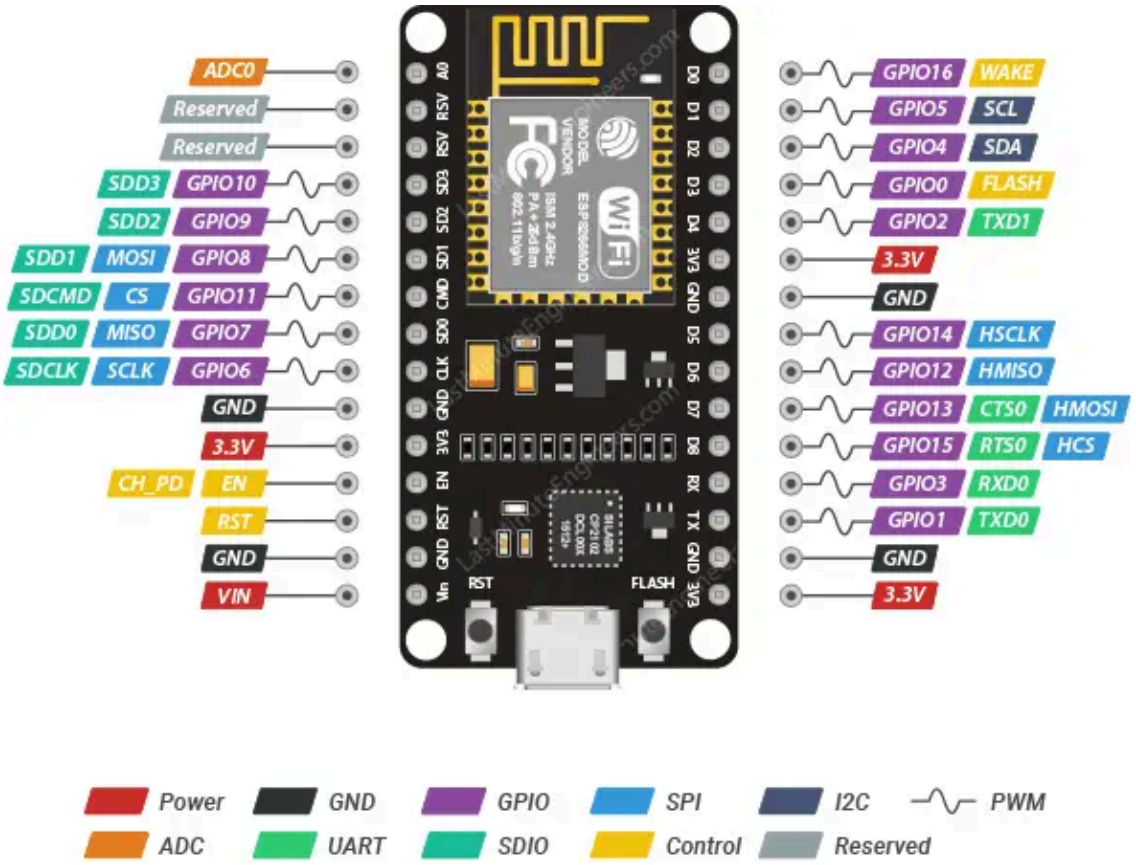
ESP 8266 NodeMCU

Pinout & features:



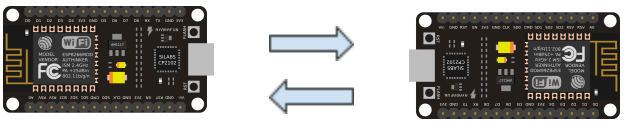
- Total 17 GPIO pins
- 1 ADC pin
- 4 dedicated PWM pins
- 1xI2C, 1xI2S, 2xUART, 2xSPI interfaces
- The system works on **3.3V** logic level. Ensure to use logic level converters / appropriate resistors while using sensors, I/O devices that work on 5V

Source : https://lastminuteengineers.com/esp8266-pinout-reference/

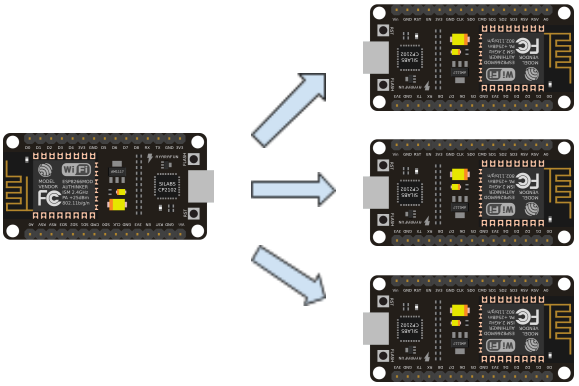
Types of communication :

1. Unidirectional (Main- Secondary)

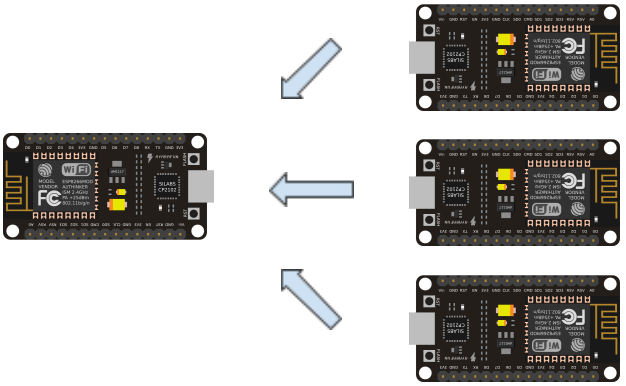
2. Bidirectional



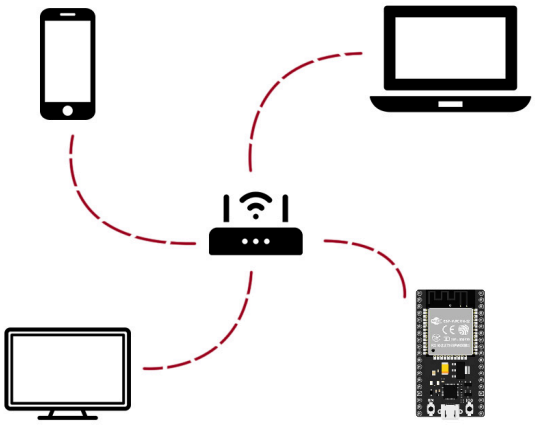
3. One to Many

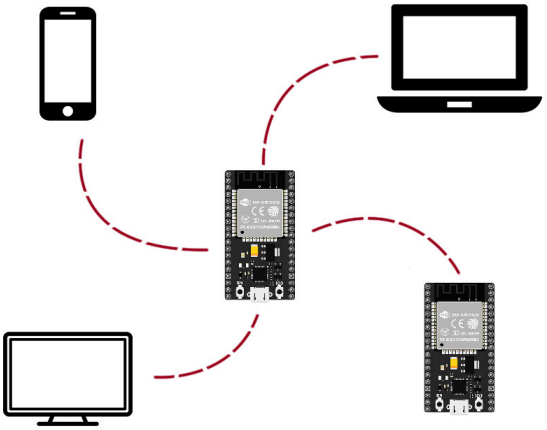


4. Many to One



5. ESP Web Server



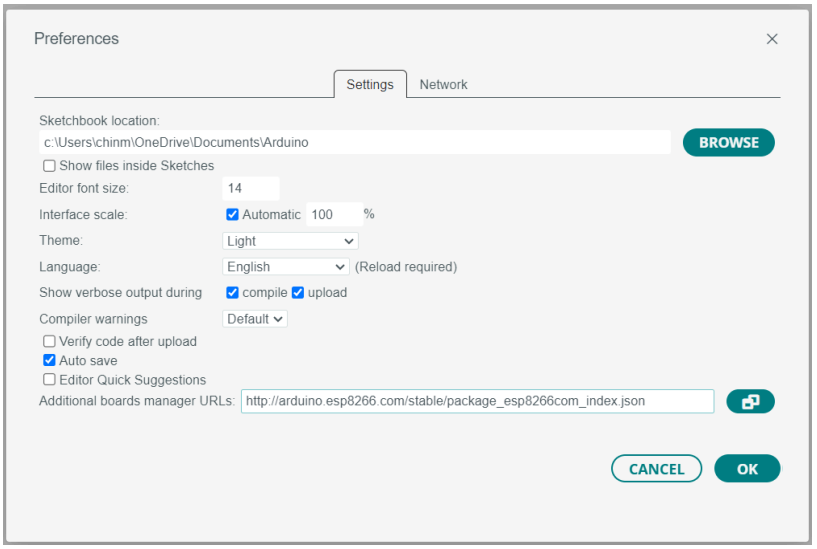


Prerequisites :

1. Install Arduino IDE from https://www.arduino.cc/en/software

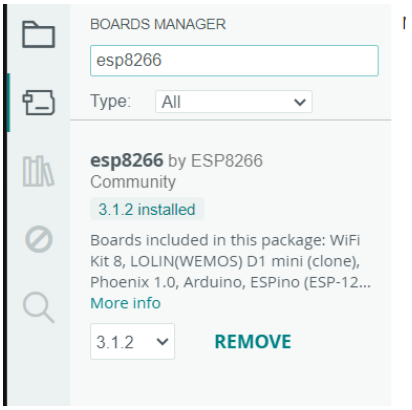
2. Go to https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads and install the drivers.

3. Open Arduino IDE. Go to file > Preference > Additional boards manager and paste the following link : https://arduino.esp8266.com/stable/package_esp8266com_index.json



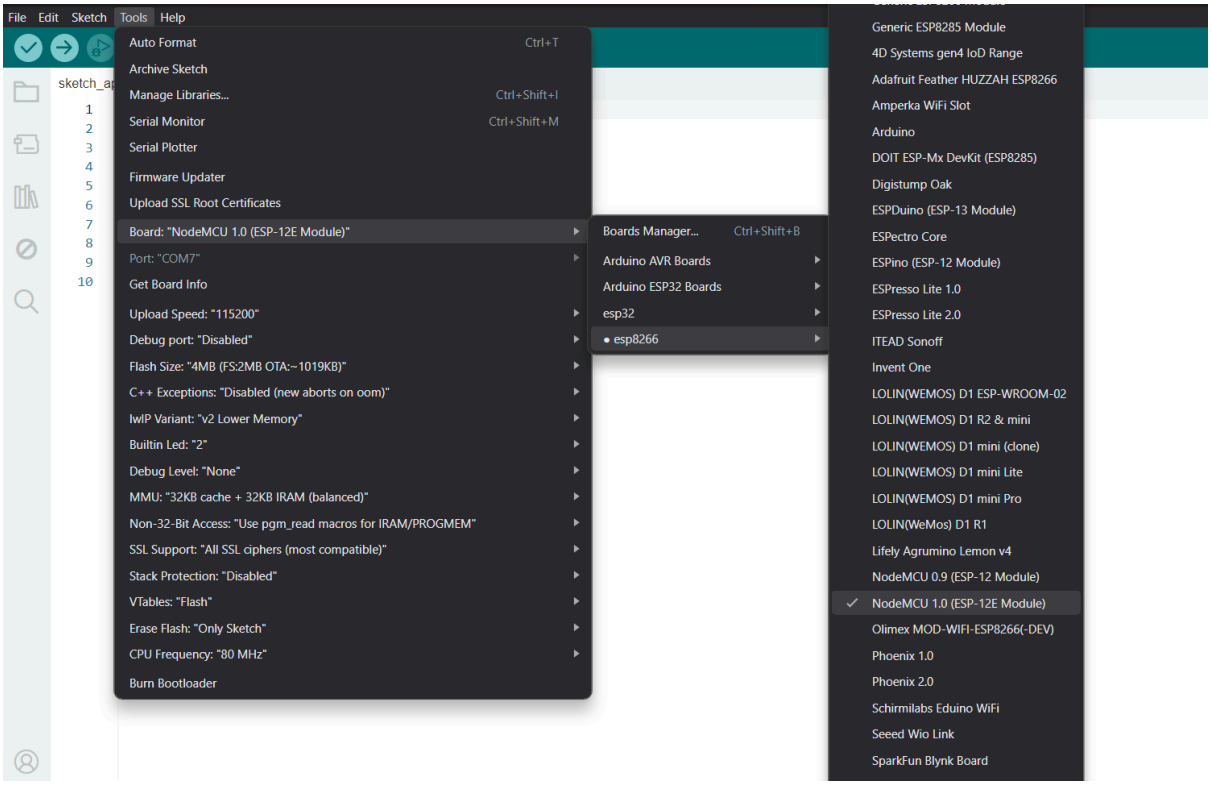
4. Click on OK

5. Go to Tools > Boards > Board Manager > type esp8266 by esp community and install it

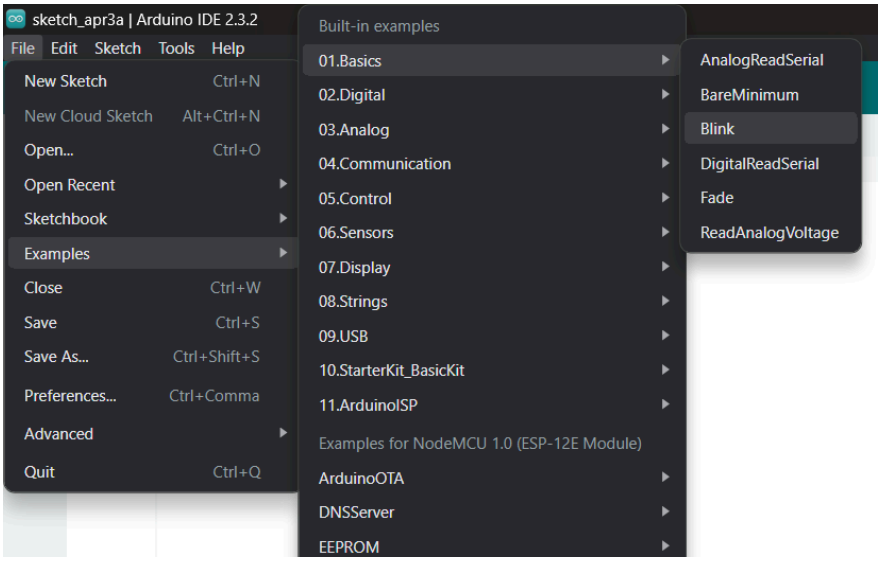


Source: https://github.com/esp8266/Arduino/?tab=readme-ov-file#installing-with-boards-manager

6. Go to Boards. Select esp8266 > NodeMCU 1.0 (ESP-12E Module)



7. Select the COM Port and upload the blink sketch.



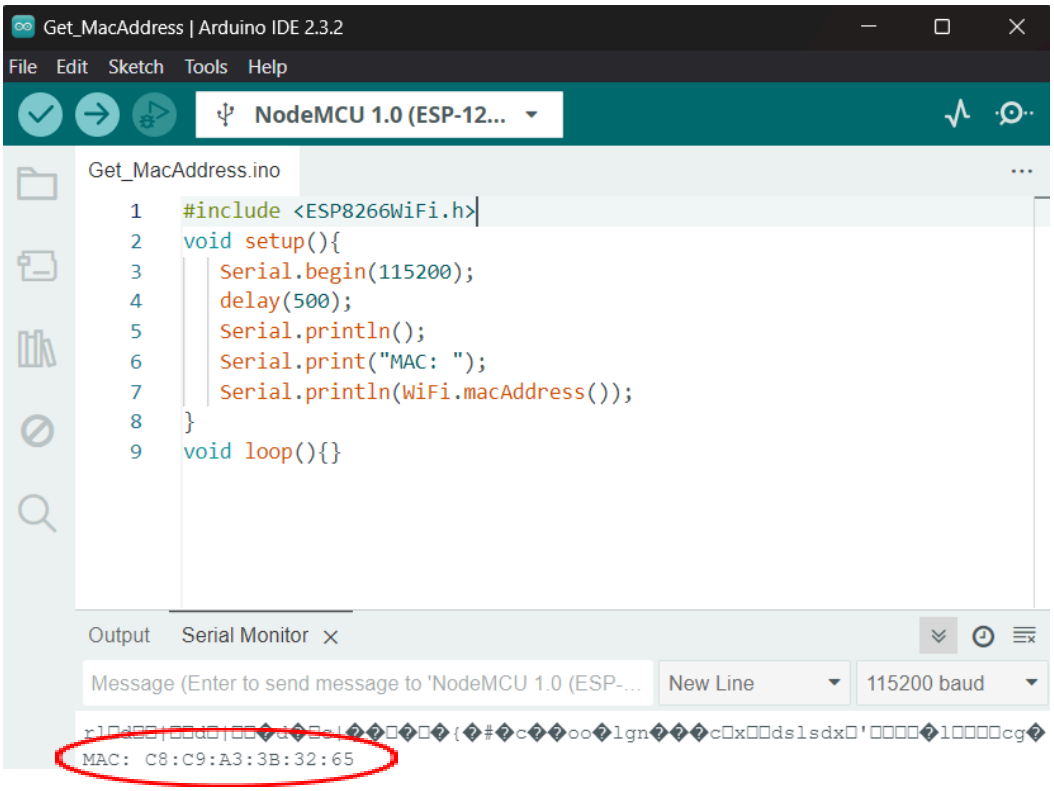
8. Once Blink is running, we move to our first program.

Getting the mac address of the device :

Each ESP device has a unique Mac Address required for any sort of wireless communication.
We use the following code in order to fetch it :

```
#include <ESP8266WiFi.h>
void setup(){
    Serial.begin(115200);
   delay(500);
   Serial.println();
   Serial.print("MAC: ");
   Serial.println(WiFi.macAddress());
 }
void loop(){}
```

Open the serial Monitor, set the baud rate to 115200 and press RST button on the ESP board if the MAC address is not visible.



It helps to add stickers of MAC address to physical boards and color code them in order to identify Main and Secondary / device nos, etc.

ESP-NOW Functions

Here's a summary of most essential ESP-NOW functions:

esp_now_init()
Initializes ESP-NOW. You must initialize Wi-Fi before initializing ESP-NOW. Returns 0, if success.

esp_now_set_self_role(role)
the role can be: ESP_NOW_ROLE_IDLE = 0,
ESP_NOW_ROLE_CONTROLLER, ESP_NOW_ROLE_SLAVE, ESP_NOW_ROLE_COMBO,
ESP_NOW_ROLE_MAX

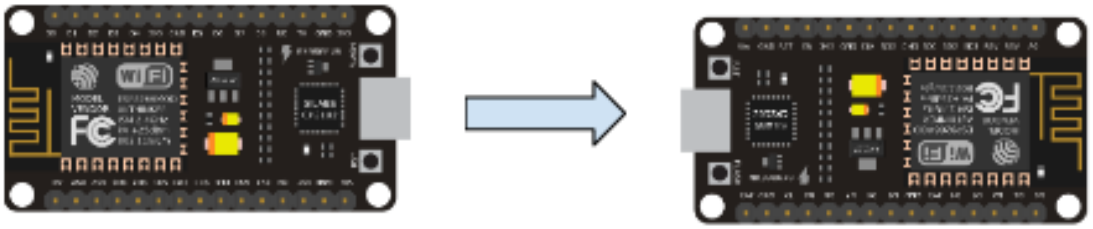
esp_now_add_peer(uint8 mac_addr, uint8 role, uint8 channel, uint8 key, uint8 key_len)
Call this function to pair a device.

esp_now_send(uint8 mac_address, uint8 data, int len)
Send data with ESP-NOW.

esp_now_register_send_cb()
Register a callback function that is triggered upon sending data. When a message is sent, a function is called – this function returns whether the delivery was successful or not.

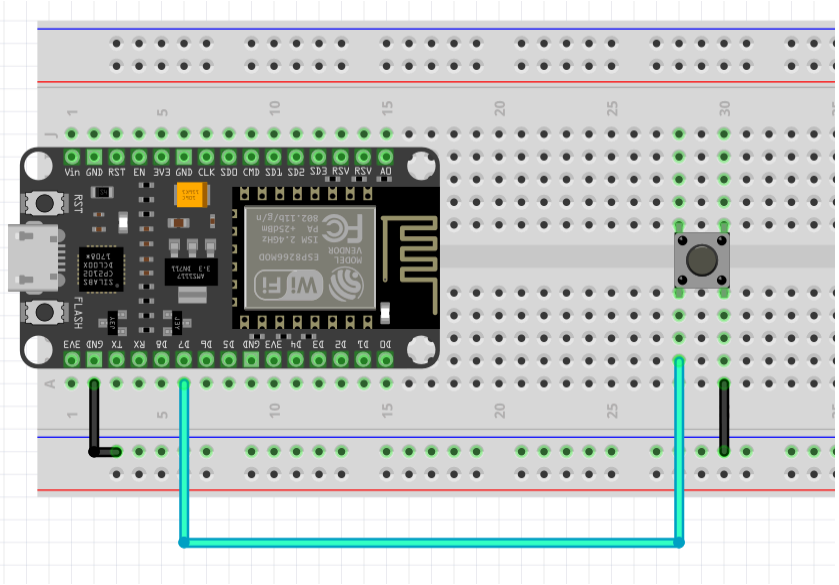
esp_now_register_rcv_cb()
Register a callback function that is triggered upon receiving data. When data is received via ESP-NOW, a function is called.

1. Unidirectional communication (Main/Secondary)



In this method, we have a main and a secondary device.
We can use the main device to send data to the secondary device to execute various operations.

Eg . Use a push button on the *Main* device to trigger the LED on the *Secondary* device.

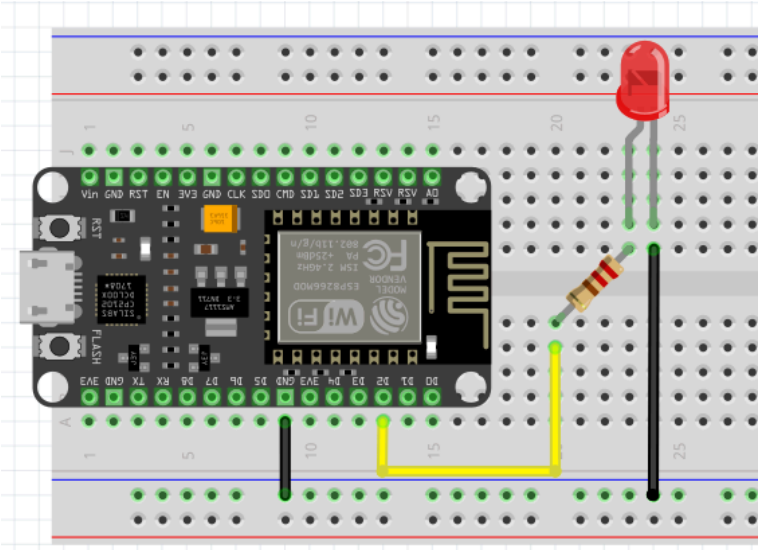


Main -

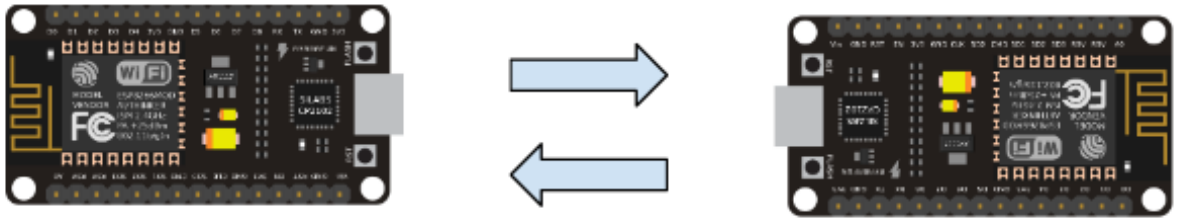
Connect the Push Button to pin D7

Secondary -

Connect the LED to pin D2 with a 120 ohm resistor

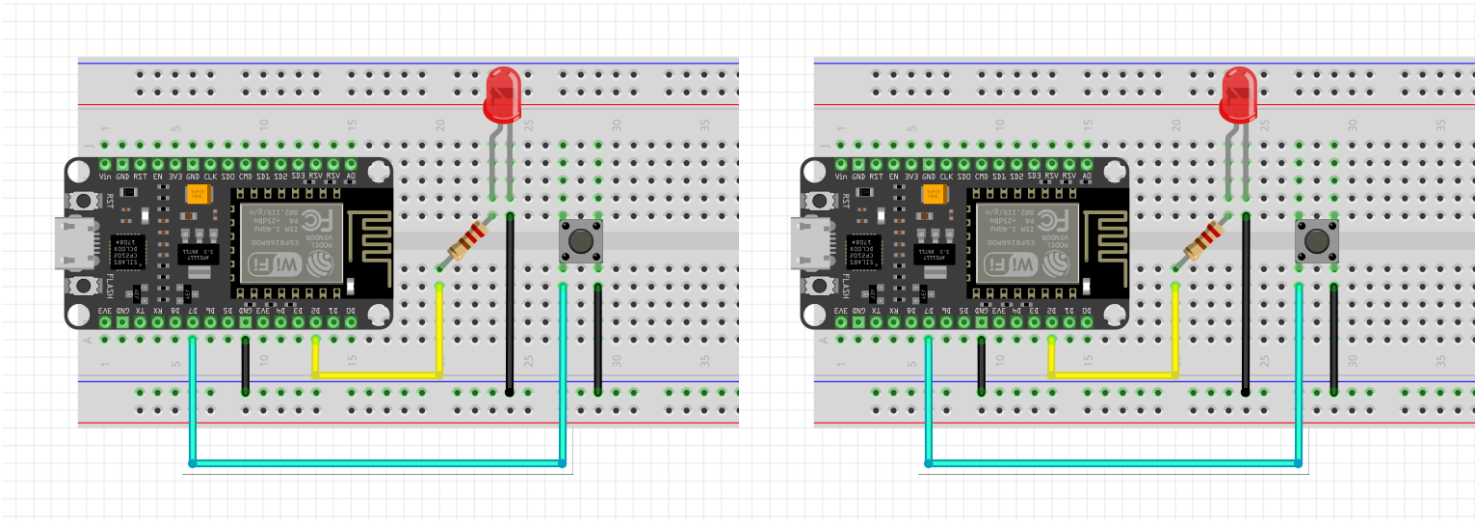


2. Bidirectional communication



In this method, we have two devices sharing data with each other i.e. both transmitting and receiving. Such devices are called “transceivers”.

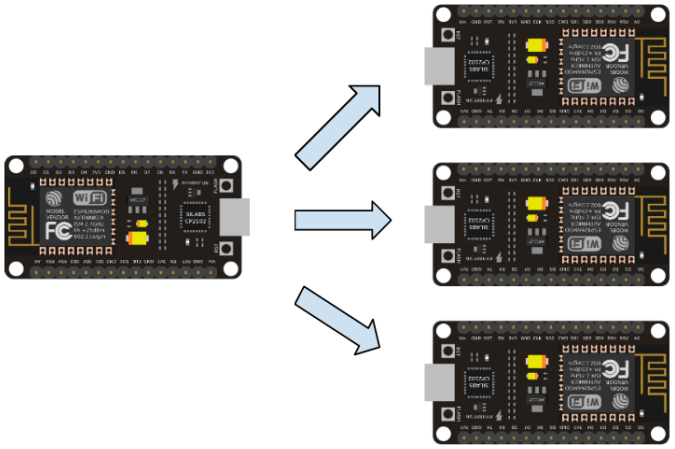
Eg. Program 2 Controllers such that when you press the push button on Controller 1, the LED connected to Controller 2 should glow and vice versa.



Controller 1 Controller 2

- D7 - Pushbutton
- D2 - LED
- Resistor - 120 ohm

3. One to Many

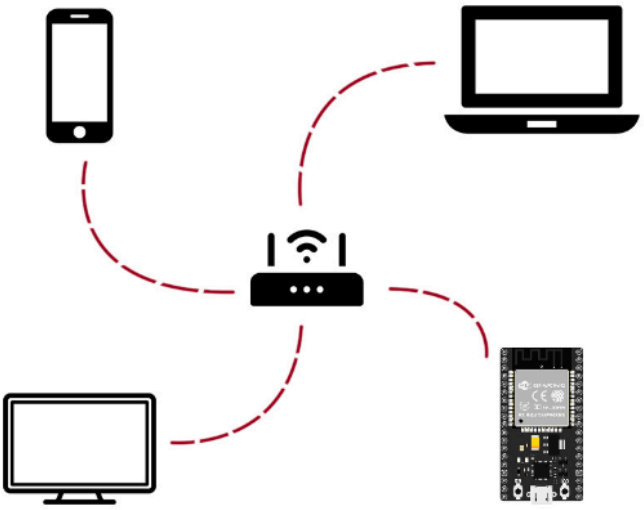


Just like the Unidirectional communication, this uses the Main- Secondary approach, except that there is one main and multiple secondary devices.

Simply replace the broadcast address line in Main's sketch to-

```
uint8_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
```

4. Web Server using Blynk.io

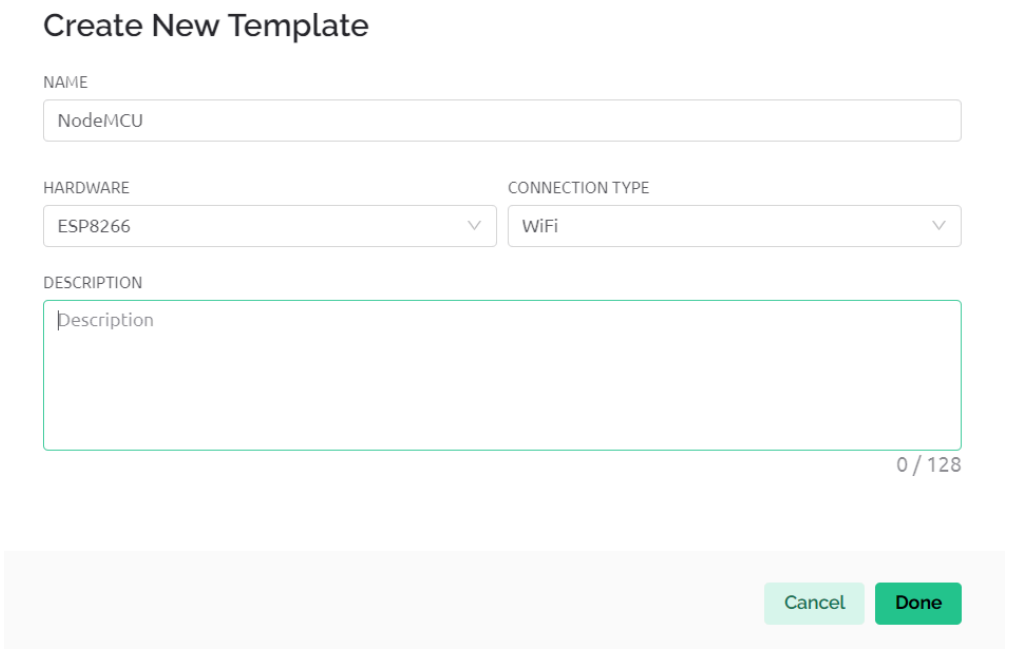


A webserver is a virtual interface to visualize and control your ESP and all of its I/Os. You can connect to the internet and use your phone / laptop to perform all operations.

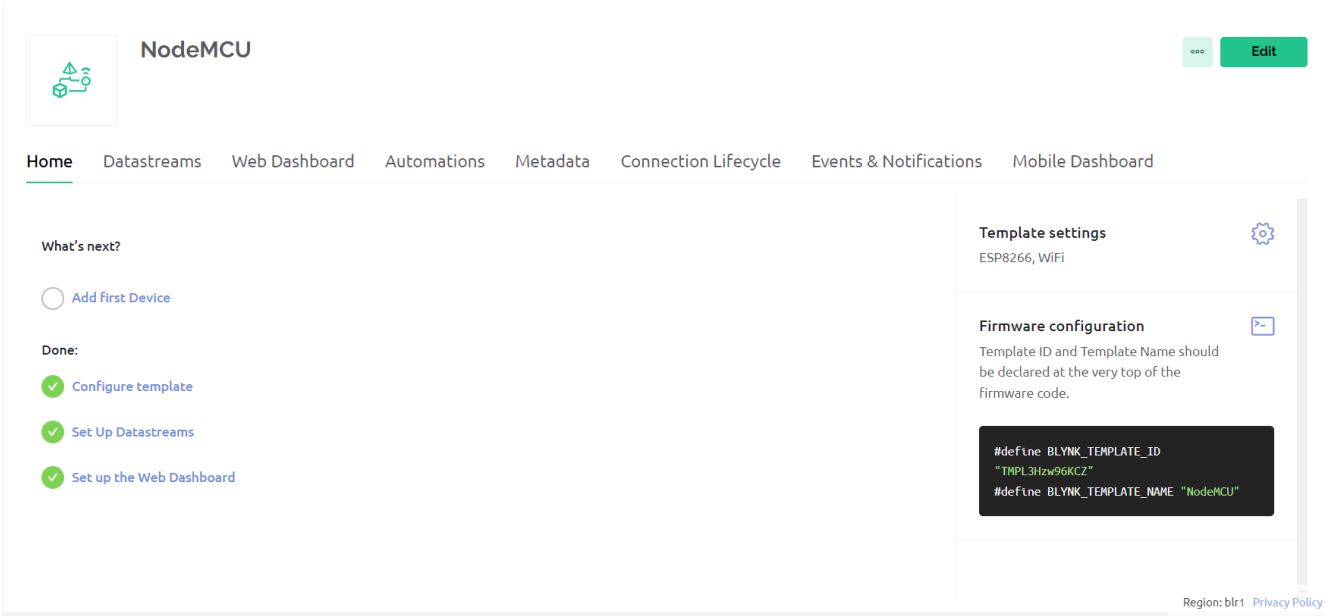
Go to blynk.io and sign up.

The page will redirect you to the dashboard interface.

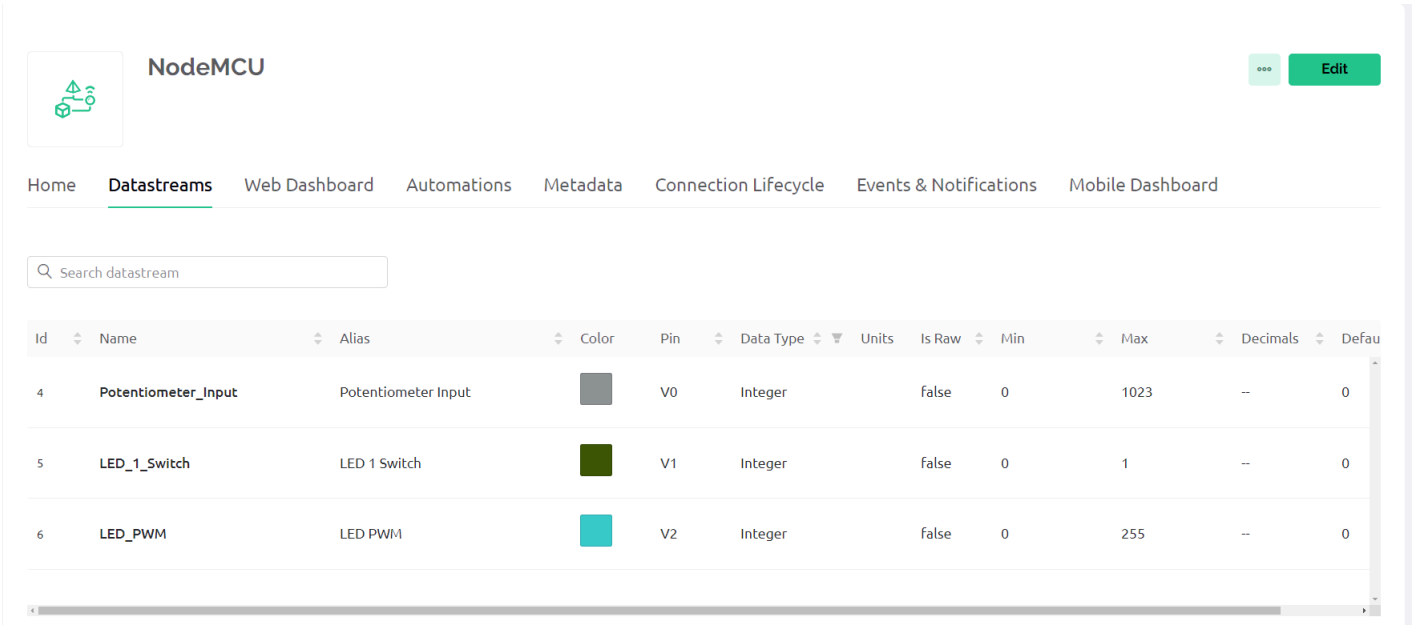
1. Go to developer zone > My templates > New Template
2. Select Esp8266 in hardware and connection type as wifi and add the description.



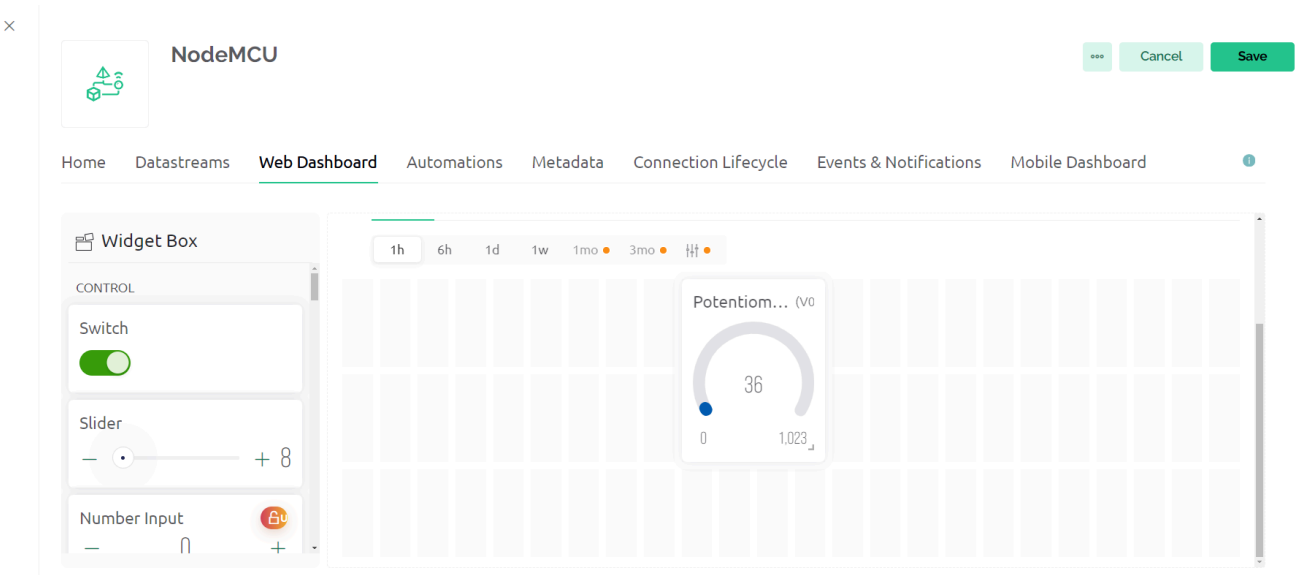
3. You'll see the following interface.



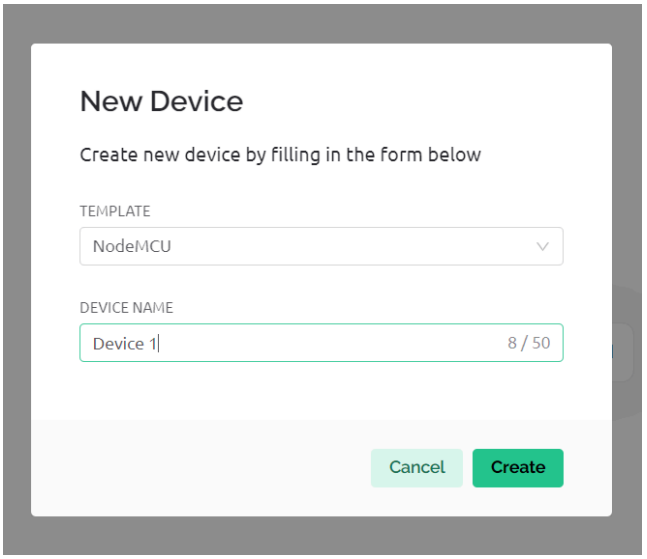
4. Go to Datastreams. Every I/O device will be defined in Datastreams. Click on edit to add/remove data streams.



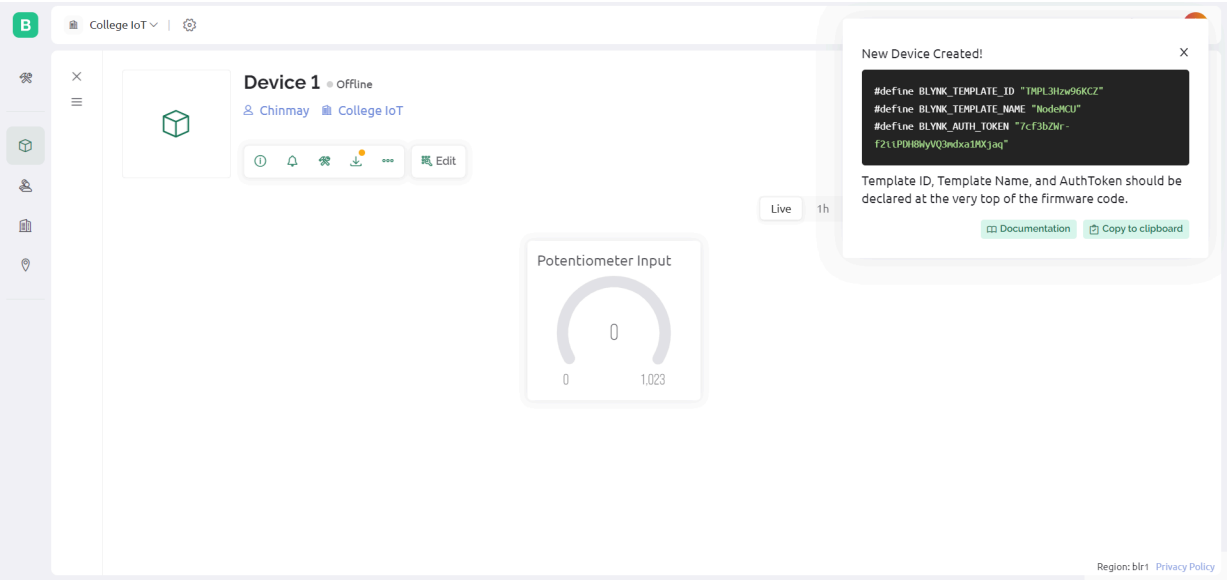
5. Go to the Web Dashboard. This is the GUI for visualizing/ controlling through a web browser. Click On edit. Go to the Widget box to add elements.



6. Go to Devices > New Device > From template. Select the template that you previously created and assign a name to your device.

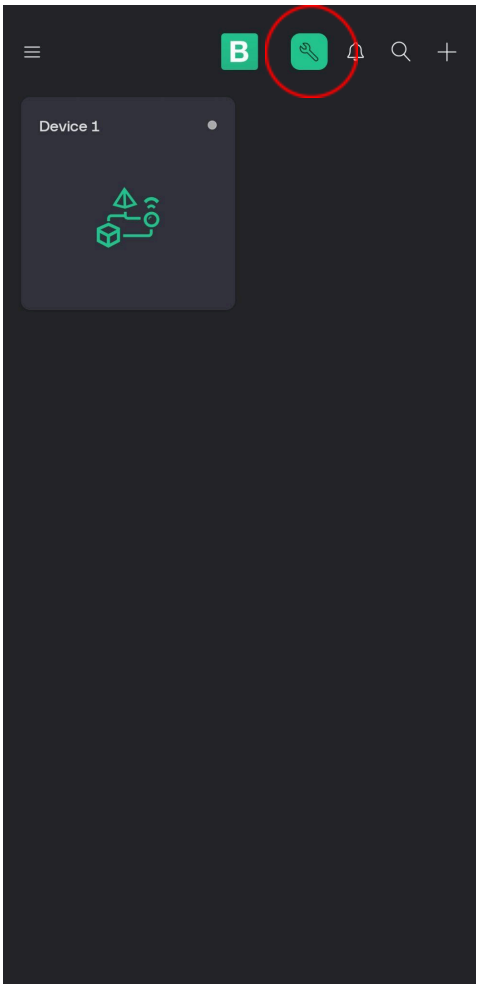


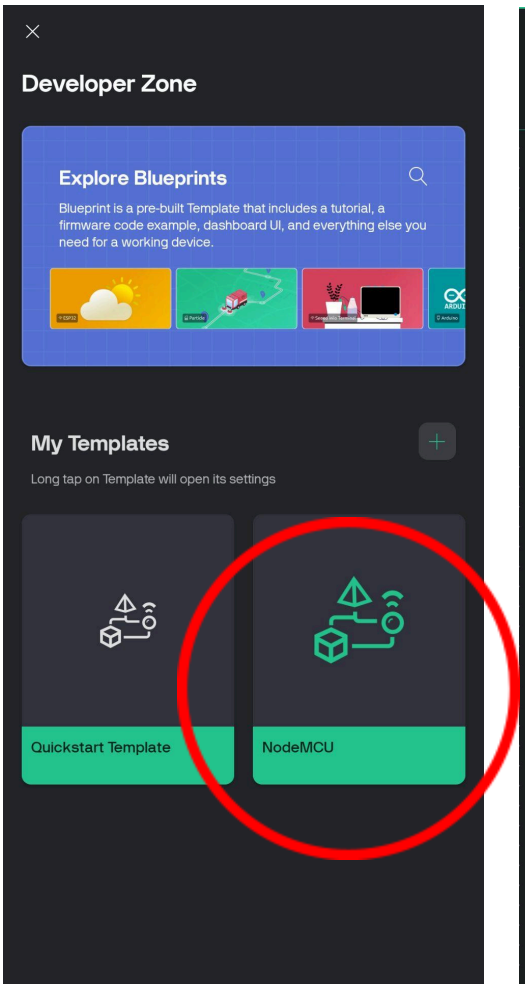
7. Once you create your device, you'll get the following screen

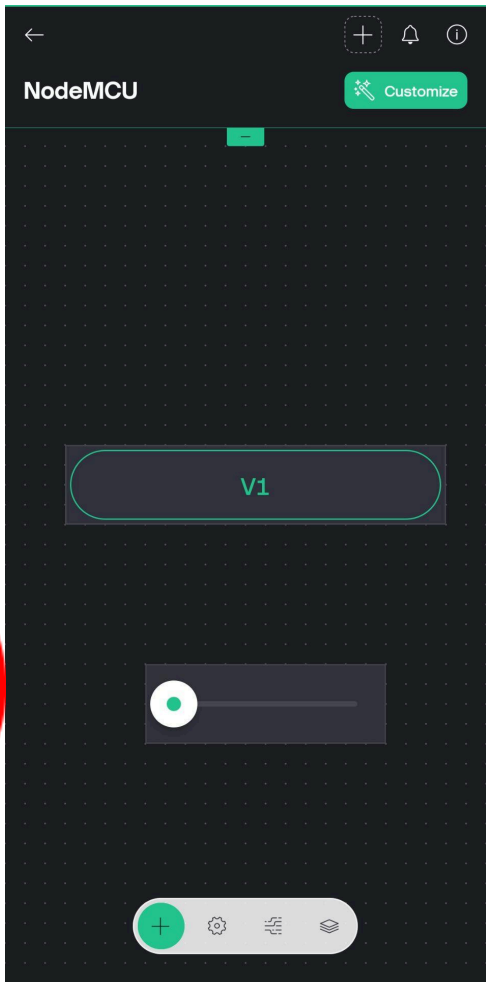


Copy the 3 parameters in your arduino sketch.

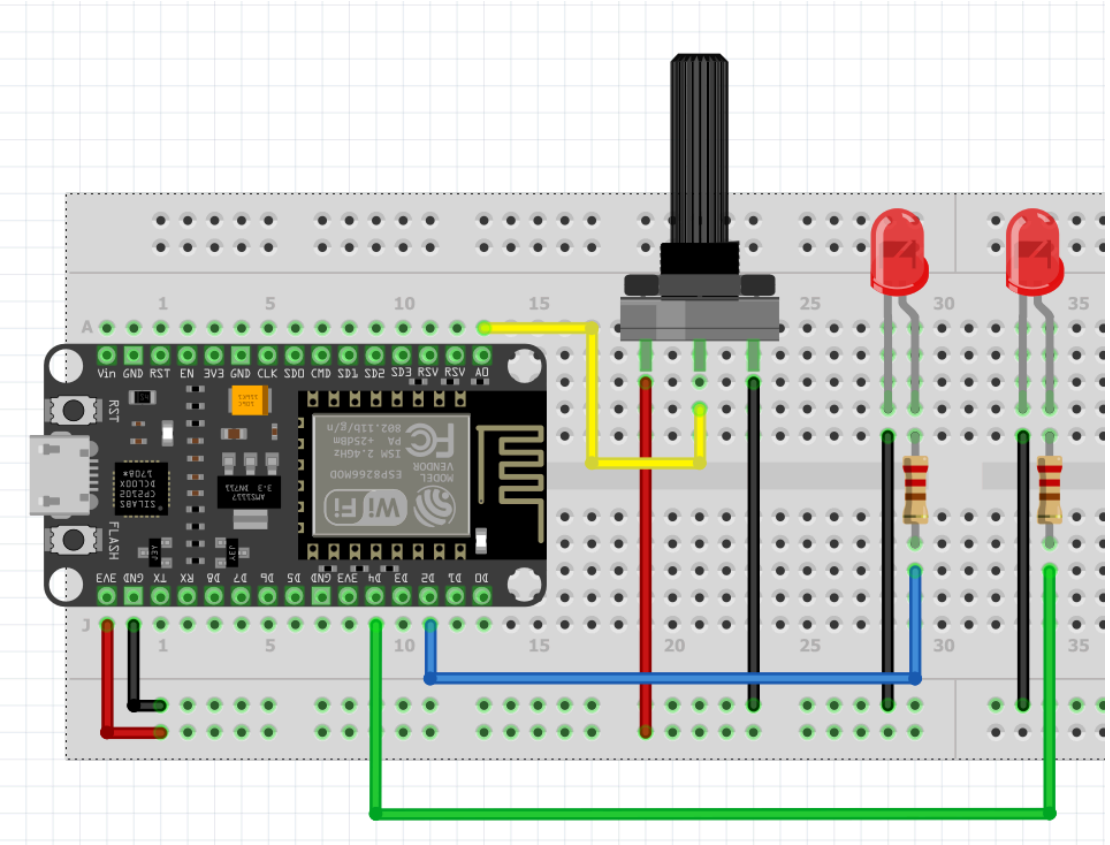
Similarly, login into the mobile app. You'll see the device added from the web. Select developer zone > My template> select the created template and add widgets. Tap on the widget and assign it the datastream.







Circuit Diagram -



- D2 - LED_Switch
- D4 - LED_PWM
- A0 - Potentiometer