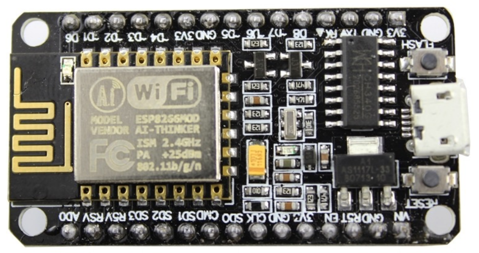

Wi Fi
MODEL ESP8266MOD
VENDOR AI-THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
FLASH
RESET

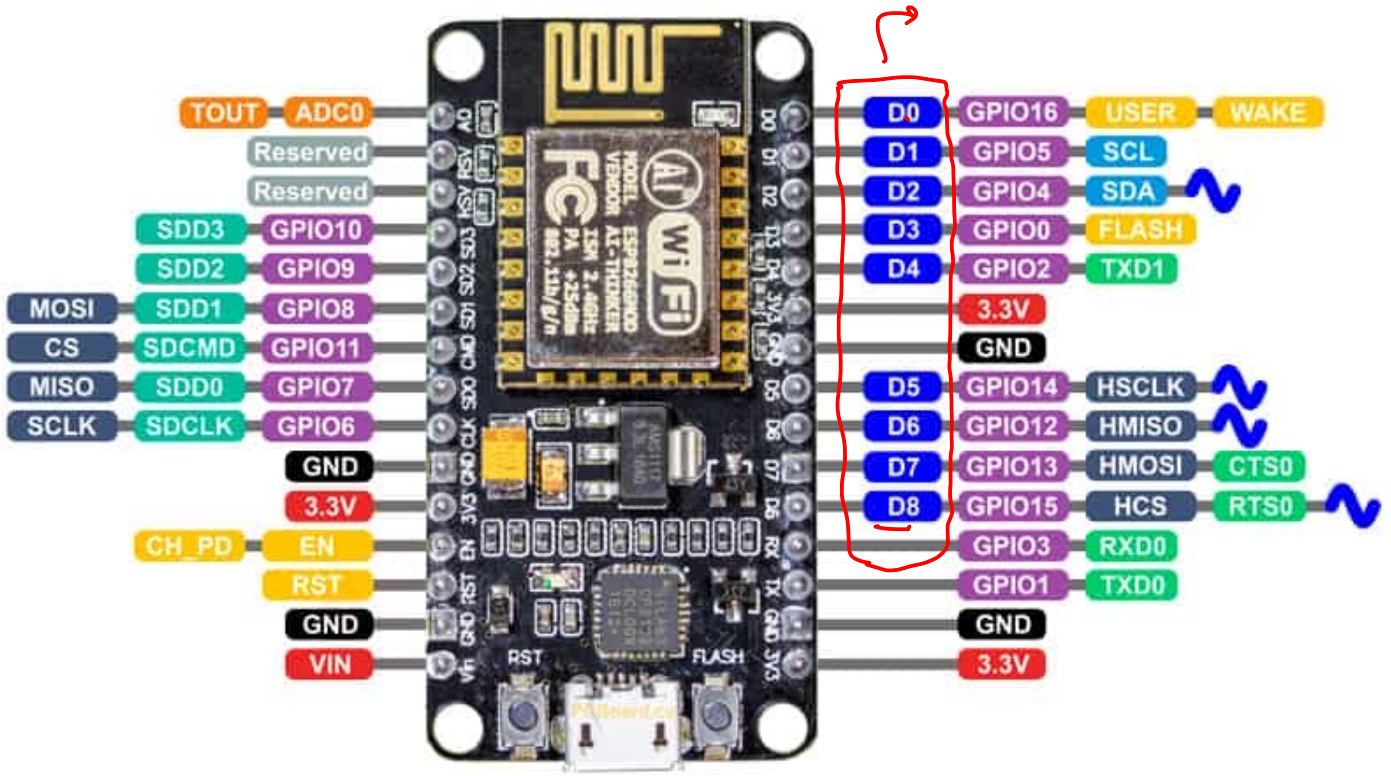
NodeMCU

Introduction

- The NodeMCU (**N**ode **M**icro**C**ontroller **U**nit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266
- The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK
- That makes it an excellent choice for the Internet of Things (IoT) projects of all kinds
- It uses an on-module flash-based SPIFFS file system
- NodeMCU is implemented in C and is layered on the Espressif NON-OS SDK
- The firmware was initially developed as is a companion project to the popular ESP8266-based NodeMCU development modules, but the project is now community-supported, and the firmware can now be run on any ESP module

Pinouts

pinmode $\rightarrow$ Read, write



🟥 Power 🟨 Control 🟦 I2C 🟦 SPI 🟩 SD Card ∿ PWM

⬛ GND 🟧 ADC 🟪 GPIO 🟩 UART ⬜ Reserved

Pinouts

- **Power Pins**
  - There are four power pins (**VIN** pin and three **3.3V** pins)
  - **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the **VIN** pin
  - **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components
- **GND**
  - are the ground pins of NodeMCU/ESP8266
- **I2C Pins (Inter-Integrated Circuit)**
  - are used to connect I2C sensors and peripherals
  - Both I2C Master and I2C Slave are supported
  - I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum
  - It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device
- **GPIO Pins**
  - NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically
  - Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance
  - When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts

Pinouts

- **ADC Channel**
  - The NodeMCU is embedded with a 10-bit precision SAR ADC
  - The two functions can be implemented using ADC
  - Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time
- **UART Pins (Universe Asynchronous Receiver – Transmitter)**
  - NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps
  - UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.
- **SPI Pins (Serial Peripheral Interface)**
  - NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes.
  - These SPIs also support the following general-purpose SPI features:
    - 4 timing modes of the SPI format transfer
    - Up to 80 MHz and the divided clocks of 80 MHz
    - Up to 64-Byte FIFO

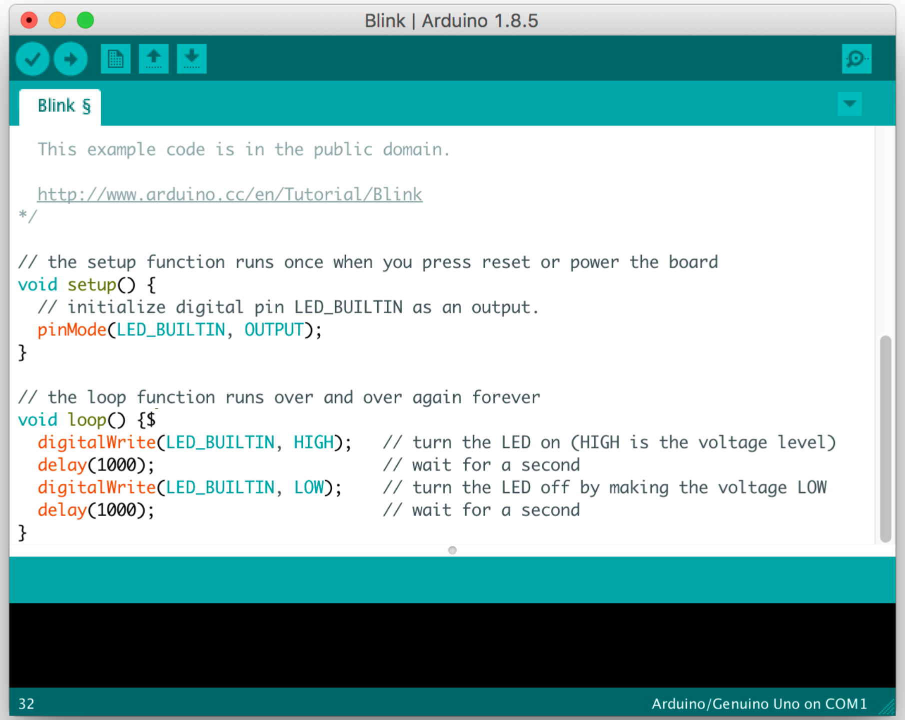
Pinouts

- **SDIO Pins**
  - NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards
  - 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported
- **PWM Pins**
  - The board has 4 channels of Pulse Width Modulation (PWM)
  - The PWM output can be implemented programmatically and used for driving digital motors and LEDs
  - PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz)
- **Control Pins**
  - These pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin
  - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
  - **RST:** RST pin is used to reset the ESP8266 chip
  - **WAKE:** Wake pin is used to wake the chip from deep-sleep

Arduino IDE

Arduino IDE

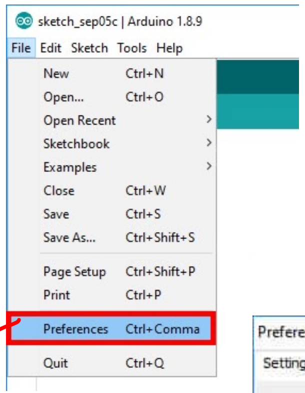
- The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board
- This software can be used with any Arduino board
- Download IDE from
  - https://www.arduino.cc/en/software

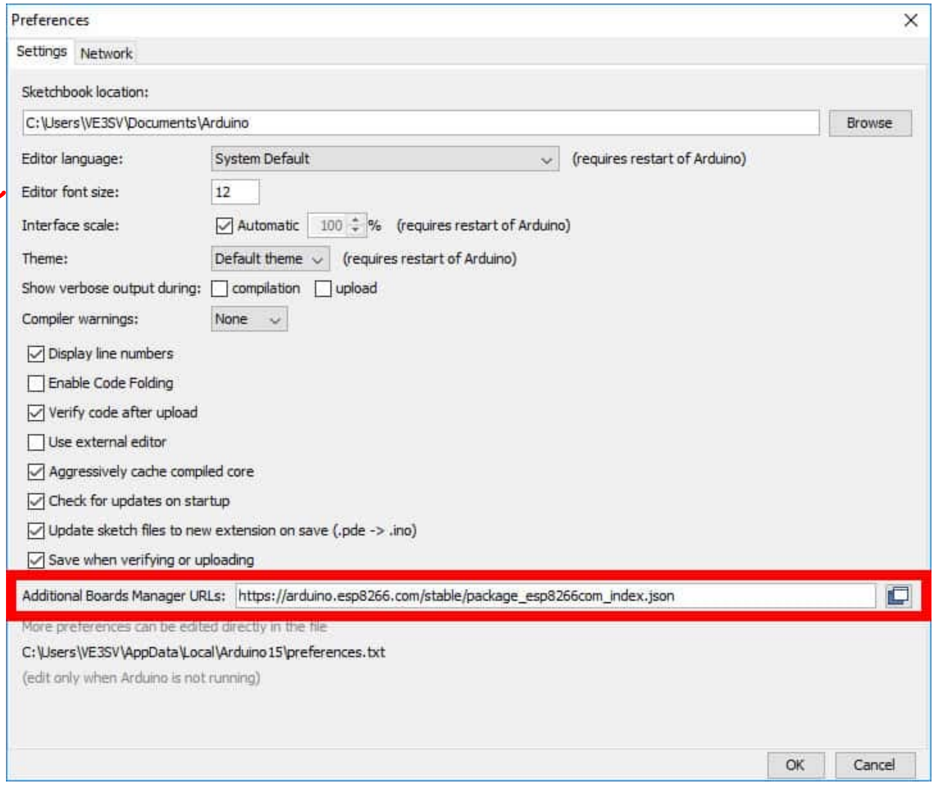


```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {$
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Add NodeMCU Library to the IDE

- The IDE needs to know details about the NodeMCU module
- This information is available in a configuration file which will is added to the IDE
- Select the **File->Preferences**
- This will open up a new window, where we set the IDE to point to the configuration file
- In the field “**Additional Boards Manager URLs:**” enter
  - **http://arduino.esp8266.com/stable/package_esp8266com_index.json**
- Select the **OK** button at the bottom of the window when the link address is entered





Sketchbook location: C:\Users\VE3SV\Documents\Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

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

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

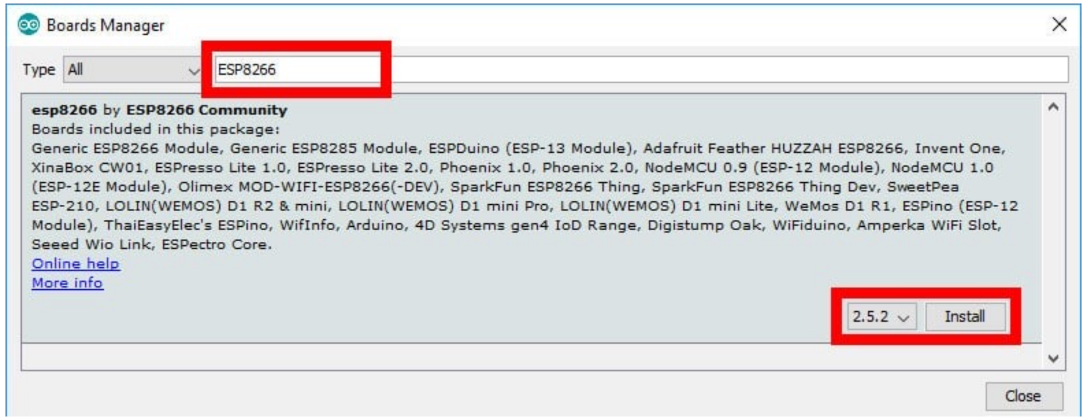
☑ Display line numbers
☐ Enable Code Folding
☑ Verify code after upload
☐ Use external editor
☑ Aggressively cache compiled core
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

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

More preferences can be edited directly in the file
C:\Users\VE3SV\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Selecting the NodeMCU Board Driver

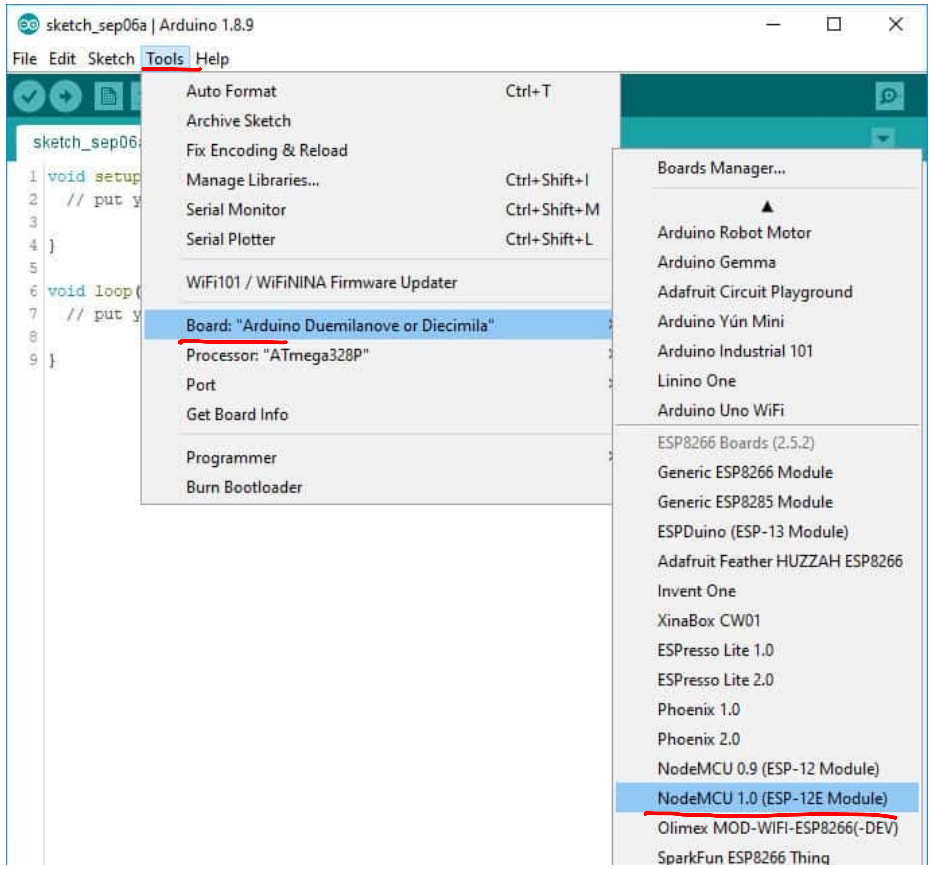
- Navigate to the Boards Manager from the
    - **Tools Board** -> **Boards Manager** menu option
- A new options windows will appear
- From the **Boards Manager** window, enter **esp8266** in the search bar
- This will display the new ESP8266 driver as supplied by the ESP8266 Community. At the bottom of the options box, select the version to be setup (select the most recent version)
- Now select **Install** to complete the installation of the driver which will take a few seconds to download and install
- Select **Close** to complete the installation



esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.

Selecting Board Version

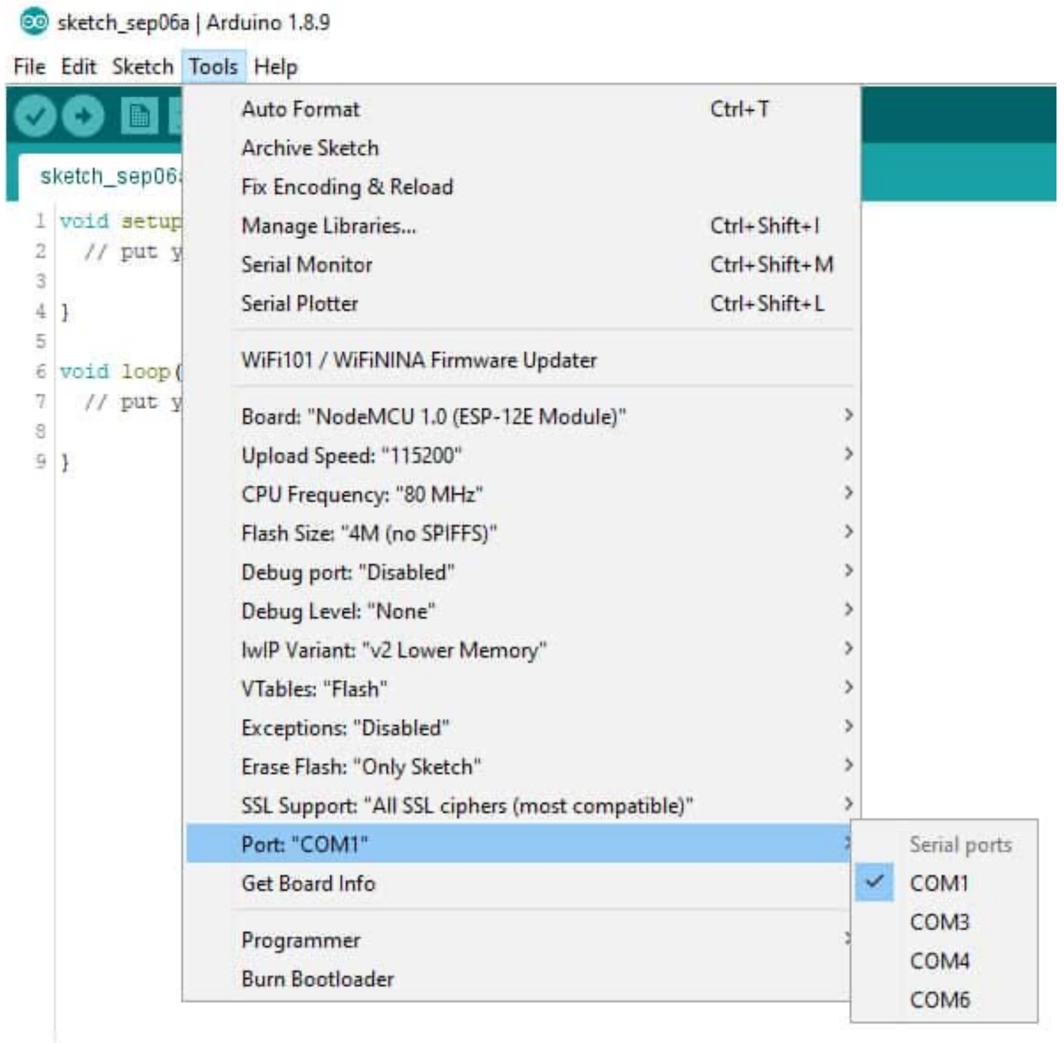
- Now from the **Tools -> Board** menu, the dropdown list will contain many different hardware board
- Scroll down and select the **NodeMCU 1.0 (ESP-12E Module)**



```
void setup
  // put y
}

void loop(
  // put y

}
```

Defining The Communications Port

- The communications port must be defined to allow the Arduino IDE to communicate with the NodeMCU. Connect your PC by a USB cable to the NodeMCU.
- After connection, the PC will provide power to the NodeMCU. The communication port used will vary by system.
- To determine which communications port has been selected on your system, select the **Tools -> Port** menu option.
- The options list will present the available COM ports. For our example, our NodeMCU added **COM6** and will be selected by clicking on **COM6**.
- Shown below the **Board** selection, are several NodeMCU settings. These include **CPU Frequency**, **Flash Size**, and also **Upload Speed**. If **Upload Speed** is not set to **115200**, select the speed of **115200** from the options from that item. The speed of 115200 is the optimum transfer rate which is used to transfer data to and from the NodeMCU.

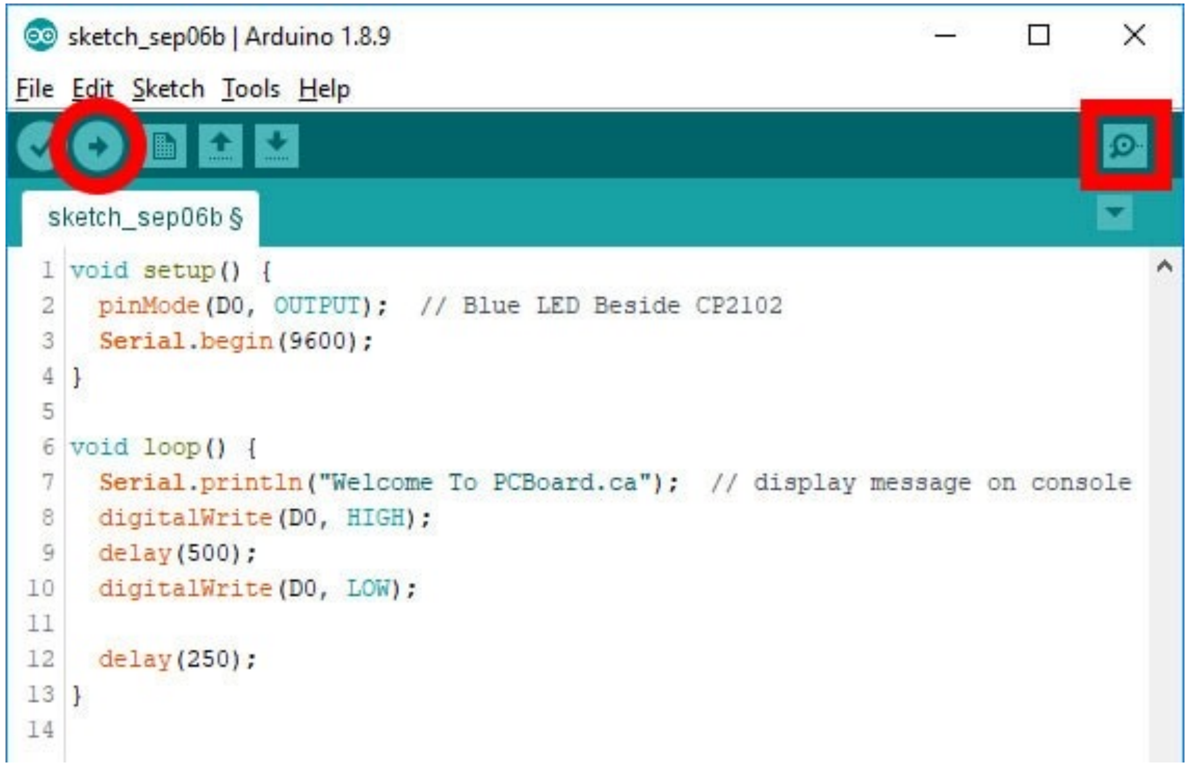

sketch_sep06a | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_sep06
void setup
// put y
}
void loop(
// put y
}
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM1
COM3
COM4
COM6

Board: "NodeMCU 1.0 (ESP-12E Module)"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4M (no SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
IwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
Port: "COM1"

Code To Test NodeMCU Functionality

- Copy the code below and paste it into a blank sketch in the IDE

```
void setup() {
        pinMode(D0, OUTPUT);
        Serial.begin(9600);
}

void loop() {
        Serial.println("Welcome To PCBoard.ca");
        digitalWrite(D0, HIGH);
        delay(500);
        digitalWrite(D0, LOW);
        delay(250);
}
```



```
void setup() {
  pinMode(D0, OUTPUT);  // Blue LED Beside CP2102
  Serial.begin(9600);
}

void loop() {
  Serial.println("Welcome To PCBoard.ca");  // display message on console
  digitalWrite(D0, HIGH);
  delay(500);
  digitalWrite(D0, LOW);

  delay(250);
}

```

ESP8266 Libraries in Arduino IDE

- The collection of procedures, subprograms, and modules that used frequently can be saved as a Library files
- Library file particular are routines and function that used in sketches frequently so we don't have to re-invent the wheel
- There are some libraries that are preinstalled in Arduino IDE
  - ✓ **Bridge**
  - ✓ **Ethernet**
  - ✓ **EEPROM**
  - **GSM**
  - **Keyboard**
  - **LiquidCrystal**
  - **Mouse**
  - **Robot Control**
  - **Robot Motor**
  - **SD**
  - **Servo**
  - **Stepper**
  - **TFT**
  - **WiFi**

Libraries available to install manually

- There is a total of 2233 libraries are registered in the Arduino Library Manager and increasing
  - **Adafruit ESP8266: a** collection of example codes for ESP8266 chipset.
  - **ArduinoJson:** Easy to used JSON examples.
  - **ArduinoOTA:** To program the ESP8266 and other Wifi microcontrollers over the air.
  - **Blynk:** To interface microcontroller with Blynk Application.
  - **DHT:** Sample codes for DHT11 and DHT22 sensor.
  - **ESP8266:** Collection of basic codes for ESP8266 microcontroller.
  - **ESP8266SDUpdater:** To interface SD card with ESP8266.
  - **ESP8266WebServer:** To create a web server with esp8266.
  - **EspSoftwareSerial:** Arduino software serial implementation for ESP8266/ESP32.
  - **FirebaseArduino:** To interface ESP devices with Google Firebase.
  - **OLED:** Example code to interface OLED displays.
  - **PubSubClient:** MQTT messaging library for ESP8266.
  - **Ping:** To generate ping to other remote devices using ESP8266.
  - **RFID:** To interface RFID reader with ESP8266.
  - **SD (esp8266):** To interface SD card module with ESP8266.
  - **Servo (esp8266):** To interface Servo with ESP8266.
  - **SPI:** To use Serial Peripheral Interface using ESP8266.
  - **ThingSpeak:** To interface Wifi devices with ThingSpeak server to send and read data.
  - **WiFi Picker:** To provide a list of Wifi access points.
  - **WiFi101OTA:** To program the ESP devices using Wifi.
  - **YouTubeApi:** Library to use YouTube APIs.