INTERNET OF THINGS

PRACTICAL HANDBOOK

RAFIZAH BINTI AB RAHMAN
RAFIDAH BINTI AB RAHMAN

INTERNET OF THINGS PRACTICAL HANDBOOK

RAFIZAH BINTI AB RAHMAN | RAFIDAH BINTI AB RAHMAN

ARFIDZ VENTURES

INTERNET OF THINGS HANDBOOK

FIRST PUBLISHED 2023

Published by
ARFIDZ VENTURES (CA0289061-X)
No. 9, Jalan Setia Jasa 1,
Bandar Satelit Muadzam Shah,
26700 Muadzam Shah, Pahang.
arfidz.ventures@gmail.com

Table of Contents

INTRODUCTION

1. Internet of Things

The Internet of Things refers to the ever-growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other Internet-enabled devices and systems. The Internet of Things extends internet connectivity beyond traditional devices like desktop and laptop computers, smartphones and tablets to a diverse range of devices and everyday things that utilize embedded technology to communicate and interact with the external environment, all via the Internet.

Examples of objects that can fall into the scope of Internet of Things include connected security systems, thermostats, cars, electronic appliances, lights in household and commercial environments, alarm clocks, speaker systems, vending machines and more. Businesses can leverage IoT applications to automate safety tasks (for example, notify authorities when a fire extinguisher in the building is blocked) to performing real-world A/B testing using networked cameras and sensors to detect how customers engage with products.

When building Internet of Things (IoT) related applications, a main problem the developer face is the communication between the micro-controller and the software application. Direct communication methods like Bluetooth / NFC can be implemented but for giving more remote access for the application we need to use internet to communicate between the two devices.

GETTING STARTED

This guide will be a completely hands-on step by step guide addressing 5 components that will then be integrated to allow communications with each other.

- Arduino IDE Setup
- NodeMCU / Hardware Setup
- Arduino Programming
- ThingSpeak Platform
- IFTTT Platform
- Blynk 2.0

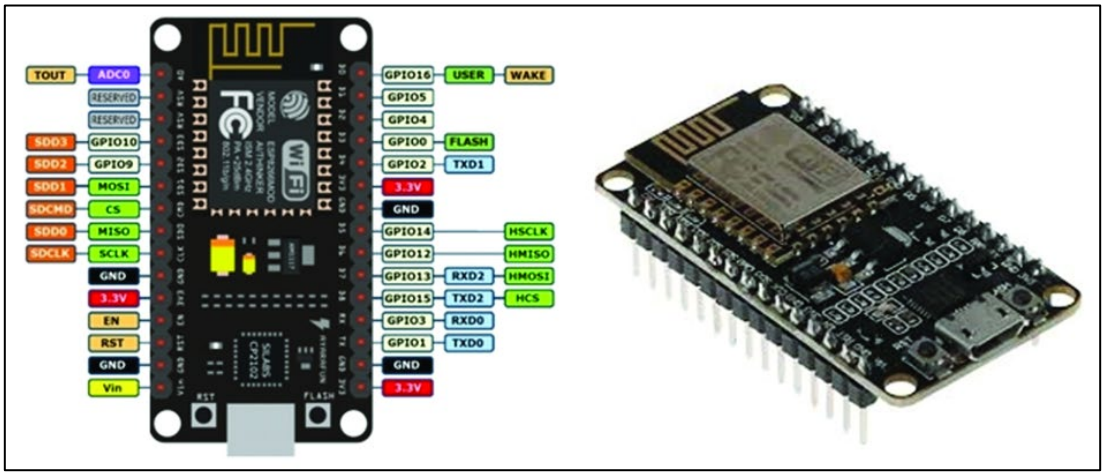
You will need these components:

1. NodeMCU Dev Board
2. Solderless Breadboard
3. 220-ohm resistor
4. Push button
5. LEDs
6. Micro USB cable
7. Jumper wires
8. DHT 11 Module

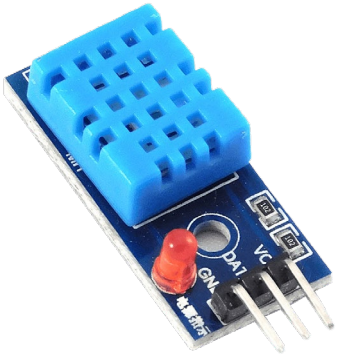
HARDWARE INTRODUCTION

NodeMCU Dev Board is based on widely explored esp8266 System on Chip from Expressif and combined features of WIFI access point and station + microcontroller and uses simple LUA based programming language. ESP8266 NodeMCU offers:

- Arduino-like hardware IO
- Event-driven API for network applications
- 10 GPIOs D0-D10, PWM functionality, IIC and SPI communication, 1-Wire and ADC A0 etc. all in one board
- Wi-Fi networking (can be uses as access point and/or station, host a webserver), connect to internet to fetch or upload data.

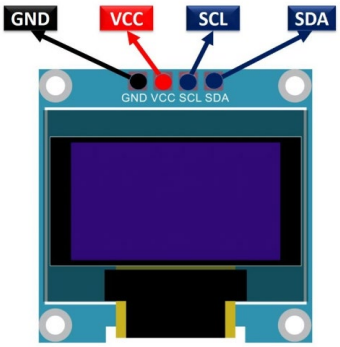


The DHT11 Module is a sensor which measures relative humidity and temperature sensor. It provides a calibrated digital output with a 1-wire protocol. This sensor is inexpensive and affordable. DHT sensors are pre-calibrated and can be directly connected with ESP8266 NodeMCU to obtain sensor output reading. They are internally composed of a humidity sensing sensor and a thermistor. These two components measure humidity and temperature.



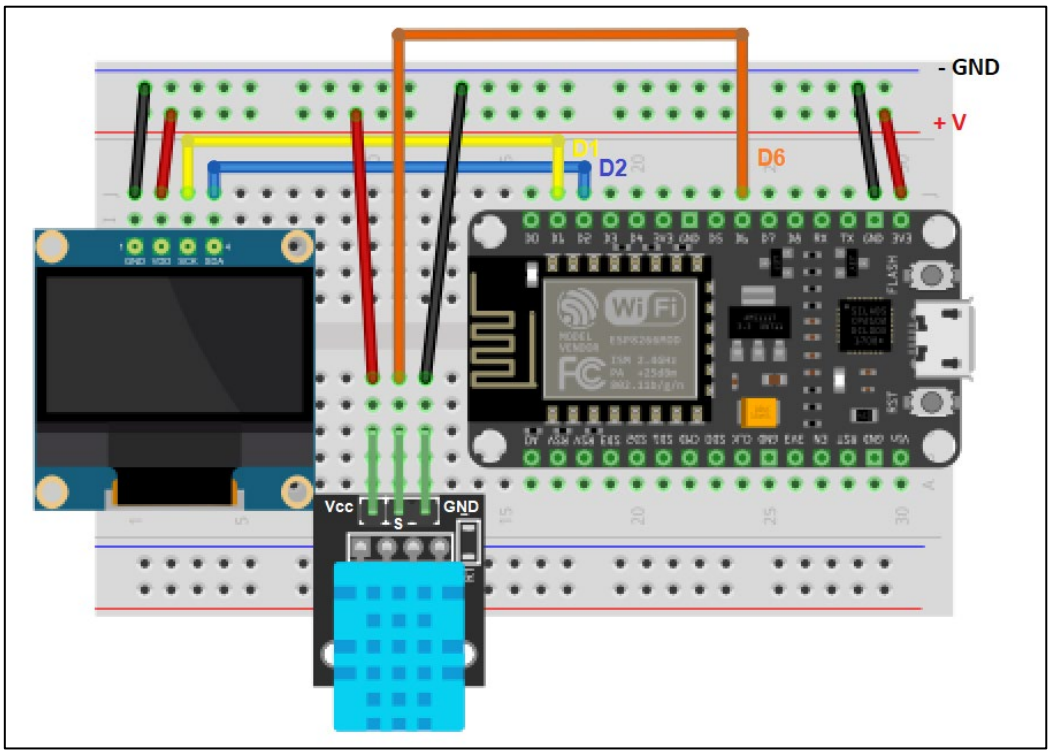
OLED stands for organic light-emitting diode. Its name shows that it is a flat light emitting technology that is developed when two organic thin films are connected in series between two electric conductors. When an electric current is supplied to these conductors then the organic compound is made which emits the bright light. Typically, one conductor is the transparent conductor between these two conductors therefore there is no need for any backlight to emit the light. Therefore, this OLED display has improved image quality, full viewing angle, high brightness, better contrast, wide color range, low power consumption, more efficient and reliable as compared to a simple LCD display.

OLED displays' core part is an SSD1306 controller, which communicates with microcontrollers via the I2C or SPI protocol. OLED performance is faster in SPI connection, although I2C communication is more popular due to the lesser pin used. Although the OLED displays' size, colour and shape can vary, they are generally configured in a similar manner. The OLED display SSD 1306 0.96-inch used for this tutorial has 128×64 pixels and communicates only via I2C protocol with the ESP development boards.



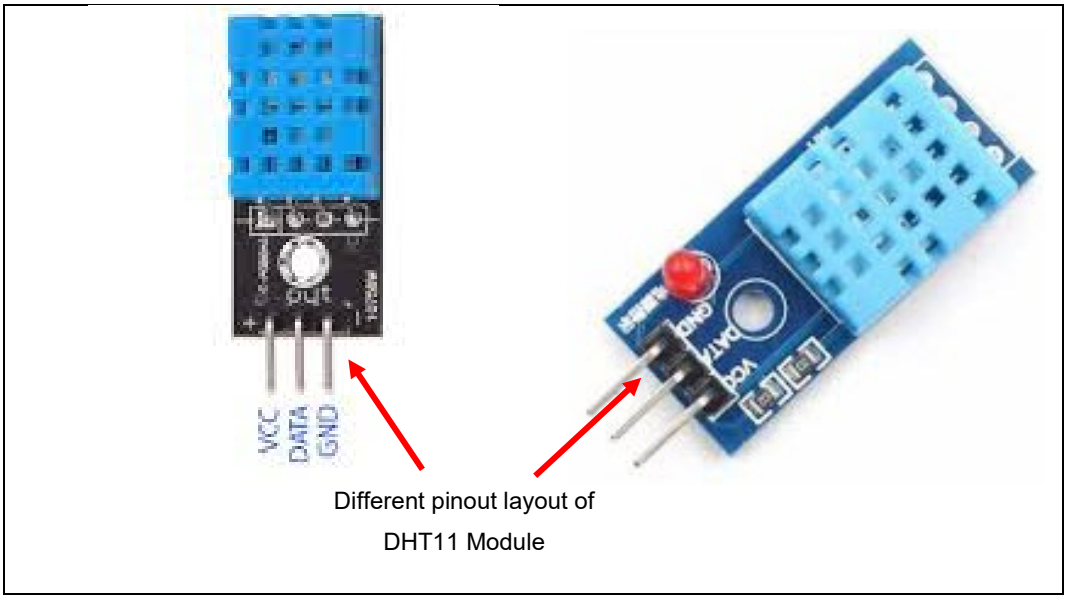
1. Interfacing Sensor, OLED Display and NodeMCU

Setup the wiring of your hardware according to the figure shown below. You can also refer to the pinout table. Always observe the polarity pin of components as some components by different manufacturers have different pin out layout.

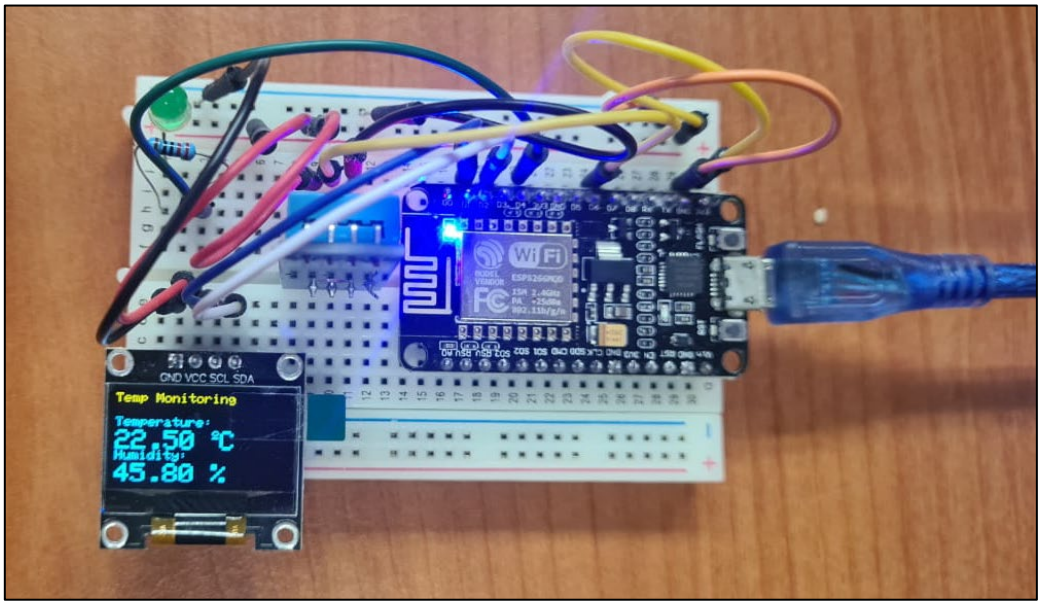


OLED Display	ESP8266 NodeMCU
VCC	VCC=3.3V
GND	GND
SCL	GPIO22 /D1
SDA	GPIO21 /D2

DHT11	ESP8266 NodeMCU
VCC=3.3V	VCC
GND	GND
Data	GPIO12/D6



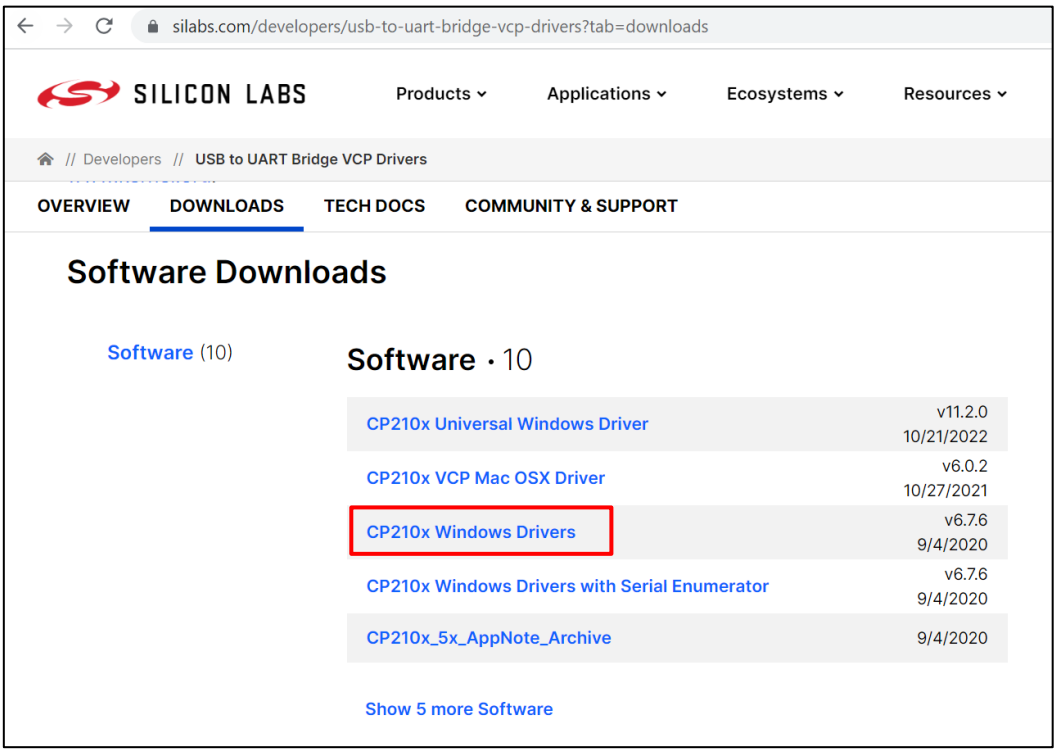
Different pinout layout of
DHT11 Module



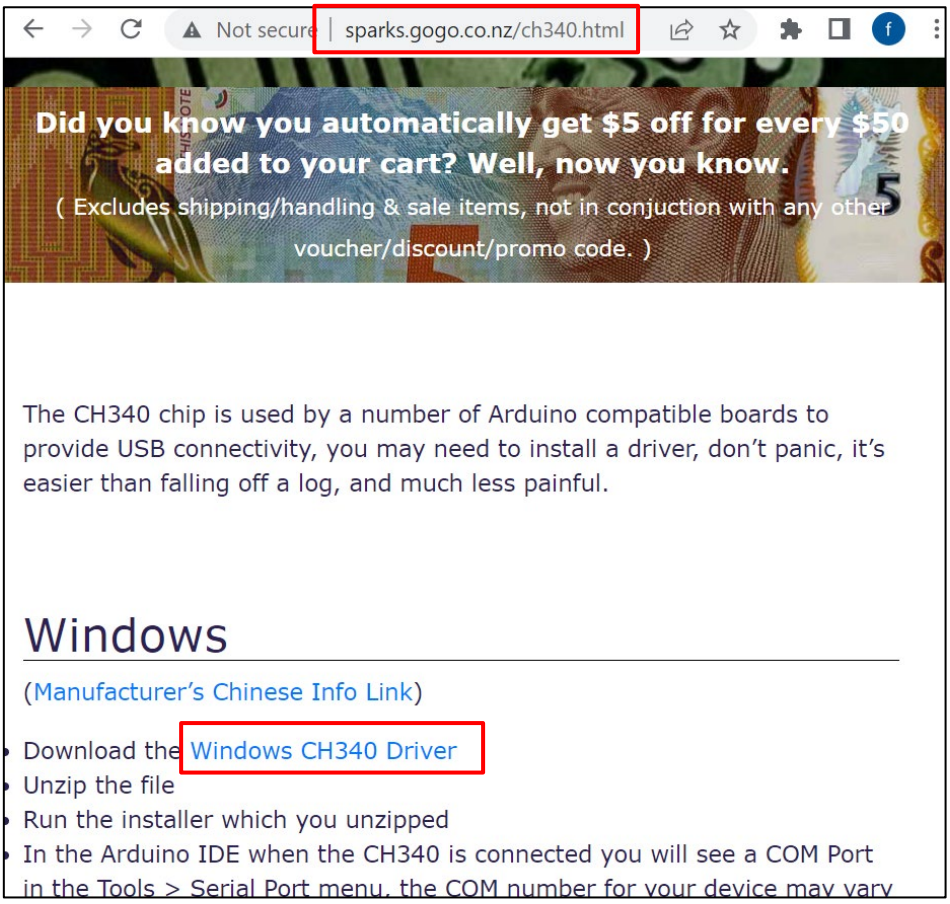
2. NodeMCU ESP8266 Driver Installation

You need to install the driver for the NodeMCU ESP8266 for it to be used on your system. There are two versions of NodeMCU ESP8266 driver, the CP210x USB to UART Bridge VCP driver and the CH340 chip driver. To confirm which module suits your requirement turn over your ESP8266 NodeMCU device and check with instructions for driver details.

1. If you are using the CP210x version, download the driver from ***https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers***. Select the appropriate driver (32-bit or 64-bit) based on your PC Operating System. Extract the files and install the driver (***CP210xVCPInstaller_x64.exe***) for a 64-bit system.



2. If you are using the CH340 chip version, download the driver from ***http://sparks.gogo.co.nz/ch340.html*** and select the appropriate driver (32-bit or 64-bit) based on your PC Operating System. Extract the files and install the driver (***CH34x_Install_Windows_v3_4.exe***).



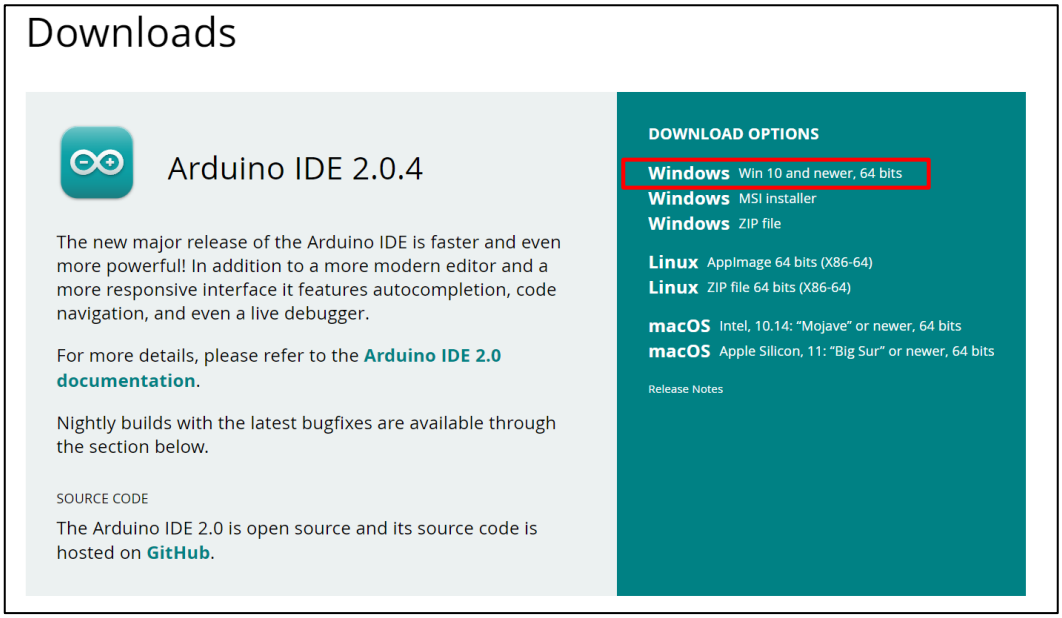
ARDUINO PROGRAMMING

Generally, ESPlorer IDE is used for writing Lua scripts for NodeMCU. It requires user to get familiar with ESPlorer IDE and Lua scripting language. However, there is another way of developing NodeMCU with a well-known IDE i.e., Arduino IDE. This makes things easy for Arduino developers than learning new language and IDE for NodeMCU.

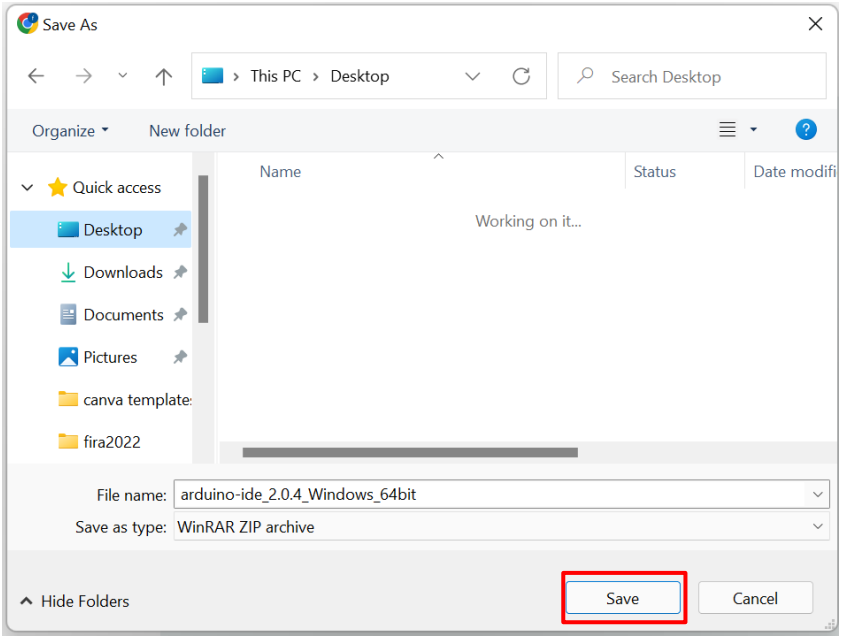
Arduino code is written in C++ with an addition of special methods and functions. C++ is a human-readable programming language. When you create a 'sketch' (the name given to Arduino code files), it is processed and compiled to machine language.

1. Arduino IDE Installation

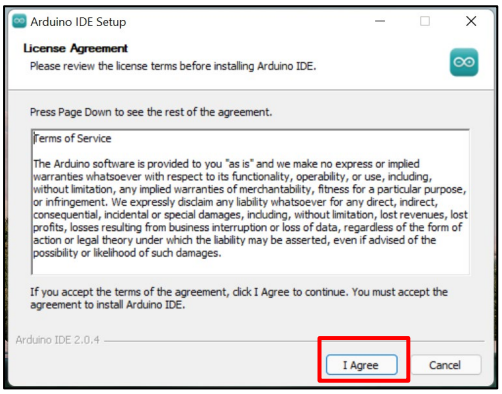
The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where you'll be typing up your code before uploading it to the board you want to program. Arduino code is referred to as **sketches**.

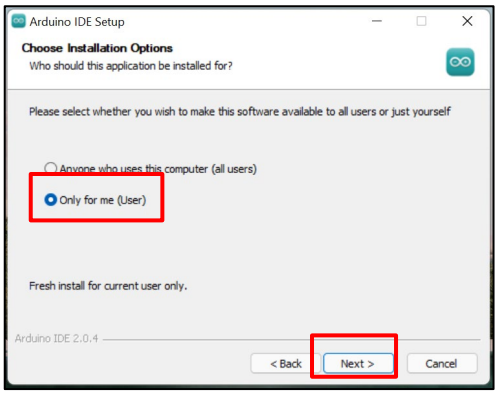


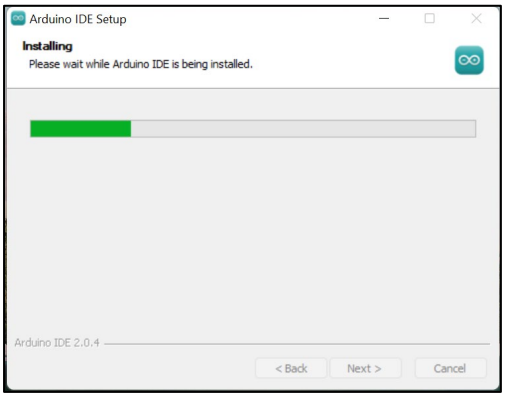
1. Download Arduino IDE 2.0.4 from ***https://www.arduino.cc/en/software***
2. Select ***Windows Win 10 and newer, 64 bits.***

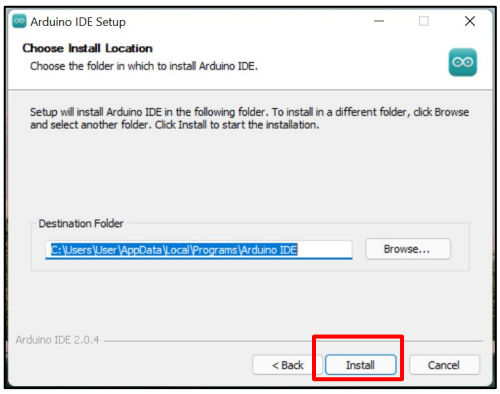


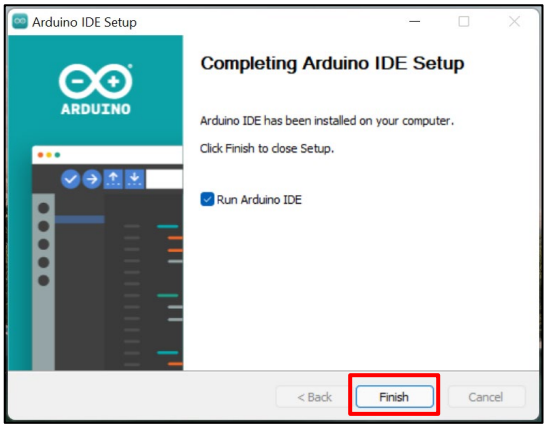
3. Save the installation file to your Desktop or any destination that you prefer.
4. Double click the ***installation*** file to start the installation.
5. Follow the instructions to complete the setup.

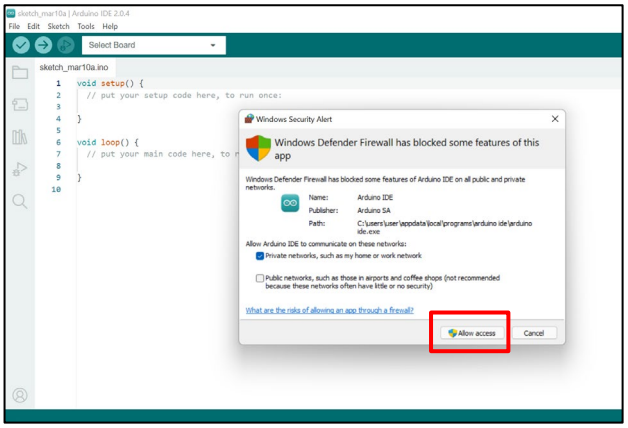






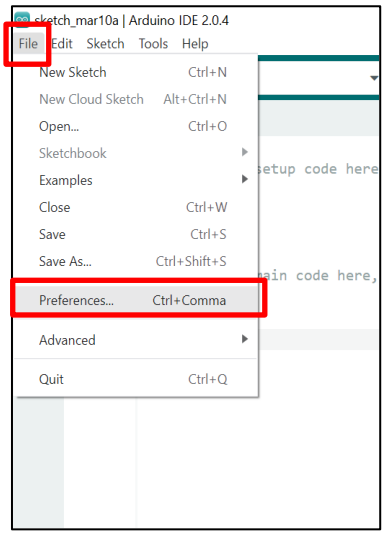


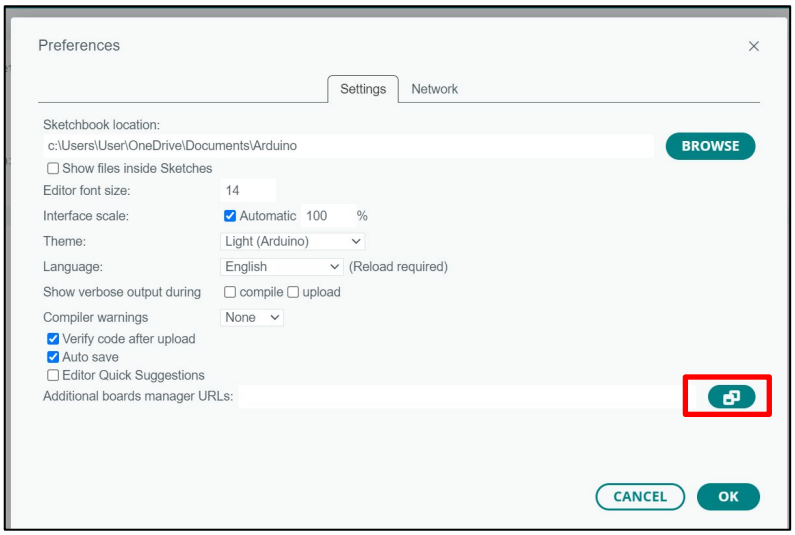


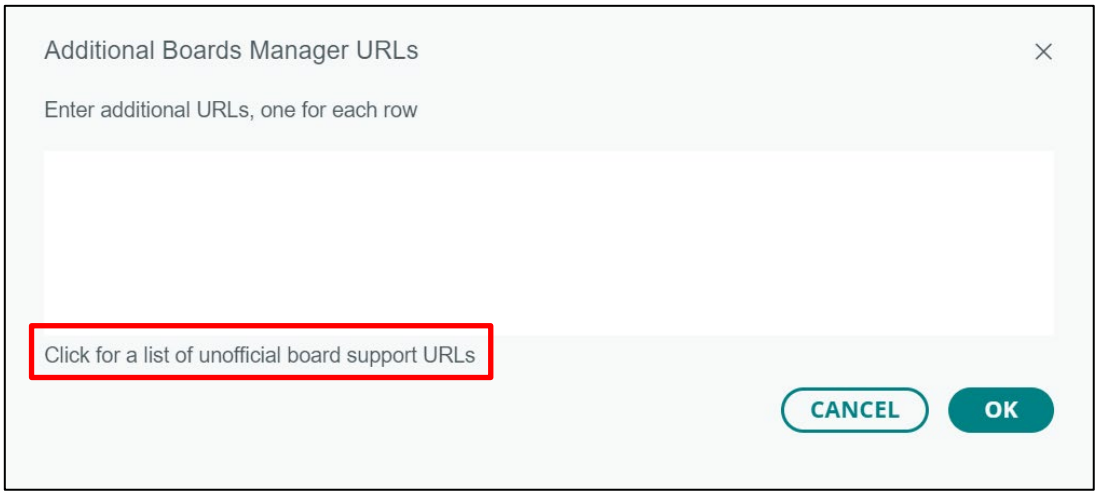


2. Arduino ESP8266 Core Installation

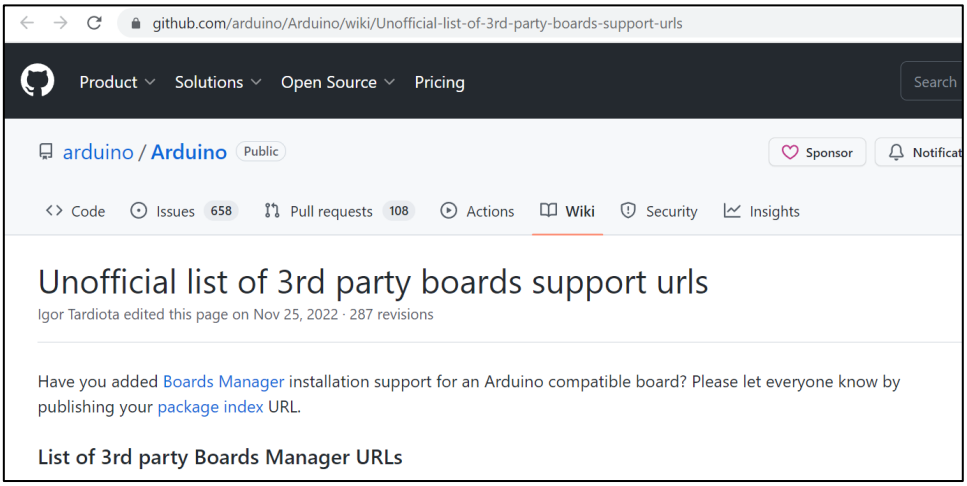
1. Launch Arduino IDE. Click File and open **Preferences** window.
2. Click the ***Additional Board Manager URLs*** icon.
3. Select ***Click for a list of unofficial board supports URLs***

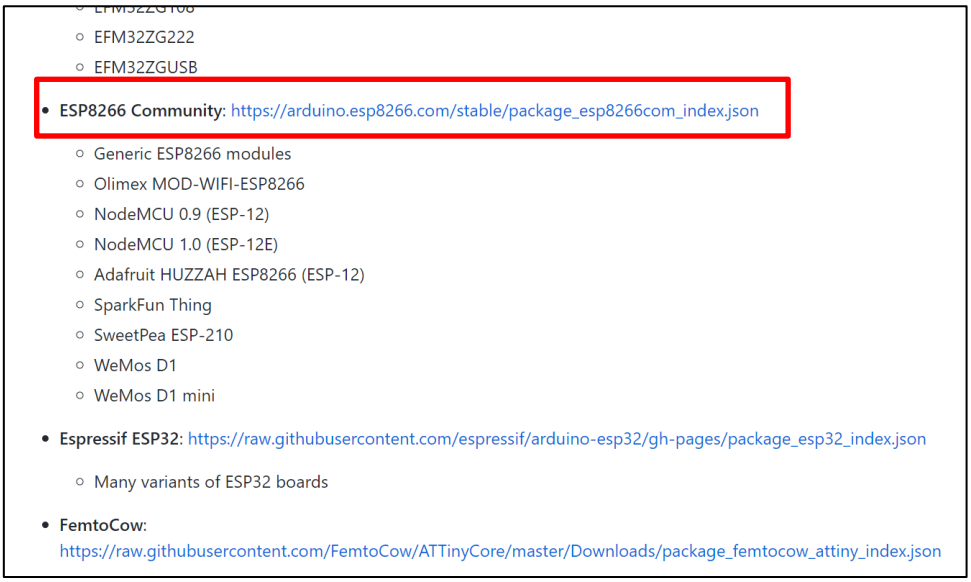




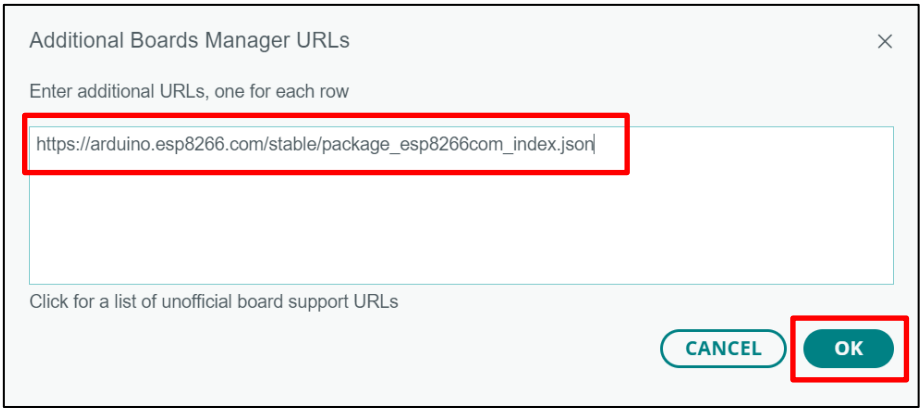


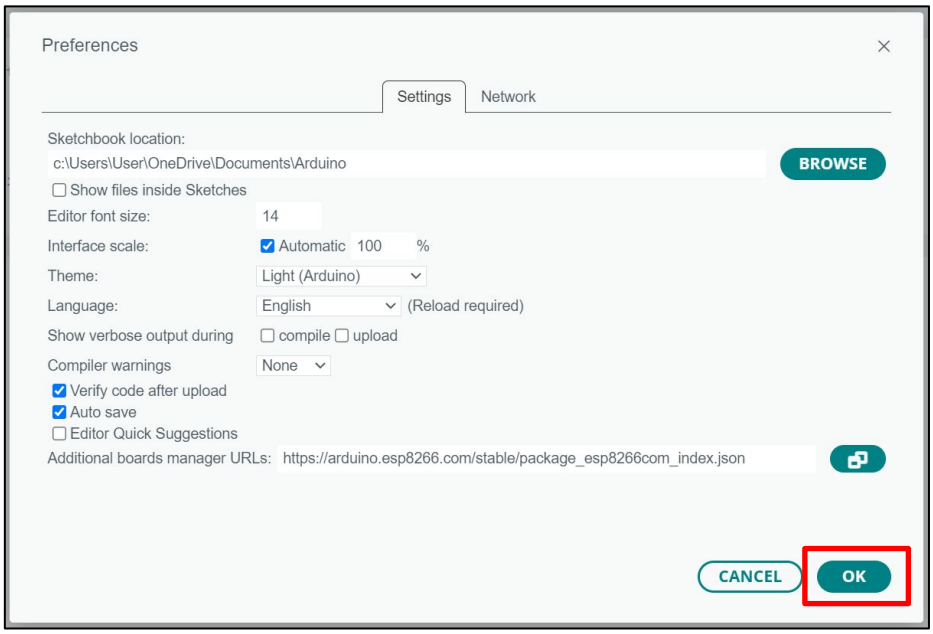
4. Scroll down the github page and search for ESP8266 Community.



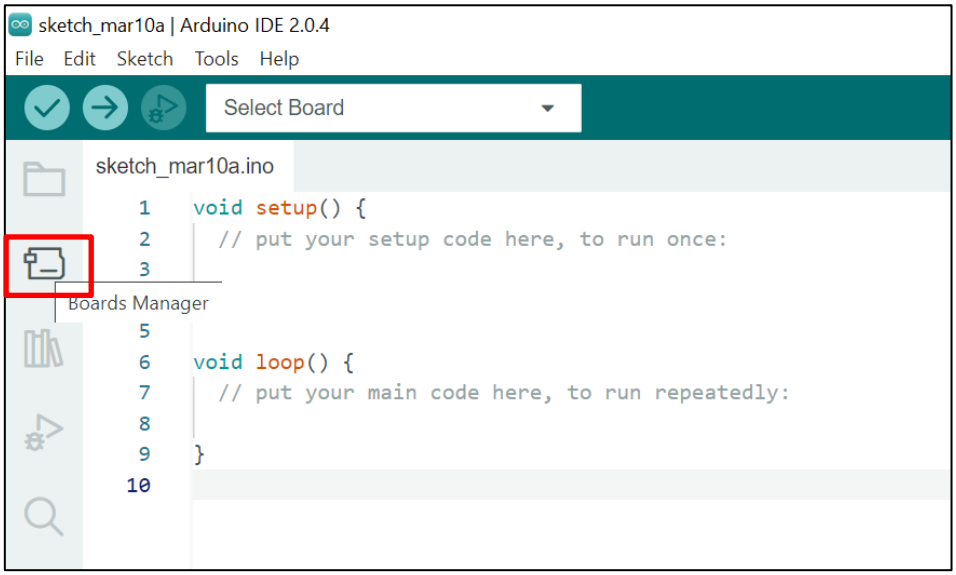


5. Copy the URL and paste into the ***Additional Board Manager URLs*** dialog box. Click **OK**.

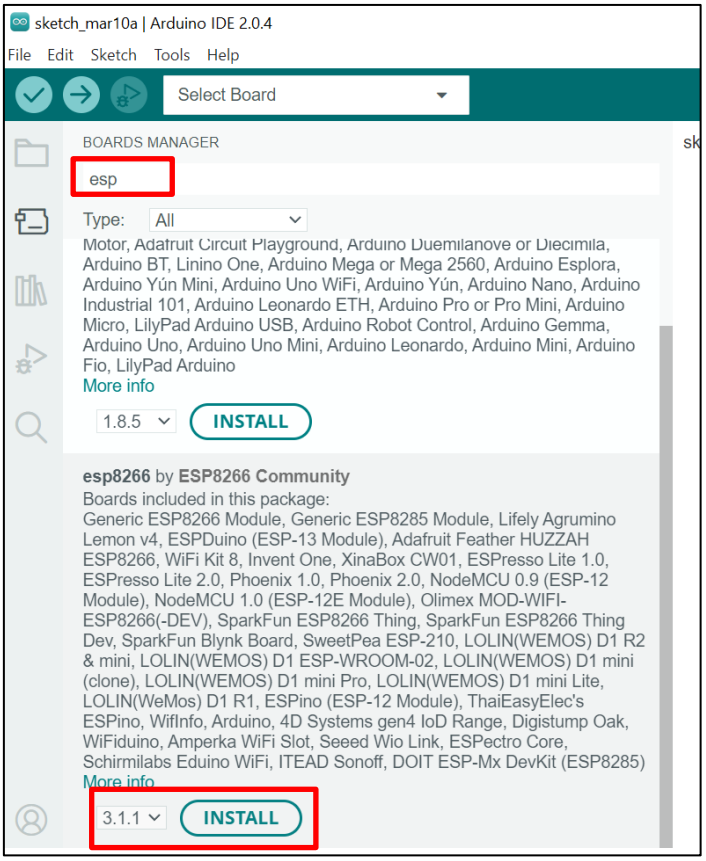




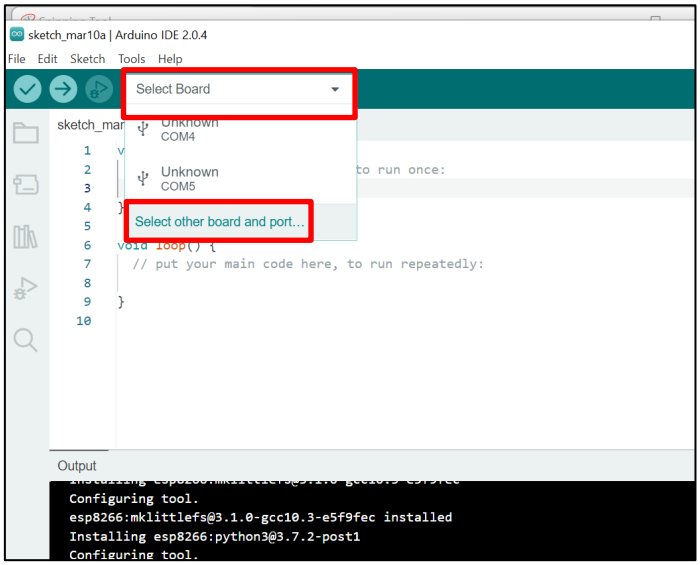
6. Select ***Boards Manager***.



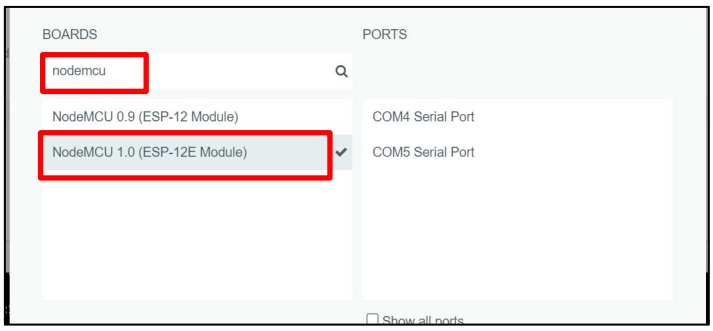
7. Search for ESP8266 in the ***Boards Manager***.
8. Select the version ***3.1.1*** and click ***INSTALL***.



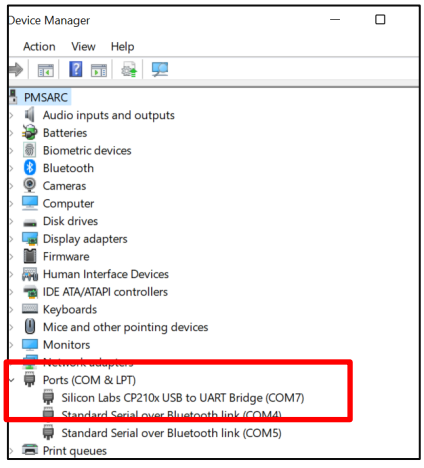
9. Click the ***Select Board*** drop down menu and click ***Select other board and port.***



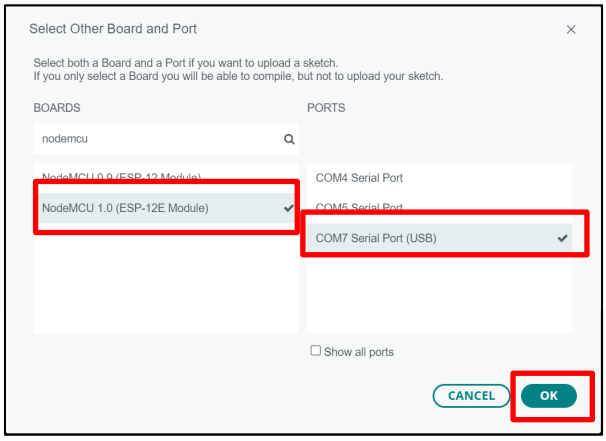
10. Search for NodeMCU and select ***NodeMCU 1.0 (ESP-12E Module)***.

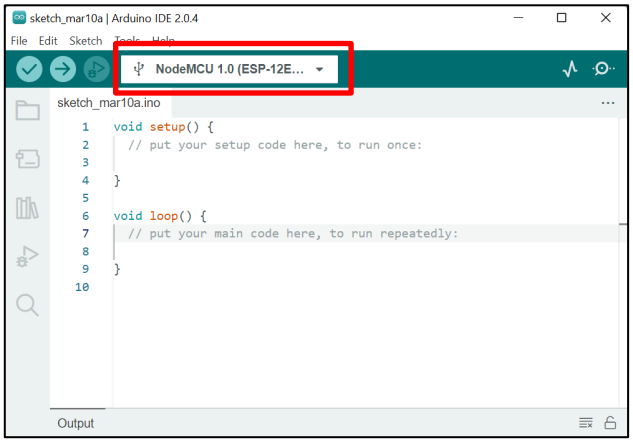


11. To find the COM port number of your board, open ***Device Manager***. Under ***Ports (COM & LPT)*** search for ***Silicon Labs CP210x USB to UART Bridge(COM7).*** Take note of the COM port number and select the corresponding COM port number during board selection.
12. This number may change when you connect your board to different USB ports of your computer.



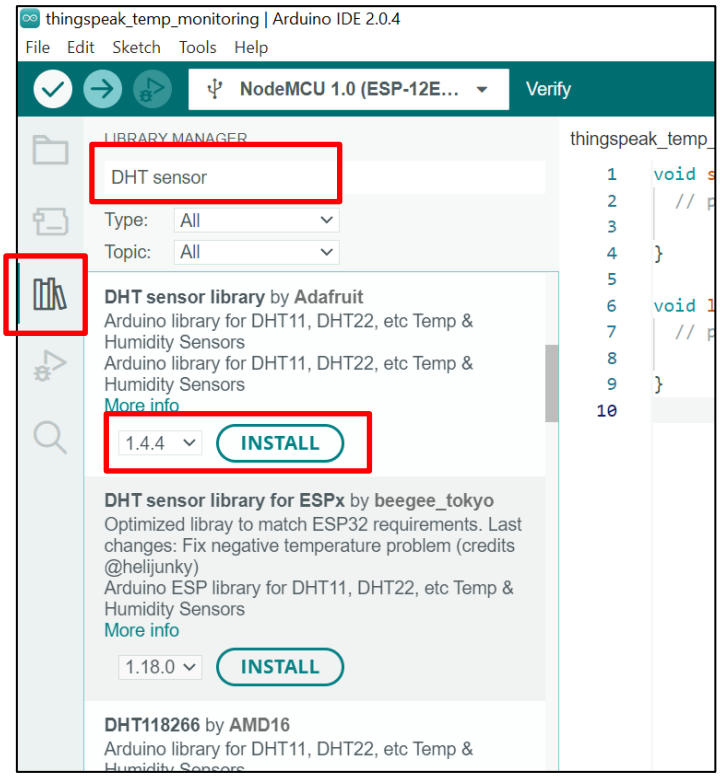
13. Going back to step 10, select the correct board and COM port number and then click ***OK***.



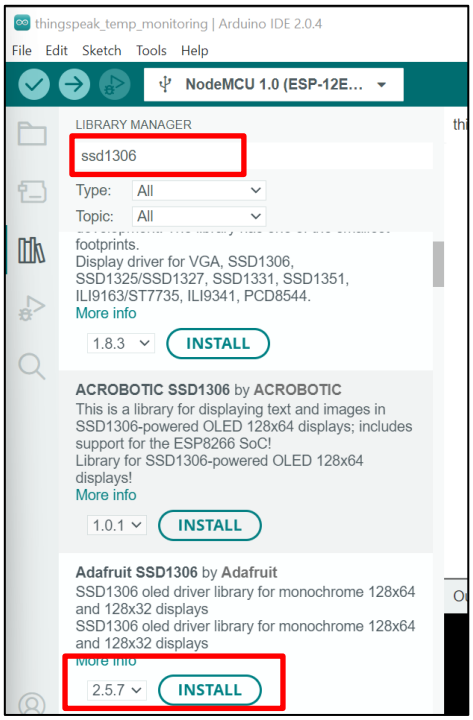


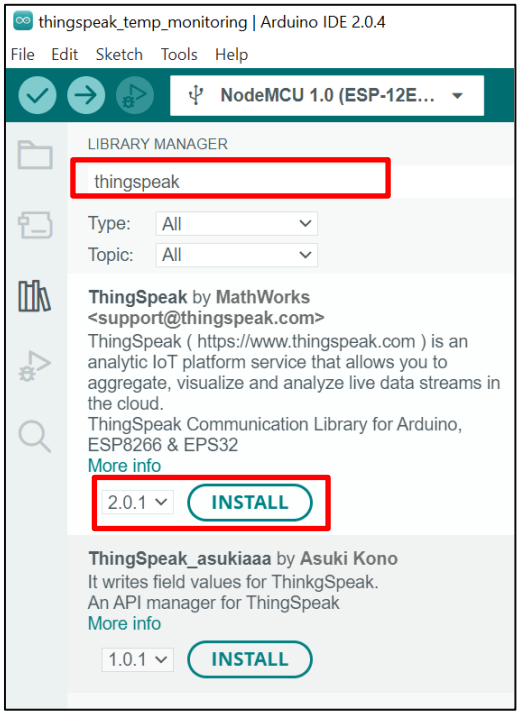
3. Sensor Library Installation

1. For this tutorial, there are 3 library packages that need to be downloaded and installed.
    a. DHT11
    b. OLED SSD1306
    c. ThingSpeak
2. Click ***Library Manager.*** Search for DHT sensor and select ***DHT sensor library by Adafruit*** and click ***INSTALL***.
3. Select ***INSTALL ALL*** to install all library dependencies.



4. In the Library Manager, search for SSD1306 and select ***Adafruit SSD1306 by Adafruit*** and click ***INSTALL***.
5. Last, search for ThingSpeak and select ***ThingSpeak by Mathworks*** and click ***INSTALL***.





4. Firmware Programming

In this section, you will learn how to measure the temperature and humidity using DHT11 and displaying the readings on Serial Monitor and on a 0.96 inch SSD1306 OLED display using Arduino IDE and NodeMCU ESP8266.

1. Copy sketch ***TempMonitoringOLED.ino*** to Arduino IDE and compile it.

```
1  //Libraries used
2  #include <ESP8266WiFi.h>
3  #include <Wire.h>
4  #include <DHT.h>
5  #include <Adafruit_GFX.h>
6  #include <Adafruit_SSD1306.h>
7
```

```
8  //OLED Display Setup
9  #define SCREEN_WIDTH 128 // OLED display width, in pixels
10 #define SCREEN_HEIGHT 32 // OLED display height, in pixels
11 Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);
12 unsigned long delayTime;
13
14 //Temperature & Humidity Setup
15 #define DHTTYPE DHT11 // DHT 11
16 //#define DHTTYPE DHT21 // DHT 21 (AM2301)
17 //#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
18
19 uint8_t DHTPin = D6;
20 DHT dht(DHTPin, DHTTYPE);
21 float Temperature;
22 float Humidity;
23 float Temp_Fahrenheit;
24
25 //WiFi Credentials
26 const char *ssid ="yourownssid";  // change to your own SSID @ hotspot
27 const char *pass = "yourssidpassword";  // change to your own Password
28 WiFiClient client;
29
30 void setup() {
31
32   // put your setup code here, to run once:
33   Serial.begin(115200);  // Initialize serial
34   while (!Serial) {
35     ; // wait for serial port to connect. Needed for Leonardo native USB
       // port only
36   }
37
38   WiFi.mode(WIFI_STA);
39   dht.begin();  // Initialized DHT11 sensor
40   ThingSpeak.begin(client); // Initialize ThingSpeak
41
42   // by default, we'll generate the high voltage from the 3.3v line
     //internally! (neat!)
43   display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize OLED Display
      // with the I2C addr 0x3C (for the 128x64)
44    // init done
45   display.display();
46   delay(100);
47   display.clearDisplay();
48   display.display();
49   display.setTextSize(1.75);
50   display.setTextColor(WHITE);
51 }
52
53 void loop() {
54   // Connect or reconnect to WiFi
55   if(WiFi.status() != WL_CONNECTED){
```

```
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network. Change this
       //line if using open or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }

//Sensor Readings
  Humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  Temperature = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  Temp_Fahrenheit= dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(Humidity) || isnan(Temperature) || isnan(Temp_Fahrenheit)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

//display temperature and humidity to Serial Monitor
  Serial.print(F("Humidity: "));
  Serial.print(Humidity);
  Serial.print(F("%  Temperature: "));
  Serial.print(Temperature);
  Serial.print(F("°C "));
  Serial.print(Temp_Fahrenheit);
  Serial.println(F("°F "));
  delay(1000);


// display temperature to OLED Display
  display.setCursor(0,0);
  display.clearDisplay();
  display.setTextSize(1.75);
  display.setCursor(0,0);
  display.print("Temp Monitoring");
  display.setTextSize(1);
  display.setCursor(0,16);
  display.print("Temperature: ");
  display.setTextSize(2);
  display.setCursor(0,26);
  display.print(Temperature);
  display.print(" ");
  display.setTextSize(1);
  display.cp437(true);
  display.write(167);
```

```
105     display.setTextSize(2);
106     display.print("C");
107
108     // display humidity
109     display.setTextSize(1);
110     display.setCursor(0, 41);
111     display.print("Humidity: ");
112     display.setTextSize(2);
113     display.setCursor(0, 51);
114     display.print(Humidity);
115     display.print(" %");
116     display.display();
117     delay(1000);
118
119  }
```

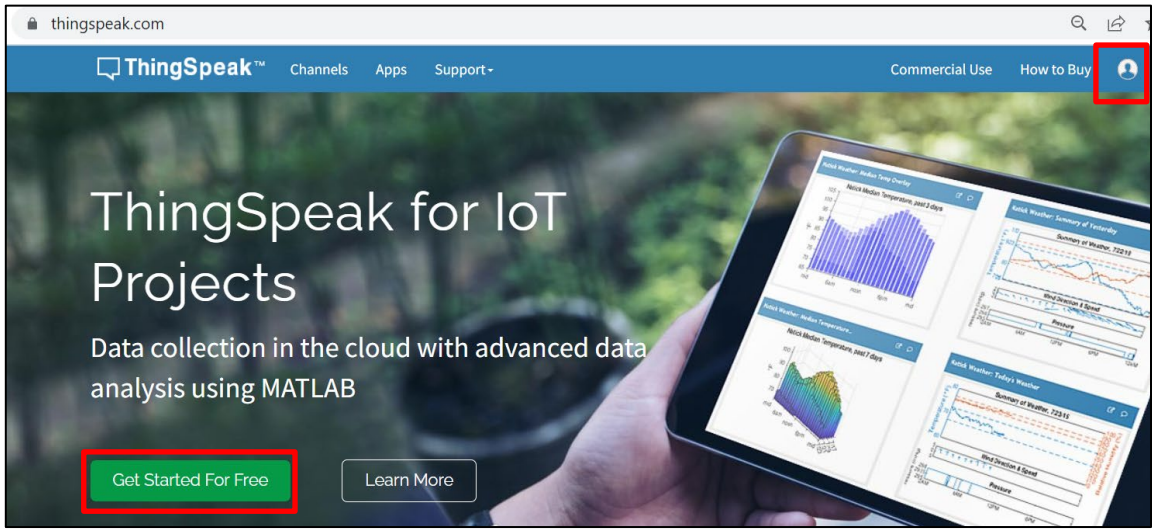
2. You should be able to see readings of temperature and humidity on the Serial Monitor and the OLED display (if available).

INTRODUCTION TO THINGSPEAK

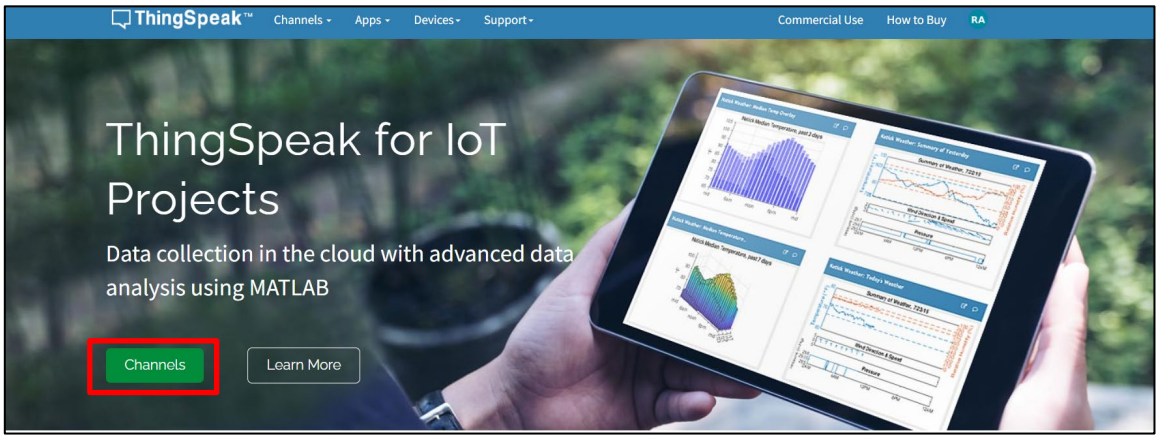
ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.

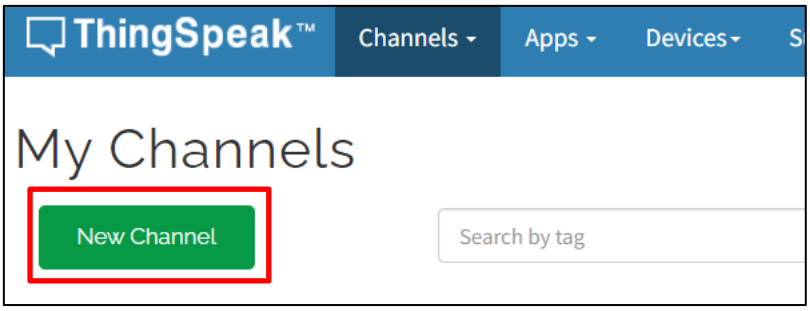
1. ThingSpeak Account Setup

1. Go to ***http://www.thingspeak.com***
2. Click ***Get Started For Free*** button and sign up as new user. If you already have an account, click on user icon to sign in.

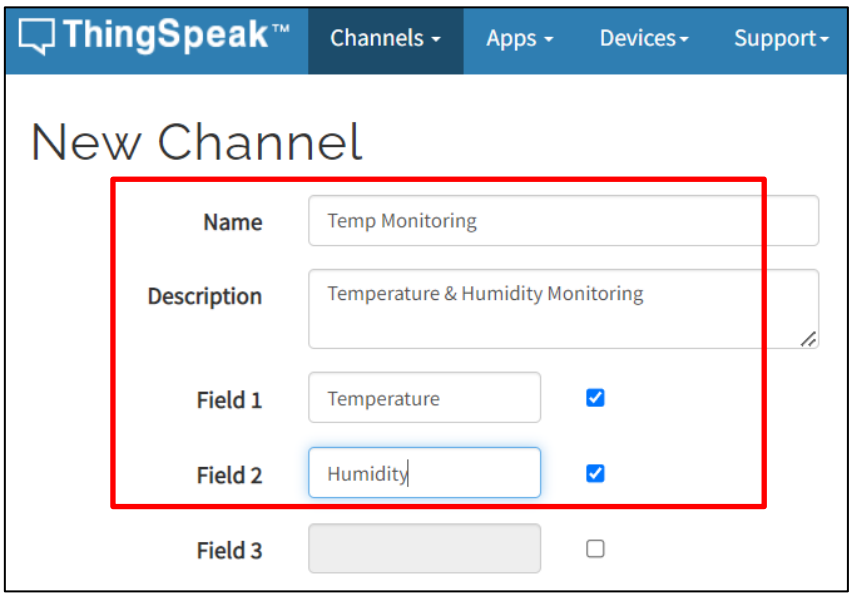


3. Go to ***My Channels*** by clicking on the ***Channels*** button. A channel is a source for your data, where you can store and retrieve data. Select ***New Channel.***
4. A channel can have a maximum of 8 fields. It means you can store 8 different data to a channel.

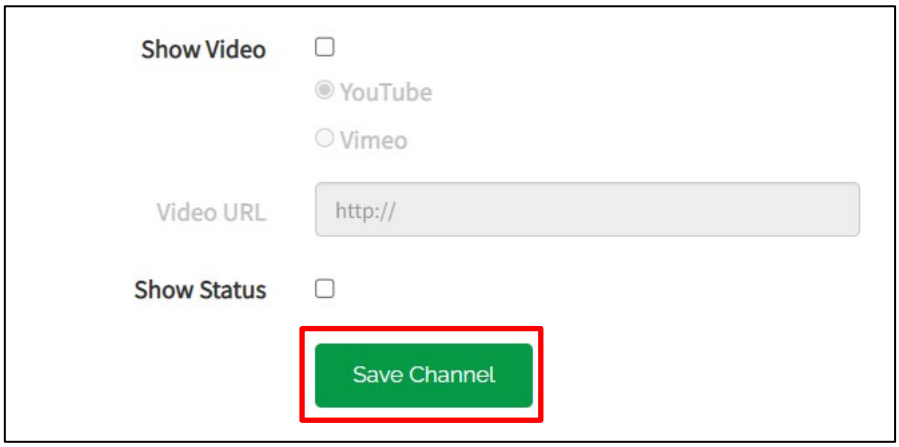




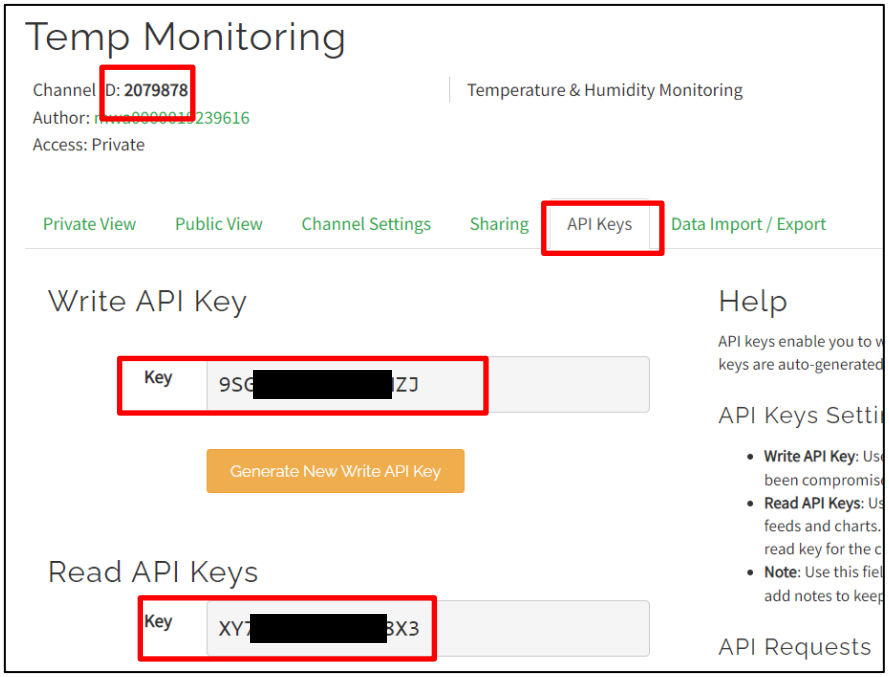
5. Set the Name, Description, Field 1 and Field 2 accordingly.



6. Scroll down and select ***Save Channel.***



7. Select the ***API Keys*** tab.
8. Copy the ***Channel ID*** number and the ***Write API Key***.



9. API (Application Programming Interface) keys are the keys to access your channel. In other words, these are passwords to access your channel. You can access your channel in two ways:
    a. To update channel / data logging: use API Write Key
    b. To retrieve data: use API Read Key

2. Integrating ThingSpeak Within Arduino Sketch

1. Open your Arduino sketch.
2. Add ThingSpeak library - **#include <ThingSpeak.h>** in ***Libraries used*** section.
3. Then add the codes below **before** the setup() function for ***MyChannelNumber*** and ***Write API Key*** accordingly.

```
//Libraries used
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <DHT.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ThingSpeak.h>

//ThingSpeak Setup
long myChannelNumber = youChannelID; // change to your own channel number
const char myWriteAPIKey[] = "yourownAPIwriteKey";  // change to your own API Write Key
```

4. Initialized ThingSpeak in **setup()** function.

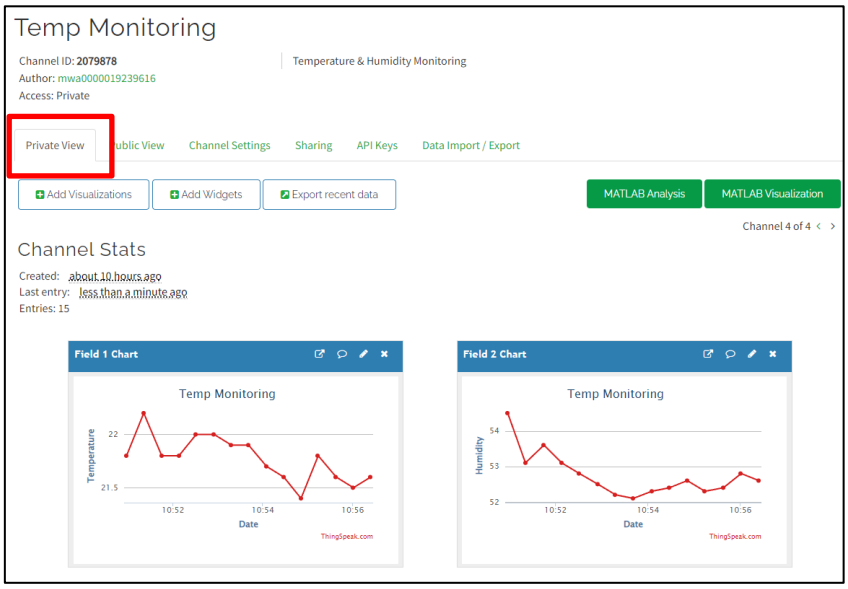
```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);  // Initialize serial
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
  }
  WiFi.mode(WIFI_STA);
  dht.begin();  // Initialized DHT11 sensor
  ThingSpeak.begin(client);  // Initialize ThingSpeak
```

5. Then before closing of the ***loop()*** function, add the following lines to send data to ThingSpeak.
6. Save and compile your sketch.

```
126  // Sending data to ThingSpeak
127  //----------------------------
128    ThingSpeak.setField(1, Temperature); // set temperature values to Field 1
129    ThingSpeak.setField(2, Humidity);    // set humidity values to Field 2
130
131  // write all data to the ThingSpeak channel
132    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
133    if(x == 200){
134      Serial.println("Channel update successful.");
135    }
136    else{
137      Serial.println("Problem updating channel. HTTP error code " +
     String(x));
138    }
139    delay(20000); // Wait 20 seconds to update the channel again
140  }
```

Closing curly brace for loop() function

7. View your ***Channel Stats*** by clicking ***Private View*** tab.
8. You should see the field charts of your channel.



9. Your final sketch should be like ***TempMonitoringThingSpeakOLED.ino.***

```
1  //Libraries used
2  #include <ESP8266WiFi.h>
3  #include <Wire.h>
4  #include <DHT.h>
5  #include <Adafruit_GFX.h>
6  #include <Adafruit_SSD1306.h>
7  #include <ThingSpeak.h>
8
9  //ThingSpeak Setup
10
11 long myChannelNumber = youChannelID; // change to your own channel number
12 const char myWriteAPIKey[] = "yourownAPIwriteKey";  // change to your own API Write Key
13
14 //OLED Display Setup
15 #define SCREEN_WIDTH 128 // OLED display width, in pixels
16 #define SCREEN_HEIGHT 32 // OLED display height, in pixels
17 Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);
18 unsigned long delayTime;
19
20 //Temperature & Humidity Setup
21 #define DHTTYPE DHT11 // DHT 11
22 //#define DHTTYPE DHT21 // DHT 21 (AM2301)
23 //#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
24
25 uint8_t DHTPin = D6;
26 DHT dht(DHTPin, DHTTYPE);
27
28 float Temperature;
```

```
28  float Humidity;
29  float Temp_Fahrenheit;
30
31  //WiFi Credentials
32  const char *ssid =  "yourownssid";  // change to your own SSID @ hotspot
33  const char *pass =  "yourssidpassword";  // change to your own Password
34  WiFiClient client;
35
36  void setup() {
37    // put your setup code here, to run once:
38    Serial.begin(115200);  // Initialize serial
39    while (!Serial) {
40      ; // wait for serial port to connect. Needed for Leonardo native USB port only
41    }
42    WiFi.mode(WIFI_STA);
43    dht.begin();  // Initialized DHT11 sensor
44    ThingSpeak.begin(client);  // Initialize ThingSpeak
45
46    // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
47    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize OLED Display with the I2C addr 0x3C (for
    the 128x64)
48     // init done
49    display.display();
50    delay(100);
51    display.clearDisplay();
52    display.display();
53    display.setTextSize(1.75);
54    display.setTextColor(WHITE);
55  }
56
57  void loop() {
58    // Connect or reconnect to WiFi
59    if(WiFi.status() != WL_CONNECTED){
60      Serial.print("Attempting to connect to SSID: ");
61      Serial.println(ssid);
62      while(WiFi.status() != WL_CONNECTED){
63        WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network. Change this line if using open or WEP
    network
64        Serial.print(".");
65        delay(5000);
66      }
67      Serial.println("\nConnected.");
68    }
69
70  //Sensor Readings
71    Humidity = dht.readHumidity();
72    // Read temperature as Celsius (the default)
73    Temperature = dht.readTemperature();
74    // Read temperature as Fahrenheit (isFahrenheit = true)
75    Temp_Fahrenheit= dht.readTemperature(true);
76
```

```
77    // Check if any reads failed and exit early (to try again).
78    if (isnan(Humidity) || isnan(Temperature) || isnan(Temp_Fahrenheit)) {
79      Serial.println(F("Failed to read from DHT sensor!"));
80      return;
81    }
82
83    //display temperature and humidity to Serial Monitor
84    Serial.print(F("Humidity: "));
85    Serial.print(Humidity);
86    Serial.print(F("%  Temperature: "));
87    Serial.print(Temperature);
88    Serial.print(F("°C "));
89    Serial.print(Temp_Fahrenheit);
90    Serial.println(F("°F "));
91    delay(1000);
92  // display temperature to OLED Display
93    display.setCursor(0,0);
94    display.clearDisplay();
95    display.setTextSize(1.75);
96    display.setCursor(0,0);
97    display.print("Temp Monitoring");
98    display.setTextSize(1);
99    display.setCursor(0,16);
100   display.print("Temperature: ");
101   display.setTextSize(2);
102   display.setCursor(0,26);
103   display.print(Temperature);
104   display.print(" ");
105   display.setTextSize(1);
106   display.cp437(true);
107   display.write(167);
108   display.setTextSize(2);
109   display.print("C");
110   // display humidity
111   display.setTextSize(1);
112   display.setCursor(0, 41);
113   display.print("Humidity: ");
114   display.setTextSize(2);
115   display.setCursor(0, 51);
116   display.print(Humidity);
117   display.print(" %");
118   display.display();
119   delay(1000);
120
121   // Sending data to ThingSpeak
122 //----------------------------
123   ThingSpeak.setField(1, Temperature); // set temperature values to Field 1
124   ThingSpeak.setField(2, Humidity);    // set humidity values to Field 2
125
126 // write all data to the ThingSpeak channel
127   int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
```

```
128     if(x == 200){
129       Serial.println("Channel update successful.");
130     }
131     else{
132       Serial.println("Problem updating channel. HTTP error code " + String(x));
133     }
134     delay(20000); // Wait 20 seconds to update the channel again
135   }
136
```

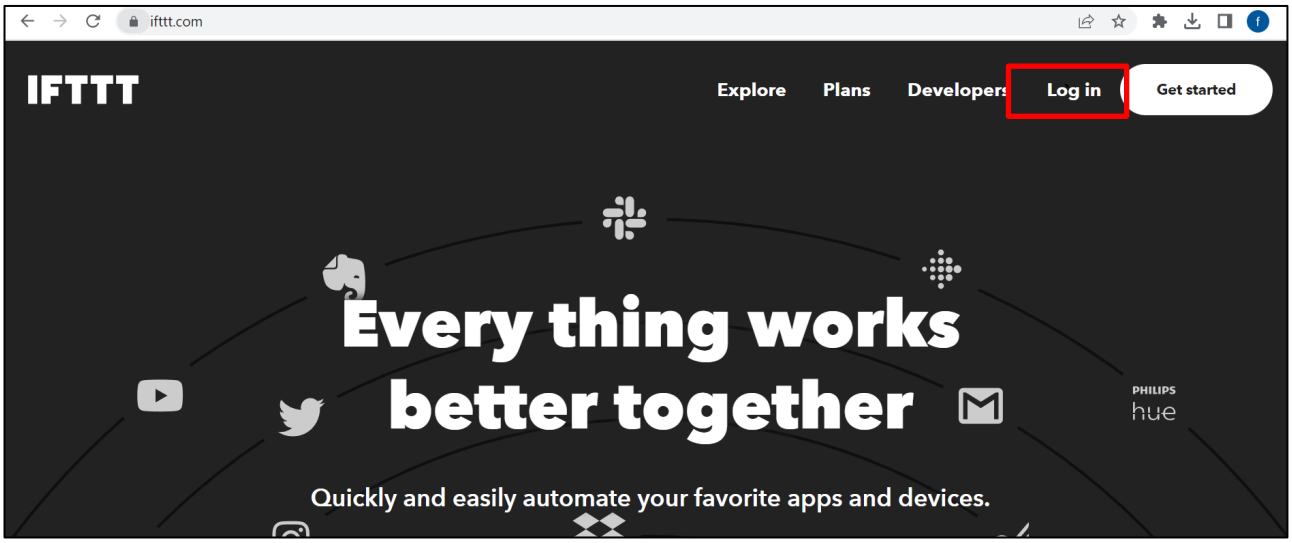
Congratulations! You have successfully sent temperature and humidity data from NodeMCU to the ThingSpeak platform.

INTRODUCTION TO IFTTT

IFTTT is a web service that lets you create applets that act in response to another action. You can use the IFTTT Webhooks service to create web requests to trigger an action. The incoming action is an HTTP request to the web server, and the outgoing action is notification in the IFTTT app on your device.

1. IFTTT Account Setup

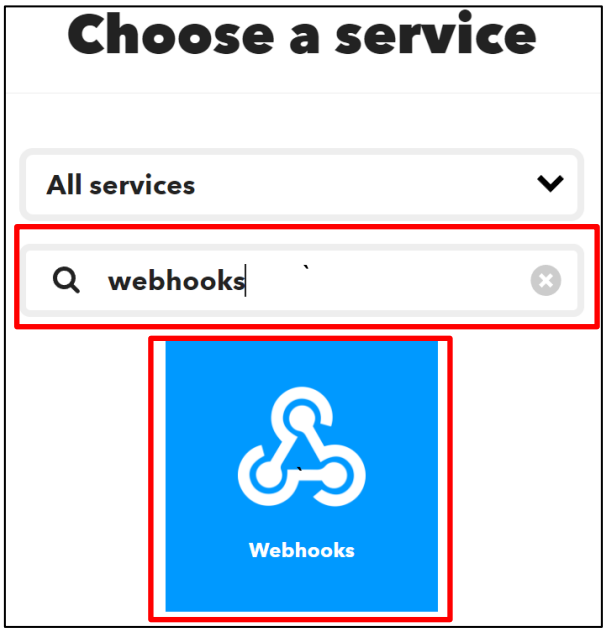
1. Download and install IFTTT App on your mobile phone.
2. Create your account.
3. Go to ***http://www.ifttt.com*** on your computer and login using the account you have created.



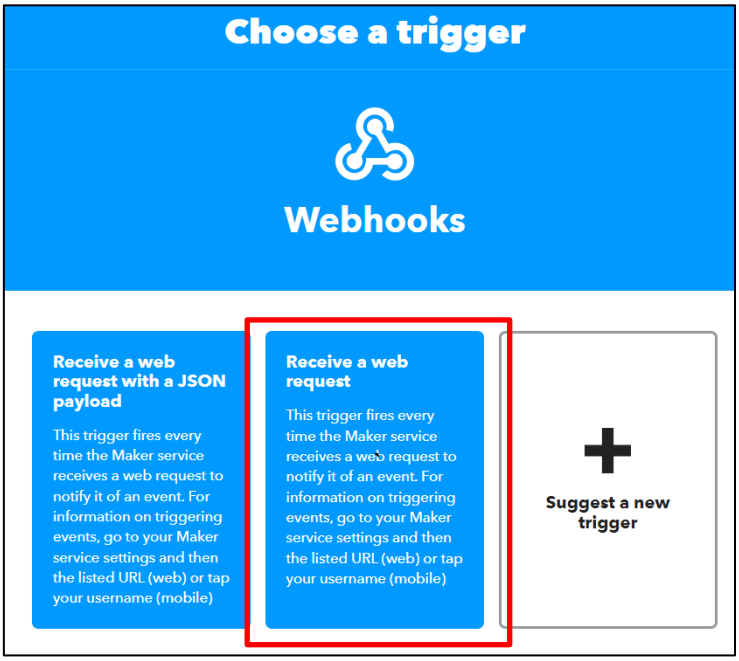
4. Click **Create** to create a new applet.

5. Select the input action by clicking ***Add*** in the **If This** section.

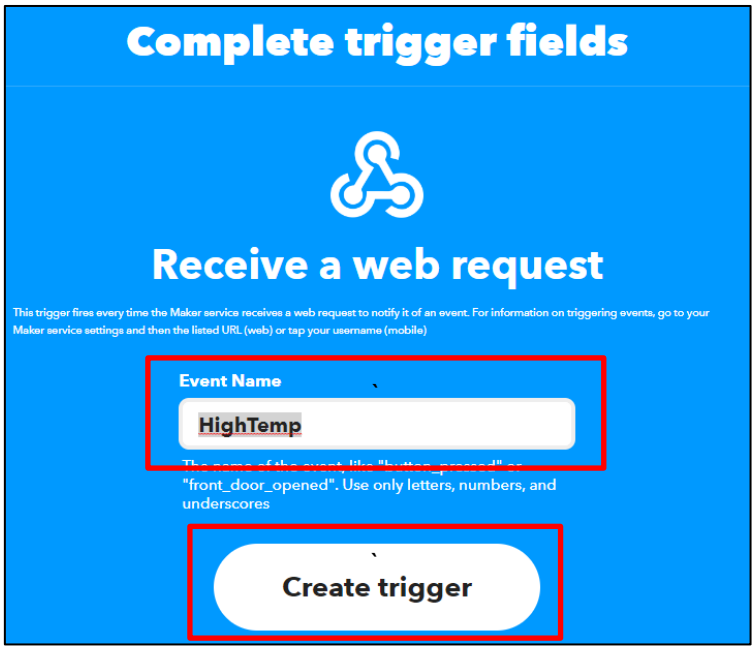
6. Search for ***Webhook*** and click the webhooks icon.



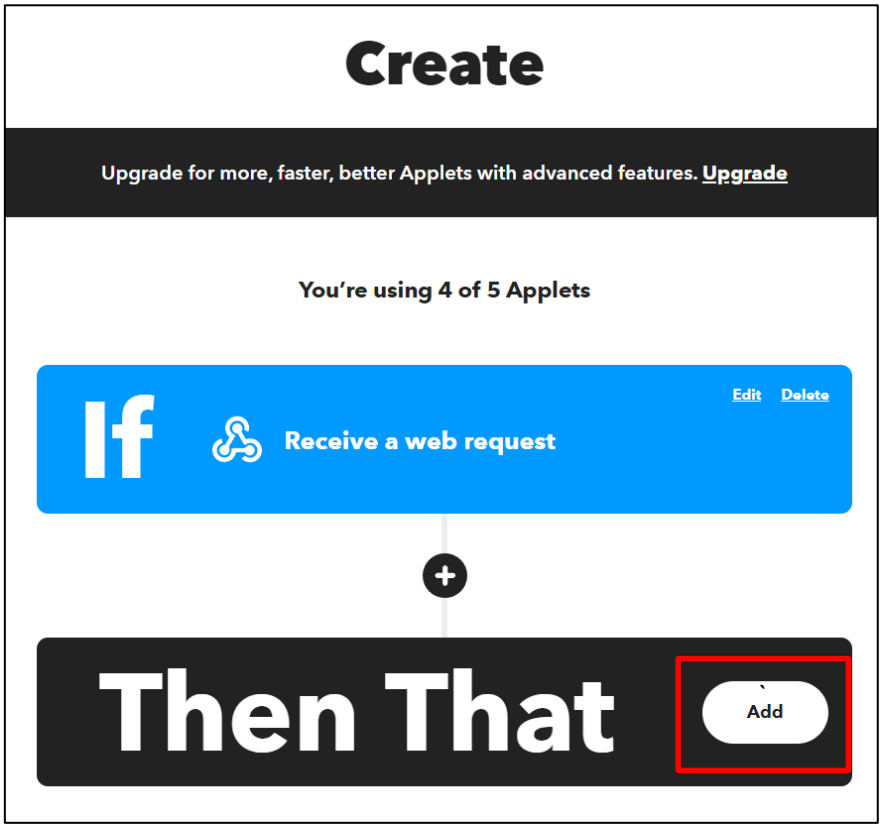
7. Select ***Receive a web request***.



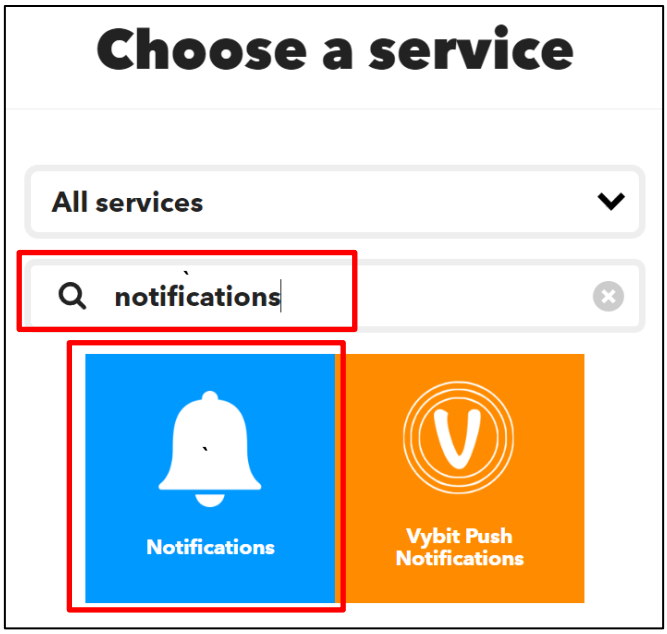
8. Set the ***Event Name*** for the trigger field. Click ***Create trigger***.



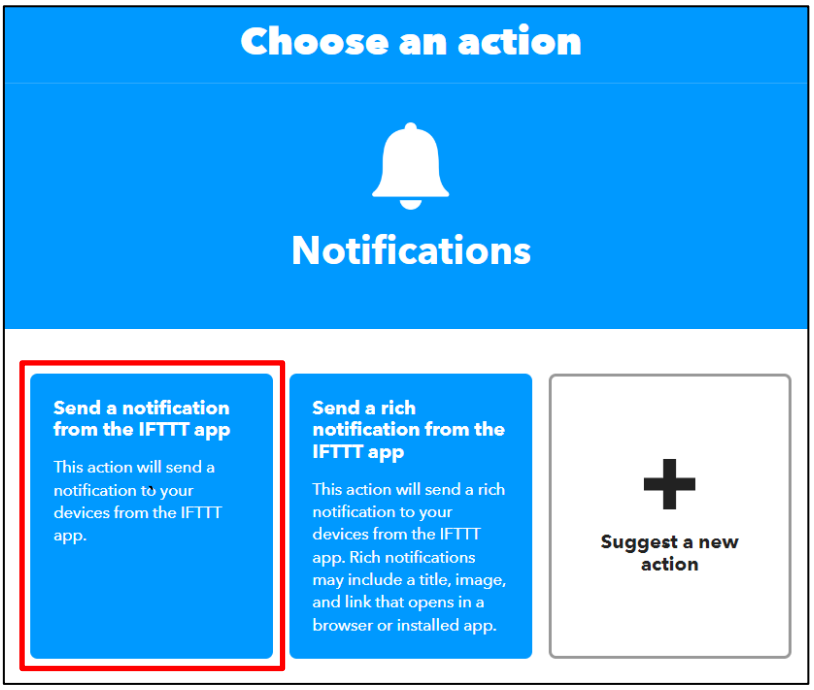
9. Select the resulting action by click the Add button in **Then That** section.



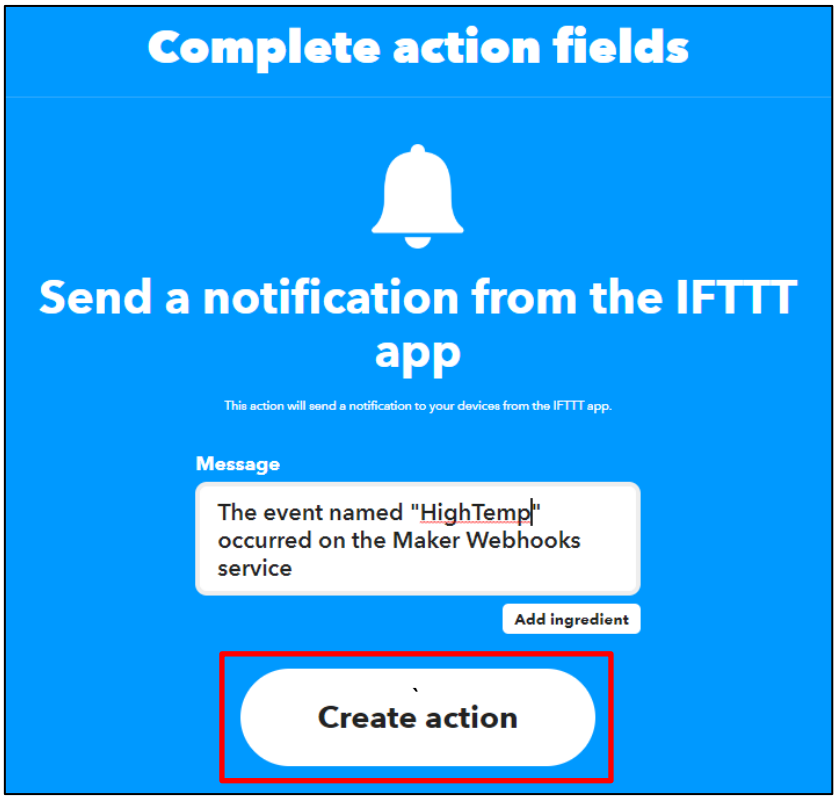
10. Choose notifications services.



11. Select ***Send a notification from the IFTTT app.***

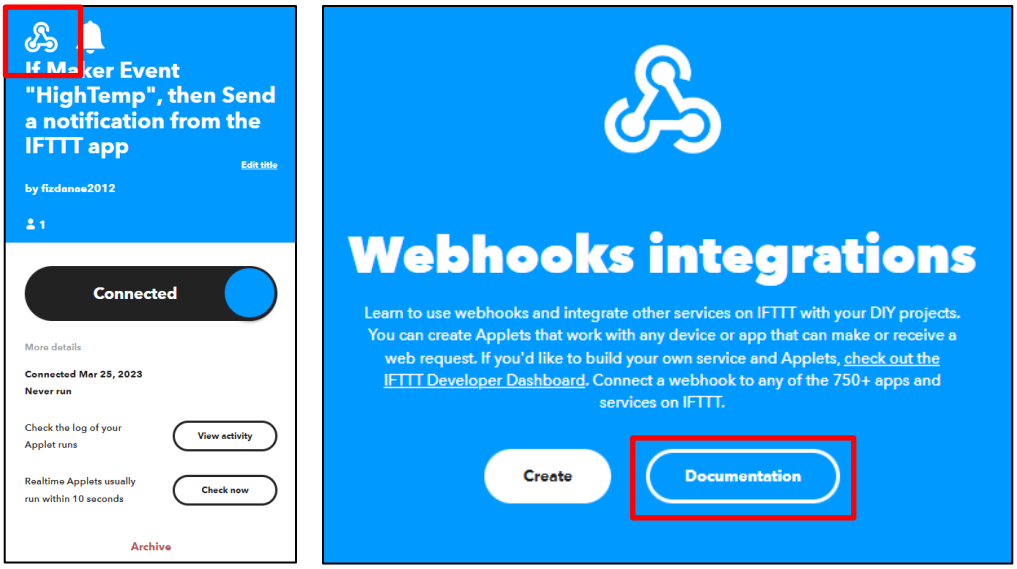


12. Complete the action fields and click the ***Create action*** button.

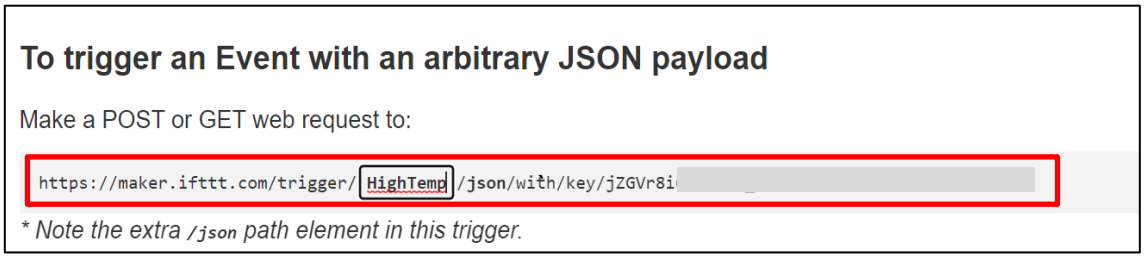


13. Select ***Continue***.

14. Review and click ***Finish***.
15. Click the **Webhooks icon** then the documentation button to retrieve ***webhook trigger information***.



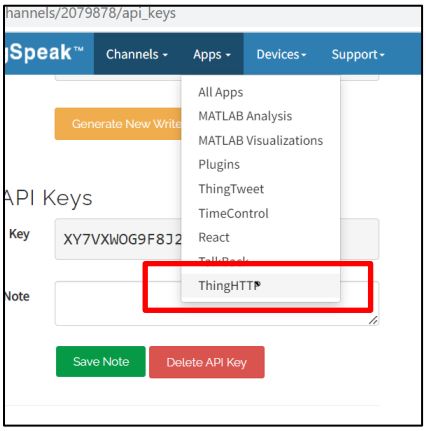
16. Take note of the key for your trigger information.
17. Replace **{event}** with event name ***HighTemp***.



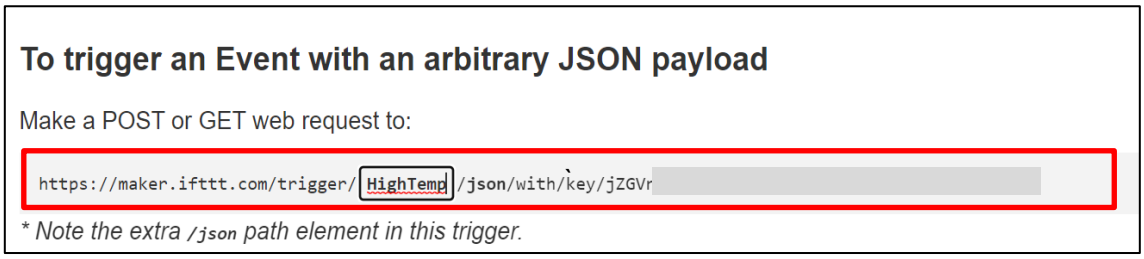
2. Integrating IFTTT with ThingHTTP

To complete the trigger request, create a **ThingHTTP**. The **ThingHTTP** app lets you trigger predefined HTTP requests with an API key and a GET request from the web or from a device. For this example, use **ThingHTTP** to trigger Webhooks at IFTTT.

1. Choose ***Apps*** > ***ThingHTTP*** and select ***New ThingHTTP***.



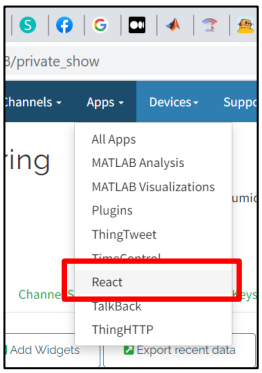
2. Edit your **ThingHTTP** configurations.
   a. ***Name*** - Name your ThingHTTP.
   b. ***URL*** - Enter the URL from the Webhooks documentation. The URL for this example has the form `https://maker.ifttt.com/trigger/HighTemp/json/with/key/jZGVr8i6NPUb1h_72xbd9Ymcv1hYMAL`[redacted]
   c. **Method** - Enter ***GET***.
   d. Click ***Save ThingHTTP***.

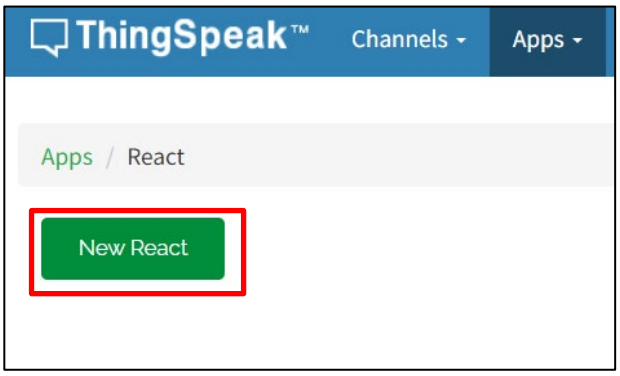


3. Create a React to Your Data

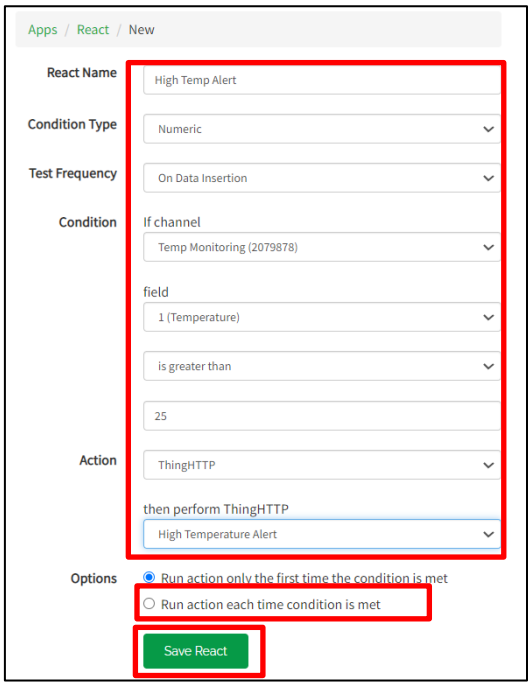
Create a React to trigger the ThingHTTP based on your channel data. You must be the author of the channel used to create a react. The React app can evaluate your ThingSpeak channel data and trigger other events. Create an instance of the React app that triggers when the temperature is too high.

1. Choose **Apps** > **React**, and then click **New React**.





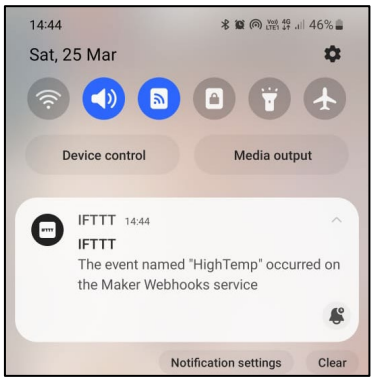
2. Edit your React configurations.
   a. **Name** - Name your React.
   b. **Test Frequency** - Select **On Data Insertion**.
   c. **Condition** - Select your temperature channel in the **If channel** list.
   d. **Field** - Select field 1, set the requirement to **is more than**, and set the temperature level to **25** or any other value that you want.
   e. **Action** - Select **ThingHTTP** and choose the name of the ThingHTTP you defined previously.
   f. **Options** - Select **Run action each time condition is met**.



3. Click ***Save React***.

4. Trigger Your IFTTT Notification

1. Once the temperature in the channel reaches the set point for your React, you will receive a notification in the IFTTT app on your device.



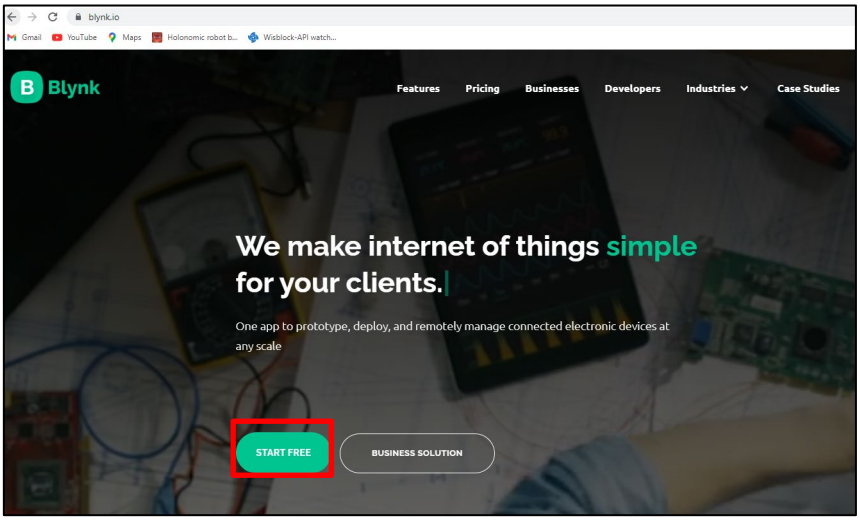
2. You can also trigger Webhooks at IFTTT from your browser. Copy the address from the Webhooks documentation to your browser address window to try triggering the event directly. If successful, IFTTT replies with ***"Congratulations! You've fired the HighTemp JSON event".***

INTRODUCTION TO BLYNK IOT

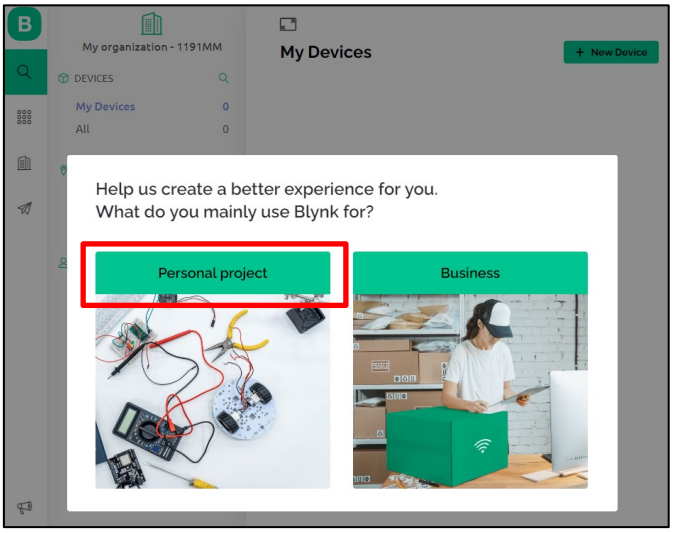
Blynk is an Internet of things (IoT) company which provides a platform for building mobile (IOS and Android) applications that can connect electronic devices to the Internet and remotely monitor and control these devices. Blynk platform offers features such as Blynk.App: a unique, no-code mobile app builder for IoT, Blynk.360: a web console to manage devices, users and data, Templates: new device creation process designed for scale, Blynk.Inject: a WI-FI manager built into the Blynk mobile app, Blynk.Air: over-the-air firmware updates, Automations and much more.

1. Blynk.Cloud Setup

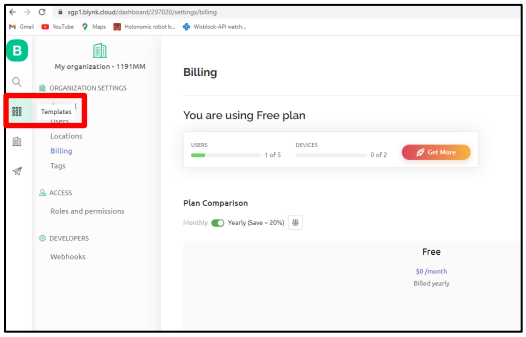
1. Open any browser and go to ***blynk.io***
2. Setup a new free account. Click ***START FREE***.

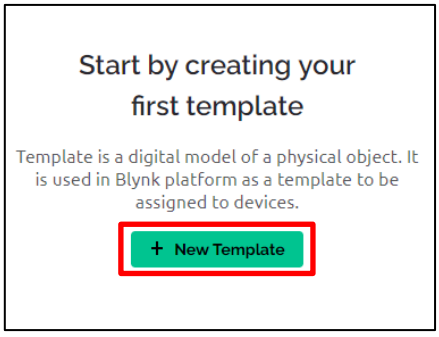


3. Sign Up using your email address and set up your password via activation link send to your registered email.
4. Then Log in using your credentials.
5. Select **Personal project**.

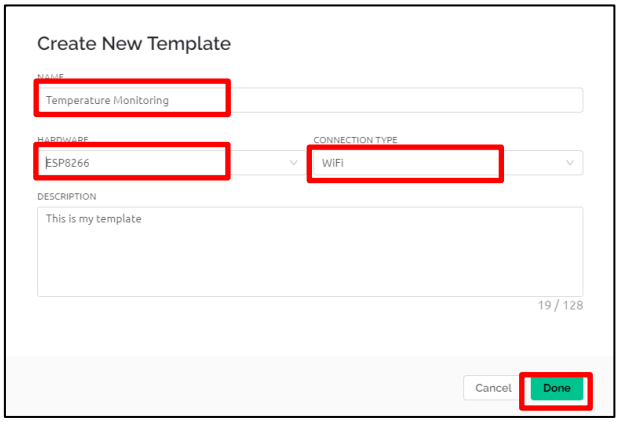


6. Select ***Templates*** and click ***+ New Templates***.

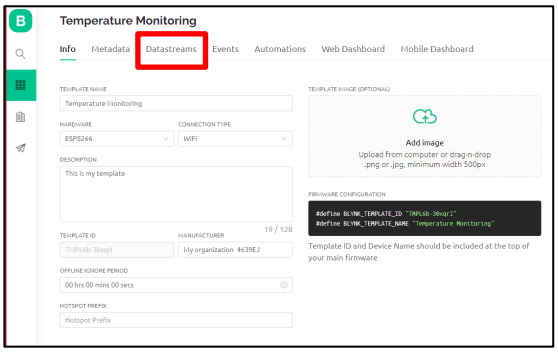


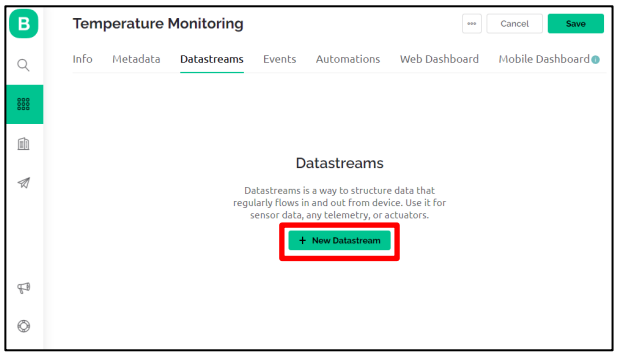


7. Configure New Template. Set the following, and click ***Done***.
    a. **NAME = *Temperature Monitoring***,
    b. **HARDWARE** = ***ESP8266***
    c. **CONNECTION TYPE** = ***WiFi***

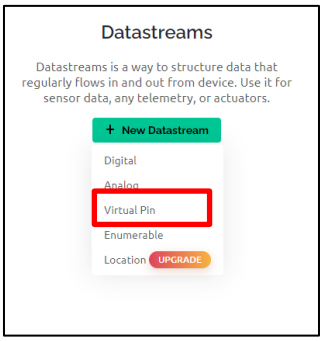


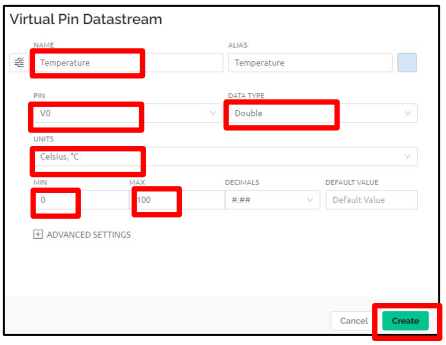
8. Create a new Datastream. Select Datastreams tab and click +New Datastream.

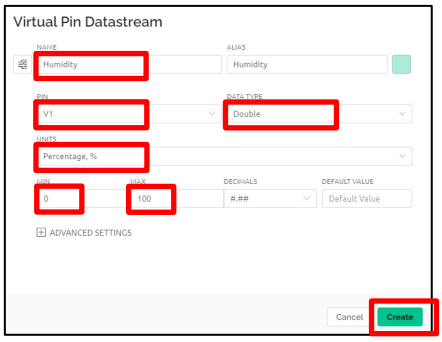




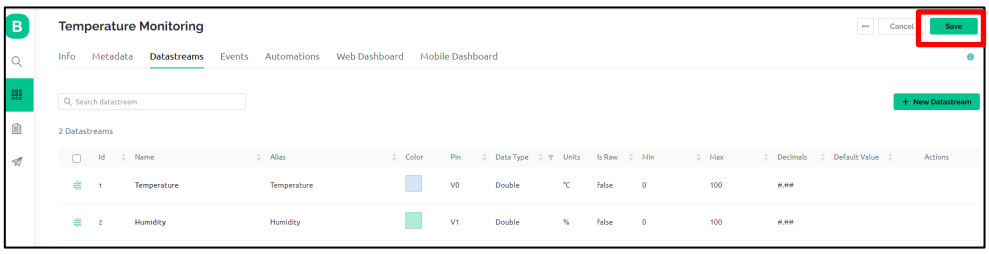
9. Select Virtual Pin. Set the following and click ***Create***.
    a. **NAME** = ***Temperature***
    b. **PIN** = ***V0***
    c. **DATA TYPE** = ***Double***
    d. **UNITS** = ***Celcius, °C***
    e. **MIN** = ***0***
    f. **MAX** = ***100***





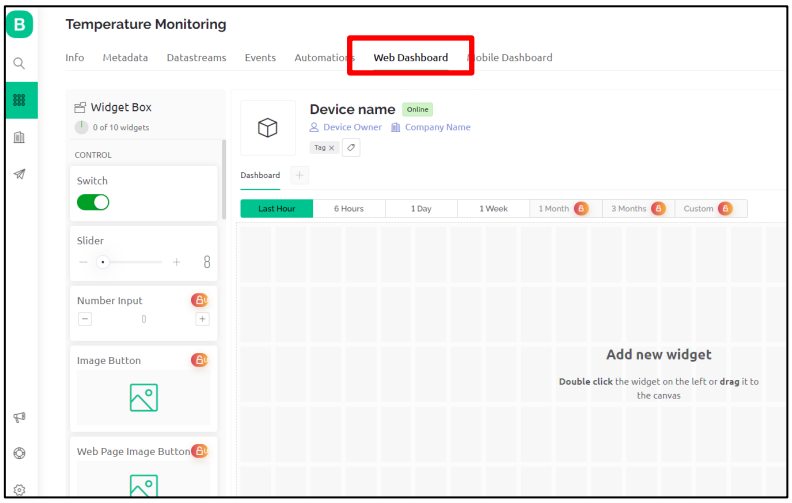


10. Create another datastream for Humidity with the following configuration:
    a. **NAME** = ***Humidity***
    b. **PIN** = ***V1***
    c. **DATA TYPE** = ***Double***
    d. **UNITS** = ***%***
    e. **MIN** = ***0***
    f. **MAX** = ***100***

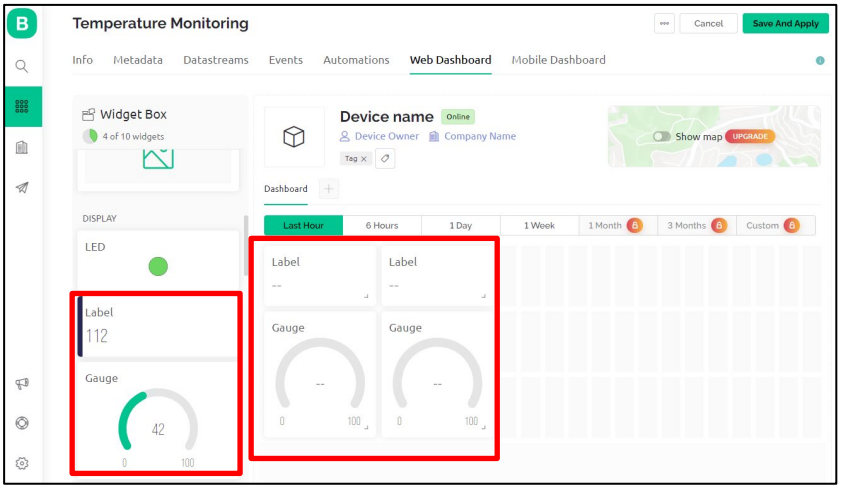


11. Click Save to save your template.

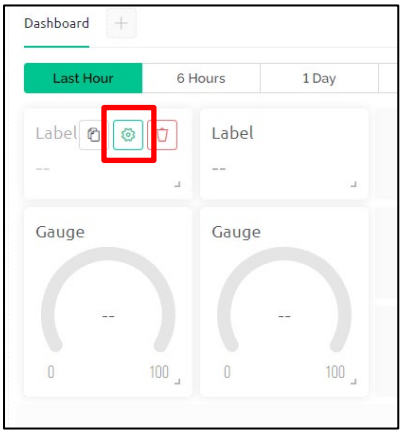
12. Select Web Dashboard to create the dashboard.

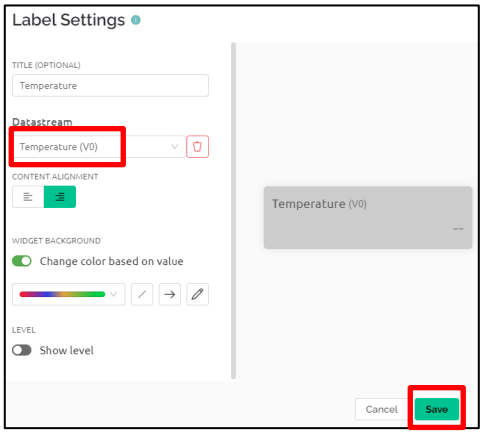


13. Create 2 ***Label*** widget and 2 ***Gauge*** widget by double clicking on the widget or drag it to the canvas.

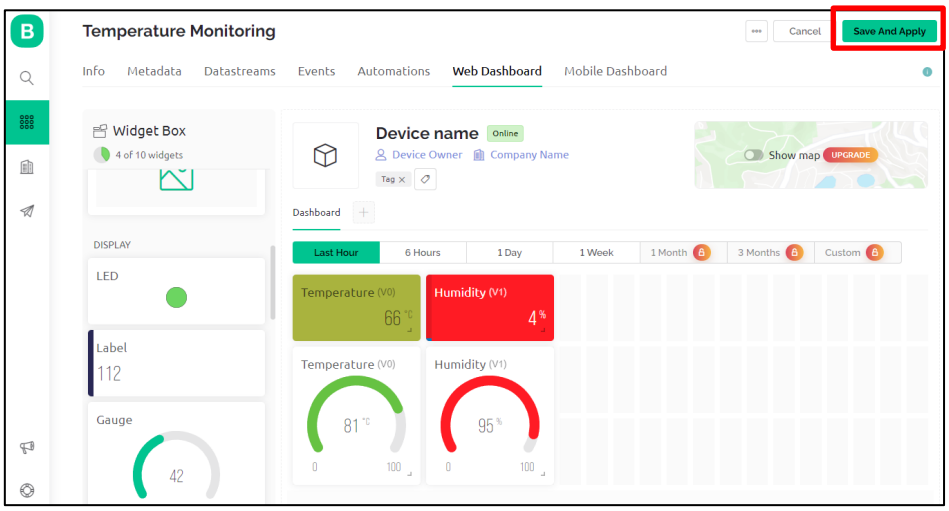


14. Configure each Label and Gauge and assign the datastreams created previously. Click ***Save***.

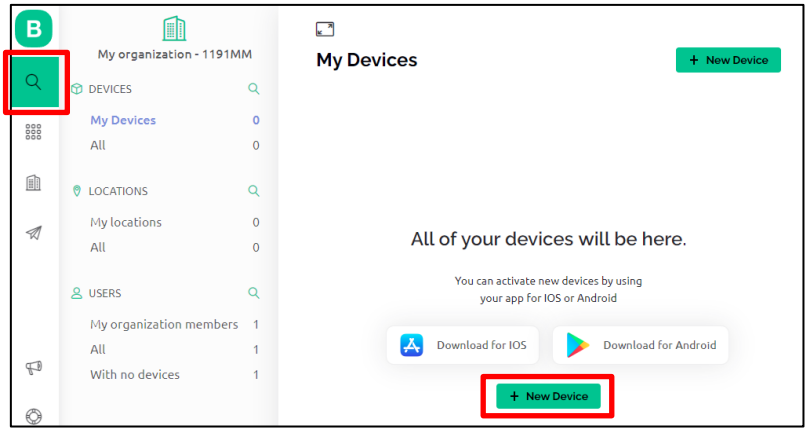




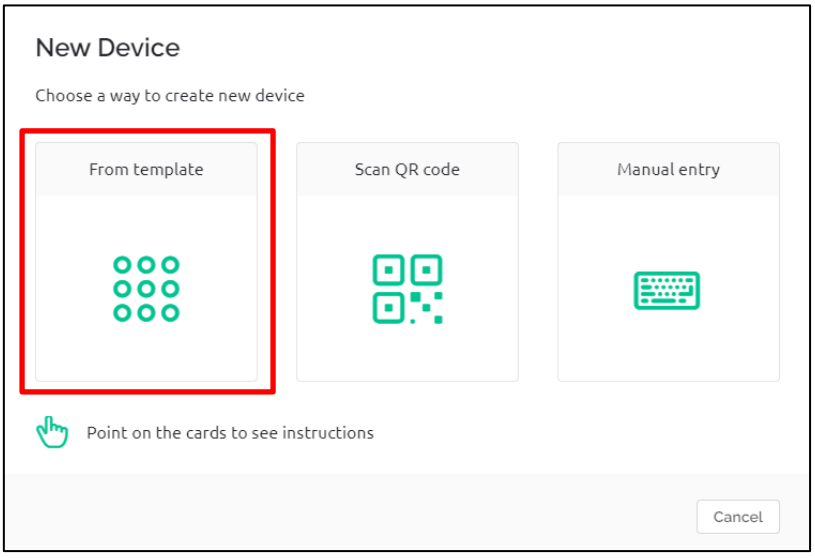
15. Configure the other Label widget and Gauge widgets accordingly. Click ***Save And Apply***.



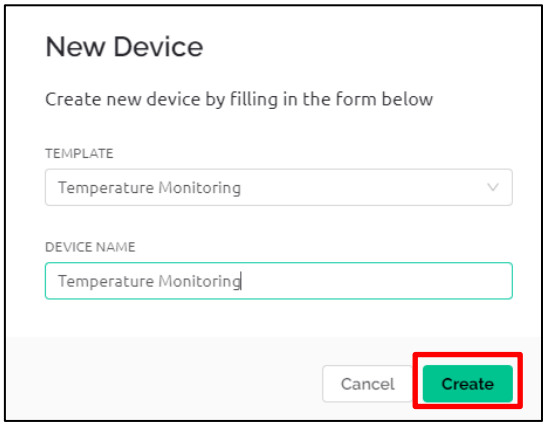
16. Select the search button and click ***+ New Device***.



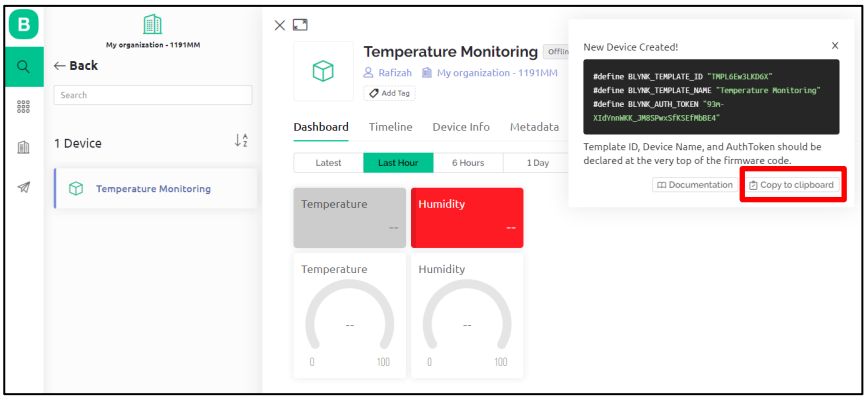
17. Select From template.



18. Select the Temperature Monitoring template created previously. Click ***Create***.



19. Click Copy to clipboard. These configurations will be used to configure Blynk connection in Arduino IDE. You can also access these in ***Device Info***.



2. Integrating Blynk with Arduino Sketch

1. Open your previous Arduino Sketch.
2. Using Library Manager install **Blynk by Volodymyr Shymanskyy version 1.2.0**.
3. Update your sketch according to the ***TempMonitoringBlynk.ino***.

```
1 //DHT11 and NodeMCU with Blynk
2 //----------------------------
3 //Libraries used
4 #include <ESP8266WiFi.h>
5 #include <Wire.h>
6 #include <DHT.h>
7 #include <Adafruit_GFX.h>
8 #include <Adafruit_SSD1306.h>
9 #include <ThingSpeak.h>
```

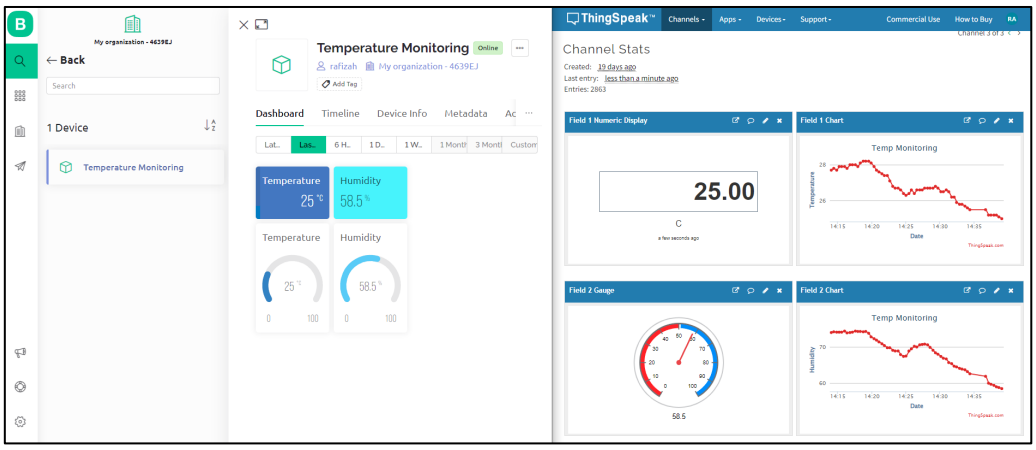
```
#include <BlynkSimpleEsp8266.h>

//Blynk Setup
//------------
#define BLYNK_PRINT Serial
/* Fill in information from Blynk Device Info here */
#define BLYNK_TEMPLATE_ID "yourownTemplateID"   // change to your own template ID
#define BLYNK_TEMPLATE_NAME "yourownTemplateName"   // change to your own template name
#define BLYNK_AUTH_TOKEN "yourownBlynkAuthToken"  // change to your own Blynk_auth_token
BlynkTimer timer;

//ThingSpeak Setup
long myChannelNumber = youChannelID; // change to your own channel number
const char myWriteAPIKey[] = "yourownAPIwriteKey";  // change to your own API Write Key

//OLED Display Setup
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);
unsigned long delayTime;

//Temperature & Humidity Setup
#define DHTTYPE DHT11 // DHT 11
//#define DHTTYPE DHT21 // DHT 21 (AM2301)
//#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

uint8_t DHTPin = D6;
DHT dht(DHTPin, DHTTYPE);
float Temperature;
float Humidity;
float Temp_Fahrenheit;

// Your WiFi credentials.
// Set password to "" for open networks.
const char *ssid =  "yourownssid";  // change to your own SSID @ hotspot
const char *pass =  "yourssidpassword";  // change to your own Password
WiFiClient client;

void oledDisplay(){
// display temperature to OLED Display
  display.setCursor(0,0);
  display.clearDisplay();
  display.setTextSize(1.75);
  display.setCursor(0,0);
  display.print("Temp Monitoring");
  display.setTextSize(1);
```

```
56    display.setCursor(0,16);
57    display.print("Temperature: ");
58    display.setTextSize(2);
59    display.setCursor(0,26);
60    display.print(Temperature);
61    display.print(" ");
62    display.setTextSize(1);
63    display.cp437(true);
64    display.write(167);
65    display.setTextSize(2);
66    display.print("C");
67
68    // display humidity
69    display.setTextSize(1);
70    display.setCursor(0, 41);
71    display.print("Humidity: ");
72    display.setTextSize(2);
73    display.setCursor(0, 51);
74    display.print(Humidity);
75    display.print(" %");
76    display.display();
77    delay(1000);
78  }
79  void sendSensor(){
80  //Sensor Readings
81    Humidity = dht.readHumidity();
82    // Read temperature as Celsius (the default)
83    Temperature = dht.readTemperature();
84    // Read temperature as Fahrenheit (isFahrenheit = true)
85    Temp_Fahrenheit= dht.readTemperature(true);
86
87    // Check if any reads failed and exit early (to try again).
88    if (isnan(Humidity) || isnan(Temperature) || isnan(Temp_Fahrenheit)) {
89      Serial.println(F("Failed to read from DHT sensor!"));
90      return;
91    }
92
93  //display temperature and humidity to Serial Monitor
94    Serial.print(F("Humidity: "));
95    Serial.print(Humidity);
96    Serial.print(F("%  Temperature: "));
97    Serial.print(Temperature);
98    Serial.print(F("°C "));
99    Serial.print(Temp_Fahrenheit);
100   Serial.println(F("°F "));
101   delay(1000);
```

```
102
103 // Sending data to ThingSpeak
104 //----------------------------
105   ThingSpeak.setField(1, Temperature); // set temperature values to Field 1
106   ThingSpeak.setField(2, Humidity);    // set humidity values to Field 2
107
108 // write all data to the ThingSpeak channel
109   int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
110   if(x == 200){
111     Serial.println("Channel update successful.");
112   }
113   else{
114     Serial.println("Problem updating channel. HTTP error code " + String(x));
115   }
116   delay(20000); // Wait 20 seconds to update the channel again
117
118 // Sending data to Blynk IoT
119   // You can send any value at any time.
120   // Please don't send more that 10 values per second.
121   Blynk.virtualWrite(V0, Temperature);  // V0 correspond to the Virtual pin for Temperature datastream
    set on Blynk2.0
122   Blynk.virtualWrite(V1, Humidity);   // V1 correspond to the Virtual pin for Humidity datastream set
    on Blynk2.0
123 }
124
125 void setup() {
126   // put your setup code here, to run once:
127   Serial.begin(115200);  // Initialize serial
128   while (!Serial) {
129     ; // wait for serial port to connect. Needed for Leonardo native USB port only
130   }
131   WiFi.mode(WIFI_STA);
132   pinMode(LED, OUTPUT);
133   Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass); //Initialized Blynk
134   dht.begin();  // Initialized DHT11 sensor
135   ThingSpeak.begin(client);  // Initialize ThingSpeak
136   // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
137   display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize OLED Display with the I2C addr 0x3C
138   // init done
139   display.display();
140   delay(100);
141   display.clearDisplay();
142   display.display();
143   display.setTextSize(1.75);
144   display.setTextColor(WHITE);
145 // Setup a function to be called every 1 seconds
```

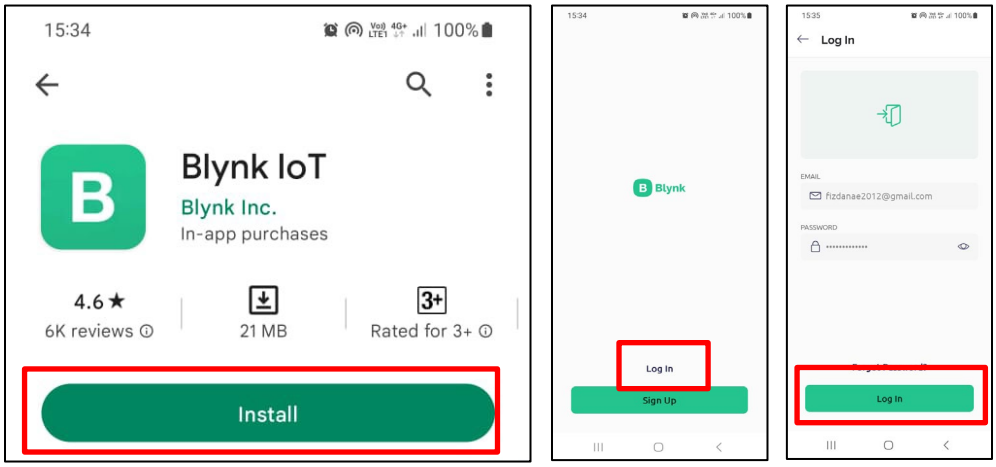
```
146   timer.setInterval(1000L, sendSensor);
147 }
148
149 void loop() {
150   // Connect or reconnect to WiFi
151   if(WiFi.status() != WL_CONNECTED){
152     Serial.print("Attempting to connect to SSID: ");
153     Serial.println(ssid);
154     while(WiFi.status() != WL_CONNECTED){
155       WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network. Change this line if using open or WEP
    network
156       Serial.print(".");
157       delay(5000);
158     }
159     Serial.println("\nConnected.");
160   }
161
162   Blynk.run();
163   timer.run();
164   oledDisplay();
165 }
166
```

4. Upload the sketch and if successful, your NodeMCU should be able to send temperature and humidity readings to both ThingSpeak and Blynk.

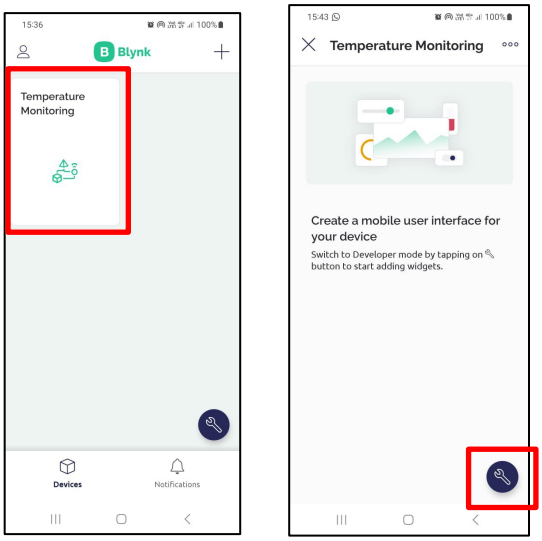


3. Blynk.App Setup

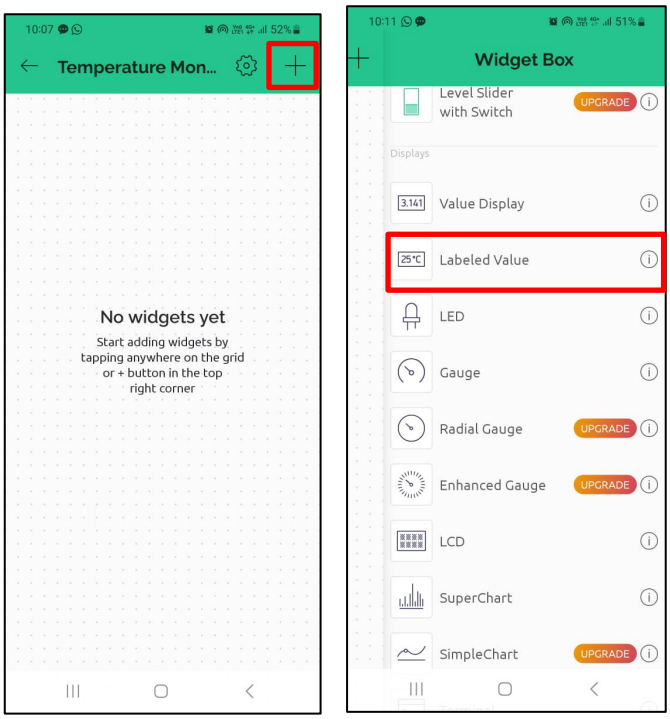
1. Download and install Blynk IoT from Google Play Store or Apple Apps Store.
2. ***Log In*** using account created earlier.



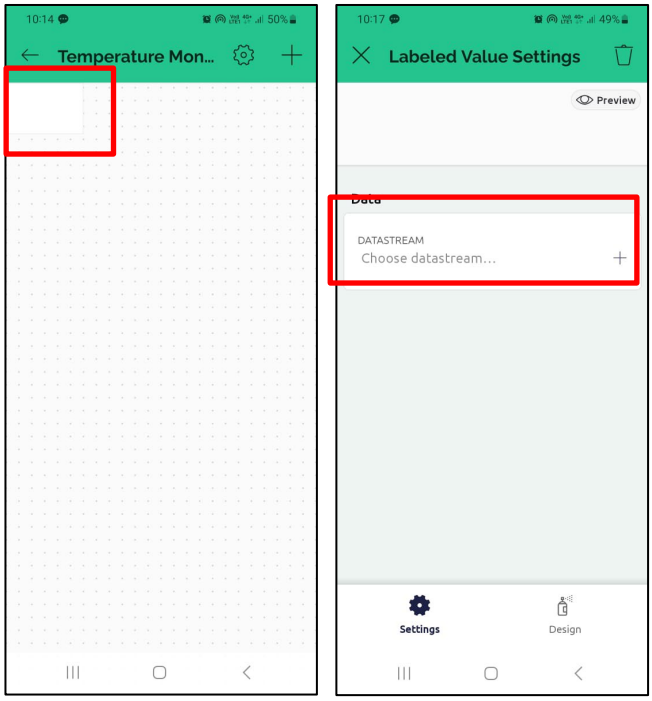
3. Click the template that have been created on Blynk.Cloud earlier.
4. Then click Developer Mode to add widget to the mobile app interface.

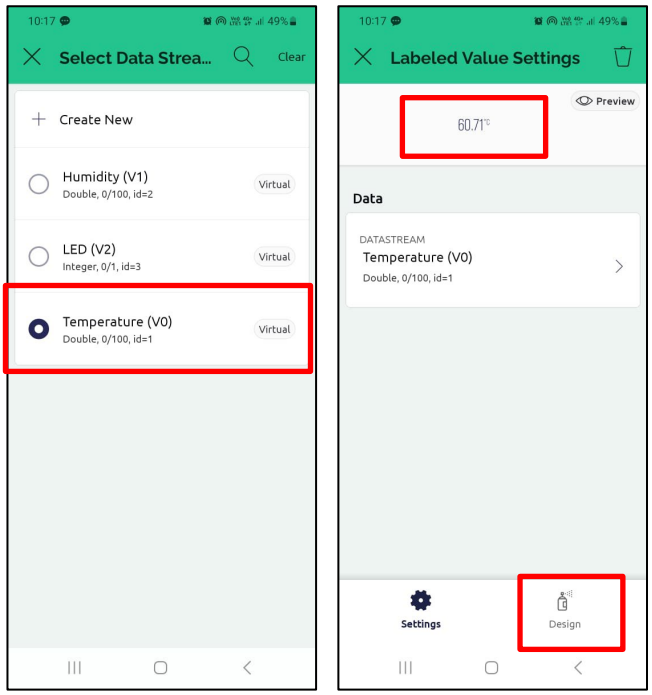


5. Click + to open the Widget Box. Scroll down and search for Labeled Value. Click to add the widget to the canvas.

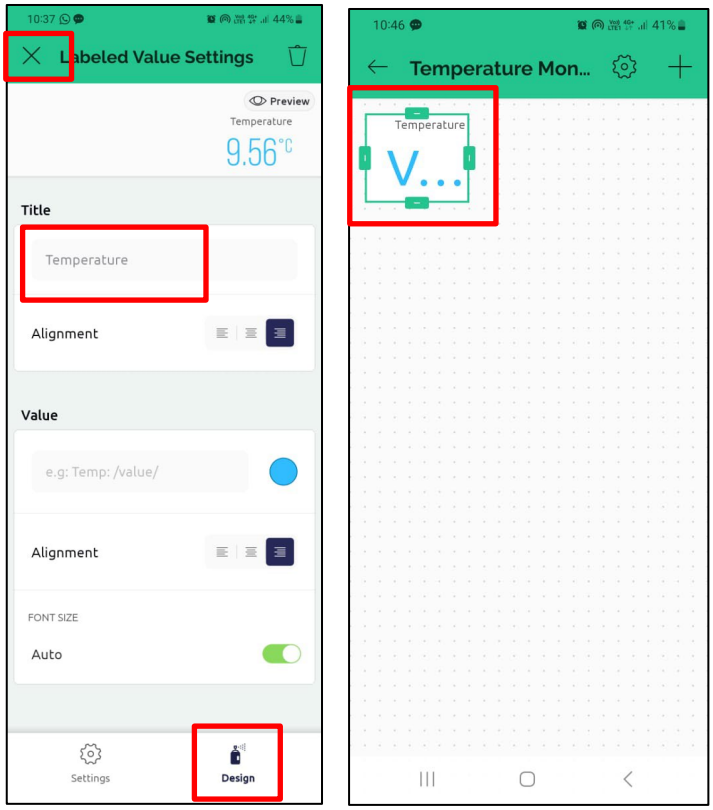


6. Click the widget on the canvas to configure the settings.
7. Click ***Choose datastream..***
8. Select ***Temperature (V0)*** and see the *Preview*. If you want to change the appearance of your widget, click ***Design***.

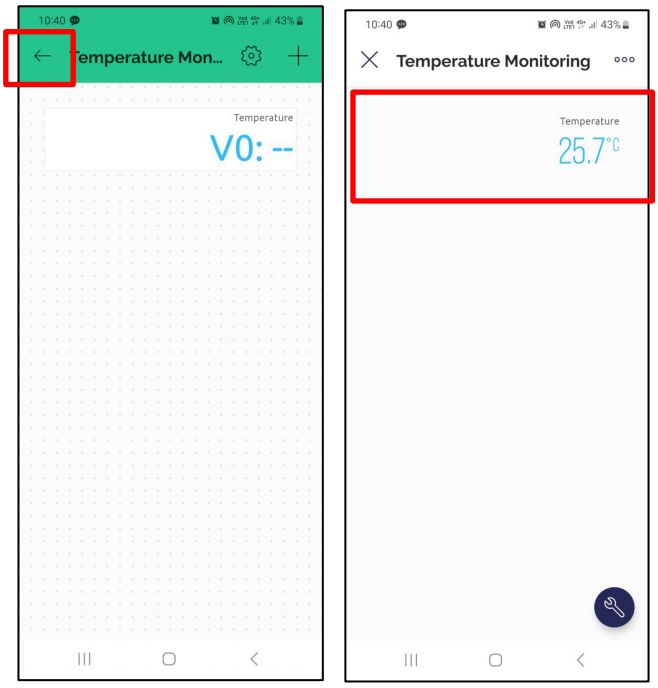




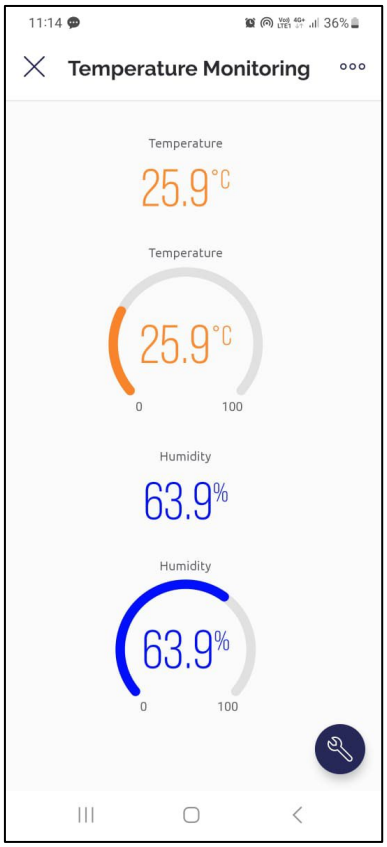
9. Set the configuration for title and value of the widget. Set font size to **Auto**.
10. Click **X** to exit. Settings made will be applied.
11. **Short pressed and release** the widget to show the bounding box. Resize the widget by dragging the sides of the box.
12. **Long press and drag** the widget to reposition the widget.



13. Click the left arrow icon to exit the Developer Mode and view the dashboard.



14. Using the same steps, configure the Labeled value widget for Humidity and Gauge Widget for both Temperature and Humidity.



Congratulations! You have successfully created an IoT Monitoring system using Blynk.