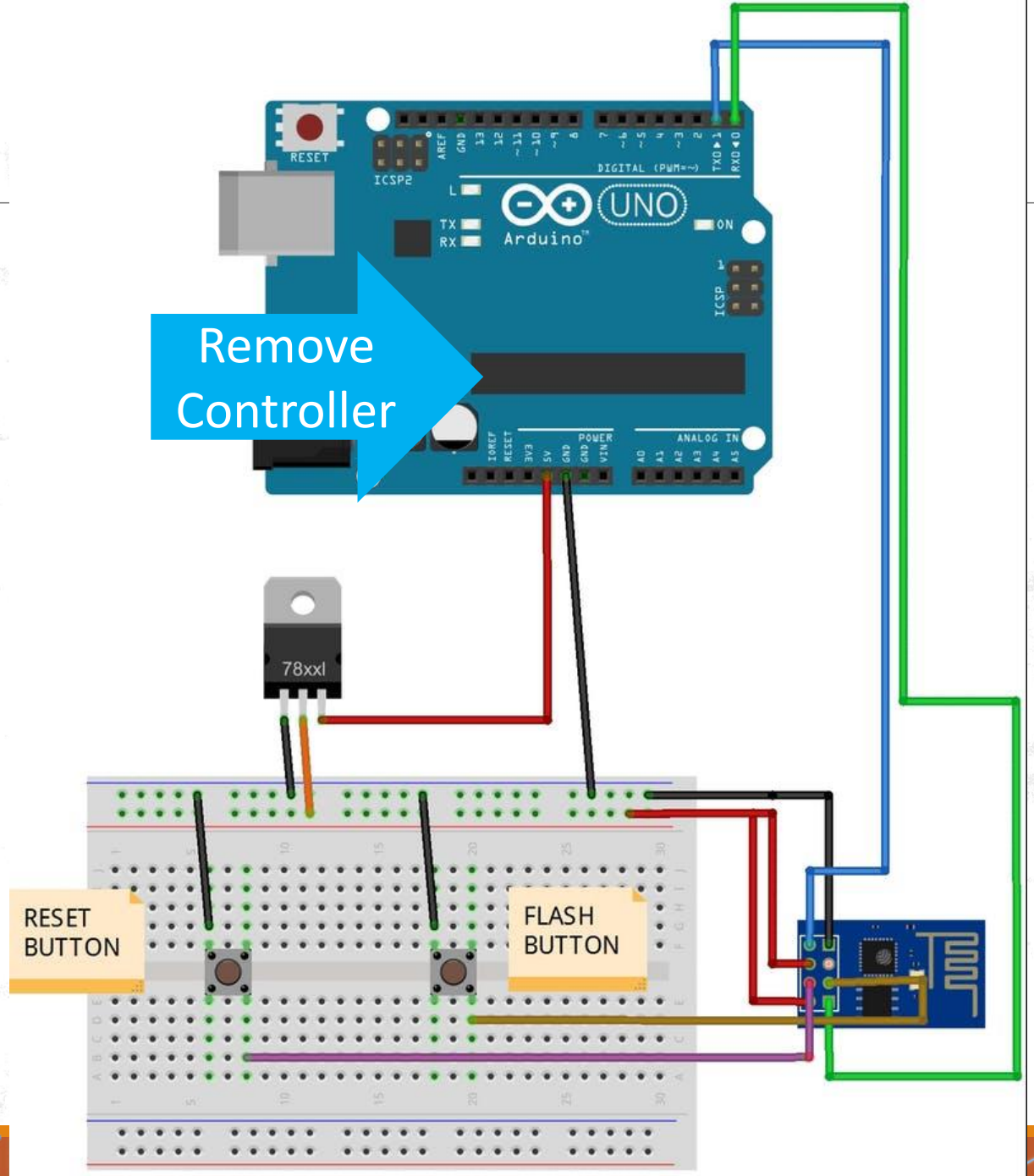
void post() ....

```
// form the header to post the WiFi data
String uri = "/iot/cht/rec.php";
String port = "80";
String http_req = "POST " + uri +
    " HTTP/1.1\r\n Host: " + DST_IP + ":" +
    port + "\r\n  Accept: */*\r\n" +
    "Content-Length: "+data.length() + "\r\n" ;
String http_req1 =
    "Content-Type: application/json\r\n\r\n" ;
```

```
void post()
// starts a TCP connection
String cmd = "AT+CIPSTART=\"TCP\",\"";
 cmd += DST_IP;      cmd += "\",80";
  WiFi_Serial.println(cmd);
  WiFi_Serial.print("AT+CIPSEND=");
  WiFi_Serial.println(Total_req_data_Length);
  if (WiFi_Serial.find(">"));       // when ">" is response from
    WiFi that means it is ready to receive the total length of
data
    {
     WiFi_Serial.print(http_req); //Send headder first
     WiFi_Serial.print(http_req1);
     WiFi_Serial.print(data); //later send data
    }
```

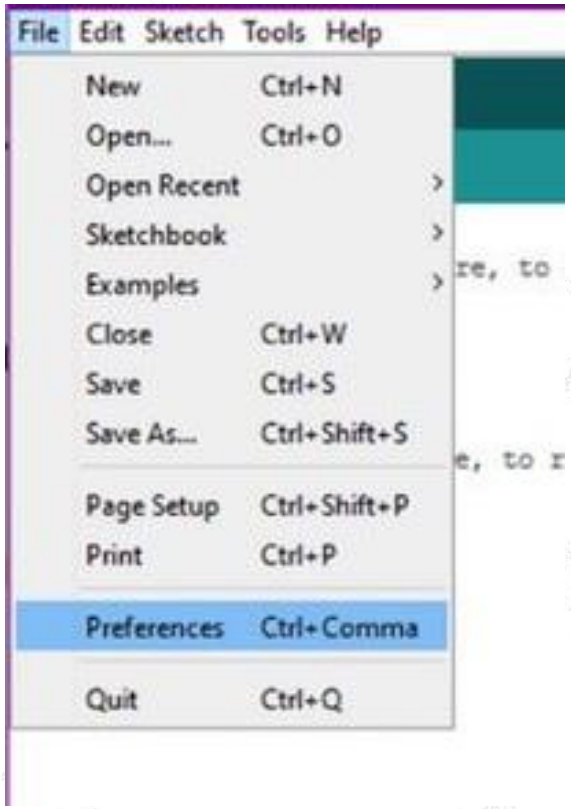
Uploading Code in ESP8266

Arduino	ESP8266 ESP-01
GND	GND
TX	TX
RX	RX
3.3V	VCC (3.3v req)

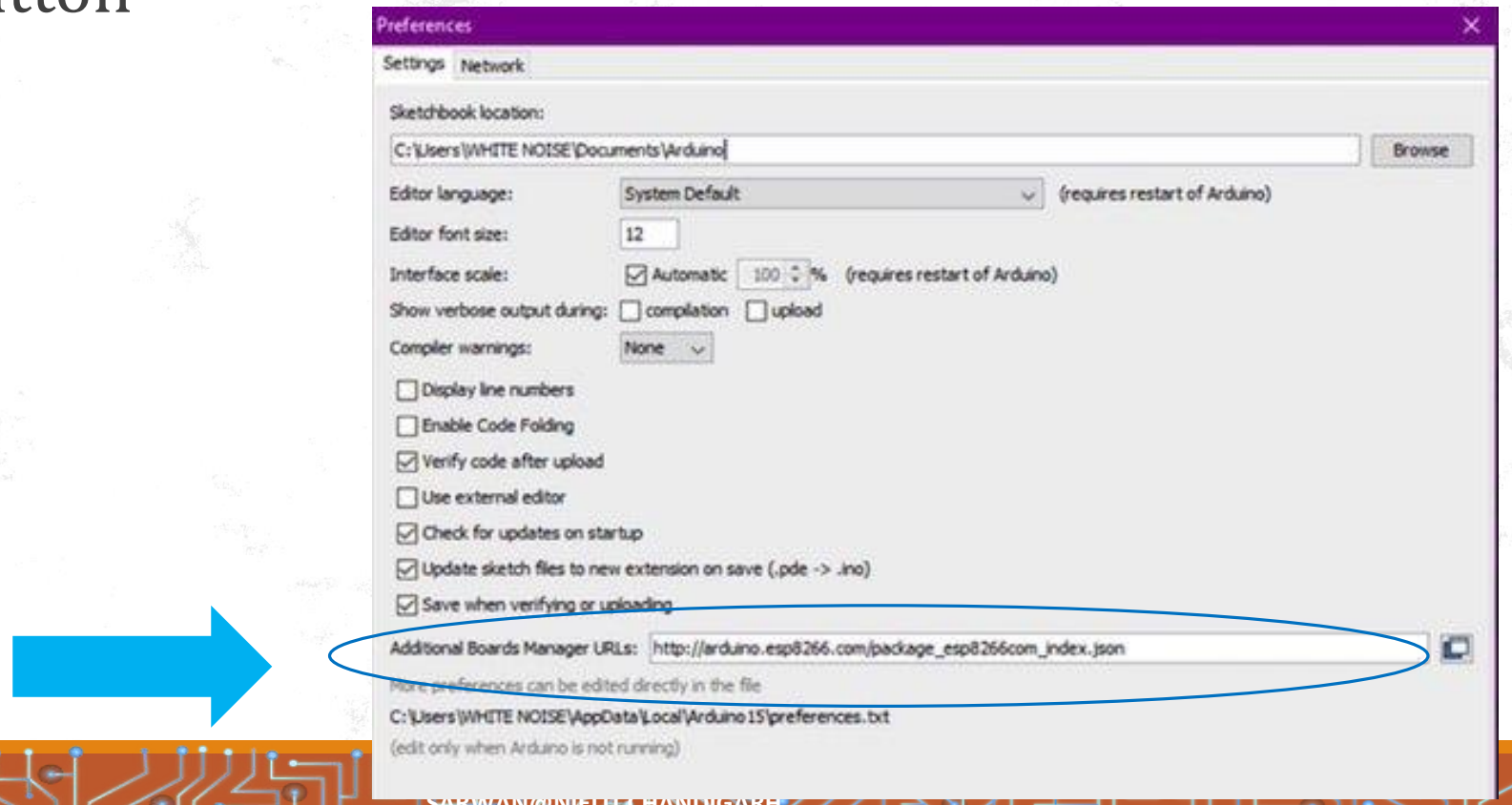


Installing the ESP8266 Platform

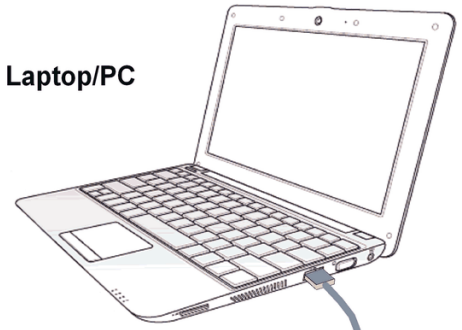
- First, the Arduino environment has to be set up to make it compatible with the **ESP-01** module.
- requirement : **Arduino version 1.6.4 or higher**
- Open the preferences window from the Arduino IDE. Go to File >Preferences

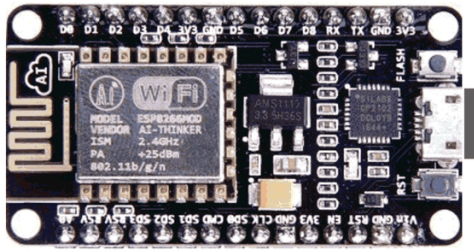


- Enter **http://arduino.esp8266.com/stable/package_esp8266com_index.json** into Additional Board Manager URLs field and click the “OK” button



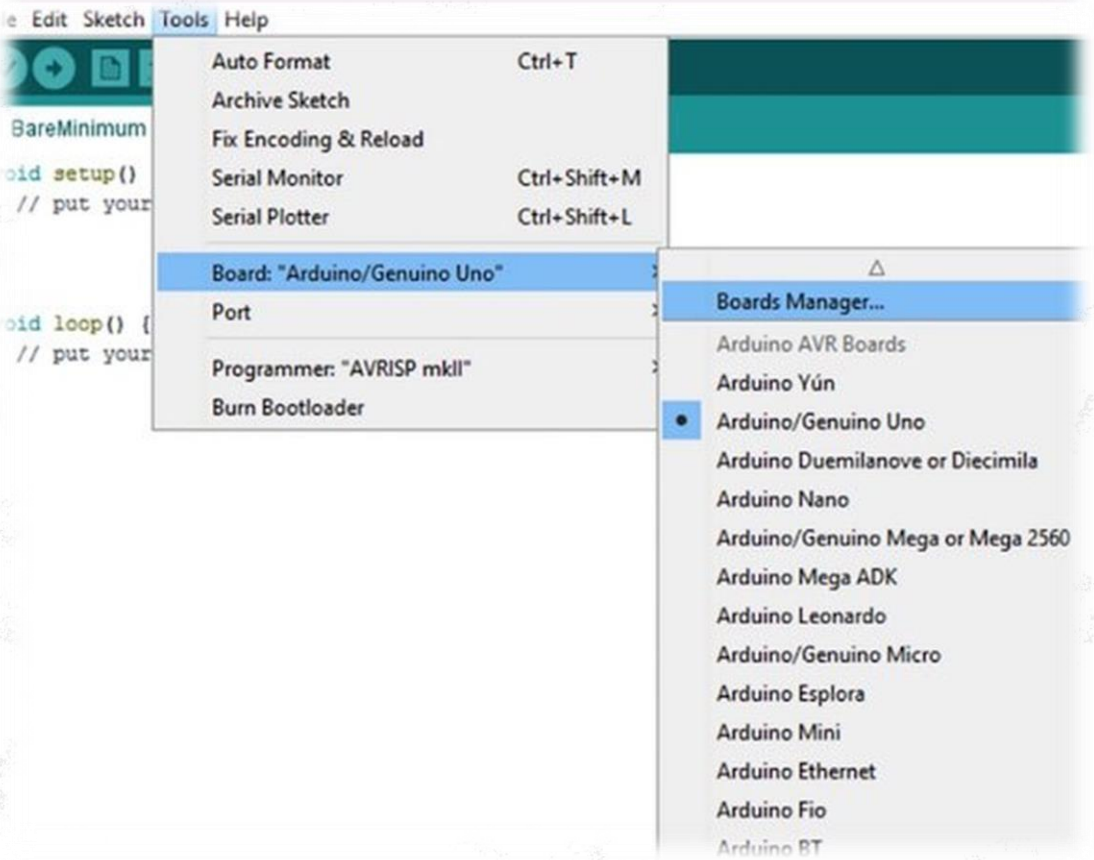
- Open boards manager. Go to: Tools > Board > Boards Manager...



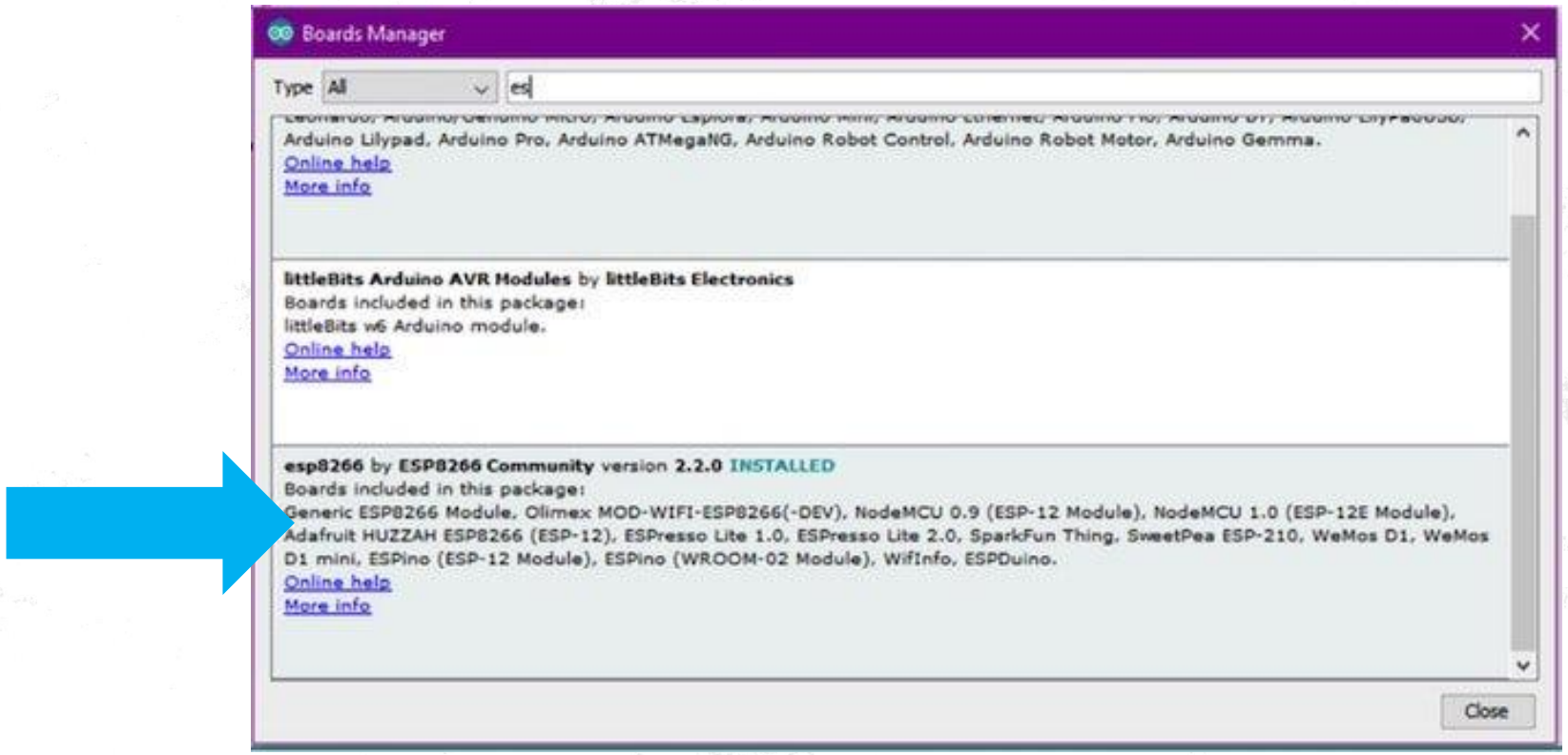


USB Cable

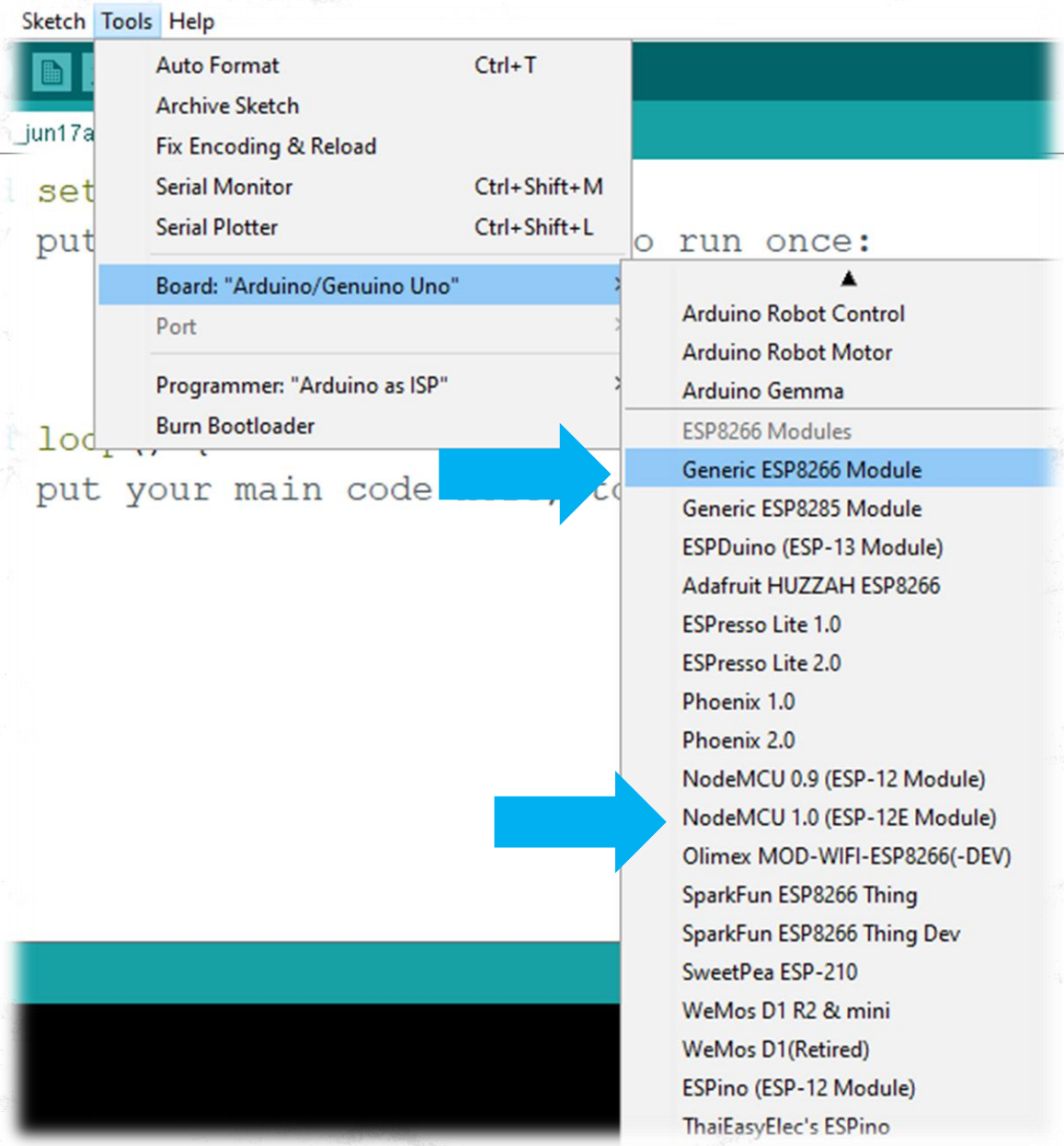
NodeMCU



- Scroll down, select the ESP8266 board menu and install “esp8266 platform”.



- Choose your ESP8266 board from Tools > Board > Generic ESP8266 Module



```
#include <ESP8266WiFi.h>

const char* ssid = "YOUR_SSID";//type your ssid
const char* password = "YOUR_PASSWORD";//type your password

int ledPin = 2; // GPIO2 of ESP8266
WiFiServer server(80);
```

https://github.com/sarwansingh/Arduino/tree/master/ESP8266_Webserver

```
void setup() {
  Serial.begin(115200);
  delay(10);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
  // Connect to WiFi network
  Serial.println();  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() !=
  WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi
connected");

  // Start the server
  server.begin();
  Serial.println("Server
started");

  // Print the IP address
  Serial.print("Use this URL to
connect: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}
```

```
void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  // Read the first line of the request
   String request = client.readStringUntil('\r');
   Serial.println(request);
   client.flush();

  // Match the request
   int value = LOW;
  if (request.indexOf("/LED=ON") != -1) {
    digitalWrite(ledPin, HIGH);
    value = HIGH;
  }
  if (request.indexOf("/LED=OFF") != -1){
    digitalWrite(ledPin, LOW);
    value = LOW;
  }
```

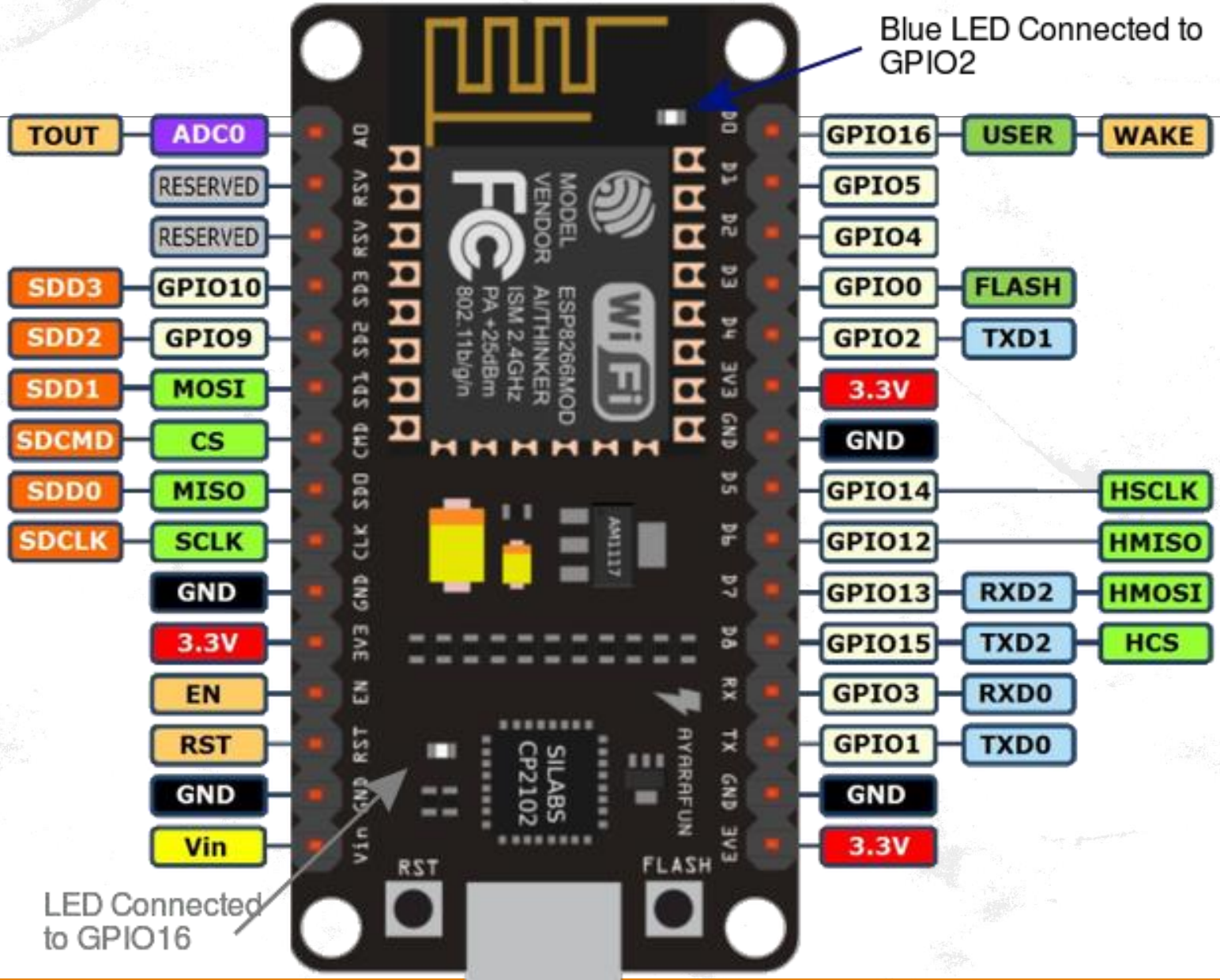
```
// Set ledPin according to the request
//digitalWrite(ledPin, value);

 // Return the response
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type:
   text/html");
 client.println(""); //  do not forget this
   one
 client.println("<!DOCTYPE HTML>");
 client.println("<html>");


 client.print("Led pin is now: ");


 if(value == HIGH) {
   client.print("On");
  client.println("<br><br>");
  client.println("Click <a
href=\"/LED=ON\"> here</a> turn the
LED on pin 2 ON<br>");
  client.println("Click <a
href=\"/LED=OFF\"> here</a> turn the
LED on pin 2 OFF<br>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disonnected");
  Serial.println("");

}
```

NodeMCU using Arduino IDE

- File > preferences

http://arduino.esp8266.com/stable/package_esp8266com_index.json

- Now close Preference window and **go to Tools -> Board -> Boards Manager**


Blue LED Connected to GPIO2
TOUT
ADC0
RESERVED
RESERVED
SDD3
GPIO10
SDD2
GPIO9
SDD1
MOSI
SDCMD
CS
SDD0
MISO
SDCLK
SCLK
GND
3.3V
EN
RST
GND
Vin
GPIO16
USER
WAKE
GPIO5
GPIO4
GPIO0
FLASH
GPIO2
TXD1
3.3V
GND
GPIO14
HSCLK
GPIO12
HMISO
GPIO13
RXD2
HMOSI
GPIO15
TXD2
HCS
GPIO3
RXD0
GPIO1
TXD0
GND
3.3V
RST
FLASH
LED Connected to GPIO16

- Github Repo

https://github.com/sarwansingh/IoT/tree/master/ClassExamples/CEPTAM_ESDA_Sept24