Internet of Things

The **Internet of things** (**IoT**) is the extension of Internet connectivity into physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled.

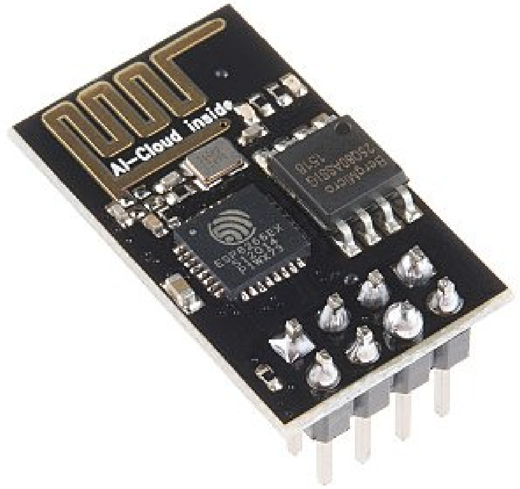
NodeMCU

NodeMCU is an eLua based firmware for the ESP8266 WiFi SOC from Espressif. The firmware is based on the Espressif NON-OS SDK and uses a file system based on spiffs. The code repository consists of 98.1% C-code.

The NodeMCU firmware is a companion project to the popular NodeMCU dev kits, ready-made open source development boards with ESP8266-12E chips

ESP8266

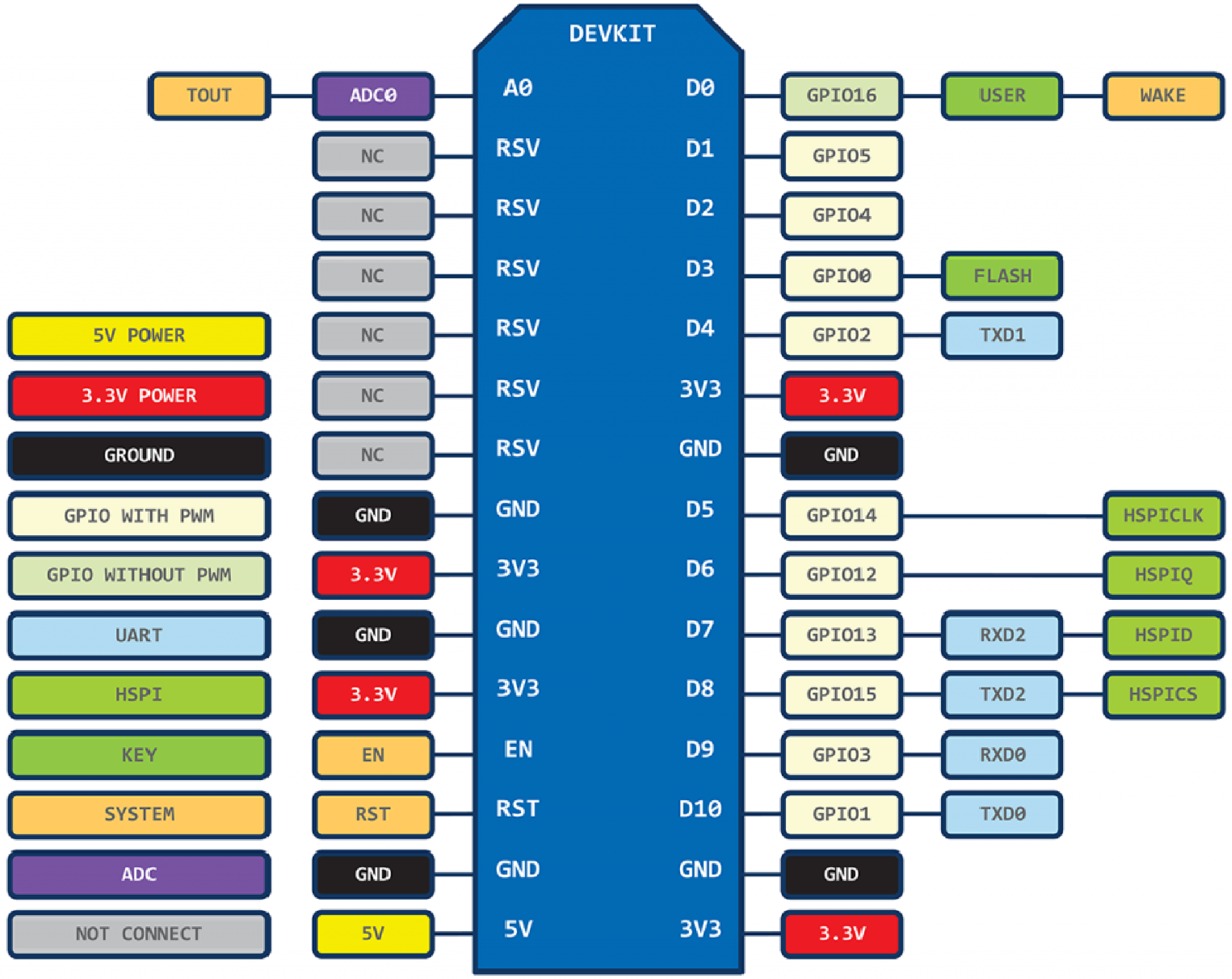
The **ESP8266** is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by manufacturer Espressif Systems in Shanghai, China.



The chip first came to the attention of western makers in August 2014 with the **ESP-01** module, made by a third-party manufacturer Ai-Thinker. The successor to these microcontroller chips is the ESP32, released in 2016.

This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands (also called the **AT commands**: "AT" meaning 'attention').

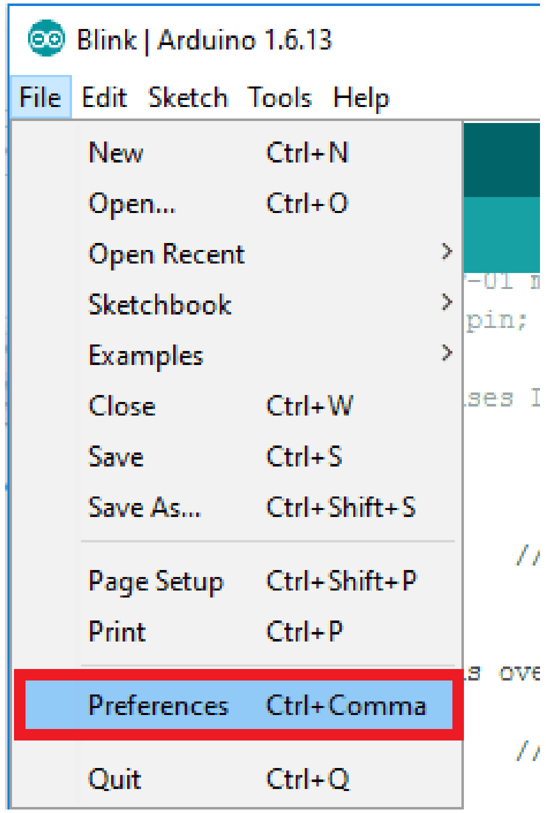
The NodeMCU programming can be as easy as in Arduino. The main difference is in the pinning of the board, described below:



Getting Started with NodeMCU

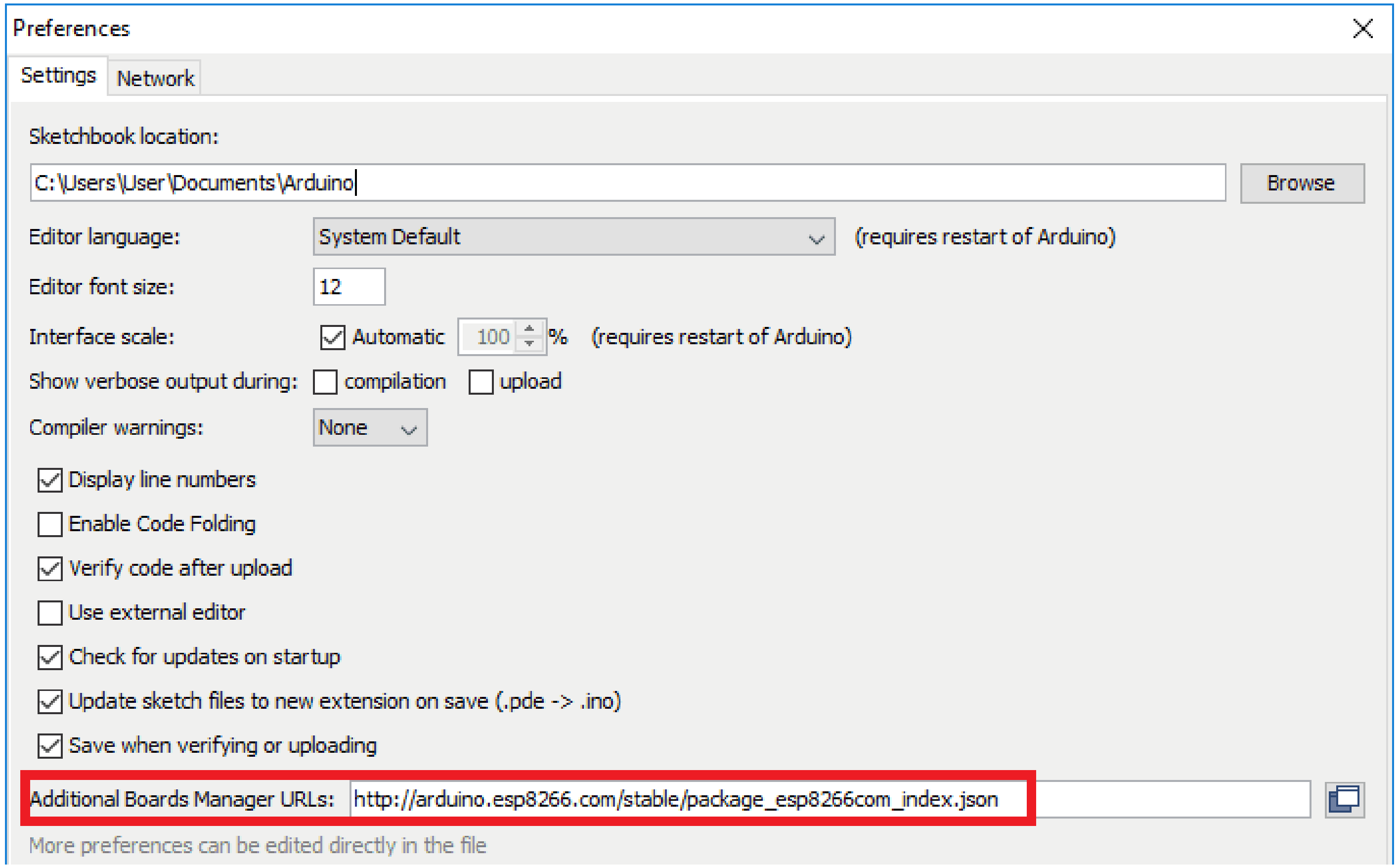
First open the Arduino IDE

Go to files and click on the preference in the Arduino IDE



copy the below code in the Additional boards Manager

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



Preferences

Settings | Network

Sketchbook location:

C:\Users\User\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☑ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

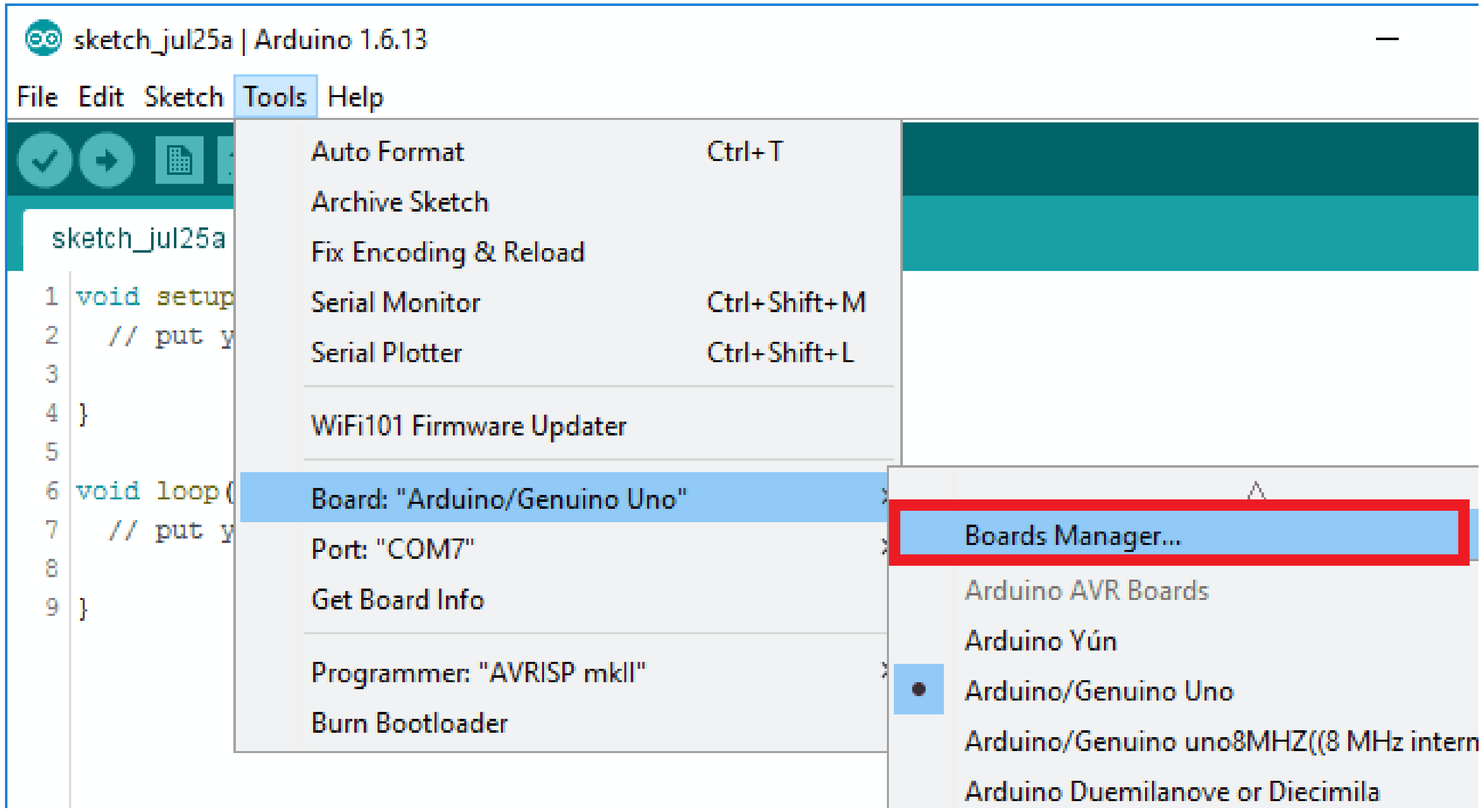
☑ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

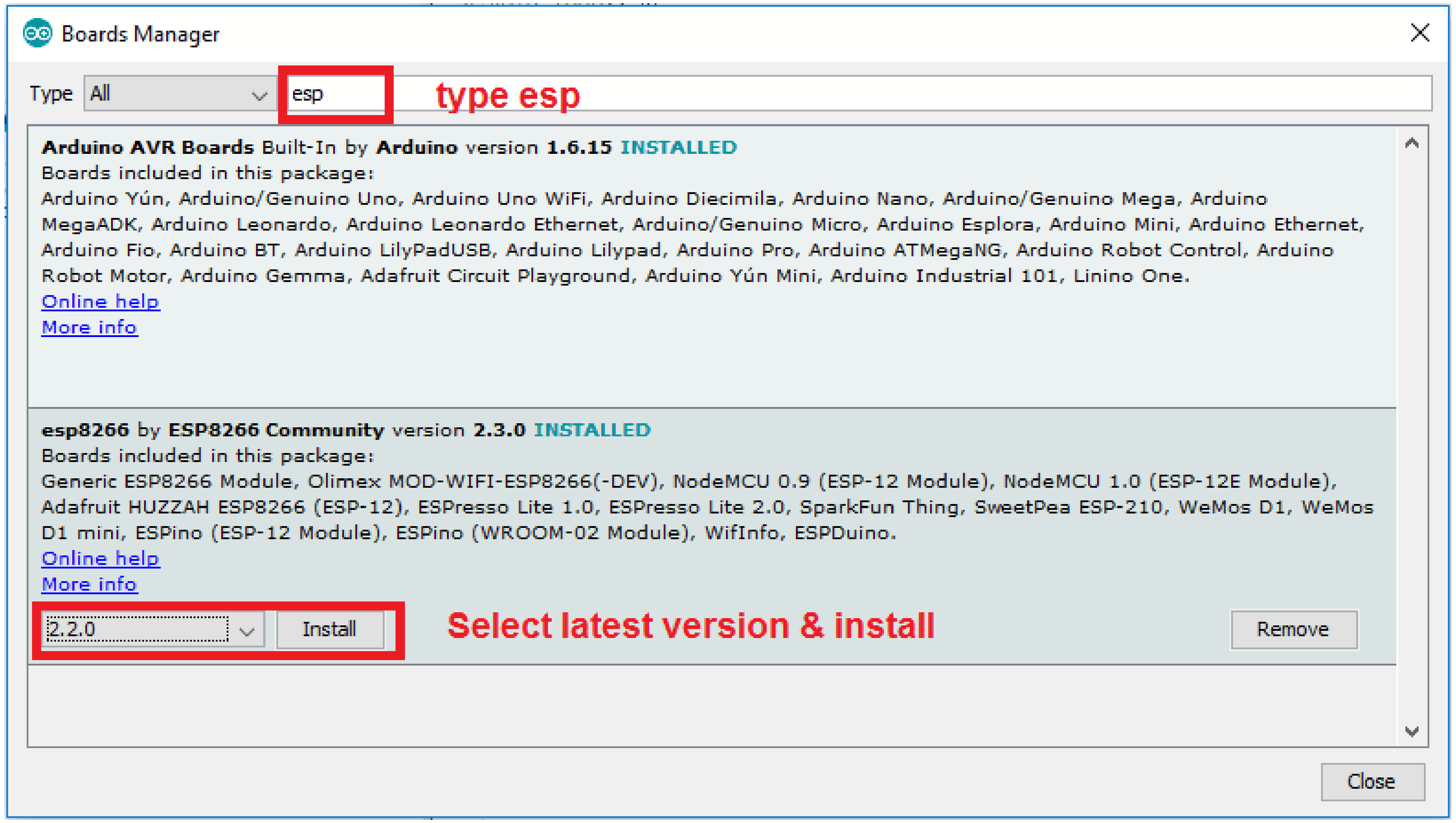
Click OK to close the preference Tab.

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



```
void setup
  // put y

}

void loop(
  // put y

}
```

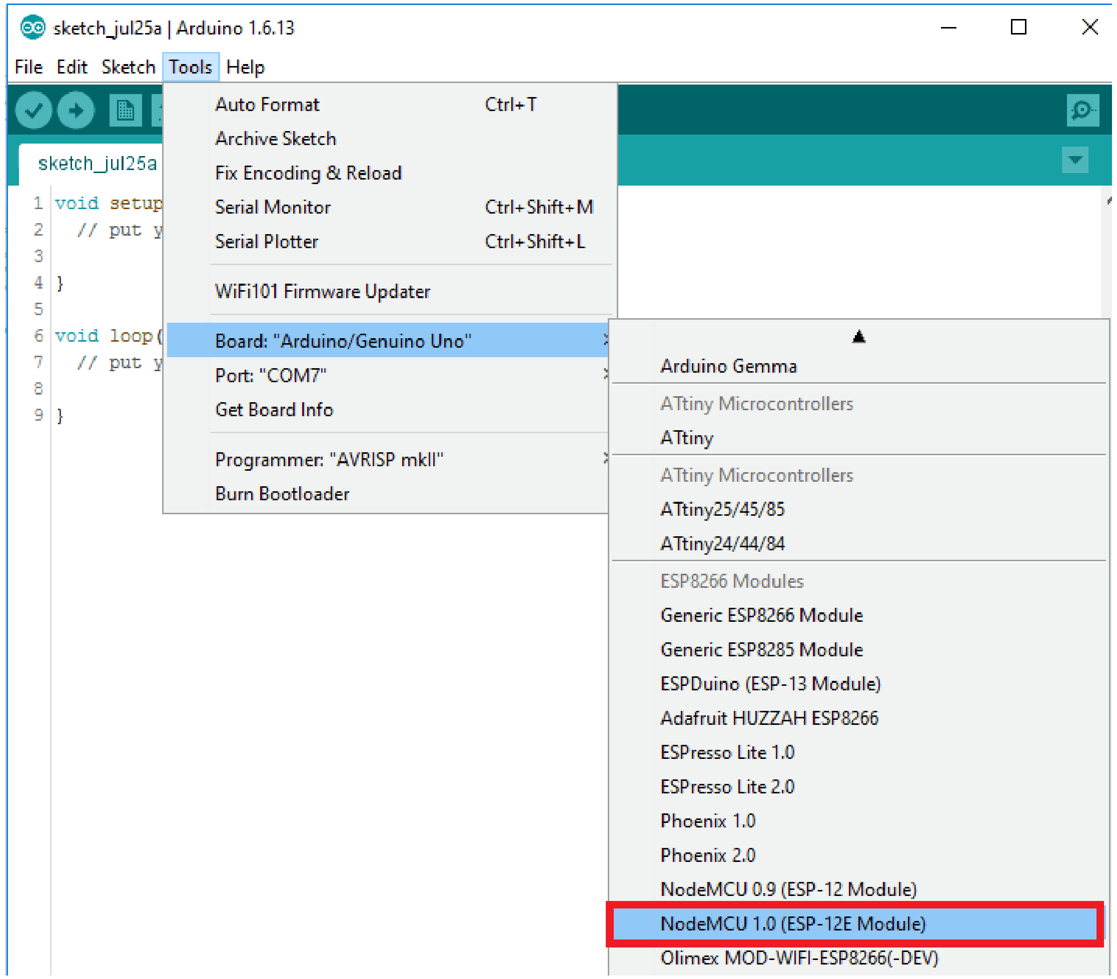
In Boards Manager window, **Type esp in the search box, esp8266 will be listed there below. Now select latest version of board and click on install**



Arduino AVR Boards Built-In by **Arduino** version **1.6.15** **INSTALLED**
Boards included in this package:
Arduino Yún, Arduino/Genuino Uno, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino/Genuino Mega, Arduino MegaADK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino/Genuino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATMegaNG, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
Online help
More info

esp8266 by **ESP8266 Community** version **2.3.0** **INSTALLED**
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

After installation of the board is complete, **open Tools->Board->and select NodeMCU 1.0(ESP-12E Module)**

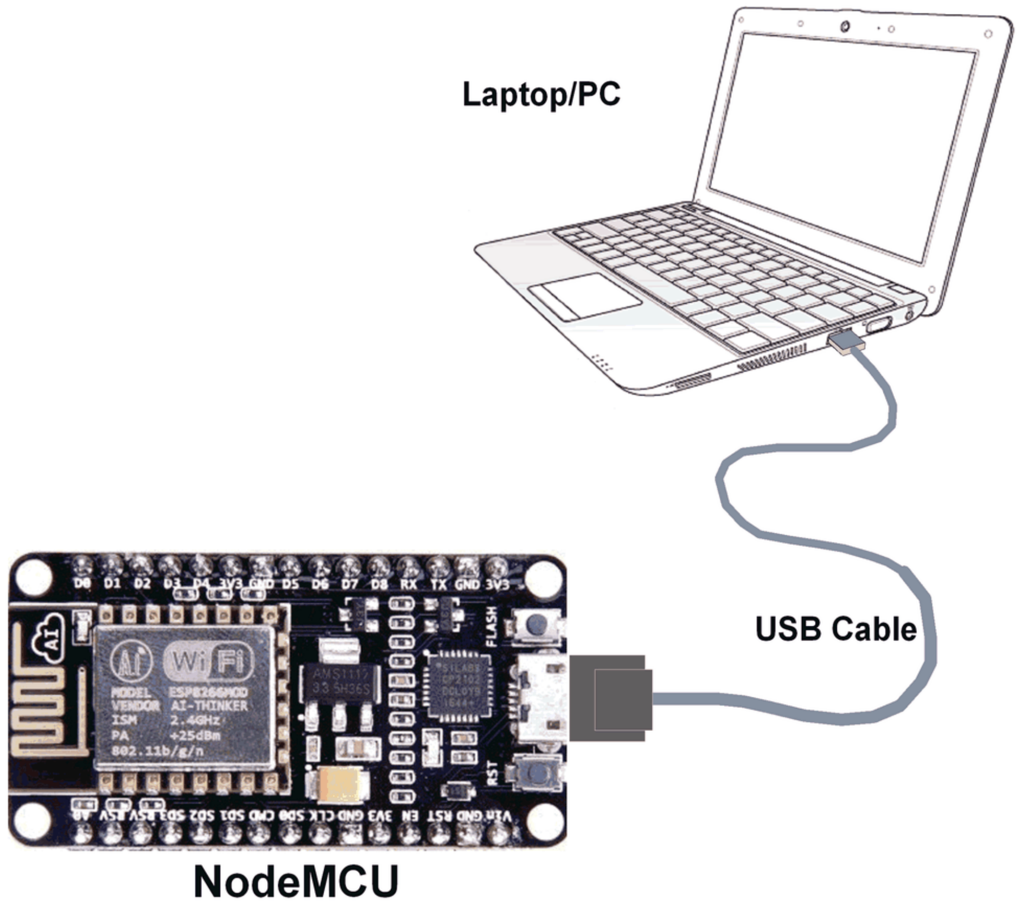


```
void setup
  // put y

}

void loop(
  // put y

}
```

Now Arduino IDE is ready for NodeMCU

Let’s see how to write simple Example Codes using Arduino IDE for NodeMCU.

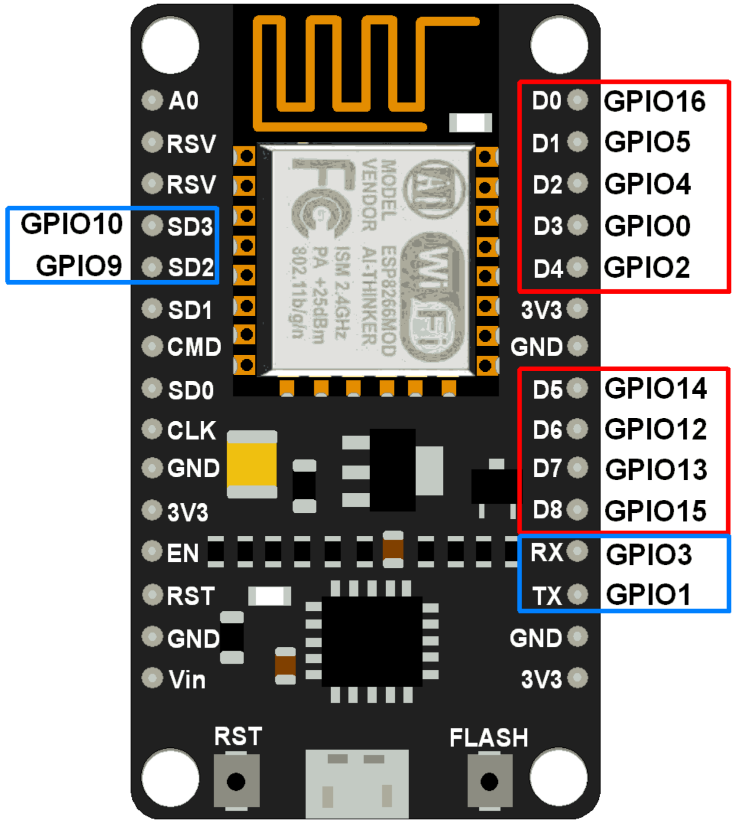
First connect NodeMCU Development Kit with PC as shown in below figure.



NodeMCU GPIO with Arduino IDE

- General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behavior can be controlled at the run time.
- NodeMCU Development kit provides access to these GPIOs of ESP8266. The only thing to take care is that NodeMCU Dev kit pins are numbered differently than internal GPIO notations of ESP8266 as shown in below figure and table. For example, the D0 pin on the NodeMCU Dev kit is mapped to the internal GPIO pin 16 of ESP8266

The GPIO's shown in blue box (1, 3, 9, 10) are mostly not used for GPIO purpose on Development Kit



Below table gives NodeMCU Dev Kit IO pins and ESP8266 internal GPIO pins mapping

Pin Names on NodeMCU Development Kit	**ESP8266 Internal GPIO Pin number**
D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15
D9/RX	GPIO3
D10/TX	GPIO1
D11/SD2	GPIO9
D12/SD3	GPIO10

- ESP8266 is a system on a chip (SoC) design with components like the processor chip. The processor has around 16 GPIO lines, some of which are used internally to interface with other components of the SoC, like flash memory.

- Since several lines are used internally within the ESP8266 SoC, we have about 11 GPIO pins remaining for GPIO purpose.

- Now again 2 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC from which compiled object code is downloaded.

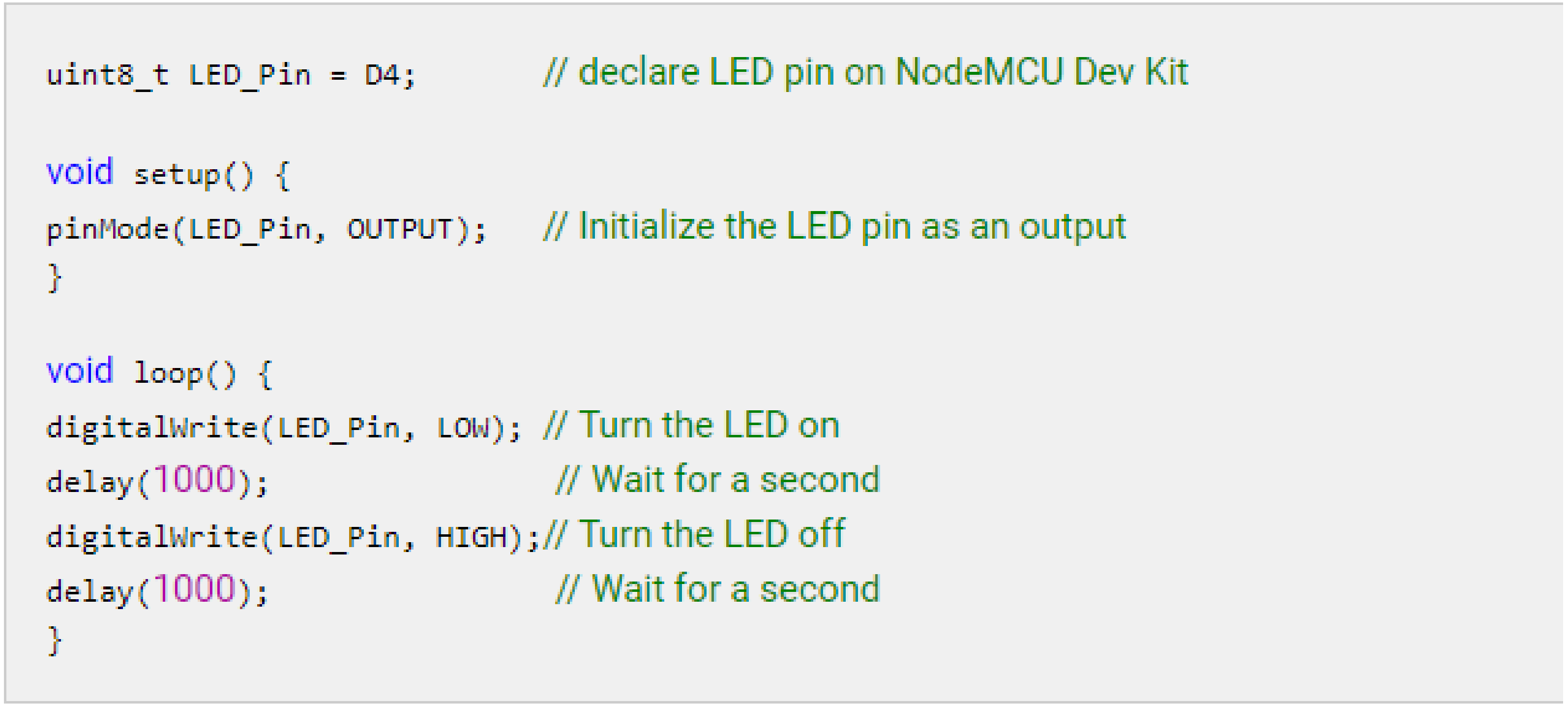
- Hence finally, this leaves just 9 general purpose I/O pins i.e. D0 to D8.

- As shown in above figure of NodeMCU Dev Kit. We can see RX, TX, SD2, SD3 pins are not mostly used as GPIOs since they are used for other internal process. But we can try with SD3 (D12) pin which mostly like to respond for GPIO/PWM/interrupt like functions.

Example

Let’s write an Arduino sketch for LED blinking on pin D4 of NodeMCU Dev Kit.

Arduino Sketch for LED Blink



```
uint8_t LED_Pin = D4;          // declare LED pin on NodeMCU Dev Kit

void setup() {
pinMode(LED_Pin, OUTPUT);      // Initialize the LED pin as an output
}

void loop() {
digitalWrite(LED_Pin, LOW);  // Turn the LED on
delay(1000);                  // Wait for a second
digitalWrite(LED_Pin, HIGH);// Turn the LED off
delay(1000);                  // Wait for a second
}
```