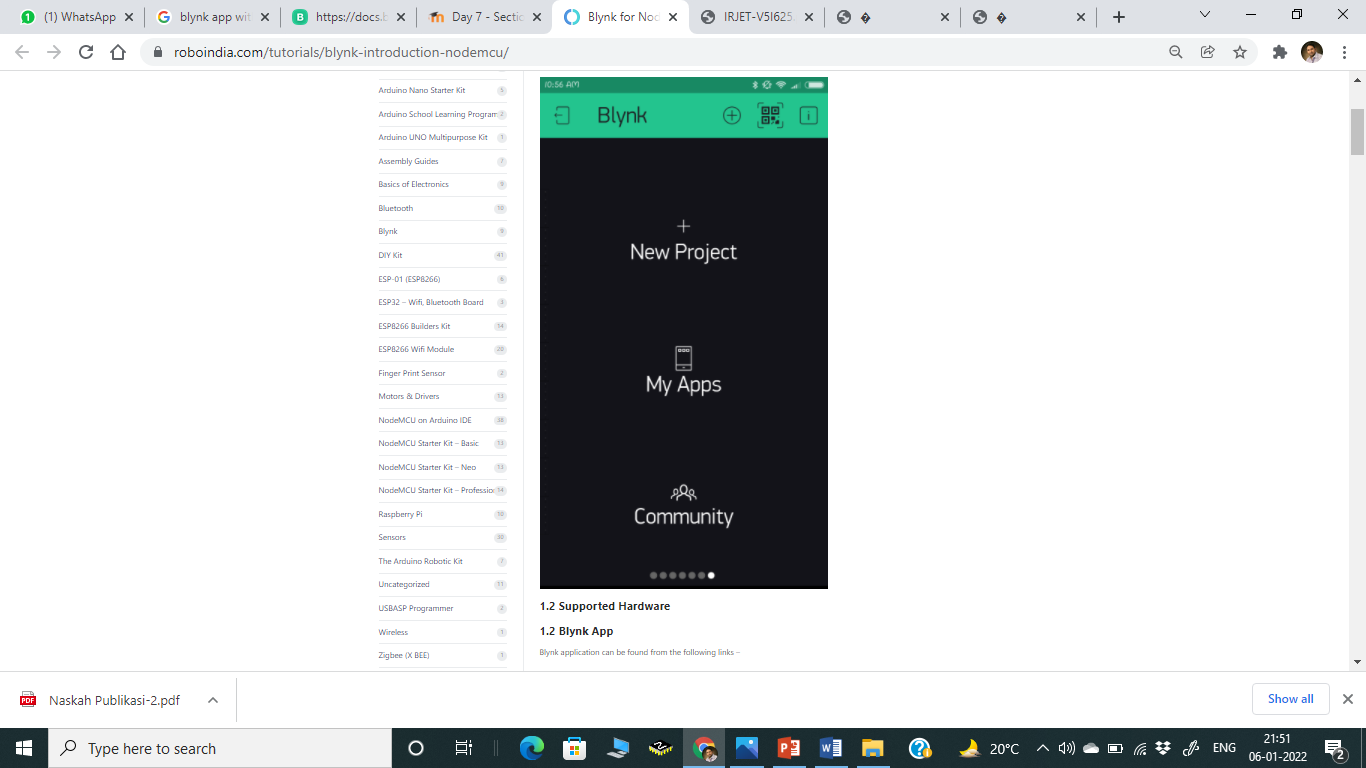
# **3.3. BLYNK FOR NODEMCU**

#### **Introduction:**

Blynk is a Platform with IOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It’s a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.



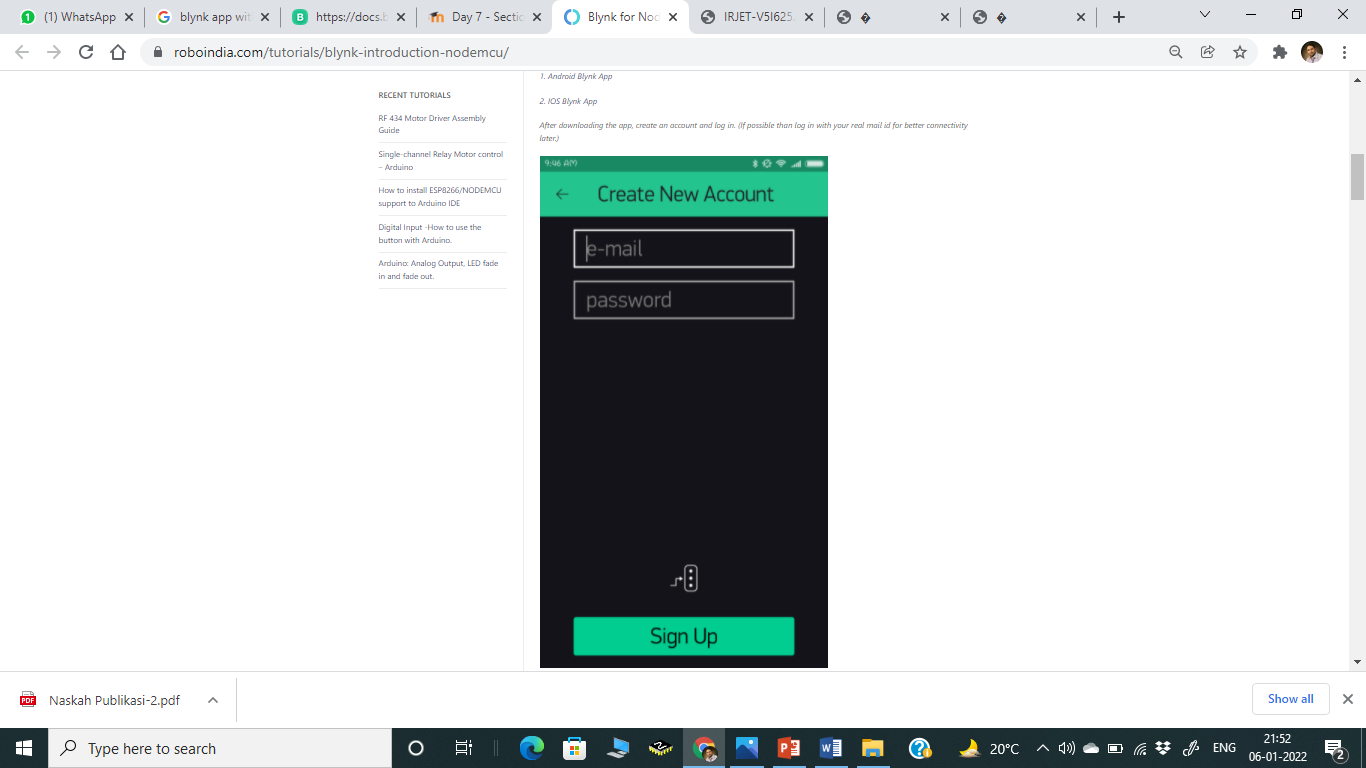
#### **1.2 Supported Hardware**

Blynk application can be found from the following links –

1. Android Blynk App

2. IOS Blynk App

After downloading the app, create an account and log in. (If possible than log in with your real mail id for better connectivity later.)

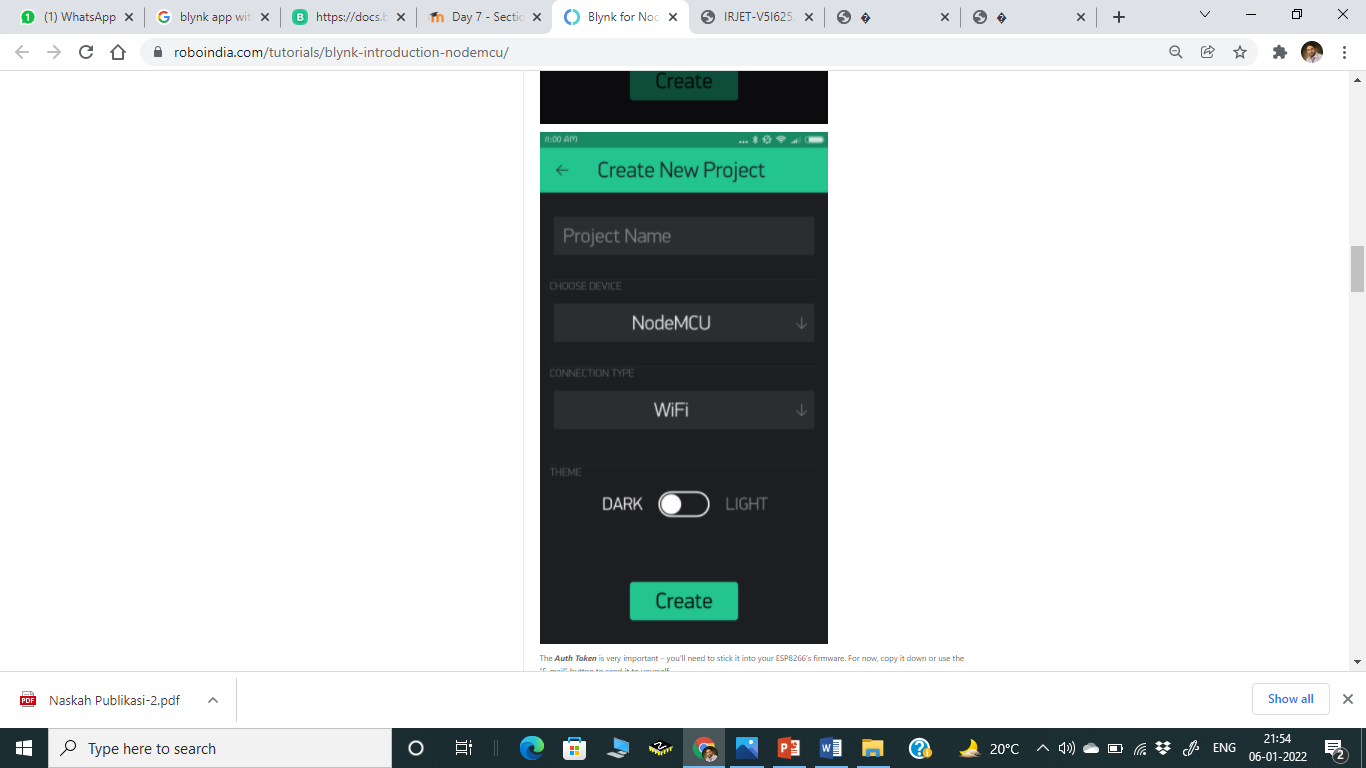
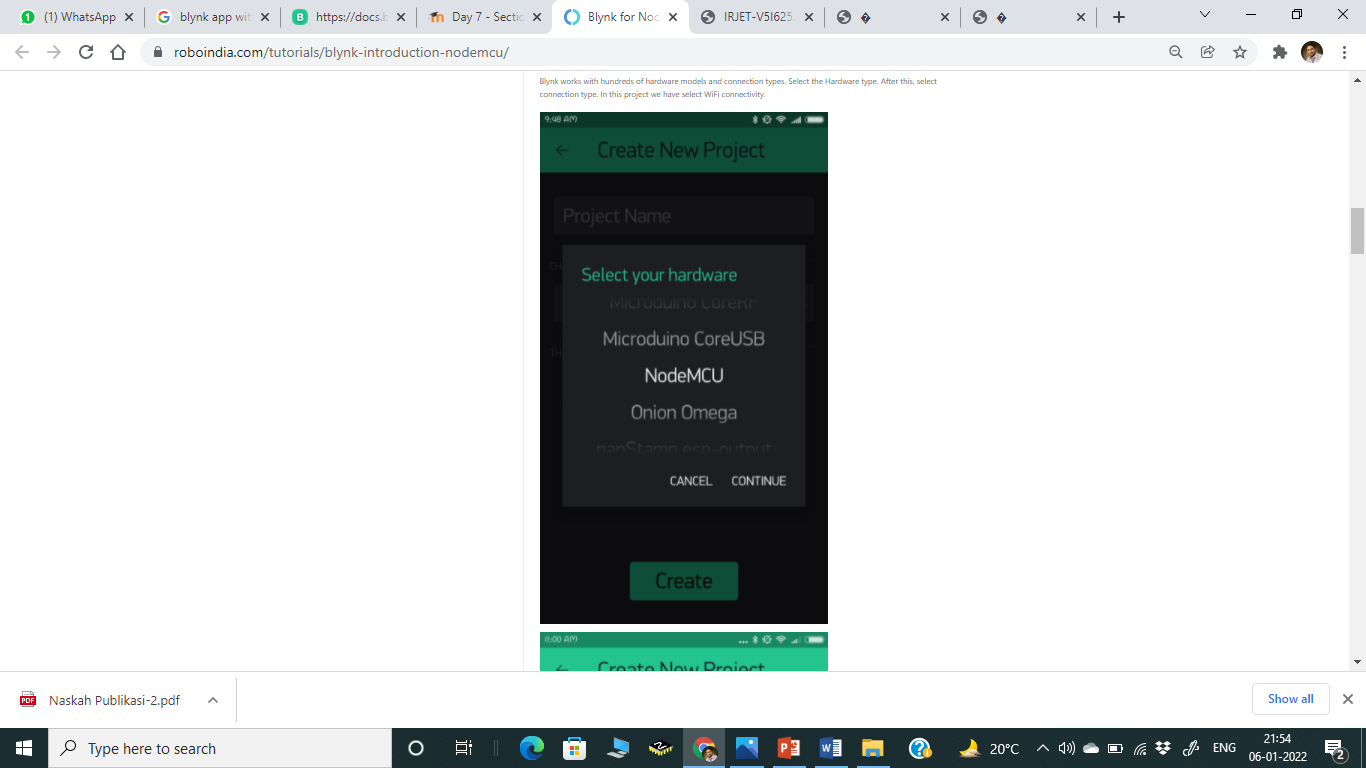


You’ll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from https://github.com/blynkkk/blynk-library/releases , and follow along with the directions there to install the required libraries.

#### **2. Create a Blynk Project**

Click the “Create New Project” in the app to create a new Blynk app. Give it any name.

Blynk works with hundreds of hardware models and connection types. Select the Hardware type. After this, select connection type. In this project we have select WiFi connectivity.



The ***Auth Token*** is very important – you’ll need to stick it into your ESP8266’s firmware. For now, copy it down or use the “E-mail” button to send it to yourself.

#### **3. Add Widgets To The Project**

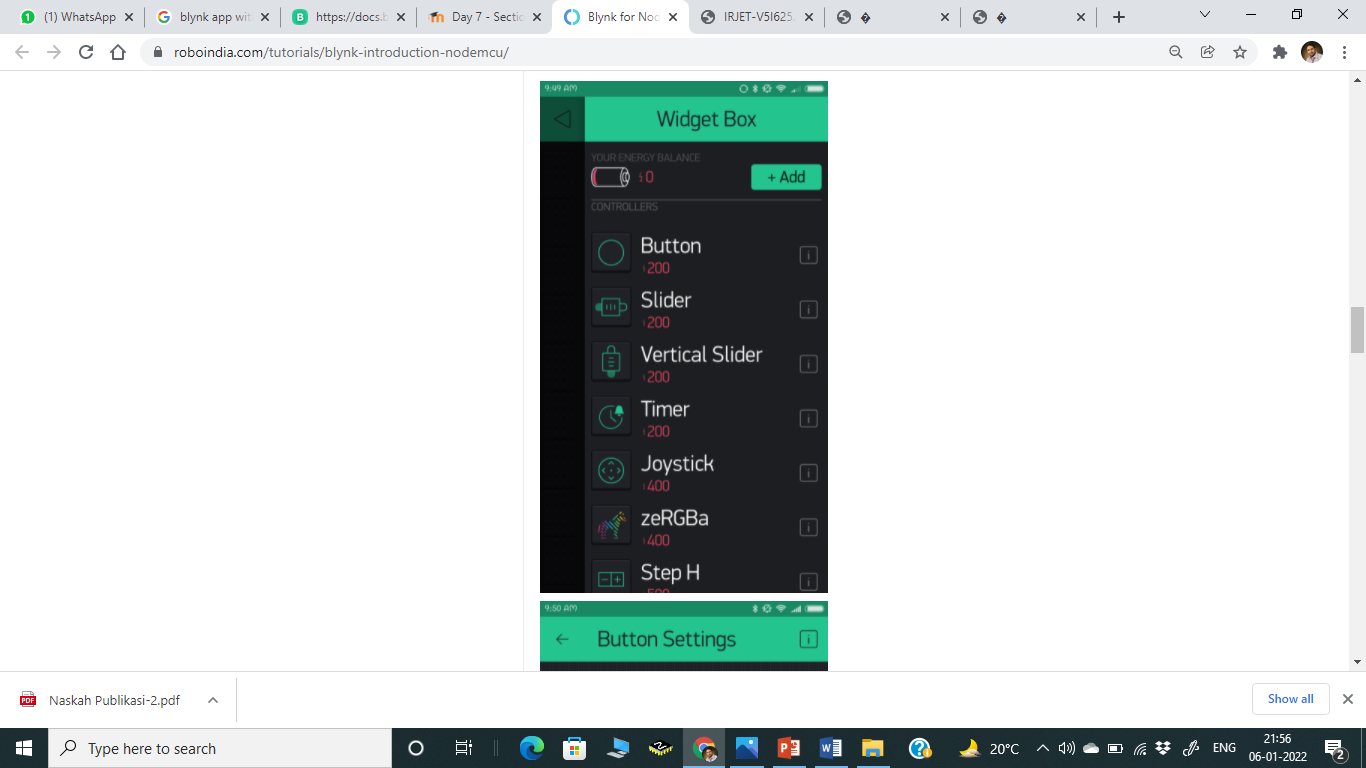
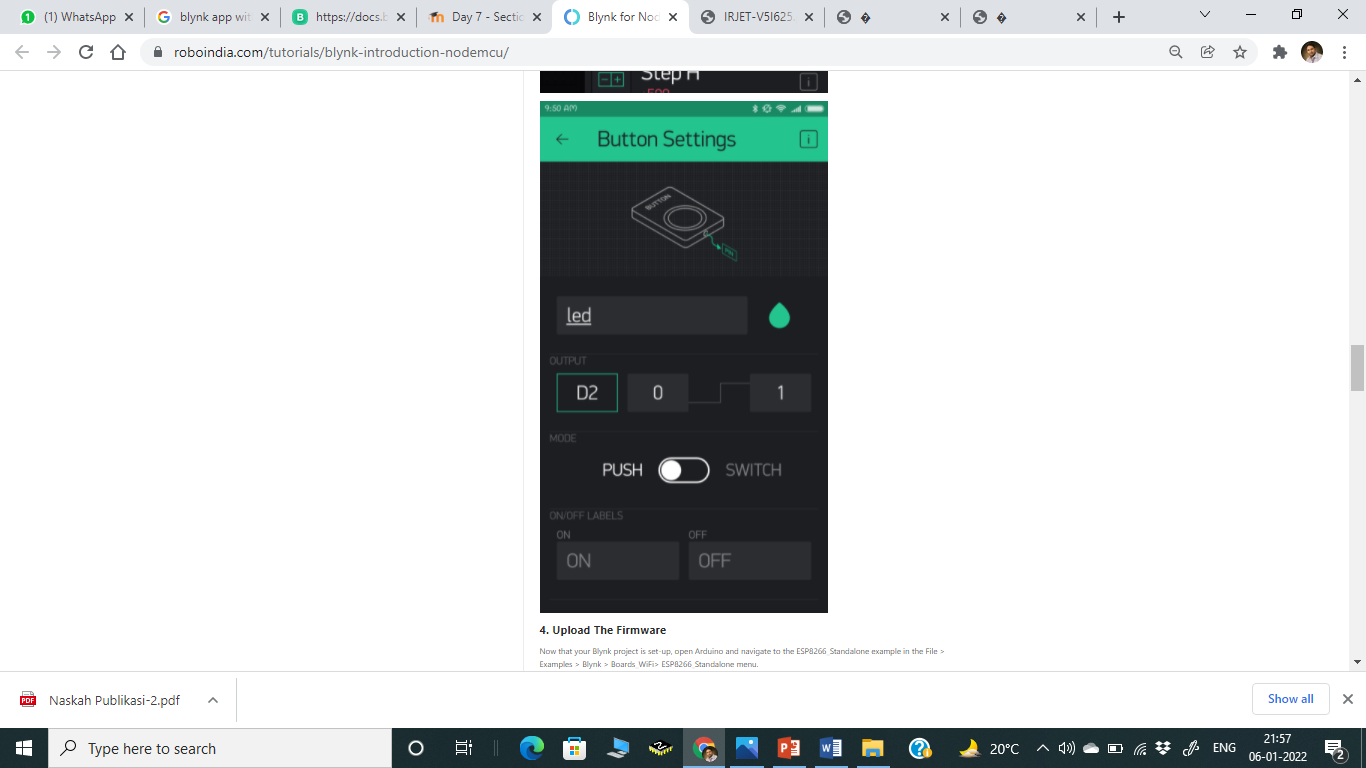
Then you’ll be presented with a blank new project. To open the widget box, click in the project window to open.

We are selecting a button to control Led connected with NodeMCU.

1. Click on Button.
2. Give name to Button say led.
3. Under OUTPUT tab- Click pin and select the pin to which led is connected to NodeMCU, here it is digital pin 2, hence select digital and under pin D2. And Click continue.

Under MODE tab- Select whether you want this button as “push button” or “Switch”.

You have successfully created a GUI for Arduino.

#### **4. Upload The Firmware**

Now that your Blynk project is set-up, open Arduino and navigate to the ESP8266\_Standalone example in the File > Examples > Blynk > Boards\_WiFi> ESP8266\_Standalone menu.

#### **5. Stand Alone Programming Code:**

Before uploading, make sure to paste your authorization token into the auth [] variable. Also make sure to load your Wifi network settings into the Blynk.begin(auth, “ssid”, “pass”) function.

#define BLYNK\_PRINT **Serial**

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).

char auth[] = "YourAuthToken";

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid[] = "YourNetworkName";

char pass[] = "YourPassword";

void setup()

{

 // Debug console

**Serial**.begin(9600);

**Blynk**.begin(auth, ssid, pass);

}

void loop()

{

**Blynk**.run();

}

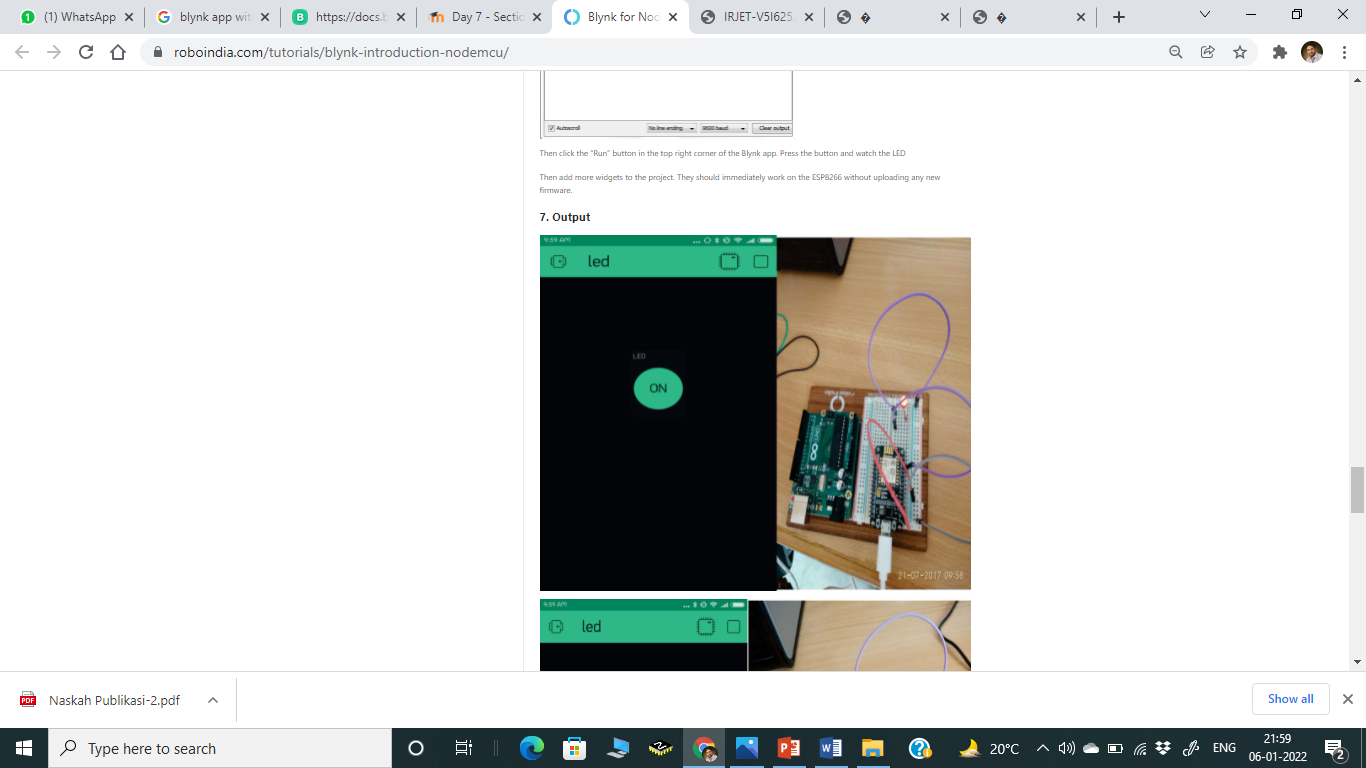
#### **6. Execution**

After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the “Ready” message.

Then click the “Run” button in the top right corner of the Blynk app. Press the button and watch the LED

Then add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware.

#### **7. Output**



**Virtual Pins**

Virtual Pins is a way to exchange any data between your hardware and Blynk app. Think about Virtual Pins as channels for sending any data. Make sure you differentiate Virtual Pins from physical GPIO pins on your hardware. Virtual Pins have no physical representation.

Virtual Pins are commonly used to interface with other libraries (Servo, LCD and others) and implement custom logic.

The device can send data to the App using Blynk.virtualWrite(pin, value) and

receive data from the App using BLYNK\_WRITE(vPIN).

Read below

## Virtual Pin data types

All Virtual Pin values are always sent as Strings and there are no practical limits on the data that can be sent. However, there are certain limitations on the hardware side when dealing with numbers. For example, the integer on Arduino is 16-bit, allowing range -32768 to 32767.

To interpret incoming data as Integers, Floats, Doubles and Strings use:

1

param.asInt();

2

param.asFloat();

3

param.asDouble();

4

param.asStr();

Copied!

You can also get the RAW data from the param buffer:

1

param.getBuffer()

2

param.getLength()

Copied!

# Blynk.virtualWrite(vPin, value)

Use BlynkTimer when you use this command to send data. Otherwise your hardware may be disconnected from the server

Send data in various formats to Virtual Pins.

1

// Send string

2

Blynk.virtualWrite(pin, "abc");

3

​

4

// Send integer

5

Blynk.virtualWrite(pin, 123);

6

​

7

// Send float

8

Blynk.virtualWrite(pin, 12.34);

9

​

10

// Send multiple values as an array

11

Blynk.virtualWrite(pin, "hello", 123, 12.34);

12

​

13

// Send RAW data

14

Blynk.virtualWriteBinary(pin, buffer, length);

Copied!

Calling virtualWrite attempts to send the value to the network immediately.

**Note:** For virtual pins with numbers > 127, the V128 syntax is not available. Please use plain virtual pin number, for example:

1

Blynk.virtualWrite(128, "abc");

Copied!

# BLYNK\_WRITE(vPIN)

BLYNK\_WRITE is a function called every time device gets an update of Virtual Pin value from the server (or app):

To read the received data use:

1

BLYNK\_WRITE(V0)

2

{

3

int value = param.asInt(); // Get value as integer

4

​

5

// The param can contain multiple values, in such case:

6

int x = param[0].asInt();

7

int y = param[1].asInt();

8

}

Copied!

**BLYNK\_WRITE can't be used inside of any loop or function. It's a standalone function.**

**Note:** For virtual pins with numbers > 127, please use BLYNK\_WRITE\_DEFAULT() API

# BLYNK\_WRITE\_DEFAULT()

Redefines the handler for all pins that are not covered by custom BLYNK\_WRITE functions.

1

BLYNK\_WRITE\_DEFAULT()

2

{

3

int pin = request.pin; // Which exactly pin is handled?

4

int value = param.asInt(); // Use param as usual.

5

}

Copied!

# Blynk.syncAll()

Requests all stored on the server latest values for all widgets. All analog/digital/virtual pin values and states will be set to the latest stored value. Every virtual pin will generate BLYNK\_WRITE() event.

1

BLYNK\_CONNECTED() {

2

Blynk.syncAll();

3

}

Copied!

# Blynk.syncVirtual(vPin)

This command updates individual Virtual Pin to the latest stored value on the server. When it's used, a corresponding BLYNK\_WRITE handler is called.

1

Blynk.syncVirtual(V0);

Copied!

To update multiple pins, use:

1

Blynk.syncVirtual(V0, V1, V6, V9, V16);