**COOK BOOK**

**SETTING UP NODEMCU**

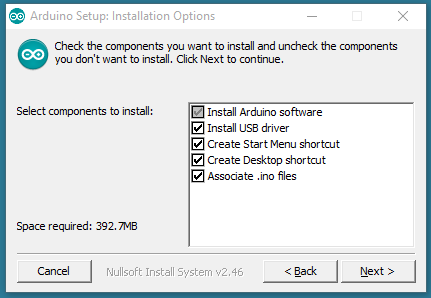
**Arduino IDE**

### Download the Arduino Software (IDE)

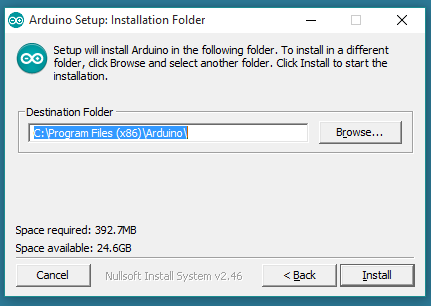
Get the latest version from the <https://www.arduino.cc/en/Main/Software>. You can choose between the Installer (.exe) and the Zip packages.

When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.

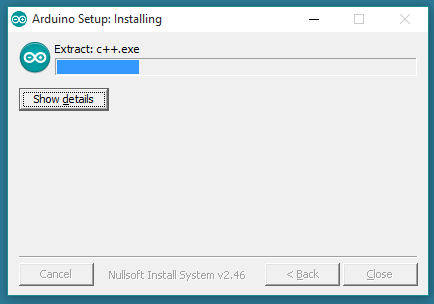
Choose the components to be installed.



Choose the installation directory (we suggest to keep the default one)



The process will extract and install all the required files to execute properly the Arduino Software (IDE)

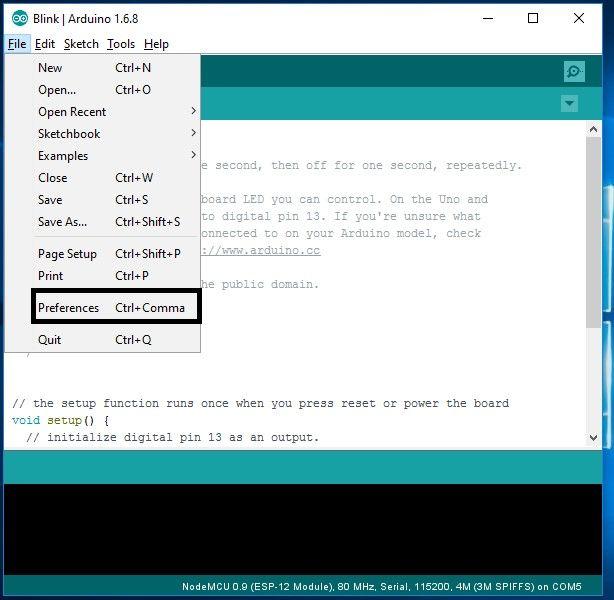


Once the installation is finished just hit the finish button.

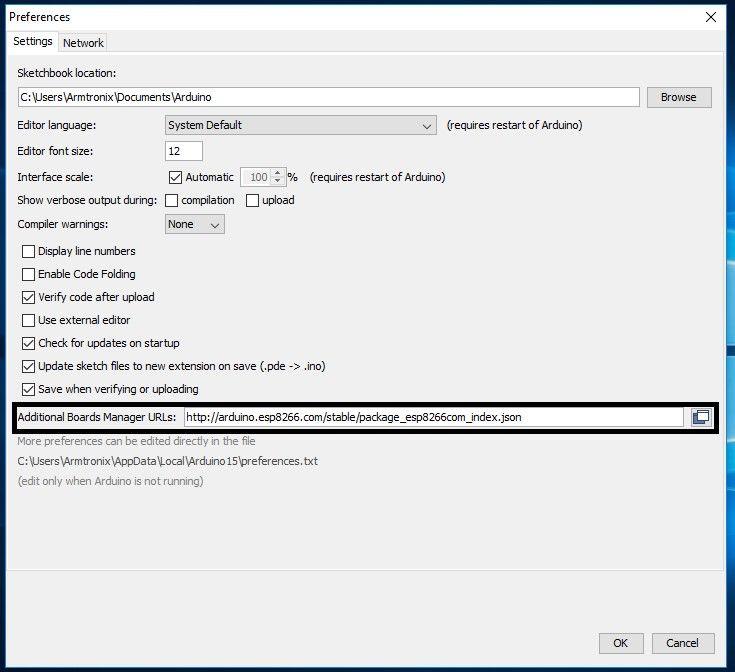
# **Steps to Setup Arduino IDE for NODEMCU ESP8266**

Install Arduino IDE software from the link http://www.arduino.cc/en/main/software.

Open File ->Preferences

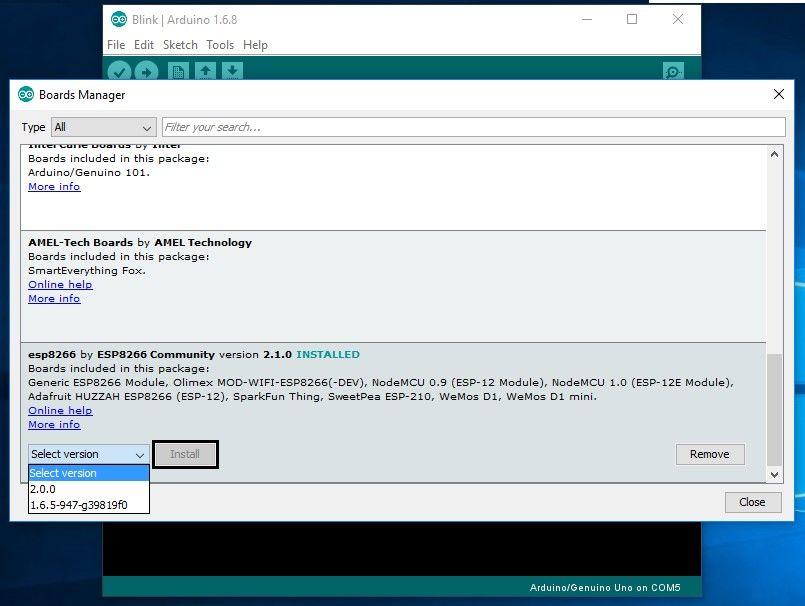


Adding ESP8266 Board Manager



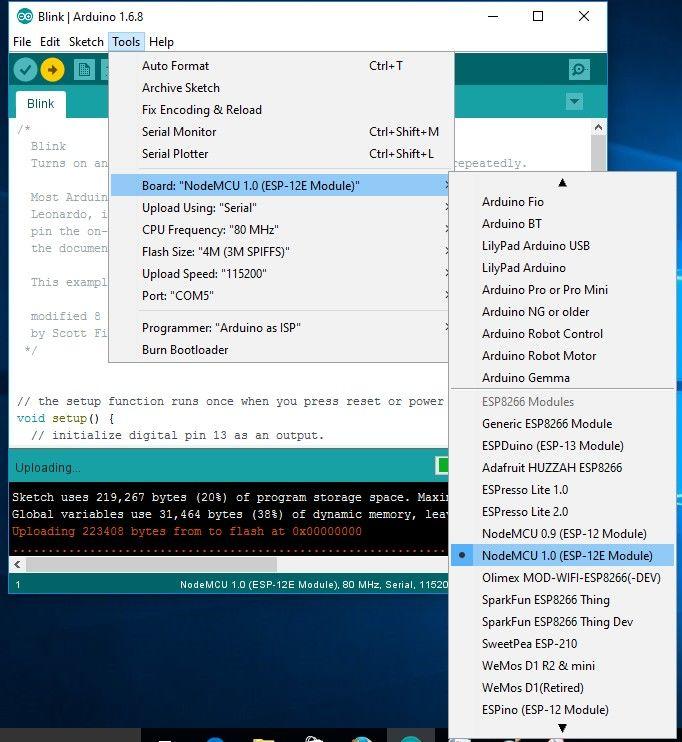
In the Additional Boards Manager enter below URL.

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



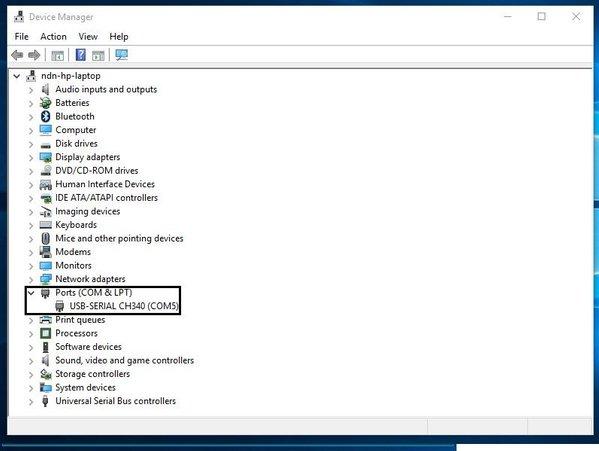
The Boards Manager window opens, scroll the window page to bottom till you see the module with the name ESP8266. Once we get it, select that module and select version and click on the Install button. When it is installed it shows Installed in the module as shown in the figure and then close the window.

1. Selecting ESP8266 Arduino Board



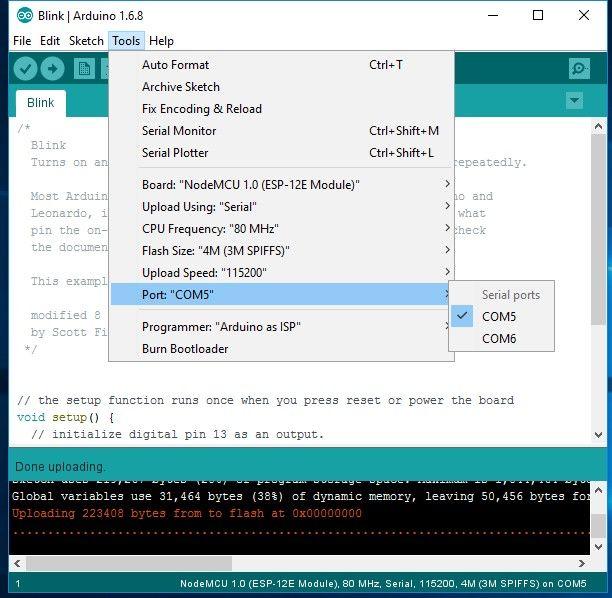
To run the esp8266 with Arduino we have to select the **Board: “Arduino/Genuino Uno”** and then change it to **NodeMCU 1.0 (ESP-12E Module)** or other esp8266 modules. This can be done by scrolling down, as shown in the figure.

5.Connecting ESP8266 to the PC



 Let’s connect the ESP8266 module to your computer through USB cable as shown in the figure. When module is connected to the USB, COM port is detected eg: here COM5 is shown in the figure.

6. Selecting COM Port



7.Upload the program to NodeMCU and check the output on the Serial Monitor.

**Connecting Temperature Sensor to the Node MCU**

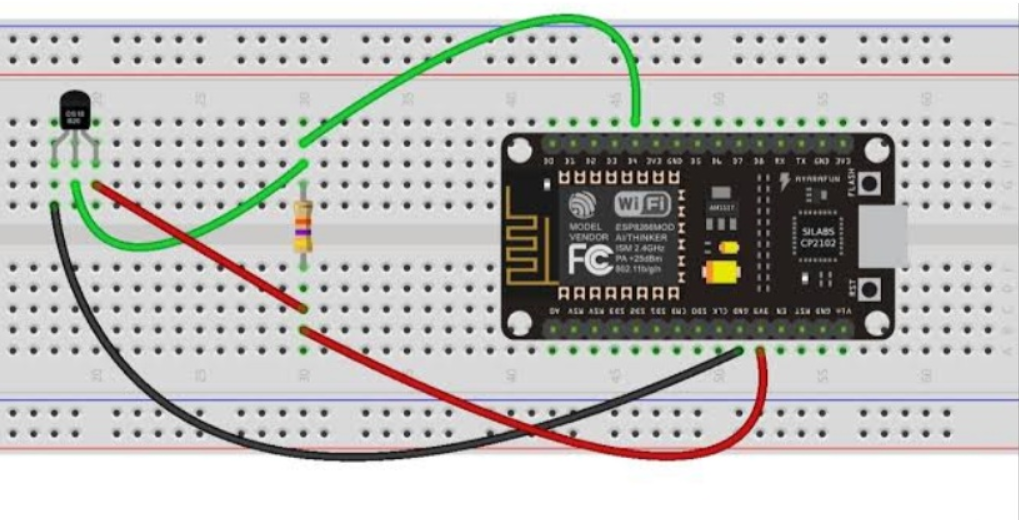
DS18B20 is used for measuring the soil temperature. It is one wire interface sensor, very common in PCB embedded electrical circuits. Its unique one wire protocol requires only one port pin for communication and needs no other external components to work. It is waterproof sensor hence it can be used in humid and wet environmental conditions. It can operate on 3.3v and 5.5v. Add Dallas Library For Temperature Sensor.

This sensor has 3 pins.

Vcc - 5V

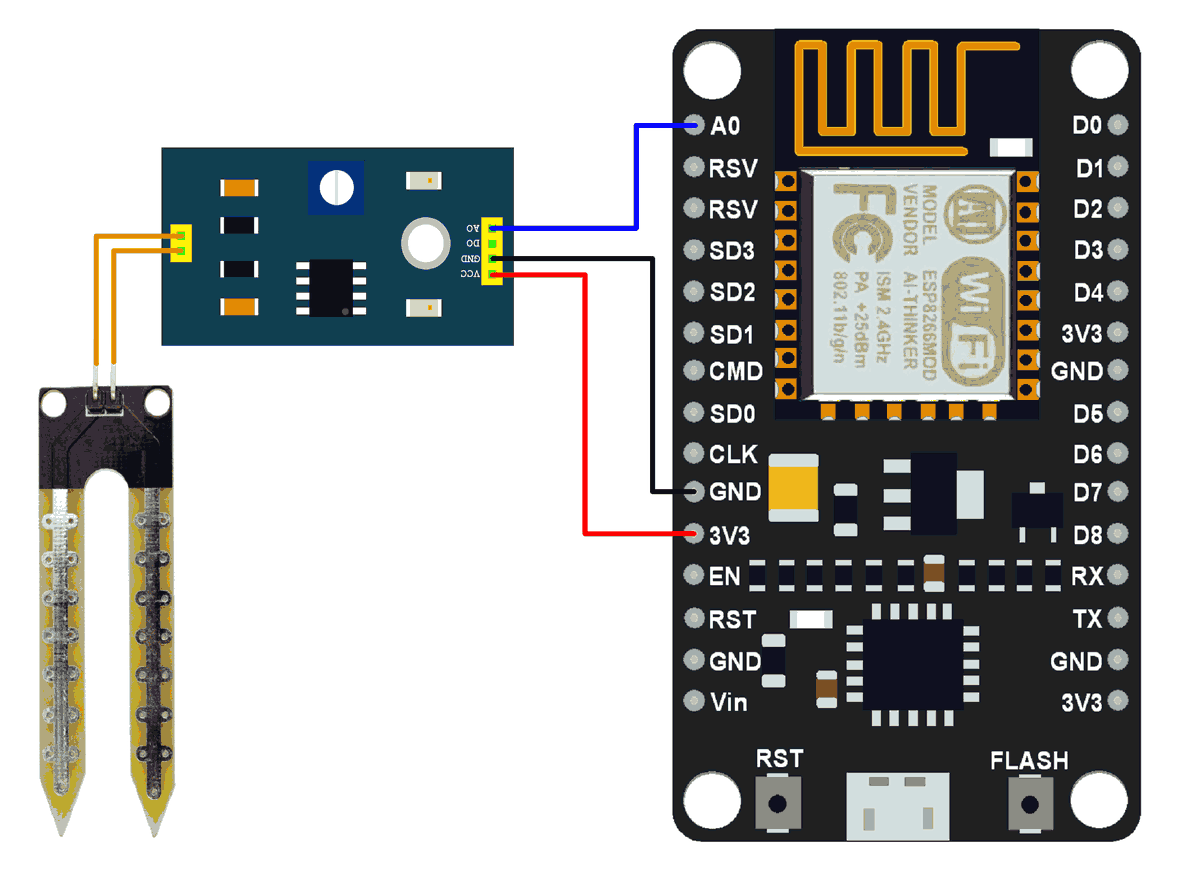
Gnd - Gnd

Data



**Connecting Moisture Sensor to the Node MCU**

KG003 soil moisture sensor is used to measure the soil moisture. This is a simple water sensor that can be used to detect soil moisture. Module output is high level when the soil moisture deficit or output is low. The sensitivity can be adjusted by using the digital potentiometer. The output is available in both analog and digital format. In the project analog interfacing is used for accurate output. The soil moisture sensor consists of two probes. The two probes allow the current to pass through the soil and then it gets the resistance value to measure the moisture value. When there is more water, the soil will conduct more electricity which means that there will be less resistance. Therefore, the moisture level will be higher. Dry soil conducts electricity poorly, so when there will be less water, the soil will conduct less electricity which means that there will be more resistance. Therefore, the moisture level will be lower. The sensor has 4 pins. First two pins are Vcc and GND. The next two pins are output pins.A0 pin is giving analog output and D0 pin is giving digital output.



**NODEMCU FINAL SCRIPT**

#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

#include <OneWire.h>

#include <DallasTemperature.h>

#define ONE\_WIRE\_BUS 2 // DS18B20 on NodeMCU pin D4

OneWire oneWire(ONE\_WIRE\_BUS);

#include<string.h>

#include<stdlib.h>

DallasTemperature DS18B20(&oneWire);

const char\* nmcu\_id="node\_1";

const char\* ssid = "123";//wifi name

const char\* password = "lasya123";//password

WiFiUDP Udp;

unsigned int localUdpPort = 1885;  // local port to listen on

char incomingPacket[255];  // buffer for incoming packets

char  replyPacket[] = "";  // a reply string to send back

char status\_reply[]="alive";

void setup()

{

  Serial.begin(115200);

  Serial.println();

  Serial.printf("Connecting to %s ", ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED)

  {

    delay(500);

    Serial.print(".");

  }

  Serial.println(" connected");

  Udp.begin(localUdpPort);

  Serial.printf("Now listening at IP %s, UDP port %d\n", WiFi.localIP().toString().c\_str(), localUdpPort);

}

void loop()

{

  int packetSize = Udp.parsePacket();

  if (packetSize)

  {

    // receive incoming UDP packets

    Serial.printf("Received %d bytes from %s, port %d\n", packetSize, Udp.remoteIP().toString().c\_str(), Udp.remotePort());

    int len = Udp.read(incomingPacket, 255);

    if (len > 0)

    {

      incomingPacket[len] = 0;

    }

    Serial.printf("UDP packet contents: %s\n", incomingPacket);

    if (incomingPacket=="status")

    {

      Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

      Udp.write(status\_reply);

      Udp.endPacket();

    }

    else{

     float temp\_0;

  float temp\_1;

  DS18B20.requestTemperatures();

  temp\_0 = DS18B20.getTempCByIndex(0); // Sensor 0 will capture Temp in Celcius

  temp\_1 = DS18B20.getTempFByIndex(0); // Sensor 0 will capture Temp in Fahrenheit

  Serial.print("Temp\_0: ");

  Serial.print(temp\_0);

  Serial.print(" oC . Temp\_1: ");

  Serial.print(temp\_1);

  Serial.println(" oF ");

  delay(1000);

  // read the input on analog pin 0:

  float moist = analogRead(A0);

  // print out the value you read:

  Serial.println(moist);

    delay(1000);

     sprintf(replyPacket,"%f,%f,%f",temp\_0,temp\_1,moist);

     Serial.println(replyPacket);

     // send back a reply, to the IP address and port we got the packet from

     Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

     Udp.write(replyPacket);

     Udp.endPacket();

    }

  }

}

NOTE: Before running the above scripts create a csv file containing the ip addresses of all the nodemcus installed in the farm with the name: “nmcu\_ip.csv”. Create another csv file with the name “ping\_ip\_list.csv”.