



UTM
UNIVERSITI TEKNOLOGI MALAYSIA



JxG 2021 WORKSHOP 3: Fun **with BLYNK**

MODULE



Table of Contents

1 - Setting Up	3
2 - Basic of FUNCTION	9
3 - Intro to NodeMCU ESP32 Board	13
4 - BLYNK Basic Understanding	15
5 - Getting Started with BLYNK	19
5.1 Template	21
5.2 Datastream	24
5.3 Basic Programming Structures	28
6 - Hands On Session	32
6.1 Virtual Button Input to Physical LED Output	38
6.2 Physical Button Input to Virtual LED Input	44
7 - What's next? Control it from anywhere in the World!	48
7.1 RC Car Component and Connection	50
7.2 RC Car Full Programming	52



UTM
UNIVERSITI TEKNOLOGI MALAYSIA



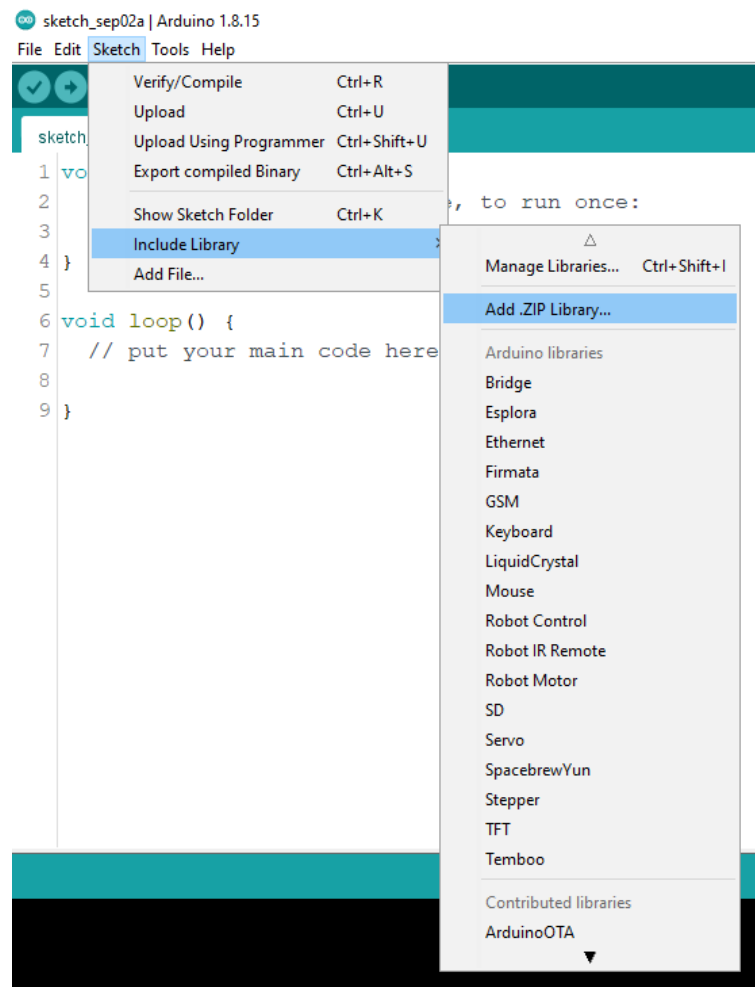
1 - Setting Up

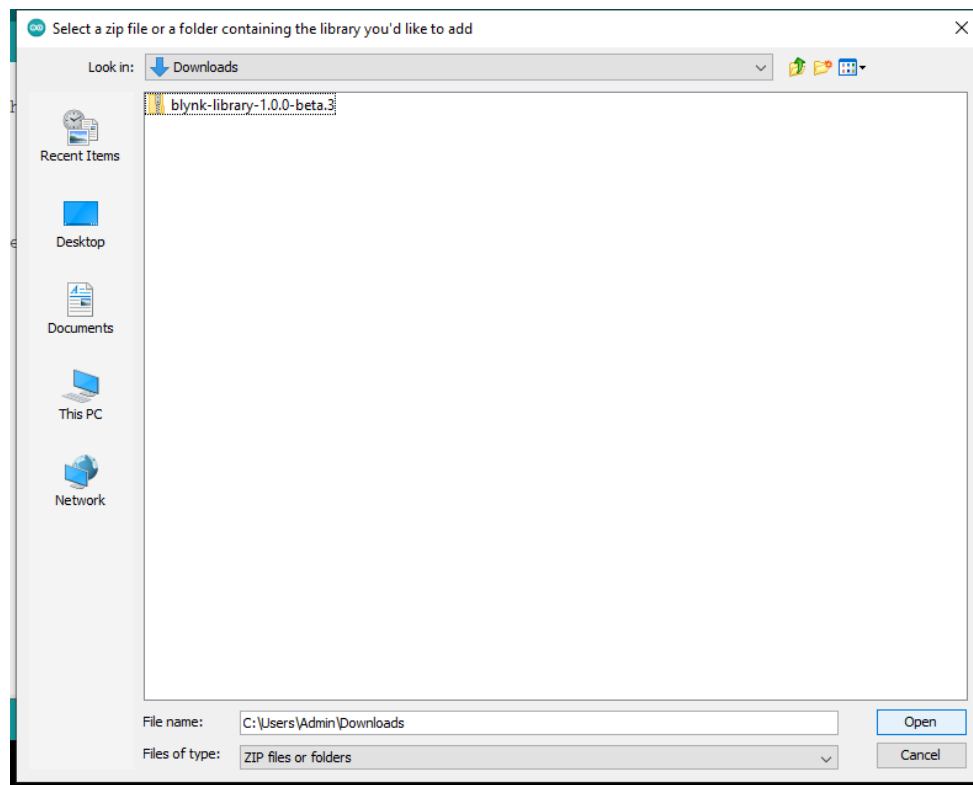
Open Arduino IDE.

Firstly, make sure you already installed the Driver on your PC / Laptop. If not, you can download it [here](#).

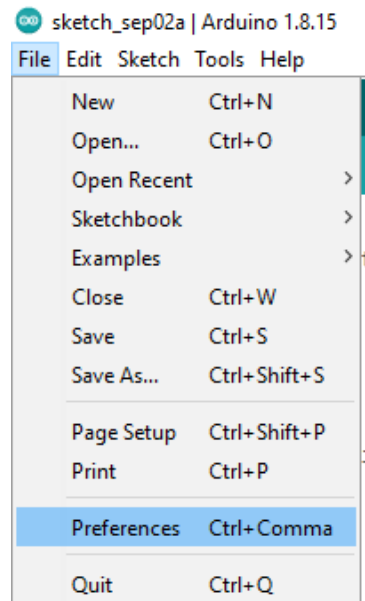
Next we need to import the ESP 32 boards and Blynk platform library to the Arduino IDE.

1. For Blynk library, download the file [here](#)
2. Open Arduino IDE, Sketch - Include Library - Add ZIP Library, find your downloaded file.

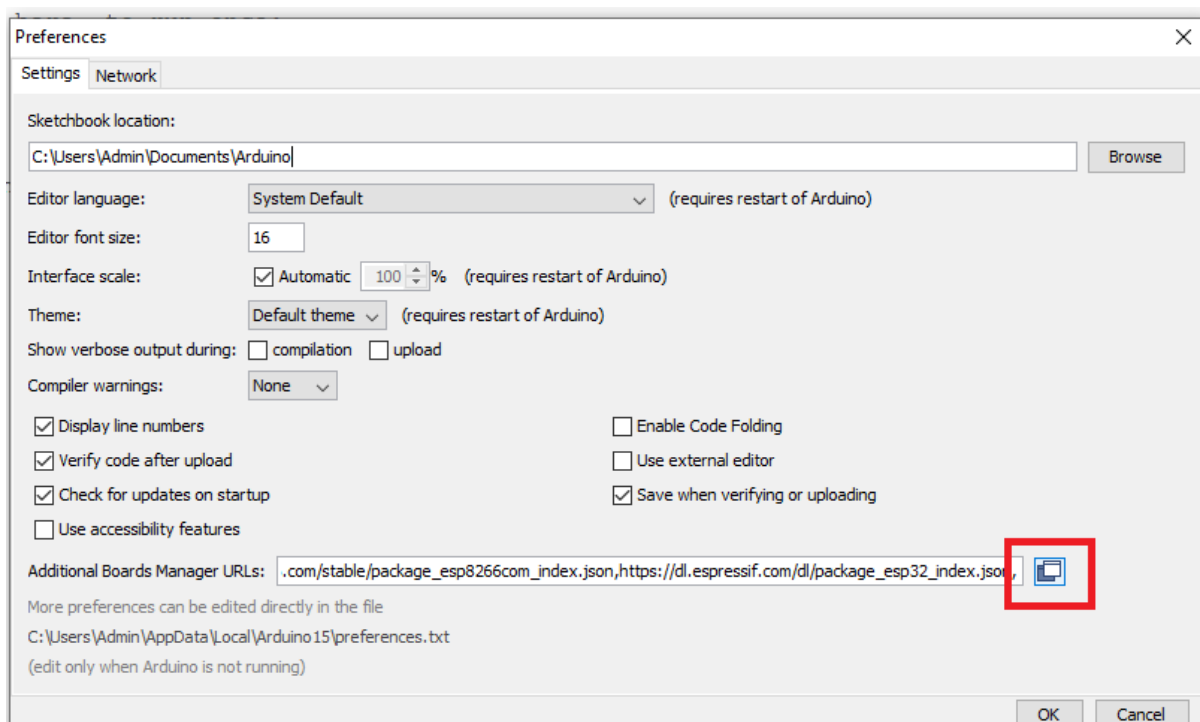




3. Next, ESP 32 Boards, click on File - Preference



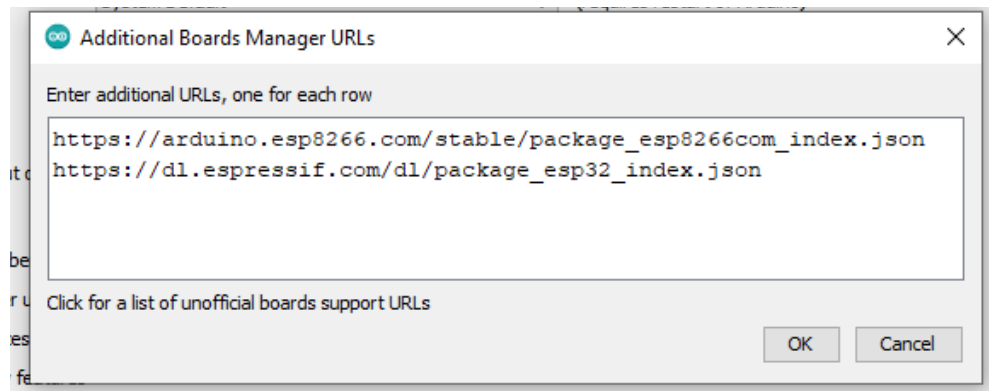
Click on the small window icon at Additional Boards Manager URLs.



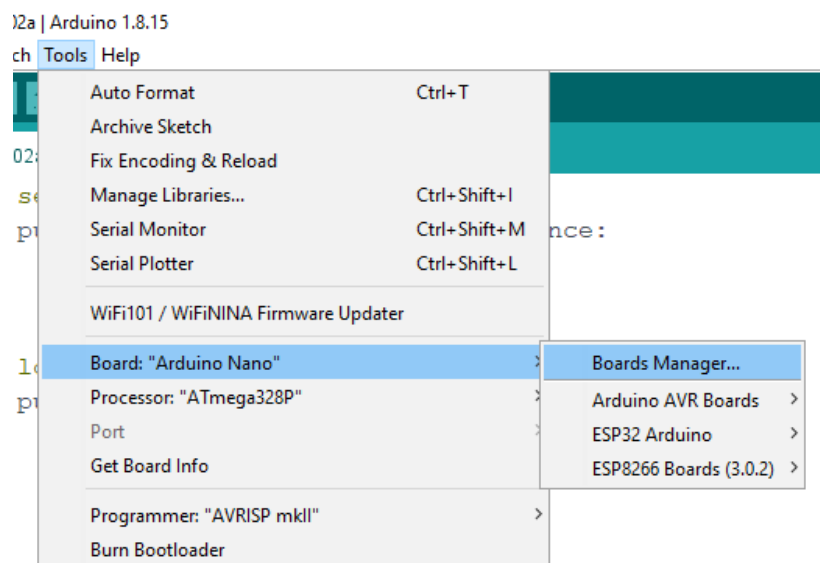
Paste these 2 links, and press Ok.

https://arduino.esp8266.com/stable/package_esp8266com_index.json

https://dl.espressif.com/dl/package_esp32_index.json

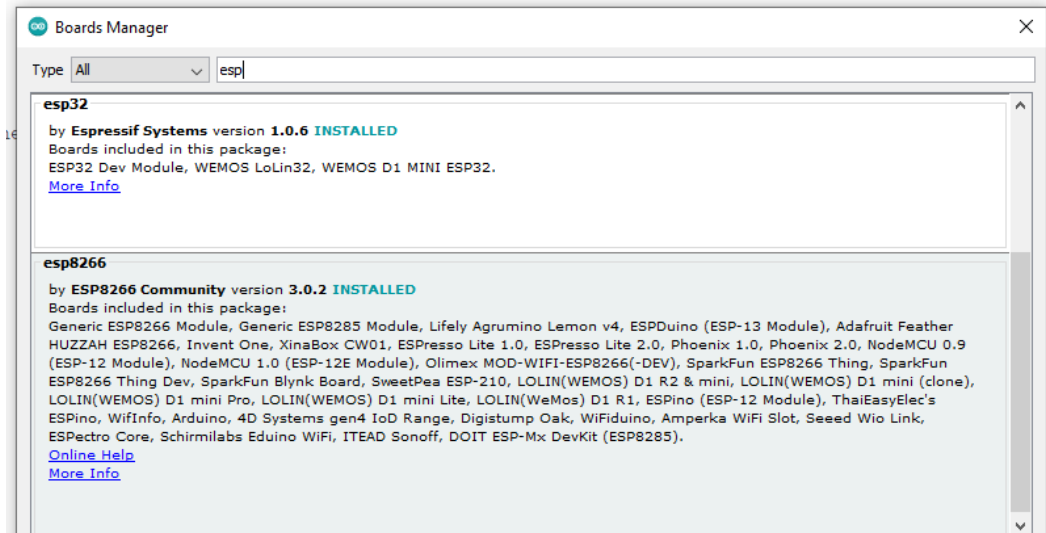


4. Next, click on Tools - Boards - Boards Manager

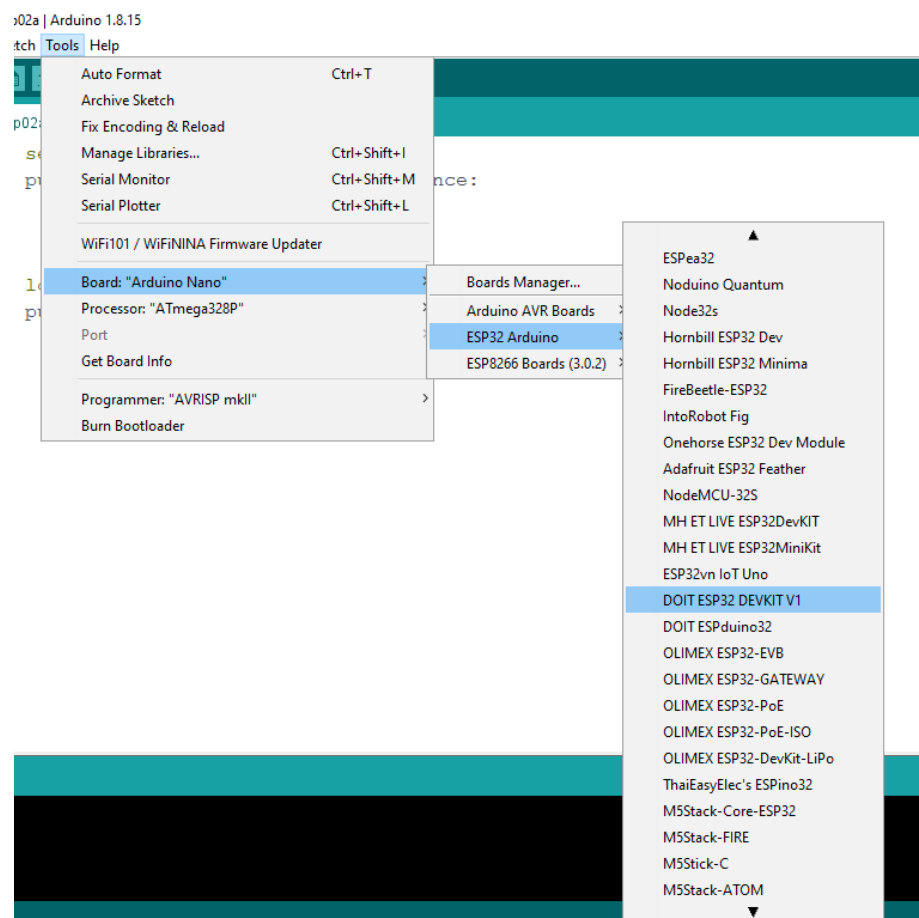


Type in 'esp' and install these two board.

here, to run once:



5. Done. Now you should be able to select ESP 32 Boards. For this module, we will be using DOIT ESP32 DEVKIT V1 Board.





UTM
UNIVERSITI TEKNOLOGI MALAYSIA



2 - Basic of FUNCTION



function() can be **easily understood** as a “flash card”.

You use a **function()** to represent a **long text display**, **formulas**, or a **process that took a long time to write** that you wanted to **use repeatedly** in your programs.

Let's take an example. You got punished at school to **write an essay 100 times**. With functions, you only need to write the **essay once inside the function**, give that function **a short name**, and then you can just **type the function's name 100 times** instead of the whole essay. How convenient is that? As you are a programmer, you **can even use a loop** instead of typing the function's name 100 times.

Second example is a formula. I'm sure most of you are familiar with this equation;

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If you want to **write this exact equation** in Arduino Language, it might be a little **time consuming** and **confusing** with lots of brackets and so on. This is especially when you want to **use the equation multiple times**.

With functions, you can just **write the equation once** in a function, and then **give it a name**. For example “quadratic”. For the next time in the program, we can just use the formula like this;

quadratic(a, b, c)

And then the **function will calculate** the answer for us. This is much easier than having to retype the whole equation over and over.



When we talk about functions, we will frequently talk about **returning value**.

Returning value means that the **function will give back a value to the user** or variables as the output at the end of any calculation or process.

The value returned can be in any data type.

Same as variables, functions() must be initialized anywhere before the use of it. The most suitable place to initialize it is at the top of the program, along with other initialization.

There are 3 types of functions;

1. **void functions():** functions that does not return any value. Usually used in displaying text or a process that does not output any useful value for the user.
2. **data_type functions():** functions that return value to the user or variables. The data_type should be replaced by the correct data type that the function will return.
3. **parametric functions():** usually used for formulas or a process that have parameters that can or want to be changed from time to time.

To **initialize** void functions(), just write as follows;

void function_name();

“Use” the function, we call it “Call” the function. To call a function, you just need to write its name;

function_name();

Naming a function applies the same rules as naming a variable.

You just learn how to initialize and call a function. Now let see how to create the usage of the function;



```
void food();

void setup() {
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  food();
  food();
}

void food(){
  Serial.println("Im so hungry. When will i get my free food?");
}
```

As you can see from the code above;

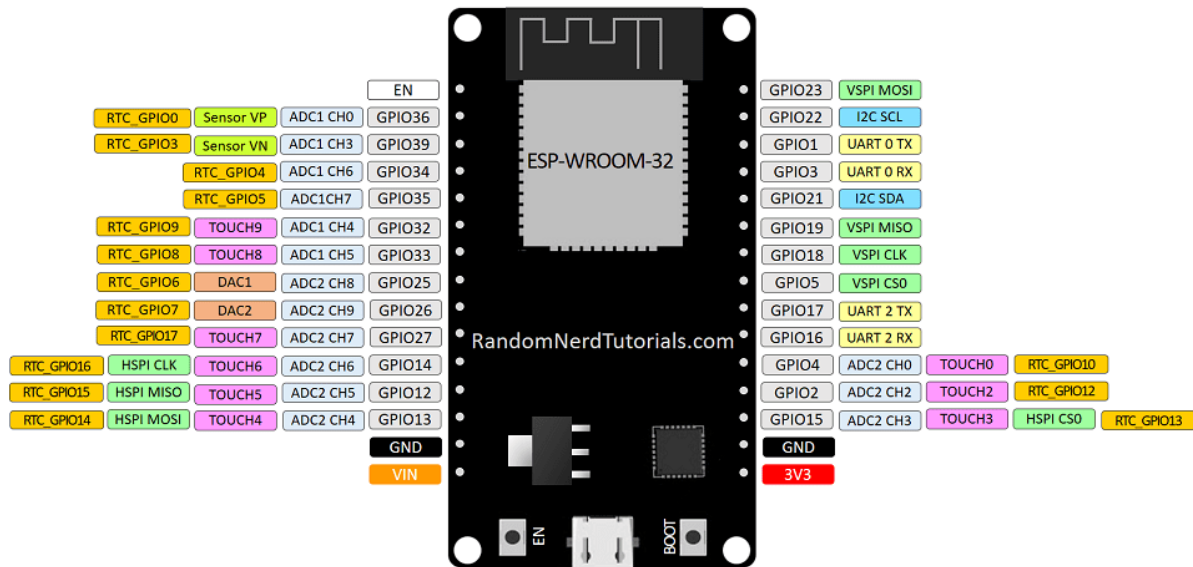
1. We initialized the function at the beginning of the program.
2. We create the usage of the program at the bottom of the program outside void loop() blocks. You can also define the usage of the function at the initialization part, but it will make your initialization part look messed up.
3. We call the function anywhere we want in the program. In this case, inside the void loop(). So, it will display the sentence forever.

This step applies to all types of functions.

3 - Intro to NodeMCU ESP32 Board

ESP32 DEVKIT V1 – DOIT

version with 30 GPIOs



You are already familiar with all the important types of pins from Arduino boards. They are:

- GND
- VIN
- 5V or 3.3V
- Analog and Digital pins
- PWM pins

What's special about ESP 32 board is, all of the pins are Digital Pins. Most importantly, ALL of Digital Pins can be used as PWM pins! That means that you will have access to almost 30 pins that can be used either as analog or digital pins.

Although there are a few special pins that have some restrictions to use, such as only can be used as input pins, etc. But don't worry. Most of the Digital pins with letter 'D' on it are usable. If you're having a problem with one pin, just change it. It will most likely solve your problem.

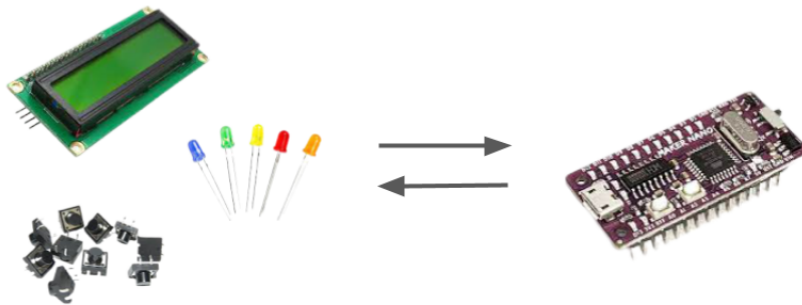


UTM
UNIVERSITI TEKNOLOGI MALAYSIA

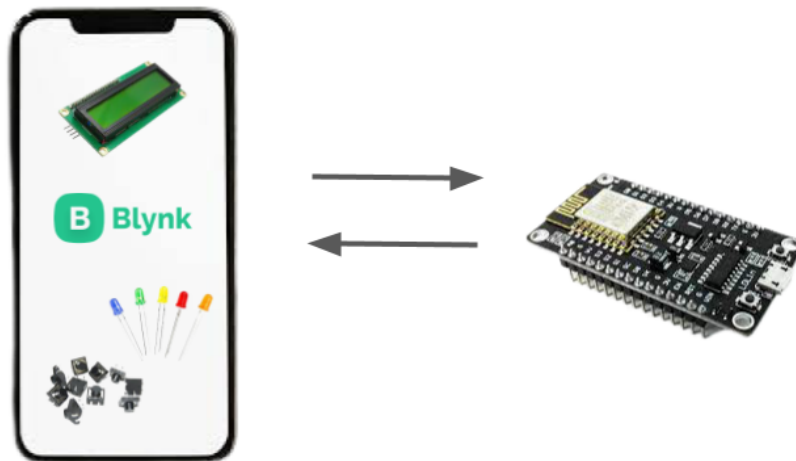


4 - BLYNK Basic Understanding

You might be familiar with normal interaction between Boards and Environment, such as Arduino Boards and Buttons, LEDs, LCD Screen, etc.

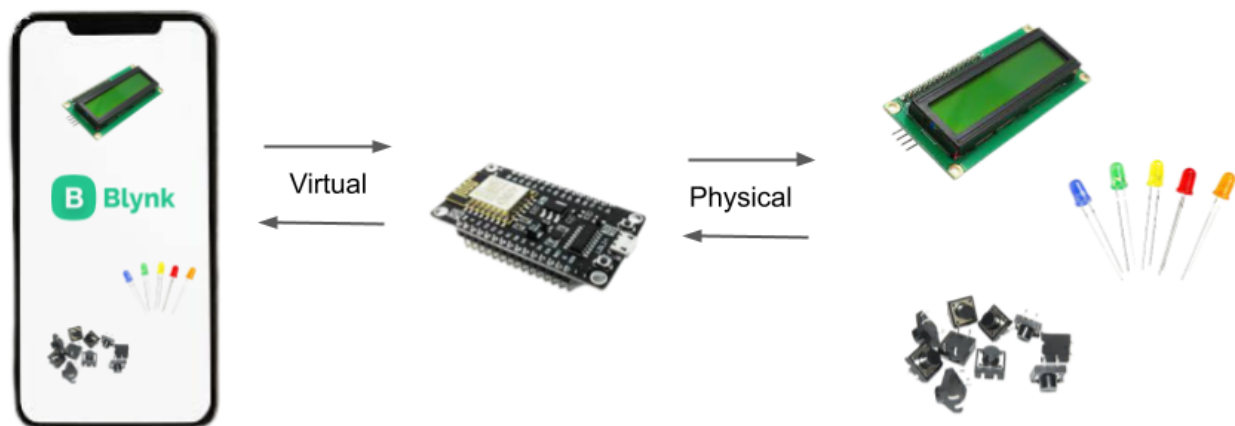


But then, you will need a lot of different items. Let's say you need all 4 items above. With Blink IOT Platform, you can have all the above items (even if you need 10 LEDs and Buttons) in just 2 simple items; a compatible board and a smartphone!



So only with these 2 Items, you can do a lot of projects, with multiple virtual components, virtually! How fun is that? Some of the compatible boards to use with Blynk IOT Platform are NodeMCU Boards. For this workshop, we will be using NodeMCU ESP 32. Not to forget also, Arduino IDE.

Even though the boards now can interact virtually with the Blynk platform, we can still use the board to do physical interaction with physical Buttons and LEDs. As a result, creating a couple of interaction type;



Normal Digital and Analog interaction that we learn with normal Arduino boards is called Physical Interaction. And now, with the Blynk platform added, we have Virtual Interaction. Of course it will be divided into its own input and output, digital and analog. A little confusing right? Lets just list up what we have now:

- Physical Interaction
 - Digital Input
 - Digital Output
 - Analog Input
 - Analog Output
- Virtual Interaction
 - Digital and Analog Input
 - Digital and Analog Output

Thankfully in most IOT platforms, it have been simplify so that both Digital and Analog can be used with the same syntax! We already learn all the Physical Interactions with Arduino. So now we only need to learn about Virtual Input and Output.



In this workshop, we will learn step by step, from setting up your Blynk Account and setup until we can make our own simple wireless system. There will also be some changes in Arduino coding, but don't worry, we will get into it soon.

For the hands on, we will learn:

- How to take Virtual Input and give Physical Output
- How to take Physical Input and give Virtual Output

Now let's get started!



UTM
UNIVERSITI TEKNOLOGI MALAYSIA



5 - Getting Started with BLYNK



First of all, you need to create a Blynk Account. Go to the [Official Blynk IOT Website](#) and [create your account](#). Don't forget to verify your email too if required. Next, download Blynk IOT mobile apps from the Playstore or App Store.

We will need to start our steps in a Computer (Laptop or PC), and then will move to Mobile Phone and our Boards. Technically you can do all the 'Computer Steps' using your internet Browser in your mobile phone. But it will be a lot easier to understand if you do it separately.

5.1 - 5.2 = Computer

5.3 = Arduino IDE

6 = Your mobile phone and NodeMCU Board.

Now, lets login your Blynk Account on your computer to get started.

5.1 Template

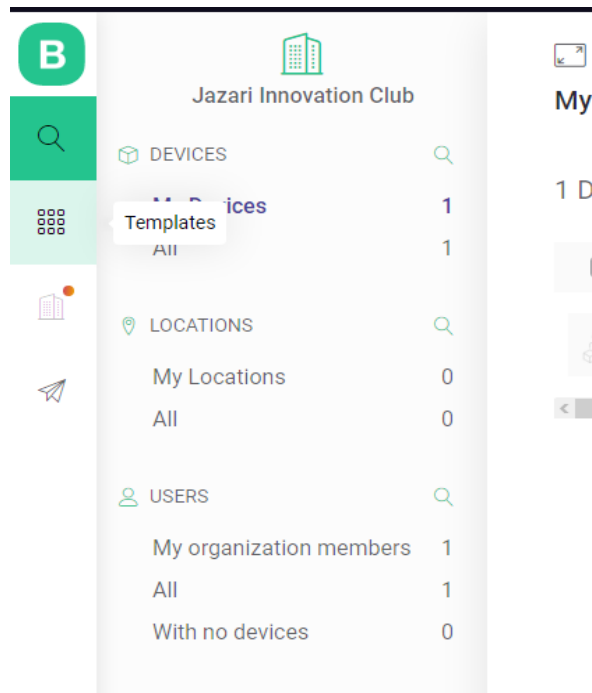
Firstly you need to imagine the Blynk Platform on your Phone as a 'Brand New Virtual Board'. So now, you need to understand the interaction between your NodeMCU and your Phone, as an interaction between a Physical Board and a Virtual Board.

In template, you want to specify what type of Physical Board that your Blynk will interact to. You also will need a specific unique identification code that will be generated later. And lastly, you want to design your Virtual Board, how many virtual pins it will have? Will the pins be Analog or Digital? In Blynk IOT Platform, you can have up to 255 Virtual Pin on your Virtual Board. That's a lot of LEDs and Buttons on your phone screen!

The main purpose of Template is the functionality and size efficiency. For example you have Template A with 5 Virtual pins on it, and Template B with 120 Virtual pins. You will not want to use Template B if you just wanna play with simple LEDs and Buttons projects. It will be a waste of Virtual pins, increasing your project size and loading time. Now if you want to create 4 same simple projects using the same type of boards, you can use a single Template for all the boards.

So this is the basic understanding about the Blynk IOT Template system.

If you already logged in, fill in a nice name for your username and 'Organization' (basically it's just a 'Group' name, in case you want to invite other people to do projects with you). Open the Template Tab.



Create a new template. Give it a nice name. Make sure you specify the correct Physical Board we use, which is ESP 32, with Wifi Connection.

Create New Template

NAME

HARDWARE

ESP32

CONNECTION TYPE

WiFi

DESCRIPTION

19 / 128

Cancel

Done



You will be greeted with a nice looking Info tab of your Template. Pay attention to this page because you will need some stuff from here later for your coding on Arduino IDE.

B

My Wireless Project

Info Metadata Datastreams Events Web Dashboard Mobile Dashboard

19 / 128

00 hrs 00 mins 00 secs

TEMPLATE NAME

My Wireless Project

HARDWARE

ESP32

CONNECTION TYPE

WIFI

DESCRIPTION

This is my template

TEMPLATE ID

TMPLbabpxn_r

MANUFACTURER

Jazari Innovation Club

CATEGORIES

Other x

OFFLINE IGNORE PERIOD

00 hrs 00 mins 00 secs

TEMPLATE IMAGE (OPTIONAL)

Add image

Upload from computer or drag-n-drop
.png or .jpg, min 500x500px

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPLbabpxn_r"
#define BLYNK_DEVICE_NAME "My Wireless Project"
```

Template ID and Device Name should be included at the top of your main firmware

Delete

Cancel

Save

Region: sgp1

Privacy Policy

Don't press anything yet, just relax a little bit, have a look at this tab and move to next step



5.2 Datastream

Moving to the next step. As we have learned in the previous step, we are going to create the virtual pins for our Virtual Board. Open the Datastreams tab.

My Wireless Project

[Delete](#)[Cancel](#)[Save](#)

[Info](#) [Metadata](#) [Datastreams](#) [Events](#) [Web Dashboard](#) [Mobile Dashboard](#)

Create a new Virtual Pin.

[Delete](#)

[Datastreams](#) [Events](#) [Web Dashboard](#) [Mobile Dashboard](#)

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

- Digital
- Analog
- Virtual Pin
- Enumerable
- Location

Lets name it Virtual LED. You can name it whatever you want. Make sure to select the Integer Data Type on Virtual Pin V0. As previously we learned, LEDs can be used as digital components, and use values 0 and 1. If you want to use an analog component, change the Min and Max value to 0 and 255.

Virtual Pin Datastream

NAME

Virtual LED

ALIAS

Virtual LED

PIN

V0

DATA TYPE

Integer

UNITS

None

MIN

0

MAX

1

Default Value

☐ Thousands separator (e.g. 10,000)



☐ ADVANCED SETTINGS

Cancel

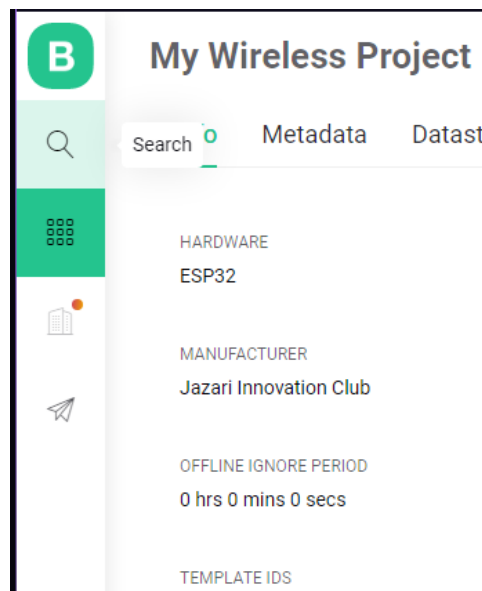
Create

Now create a second Integer Virtual Pin on V1 for our Virtual Button. Now we have two Virtual Pin for our Virtual LED on V0 and Virtual Button on V1.

2 Datastreams

<input type="checkbox"/>	Id	Name	Alias	Color	Pin	Data Type	Units	Mir	Actions
<input checked="" type="checkbox"/>	1	Virtual LED	Virtual LED		V0	Integer		0	
<input checked="" type="checkbox"/>	2	Virtual Button	Virtual Button		V1	Integer		0	

Now save your Template. And open the Search Tab. We need to register our Physical Board which is ESP 32, to the Blynk Platform.




Create a New Device, select From Template and give it a name.


New Device

Choose a way to create new device


From template




Scan QR code



Manual entry



 Create a device by filling in a simple form

Cancel

New Device

Create new device by filling in the form below

TEMPLATE

My Wireless Project

DEVICE NAME

My ESP 32

Cancel

Create



Now you will get into the Device tab with a small popup window. Don't worry, just close it, and open the Device Info tab. You will see an exam same thing, which is important for use later.

The screenshot shows the Blynk web interface for a device named 'My ESP 32'. The 'Device Info' tab is selected, displaying various details about the device's status, activation, and firmware configuration.

Dashboard	Timeline	Device Info	Metadata	Service
STATUS ● Offline		LAST UPDATED Not updated yet	FIRMWARE CONFIGURATION <pre>#define BLYNK_TEMPLATE_ID "TMPLbapxn_r" #define BLYNK_DEVICE_NAME "My ESP 32" #define BLYNK_AUTH_TOKEN "Cy8W0YL7c_vVALqM8UdnJgE6EnFTx9Eh";</pre> <p>Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.</p>	
DEVICE ACTIVATED 6:02 AM Today by xtron135@gmail.com		ORGANIZATION Jazari Innovation Club		
TOTAL ONLINE TIME Wasn't online yet		TEMPLATE NAME My Wireless Project		
AUTHTOKEN **** - **** - **** - x9Eh				

Region: sgp1 Privacy Policy

Stay on this page, and let's open Arduino IDE and proceed to the next step.

5.3 Basic Programming Structures

As we already learned, normal Arduino program structure will look like this:

```
1  
2 void setup() { 2  
3   // put your setup code here, to run once:  
4  
5 }  
6  
7 void loop() { 3  
8   // put your main code here, to run repeatedly:  
9  
10 }
```

It will look exactly like this with these section:

- 1- Declaration Section. Where you declare and define your variables, functions and library.
- 2- Void Setup. Where you initialize some functions, like Serial Monitor, pinMode or running your program once.
- 3- Void Loop. Where most of the time will hold your main program, running continuously.

Blynk's basic programming structure got some addition to it. And it also depends on our ESP 32 Board. Different boards might need to change a little bit at specific lines.

```
1 #define BLYNK_PRINT Serial
2 #define BLYNK_TEMPLATE_ID "Your Template ID"
3 #define BLYNK_DEVICE_NAME "Your Device Name"
4 #define BLYNK_AUTH_TOKEN "Your Token"
5
6 #include <BlynkSimpleEsp32.h>
7
8 char auth[] = BLYNK_AUTH_TOKEN;
9 char ssid[] = "YourNetworkName";
10 char pass[] = "YourPassword";
11
12 BlynkTimer timer;
13
14 //----- 1
15
16 //-----
17
18 void setup() { //Core
19   Serial.begin(115200);
20   Blynk.begin(auth, ssid, pass); 2
21
22   timer.setInterval(1000L, myTimerEvent);
23
24 }
25
26 //-----
27
28 void myTimerEvent() {
29                                     3
30 }
31
32 //-----
33
34 void loop() {
35   Blynk.run();
36   timer.run(); 4
37 }
38
39 //-----
40                                     5
```

Let's have a look at each part of the basic program.



1- Declaration Section. This is where you normally declare and define your variables and functions.

2- Void Setup. This is where you initialize stuff. pinMode, Serial Monitor and for our new Blynk friends 3 new lines to set up your network to connect your board and status report.

3- Virtual Output Section. Inside this function, you will put all the programs for your Virtual Output. Anything related to Virtual Output that you want to display on Blynk such as LEDs, Meters and LCDs goes inside here. Leave it empty if not used.

4- Void Loop Section. Other than these 2 new lines to run the Blynk system, you can input any program related to normal Physical-to-Physical Interaction here. You can also define them as a function in Section 5, and call it inside Void Loop.

5- Virtual Input and Physical Interaction Function. Last part is where you put your program related to Virtual Input. We need to create a new function for each Virtual Pin used for input here. We will learn them shortly. It is empty for now in case we don't want to use it. You can also create functions for Physical-to-Physical Interaction here, and call it inside Section 4 which is Void Loop.

On Arduino IDE you can find some examples of Blynk basic programs. But it might contain some errors or incompatible boards. so you can copy from us instead. This basic program template has been optimized for use with Blynk and ESP 32 Board, with any project, without worrying about errors.



Basic ESP 32 Blynk Template

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "Your Template ID"
#define BLYNK_DEVICE_NAME "Your Device Name"
#define BLYNK_AUTH_TOKEN "Your Token"
```

```
#include <BlynkSimpleEsp32.h>
```

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

```
BlynkTimer timer;
```

```
//-----
```

```
//-----
```

```
void setup(){ //Core
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  timer.setInterval(1000L, myTimerEvent);
```

```
}
```

```
//-----
```

```
void myTimerEvent(){
```

```
}
```

```
//-----
```

```
void loop(){
  Blynk.run();
  timer.run();
}
```

```
//-----
```

6 - Hands On Session

Now let's start our hands on. First of all, let's start to connect our Blynk system with our NodeMCU Board, and our Blynk apps on mobile phones.

1. Copy the Basic ESP 32 Blynk Template to your Arduino IDE.
2. Return back to our Blynk website on Computer. Last time we stopped there, on the Device page. For this part of your program,

```
1 #define BLYNK_PRINT Serial
2 #define BLYNK_TEMPLATE_ID "Your Template ID"
3 #define BLYNK_DEVICE_NAME "Your Device Name"
4 #define BLYNK_AUTH_TOKEN "Your Token"
5
```

You will need to copy from your Device Info on the Blynk Website (We called it Blynk.Console starting from now). Copy from this box

Click to copy Code

```
#define BLYNK_TEMPLATE_ID "TMPLbabpxn_r"
#define BLYNK_DEVICE_NAME "My ESP 32"
#define BLYNK_AUTH_TOKEN "Cy8W0YL7c_vVALqM8UdnJgE6EnFTx9Eh";
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

into your Arduino IDE like this,

```
1 #define BLYNK_PRINT Serial
2 #define BLYNK_TEMPLATE_ID "TMPLbabpxn_r"
3 #define BLYNK_DEVICE_NAME "My ESP 32"
4 #define BLYNK_AUTH_TOKEN "Cy8W0YL7c_vVALqM8UdnJgE6EnFTx9Eh";
5
```

Note that your Template ID and Auth Token will not be the same as above because this is uniquely generated per user.

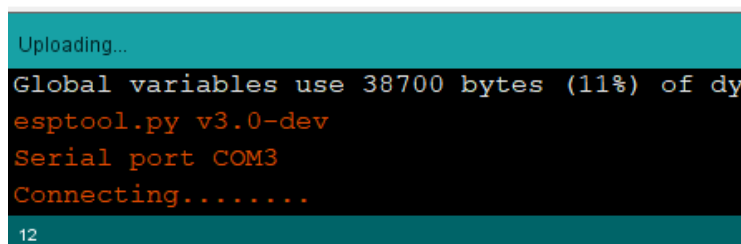
3. Fill up your network details. Make sure your network name and password is spelled correct.

```
7
8 char auth[] = BLYNK_AUTH_TOKEN;
9 char ssid[] = "YourNetworkName";
10 char pass[] = "YourPassword";
11
```

4. Proceed to Upload the program into the ESP 32 board. Make sure to select the correct port and board, DOIT ESP32 DEVKIT V1.

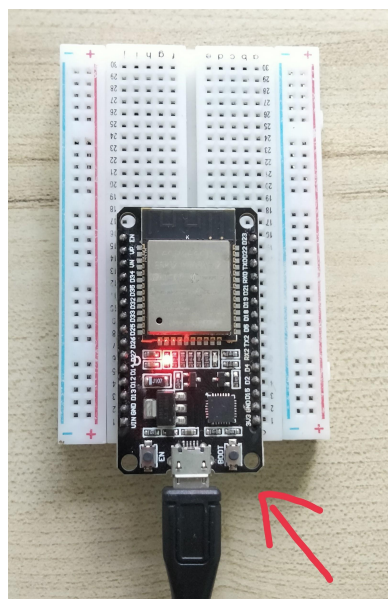
Plug in the micro usb cable provided. Please be aware that ONLY the micro usb provided buy board supplier, or original out-of-the-box micro usb phone charger cable will work. If you buy your micro usb cable elsewhere, it might not work and will cause errors.

Press the upload button, and pay attention at the bottom of the Arduino IDE.



```
Uploading...
Global variables use 38700 bytes (11%) of dy
esptool.py v3.0-dev
Serial port COM3
Connecting.....
12
```

When it shows this **Connecting.....** status, PRESS AND HOLD Boot button on your ESP 32 Board.



While still holding the button, wait for Arduino IDE to show as follow

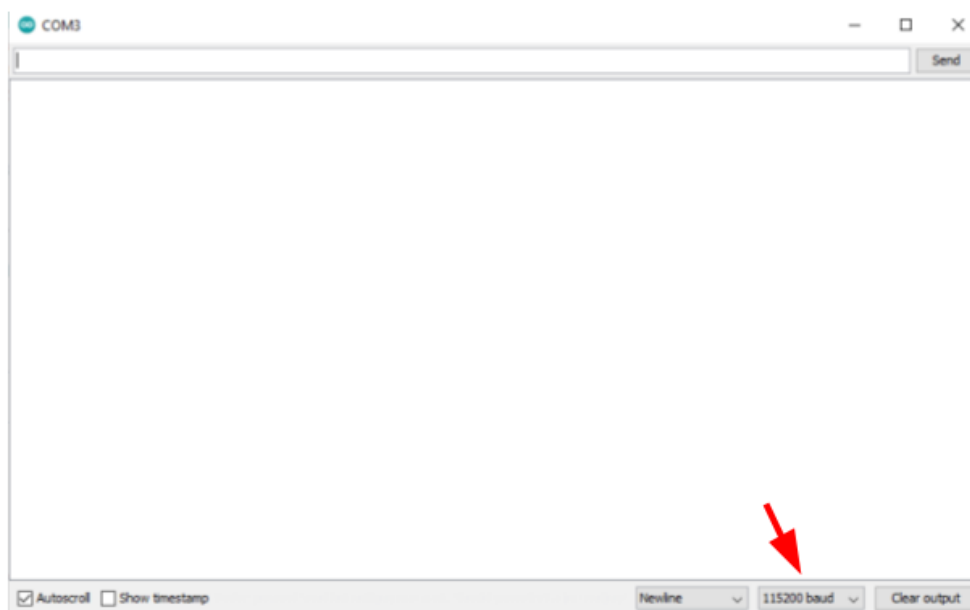
```
Uploading...  
Writing at 0x00010000... (4 %)  
Writing at 0x00014000... (8 %)  
Writing at 0x00018000... (12 %)  
12
```

Now you can release the Boot button.

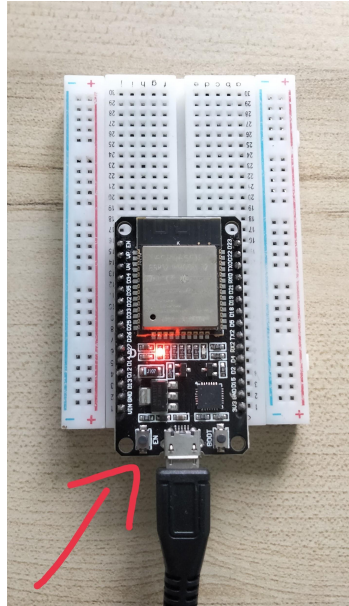
Next, after its successfully uploaded, it will show like this, with the normal Done Uploading status.

```
Leaving...  
Hard resetting via RTS pin...  
12
```

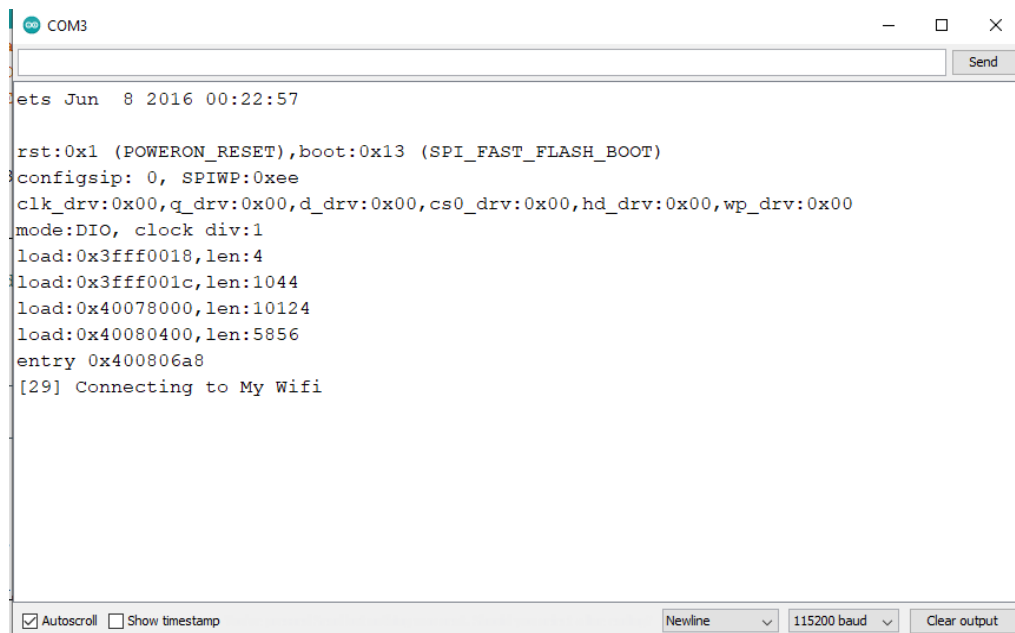
Open up Serial Monitor. Change the Baud Rate as we initialize in our program, which is 115200. It should be empty.



Now press the Reset or En button once.



And then your Serial Monitor will show that your Board is trying to connect to your network.



```

COM3
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
[29] Connecting to My Wifi
  
```



```
COM3
Send

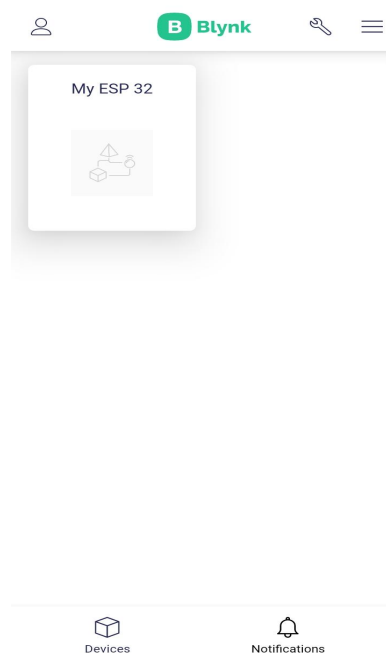
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
[29] Connecting to My Wifi
[4696] Connected to WiFi
[4696] IP: 192.168.61.107
[4697]

  _ _ ) / _ _ _ _ _ / / _
 / _ / / / / / _ \ / ' /
 / _ _ / _ \ _ , / _ / _ / _ \
   / _ _ / v1.0.0-beta.3 on ESP32

[4703] Connecting to blynk.cloud:80
[5333] Ready (ping: 159ms).
```

Your Serial Monitor should display these. Pay attention to the line 'Connected to --Your Network-- '. This shows that your board successfully connected to your network. If these do not show up, please check your network details on your program.

5. Next, open up Blynk mobile apps. Log in your account. and it should display the only board that we already registered.



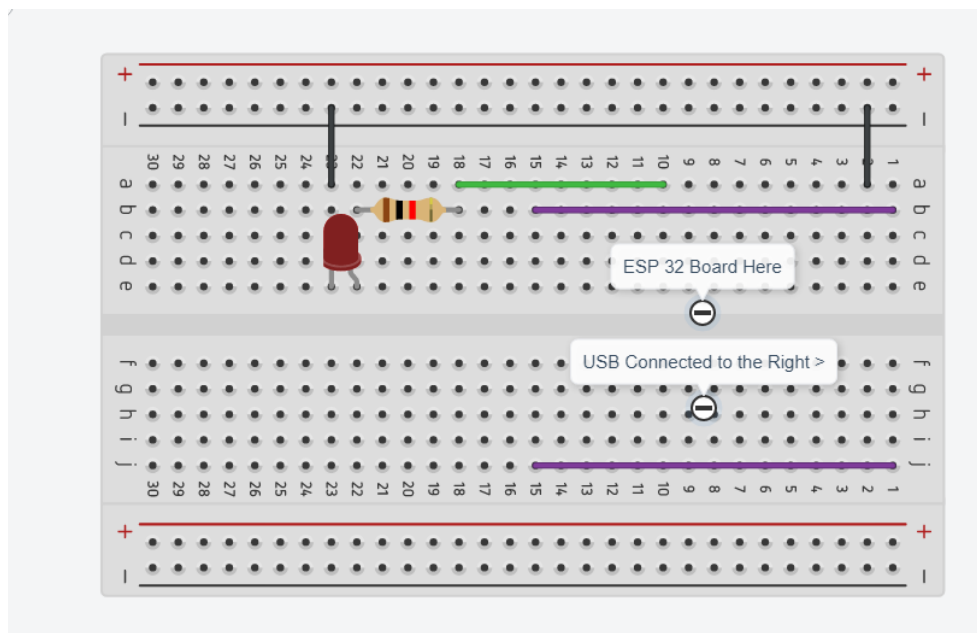
6. Now we are ready to go. Let's get to our first hands on.

6.1 Virtual Button Input to Physical LED Output

You will need these components:

- ESP 32 Board
- Breadboard x 1
- LED x 1 - Connected to Pin D19
- Resistor 200 - 330 Ω
- Jumper Wire Male to Male

Now follow the connection as below:



Make sure the LED pins are correct. Anode (Longer pin) at coordinate [e,22] and Cathode (Shorter Pin) at coordinate [e,23].

We are going to use Virtual Pin V1 that we created on the Datastream. Now let's have a look at the program:



```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "Your Template ID"
#define BLYNK_DEVICE_NAME "Your Device Name"
#define BLYNK_AUTH_TOKEN "Your Token"
```

```
#include <BlynkSimpleEsp32.h>
```

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

```
BlynkTimer timer;
```

```
//-----
#define PLED 19
#define VButton V1
//-----
```

```
void setup(){ //Core
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  timer.setInterval(1000L, myTimerEvent);
```

```
  pinMode(PLED, OUTPUT);
}
```

```
//-----
```

```
void myTimerEvent(){
```

```
}
```

```
//-----
```

```
void loop(){
  Blynk.run();
  timer.run();
}

//-----
BLYNK_WRITE(VButton)
{
  int value = param.asInt();
  Blynk.virtualWrite(VButton, value);

  if(value == 1) digitalWrite(PLED, HIGH);
  else digitalWrite(PLED, LOW);
}
```

As you can see, we add a BLYNK_WRITE function at Section 5 of the programming structure. The first 2 lines is the core of the function;

```
int value = param.asInt();
Blynk.virtualWrite(Pin_Name, value);
```

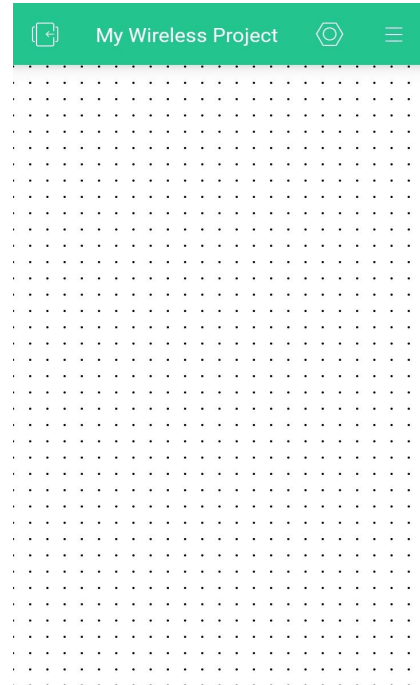
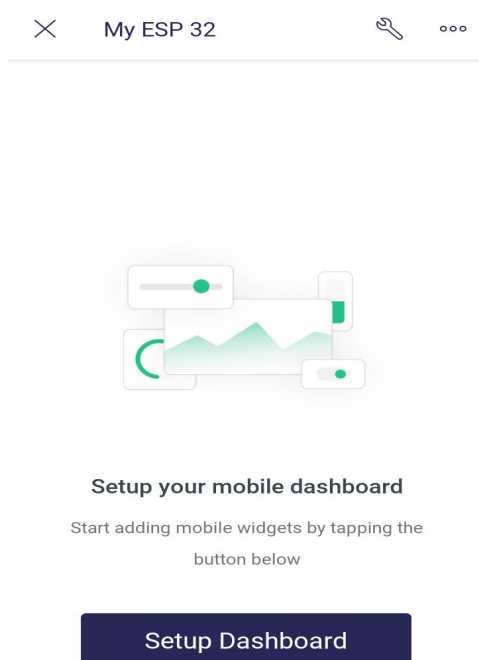
These 2 lines read data input from Blynk mobile apps.

You can use any Virtual Pin that you set as Input on this section. Please note that each Virtual Pin needs a separate function. So if you have 3 Virtual Input pins, you will need 3 different functions on section 5.

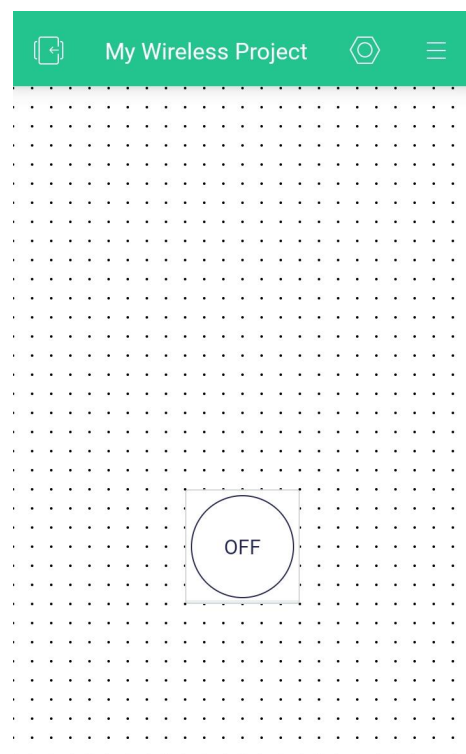
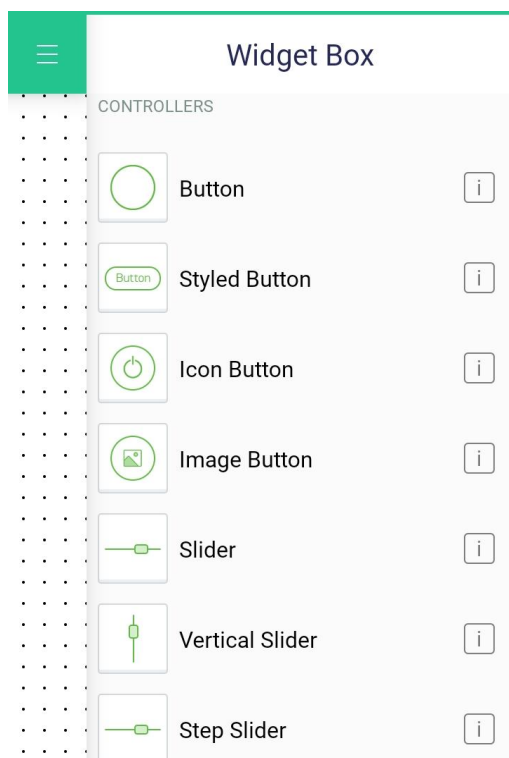
Now upload your program. Don't forget the steps to hold the Boot button, open Serial Monitor when finished uploading and lastly pressing the Reset or En button to make sure your board is connected to your network.

Next we need to set up our Mobile Apps.

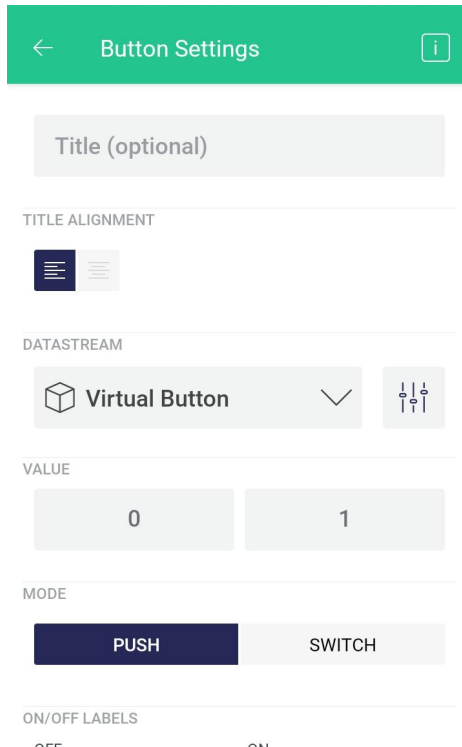
Open up your Blynk Apps and select your device name. Set up the dashboard.



On the upper right corner, there are various choices of widgets that you can use with Blynk. We will just stick with Button and LED. Choose a Button and place it in the middle of the screen.



Tap the button you just placed to enter the setting. Choose the correct Datastream. If you don't remember the Virtual Pin number, you can press on the right of the Data stream to refer.



← Button Settings ⓘ

Title (optional)

TITLE ALIGNMENT

DATASTREAM

Virtual Button ✓ ⚙️

VALUE

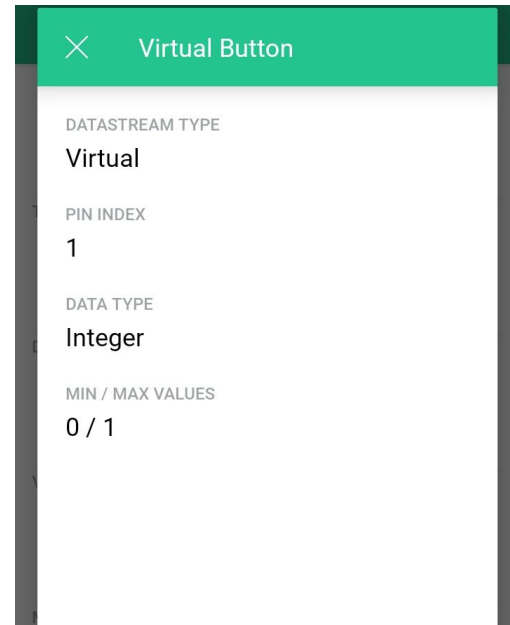
0 1

MODE

PUSH SWITCH

ON/OFF LABELS

OFF ON



✕ Virtual Button

DATASTREAM TYPE

Virtual

PIN INDEX

1

DATA TYPE

Integer

MIN / MAX VALUES

0 / 1

Now hit the Back button on the top left. And you should have your clean mobile apps interface as below. And if you've done everything correctly, our LED on the Breadboard will turn on when we press the Button in the mobile apps. The LED will turn off when we release the button.

✕ My ESP 32 🔑 ⋮



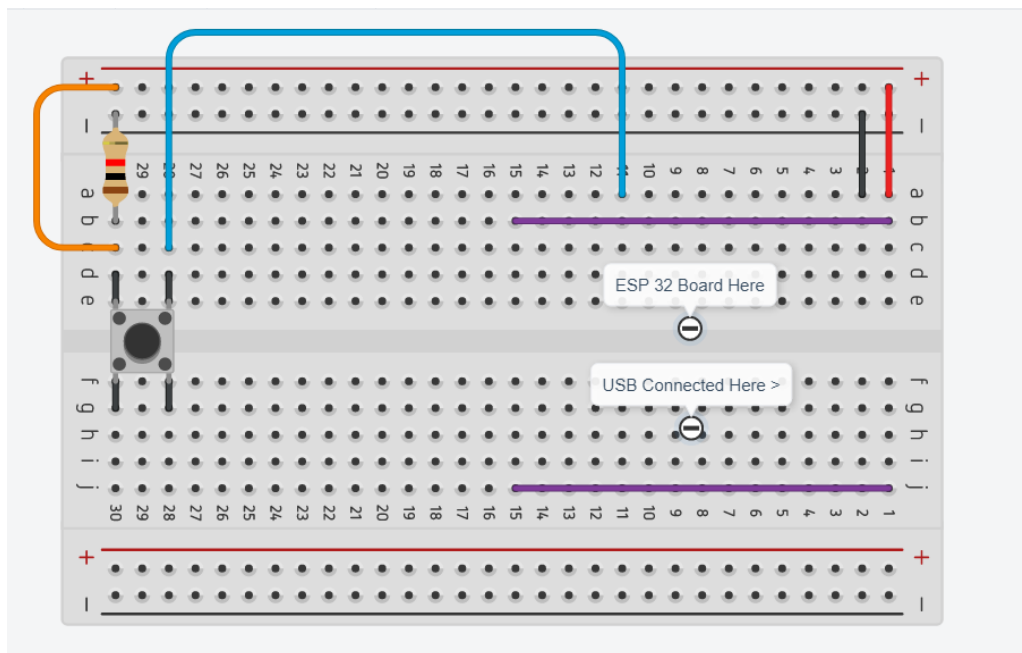
So now you already know how to take Virtual Input from Virtual Button, and give Physical Output on Physical LED. How about adding some more Physical LEDs and Virtual buttons? You got extra LEDS right?

6.2 Physical Button Input to Virtual LED Input

Now for our next hands on, you will need components as below. If you continue from previous hands on, you don't need to clean your Breadboard. Just add the new stuffs.

- ESP 32 Board
- Breadboard x 1
- Push Button x 1 - Connected to Pin D21
- Resistor 1k Ω
- Jumper Wire Male to Male

Follow the connection as below:



Make sure your connection is correct, especially the + and - lines. If you make a mistake with these two, it might break your board.

If everything is good, proceed to the program.



```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "Your Template ID"
#define BLYNK_DEVICE_NAME "Your Device Name"
#define BLYNK_AUTH_TOKEN "Your Token"
```

```
#include <BlynkSimpleEsp32.h>
```

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

```
BlynkTimer timer;
```

```
//-----
#define PButton 21
#define VLED V0
//-----
```

```
void setup(){ //Core
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  timer.setInterval(100L, myTimerEvent);

  pinMode(PButton, INPUT);
}
```

```
//-----
```

```
void myTimerEvent(){
  int value = digitalRead(PButton);

  if(value == 1) Blynk.virtualWrite(VLED,1);
  else Blynk.virtualWrite(VLED,0);
}
```

```
//-----
```



```
void loop() {  
  Blynk.run();  
  timer.run();  
}
```

```
//-----
```

So for this hands on, we just add some stuff inside Section 3 which is the myTimerEvent function. This function will run every 100 milliseconds as we write in the Void Setup. We can change the value to fix any delay or lag.

The new stuffs we add are as below:

```
int value= digitalRead(Physical_Pin);  
  
if(value == 1) Blynk.virtualWrite(Virtual_Pin,Value)  
else Blynk.virtualWrite(Virtual_Pin,Value);
```

Firstly we just read the Physical Input exactly as the same as a normal arduino.

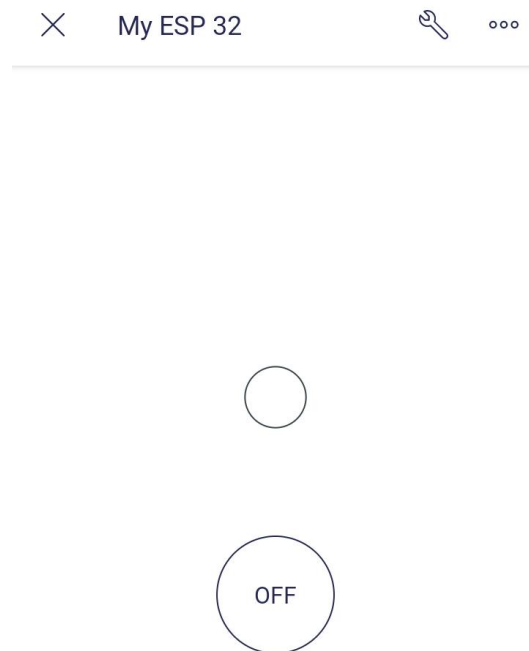
And then, we use Blynk.virtualWrite to create Virtual Output on the mobile apps. For Virtual Output, you can choose either to squeeze them all into the myTimerEvent function or create a new Timer Function for each Virtual Pins. Please take note that you cannot send more than 10 values every 1 second. This is to prevent the runtime errors.

Now upload your program, and don't forget the usual important steps. Boot, Serial Monitor and Reset or En. Make sure your board is connected to the internet.

Next for the mobile apps, most simple widget just use the exact same steps to set up. Just choose your widget, and select the correct Datastream.

Now open the setting on the top right (Wrench Icon), find a LED widget, and select the Virtual LED Datastream that we have created.

If you are not removing previous program and components, your mobile apps should look something like this:



You can even play both of them at the same time. Press Virtual Button, Physical LED turns on. Press Physical Button, Virtual LED turns on.

Now you have mastered how the Virtual Interaction works between NodeMCU Boards and Blynk IOT Platform. Have fun by adding more buttons and LEDs, or even explore new widgets. You can refer to [Blynk Official Documentation](#) on how to use other widgets.

7 - What's next? Control it from anywhere in the World!

Yes. It's exactly as you read. You can control your project from anywhere around the world! As long as you and your board are connected to the internet, even if both of you are not in the same network.

What you need to do is, just log in to your account on your Blynk mobile apps. Just as simple as that. You can even play with your friends. Create a project, give your login credentials to your friends, and that's all, now your friends can control your robots at your home, from his home!

Exclusively for this JxG 2021 Workshop 3, we will demo to you a Remote Controlled Car that we make. We would not have enough time to teach and explain to you about this. But don't worry, we included the components list, pins connection list (No Diagram included) and of course, full coding of the program. If you have all the components, feel free to try yourself. But for those who don't have the components yet, grab the opportunity to try to control it from home during the workshop stream!

That's all from us. We hope you are having fun learning with us, and hopefully you can use this beneficial knowledge for yourself in the future.

7.1 RC Car Component and Connection

You will need:

- [NodeMCU ESP 32](#) x 1 with [micro usb cable](#).
- [Breadboard Small](#) x 1
- [3.7v Li-Ion Rechargeable Batteries](#) x 2
 - You can also use normal 1.5v AA or AAA Batteries x 4. But it will drain faster.
 - Another type is 9v Batteries x 1. Last longer than 1.5v groups, but still not efficient enough as 3.7v Li-Ion Batteries or any rechargeable batteries.
- Battery Holder matching your batteries.
 - If you use 3.7v Li-Ion, you can get the holder [here](#).
 - Don't forget the [charger](#).
- [L298N Motor Driver](#) x 1
- All of the items below you can get in [one set here](#). Or you can buy them separately if you want to use other type of body robot:
 - A pair of [TT Motors with Wheels](#) and [Brackets](#) x 2 if necessary. Make sure these motors come with pre-soldered wires. Or else you need to solder by yourself. You can ask the supplier to confirm this.
 - [Castor Wheel](#) x 1
 - Any suitable Body Robot.
- Male to Male and Male to Female [Jumper wires](#), 1 set each type.



This is our JxG RC robot with a 3D printed body.



For the connection, refer the list below:

- Batteries + connected to +12v terminal on Motor Driver
- Batteries - connected to GND terminal on Motor Driver and GND pin on NodeMCU.
- +5v terminal on Motor Driver Connected to Vin pin on NodeMCU
- With the black Heat Sink block on the Motor Driver facing back of your robot:
 - Left Motor:
 - ❖ Red wires connected to OUT4 terminal on the Motor Driver.
 - ❖ Black wires connected to OUT3 terminal on the Motor Driver.
 - Right Motor:
 - ❖ Red wires connected to OUT2 terminal on the Motor Driver.
 - ❖ Black wires connected to OUT3 terminal on the Motor Driver.
- Next, connection from 6 pin on Motor Driver to NodeMCU
 - ENA pin to D33 pin
 - IN1 pin to D25 pin
 - IN2 pin to D26 pin
 - IN3 pin to D27 pin
 - IN4 pin to D14 pin
 - ENB pin to D12 pin

That's all the connection. Hopefully you can do all of them correctly. Be extra careful with power pins such as + an -, 12v, 5v, VIN and GND. Any mistakes with this pins may result in your board being broken.

Next, set up a brand new template for this robot. We need to use at least 4 Datastream to take Virtual Input from buttons on Mobile Apps. These buttons will control the movement of the robot. You may add any LEDs or Buttons for fun. Dont forget to select the correct board, ESP 32.

Next, register your ESP 32 board as a new device. Copy the Template ID, Auth Token and Template Name from the Device Info page.

Now you are ready to program your robot.

7.2 RC Car Full Programming

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPLPBXTTPHv"
#define BLYNK_DEVICE_NAME "JxG Car"
#define BLYNK_AUTH_TOKEN "irwZ_ZYrPr_WdsjfidNDZlTYxsN-zPOp";

#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "My Wifi";
char pass[] = "pergiokedaimembelibarangkedaitutup";

BlynkTimer timer;

//-----
#define LeftPower 12
#define LeftA 27
#define LeftB 14
#define RightPower 33
#define RightA 26
#define RightB 25

int freq = 5000;
int channelLeft = 0;
int channelRight = 1;
int resolution = 8;
//-----
```



```
void setup(){
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  timer.setInterval(1000L, myTimerEvent);

  pinMode(LeftPower, OUTPUT);
  pinMode(LeftA, OUTPUT);
  pinMode(LeftB, OUTPUT);
  pinMode(RightPower, OUTPUT);
  pinMode(RightA, OUTPUT);
  pinMode(RightB, OUTPUT);

  ledcSetup(channelLeft, freq, resolution);
  ledcSetup(channelRight, freq, resolution);
  ledcAttachPin(LeftPower, channelLeft);
  ledcAttachPin(RightPower, channelRight);

  delay(1000);
}

//-----

void myTimerEvent(){

}

//-----

void loop(){
  Blynk.run();
  timer.run();
}

//-----
```



```
BLYNK_WRITE(V0){ //Move Forward
  int value = param.asInt();

  if(value == 1){ //Move if button pressed
    digitalWrite(LeftA, HIGH);
    digitalWrite(LeftB, LOW);
    ledcWrite(channelLeft, 200);
    digitalWrite(RightA, HIGH);
    digitalWrite(RightB, LOW);
    ledcWrite(channelRight, 200);
  }
  else { //Stop when button released
    digitalWrite(LeftA, LOW);
    digitalWrite(LeftB, LOW);
    ledcWrite(channelLeft, 0);
    digitalWrite(RightA, LOW);
    digitalWrite(RightB, LOW);
    ledcWrite(channelRight, 0);
  }
}

BLYNK_WRITE(V1){ //Move Backward
  int value = param.asInt();

  if(value == 1){ //Move if button pressed
    digitalWrite(LeftA, LOW);
    digitalWrite(LeftB, HIGH);
    ledcWrite(channelLeft, 200);
    digitalWrite(RightA, LOW);
    digitalWrite(RightB, HIGH);
    ledcWrite(channelRight, 200);
  }
}
```

```
else { //Stop when button released
    digitalWrite(LeftA, LOW);
    digitalWrite(LeftB, LOW);
    ledcWrite(channelLeft, 0);
    digitalWrite(RightA, LOW);
    digitalWrite(RightB, LOW);
    ledcWrite(channelRight, 0);
}
}

BLYNK_WRITE(V2){ //Turn Left
    int value = param.asInt();

    if(value == 1){//Move if button pressed
        digitalWrite(LeftA, LOW);
        digitalWrite(LeftB, HIGH);
        ledcWrite(channelLeft, 200);
        digitalWrite(RightA, HIGH);
        digitalWrite(RightB, LOW);
        ledcWrite(channelRight, 200);
    }
    else { //Stop when button released
        digitalWrite(LeftA, LOW);
        digitalWrite(LeftB, LOW);
        ledcWrite(channelLeft, 0);
        digitalWrite(RightA, LOW);
        digitalWrite(RightB, LOW);
        ledcWrite(channelRight, 0);
    }
}
```



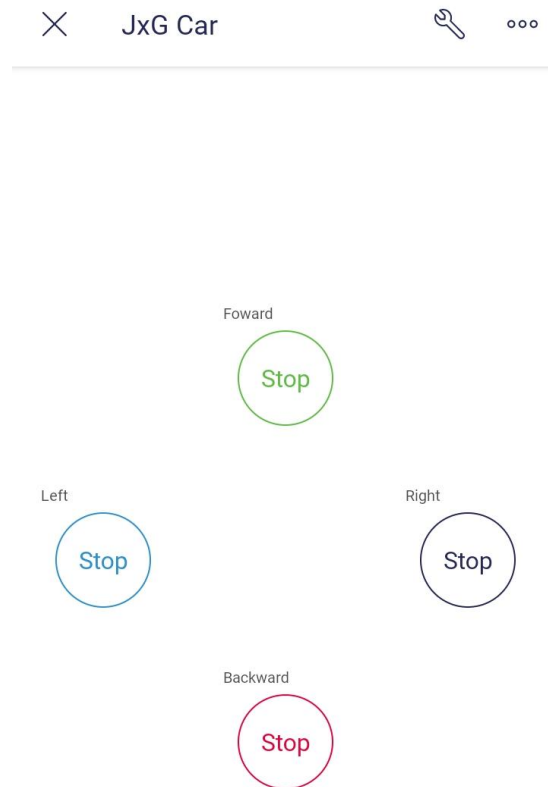
```
BLYNK_WRITE(V3){ //Turn Right
  int value = param.asInt();

  if(value == 1){//Move if button pressed
    digitalWrite(LeftA, HIGH);
    digitalWrite(LeftB, LOW);
    ledcWrite(channelLeft, 200);
    digitalWrite(RightA, LOW);
    digitalWrite(RightB, HIGH);
    ledcWrite(channelRight, 200);
  }
  else { //Stop when button released
    digitalWrite(LeftA, LOW);
    digitalWrite(LeftB, LOW);
    ledcWrite(channelLeft, 0);
    digitalWrite(RightA, LOW);
    digitalWrite(RightB, LOW);
    ledcWrite(channelRight, 0);
  }
}
```

So that's all for the programming part. Upload your program. Don't forget your steps; Boot button, Serial Monitor and Reset or En button. Make sure your board is connected to the internet.

Next open up your Blynk mobile apps and set up your remote control. Use 4 button widgets and select the correct Datastream.

Your mobile apps interface should look something like this



That's all. Now go have fun with your robot. Don't forget that you can control your robot from anywhere in the world. And of course, you can have with your friends far away with a single robot!