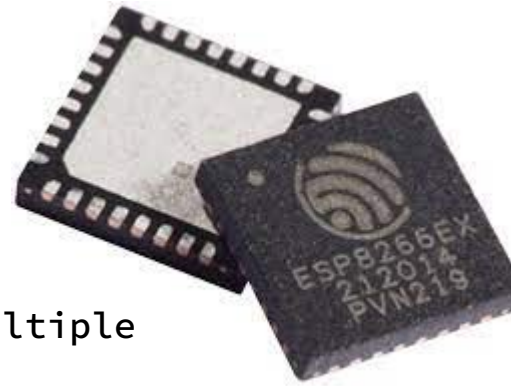
NodeMCU/ESP8266

Exploring the World of IoT

What is ESP8266 ?



The ESP8266 is a 32-Bit System on a Chip (SoC).

SoC

- Soc is IC (integrated Circuit) that incorporates multiple electronic components.
- These components and subsystems typically include
  Processor
  Memory units
  Input/output interfaces
  Timer
  and various other peripherals.

It is manufactured by the Chinese company Espressif

ESP-12E module

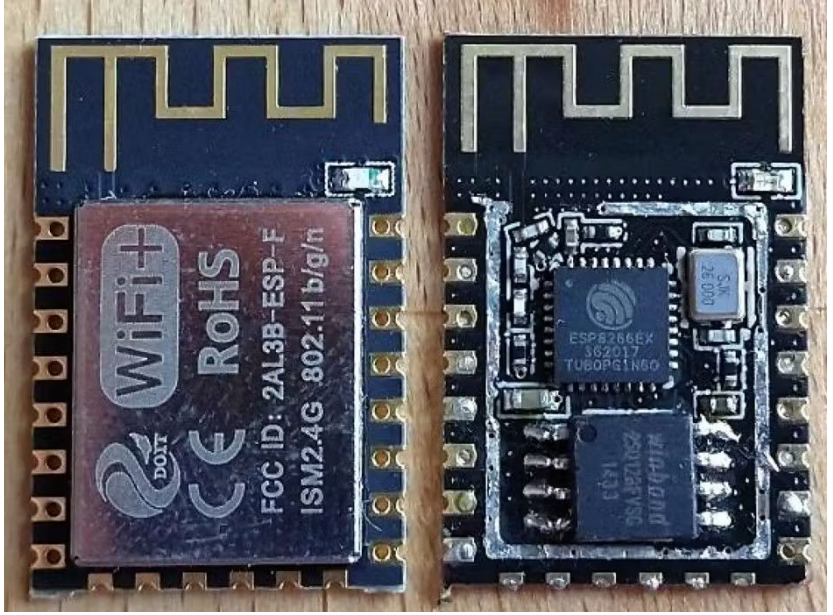
ESP-12E is Wi-Fi module based on ESP8266 SoC.

It is also known as ESP8266-12E.

It has antena and other necessary components like External SPI flash, GPIO pin, LED.

ESP-12E module was made by AI-Thinker.

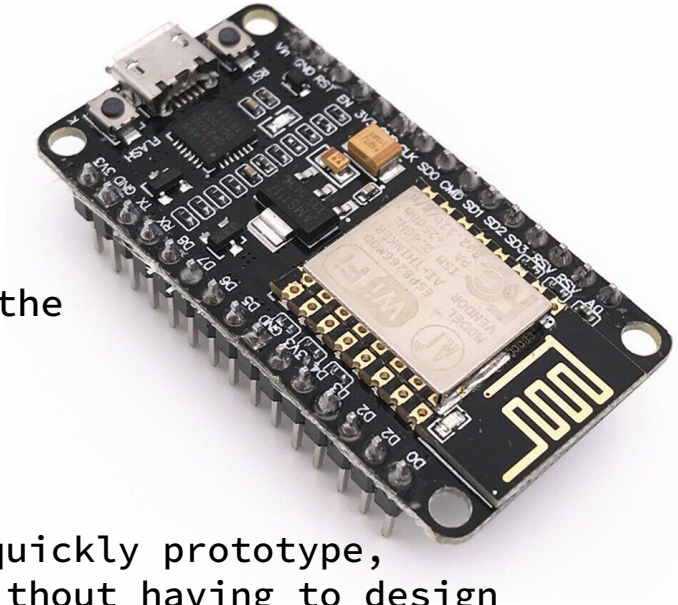
**To program this module external Programmer need to be connect.



What is NodeMCU ?

The NodeMCU is a Open source popular IoT development board.

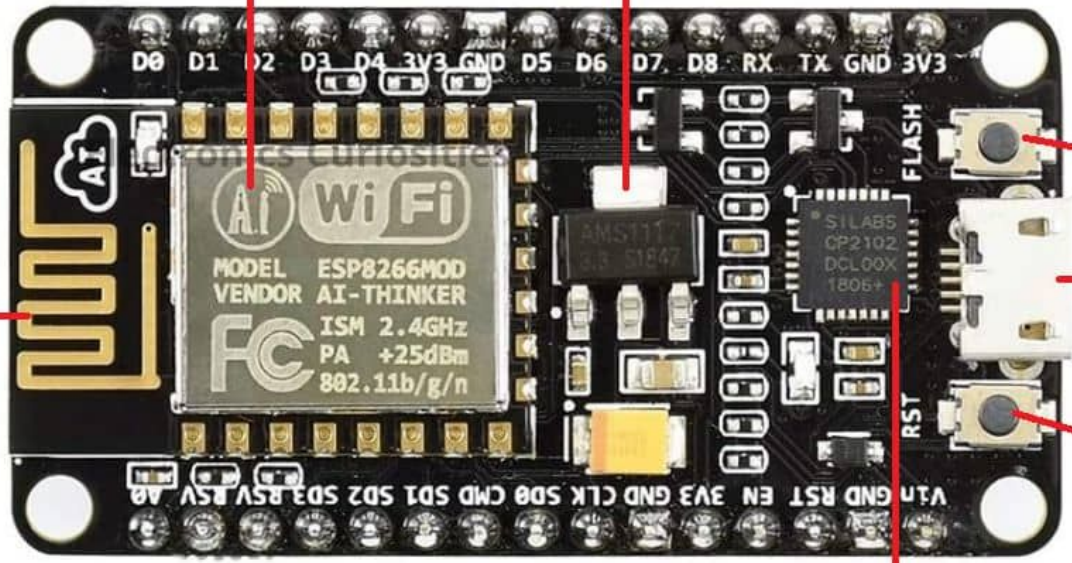
It is based on the ESP-12E module (which contains the ESP8266 SoC).



What is Development Board ?

- Development boards provide an environment to quickly prototype, experiment, and develop electronic projects without having to design and manufacture custom PCB.

ESP8266 NODE MCU



ESP8266-12E
Wi-Fi Chip

3.3V Voltage Regulator IC

Flash Button

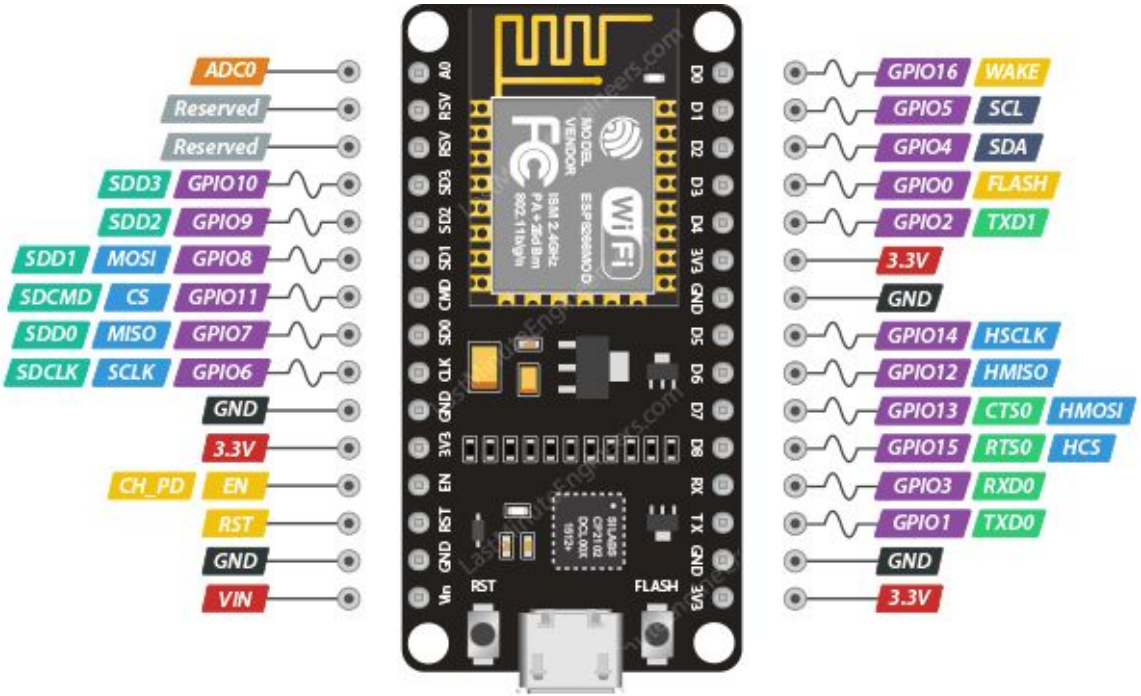
Micro USB
Connector

Reset Button

2.4GHz
Antenna

USB to TTL
Converter IC

NodeMCU Pinout



GPIO	Pins
GPIO 0	D3
GPIO 1	D10
GPIO 2	D4
GPIO 3	D9
GPIO 4	D2
GPIO 5	D1
GPIO 12	D6
GPIO 13	D7
GPIO 14	D5
GPIO 15	D8
GPIO 16	D0
ADC 0	A0

About NodeMCU

Microcontroller : ESP8266 (32 Bit Microcontroller)

Clock Speed : 80Mhz

SRAM : 64 KB

Flash Memory : 4 MB

Digital Pins : 9
Analog Pin : 1

Input Voltage : 3.3V - 3.6 V

Communication Protocol : UART, SPI, I2C

WiFi : 2.4Ghz with IEEE 802.11 b/g/n

Built In LED :

For ESP12e built in LED at
GPIO 2 (D4) {ACTIVE LOW}

For NodeMCU built in LED at
GPIO 16 (D0) {ACTIVE LOW}

ADC: 10 Bit

PWM : 10 Bit

NodeMCU Programming

Following are some of the ways to program NodeMCU

1. Lua Scripting language
2. Arduino IDE
3. MicroPython
4. PlatformIo
5. Espressif IDE

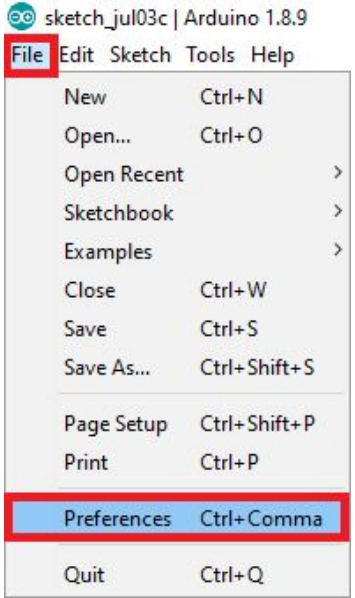
NodeMCU Programming with Arduino IDE

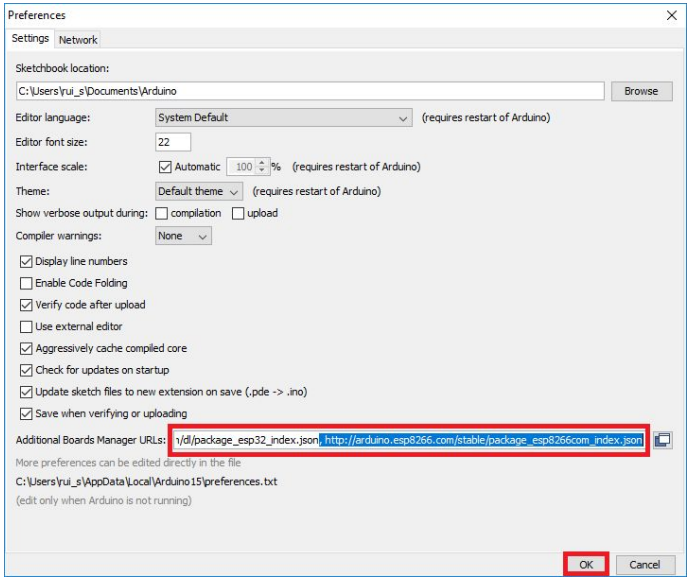
Step 1:

Download Arduino IDE (https://www.arduino.cc/en/software)

Step 2:

Open Arduino IDE -> Go to Files -> Preferences , Past the URL (http://arduino.esp8266.com/stable/package_esp8266com_index.json)

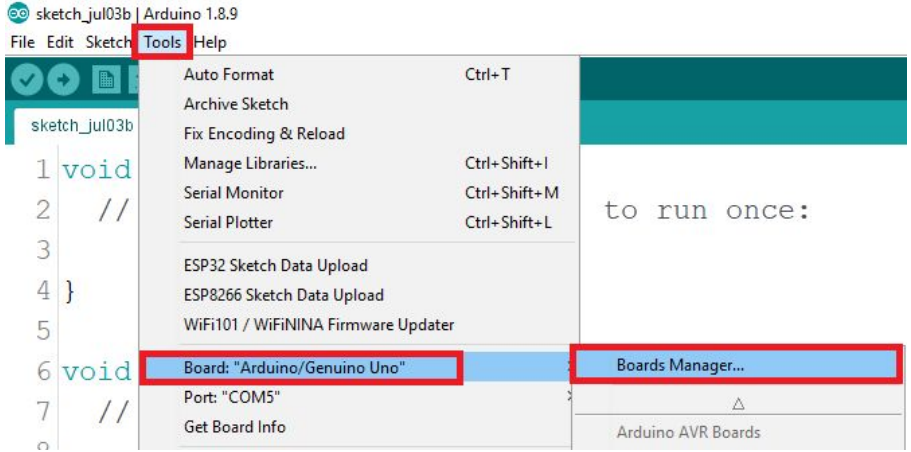




NodeMCU Programming with Arduino IDE

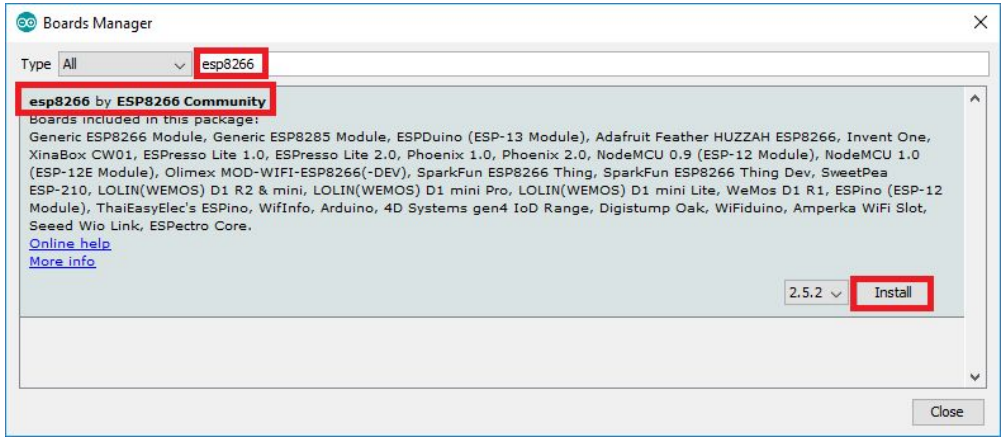
Step 3:

Tools > Board > Boards Manager…



Step 4:

Search for **ESP8266** and press install button for the **“ESP8266 by ESP8266 Community”**

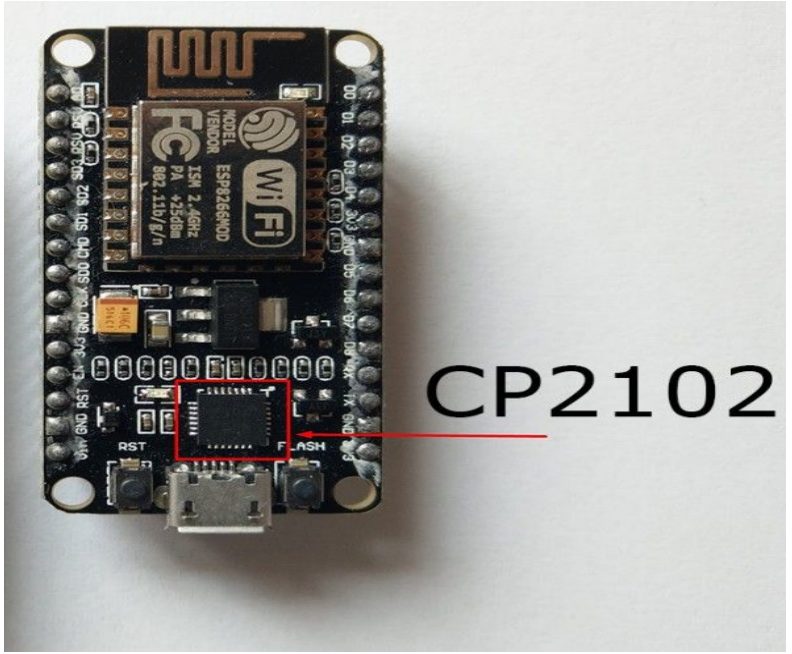


Drivers for NodeMCU

1. Download CP210x USB to UART Bridge VCP Drivers (https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads)

2. Unzip the folder

3. Right click on the silabser.inf file and select Install



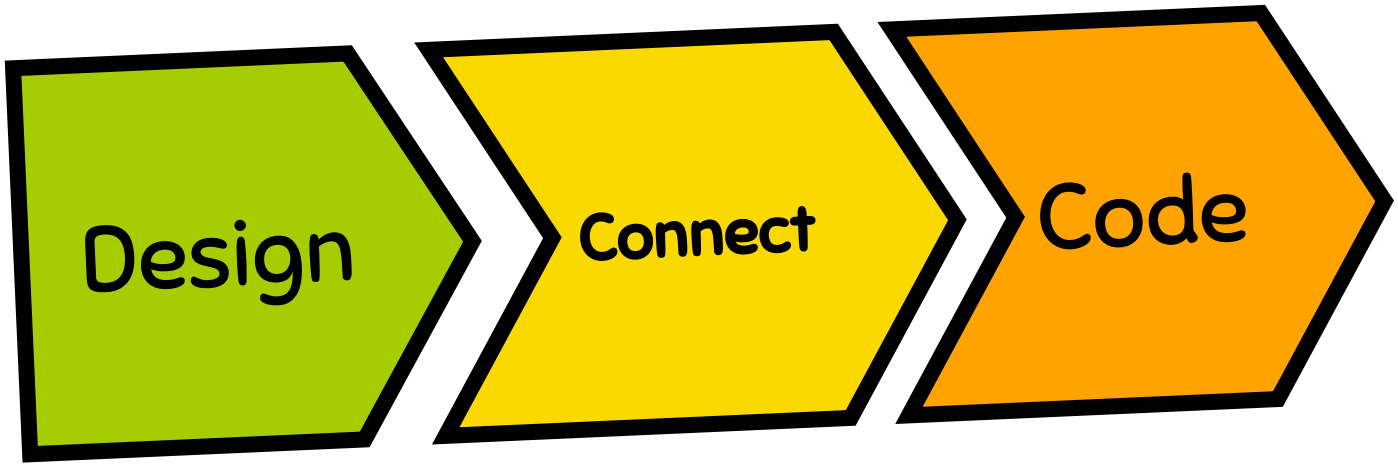
Drivers

Drivers is software components which interface between the hardware and the operating system or application software.

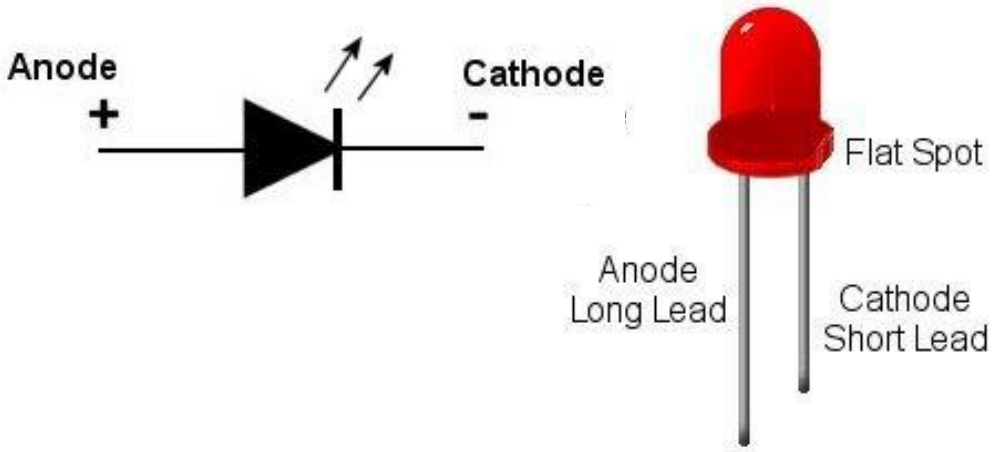
Their primary function is to enable communication and control of various hardware peripherals.

IT'S TIME TO HANDS-ON

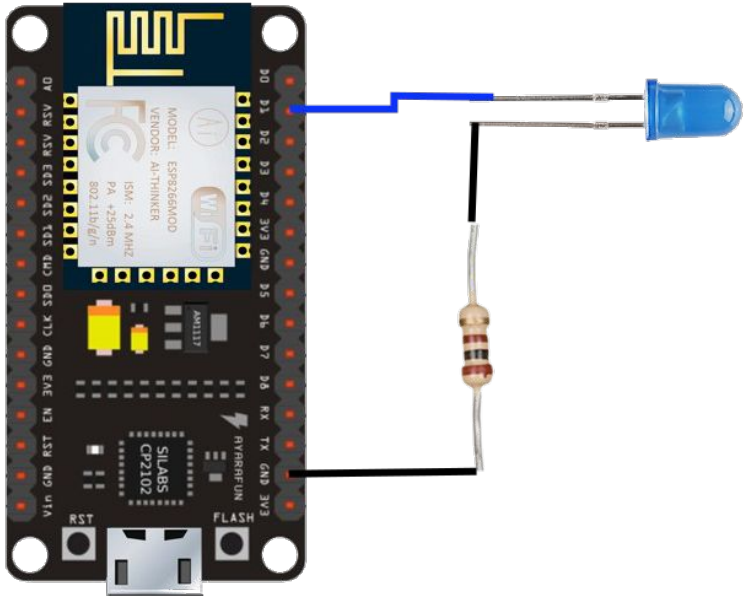
OUR PROCESS IS EASY



LED



Design


MODEL: ESP8266MOD
VENDOR: AI-THINKER
ISM: 2.4 MHZ
PA +25dBm
802.11b/g/n
AM1117
SILABS
CP2102
RST
FLASH

Can you calculate Resistance ?

Step 1:

Find out the **Forward voltage drop** (VF) and Maximum **forward current** (IV).

Given in datasheet.

Step 2:

Know the Power supply (VS)

Step 3:

R= (VS-VF)/IV

Step 4:

Choose nearest standard value.

Basics of Programming

```
void setup()
{
        pinMode(pin_number, INPUT/OUTPUT);
        Serial.begin(Baud_rate);
}

void loop()
{
    digitalWrite(pin_number,HIGH/LOW);
    digitalRead(pin_number);
     analogWrite(pin_number, value);
     analogRead(pin_number);
    delay(ms);
}
```

pinMode()

It is used to select the pin and set the direction of pin.

pinMode(pin_number, **INPUT/OUTPUT**);

pin_number : D0,D1,D2,D3,D4,D5,D6,D7,D8, A0

digitalWrite()

It is used produce an output to the digital I/O pin.

digitalWrite(pin_number, **HIGH/LOW**);

Pin_number : D0,D1,D2,D3,D4,D5,D6,D7,D8

HIGH : 3.3V **LOW** : 0V

Task 1. Blink the LED

```
void setup()
{
  pinMode(pin_number, OUTPUT);
}

void loop()
{
   digitalWrite(pin_number,HIGH);
    delay(1000);
   digitalWrite(pin_number,LOW);
    delay(1000);
}
```

Serial.begin()

It is used to setup the communication between IDE and NodeMCU.

It uses Serial Port UART (Universal Asynchronous Receiver Transmitter.

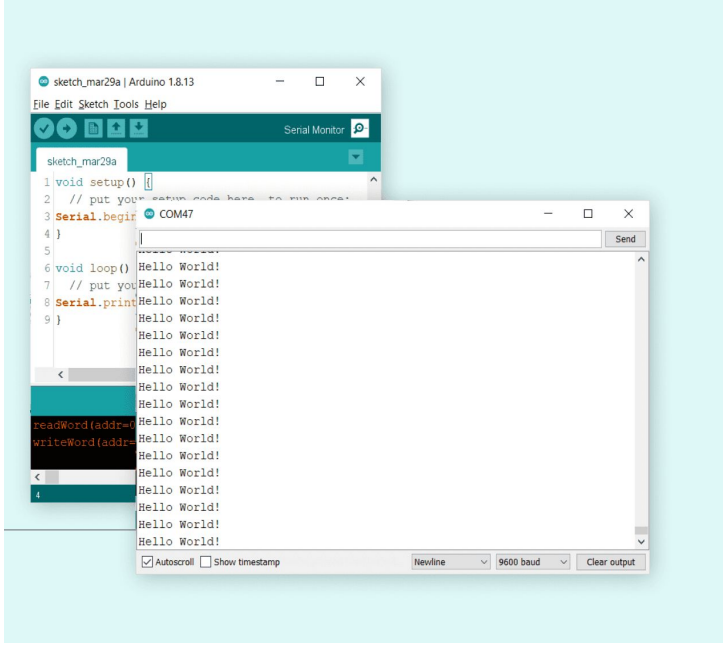
Serial.begin(Baud_rate);

Baud_rate: it number of bits transmitted/received per second

Serial.print()

This function is used to print the output on serial monitor.

Serial.print("Your Message");
Serial.print(Sensor_value);



Task 2. Send message while Blinking the LED

```
void setup()
{
   pinMode(pin_number, OUTPUT);
    Serial.begin(115200);
}
 void loop()
 {
   digitalWrite(pin_number,HIGH);
    Serial.println("LED is ON");
     delay(1000);

    digitalWrite(pin_number,LOW);
   Serial.println("LED is OFF");
     delay(1000);

 }
```

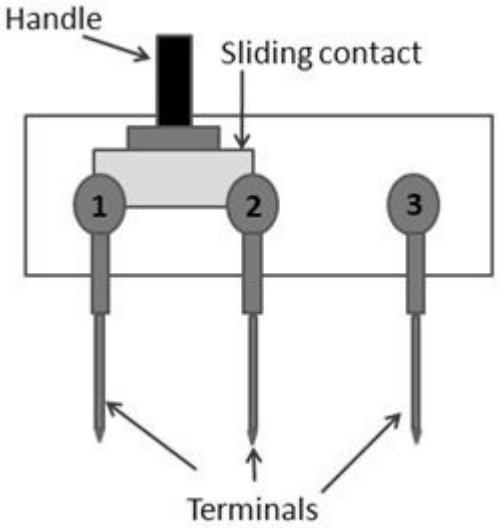
Task 3. Build Traffic light using 3 LED

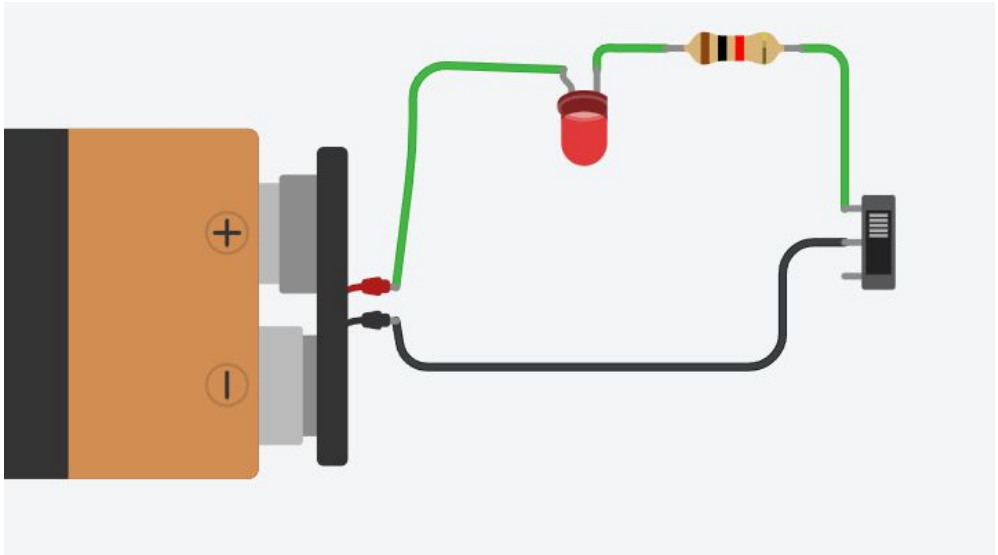
```
int Red    =  pin_number  //D0
int Yellow=  pin_number  //D1
int Green =  pin_number  //D2
```

```
void setup()
{
    pinMode(Red, OUTPUT);
    pinMode(Yellow, OUTPUT);
    pinMode(Green, OUTPUT);

}
```

```
void loop()
{
    digitalWrite(Red,HIGH);
    delay(10000);
    digitalWrite(Red,LOW);

    digitalWrite(Yellow,HIGH);
    delay(2000);
    digitalWrite(Yellow,LOW);

    digitalWrite(Green,HIGH);
    delay(10000);
    digitalWrite(Green,LOW);
}
```

Sliding Switch





We will connect the wire with two adjacent terminals.

If-Else

```
if (condition){

   …execute if condition is true

}

else{

  …execute if condition is not true
}
```

digitalRead()

It is used to receive the input at digital I/O pin.

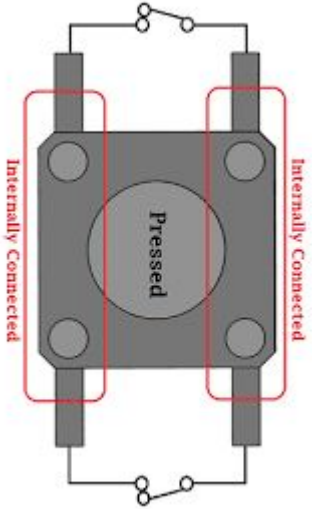
digitalRead(pin_number);

Pin_number : D0,D1,D2,D3,D4,D5,D6,D7,D8

Task 4. Print the state of sliding switch on serial monitor

```
void setup()
{
   pinMode(pin_number, INPUT);
    Serial.begin(115200);
}
 void loop()
 { If (digitalRead(pin_number)==HIGH ){
    Serial.println("LED is ON");
   }
   else {
   Serial.println("LED is OFF");
   }
 }
```

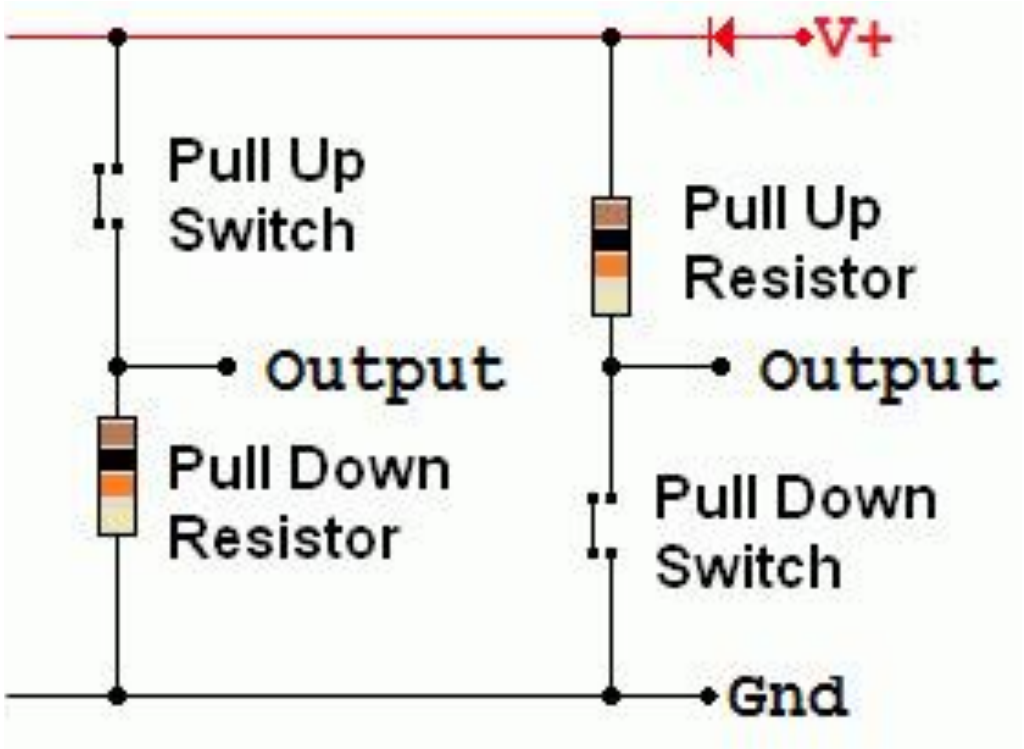
Switches



Curved legs (facing each other) are internally connected.

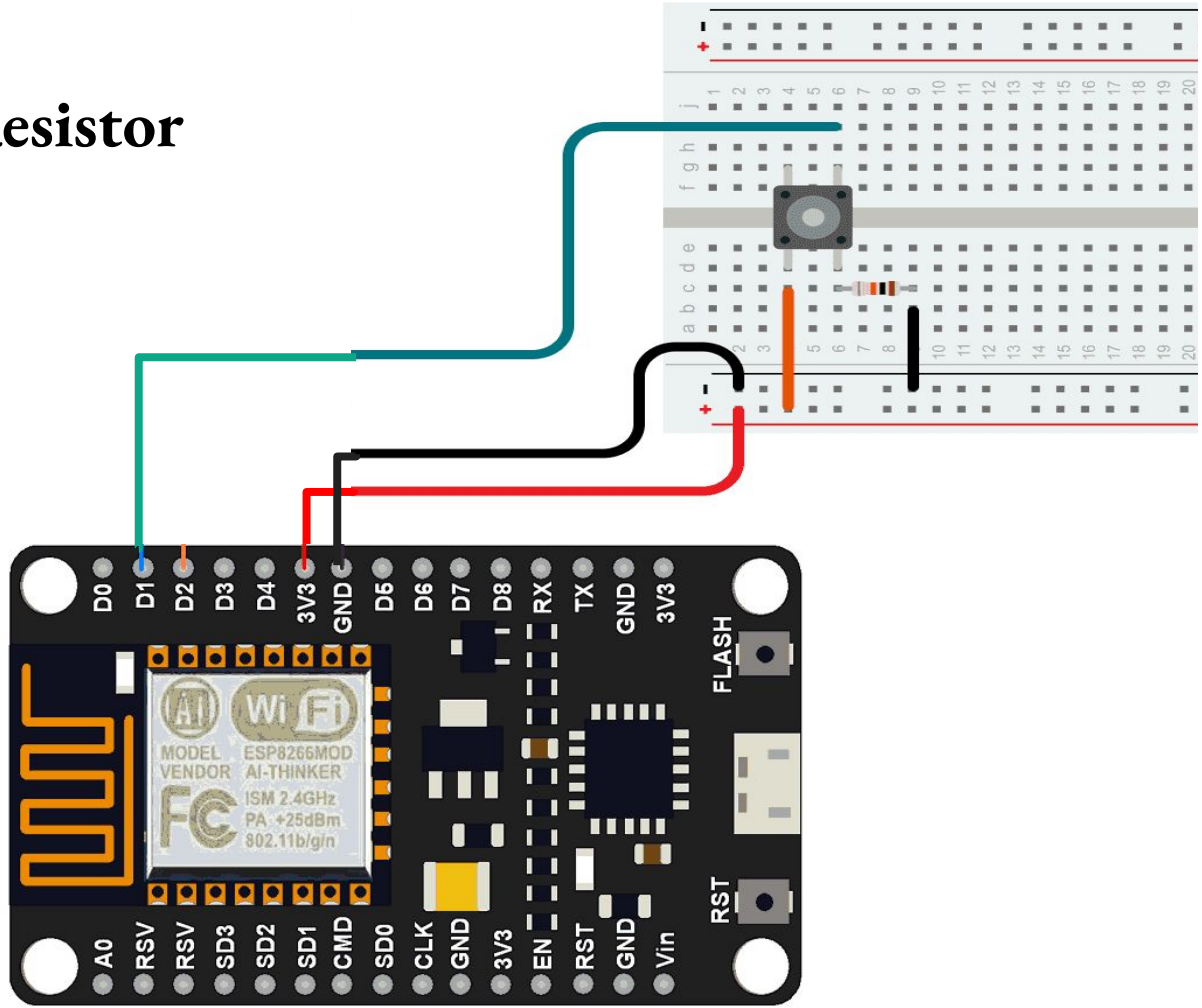
Pull Up and Pull Down Resistor

Open = 0
Close = 1



Open = 1
Close = 0

PULL Down Resistor



Task 5. Turn on and off LED using 4 leg switch.

```
const int buttonPin = D0;
const int ledPin = D4;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {

  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
     digitalWrite(ledPin, LOW);
  } else {
    digitalWrite(ledPin, HIGH);
  }
}
```

analogRead()

It is used to receive the input at analog I/O pin.

analogRead(pin_number);

Pin_number : A0

The function gives value between 0 to 1024.

The ADC present in NodeMCU is 10 bit ADC.

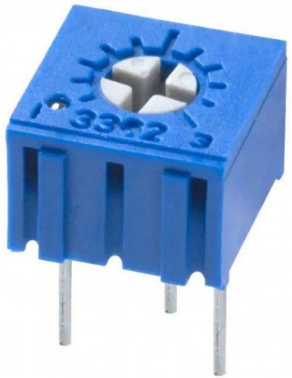
$$\text{Resolution} = \frac{V_{max} - V_{min}}{2^{10}}$$

$$\text{Resolution} = \frac{3.3v - 0}{1024} = 3.2mV$$

Potentiometer

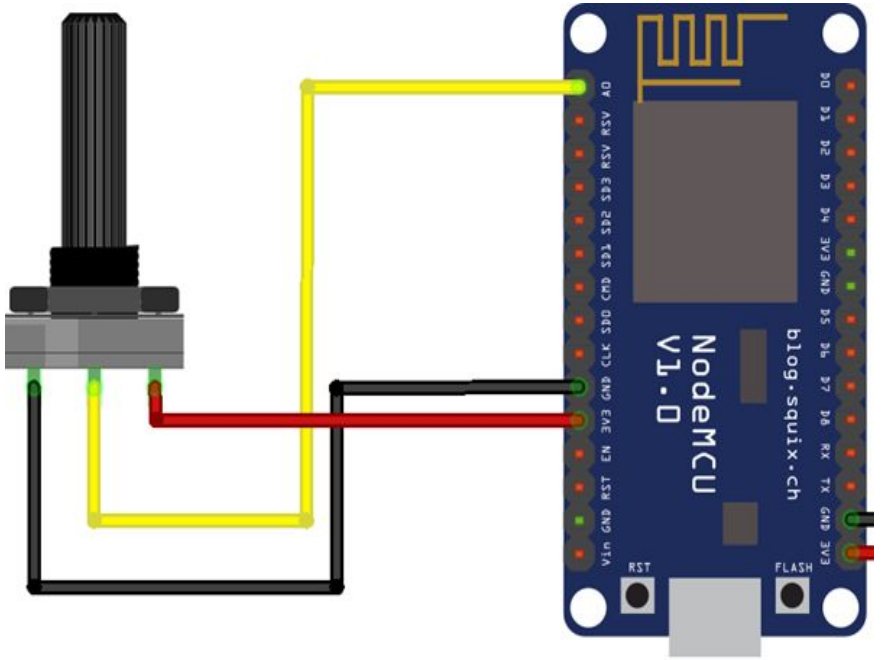
Pot, is a three-terminal device used to control the flow of electric current by varying its resistance.

it is widely used in electronic circuits as a variable resistor to adjust voltage, control volume, or set the brightness of a display, among other applications.



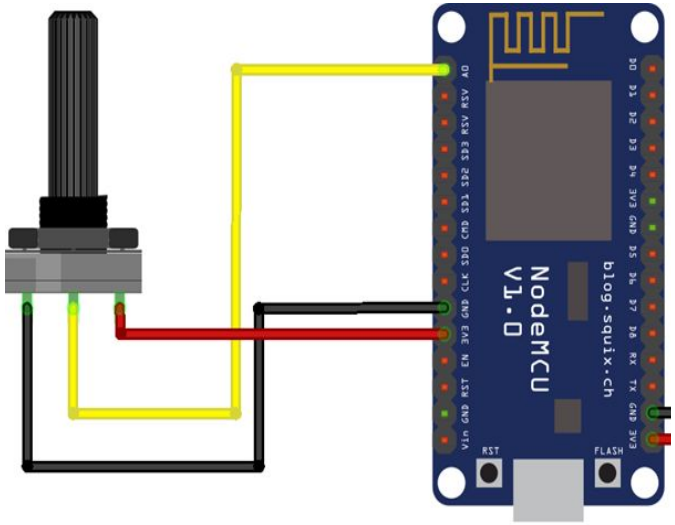
Task 6. Interface Pot with NodeMCU and print its values

```
void setup()
{
    pinMode(A0, INPUT);
    Serial.begin(115200);
}

void loop()
{
  Int sensVal= analogRead(pin_number);

  Serial.println(sensVal);
}
```



Task 7. Interface Pot with NodeMCU and print actual voltage value.

```
void setup()
{
    pinMode(A0, INPUT);
   Serial.begin(115200);
}

void loop()
{
  Int sensVal= analogRead(pin_number);
  float voltage = (sensVal*3.3)/1024;
  Serial.println(voltage);
}
```



Sensor_Value*Max_value/1024 , Sensor_Value*3.3/1024

How can get voltage values between 0V(min) and 3.3V(max)? If want 2V then.....?

analogWrite()

It is used produce an PWM signal at the GPIO pin.

analogWrite(pin_number, **value**);

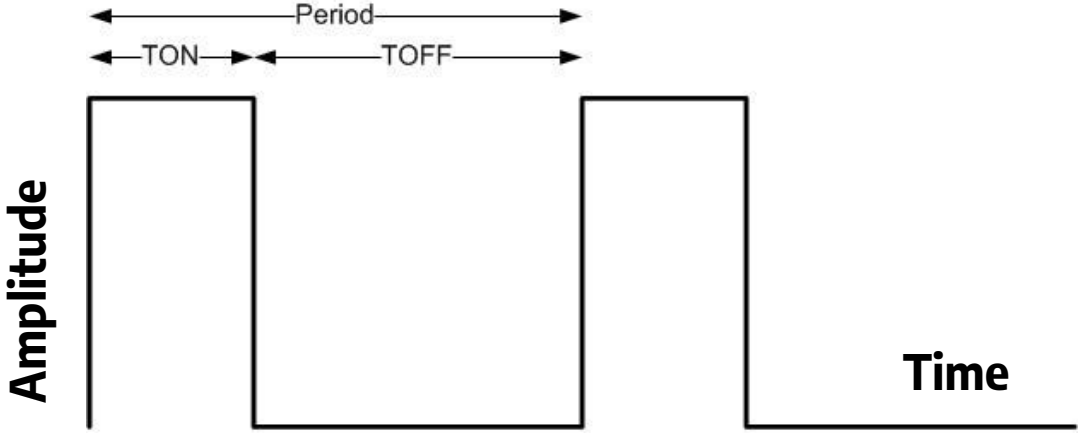
Pin_number : D0,D1,D2,D3,D4,D5,D6,D7,D8

Value is range between **(0 to 1023)**

PWM

Pulse Width Modulation

Best and efficient way of gating variable DC voltage.

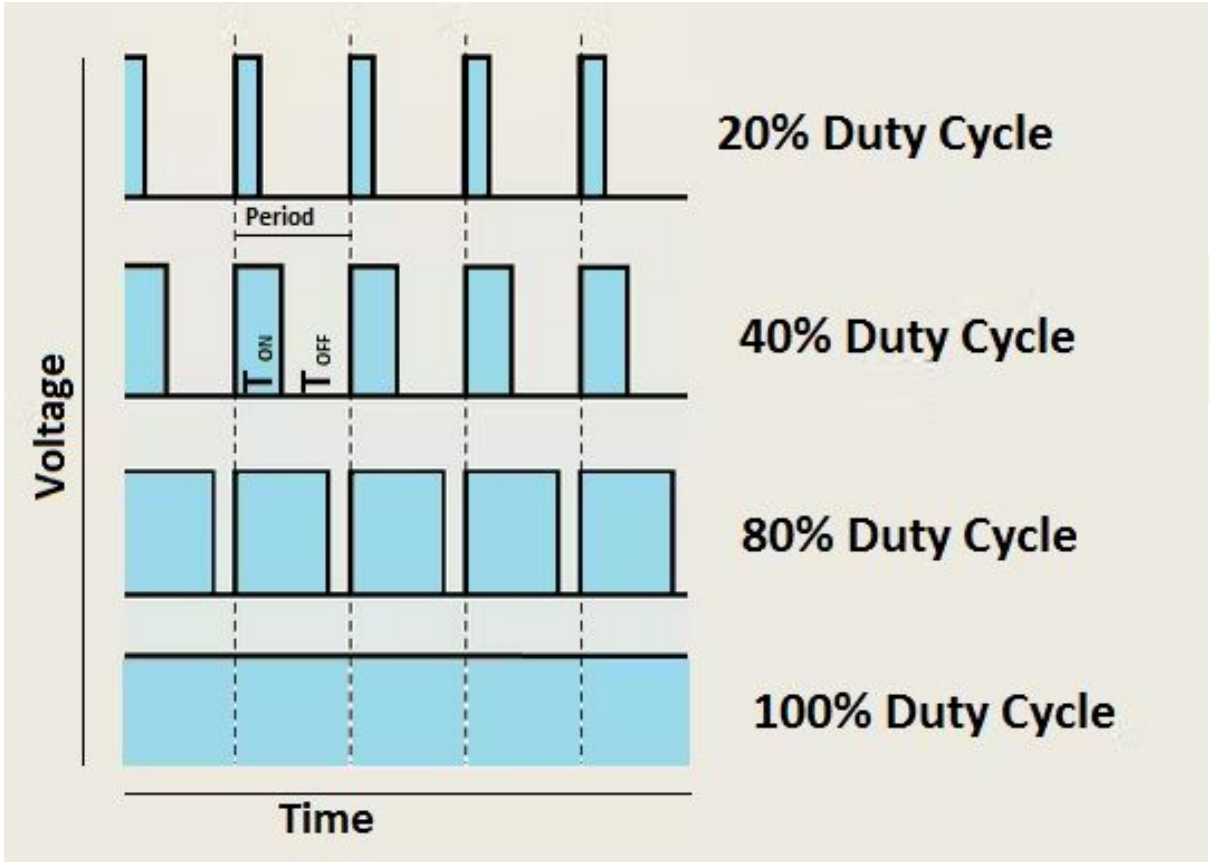


Period = TON + TOFF

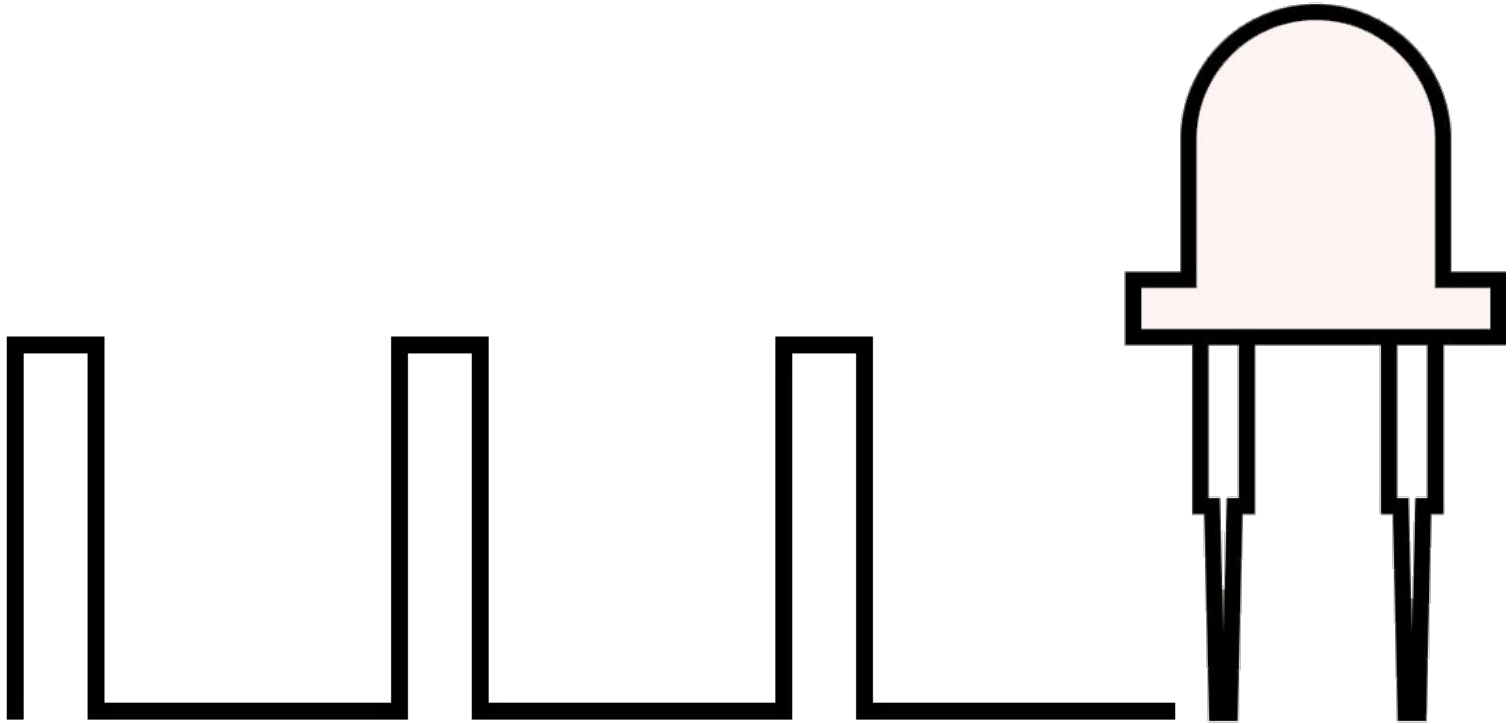
Frequency = 1 / Period

Duty Cycle = $\frac{\text{TON}}{\text{TON + TOFF}}$ *** 100**

Ton = 3.3v
Toff = 0v


Voltage
Period
T ON
T OFF
20% Duty Cycle
40% Duty Cycle
80% Duty Cycle
100% Duty Cycle
Time

Controlling the brightness of LED using PWM.



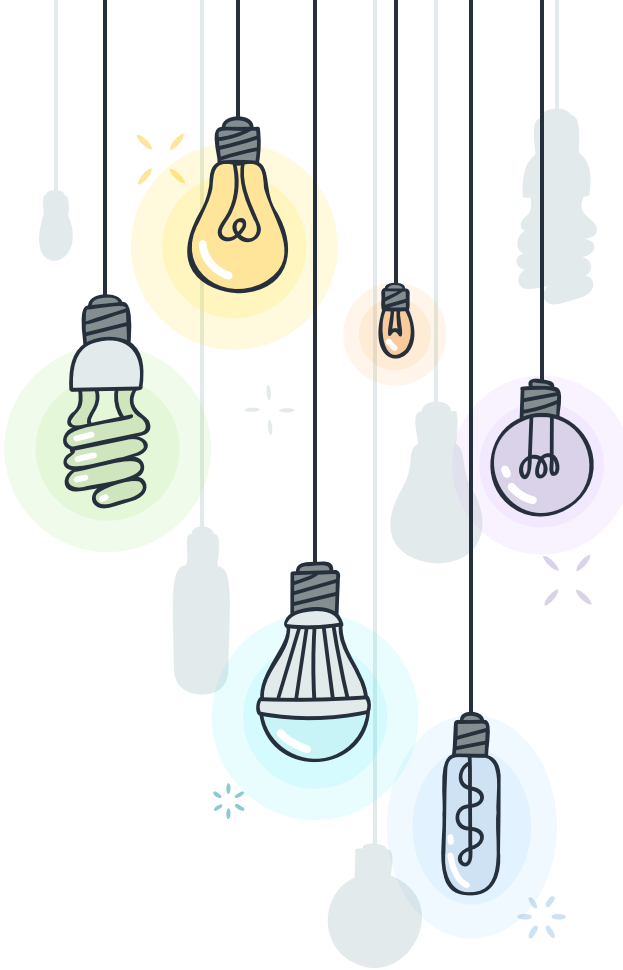
What output we get ?

If signal of duty cycle ' **D**' is given to system then system we receive D% of Amplitude

Ex. 20% of 5v is 1v
40% of 5v is 2v

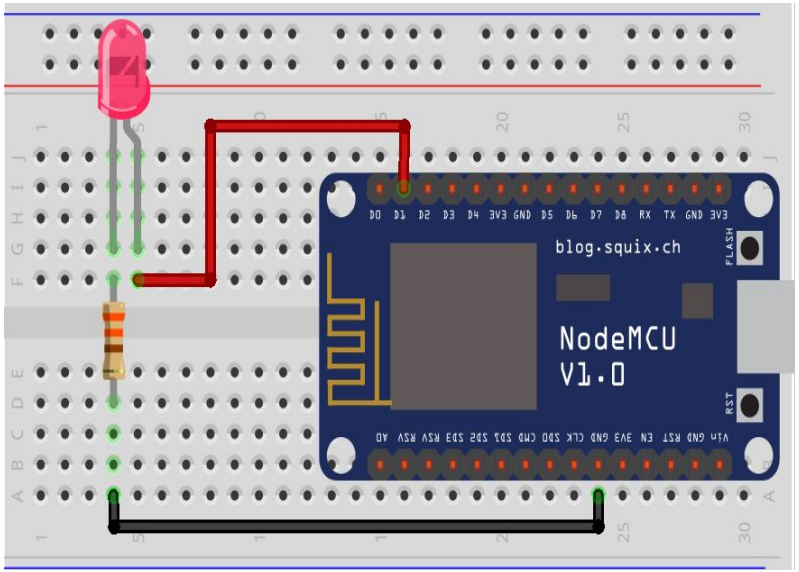
How to get such signal ?

analogWrite(pin, 0-1023)
0 -> 0% Duty cycle
512 -> 50% Duty cycle
1023 -> 100% Duty cycle



Task 8. Interface LED with NodeMCU and control the brightness with PWM.

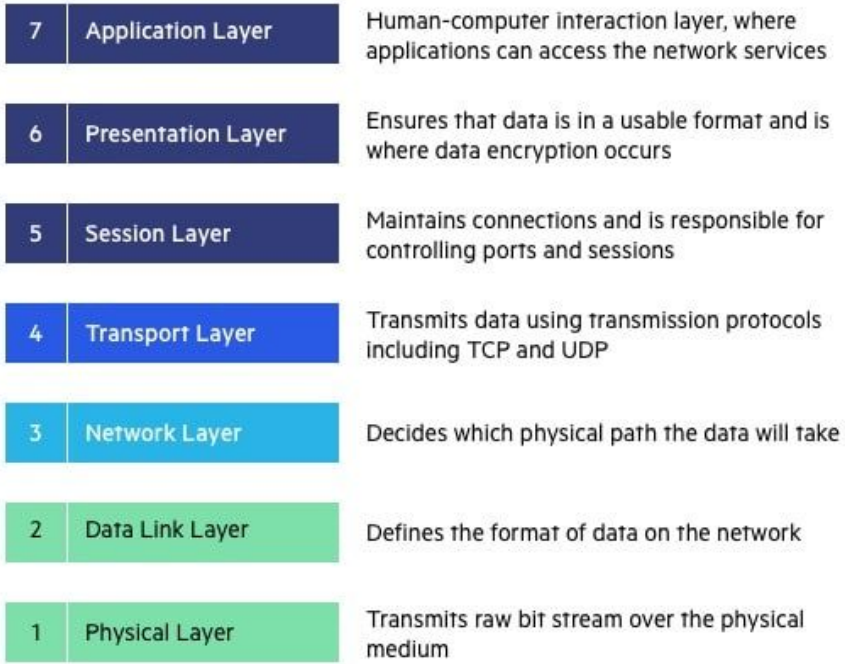
```
void setup() {
  pinMode(D3,OUTPUT);
  Serial.begin(115200);
}

void loop() {
  for(int i=0;i<1024;i++){
    analogWrite(D3,i);
    Serial.println(i);
  }
  for(int i=1024;i>0;i--){
    analogWrite(D3,i);
    Serial.println(i);
    }
  }
```




```
for( initial_value; required_value;step){


}
```

OSI Model

The OSI (Open Systems Interconnection) model is like a set of rules that computers use to talk to each other over the internet or any network. It's like a blueprint that helps organize how data travels between computers.



The TCP/IP stack

'The Internet' isn't just one protocol: It is an entire stack of layers of protocols, often referred to as the TCP/IP stack.

Layer	Protocols
Application	HTTP, FTP, mDNS, WebSocket, OSC ...
Transport	TCP, UDP
Internet	IP
Link	Ethernet, Wi-Fi ...

Link Layer

The link layer contains the physical link between two devices.
for example, Ethernet cable or a Wi-Fi connection
This is the layer that is closest to the hardware.

Internet Layer

Every device on the network has a personal IP address. It is given by Internet Protocol(IP).
There are two versions of the Internet Protocol: IPv4 and IPv6.
The IP address consists of 4 numbers,
for example 192.168.1.5 is a valid IPv4 address.
It actually consists of two parts:
first part is 192.168.1, this is the address of the local network.
last digit, 5 in this case, is specific to the device.

By using IP addresses, we can find the ESP8266 on the network, and send messages to it. The ESP can also find our computer or our phone, if it knows their respective IP addresses.

WiFi

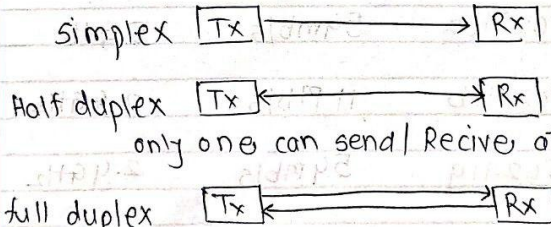
WiFi → Wireless fidelity.

WiFi/WLAN is Network where device are connected without wires (No cables)
Wireless mainly used for end device connectivity.

WLAN are Not so secure, not so fast & Not so reliable as compared to wired Network.

Official standard ^IEEE 802.11

It is mostly half duplex.



WiFi use (ISM, industrial, scientic, Medical) without any licence.

SSID (Service set Identifier)
It is friendly name of WiFi.

WiFi Routers
has some wired port for client and WAN (wide Area Network) port for providing or connecting with internet.

router provides WLAN security
DHCP service
DNS service
(Public → PrivateIP) NAT (Private → Public IP)
QoS.

1997	802.11	2Mb/s	2.4 GHz	20m
1999	802.11a	54mb/s	5 GHz	35m
1999	802.11b	11 Mb/s	2.4 GHz	35m
2003	802.11g	54 Mb/s	2.4 GHz	70m
2009	802.11n (WiFi 4)	600 mb/s	2.4/5 GHz	70m
2013	802.11 ac (WiFi 5)	6.9 Gb/s	5 GHz	35m
2019	802.11 ax (WiFi 6)	9.6 Gb/s	2.4/5 GHz	9m

WiFi 4 → 802.11n
WiFi 5 → 802.11ac
WiFi 6 → 802.11ax

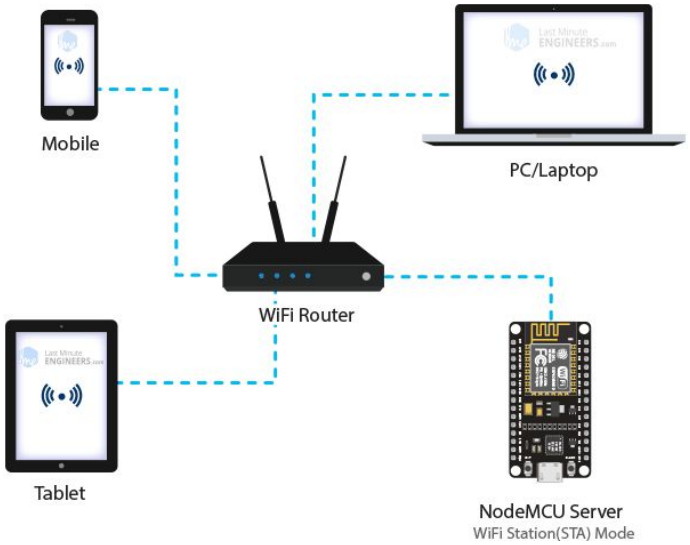
2.4 GH$_2$	5 GH$_2$
14 channel total	24 channel total
(max 3 nonoverlapping) at a time	all 24 are nonoverlap -ping at any time.
Slow data rate	Fast data rate
covers long distance	covers short distanc
IEEE 802.11 b/n/g/ax	IEEE 802.11 a/n/ac/ax

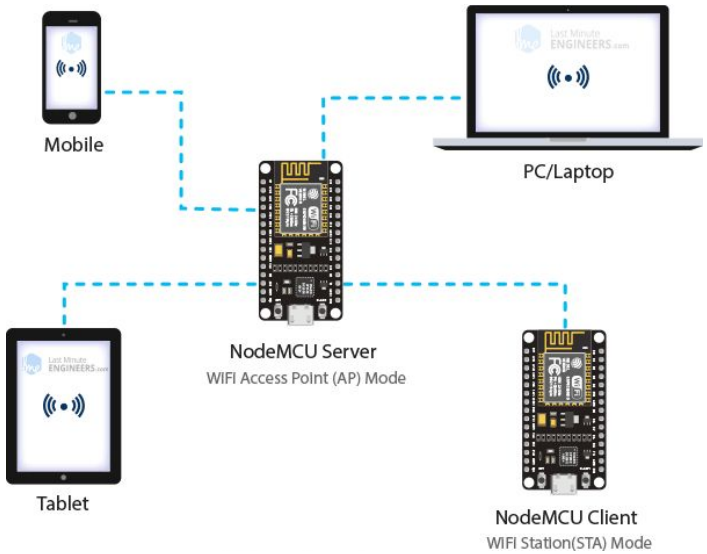
WPA: WiFi Protected Access.
WPA 2 (2003)
WPA 2 (802.11i) 2005 (AES)
WPA 3 (2018) AES.

Wi-Fi in NodeMCU

ESP8266 can operate in three modes:

1. **Station (STA) mode**
2. **Soft Access Point (AP)**





Let's connect NodeMCU with Wi-Fi

To access the Wi-Fi, header file is "ESP8266WiFi.h"

WiFi.mode()

This function is used to specify the mode of NodeMCU
(WIFI_STA, WIFI_AP, WIFI_OFF)

WiFi.begin()

It is used to connect NodeMCU with WiFi.

WiFi.begin("SSID", "Password")

WiFi.localIP()

It gives the IP address of NodeMCU .

WiFi.macAddress()

It gives the mac address(physical address) of NodeMCU
.

WiFi.status()

This function will give the status of the WiFi.
Weather it is connected or not.

- WL_IDLE_STATUS: The Wi-Fi is in idle mode and not connected to any network.
- WL_NO_SSID_AVAIL: The Wi-Fi is unable to find any available Wi-Fi networks to connect to.
- WL_SCAN_COMPLETED: The Wi-Fi scan for available networks has completed.
- WL_CONNECTED: The Wi-Fi is connected to a network.
- WL_CONNECT_FAILED: The Wi-Fi failed to connect to the specified network.
- WL_CONNECTION_LOST: The Wi-Fi connection was lost after a successful connection
- WL_DISCONNECTED: The Wi-Fi is disconnected.

NodeMCU as STA (Station)

Task 9 . Connect NodeMCU with your Mobile Hotspot and print IP address.

```
#include<ESP8266WiFi.h>
void setup() {
Serial.begin(115200);
WiFi.begin("Home","12345678");
while(WiFi.status()!=WL_CONNECTED){
  Serial.print(".");
  delay(100);
}
Serial.println("Successfuly Connected to Home ");
Serial.println(WiFi.localIP());
Serial.println(WiFi.macAddress());
}
void loop() {
}
```

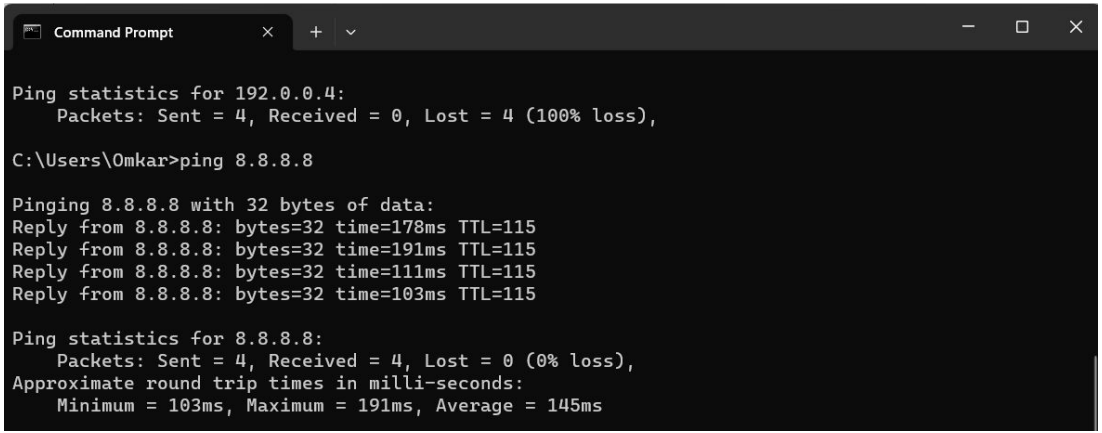
Is it really connected in LAN ?

Ping command

Ping command is used to check whether the device is connected in network or not.

Website is stored in server and server has IP address.

Use command prompt to ping the IP.



Task . Obtain an IP address of NodeMCU and try to Ping that IP.

NodeMCU as AP (Access Point)

Task 10 . Configure NodeMCU as Access Point and print IP address.

```
#include<ESP8266WiFi.h>
void setup(){
Serial.begin(115200);
WiFi.softAP("MY_ESP","12345678");
Serial.println("\n");
Serial.println(WiFi.softAPIP());
}
void loop() {

}
```

Task . Connect two devices with ESP AP and Ping the IP of each other.

ESP8266 as WebServer

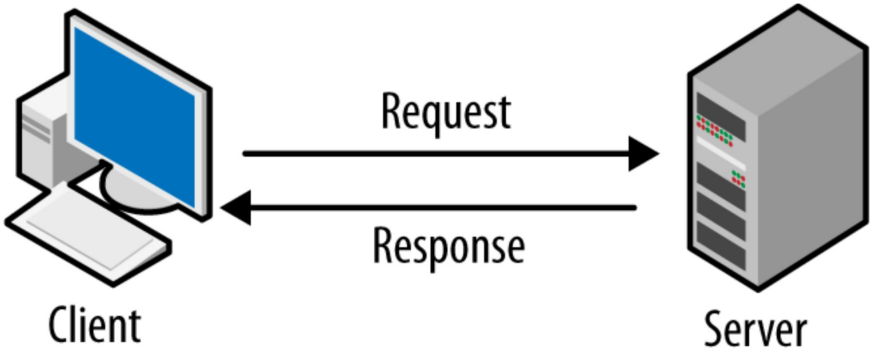
Client:

A client is a device or software application that requests and consumes services, resources, or information from a server.
Clients initiate communication with servers to obtain data or perform specific actions. Examples of clients include web browsers (like Chrome or Firefox), email clients (like Outlook or Thunderbird), mobile apps, and more.

Server:

A server is a device or software application that provides services, resources, or information to clients.
Servers are designed to respond to incoming requests from clients by delivering the requested data or performing the requested actions.
There are various types of servers, each serving different purposes.
For example, web servers host websites and web applications, email servers handle sending and receiving emails, file servers store and manage files, and game servers manage online multiplayer games.

Client-Server Architecture



1. The user enters a URL (Uniform Resource Locator) in their web browser and presses Enter.
2. The browser acts as a client and sends a request to a web server identified by the URL.
3. The web server, acting as a server, receives the request, processes it, and sends back the requested web page's content.
4. The browser, as the client, receives the response from the server and renders the web page on the user's screen.

This interaction is the basis for most online activities, from browsing the web to sending emails, accessing cloud storage, and using various online services.

Web servers

A web server is an Internet-connected device that stores and serves files. Clients can request such a file or another piece of data, and the server will then send the right data/files back to the client.
Requests are made using **HTTP**.

HTTP

HTTP or the Hypertext Transfer Protocol is the text-based protocol used to communicate with (web) servers.
There are multiple HTTP request methods, but we will only cover the two most widely used ones: **GET** and **POST**.

HTTP GET

GET requests are used to retrieve data from a server, a web page for instance. It shouldn't change anything on the server, it just gets the data from the server, without side effects.

at an example: If you click the following link: https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html , your browser will send out the following HTTP request:

```
GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.1
Host: www.w3.org
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
```

The first line is the **request line**: it contains the request method: GET, in this case, the **URI or Uniform Resource Identifier**: /Protocols/rfc2616/rfc2616-sec5.html, and the **HTTP** version: 1.1.
The second line is the host header, it specifies the domain name of the host (server).

Press F12

HTTP POST

POST requests are used to send data to the server, for example, to send your user name and password to the server when you log in, or when you upload a photo. Unlike GET, POST can change the data on the server or the state of the server. POST has a body that can contain data that is sent to the server.

The anatomy of a POST request

For example, the login page of your favorite site might send something like this when you enter your credentials and click the login button:

```
POST /login.php HTTP/1.1
Host: www.example.com
Connection: keep-alive
Content-Length: 480
Origin: http://www.example.com Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryQNEJOasMvgAOg8Kt ...
```

As you can see, the request line now has the POST method in it, and is still followed by a URI, /login.php, and the HTTP version, 1.1. The host header still contains just the domain name.

HTTP status codes

A server should answer all requests with an HTTP status code. This is a 3-digit number indicating if the request was successful or telling the client what went wrong. Here's a table with some of the most important and useful ones.

Status Code	Meaning
200	OK: the request was successful
303	See Other: used to redirect to a different URI, after a POST request, for instance
400	Bad Request: the server couldn't understand the request, because the syntax was incorrect
401	Unauthorized: user authentication is required
403	Forbidden: the server refuses to execute the request, authorization won't help
404	Not Found: the requested URI was not found
500	Internal Server Error: The server encountered an unexpected condition and couldn't fulfill the request

TCP & UDP Ports

In most cases, one device has many different services,
for example, a web server, an email server, an FTP server, a Spotify streaming service, ...
If the device had just an IP address, it would be impossible to know which application a packet was sent to. That's why every service has a **port number**.
It's an identifier for all different services or applications on a single device.
The example
the web server will only listen for requests on port 80,
the email server only on port 25,
the FTP server only on port 20,

Task 11 : Configure ESP8266 as Web Server and handle client request efficiently (respond with status code).

```
#include<ESP8266WiFi.h>
#include<ESP8266WebServer.h>
void handleRoot();
void NotRoot();
ESP8266WebServer techno(80);
void setup() {
   Serial.begin(115200);
   WiFi.begin("Home","12345678");
   while(WiFi.status()!= WL_CONNECTED){
    Serial.println(".");
    delay(100);
   }
   Serial.println("/n");
   Serial.println("Connected to Home");
   Serial.println(WiFi.localIP());
   techno.on("/",handleRoot);
   techno.onNotFound(NotRoot);
   techno.begin();
   Serial.println("Server started");
}
```

```
void loop() {
  techno.handleClient();
}

void handleRoot(){
  techno.send(200,"text/plain","hello World");
}

void NotRoot(){
  techno.send(404,"text/plain","404:Not found");
}
```

Task 12 : Configure ESP8266 as Web Server and design button on server to toggle LED on ESP8266

```
#include<ESP8266WiFi.h>
#include<ESP8266WebServer.h>
ESP8266WebServer techno(80);
void handleRoot();
void LED();
void NotRoot();
int led = D4;
void setup() {
  pinMode(D4,OUTPUT);
  Serial.begin(115200);
  WiFi.begin("Home","12345678");
  while(WiFi.status()!=WL_CONNECTED){
    Serial.print(".");
    delay(100);
  }
  Serial.println("\n");
  Serial.println("Connected to Home");
  Serial.println(WiFi.localIP());
  techno.on("/",HTTP_GET,handleRoot);
  techno.on("/LED",HTTP_POST,LED);
  techno.onNotFound(NotRoot);
  techno.begin();
}
```

```
void loop() {
     techno.handleClient();
}

void handleRoot(){
  techno.send(200,"text/html","<form action=\"/LED\"method=\"POST\">
  <input type=\"submit\"value=\"Tooggle LED\"></form>");
}

void LED(){
  digitalWrite(led, !digitalRead(led));
  techno.sendHeader("Location","/");
  techno.send(303);
}

void NotRoot(){
  techno.send(404, "text/plain","404:Not Found");
}
```

IoT Cloud Platform

IoT platforms are specialized cloud-based services that provide infrastructure and tools for managing, analyzing, and processing data generated by IoT devices.

Services offered by IoT platform

- Device Management
- Connectivity
- Data Ingestion and Storage
- Data Processing and Analytics
- Visualization and Dashboards
- Alerts and Notifications
- Security and Authentication
- Integration and APIs
- Scalability and Performance
- Cost Management
- Edge Computing

Following are some of the IoT clouds

- Amazon Web Services (AWS) IoT:
- Google Cloud IoT Core:
- Microsoft Azure IoT Hub
- IBM Watson IoT Platform
- Arduino IoT cloud
- Blynk
- Thingspeak
- Particle cloud

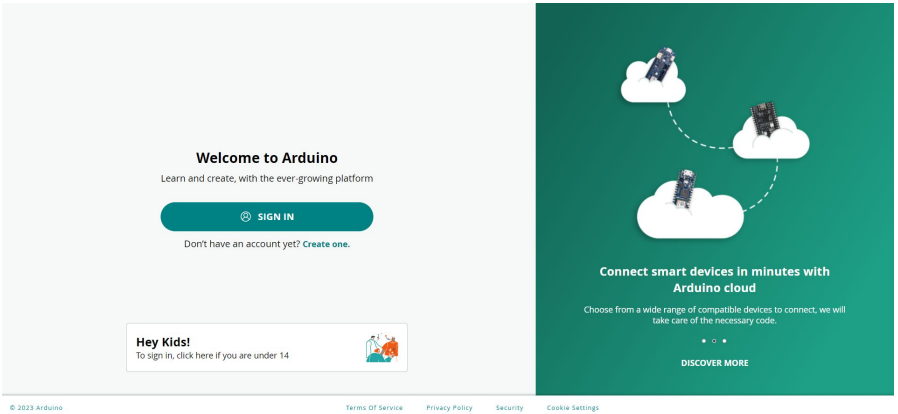
Arduino IoT Cloud (Setup)

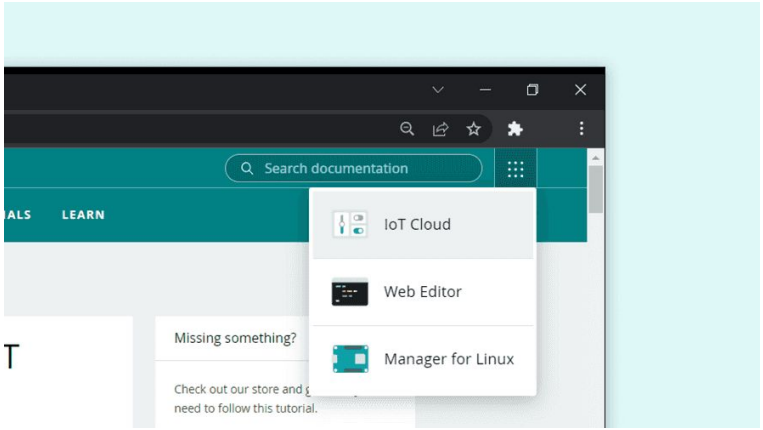
1. Creating an Arduino Account

To starting using the Arduino IoT cloud, we first need to log in or sign up to Arduino.

2. Go to the Arduino IoT Cloud

After we have signed up, you can access the Arduino IoT Cloud from any page on arduino.cc by clicking on the four dots menu in the top right corner. You can also go directly to the Arduino IoT Cloud.

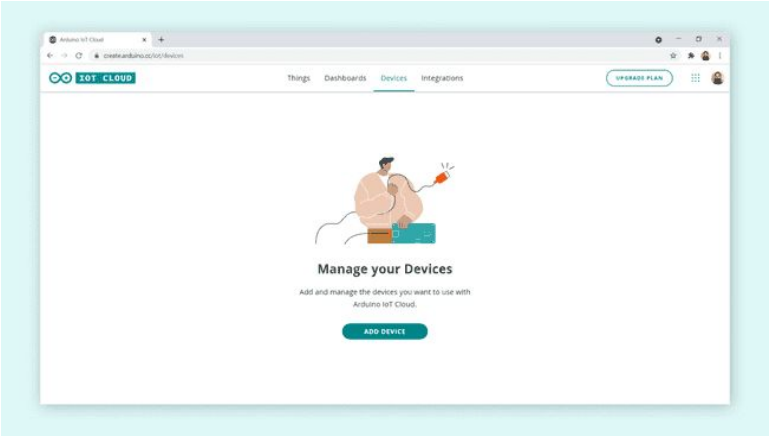


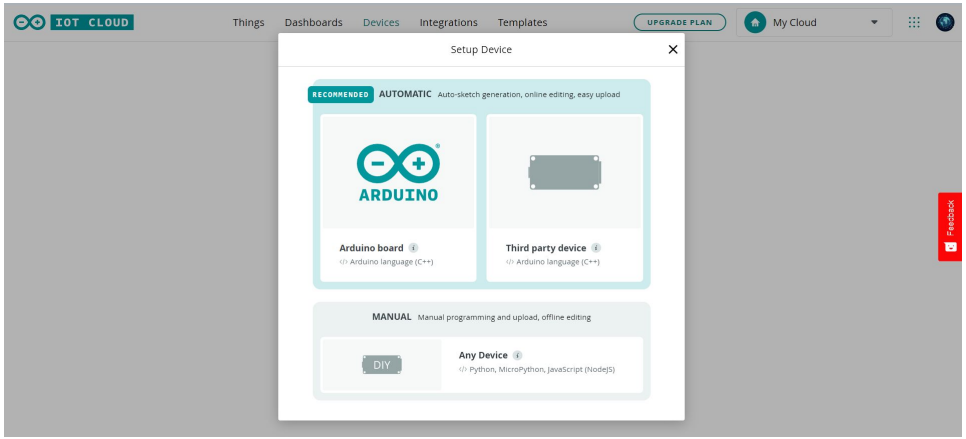


Step 1: Setting up the device

1. Once in the cloud, click on the **"Devices"** tab.

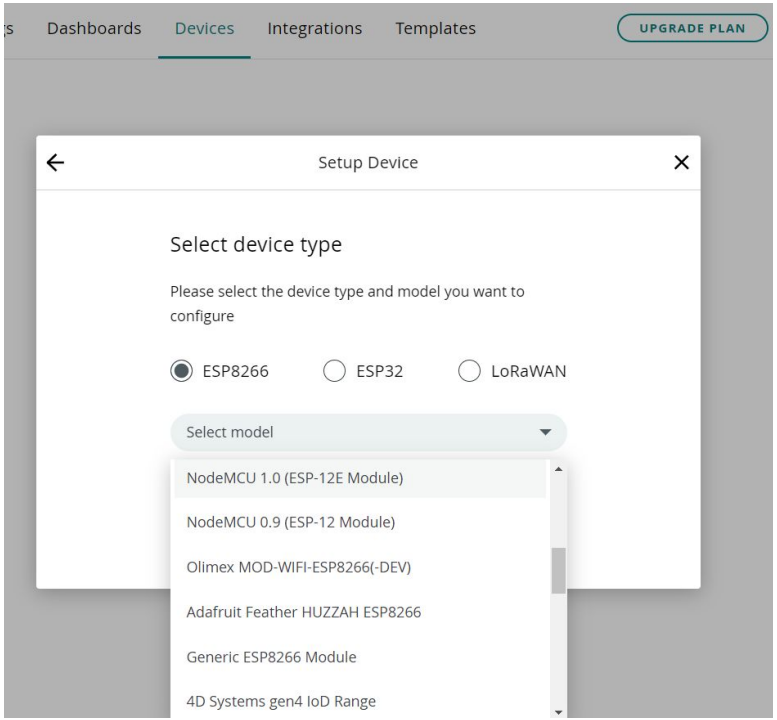
Then, click on the **"Add device"** button

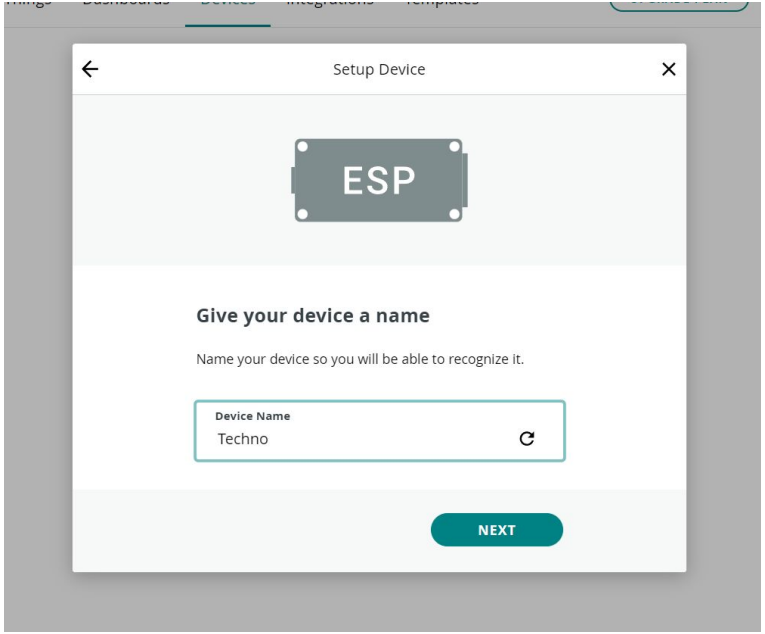




2. Then, click on **"Set up a 3rd party device"**.

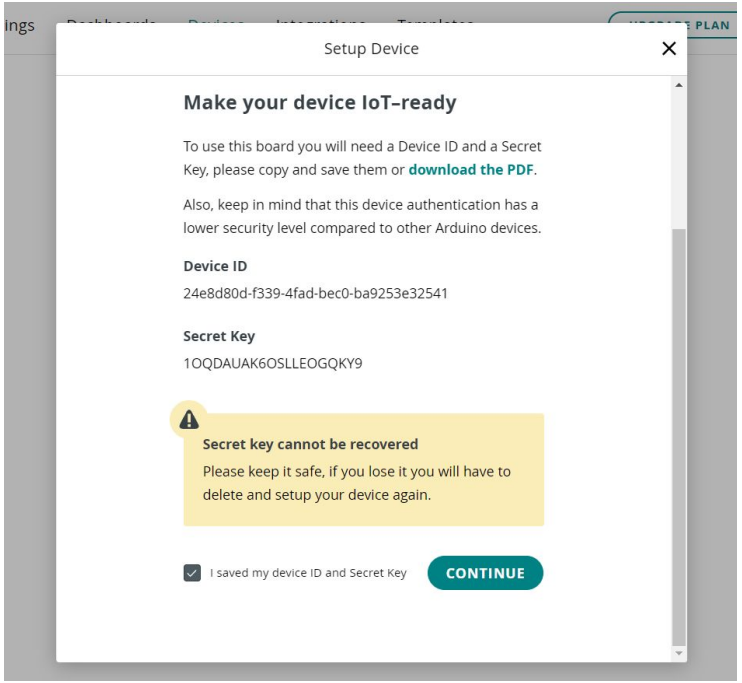
3. Now we need to select what board we are using. Select ESP8266, and then choose the board from the drop-down menu.



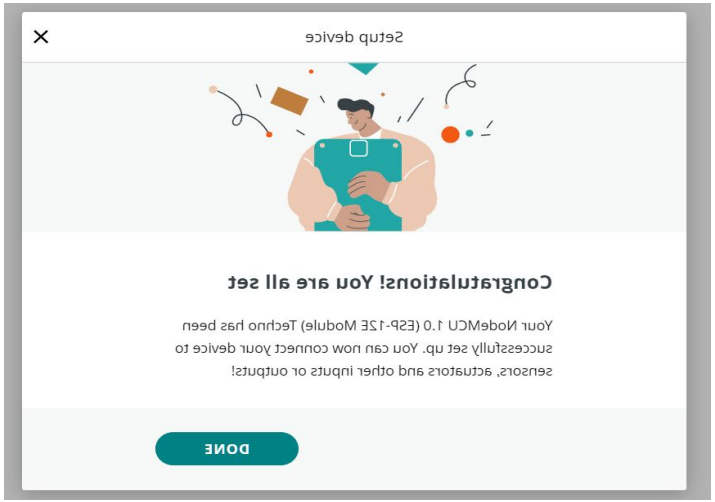


4. Now the board needs a name. You can give any name as per your choice.

5. You will now receive your **Device ID** and **Secret key**. Please note that the secret key cannot be recovered, so make sure you note it down. You can also download a PDF with the information.

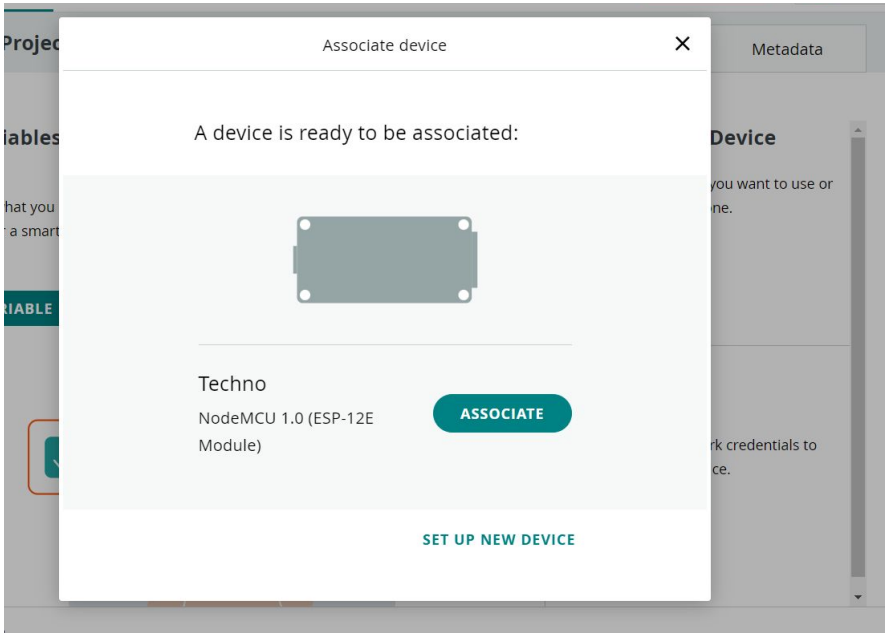


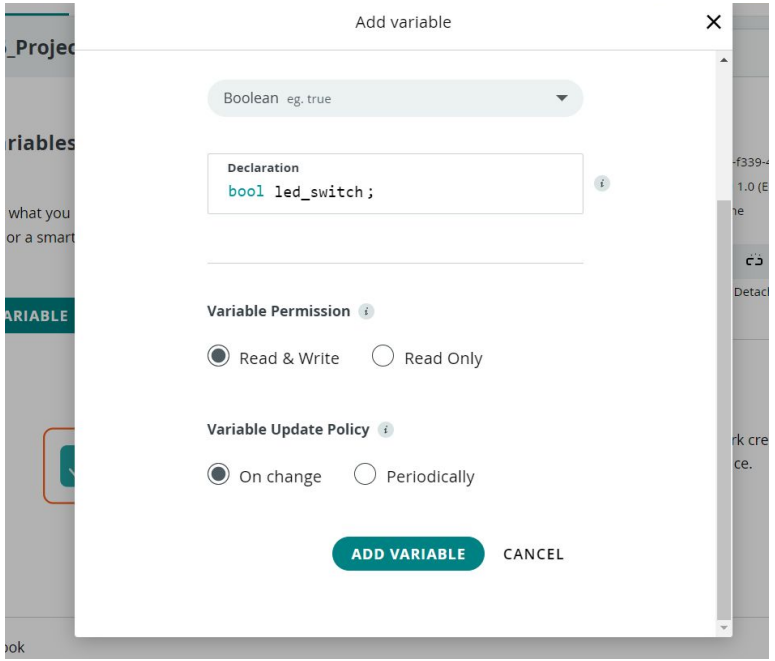
When you have saved it, tick the box at the bottom, and click on **"Continue"**.



Congratulations! You have now configured your ESP8266 board with the Arduino Cloud IoT. Your device will now appear in the list of devices.

Now, we need to link our device with our Thing.
This is done by clicking on the button in the **"Device"** section. This will open a window, where your created ESP8266 should be available to select.





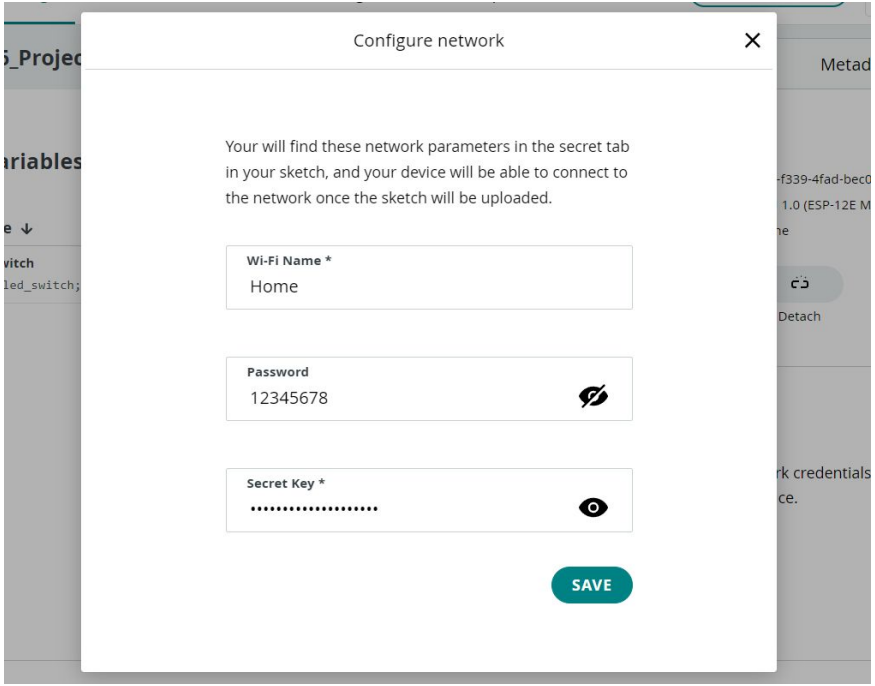
Click on **"Add variable"** button. This will open a window where you need to fill in variable information. Now, let's add the led_switch variable. The data type for this variable is **boolean**, the permission is **read-write**, and update policy is **on change**.

Once done, click on the **"Add variable"** button.

Step 3: Adding credentials

Now, we need to enter the credentials for our network, plus the secret key generated during the device configuration.

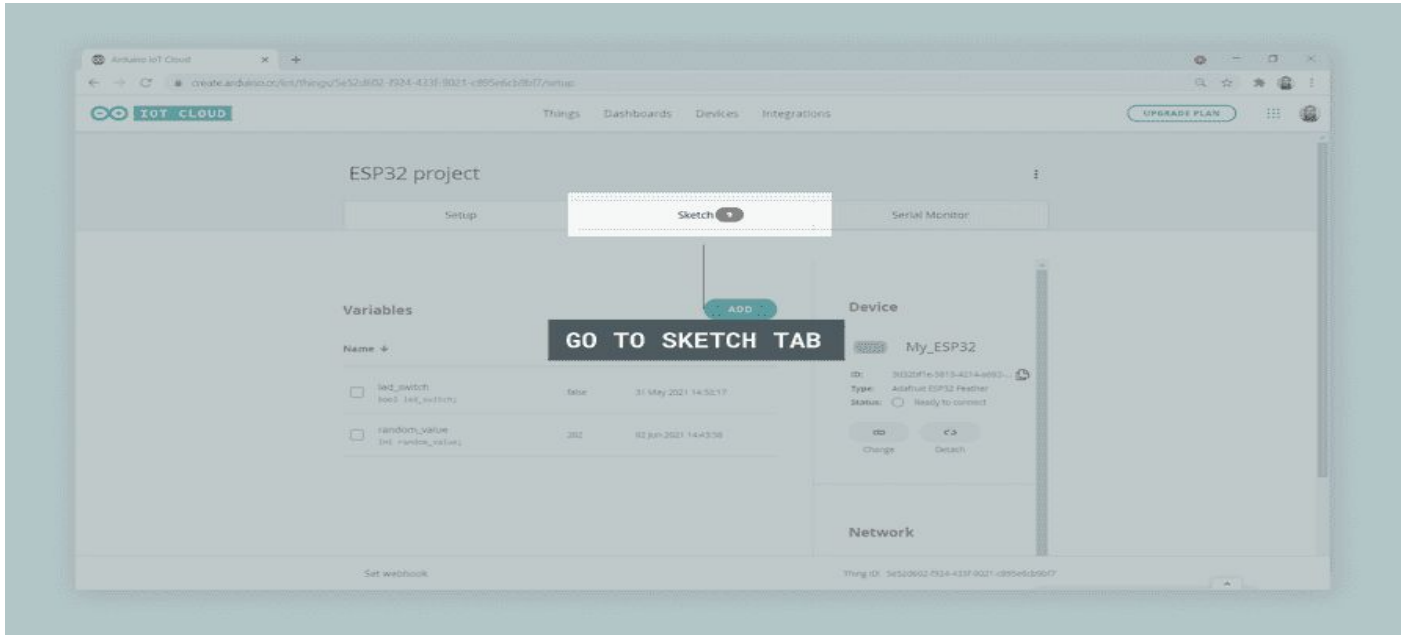
First, click on the button in the **"Network Section"**.



Then, enter the credentials (network name, network password and secret key). Click **"Save"** when finished.

Programming the board

The next step is to program the board. To do so, we need to go to the **"Sketch"** tab



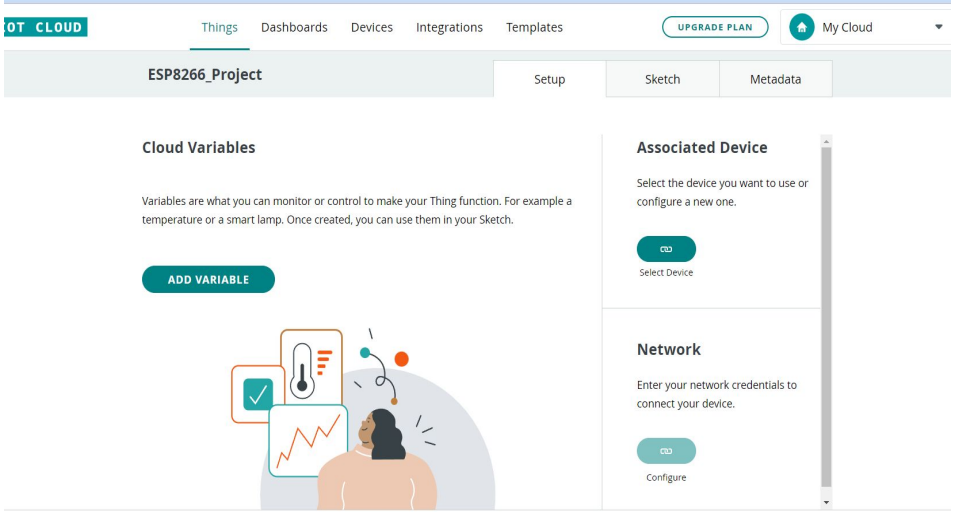
The next step is to program the board. To do so, we need to go to the **"Sketch"** tab.

Basic structure of code is already there, write the necessary code inside sketch.

Step 2: Creating a Thing

The next step is to create a Thing.

This is done by navigating to the **"Things"** tab.

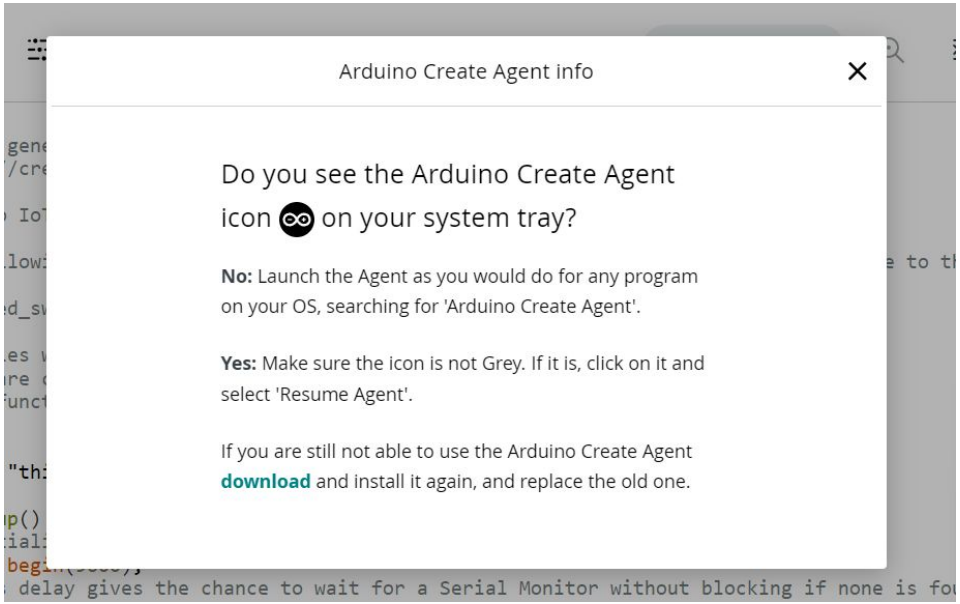


Then, we need to create a new Thing, by cli on the **"Create Thing"** button.

We can then rename our Thing something appropriate to what we are doing, in this ca simply chose **ESP32 Project**.

Uploading the code

If you connect NodeMCU to PC then Arduino IoT cloud we not recognize it. To do so you need to install Arduino Create Agent.



A Pop-up message will occur on screen regarding installation of Arduino agent. Click on learn more and new Pop-up Window named Arduino Create Agent will open. This window has Download link of Arduino Create Agent. Download and install the arduino agent.

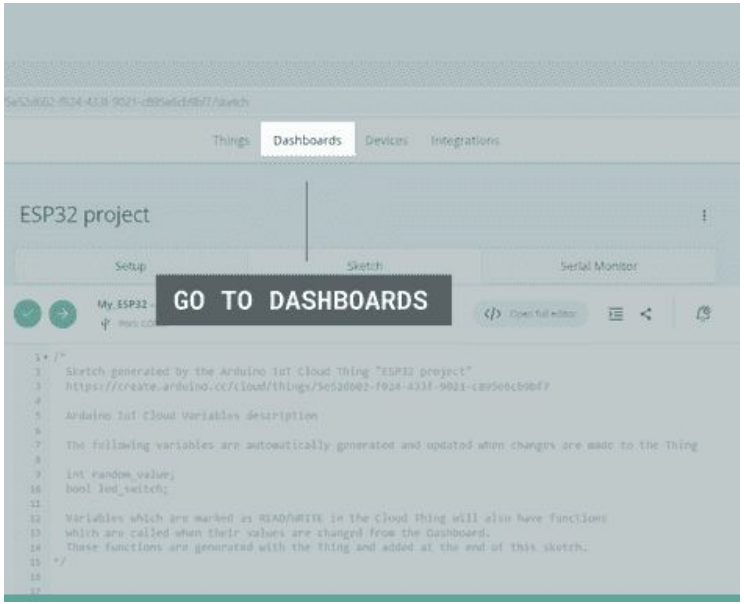
Ones Arduino Create Agent is installed open it, it should not be in gray colour if it is then click on it and select Resume Agent. You can find it in system tray.

Now, connect the NodeMCU with PC or Laptop. Connected device can be seen on top of the sketch. Now click on upload button.

You can see Done uploading message printed in the console at the bottom of the editor at bottom of the sketch.

Creating a dashboard

First, navigate to the **"Dashboards"** tab.



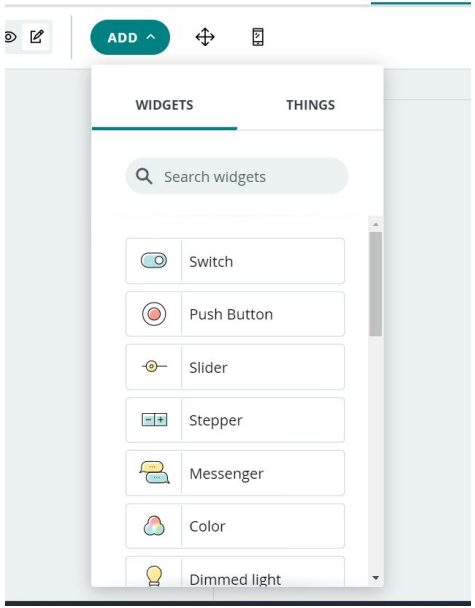
Monitor your Things

Build a Dashboard to easily monitor the status of your Things and control them.

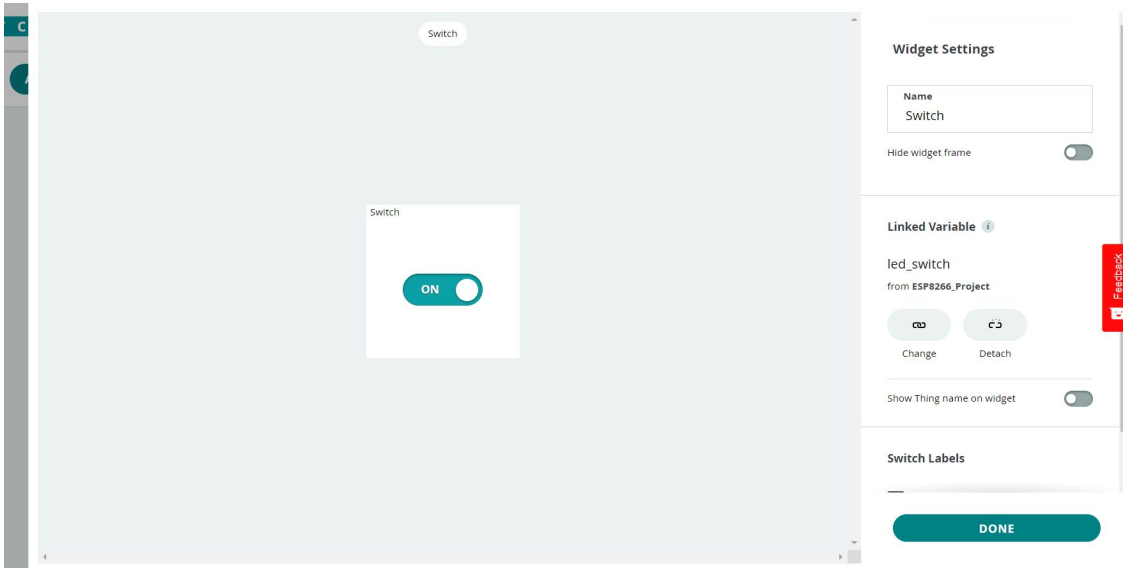
BUILD DASHBOARD

Then, click on the **"Build dashboard"** button.

To edit the dashboard, click on the pencil icon at the top left icon, Name the dashboard I have used ESP8266_DashBoard

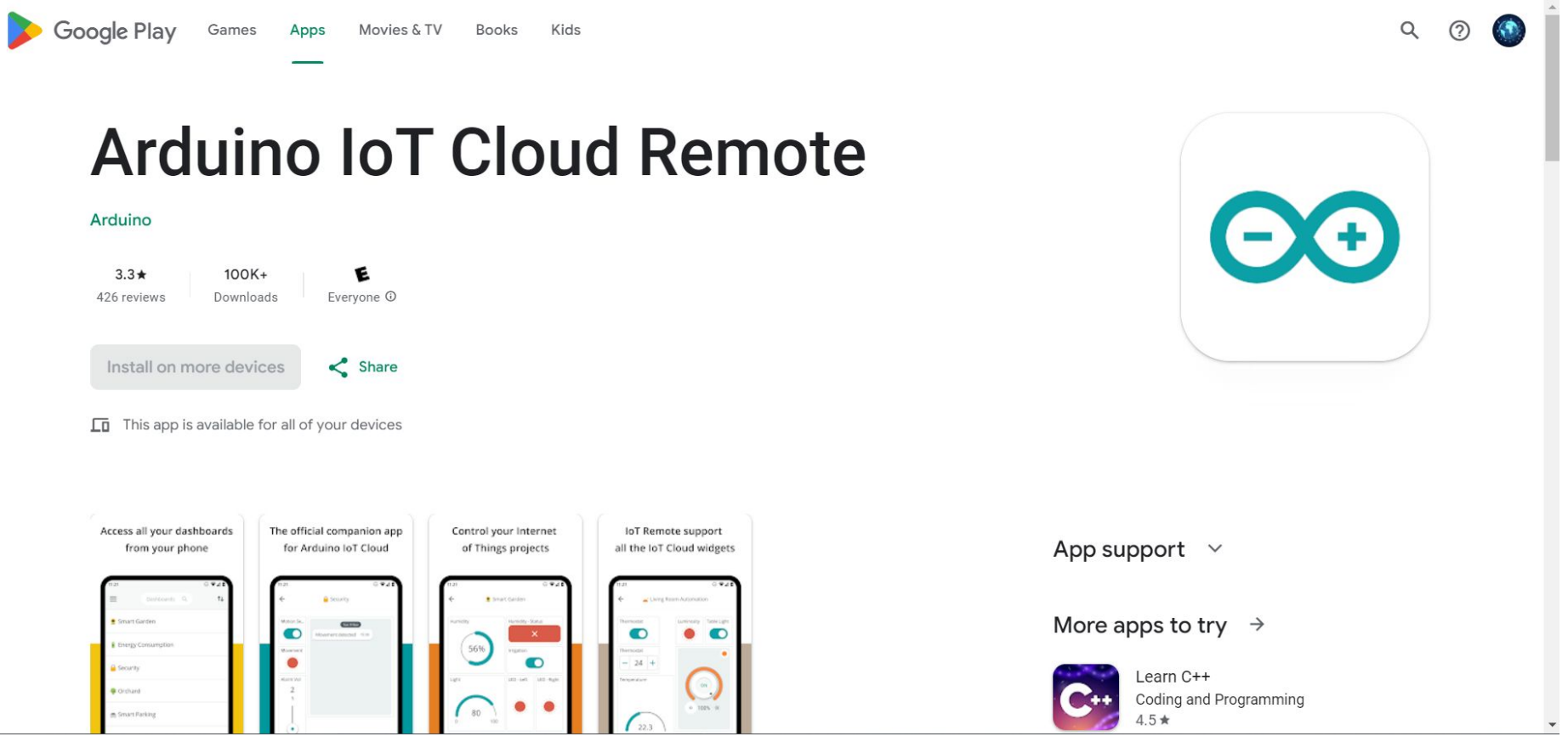


Click on ADD button and go into things tab. Select the things in my case it is ESP8266_Project. Also select the appropriate widgets from widgets tab.

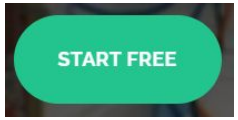


By clicking on widget setting of widget tab will open, here you can gave name to switch. Important here is you need to link variable. So we created led_switch variable is linked with switch using Link variable button. After doing all this click on done. **Now your project is ready to RUN……**

You can use the dashboard on Arduino IoT cloud Remote App on your phone.

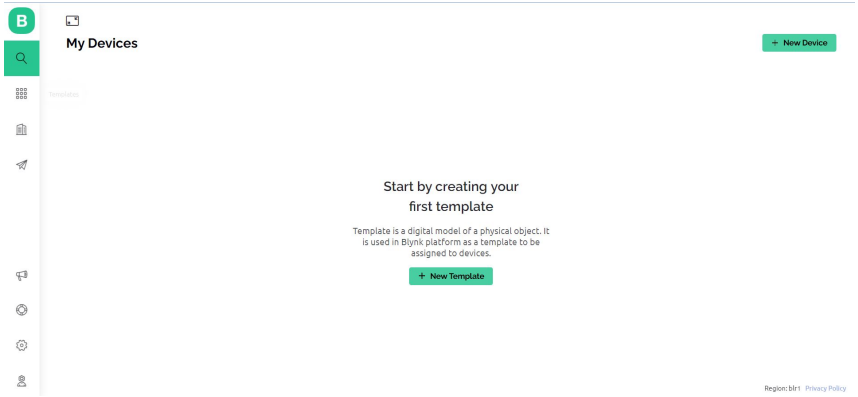


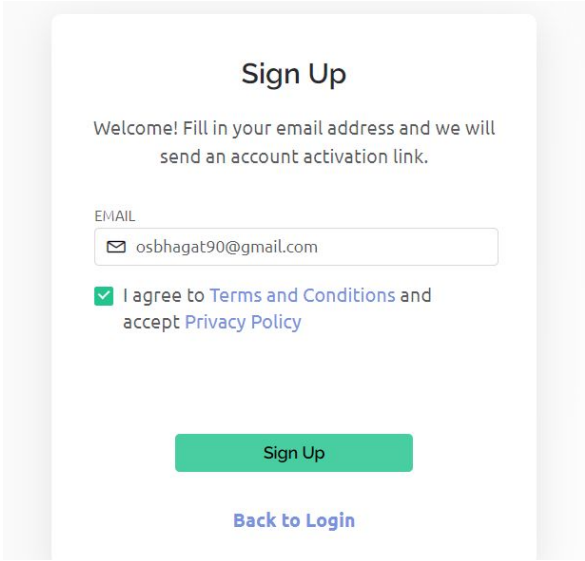
Blynk IoT Cloud

1. Search Blynk Iot on Google, Click on first result http://blynk.io
2. Click on start for free 

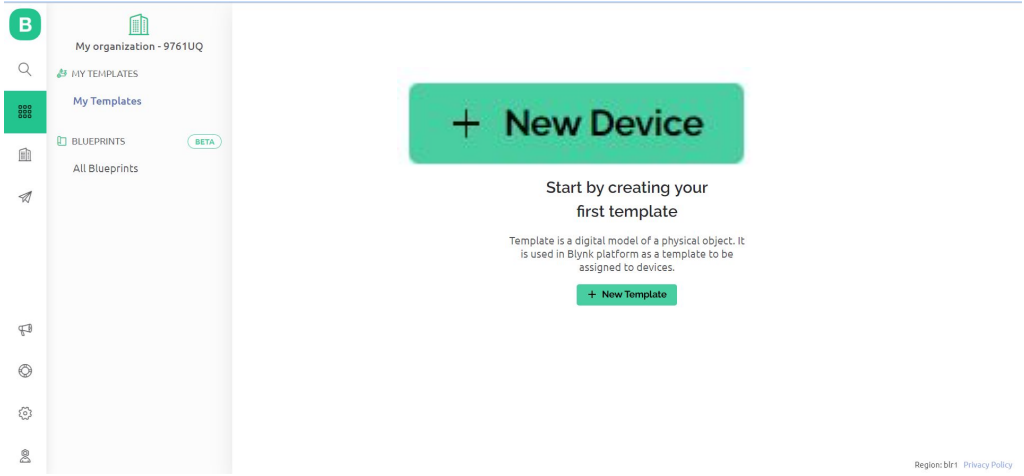
or else login by clicking login 

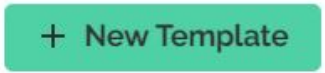
button on upper right corner.
3. By filling appropriate details you will redirect to home page.
4. Close all the pop-up and quick-start guides.





Click on Template option  which is on left side of the window. New option tray will open.



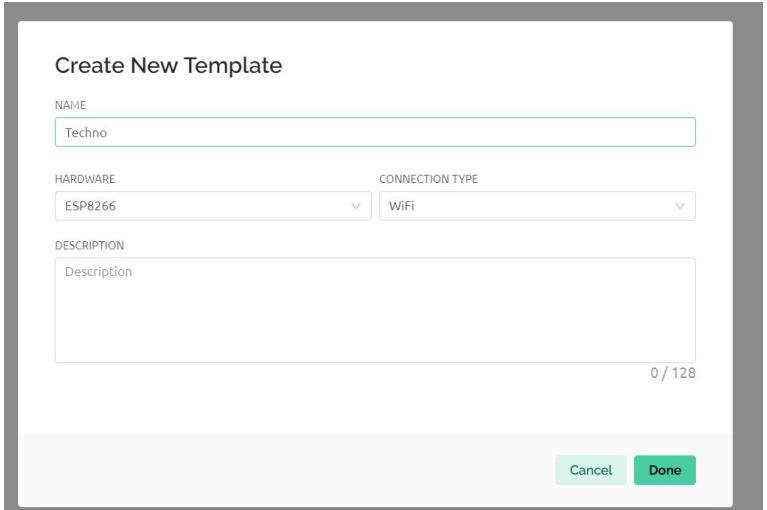
Click on new Template 

Select Hardware as ESP8266

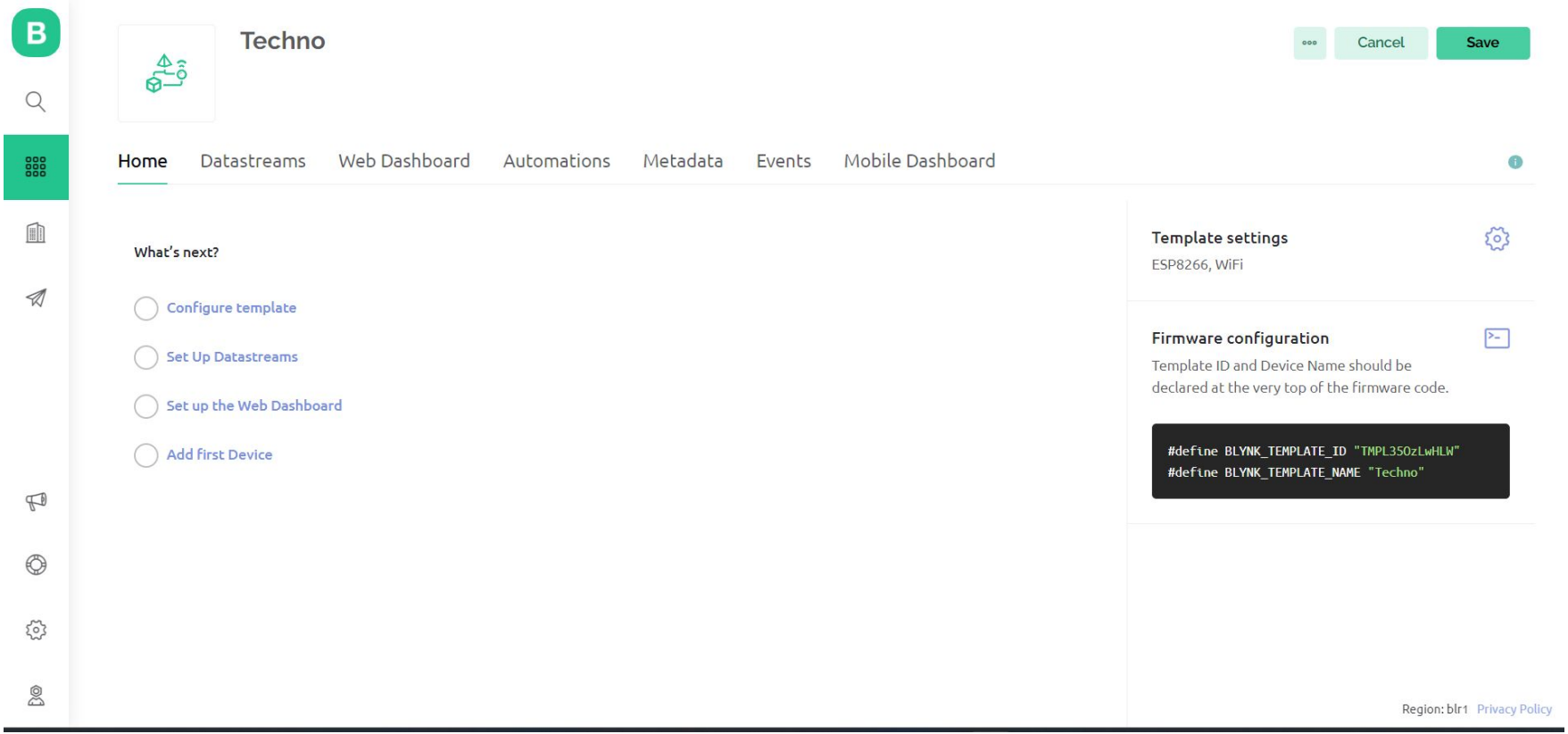
Connection type as WiFi and

Name the template

Click on Done.

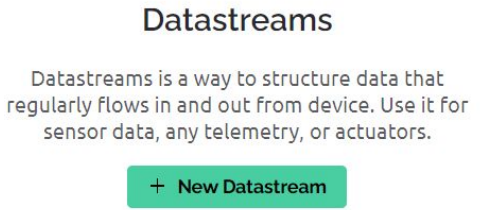


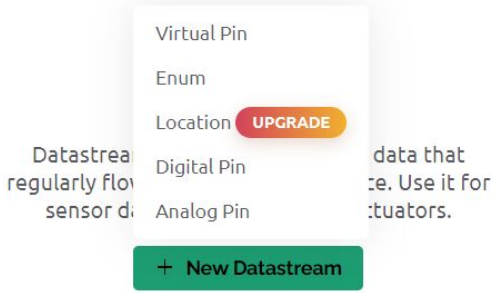
New Window naming your template will open, Lots of tabs will be on this window, you will get Template ID and Template name on bottom left.



Click on Datastreams tab, New tab will open.

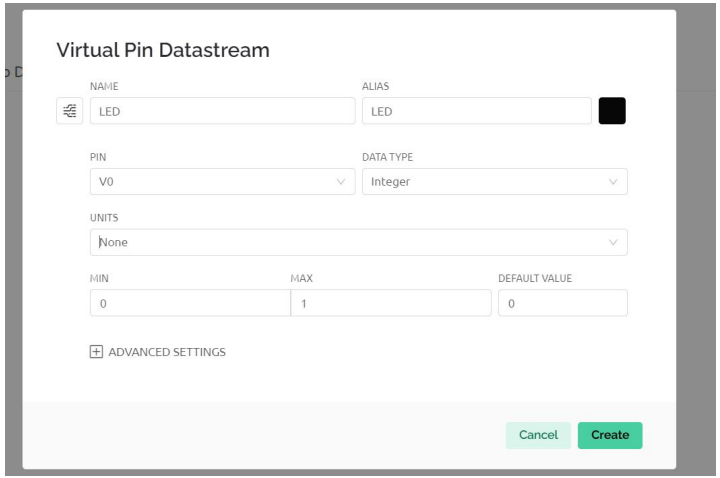
Click on New Datastreams



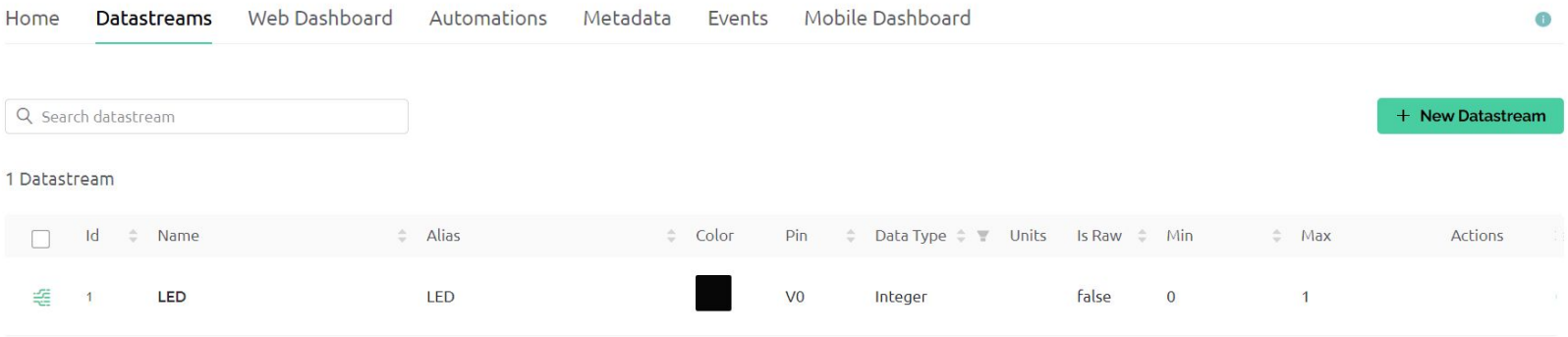


Lots of option can be observed on screen click on Virtual Pin option.

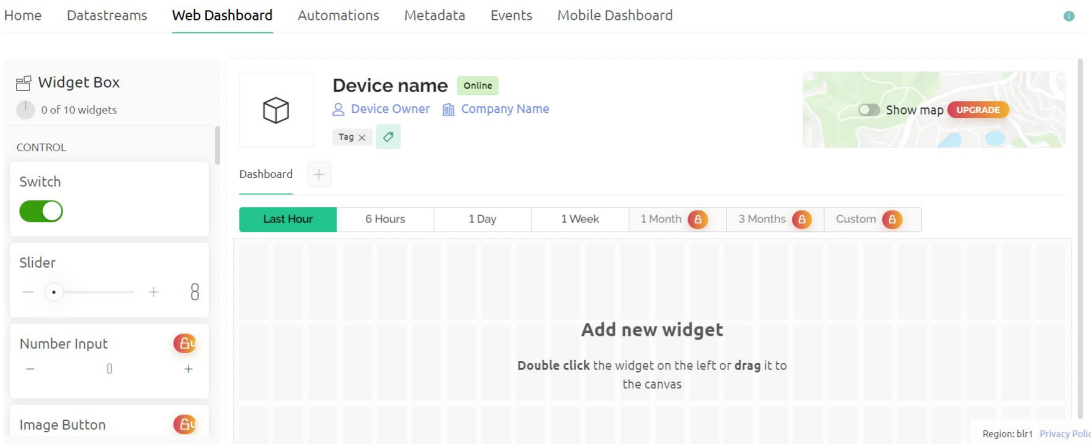
Virtual Pin Datastream setting will open, Name the pin and remaining things kept as it is.



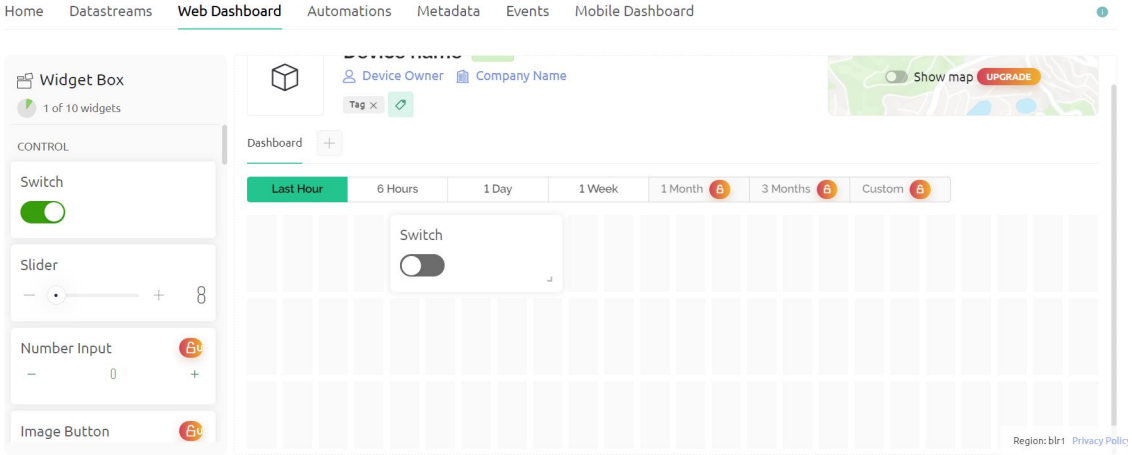
After setting all things click on create button.
New datastream named LED will be created.



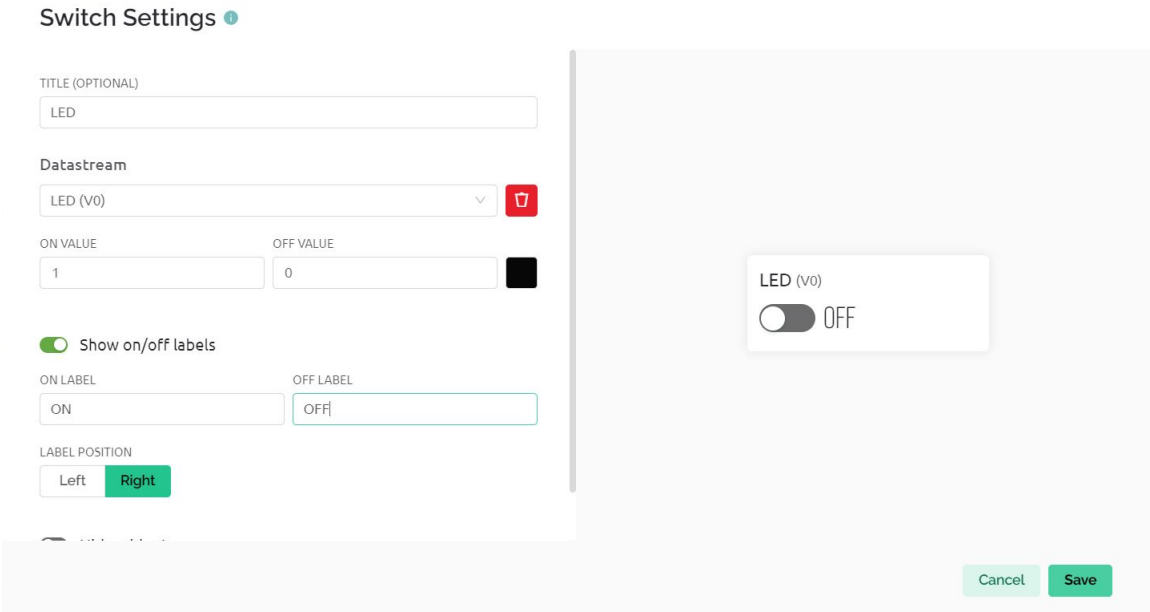
Click on Web Dashboard, new tab will be open.



Click on switch Widget and move to dashboard.



Click on widget setting and setting window will open

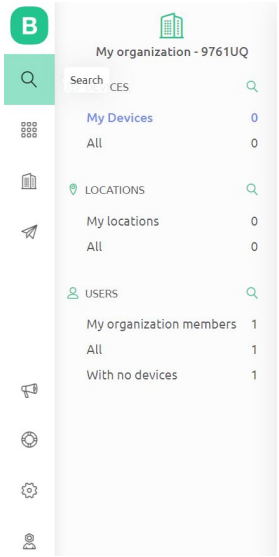


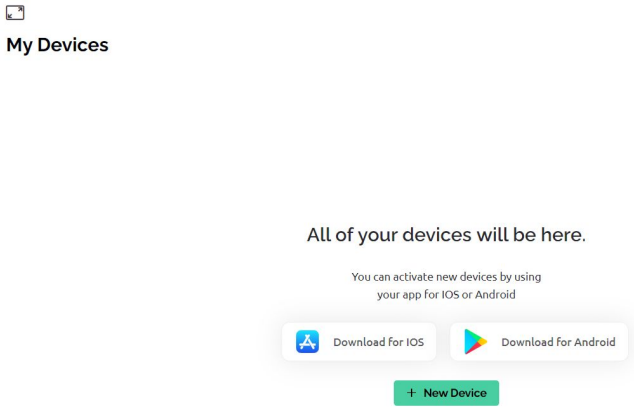
Click on datastream and choose LED(V0) and Turn on Show on/off labels option. Put name for labels.

Click on save button.

Click on Save button on upper right corner.

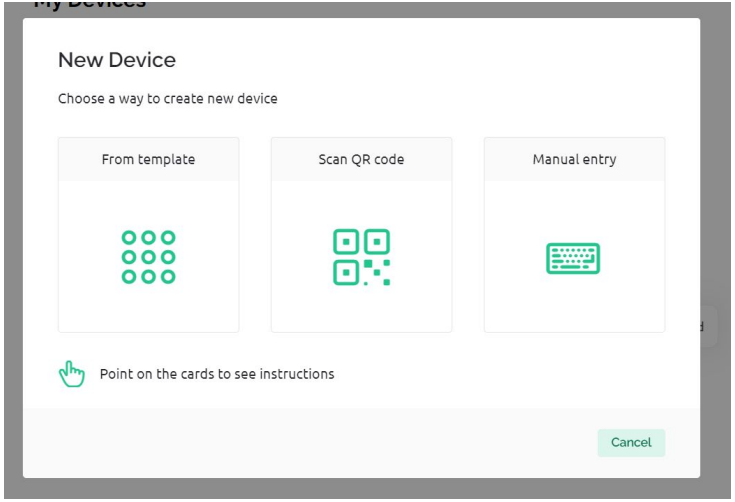
By clicking on search option,



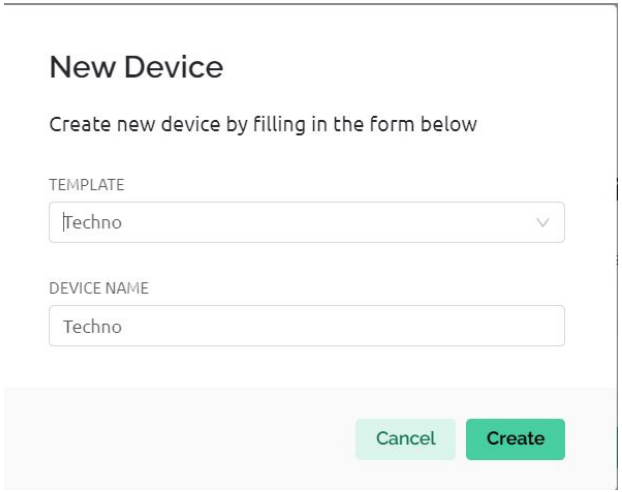


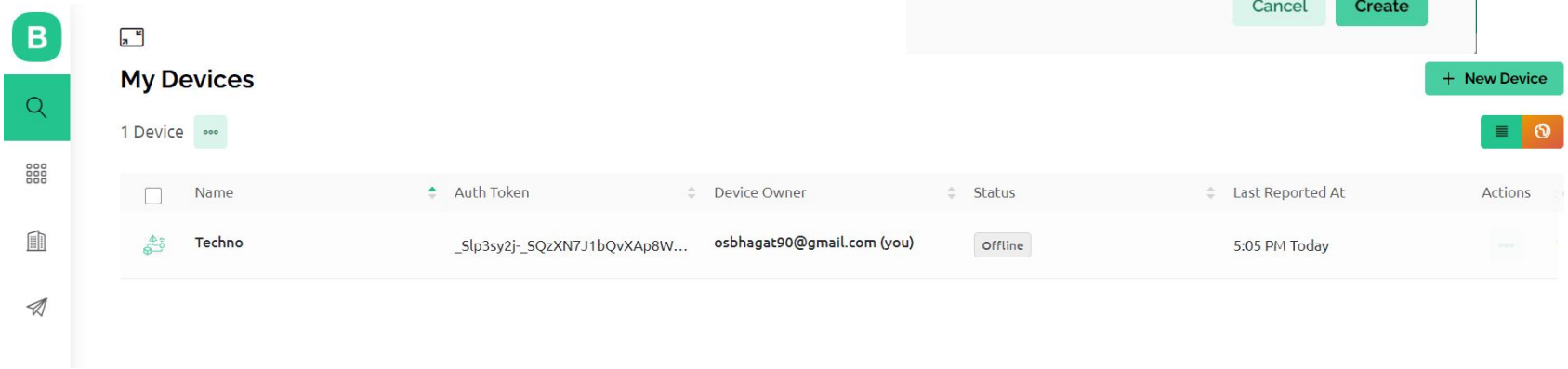
Click on new Device

New window will open click on from template.



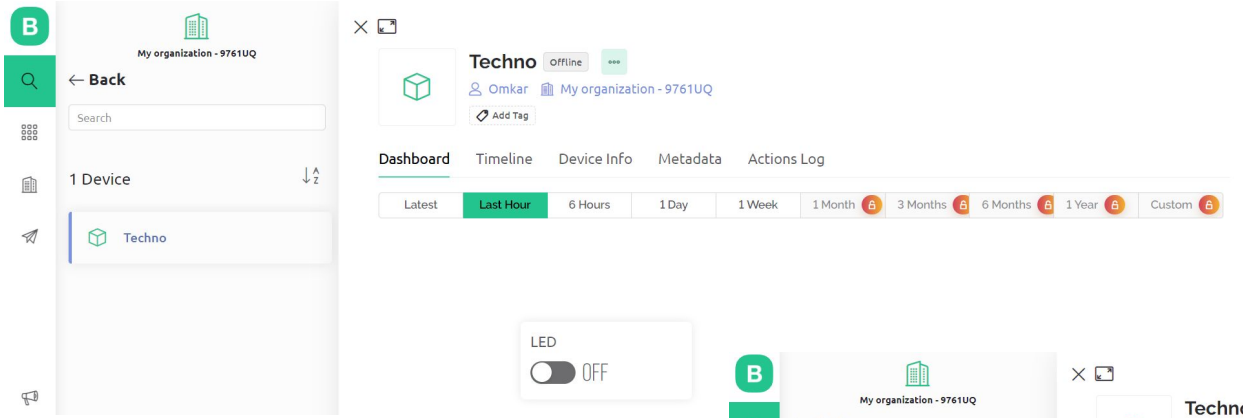
Previously created template will be shown in template option, click on that and click on create.



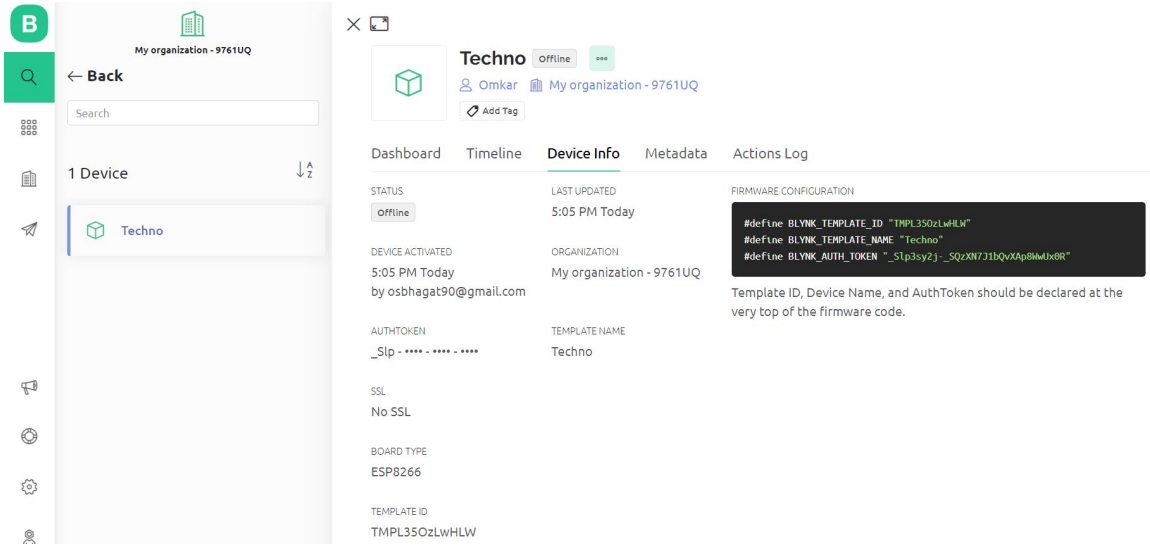


Created device will be appear in my Device list.

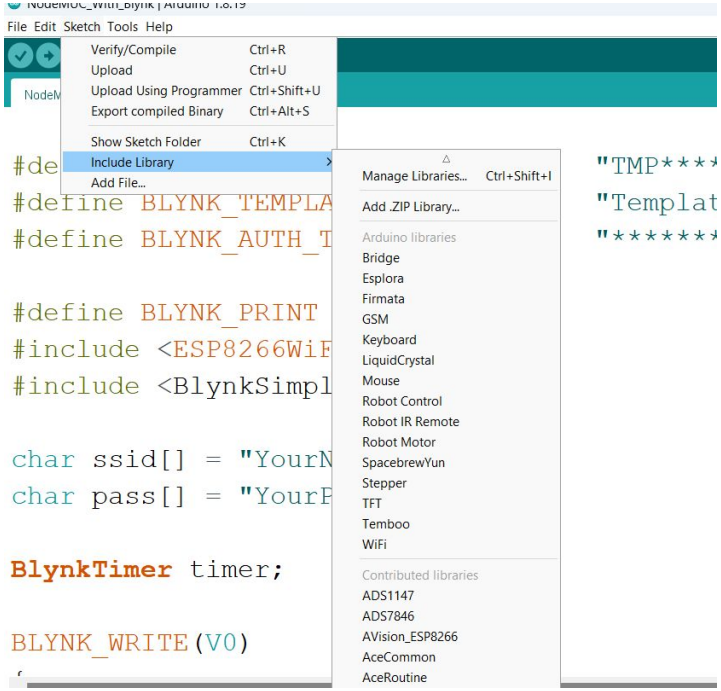
Click on created template or device click on it, window having multiple tabs will be shown.



Click on device info you will get Template id, template name, authentication token
Save that info.



Open Arduino IDE
Click on Sketch tab => Include Library => Manage Library



Search for Blynk and Install first library.

Go to this link :
https://drive.google.com/drive/folders/1bgqPTNDGQ-gIxOg0Ast97tNompkXckJb?usp=sharing

Download the NodeMCU_with_Blynk folder

Open NodeMCU_with_Blynk.ino file in arduino.

Write down the information stored from device info this code.

```
#define BLYNK_TEMPLATE_ID           "TMP*********"
#define BLYNK_TEMPLATE_NAME         "Template Name"
#define BLYNK_AUTH_TOKEN            "***************"
```

Enter WiFi SSID and Password in respective space

```
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";
```

BLYNK_WRITE() function is used to read value form button created on dashboard. We write code inside this function.

```
BLYNK_WRITE(V0)
{
  int value = param.asInt();
  if(value==1){
    digitalWrite(D4,LOW);
  }else{
    digitalWrite(D4,HIGH);
  }
  Blynk.virtualWrite(V1, value);
}
```

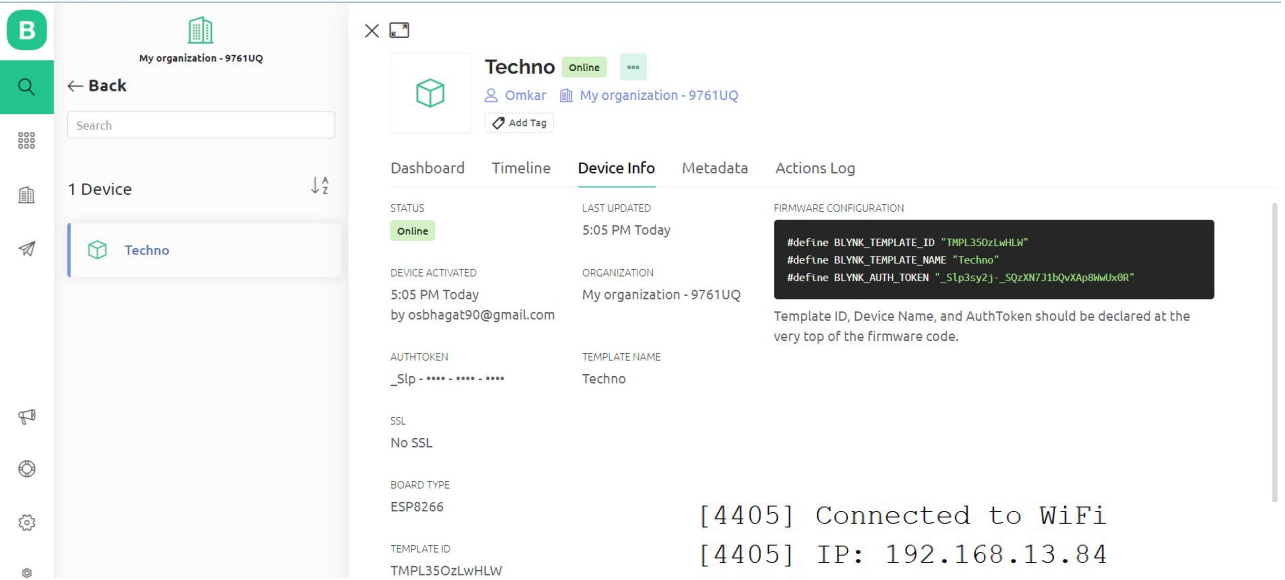
**Here D4 led is Active LOW.

Value of Switch V0 is stored in value variable. If value become 1 then turn on LED(D4) or else turn off the LED(D4).

Also mention pinMode(D4,OUTPUT); in void setup() function.

Connect NodeMCU with Laptop and Upload the code.

Once NodeMCU is connected to WiFi network the Serial monitor and Blynk Dashboard will look like this

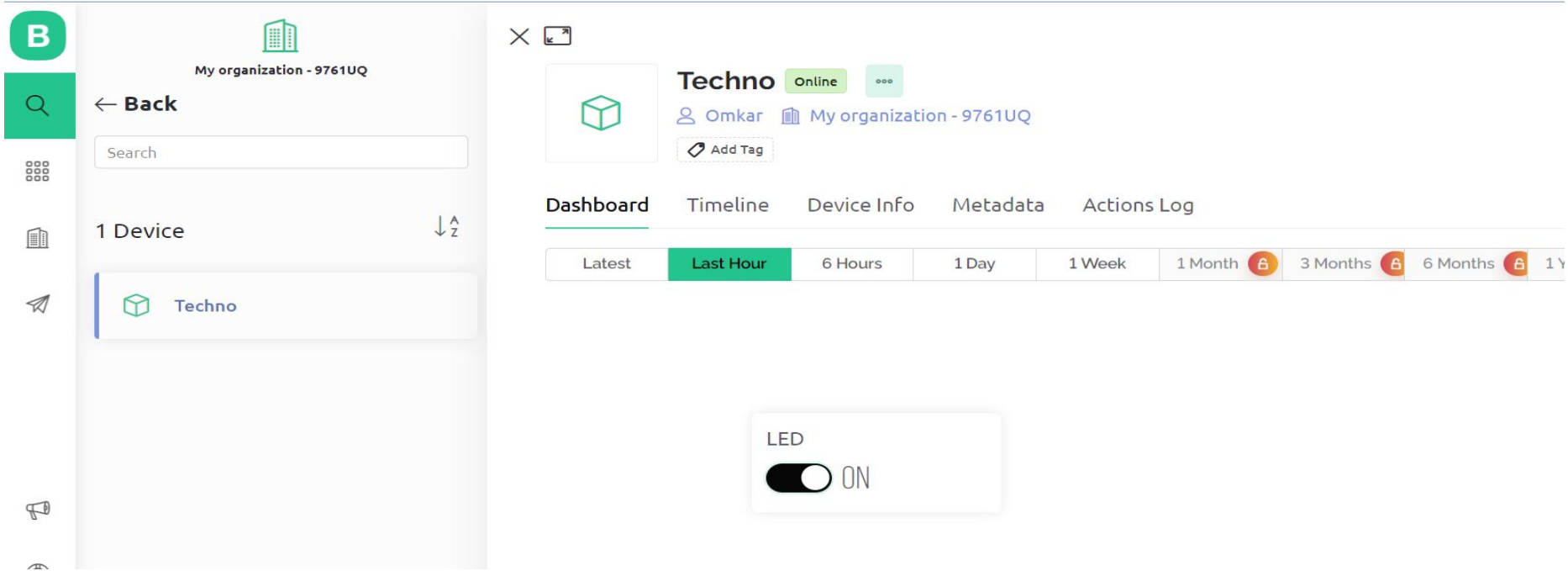


Device status is online on dashboard

IP address of device will be printed on Serial monitor.

```
[4405] Connected to WiFi
[4405] IP: 192.168.13.84
[4405]
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v1.3.0 on ESP8266

#StandWithUkraine    https://bit.ly/swua
```

Now you can control on board of NodeMCU using switch created on Blynk Dashboard

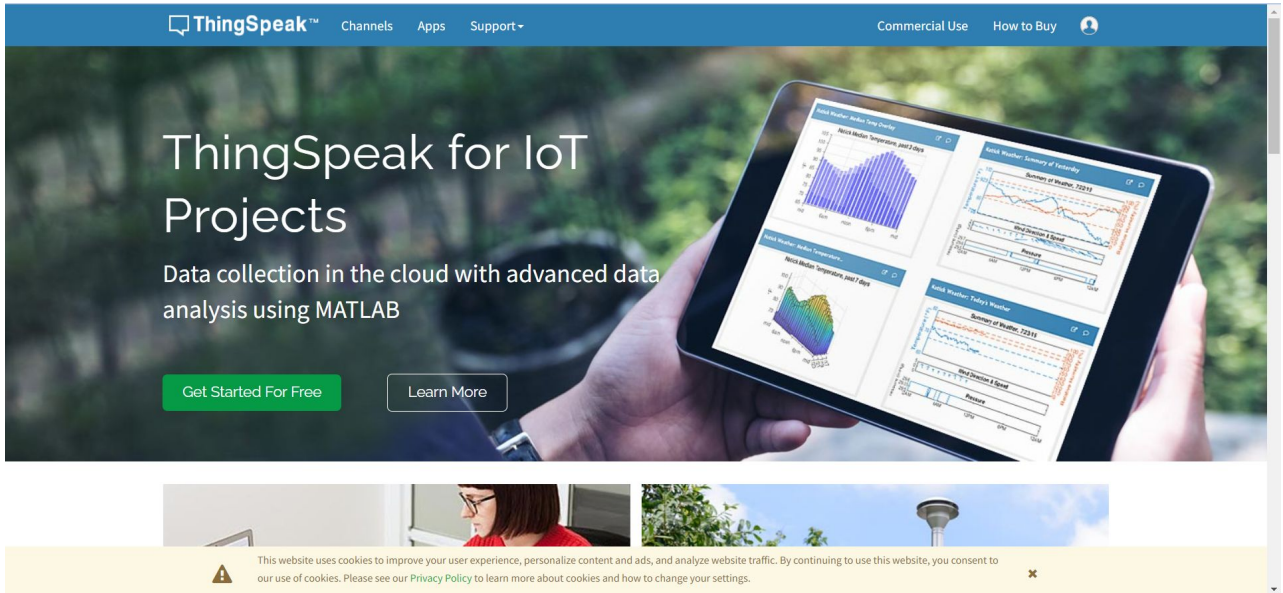


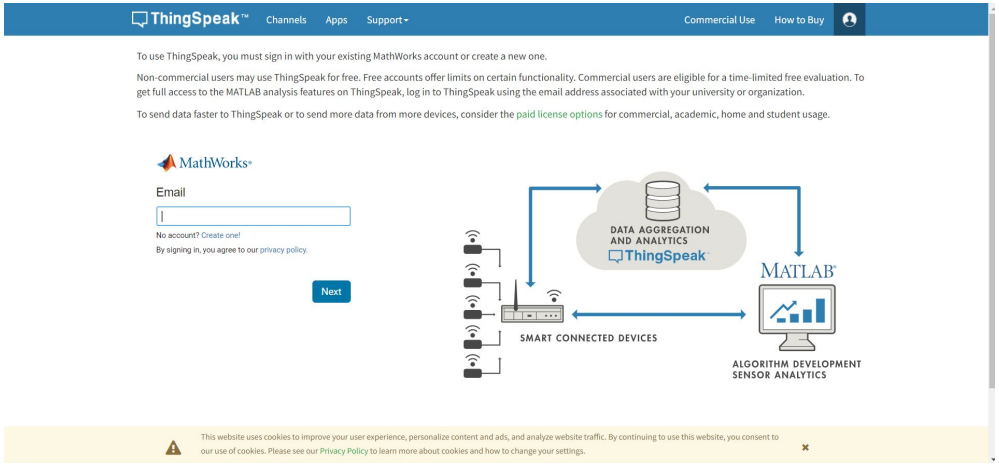
Similar dashboard can be made on Mobile App and LED can be controlled using mobile App.

ThingSpeaK

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

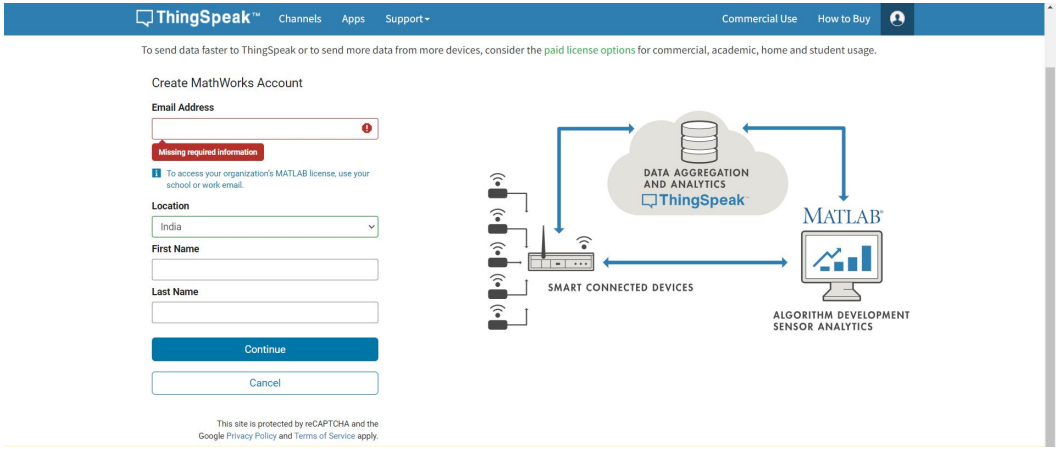
1. Visit https://thingspeak.com/
2. Click on Get started for Free





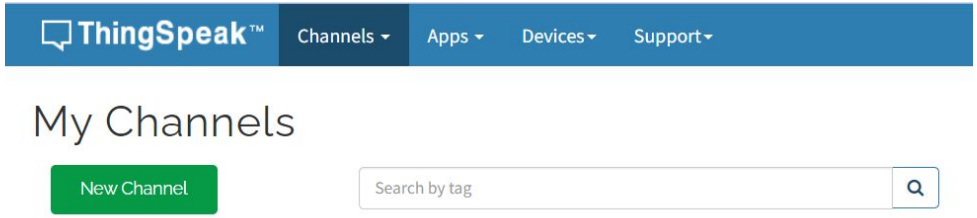
If you already have an account then enter your email

If not then click on create account.
Enter right details and create account



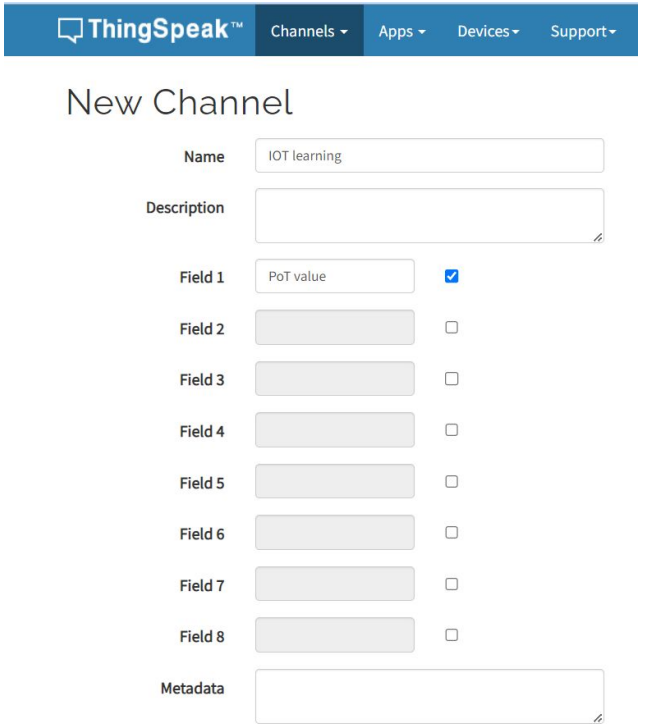
By creating account and verifying Email id you will redirect to the…

By creating account and verifying Email id you will redirect to the channel page. Click on New Channels



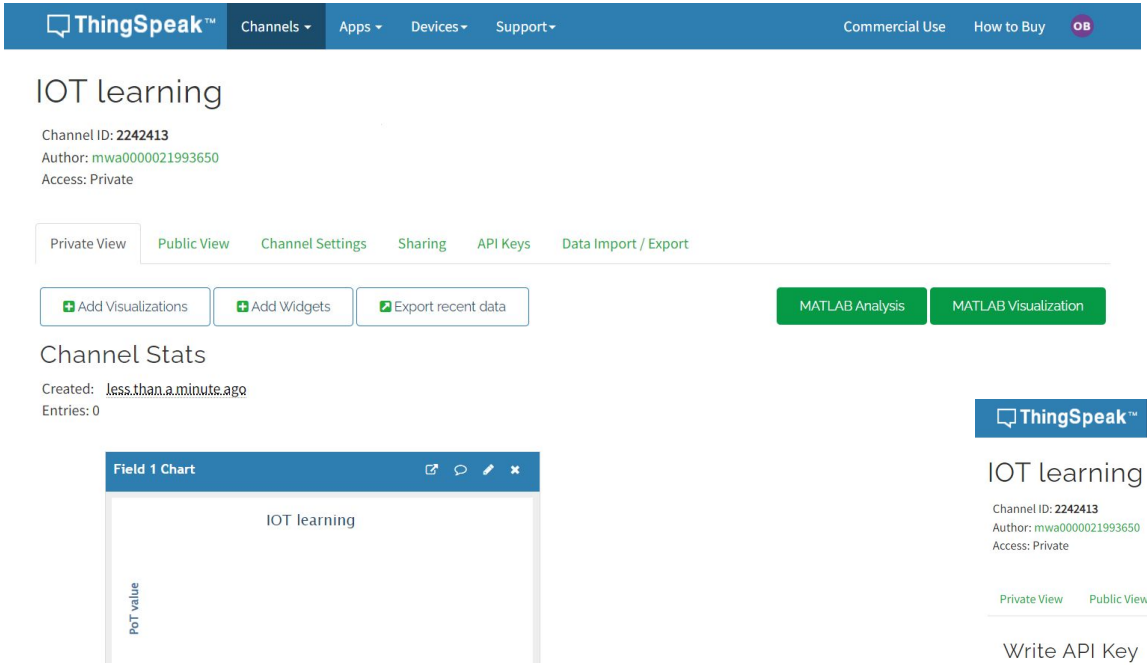
New page having space for channel details will open. Here you can name your channel.
There is option of Fields.
Field is nothing but data you want to upload using ESP8266.

Depending upon your data you can choose fields by clicking on it and naming them.

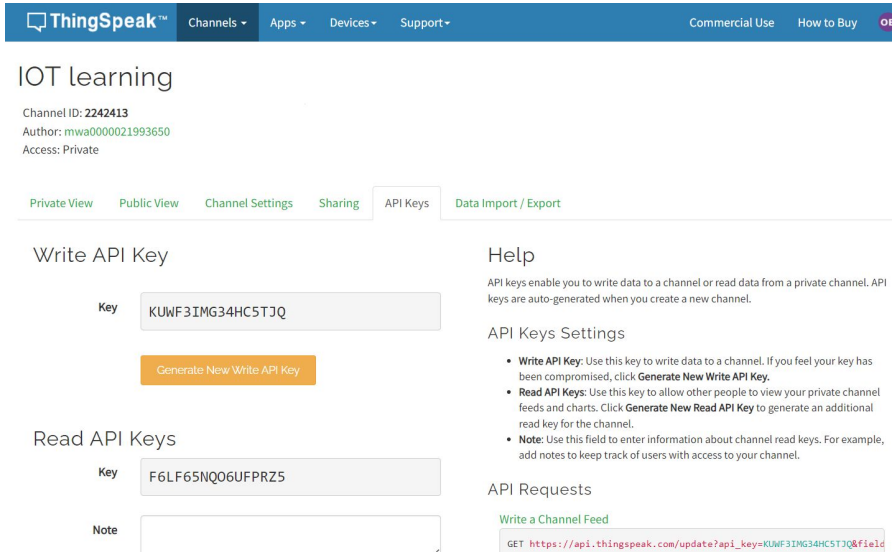


At bottom of this page there is save button, click on that button to save setting.

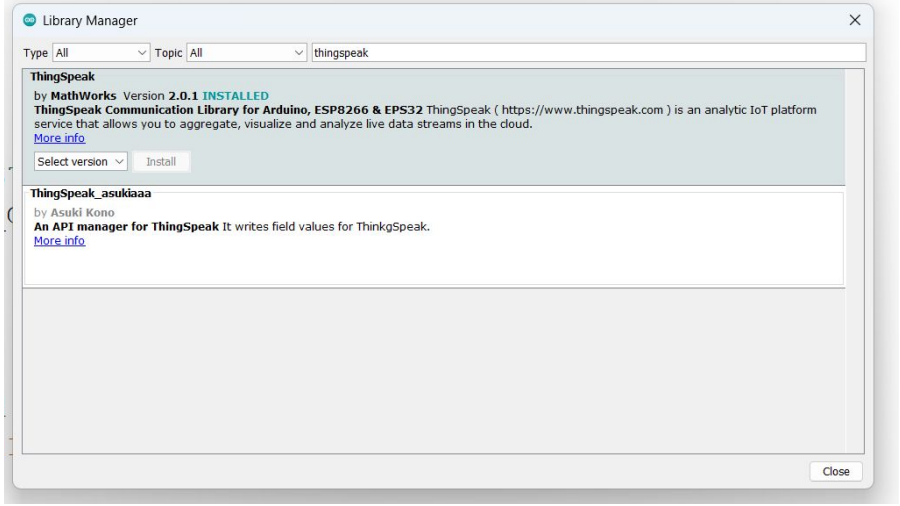
Now you can see page showing details about created channel. This page has dashboard show showing uploaded values.

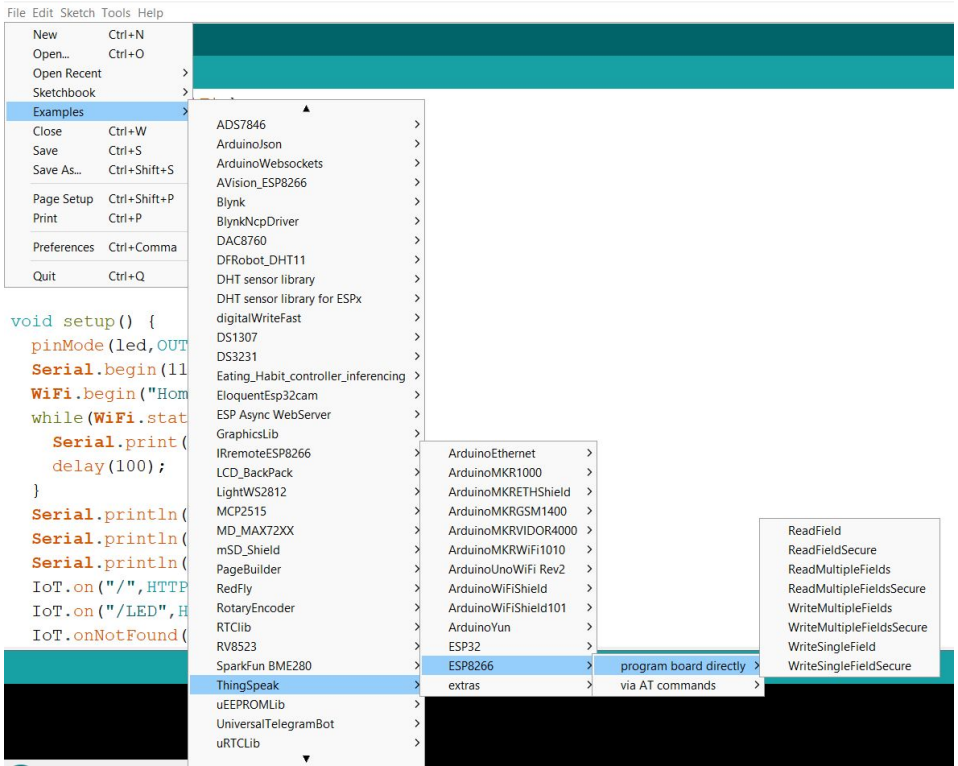


By clicking on API Keys tab you will get all the Write and Read API Keys.



Install ThingSpeak library from library manager in arduino.





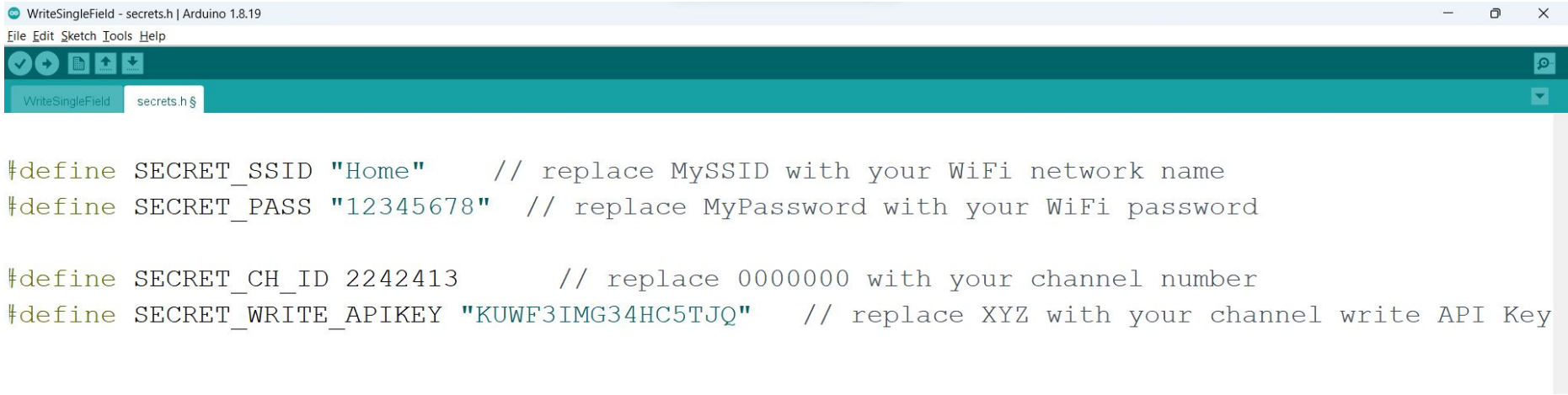
Now open Example.

Files => Examples => ThingSpeak => ESP8266 => Program Board directly => WriteSingleFiedl

This program has two tabs WriteSingleField and Secrets.h

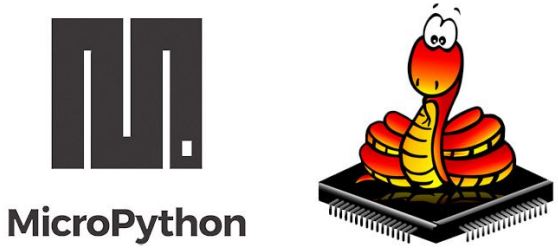
Now open the Secrets.h file and put SSID and Password of WiFi to which you want to connect NodeMCU.

Channel Id and API Key is taken form API key tab of your newly created Channel.



WriteSingleField is uploading number 0 to 99 to respective field.
By changing the contain of number variable you can upload your desired data.

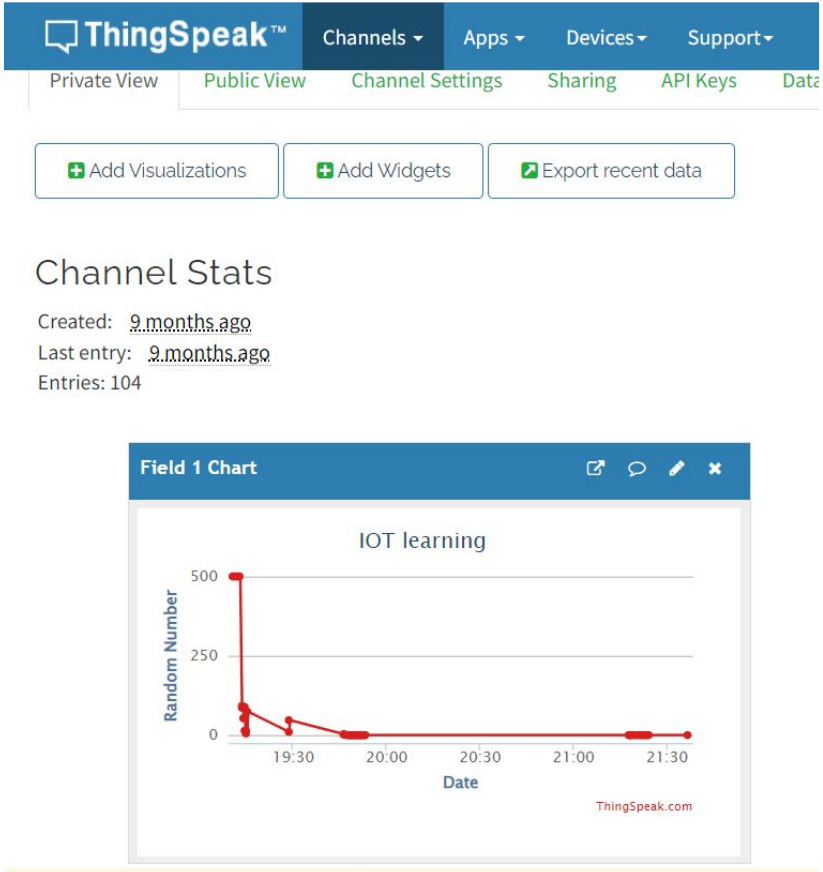
MicroPython with ESP8266



MicroPython is a software implementation of the Python programming language optimized for microcontrollers and embedded systems. It provides a Python interpreter that runs on these resource-constrained devices, allowing developers to program them using Python syntax and libraries.

The first thing you need to do is download the most recent MicroPython firmware .bin file to load onto your ESP8266 device. You can download it from the MicroPython downloads page.

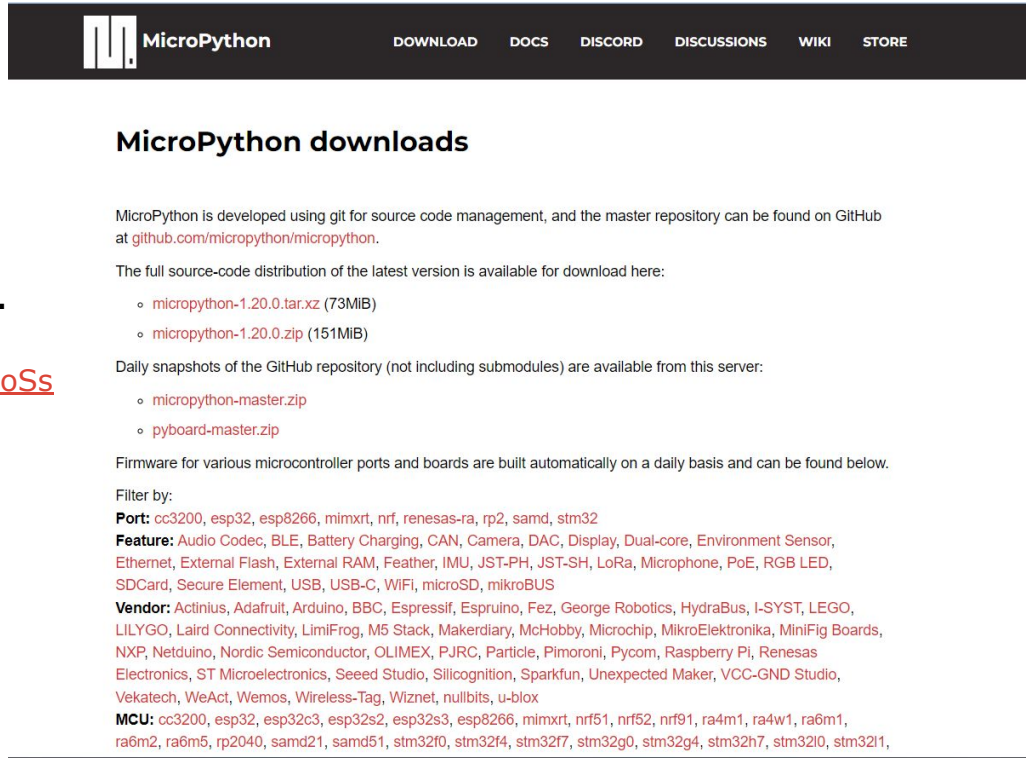
The uploaded value can be seen on dashboard. Inside Public View tab.



The first thing you need to do is download the most recent MicroPython firmware .bin file to load onto your ESP8266 device. You can download it from the MicroPython downloads page.

There is lots of options available there but you have to choose write option depending upon SOC, Board, RAM and Flash.

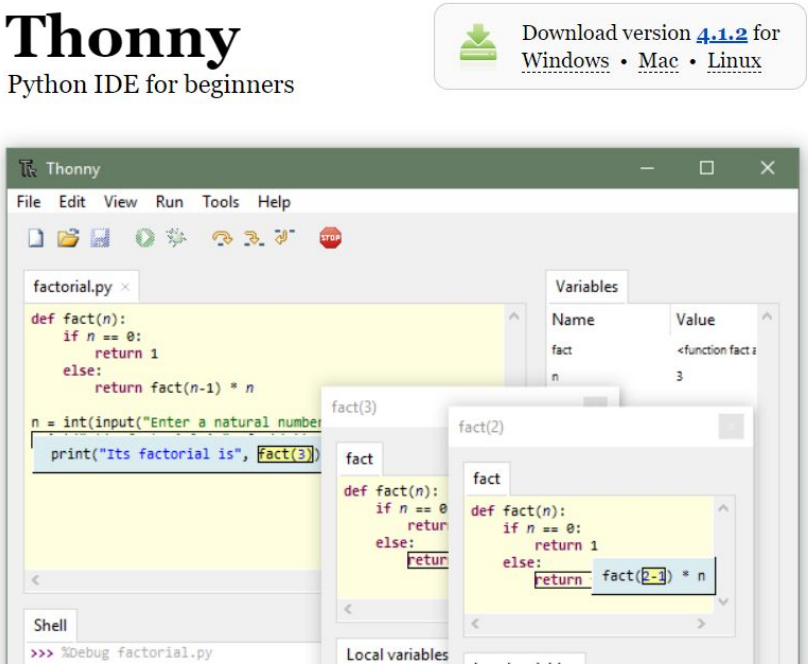
But to make your life more easy we have selected right firmware for you. It is here https://drive.google.com/file/d/1KObWKTukzCoSsEeyqXEoa2hdzblC0-BA/view?usp=drive_link



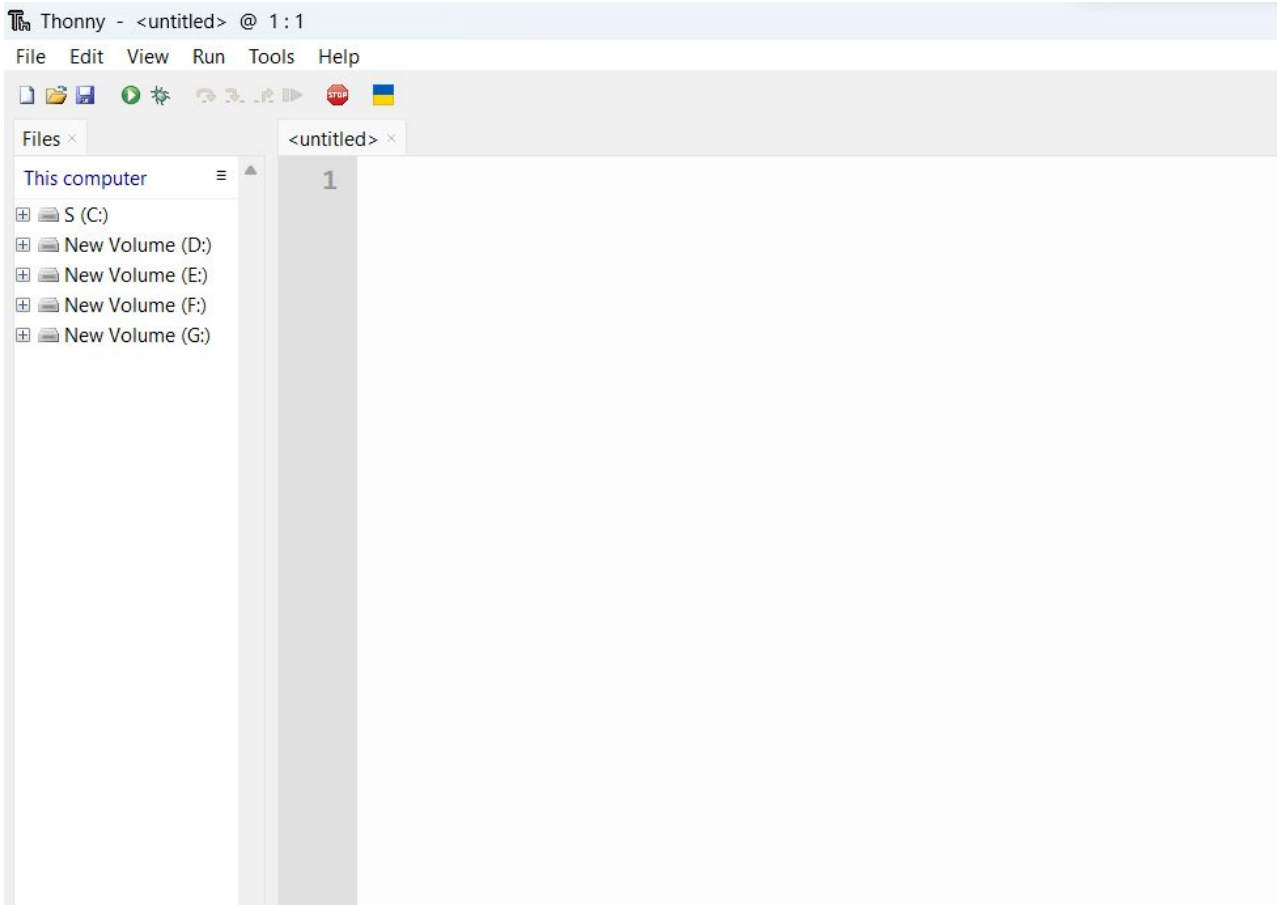
Now you also need IDE to run the micropython code, we are using Thonny IDE.

Google search Thonny
Open first link
On home page itself you will get Download option.
By clicking appropriate OS the respective IDE will be downloaded.

After Downloading install the Thonny IDE.

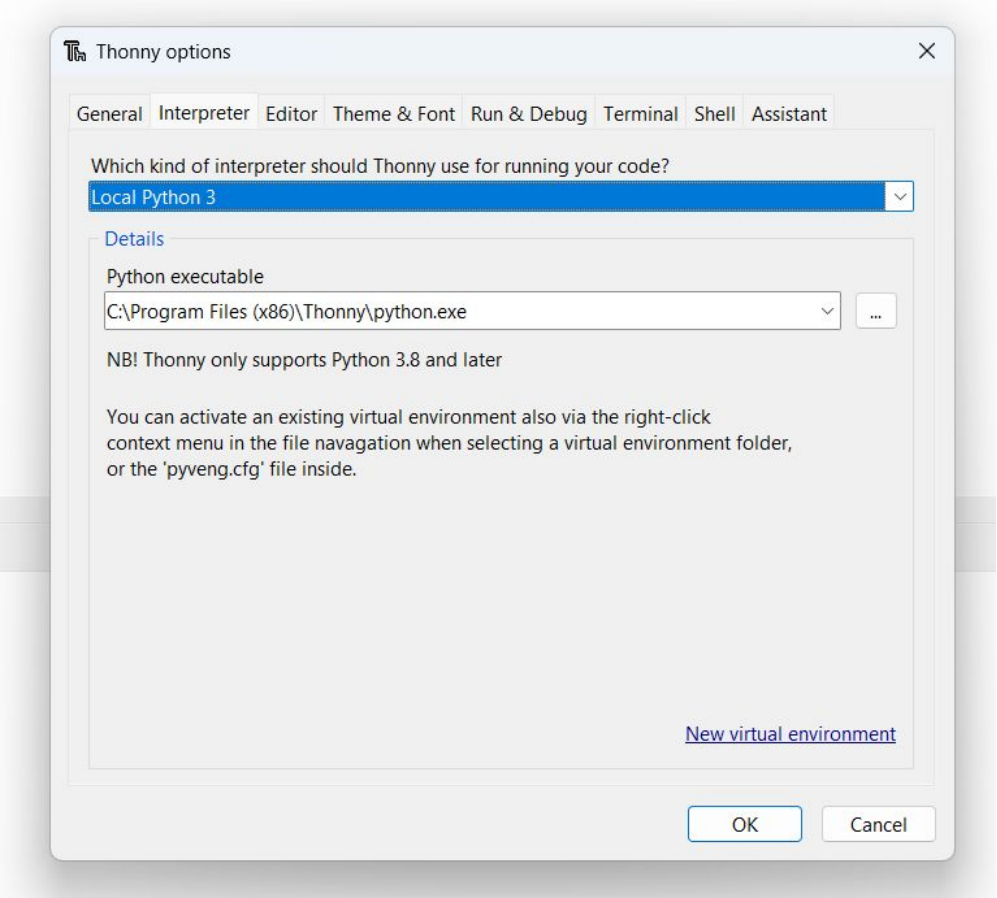


Opening Thonny IDE you will see simple interface.



To make thonny IDE compatible for MicroPython on ESP8266 we need to make some settings.

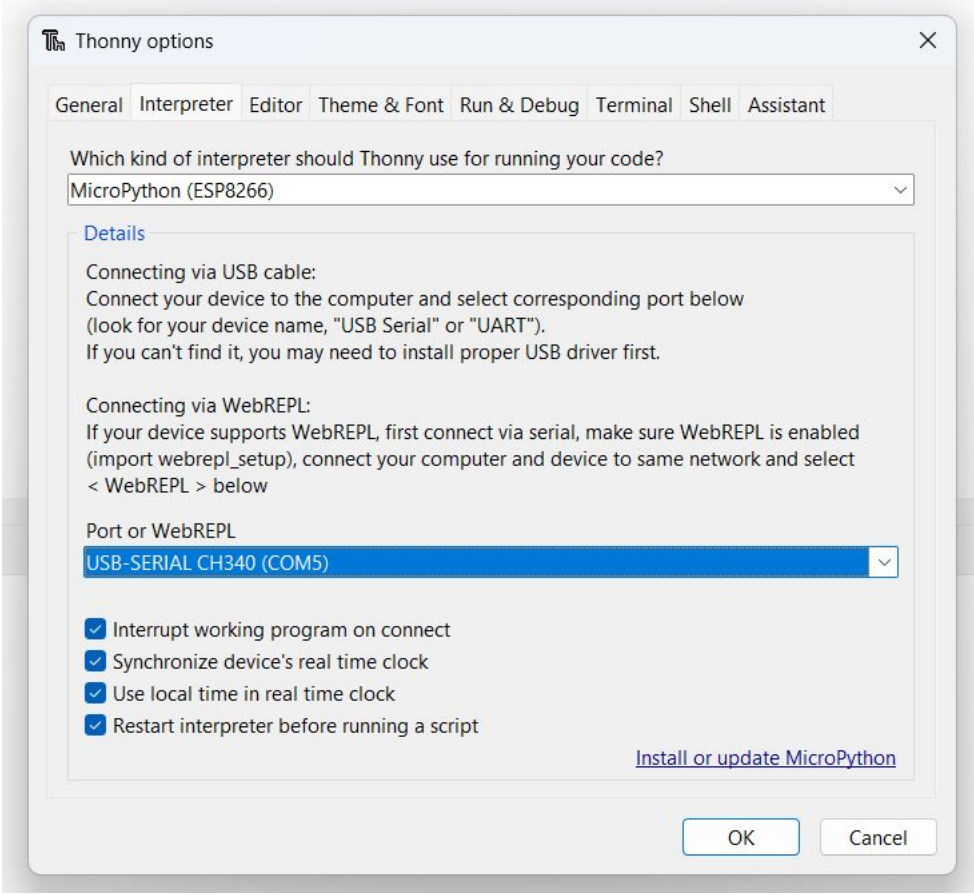
Now click on RUN tab and select configure Interpreter option. New window will be open.



There is drop down menu for selecting python interpreter. Select MicroPython (ESP8266) option for Dropdown menu.

There is another drop down menu for selecting Port.
Select the right port.
(Make sure that NodeMCU is connected to laptop)

Click on Install or update MicroPython option in bottom right corner on new window.

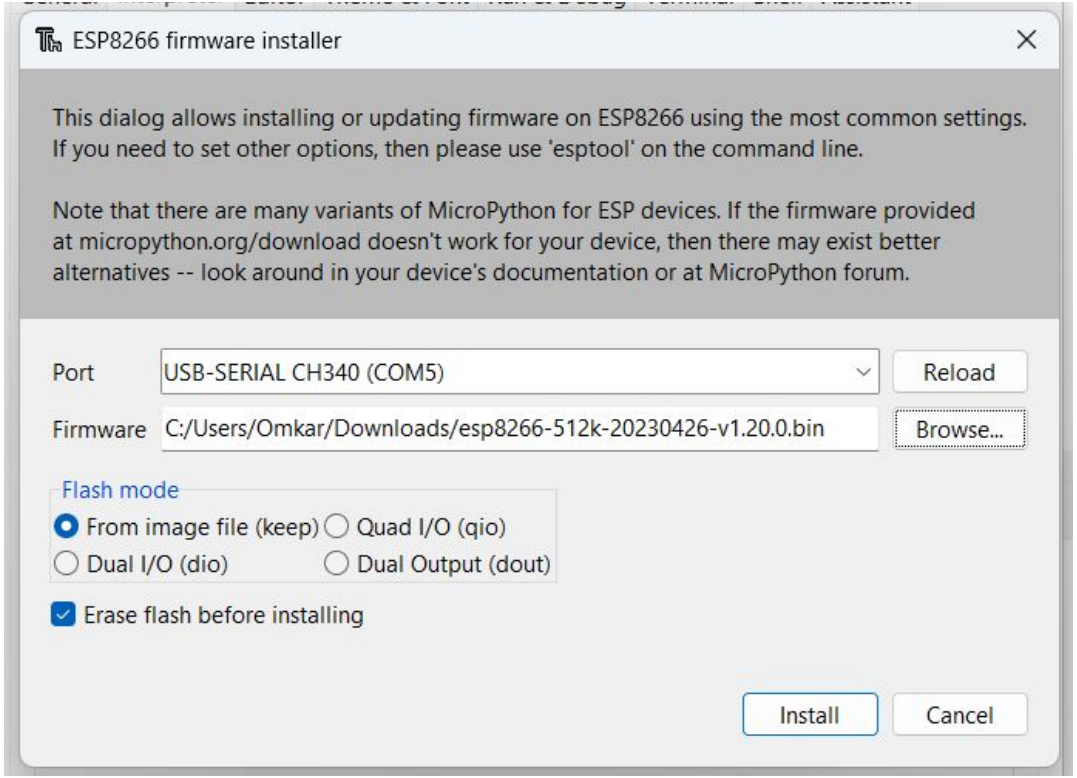


New Window will be open named ESP8266 firmware installer.
Select right port from drop down menu for port and past the path of firmware which downloaded from link.

Click on From image file (keep) option.

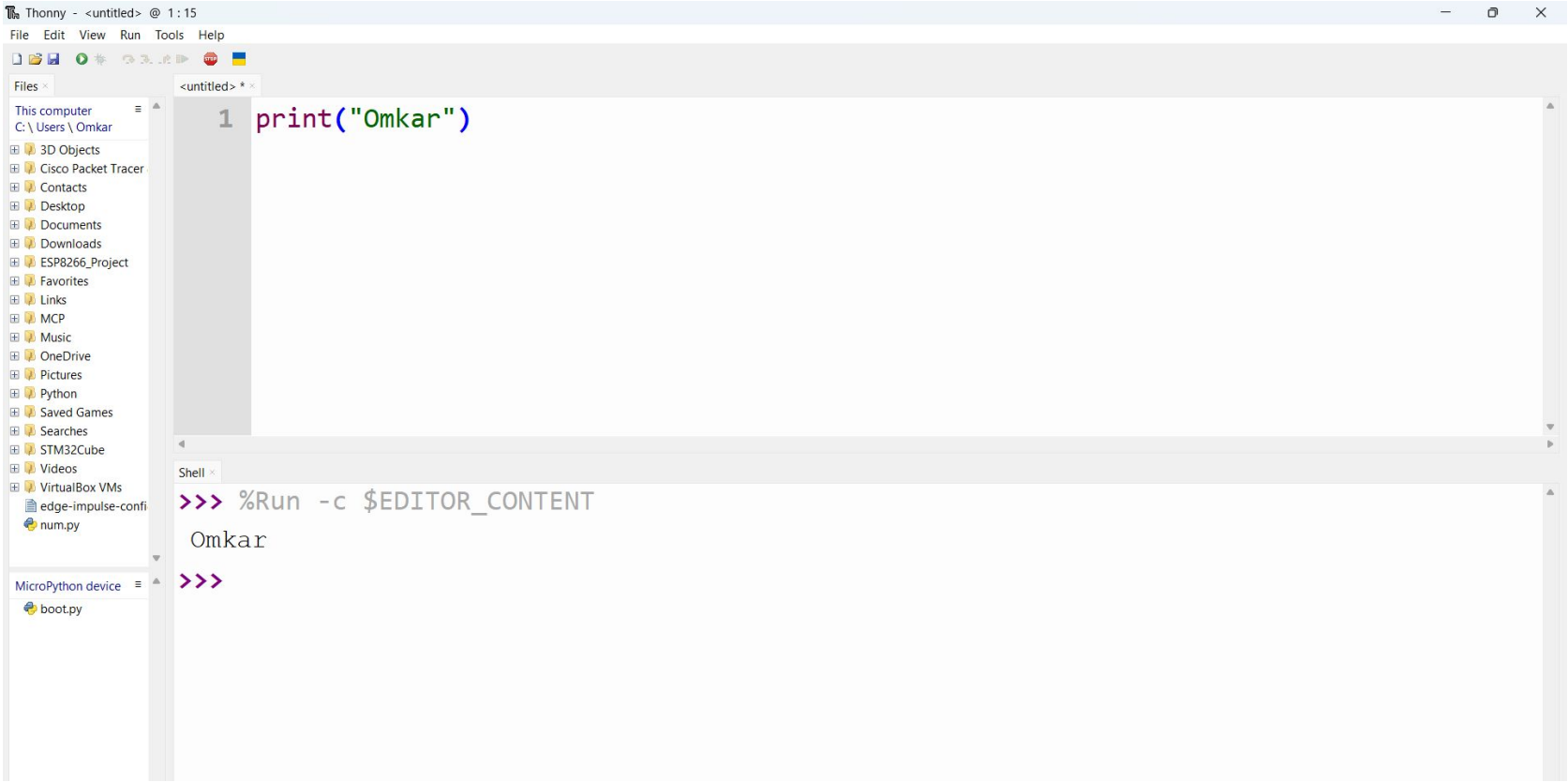
Tick the Erase flash before installing.

Click on Install button.



Done message will appear after installation is complete then click on close and save to close the both the windows.

Now you will be able run the python code. The output can be seen on shell.



In Micropython to configure GPIO pin of ESP8266, Pin class is used which comes under Machine module. To use this first we need to import that.

from machine import Pin

Pin.OUT => to set GPIO pin as OUTPUT

Pin.IN => to set GPIO pin as INPUT

Pin.on() => to prived HIGH state

Pin.off() => to provide LOW state

Task : Turn onboard NodeMCU LED on and off.

```
MicroPython v1.19.1 on 2022-06-18; ESP module with ESP8266
Type "help()" for more information.
>>> from machine import Pin
>>> pin=Pin(2,Pin.OUT)
>>> pin.on()
>>> pin.off()
>>>
```

GPIO pin numbering is based on ESP8266 chip numbering, not some “logical” numbering of a particular board.

Task : Blinking the onboard LED.

sleep class is used to create delay. It comes under time module. It takes second as Input parameter.

from time import sleep

```
from machine import Pin
from time import sleep
pin=Pin(2,Pin.OUT)
while(1):
    pin.on()
    sleep(1)
    pin.off()
    sleep(1)
```