

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 antenna and other necessary components like External SPI flash, GPIO pin, LED.

ESP-12E module was made by [AI-Thinker](#).

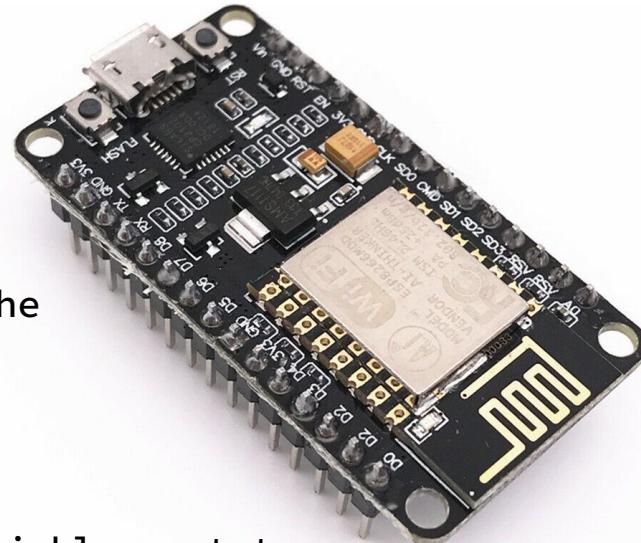
**To program this module external Programmer need to be connect.



What is NodeMCU ?

The NodeMCU is a Open source popular IoT development board.

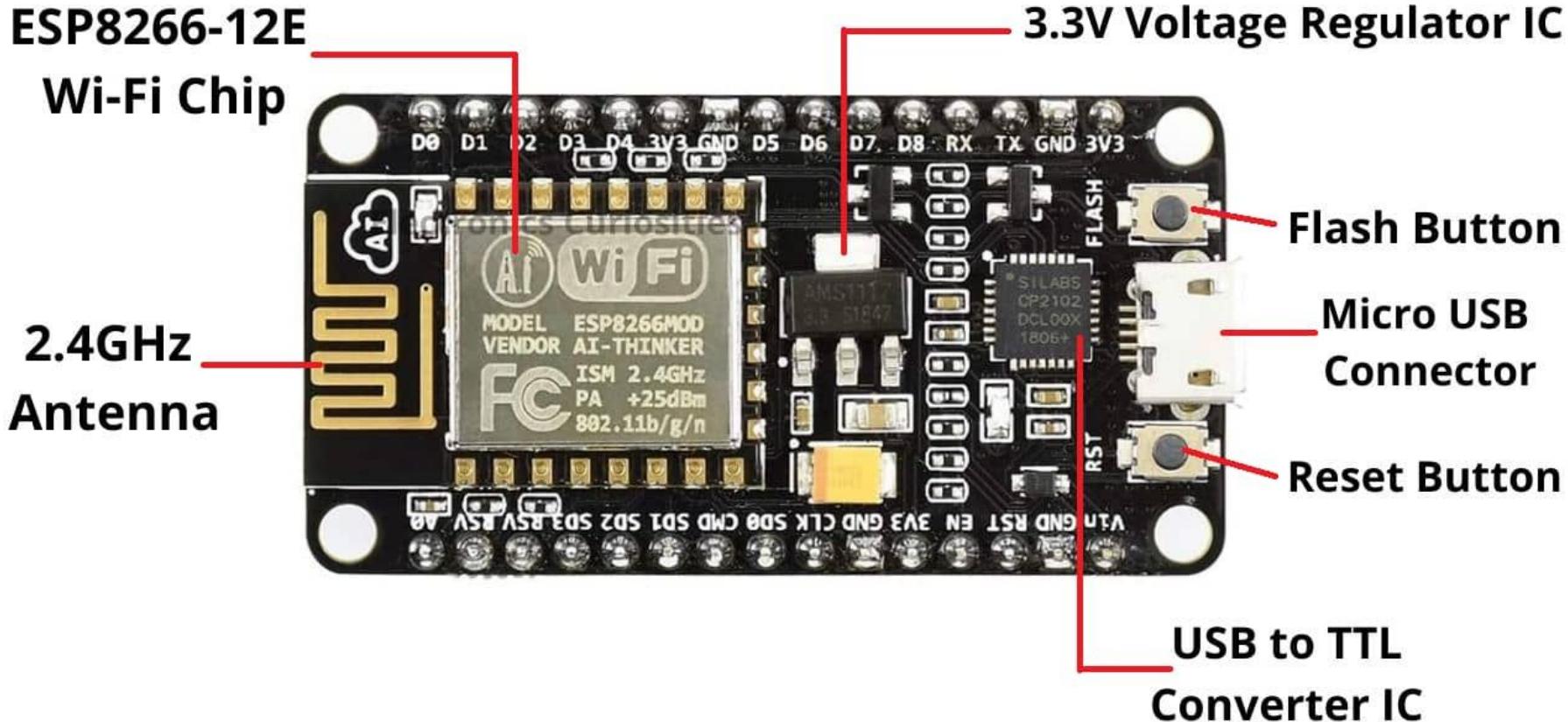
It is based on the ESP-12E module (which contains the ESP8266 SoC).



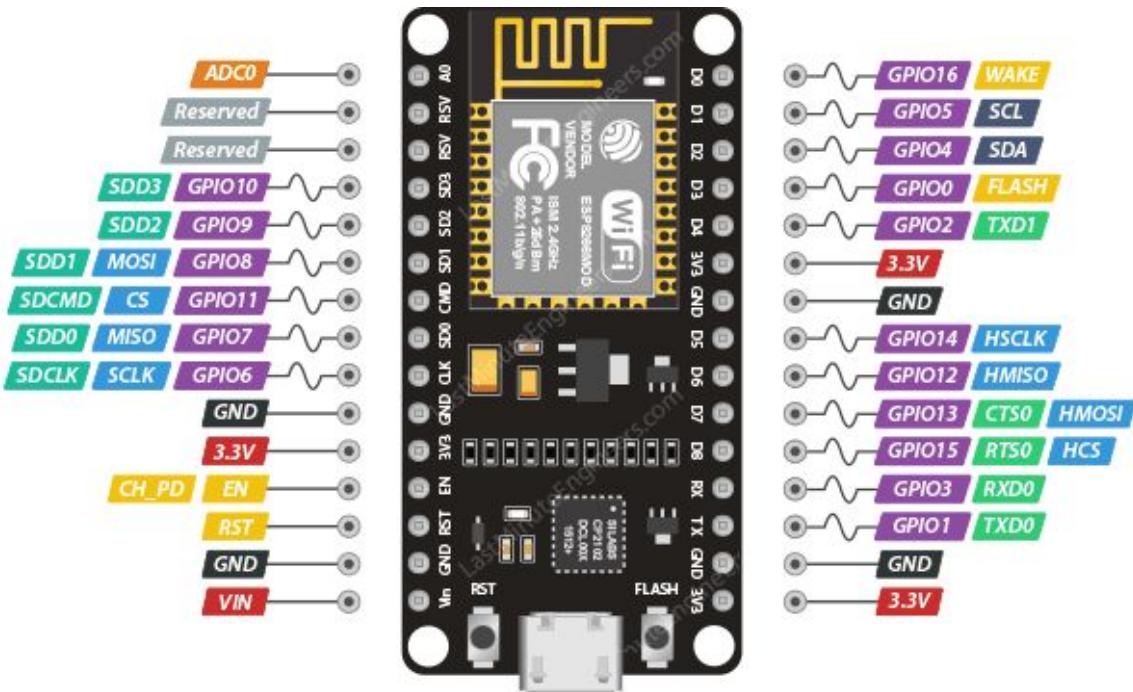
What is Development Board ?

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

ESP8266 NODE MCU



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)

Built In LED :

Clock Speed : 80Mhz

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

SRAM : 64 KB

Flash Memory : 4 MB

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

Digital Pins : 9

Analog Pin : 1

Input Voltage : 3.3V - 3.6 V

ADC: 10 Bit

Communication Protocol : UART, SPI, I2C

PWM : 10 Bit

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

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



MicroPython



NodeMCU Programming with Arduino IDE

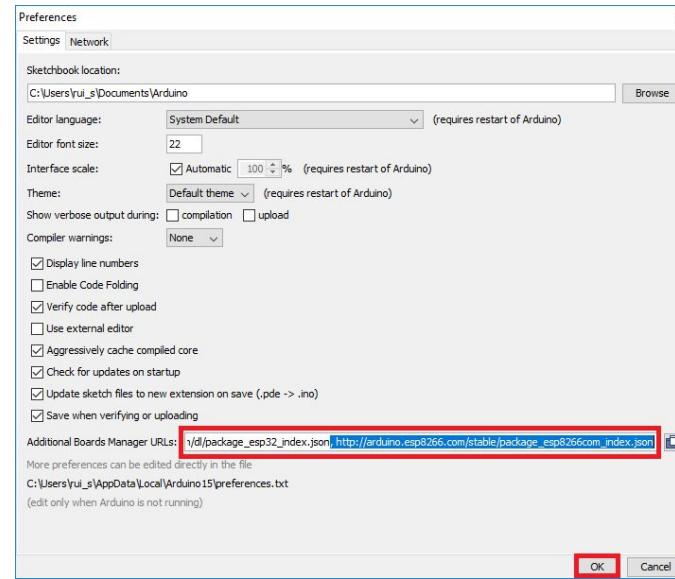
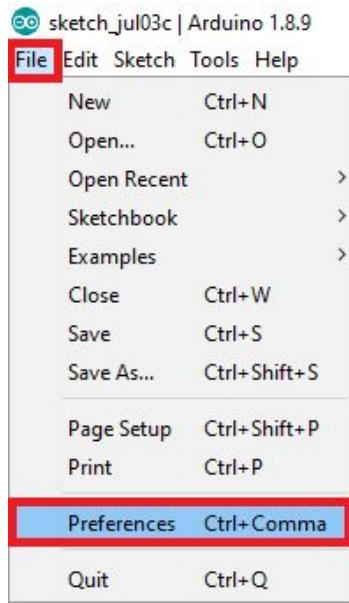


Step 1:

Download Arduino IDE (<https://wwwarduino.cc/en/software>)

Step 2:

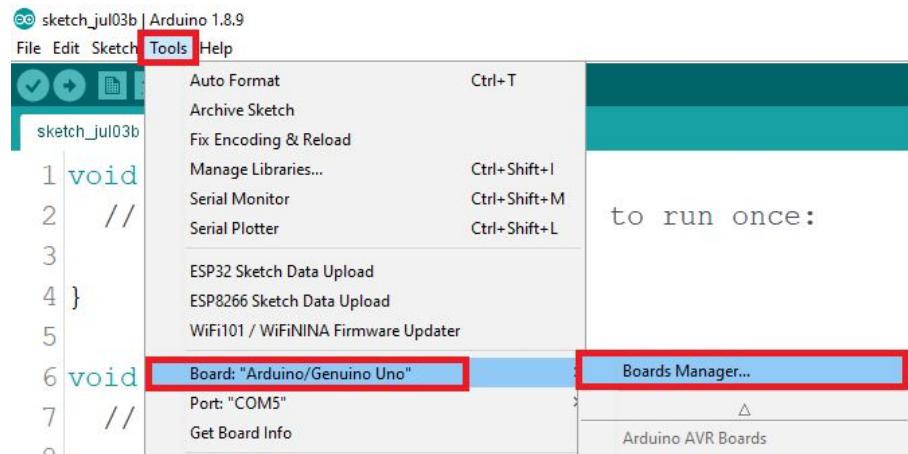
Open Arduino IDE -> Go to Files -> Preferences , Past the URL
(http://arduino.esp8266.com/stable/package_esp8266com_index.json)



NodeMCU Programming with Arduino IDE

Step 3:

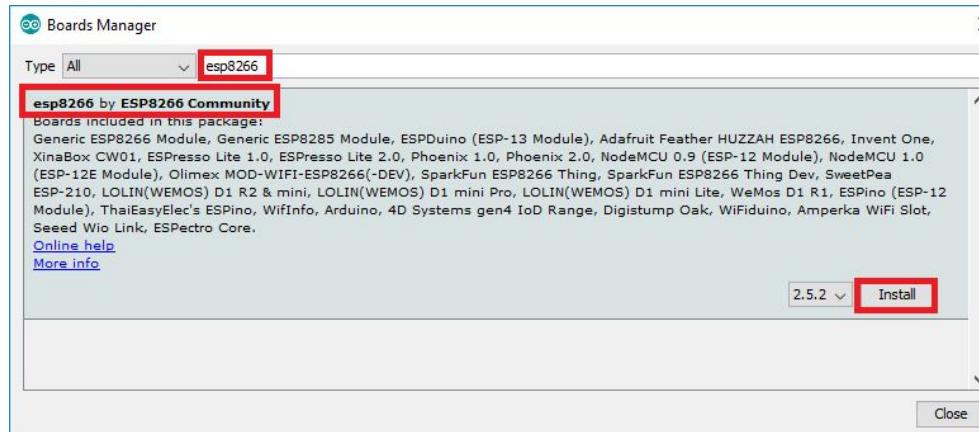
Tools > Board > Boards Manager...



to run once:

Step 4:

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



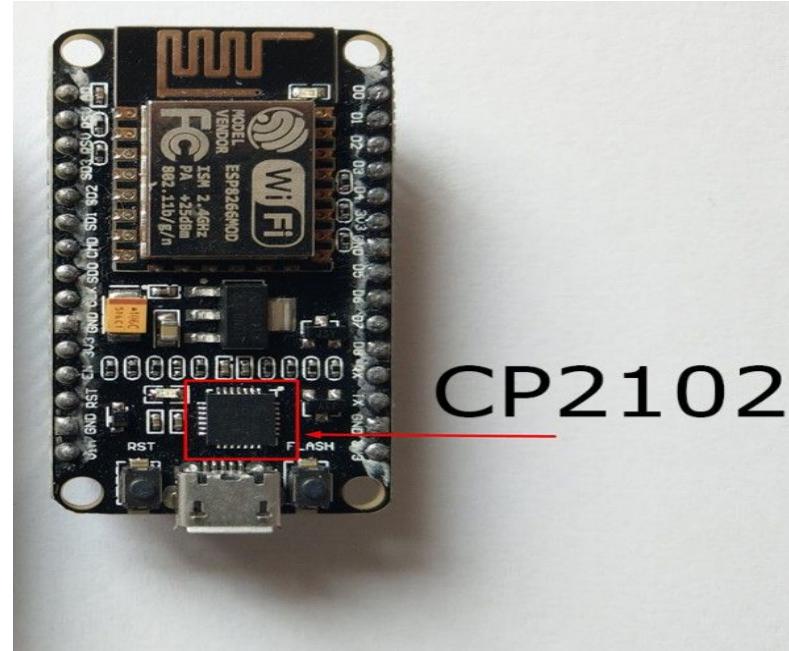
Drivers for NodeMCU

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.

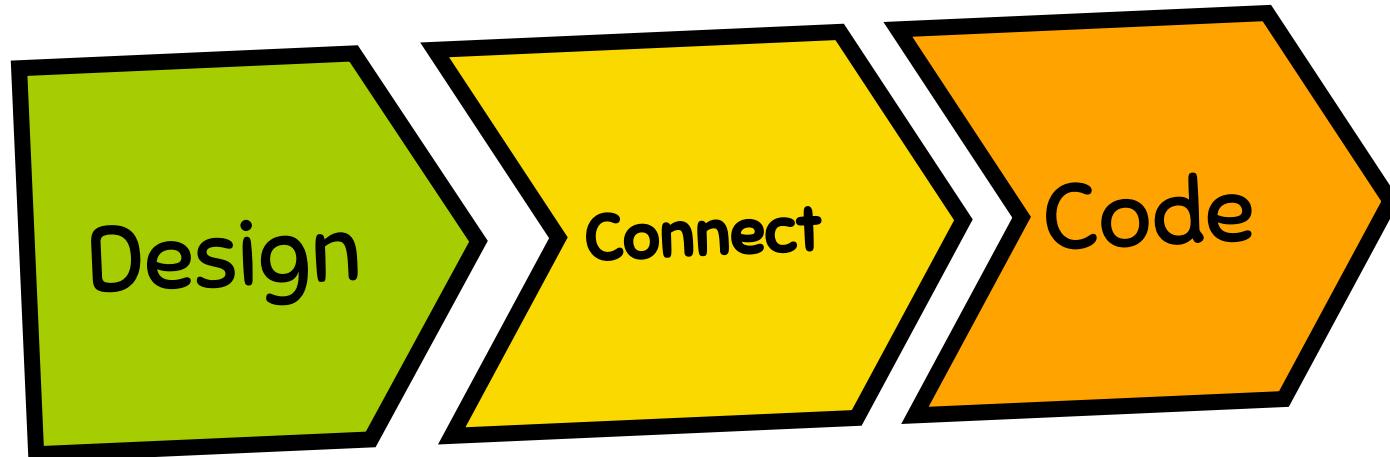
Their primary function is to enable communication and control of various hardware peripherals.



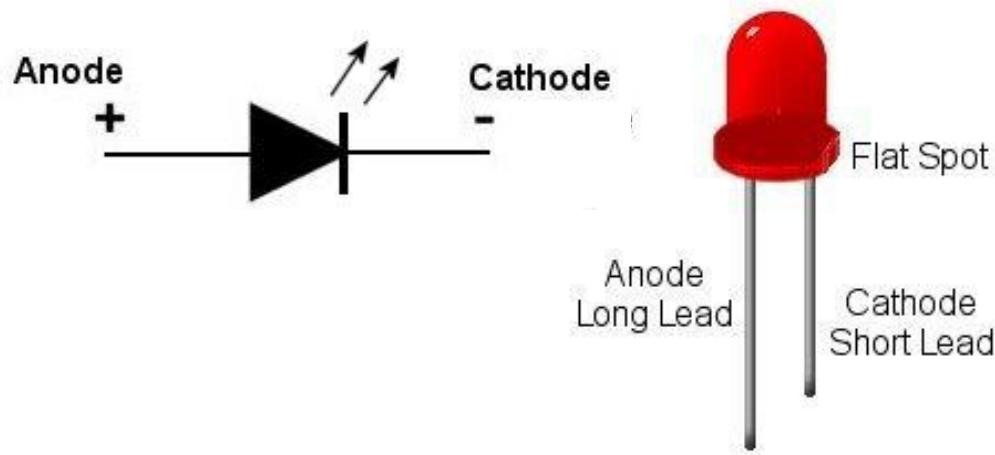


IT'S TIME TO
HANDS-ON

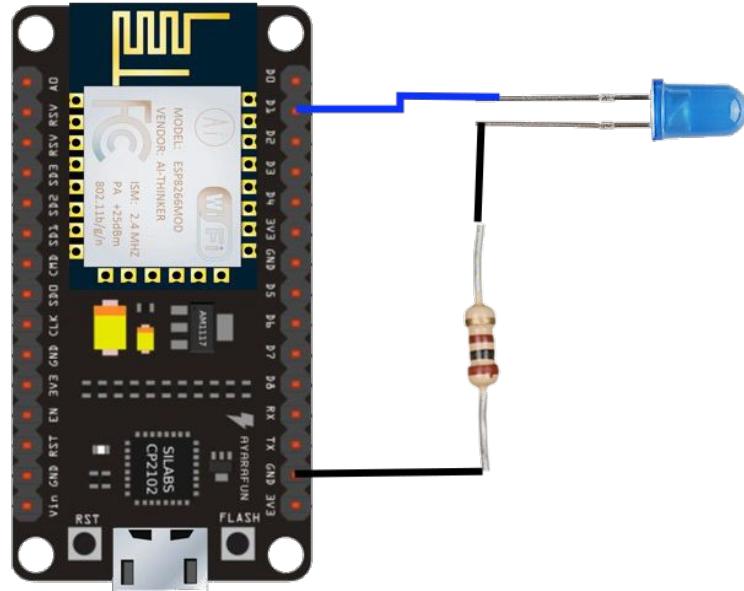
OUR PROCESS IS EASY



* LED



Design



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).

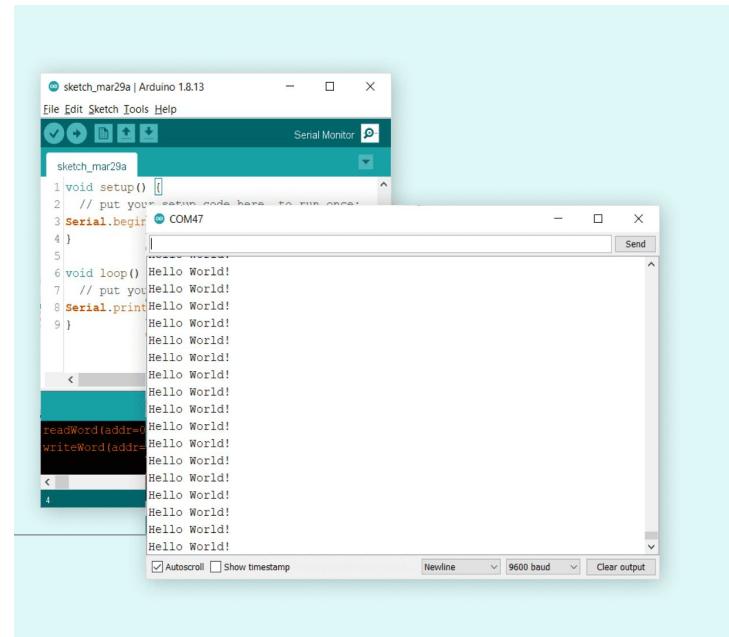
Serial.begin(Baud_rate);

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);
}
```

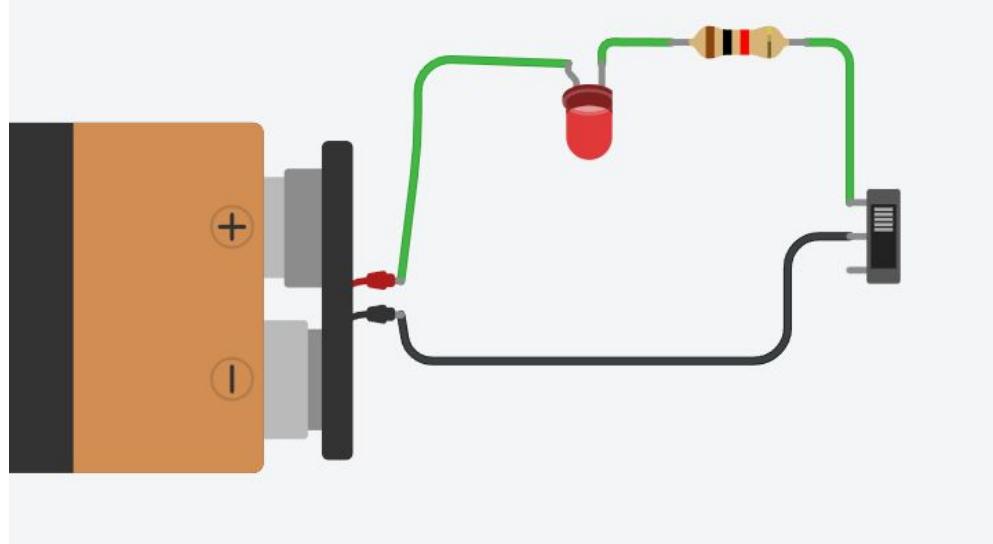
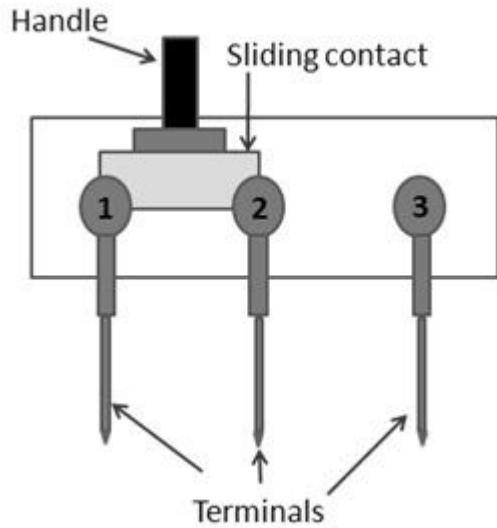
Task 3. Build Traffic light using 3 LED

```
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

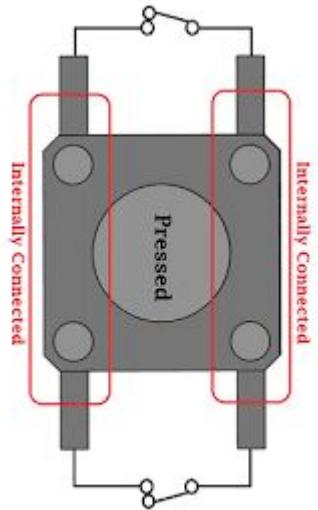
Task 4. Print the state of sliding switch on serial monitor

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

void loop()
{
    If (digitalRead(pin_number)==HIGH ){
        Serial.println("LED is ON");
    }

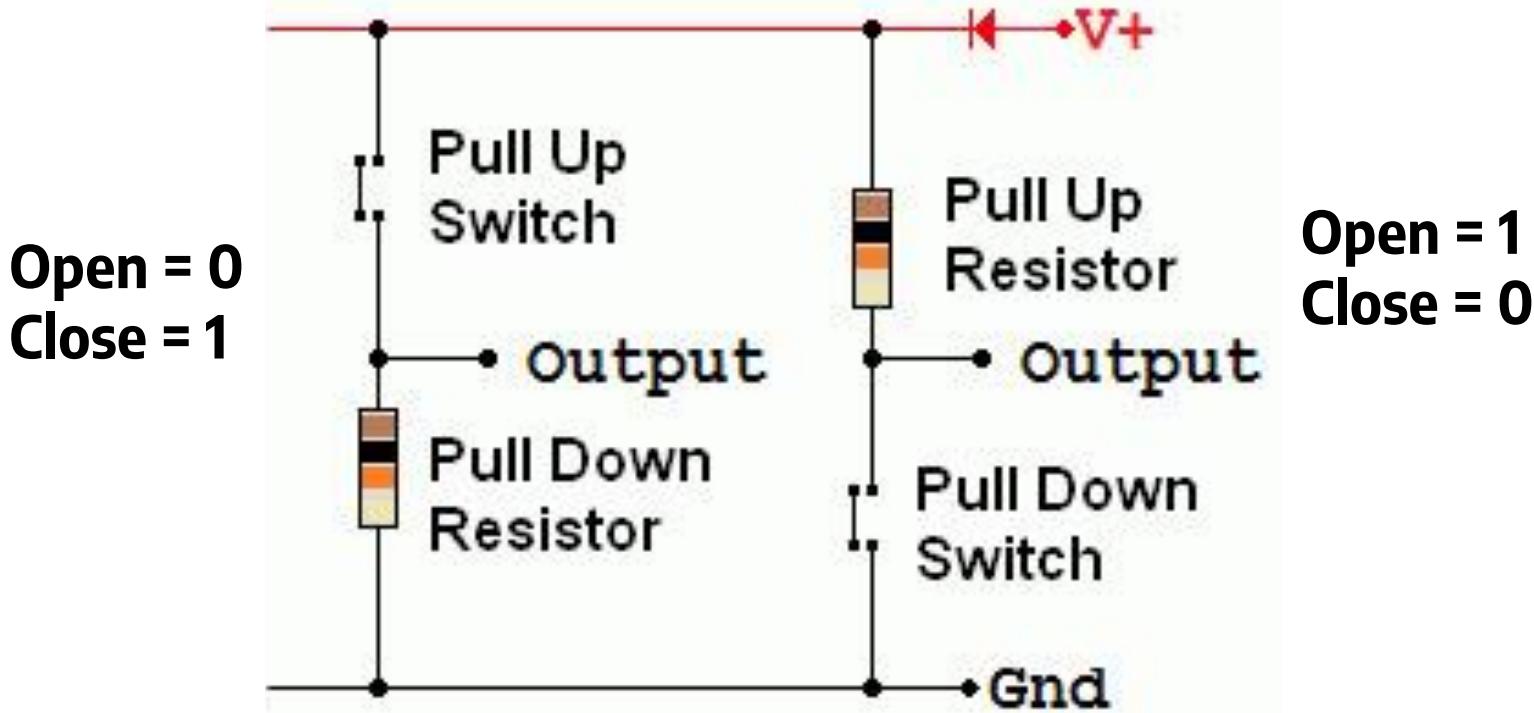
    else {
        Serial.println("LED is OFF");
    }
}
```

Switches

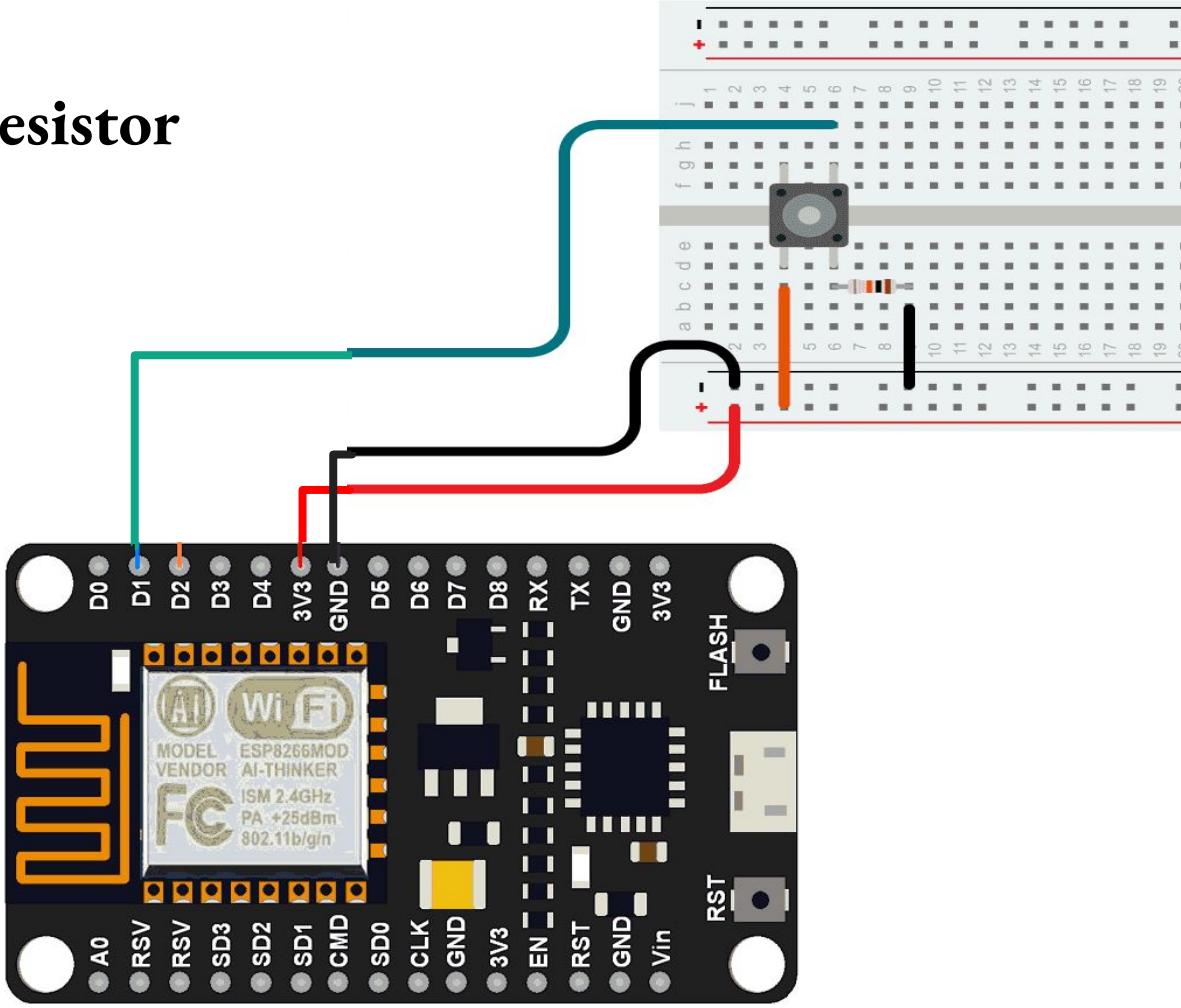


Curved legs (facing each other) are internally connected.

Pull Up and Pull Down Resistor



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.

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

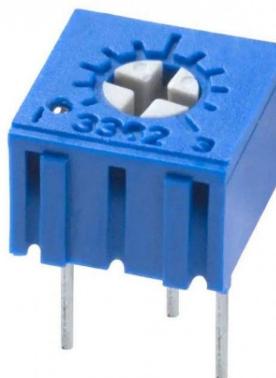
$$\text{Resolution} = \frac{3.3v - 0}{1024} = 3.2mV$$

Potentiometer

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



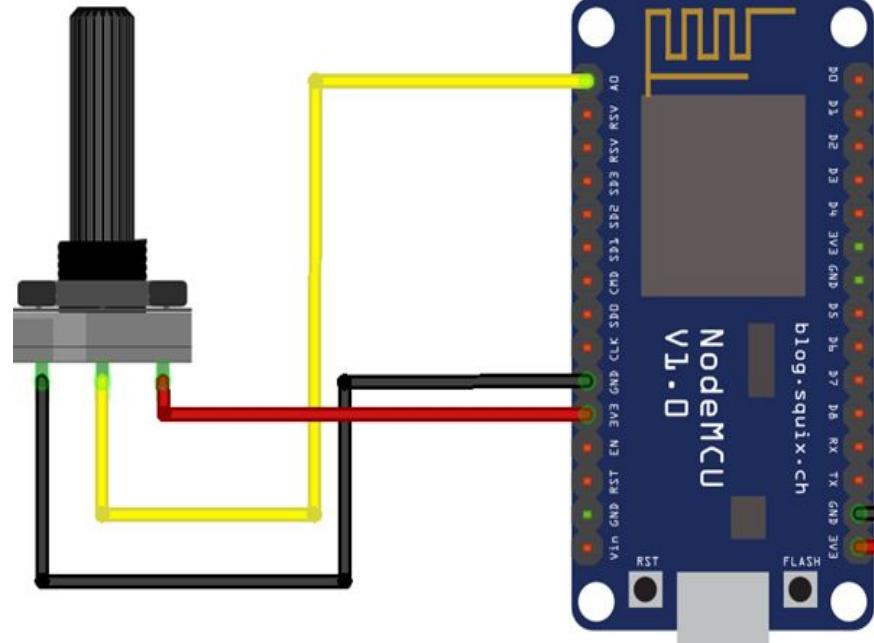
it is widely used in electronic circuits as a variable resistor to adjust voltage, control volume, or set the brightness of a display, among other applications.



Task 6. Interface Pot with NodeMCU and print its values

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

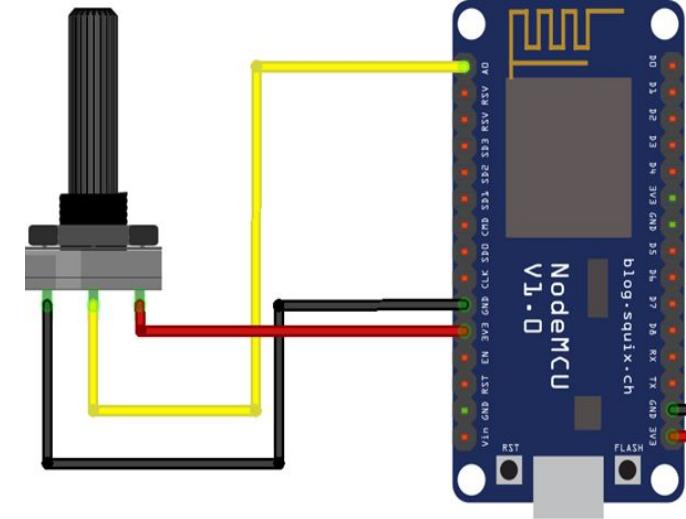
void loop()
{
    Int sensVal= analogRead(pin_number);
    Serial.println(sensVal);
}
```



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

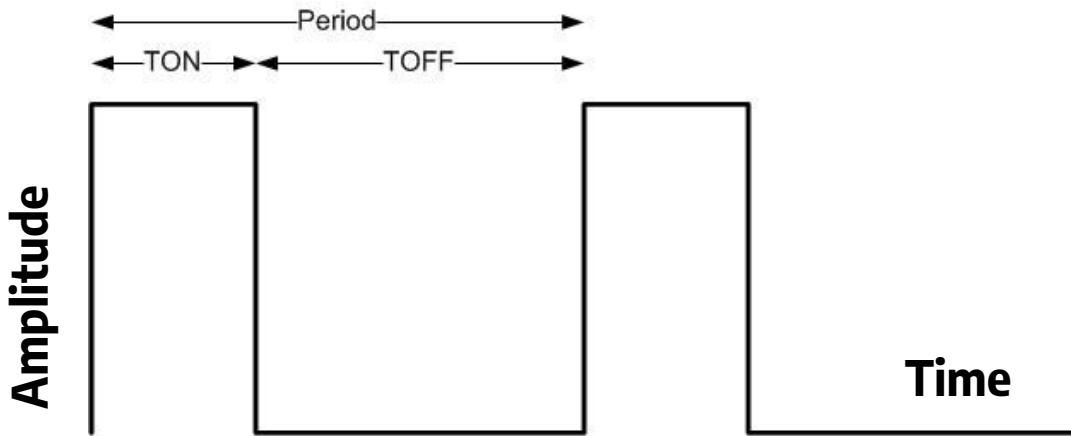
Value is range between **(0 to 1023)**

PWM

Pulse Width Modulation

Best and efficient way of gating
variable DC voltage.





$$\text{Period} = \text{TON} + \text{TOFF}$$

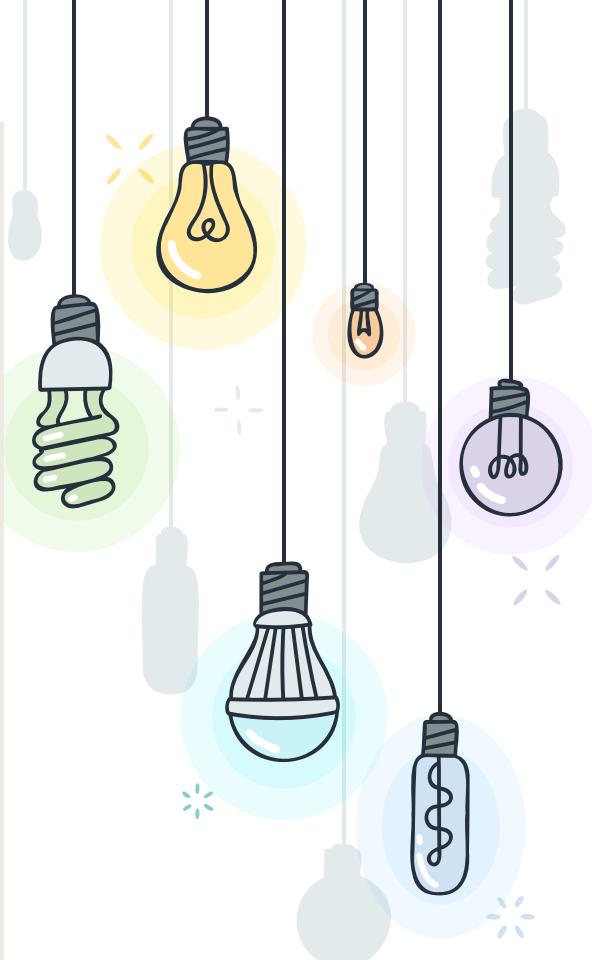
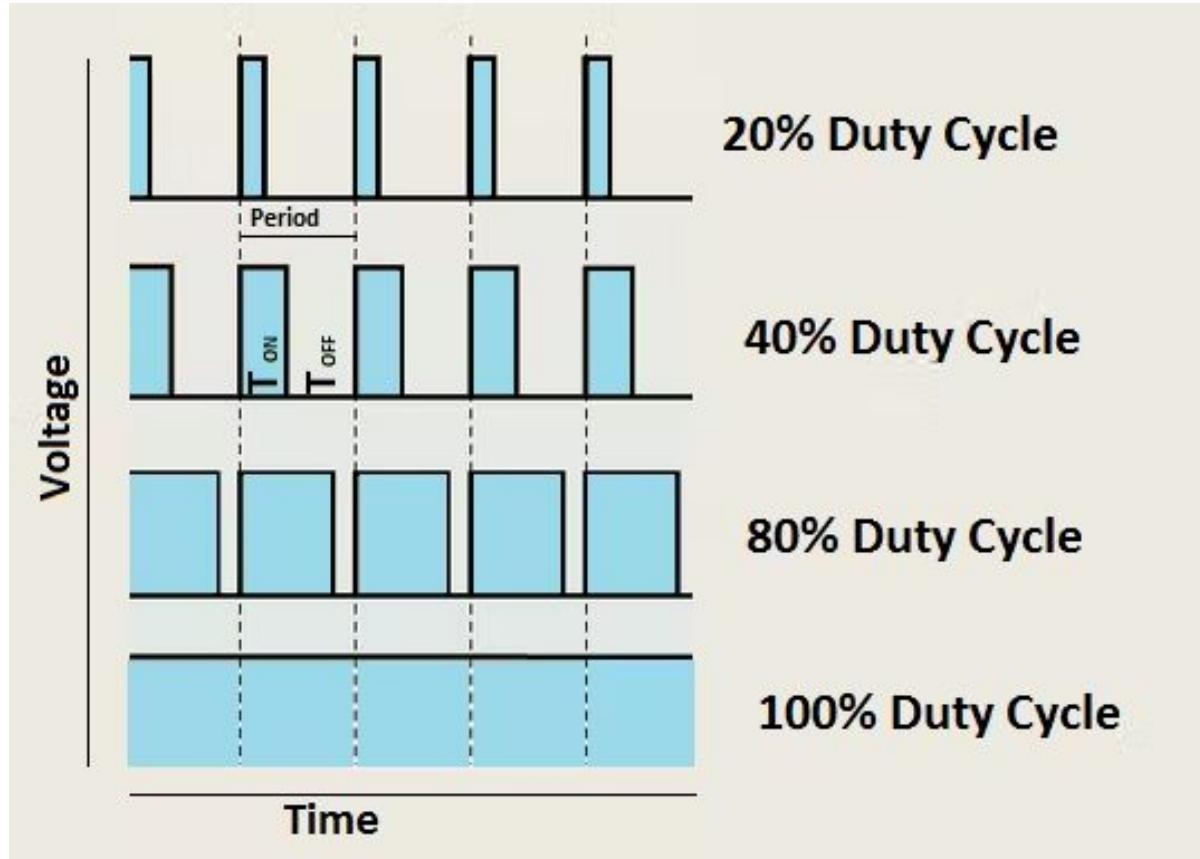
$$\text{Frequency} = 1 / \text{Period}$$

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

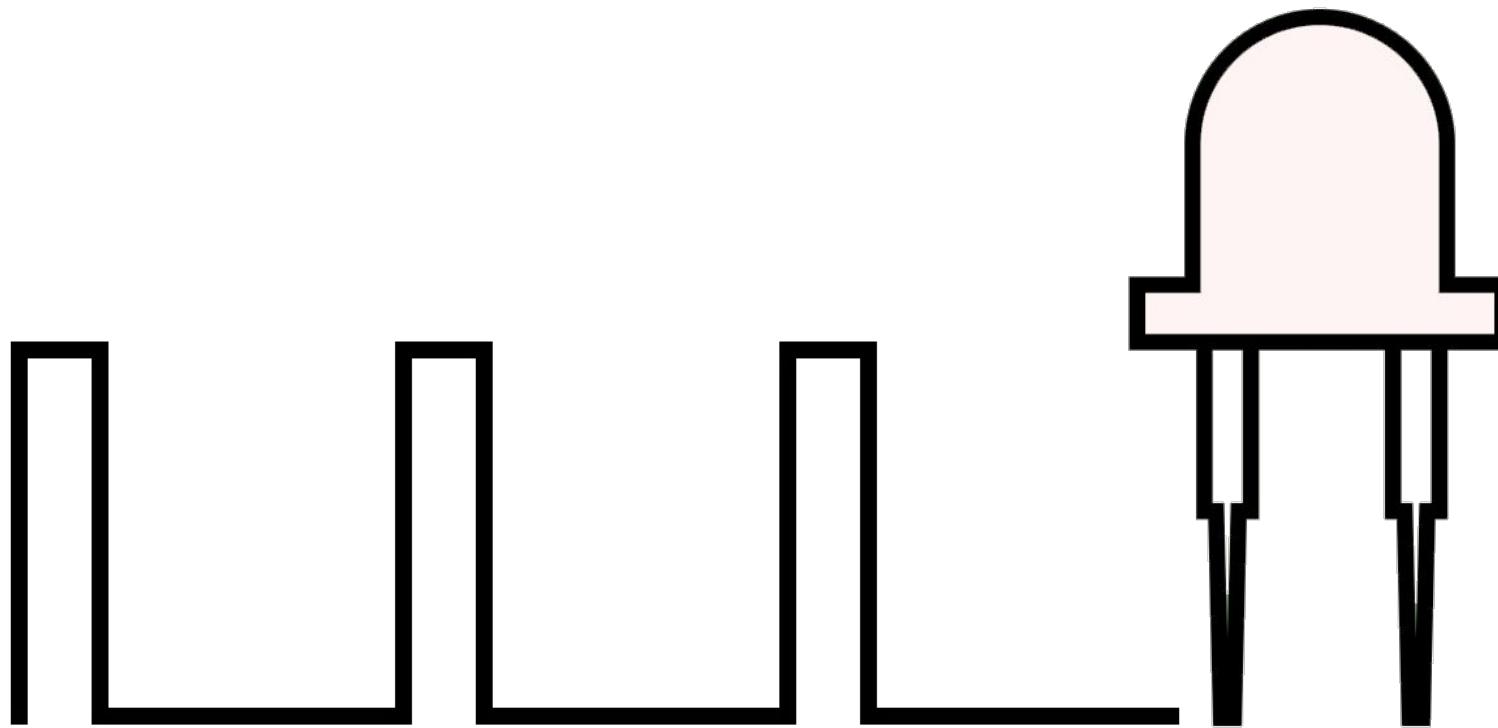
$$T_{\text{on}} = 3.3\text{v}$$

$$T_{\text{off}} = 0\text{v}$$





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

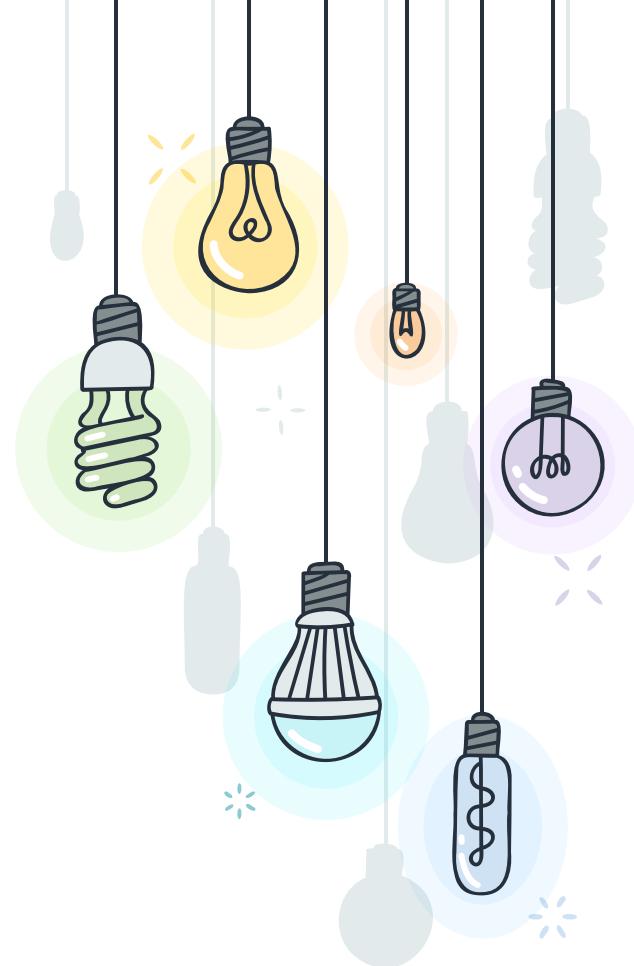
How to get such signal ?

analogWrite(pin, 0-1023)

0 -> 0% Duty cycle

512 -> 50% Duty cycle

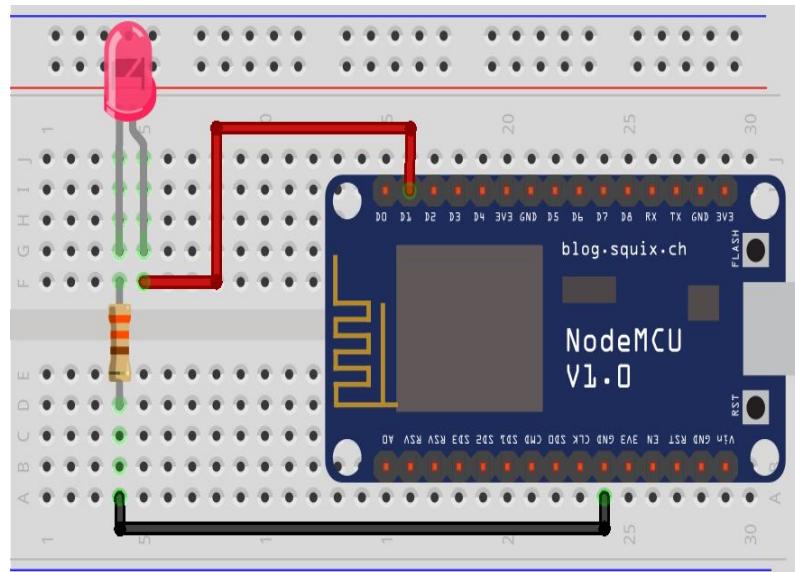
1023 -> 100% Duty cycle



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.

7	Application Layer	Human-computer interaction layer, where applications can access the network services
6	Presentation Layer	Ensures that data is in a usable format and is where data encryption occurs
5	Session Layer	Maintains connections and is responsible for controlling ports and sessions
4	Transport Layer	Transmits data using transmission protocols including TCP and UDP
3	Network Layer	Decides which physical path the data will take
2	Data Link Layer	Defines the format of data on the network
1	Physical Layer	Transmits raw bit stream over the physical medium

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

130

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.

(IEEE 802.11) official standard, 802.11

It is mostly half duplex.



Half duplex 
only one can send / Receive at a time

Full duplex 
Send or receive simultaneously.

WiFi use (ISM, industrial, scientific, 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

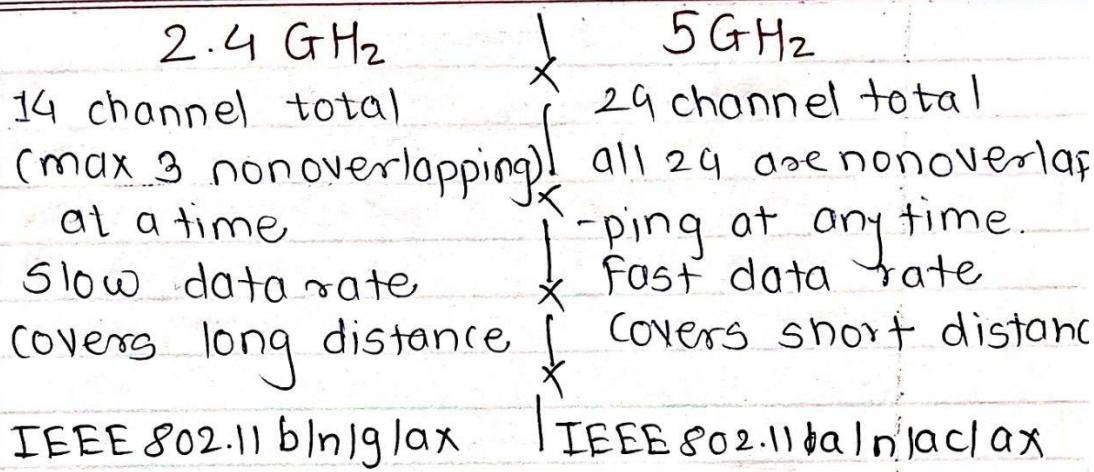
(Public → Private IP) NAT (Private → Public IP)

QoS.

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

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

132



WPA: WiFi Protected Access.

WPA 1 (2003)

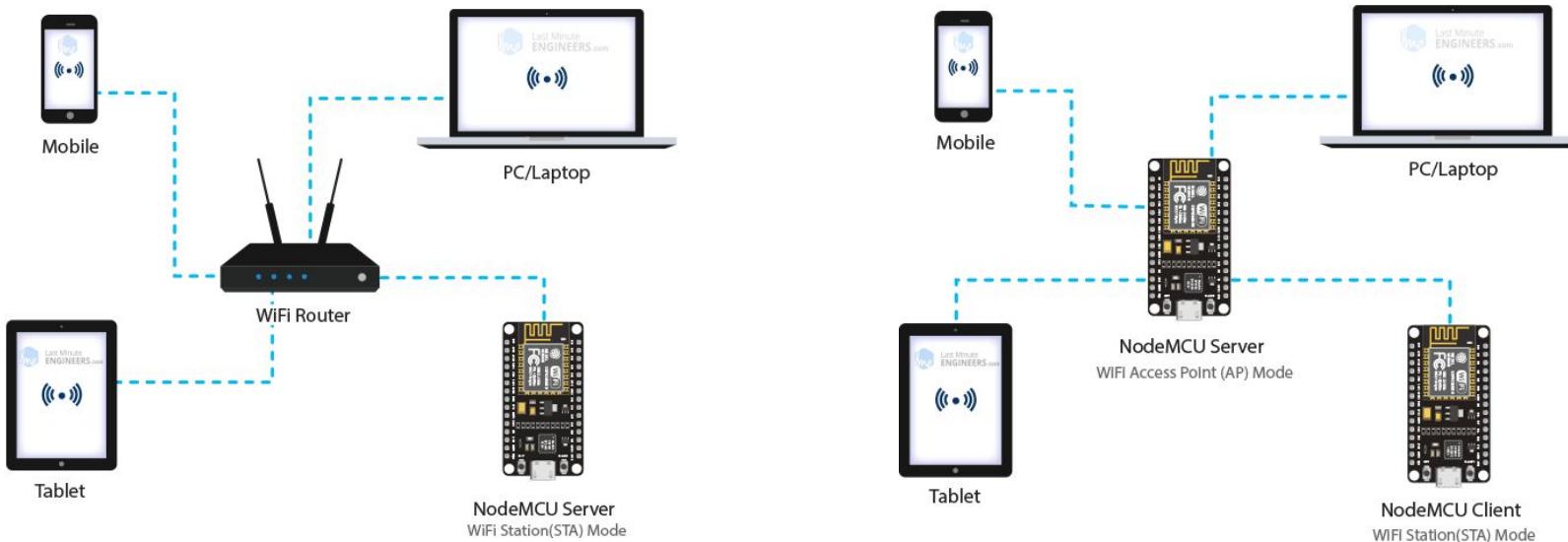
WPA 2 (802.11i) 2005 (AES)

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("Successfully Connected to Home ");
    Serial.println(WiFi.localIP());
    Serial.println(WiFi.macAddress());
}
void loop() {
```

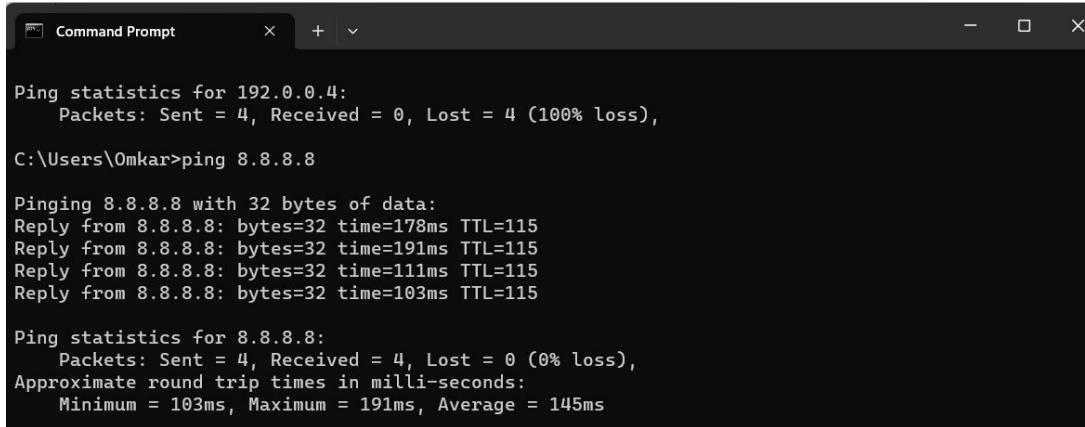
Is it really connected in LAN ?

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.



```
Command Prompt

Ping statistics for 192.0.0.4:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\Users\Omkar>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=178ms TTL=115
Reply from 8.8.8.8: bytes=32 time=191ms TTL=115
Reply from 8.8.8.8: bytes=32 time=111ms TTL=115
Reply from 8.8.8.8: bytes=32 time=103ms TTL=115

Ping statistics for 8.8.8.8:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 103ms, Maximum = 191ms, Average = 145ms
```

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:

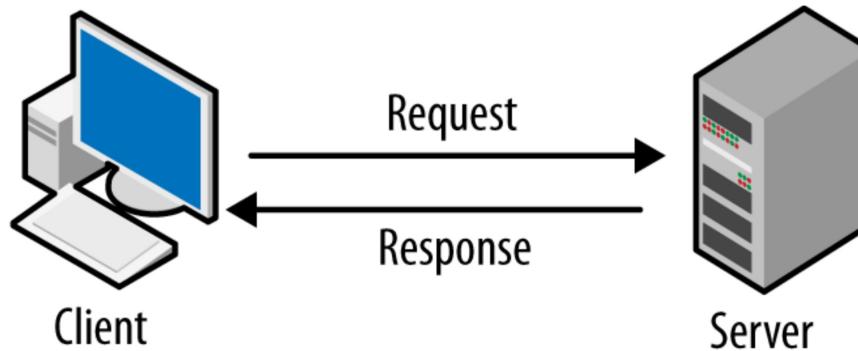
A server is a device or software application that provides services, resources, or information to clients.

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=\"Toggle 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)

IoT CLOUD

1. Creating an Arduino Account

To start 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](#).

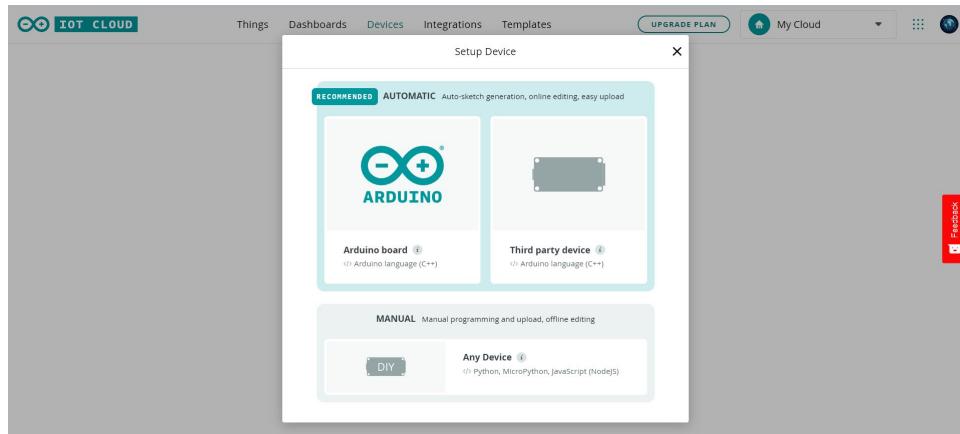
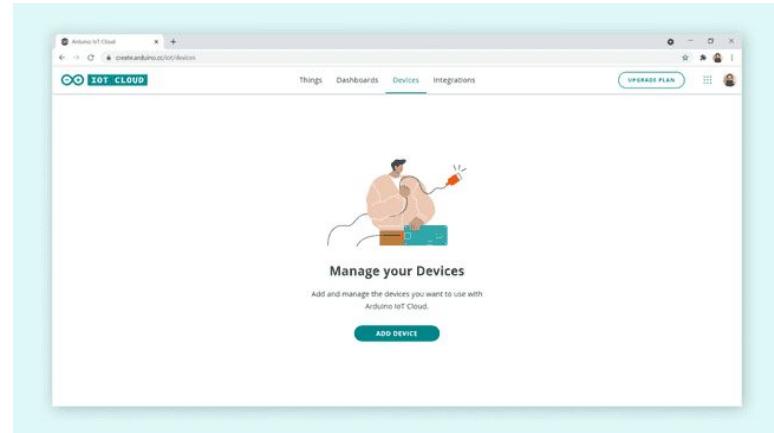
The screenshot shows the Arduino website's main landing page. It features a large green header with the text "Welcome to Arduino" and "Learn and create, with the ever-growing platform". Below this is a "SIGN IN" button and a link to "Create one." A "Hey Kids!" section with a "Sign in" link for users under 14 is also visible. The main content area has a green background with three small clouds, each containing a microcontroller board, connected by dashed lines. The text "Connect smart devices in minutes with Arduino cloud" is displayed above a "DISCOVER MORE" button. At the bottom, there are links for "TERMS OF SERVICE", "PRIVACY POLICY", "SECURITY", and "COOKIE SETTINGS".

The screenshot shows a web browser window with the Arduino website loaded. In the top right corner, the four-dot menu icon is open, revealing a dropdown menu with options: "IoT Cloud", "Web Editor", and "Manager for Linux". The "IoT Cloud" option is highlighted. The main content area of the browser shows a "Missing something?" section with a "Check out our store and" message.

Step 1: Setting up the device

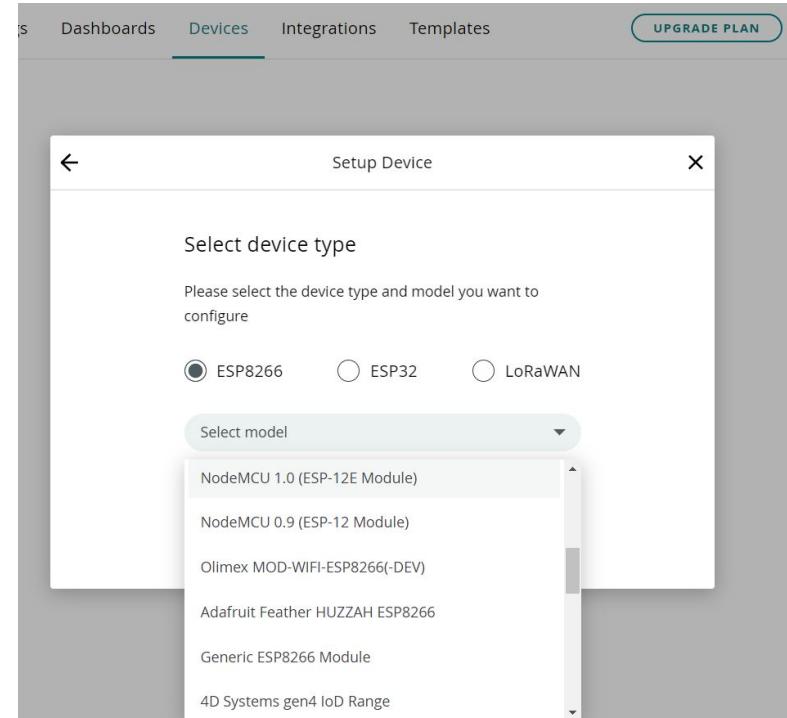
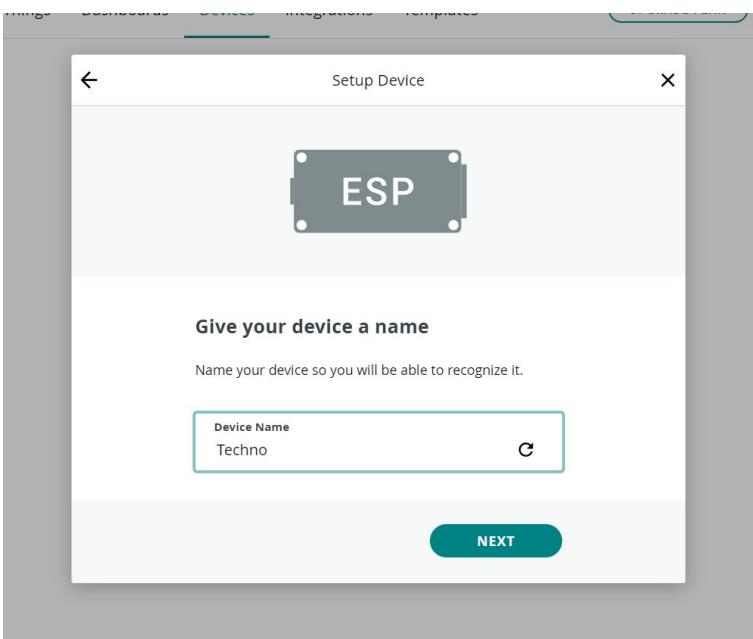
1. Once in the cloud, click on the "**Devices**" tab.

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.

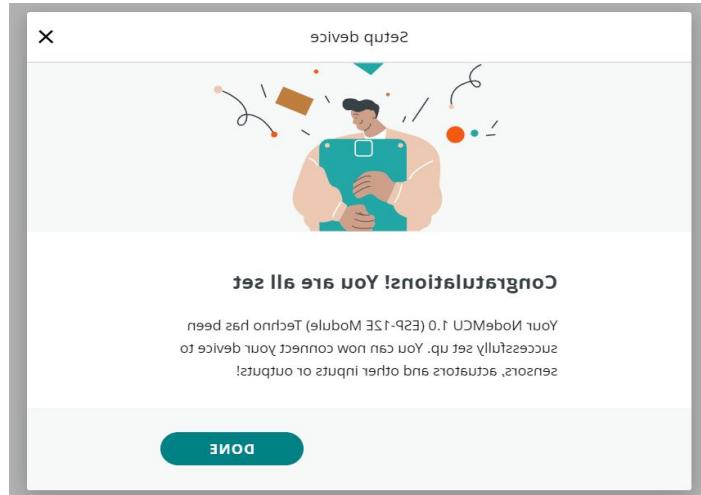


4. Now the board needs a name. You can give any name as per your choice.

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**".

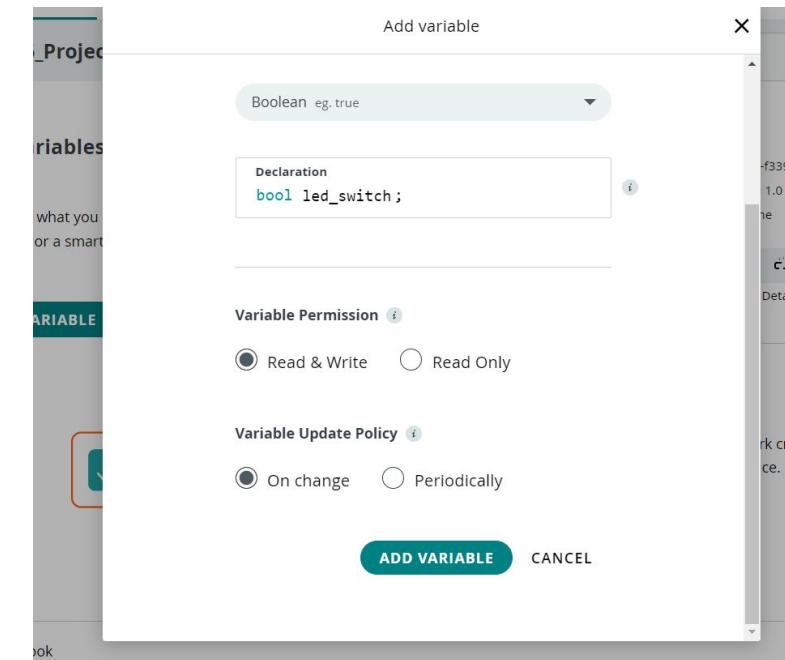
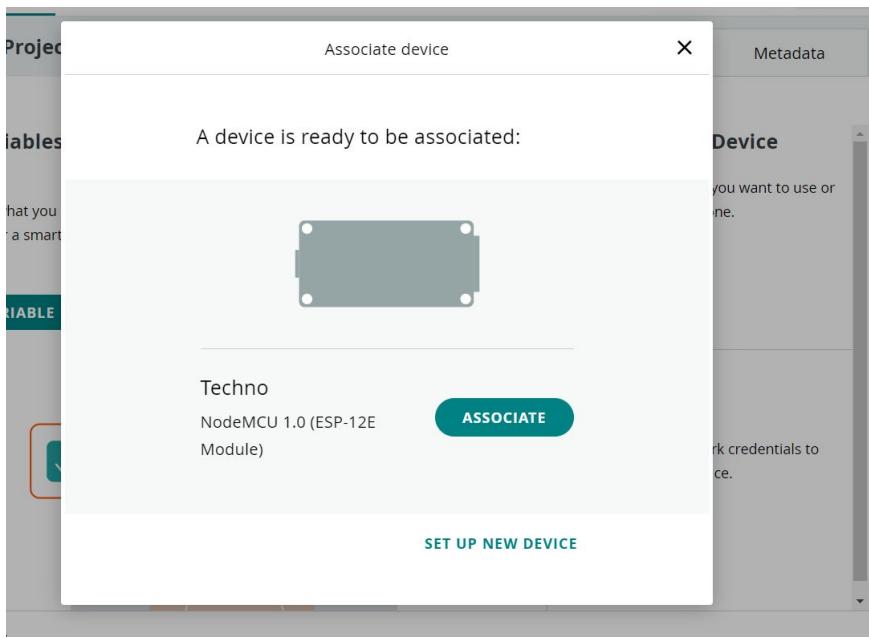


A screenshot of the Arduino Cloud IoT setup device configuration screen. It shows a "Setup Device" section with "Device ID" and "Secret Key" fields filled in. Below the fields is a yellow warning box with an exclamation mark containing the text "Secret key cannot be recovered. Please keep it safe, if you lose it you will have to delete and setup your device again." At the bottom left is a checked checkbox labeled "I saved my device ID and Secret Key" and a blue "CONTINUE" button.

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.



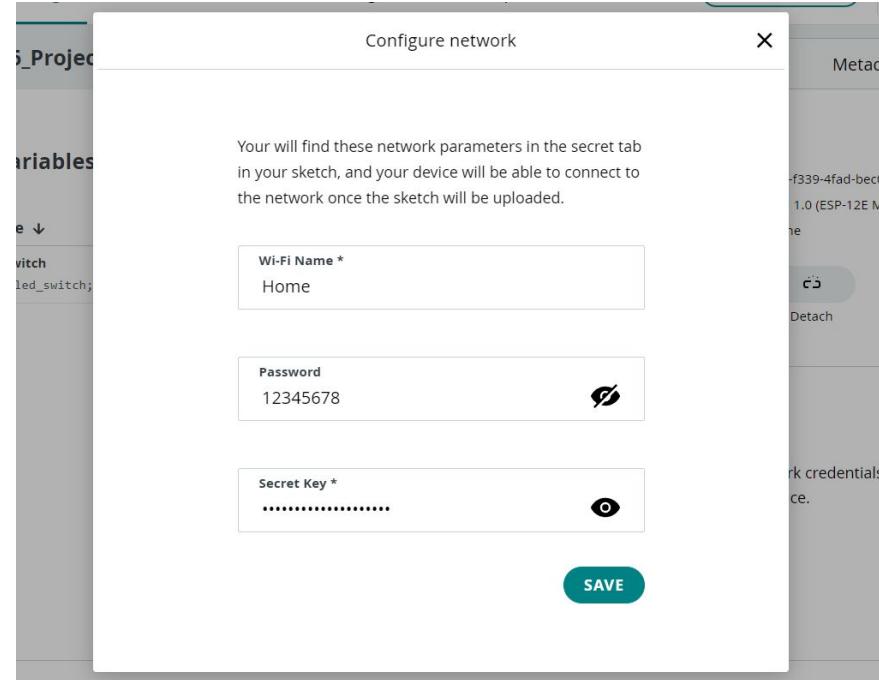
Click on "**Add variable**" button. This will open a window where you need to fill in variable information. Now, let's add the **led_switch** variable. The data type for this variable is **boolean**, the permission is **read write**, and update policy is **on change**.

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.

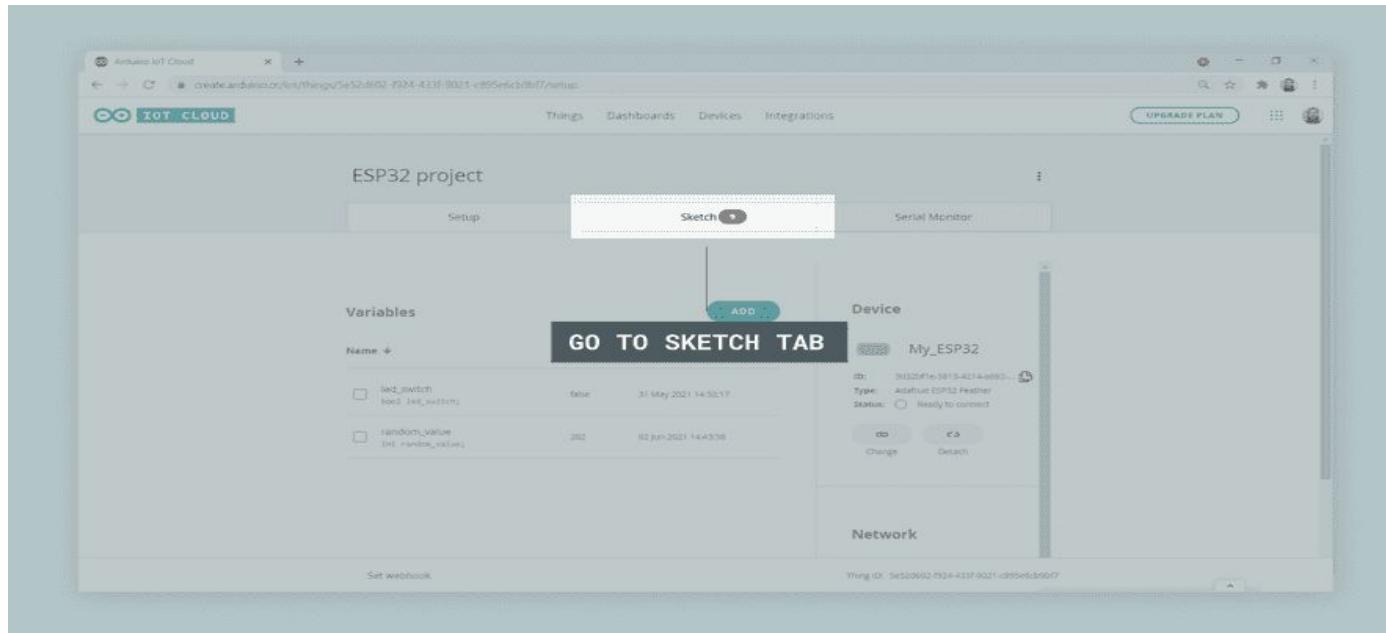
First, click on the button in the "**Network Section**".



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.

A screenshot of the IoT Cloud interface showing the "Things" tab selected in the top navigation bar. The main area displays a list of devices, with one entry for "My_ESP32" highlighted. A large, bold button in the center of the screen says "GO TO THINGS TAB".

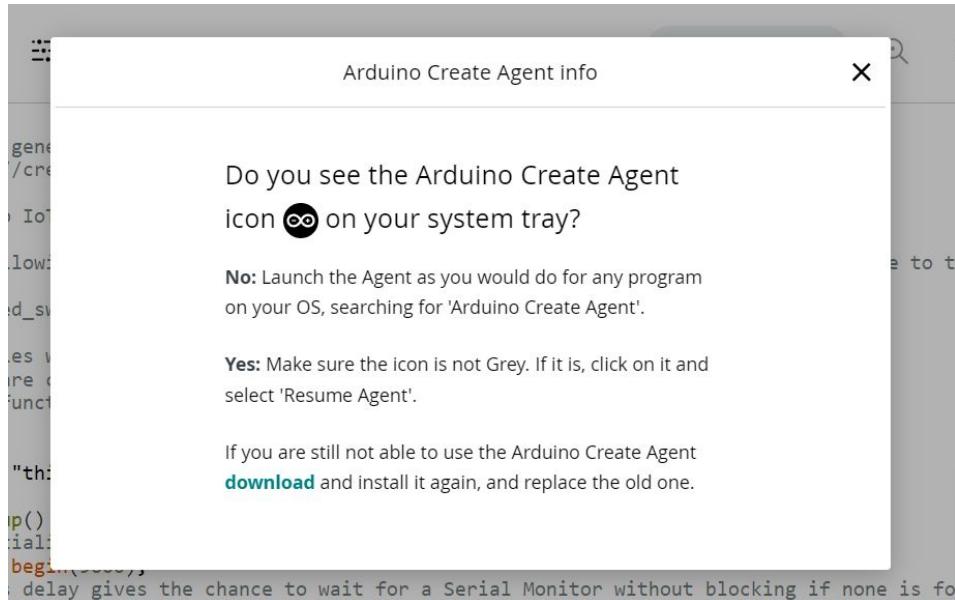
A screenshot of the "Cloud Variables" section in the IoT Cloud interface. The top navigation bar shows "OT CLOUD" and the "Things" tab is selected. The main content area includes sections for "Cloud Variables" (with a "ADD VARIABLE" button), "Associated Device" (with a "Select Device" button), and "Network" (with a "Configure" button). On the left, there is a decorative illustration of a person interacting with various icons representing data and connectivity.

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

We can then rename our Thing something appropriate to what we are doing, in this case I 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.

Once 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.

The screenshot shows the Arduino IDE interface. At the top, there are two circular icons: one with a checkmark and another with a right-pointing arrow. To the right of these is the text "Techno - NodeMCU 1.0 (ESP-12E Module)" and "Port: COM5". Further right are buttons for "Open full editor", a magnifying glass for search, and a list icon. Below the header is a bell icon. The main area contains the Arduino sketch code. Lines 58 and 59 are highlighted with a grey background. A green bar at the bottom displays the message "Success: Done Uploading ESP8266_Project_aug10a".

```
37     setDebugMessageLevel(2);
38     ArduinoCloud.printDebugInfo();
39 }
40
41 void loop() {
42     ArduinoCloud.update();
43     // Your code here
44
45 }
46
47
48
49
50 /*
51   Since LedSwitch is READ_WRITE variable, onLedSwitchChange() is
52   executed every time a new value is received from IoT Cloud.
53 */
54 void onLedSwitchChange() {
55   // Add your code here to act upon LedSwitch change
56   if(led_switch==1){
57     digitalWrite(D4,LOW);
58   }else{
59     digitalWrite(D4,HIGH);
60   }
61 }
```

Success: Done Uploading ESP8266_Project_aug10a

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](#)

```
1+ /*  
2+ Sketch generated by the Arduino IoT Cloud Thing "ESP32 project"  
3+ https://create.arduino.cc/projecthub/Se52d002-f024-433f-9021-ca95e0c0bf7  
4+  
5+ Arduino IoT Cloud Variables description  
6+  
7+ The following variables are automatically generated and updated when changes are made to the Thing  
8+  
9+ int random_value;  
10+ bool led_switch;  
11+  
12+ Variables which are marked as READ/WRITE in the Cloud Thing will also have functions  
13+ which are called when their values are changed from the Dashboard.  
14+ These functions are generated with the Thing and added at the end of this sketch;  
15+ */  
16+  
17+
```

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 give 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.

Google Play Games Apps Movies & TV Books Kids

Search ? 

Arduino IoT Cloud Remote

Arduino

3.3★ 426 reviews | 100K+ Downloads | Everyone

Install on more devices Share

This app is available for all of your devices

Access all your dashboards from your phone

The official companion app for Arduino IoT Cloud

Control your Internet of Things projects

IoT Remote support all the IoT Cloud widgets



App support

More apps to try →

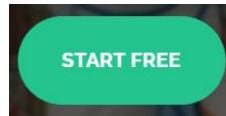
 Learn C++
Coding and Programming
4.5★

Blynk IoT Cloud

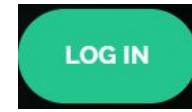


1. Search Blynk IoT on Google, Click on first result <http://blynk.io>

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.

Start by creating your first template
Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.
+ New Template

New Device

Region: bkt | Privacy Policy

Sign Up

Welcome! Fill in your email address and we will send an account activation link.

EMAIL

I agree to [Terms and Conditions](#) and accept [Privacy Policy](#)

Sign Up

[Back to Login](#)

Click on Template option



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

My organization - 9761UQ

MY TEMPLATES

My Templates

BLUEPRINTS BETA

All Blueprints

+ New Device

Start by creating your first template

Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.

+ New Template

Click on new Template

+ New Template

Select Hardware as ESP8266

Connection type as WiFi and

Name the template

Click on Done.

Create New Template

NAME
Techno

HARDWARE
ESP8266

CONNECTION TYPE
WiFi

DESCRIPTION
Description
0 / 128

Cancel 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.

The screenshot shows the Blynk Template Editor interface. At the top, there's a header with a 'B' icon, a search bar, and tabs for Home, Datastreams, Web Dashboard, Automations, Metadata, Events, and Mobile Dashboard. The 'Home' tab is selected. On the left, there's a sidebar with icons for Home, Datastreams, Web Dashboard, Automations, Metadata, Events, and Mobile Dashboard. Below the sidebar, a 'What's next?' section lists five items: 'Configure template', 'Set Up Datastreams', 'Set up the Web Dashboard', 'Add first Device', and 'Get Started'. The main area is titled 'Techno' and shows 'Template settings' with 'ESP8266, WiFi' and a gear icon. A 'Firmware configuration' section contains the following code:

```
#define BLYNK_TEMPLATE_ID "TMPL350zLwHLW"
#define BLYNK_TEMPLATE_NAME "Techno"
```

At the bottom right, it says 'Region: blr1' and 'Privacy Policy'.

Click on Datastreams tab, New tab will open.

Home **Datastreams** Web Dashboard Automations Metadata Events Mobile Dashboard

Click on New Datastreams

+ New Datastream

Virtual Pin
Enum
Location **UPGRADE**
Digital Pin
Analog Pin

Datastreams are regularly flowing sensor data from the device. Use it for actuators.

+ New Datastream

Datastreams

Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.

+ New Datastream

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.

Virtual Pin Datastream

NAME	LED	ALIAS	LED
PIN	V0	DATA TYPE	Integer
UNITS	None		
MIN	0	MAX	1
DEFAULT VALUE 0			
ADVANCED SETTINGS			
Cancel Create			

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

Home [Datastreams](#) Web Dashboard Automations Metadata Events Mobile Dashboard

[+ New Datastream](#)

1 Datastream

	Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Actions
	1	LED	LED		V0	Integer		false	0	1	

Click on Web Dashboard, new tab will be open.

The screenshot shows the 'Web Dashboard' interface. On the left, there's a sidebar titled 'Widget Box' containing four sections: 'Switch' (a green toggle switch), 'Slider' (a horizontal slider set to 8), 'Number Input' (a field set to 0), and 'Image Button' (an orange button). The main canvas area has a title 'Device name' with status 'Online'. It includes fields for 'Device Owner' and 'Company Name', and buttons for 'Tag X' and 'Edit'. Below this is a map with a 'Show map' button and an 'UPGRADE' button. A time selector at the bottom shows 'Last Hour' selected. The bottom right corner displays 'Region: blr1 Privacy Policy'.

Click on switch Widget and move to dashboard.

This screenshot shows the same 'Web Dashboard' interface after the 'Switch' widget has been moved from the sidebar to the main canvas area. The sidebar now shows '1 of 10 widgets'. The main canvas now contains two widgets: the 'Device name' card and the 'Switch' widget, which is identical to the one in the sidebar. The rest of the interface remains the same, including the time selector and the bottom region information.

Click on widget setting and setting window will open

Switch Settings ⓘ

TITLE (OPTIONAL)
LED

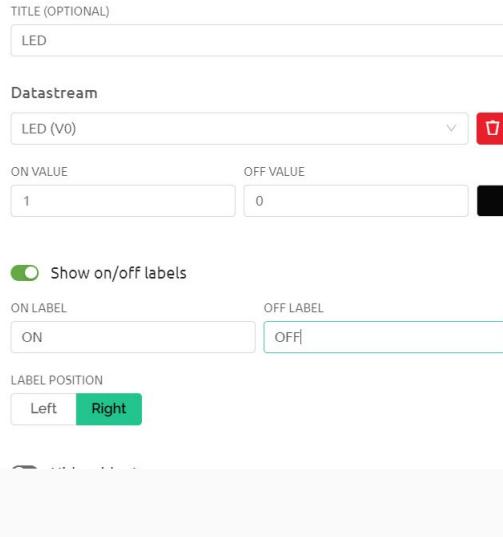
Datastream
LED (V0) ✖

ON VALUE 0 OFF VALUE 1 █

Show on/off labels

ON LABEL ON OFF LABEL OFF

LABEL POSITION Left Right



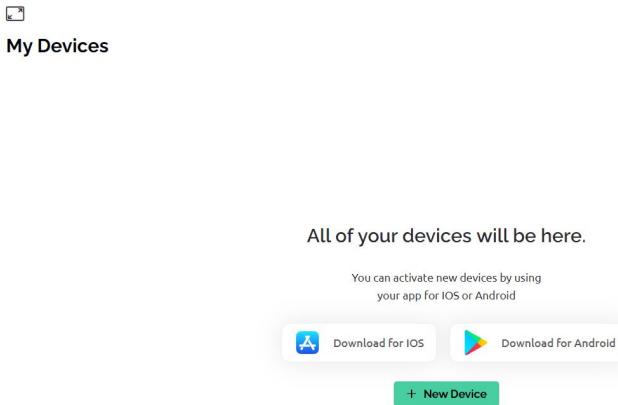
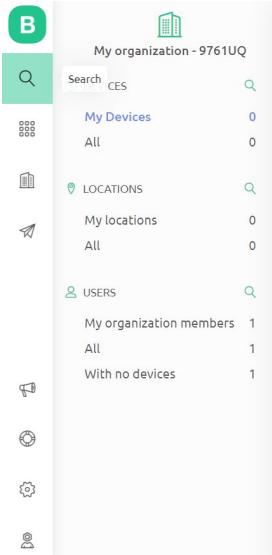
Click on datastream and choose LED(V0) and Turn on Show on/off labels option. Put name for labels.

Click on save button.

Click on Save button on upper right corner.



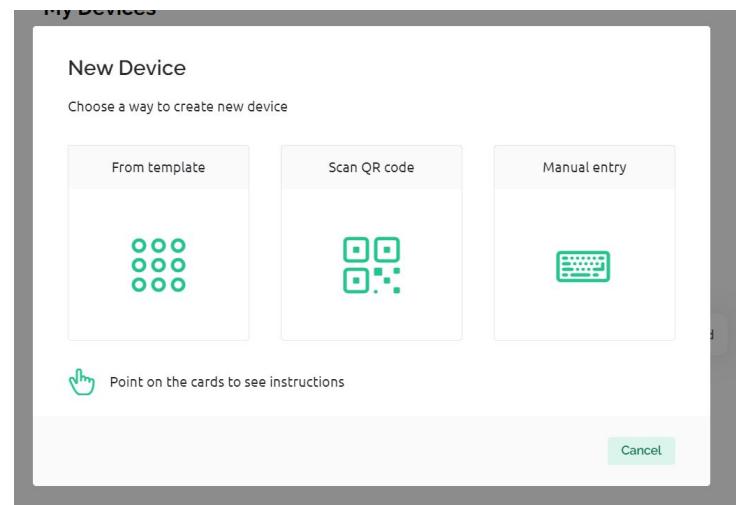
By clicking on search  option,



Click on new Device

+ New Device

New window will open click on from template.

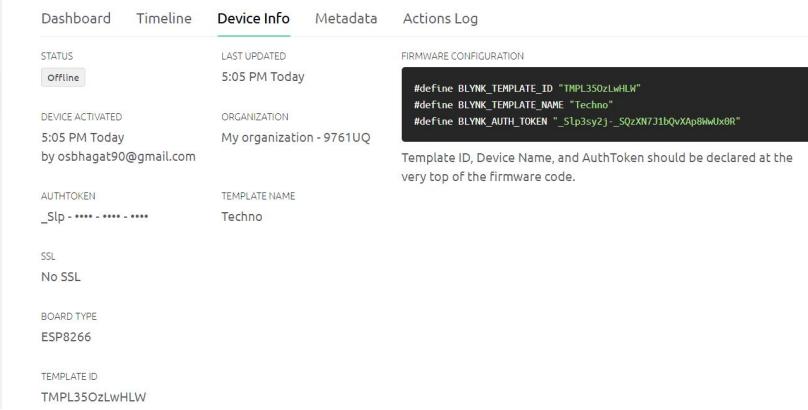
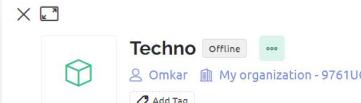
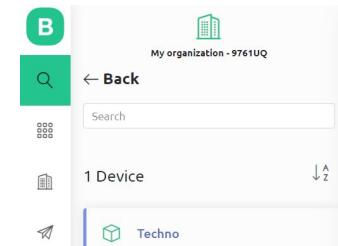
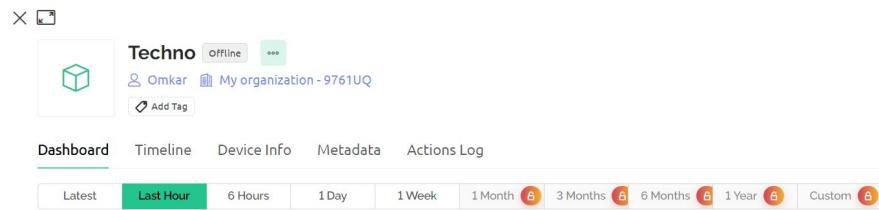
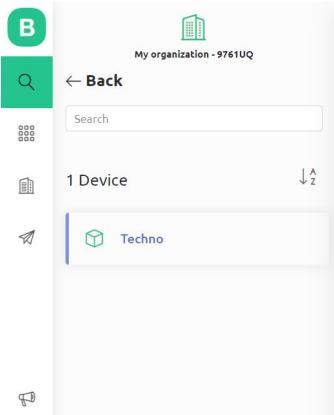


Previously created template will be shown in template option, click on that and click on create.

The screenshot shows the 'My Devices' section of a mobile application. On the left, there's a vertical navigation bar with icons for 'B' (blue), a camera, a search (magnifying glass), a grid, a document, and a signal. The main area is titled 'My Devices' with a subtitle '1 Device'. A table lists one device: 'Techno' with an auth token starting with '_Slp3sy2j_...'. The status is 'Offline' and it was last reported at '5:05 PM Today'. To the right of the table is a green button labeled '+ New Device'. A modal window titled 'New Device' is open, prompting the user to 'Create new device by filling in the form below'. It contains a 'TEMPLATE' dropdown set to 'Techno', a 'DEVICE NAME' input field containing 'Techno', and two buttons: 'Cancel' and '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

The screenshot shows the Arduino IDE interface with the following code:

```
#include <Blynk.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char ssid[] = "YourNetwork";
char pass[] = "YourPassword";

BlynkTimer timer;

BLYNK_WRITE(v0)
```

The code includes the Blynk library, WiFi library, and BlynkSimpleEsp8266 library. It defines a WiFi connection with the SSID and password, and initializes a BlynkTimer.

"TMP*****"
"Templat
"*****"

Search for Blynk and Install first library.

The screenshot shows the Arduino Library Manager with the search term "Blynk" entered. The results page displays the following libraries:

- Blynk**
by Volodymyr Shymansky Version 1.3.0 INSTALLED
Build a smartphone app for your project in minutes! It supports WiFi, BLE, Bluetooth, Ethernet, GSM, USB, Serial. Works with many boards like ESP8266, ESP32, Arduino UNO, Nano, Due, Mega, Zero, MKR100, Yun, Raspberry Pi, Particle, Energia, ARM mbed, Intel Edison/Galileo/Joule, BBC micro:bit, DFRobot, RedBearLab, Microduino, Linkit ONE ...
[More info](#)
- Blynk For Chinese**
by hznpuper qiuqiongtao@163.com
Build a smartphone app for your project in minutes! 利用Blynk平台，可以快速搭建物联网应用。
[More info](#)
- Blynk_Async_ESP32_BT_WF**
by Khoi Hoang
Simple WiFiManager for Blynk and ESP32 with or without SSL, configuration data saved in either SPIFFS or EEPROM. Enable inclusion of both ESP32 Blynk BT/BLE and WiFi libraries. Then select one at reboot or run both. Eliminate hardcoding your WiFi and Blynk credentials and configuration data saved in either LittleFS, SPIFFS or EEPROM. Using AsyncWebServer instead of WebServer, with WiFi networks scanning for selection in Configuration Portal. By design, Blynk user can run ESP32 boards with either WiFi or BT/BLE by using different sketches, and have to upload / update firmware to change. This library enables user to include both Blynk BT / BLE and WiFi libraries in one sketch, run both WiFi and BT/BLE simultaneously, or select one to use at runtime after reboot. This library also supports (auto)connection to MultiWiFi and MultiBlynk, dynamic custom as well as static parameters in Config Portal. Eliminate hardcoding your WiFi and Blynk credentials and configuration data saved in either LittleFS, SPIFFS or EEPROM. Optional default Credentials to be autoloaded into Config Portal to use or change

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 from button created on dashboard. We write code inside this function.

```
BLYNK_WRITE(V0)
{
    int value = param.toInt();
    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

The screenshot shows the Blynk Dashboard interface. On the left, there's a sidebar with icons for Back, Search, and various settings. The main area shows 'My organization - 9761UQ'. A list of devices shows '1 Device' named 'Techno'. The 'Device Info' tab is selected, displaying the following details:

STATUS	LAST UPDATED	FIRMWARE CONFIGURATION
Online	5:05 PM Today	<pre>#define BLYNK_TEMPLATE_ID "TMPL350zLwHLW" #define BLYNK_TEMPLATE_NAME "Techno" #define BLYNK_AUTH_TOKEN "Stp3sy2j...S0zXN7J1bQvXApx8WwUx8R"</pre>
DEVICE ACTIVATED 5:05 PM Today by osbhagat90@gmail.com	ORGANIZATION My organization - 9761UQ	Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.
AUTHTOKEN _Slp - * - * - *	TEMPLATE NAME Techno	
SSL No SSL		
BOARD TYPE ESP8266		
TEMPLATE ID TMPL350zLwHLW		

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
```

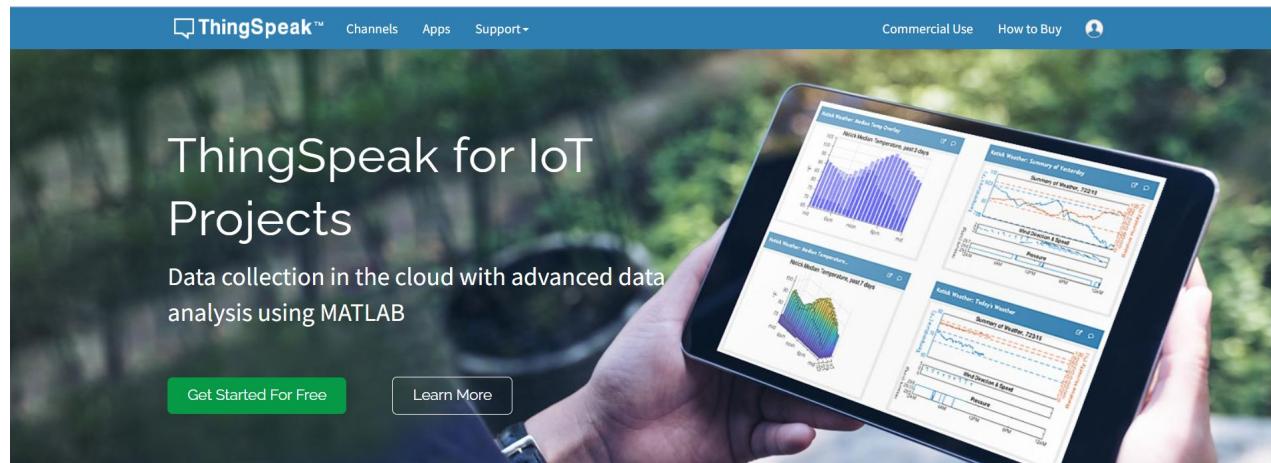
Now you can control on board of NodeMCU using switch created on Blynk Dashboard

The image displays two screenshots of the Blynk mobile application interface. The left screenshot shows the organization dashboard, which includes a search bar, a list of devices (1 Device), and a specific device card for 'Techno'. The right screenshot shows the detailed view for the 'Techno' device, featuring a timeline with various time intervals (Latest, Last Hour, 6 Hours, 1 Day, 1 Week, 1 Month, 3 Months, 6 Months, 1 Year) and a control switch labeled 'LED' with the status 'ON'.

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

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





To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

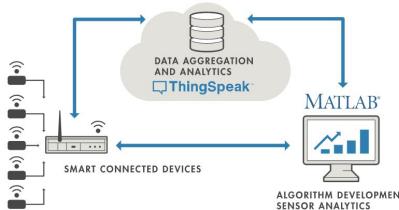
Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the [paid license options](#) for commercial, academic, home and student usage.



Email

No account? Create one!
By signing in, you agree to our [privacy policy](#).

[Next](#)

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our [Privacy Policy](#) to learn more about cookies and how to change your settings.

If you already have an account then enter your email

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

ThingSpeak™ Channels Apps Support

Commercial Use How to Buy

Create MathWorks Account

Email Address Missing required information

To access your organization's MATLAB license, use your school or work email.

Location

First Name

Last Name

[Continue](#) [Cancel](#)

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

The diagram is identical to the one in the top-left section, illustrating the architecture of the ThingSpeak-MATLAB integration for data aggregation, analytics, and algorithm development.

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

The screenshot shows the ThingSpeak interface. At the top, there's a navigation bar with the logo, 'Channels', 'Apps', 'Devices', and 'Support'. Below the navigation bar, the title 'My Channels' is displayed. A green button labeled 'New Channel' is visible. To the right of the button is a search bar with the placeholder 'Search by tag' and a magnifying glass icon.

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.

The screenshot shows the 'New Channel' configuration page. At the top, the title 'New Channel' is displayed. Below the title, there are input fields for 'Name' (containing 'IOT learning') and 'Description'. Under the 'Fields' section, there are eight rows, each labeled 'Field 1' through 'Field 8'. Each row contains a text input field and a checkbox. The first checkbox, next to 'PoT value', is checked. The other seven checkboxes are empty. At the bottom, there is a 'Metadata' input field.

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

Save Channel

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

ThingSpeak™

Commercial Use How to Buy OB

IOT learning

Channel ID: 2242413
Author: mwa0000021993650
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Add Visualizations Add Widgets Export recent data

MATLAB Analysis MATLAB Visualization

Channel Stats

Created: less than a minute ago
Entries: 0

Field 1 Chart

IOT learning

PoT value

ThingSpeak™

Commercial Use How to Buy OB

IOT learning

Channel ID: 2242413
Author: mwa0000021993650
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Write API Key

Key: KUWF31MG34HC5TJQ

Generate New Write API Key

Read API Keys

Key: F6LF65NQ06UFPRZ5

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key: Use this key to write data to a channel. If you feel your key has been compromised, click Generate New Write API Key.
- Read API Keys: Use this key to allow other people to view your private channel feeds and charts. Click Generate New Read API Key to generate an additional read key for the channel.
- Note: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

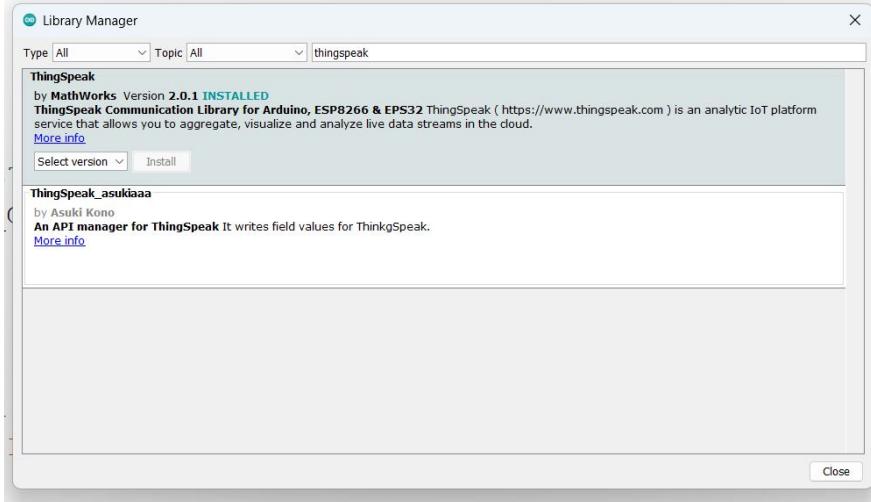
API Requests

Write a Channel Feed

GET https://api.thingspeak.com/update?api_key=KUWF31MG34HC5TJQ&Field

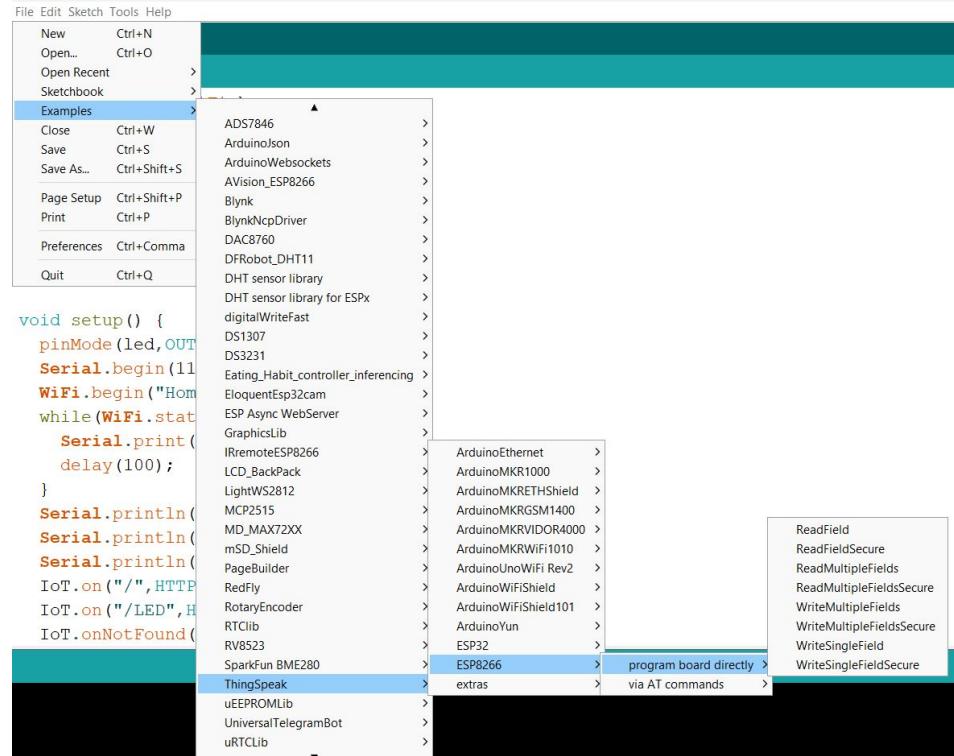
By clicking on API Keys tab you will get all the Write and Read API Keys.

Install ThingSpeak library from library manager in arduino.



Now open Example.

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



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.



The screenshot shows the Arduino IDE interface with the title bar "WriteSingleField - secrets.h | Arduino 1.8.19". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for back, forward, search, and file operations. The main workspace shows the code for the Secrets.h file:

```
#define SECRET_SSID "Home"      // replace MySSID with your WiFi network name
#define SECRET_PASS "12345678"    // replace MyPassword with your WiFi password

#define SECRET_CH_ID 2242413     // replace 0000000 with your channel number
#define SECRET_WRITE_APIKEY "KUWF3IMG34HC5TJQ" // replace XYZ with your channel write API Key
```

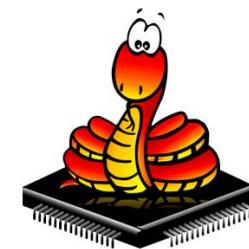
WriteSingleField is uploading number 0 to 99 to respective field.

By changing the contain of number variable you can upload your desired data.

MicroPython with ESP8266



MicroPython



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.

ThingSpeak™

Channels ▾ Apps ▾ Devices ▾ Support ▾

Private View Public View Channel Settings Sharing API Keys Data

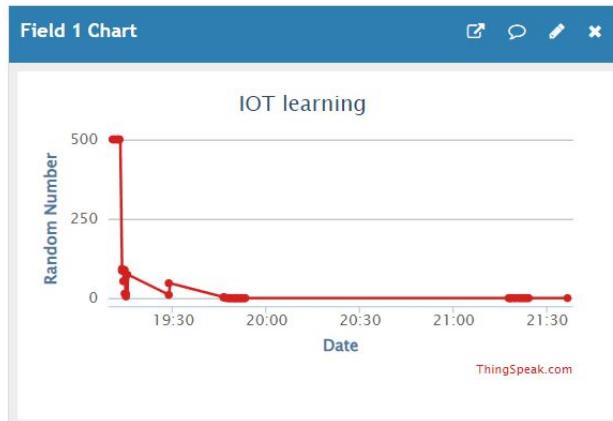
+ Add Visualizations + Add Widgets Export recent data

Channel Stats

Created: 9.months.ago

Last entry: 9.months.ago

Entries: 104



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/1KObWKTukzCoSSxEyqXEoa2hdzbIC0-BA/view?usp=drive_link

MicroPython downloads

MicroPython is developed using git for source code management, and the master repository can be found on GitHub at github.com/micropython/micropython.

The full source-code distribution of the latest version is available for download here:

- [micropython-1.20.0.tar.xz](#) (73MB)
- [micropython-1.20.0.zip](#) (151MB)

Daily snapshots of the GitHub repository (not including submodules) are available from this server:

- [micropython-master.zip](#)
- [pyboard-master.zip](#)

Firmware for various microcontroller ports and boards are built automatically on a daily basis and can be found below.

Filter by:

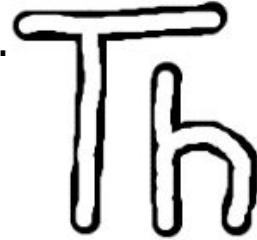
Port: cc3200, esp32, esp8266, mimxrt, nrf, renesas-ra, rp2, samd, stm32

Feature: Audio Codec, BLE, Battery Charging, CAN, Camera, DAC, Display, Dual-core, Environment Sensor, Ethernet, External Flash, External RAM, Feather, IMU, JST-PH, JST-SH, LoRa, Microphone, PoE, RGB LED, SDCard, Secure Element, USB, USB-C, WiFi, microSD, mikroBUS

Vendor: Actinius, Adafruit, Arduino, BBC, Espressif, Espruino, Fez, George Robotics, HydraBus, I-SYST, LEGO, LILYGO, Laird Connectivity, LimiFrog, M5 Stack, Makery, McHobby, Microchip, MikroElektronika, MiniFig Boards, NXP, Netduino, Nordic Semiconductor, OLIMEX, PJRC, Particle, Pimoroni, Pycom, Raspberry Pi, Renesas Electronics, ST Microelectronics, Seeed Studio, Silicognition, Sparkfun, Unexpected Maker, VCC-GND Studio, Vekatech, WeAct, Wemos, Wireless-Tag, Wiznet, nullbits, u-blox

MCU: cc3200, esp32, esp32c3, esp32s2, esp32s3, esp8266, mimxrt, nrf51, nrf52, nrf91, ra4m1, ra4w1, ra6m1, ra6m2, ra6m5, rp2040, samd21, samd51, stm32f0, stm32f4, stm32f7, stm32g0, stm32g4, stm32h7, stm32l0, stm32l1,

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.

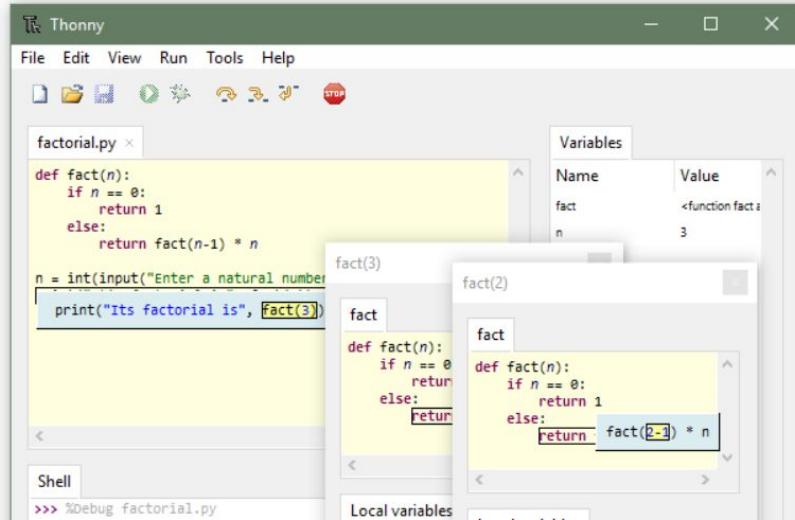
By clicking appropriate OS the respective IDE will be downloaded.

Thonny
Python IDE for beginners

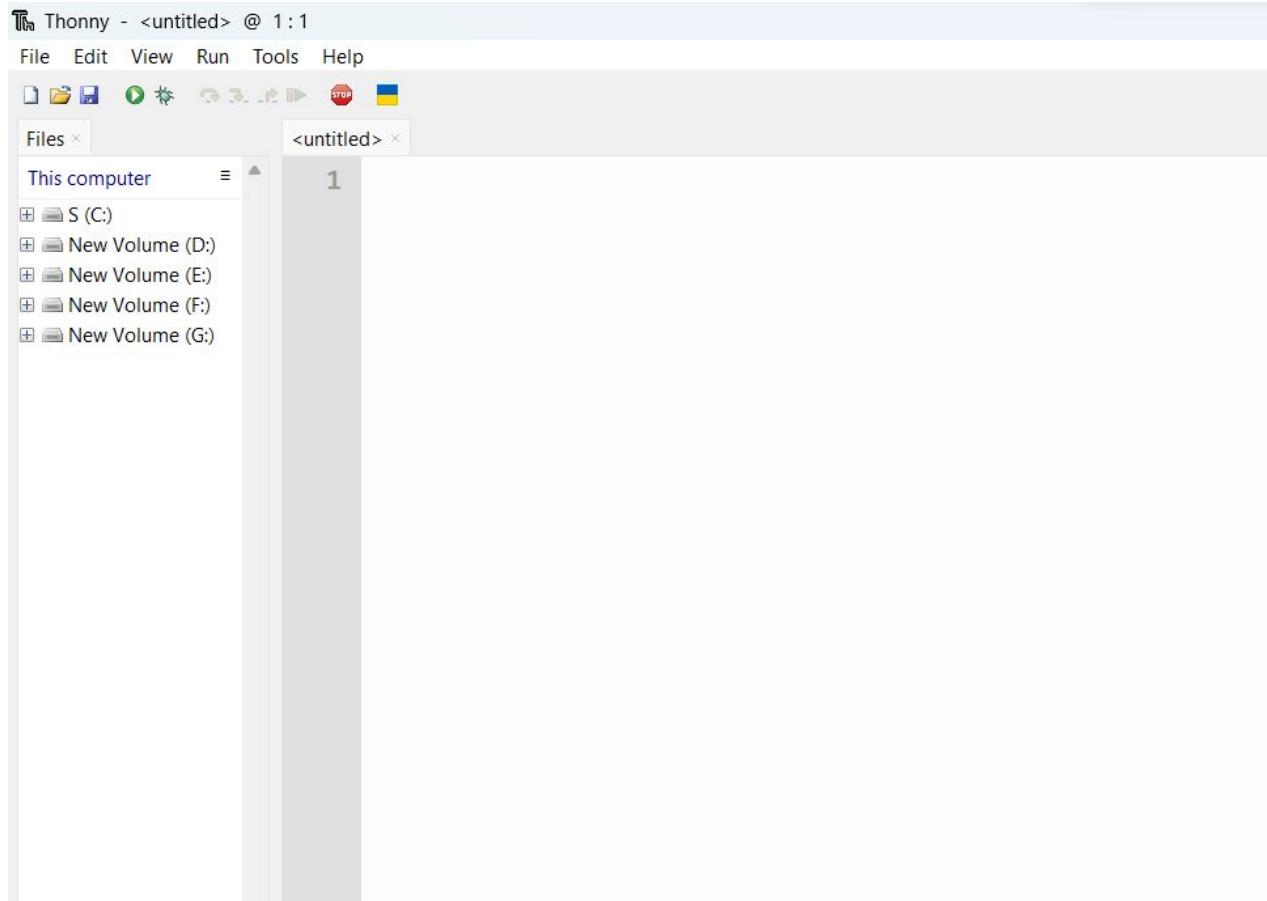


Download version [4.1.2](#) for
Windows • Mac • Linux

After Downloading install the Thonny IDE.

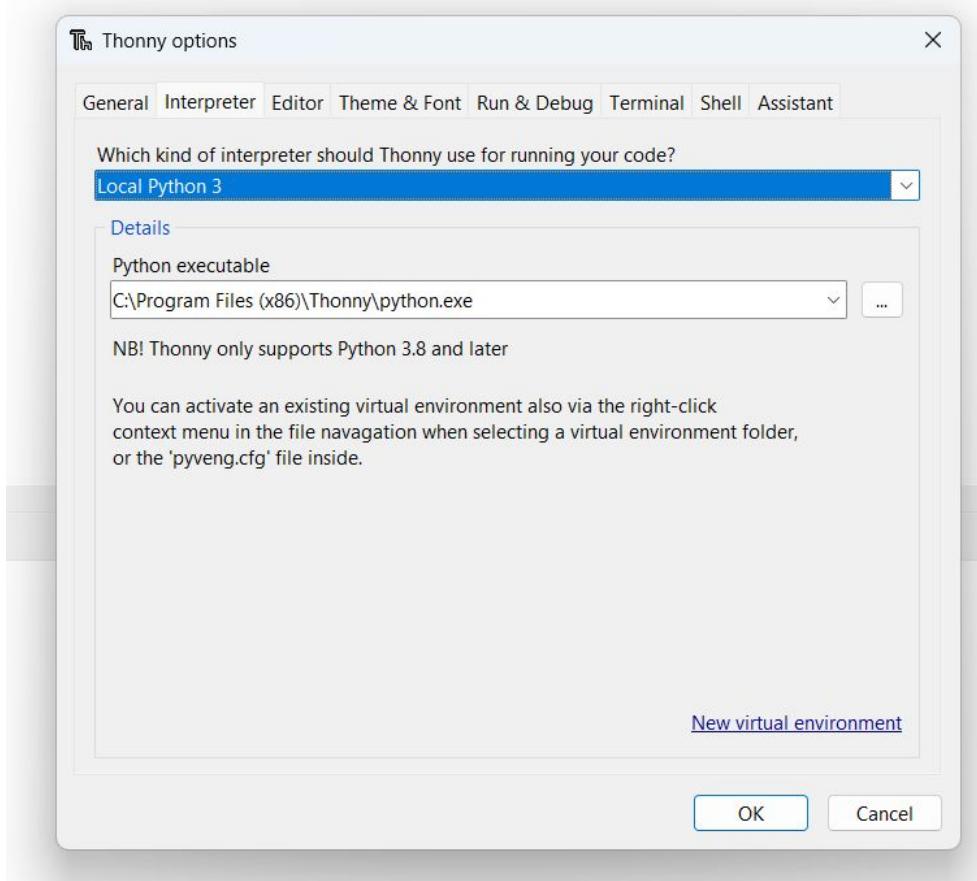


Opening Thonny IDE you will see simple interface.



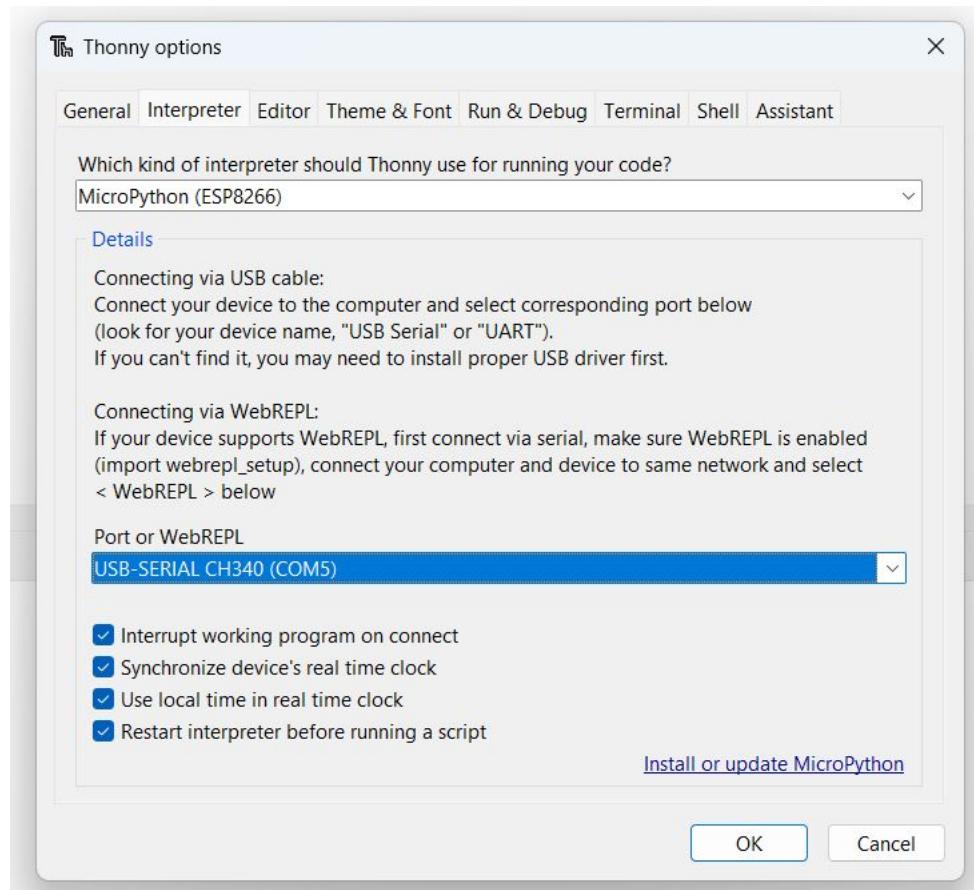
To make thonny IDE compatible for MicroPython on ESP8266 we need to make some settings.

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.

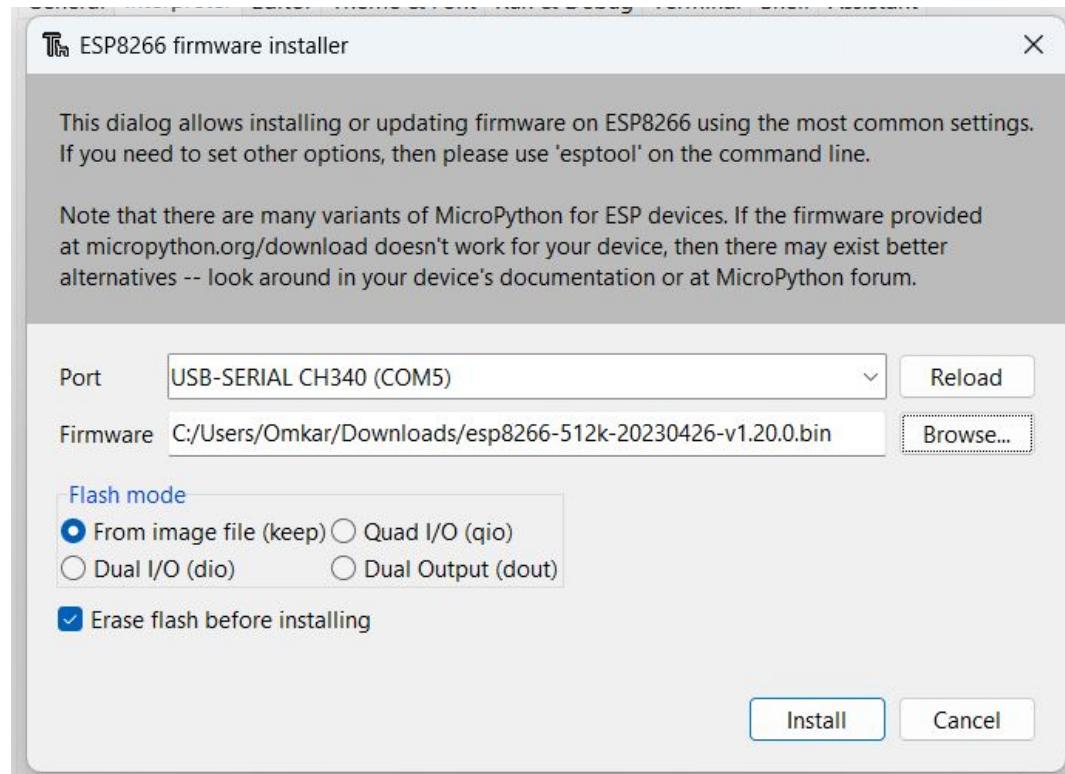
New Window will be open named ESP8266 firmware installer.

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.

The screenshot shows the Thonny IDE interface. The top menu bar includes File, Edit, View, Run, Tools, and Help. The title bar indicates "Thonny - <untitled> @ 1:15". The left sidebar has sections for "Files" (listing "This computer" and "MicroPython device" with "boot.py") and "Shell". The main area contains a code editor with the following content:

```
1 print("Omkar")
```

Below the code editor is the "Shell" pane, which displays the output of the executed code:

```
>>> %Run -c $EDITOR_CONTENT
Omkar
>>>
```

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 provide 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)
```