INDUSTRIAL IOT AND AUTOMATION

LAB MANUAL

Date:

Experiment No: 1

Introduction and Architecture to IoT

IoT (Internet of Things)

IoT means Internet of Things. This is representative of the connection with the superior ability to increase efficiency, influence, save time and costs for organizations. Sensitive connections from devices in the enterprise without human intervention are the highlight that IoT brings to your company.

IoT technology will provide better quality insights to all parts of the business. From the supply of raw materials to inventory control, asset information to assess and prepare for maintenance work, improve the quality of goods in production, and closely monitor the delivery of goods. and a fleet of delivery vehicles, ensuring the quality of customer service experience, etc. Cloud-based IoT applications also allow organizations and businesses to directly access external and internal data in real-time, making decisions faster.

IIoT (Industrial Internet of Things)

IIoT is the Industrial Internet of Things. This is a network of smart devices with their computing capabilities, connected to industry-level data collection, monitoring, exchange, and analysis systems. The main focus of IIoT is on industrial applications such as manufacturing, power plants, agriculture, oil, and gas.

Industrial Internet of Things is a part or can be said to be a subset of Internet of Things, i.e. IIoT under IoT mainly focused on industrial applications. Smart devices play an important role in IIoT, helping to communicate important information in a better way and analyze and capture data in real-time. Using IIoT, business decisions can be made more quickly and accurately. IIoT also helps companies grow by understanding business processes in a better way and making them more efficient.

Difference between IIoT and IoT:

Sl. No	IIOT	IOT
1.	IIoT is more focused on improving the performance of a device, a machine, or an entire business process.	Consumer IoT is deployed to enhance or improve aspects of individuals' daily lives such as smartphones, smart homes, and smart cities.
2.	Use more complex instruments to scale to provide more detailed visibility and enable automated controls and perform complex analyses. These are systems where failure often leads to life-threatening or other emergency situations.	IoT tends to be consumer-grade devices with a low risk of failure. Incidents also do not create an immediate emergency.
3.	IIoT connects critical machines and sensors across industries at scale.	IoT can be used at the consumer level.
4.	Can be programmed remotely onsite.	Provides easy off-site programming.
5.	Processing very large data due to industrial scale.	Handling volumes from very small like Wearable devices to objects you use in everyday life, such as thermostats, irrigation pumps, kitchen appliances, TVs with Internet
6.	Strong security requirements to protect data.	Requires identity and privacy.
7.	There are strict requirements.	Moderately required attribute.
8.	Has a very long life cycle.	Has a short product life cycle.
9.	Has a very long life cycle.	Reliability is getting better and better.

1. Introduction to Arduino :

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

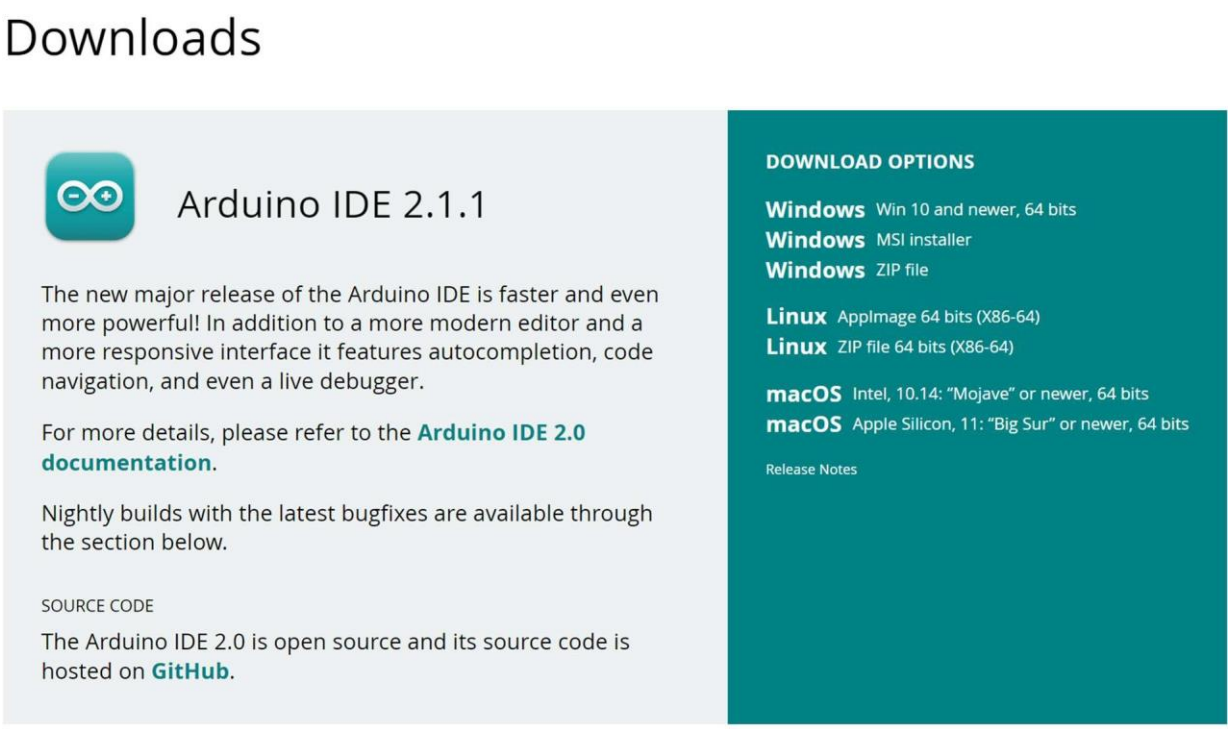
The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board -- you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

1.A) Introduction to Software :

Installing Arduino IDE

The first thing you need is the Arduino IDE. If your computer doesn't have Arduino IDE installed, then visit the official Arduino download page and download the installation file for your preferred operating system

Link: https://www.arduino.cc/en/software

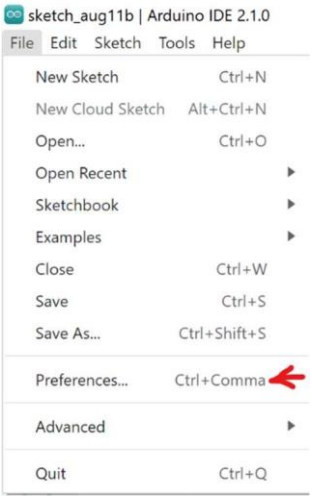


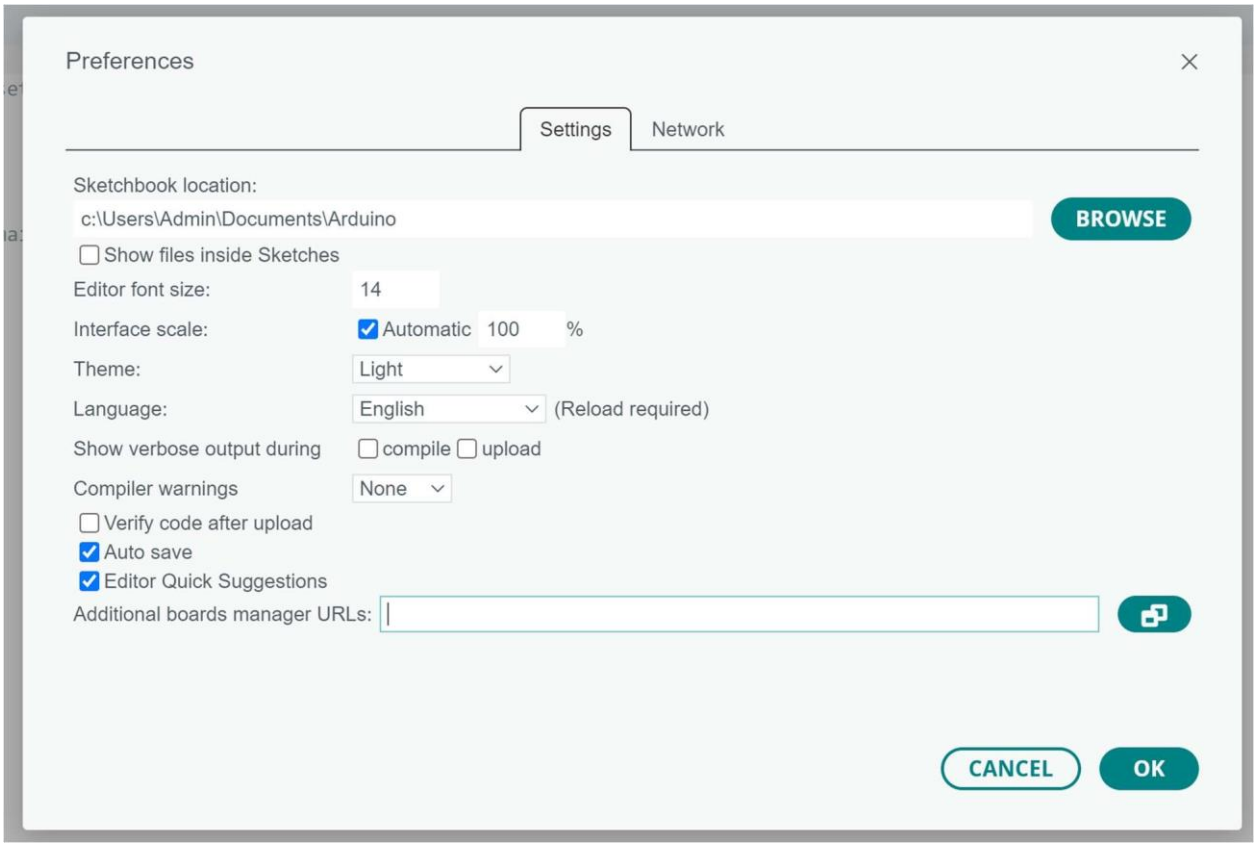
Click and Win 10 and newer and install the software.

Installing the Microcontroller Board in Arduino IDE 1.ESP 32

There's an add-on for the Arduino IDE that allows you to program the ESP32 using the Arduino IDE and its programming language. We'll show you how to install the ESP32 board in Arduino IDE.

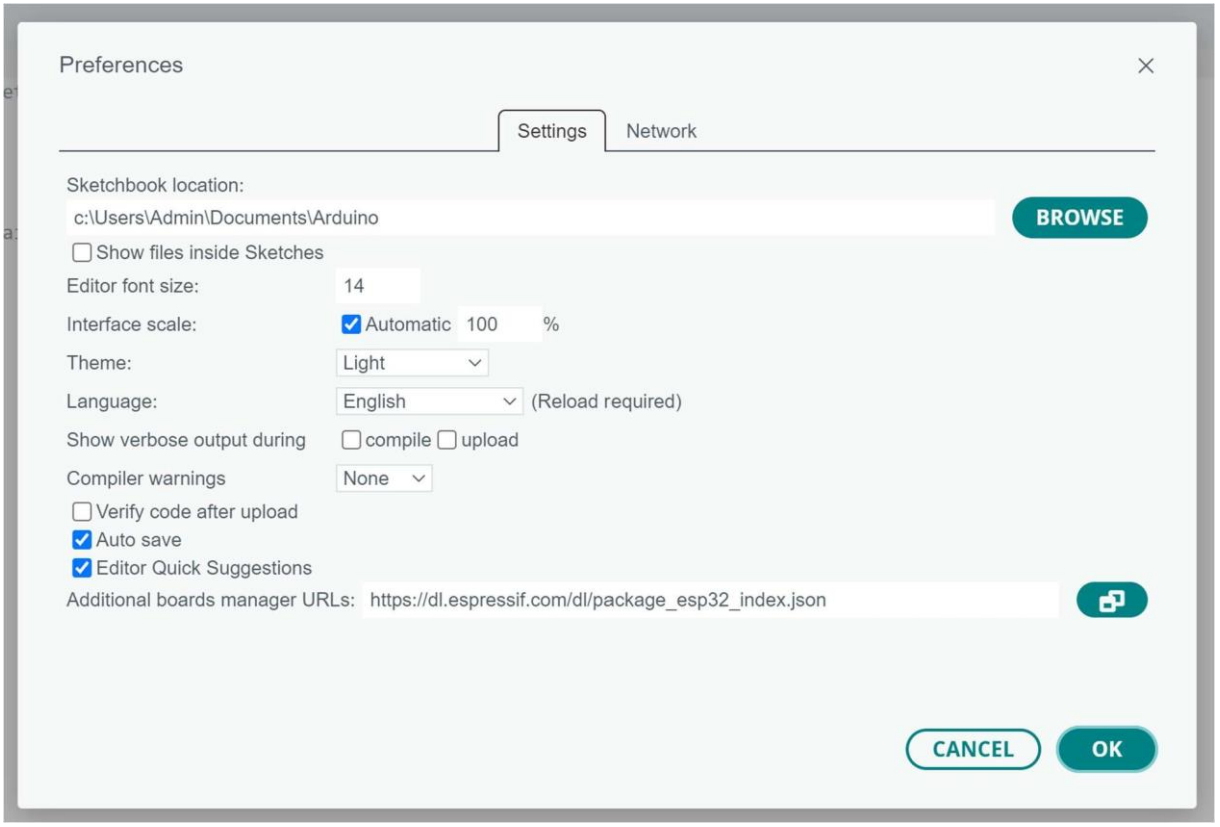
After Installing Arduino IDE open the IDE and Click on Files and select Preference.





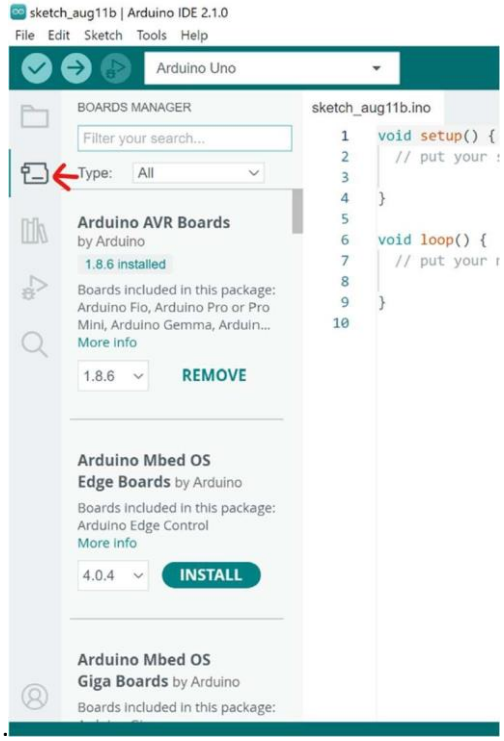
Copy this link and Paste it in the Additional Boards Manager URLs Box as shown Below

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

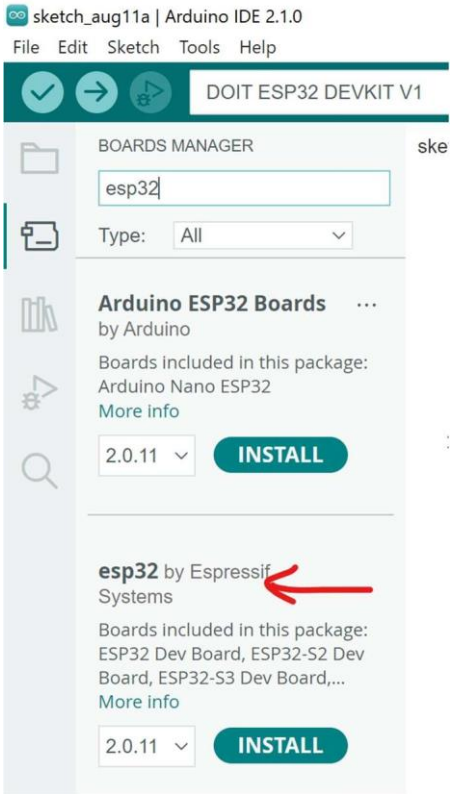


And Click on OK.

Then click on Boards Manager as shown.



Type ESP32 in Search Bar and click on second one and click Install as shown below.



Now ESP32 Board is successfully installed in Arduino IDE and ready to use.

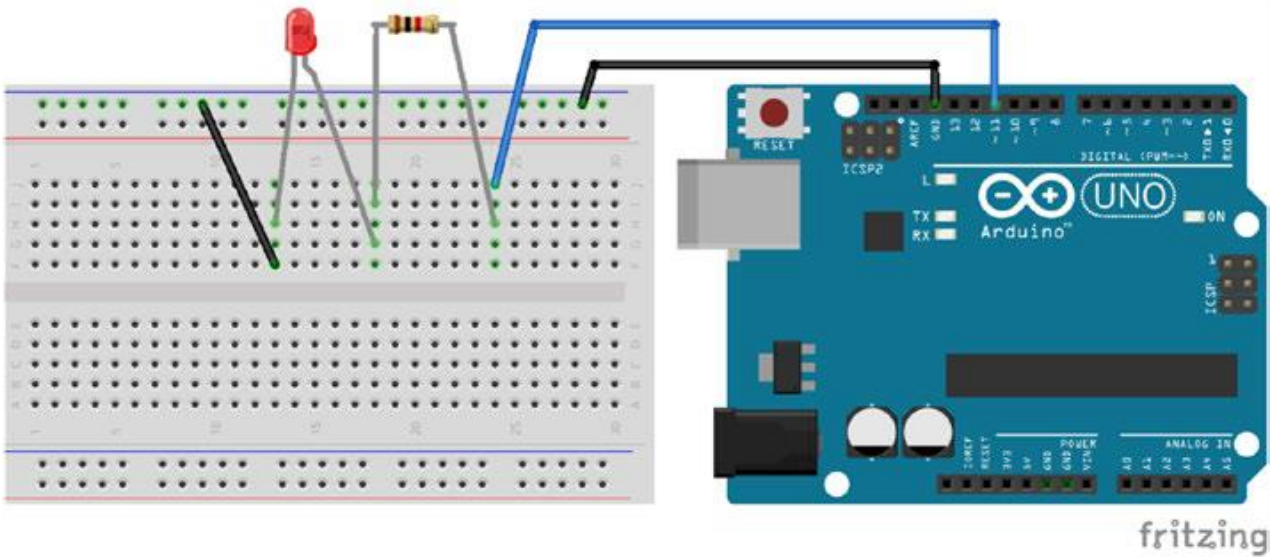
Program:

```
constint led = 2;
void setup() {
pinMode(led, OUTPUT);
}
void loop()
{
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

OUTPUT:

Wait a few seconds while it uploads. Afterward, you should see the LED connected to pin 13 blinking.

(ii) Pulsing LED



LED Fade Exercise Circuit Layout

Result:

Conclusion:

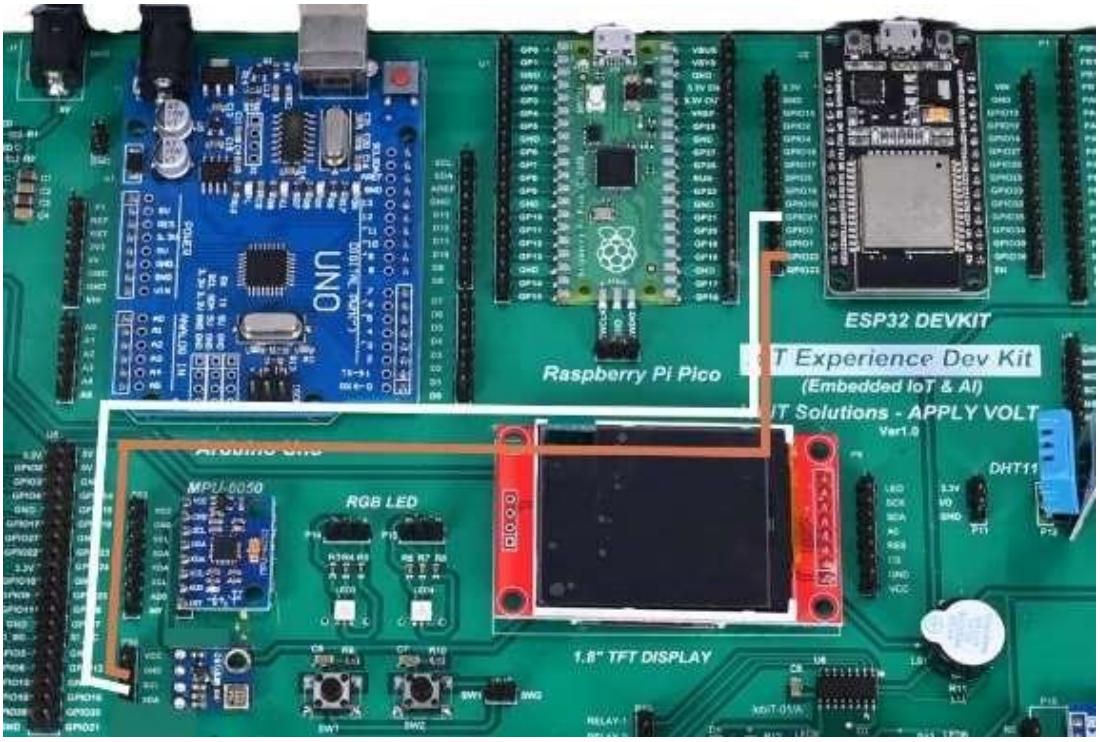
Viva Questions:

Date:

Experiment No: 2

1. Measurement of Temperature and Pressure using ESP32

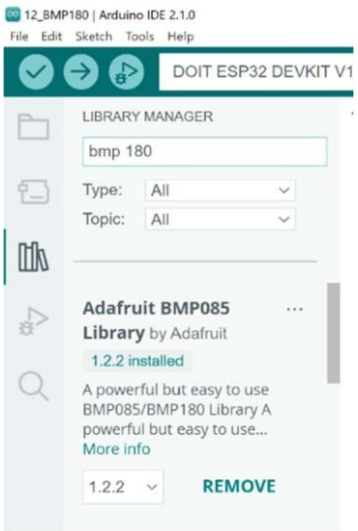
Hardware Connection



Note : Make the connections as per the image shown above

Coding Part

Library Installation : (Download the below Library as shown below)



Code : (Copy below Code and Paste it on your Arduino IDE and upload)

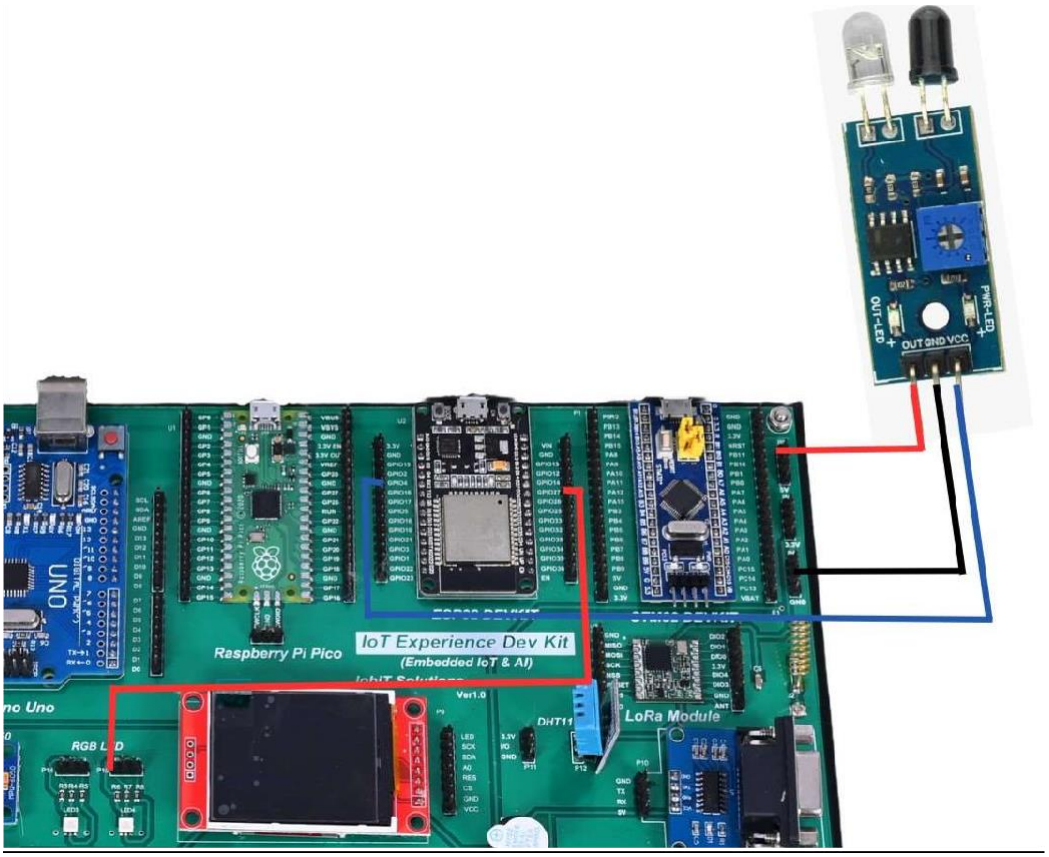
```
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;

void setup() {
  Serial.begin(9600);
  if (!bmp.begin()) {
  Serial.println("Could not find a valid BMP085 sensor, check wiring!");
  while (1) {}
  }
}
  void loop() {
    Serial.print("Temperature = ");
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");

    Serial.print("Pressure = ");
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");
    Serial.println();
    delay(500);
}
```

After successfully Uploading the code, you just Open the serial monitor in the Arduino IDE and see the Output of the Sensor Value.

2. Modules and Sensors Interfacing (IR Sensor, Ultrasonic Sensor ,Soil Moisture Sensor) using ESP32.

2.A).Interfacing IR Sensor and LED with ESP32.

Hardware Connection



IR Sensor OUT pin -- ESP32 GPIO4

LED Pin – ESP32 GPIO 27

Code :

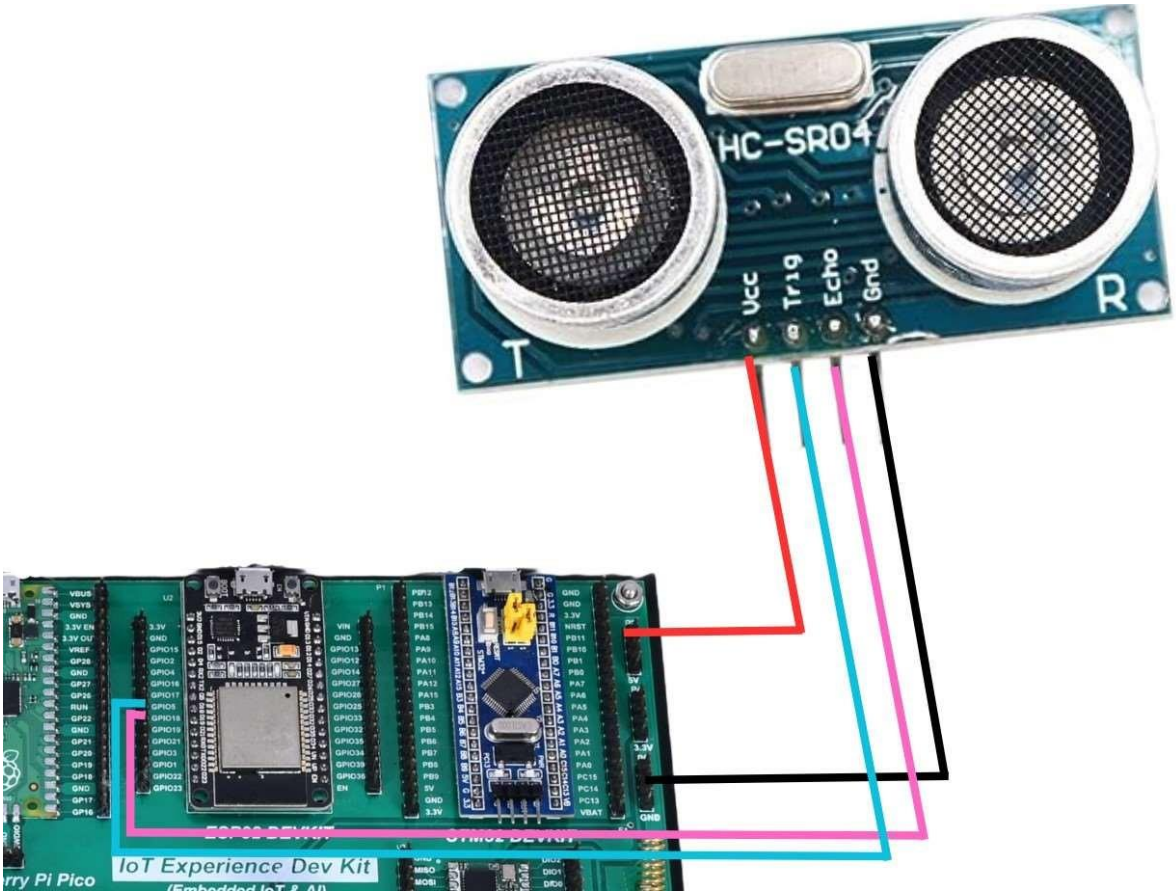
(Copy below Code and Paste it on your Arduino IDE and upload)

```
#define IRSensor = 14; // connect ir sensor to arduino pin 2
#define LED = 27; // conect Led to arduino pin 13

void setup()
{

  pinMode (IRSensor, INPUT); // sensor pin INPUT
  pinMode (LED, OUTPUT); // Led pin OUTPUT
  Serial.begin(9600);
}

void loop()
{
  bool statusSensor = digitalRead (IRSensor);

  if (statusSensor == 1){
```

```
  digitalWrite(LED, LOW); // LED LOW
  Serial.println("Obstable Not Detected, LED OFF");
 }
 else
 {
  digitalWrite(LED, HIGH); // LED High
  Serial.println("Obstable Detected, LED ON");
 }

}
```

2.B).Interfacing Ultrasonic sensor with ESP32.

Hardware Connection



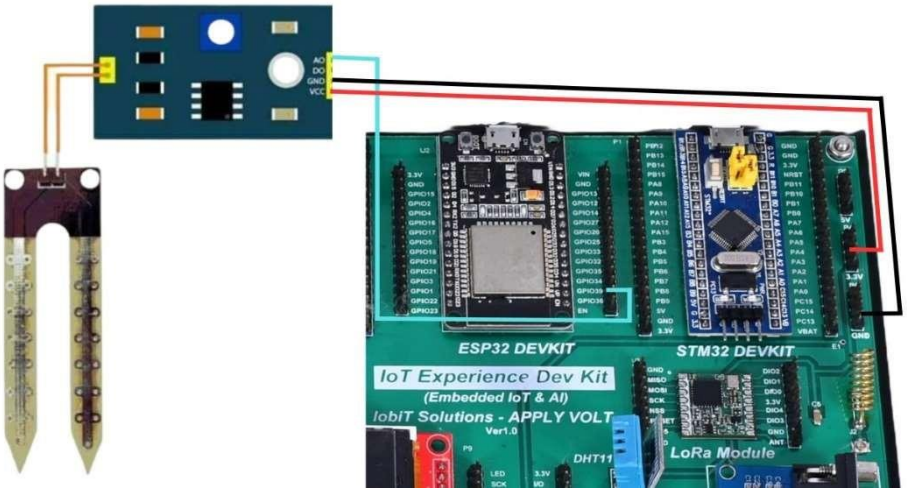
Code :

(Copy below Code and Paste it on your Arduino IDE and upload)

```
const int trigPin = 5;
const int echoPin = 18;

void setup() {
  Serial.begin(9600); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034/2;
  Serial.print("Distance :     cm");
  Serial.println(distance);
  delay(1000);
}
```

2.C).Interfacing Soil Moisture sensor with ESP32.

Hardware Connection



Measuring soil moisture in terms of percentage.

Here, the analog output of the soil moisture sensor is processed using ADC. The moisture content in terms of percentage is displayed on the serial monitor.

The output of the soil moisture sensor changes in the range of ADC value from 0 to 4095.

Code :

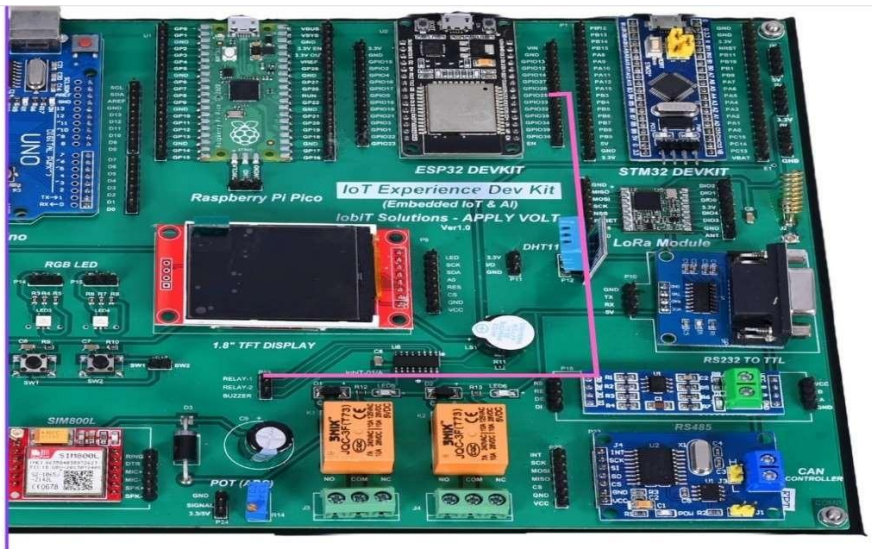
(Copy below Code and Paste it on your Arduino IDE and upload)

```
int _moisture,sensor_analog;
const int sensor_pin = A0;  // Soil moisture sensor O/P pin

void setup(void){
  Serial.begin(115200);     //Set the baudrate to 115200
}

void loop(void){
  sensor_analog = analogRead(sensor_pin);
  _moisture = ( 100 - ( (sensor_analog/4095.00) * 100 ) );
  Serial.print("Moisture = ");
  Serial.print(_moisture); //Print Temperature on the serial window
  Serial.println("%");
  delay(1000);            //Wait for 1000mS
}
```

3.Modules and Actuators Interfacing (Relay, Motor, Buzzer) using ESP32

3.A).Interfacing Relay with ESP32

Hardware Connection

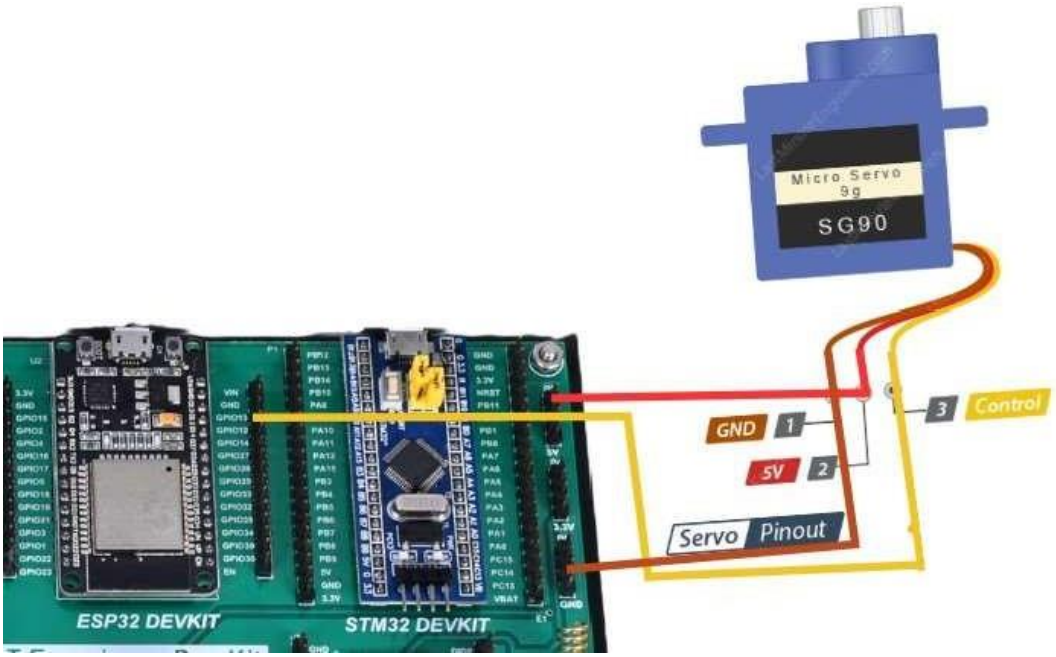


```
#define RELAY1 25

void setup() {
  // put your setup code here, to run once:
  pinMode(RELAY1, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(RELAY1, HIGH);    // turn the RELAY1 on (HIGH is the voltage level)
  Serial.println(" RELAY1 ON ");
  delay(2000);                   // wait for a second
}
```

3.B) Interfacing Motor with ESP32.

Hardware Connection

In our examples we'll connect the signal wire to GPIO 13. So, you can follow the next schematic diagram to wire your servo motor.



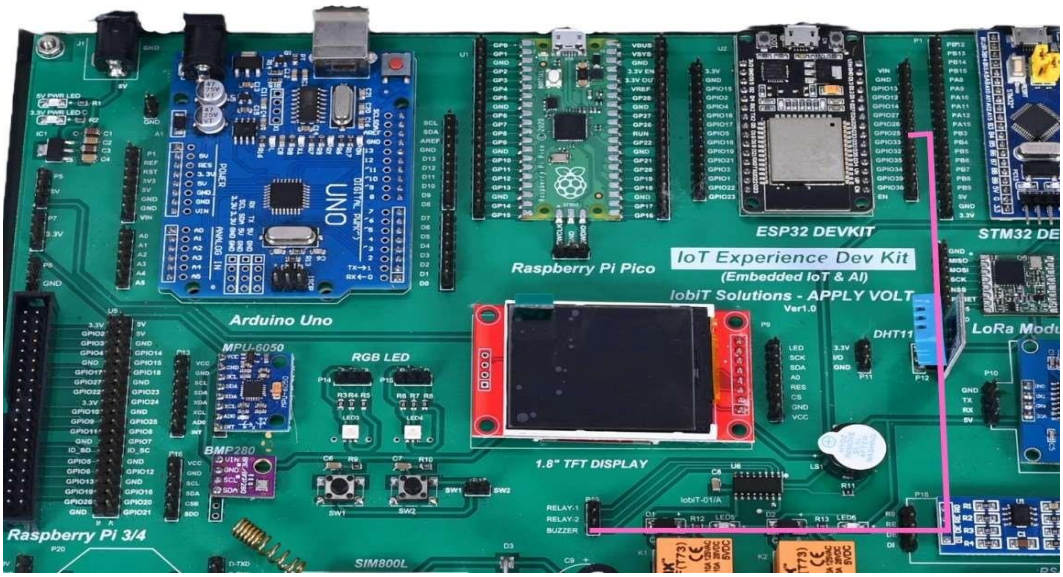
```
#include <ESP32Servo.h>

Servo myServo;
int servoPin=13;

void setup() {
  // put your setup code here, to run once:
  myServo.attach(servoPin);
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
 if(Serial.available()){
   int angle = Serial.parseInt();
   myServo.write(angle);

 }
 delay(20);
}
```

Select BaudRate as 115200

Enter the Number of Degrees to which the motor needs to rotate in Serial Monitor.

3.C) Interfacing Buzzer with ESP32.

Hardware Connection



```
#define BUZZER 13

void setup() {
  // put your setup code here, to run once:
  pinMode(BUZZER, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(BUZZER,HIGH);     // turn the RELAY1 on (HIGH is the voltage level)
  Serial.println(" BUZZER ON ");
  delay(1000);
  digitalWrite(BUZZER,LOW);     // turn the RELAY1 on (HIGH is the voltage level)
  Serial.println(" BUZZER OFF ");
  delay(1000);                  // wait for a second
}
```

Result:

Conclusion:

Viva Questions:

Date:

Experiment No: 3

1. Demonstration of MQTT Communication

MQTT Protocol

The MQ Telemetry Transport (MQTT) is a protocol for publish and subscribe style messaging. It was originally invented by IBM as part of the MQSeries family of products but since has become an industry standard governed by the Oasis standards group. The latest specification version is 3.1.1.

Being Pub/Sub, this means that there is a broker (an MQTT Broker) to which subscribers can register their subscriptions and publishers can submit their publications. Publications and subscriptions agree on the topics to be used to link the messages together. A client can be a publisher, a subscriber or both.

The value of MQTT is that it can be used to deliver data from an application running on one machine to an application running on another. Immediately we seem to see an overlap between MQTT and REST calls but there are some major differences. In a REST environment, when you form a connection from a client to a server, the server must be available in order for the client to deliver the data. With MQTT that is not necessarily the case. The client can publish a message which can then be held by the broker until such time as the receiving application comes on-line to retrieve it.

This is a store and forward mechanism. Every published message must have a topic associated with it that is used to determine which subscribers would be interested in receiving a copy.

The structure of a topic is broken into topic levels separated by a "/". Subscribers can include wild cards in their topic selections of copies of messages that they would like to receive:

• + – Single topic level wild-card

eg. a/+/c

would subscribe to a/b/c and a/x/c.

• # – Multi topic level wild-card

eg. a/#

would subscribe to a/.<anything>

MQTT is commonly implemented on top of TCP/IP. Clients connect to the broker (not to each other) over a TCP connection. There is a quality of service requested by a client. This is encoded in the QoS field:

• QoS=0 – Send at most once. This can lose messages. At most once means perhaps never.

• QoS=1 – Send at least once. This means that the message will be delivered. Saying this

another way, a message will not be discarded or lost. However, duplicates can arrive … i.e.

the message can be delivered twice or more.

• QoS=2 – Send exactly once. This means that the message will not be lost and will be

delivered once and once only. MQTT also has the capability to buffer messages for subsequent delivery. For example, if a client subscriber is not currently connected, a message can be queued or stored for delivery to the client when it eventually re-connects. We call a client that is not connected an off-line client. For a subscription, we have the choice to deliver all the queued messages for a client or just the last message. To understand the difference, we can imagine a published message that says "I sold your stock for price" … we want all such messages sent to the client because they are all of interest.

"Experiencing the Value of Technology"

However if we think of a published message of "Today's forecast is sunny and warm" then there may be no need for old messages and only the current weather forecast is of interest to us. During publishing we can declare that a message is eligible for retention and this is called the "retained message". When a client subscribes, it can ask to receive the last retained message immediately … so even if a subscription takes place after a previous publication, it can still receive data immediately.

Clients make their status known to the broker so the broker can tell if a client is connected. This is achieved via a keep-alive/heartbeat. When forming a connection to a broker, the client provides a keep-alive interval (in seconds). If the broker hasn't received a message from the client in this

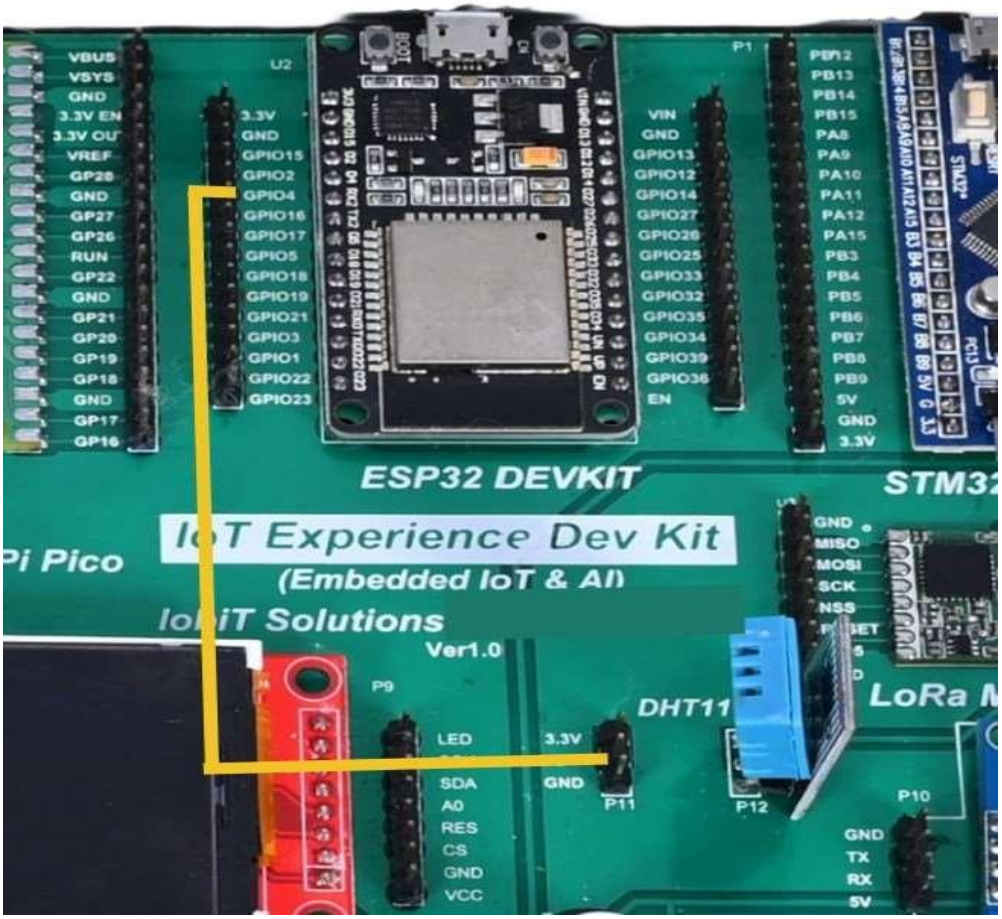
interval then the broker can disconnect the client assuming it to have been lost/disconnected. If the keep-alive interval is set to 0, then there will be no validation from the broker.

If a client connection is lost because of a network disconnection, the broker can detect that occurrence. This is where we get morbid. We define this as the client having "died". In the real world, when someone dies, there may be a last "will and testament" which are the desired instructions of what the person wanted to happen when they die. MQTT has a similar concept. A client can register a message to be published in the event of the clients death. This is remembered by the broker and in the event of the client dyeing, the broker will perform the role of the attorney and publish the last registered "will and testament" message on behalf of the deceased client.

The default port number for an MQTT broker is 1883

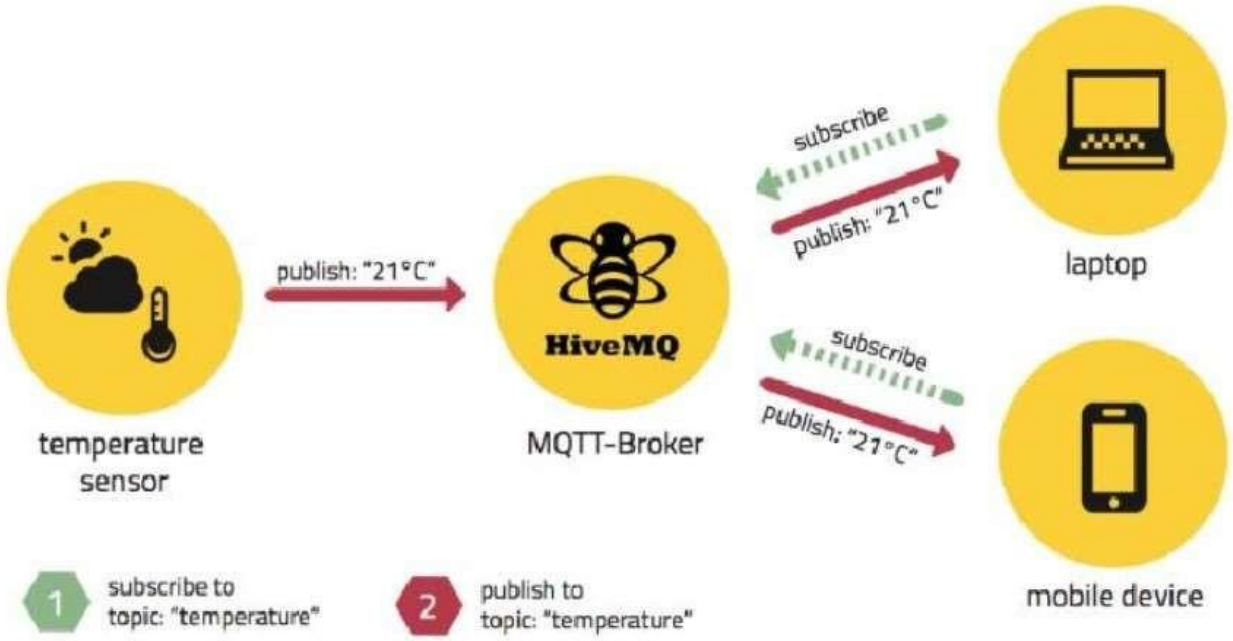
Experience to send the DHT11 data via MQTT protocol and control the Appliance via MQTT protocol

Hardware Connection



Type	Name	Notes
int	cmd	
struct mg_str	payload	The payload of a received message.
uint8_t	connack_ret_code	Used on MG_EV_MQTT_CONNACK.
uint16_t	message_id	Used on MG_EV_MQTT_PUBLISH.
char *	topic	The topic on which the message was received.

Library Installation :



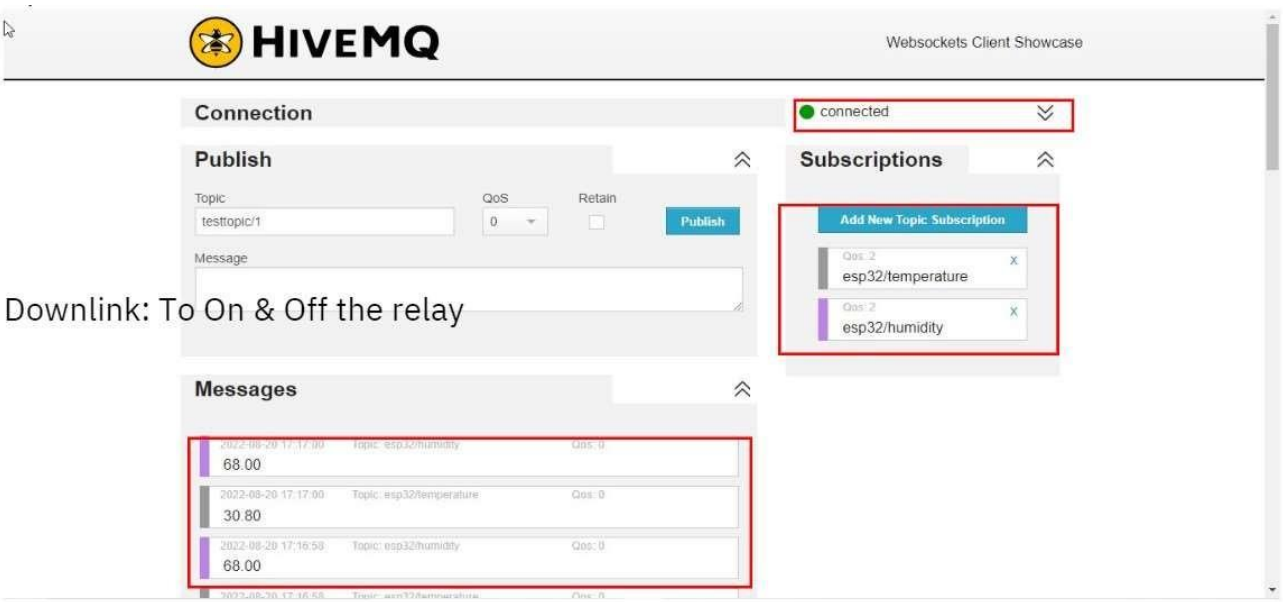
Code :

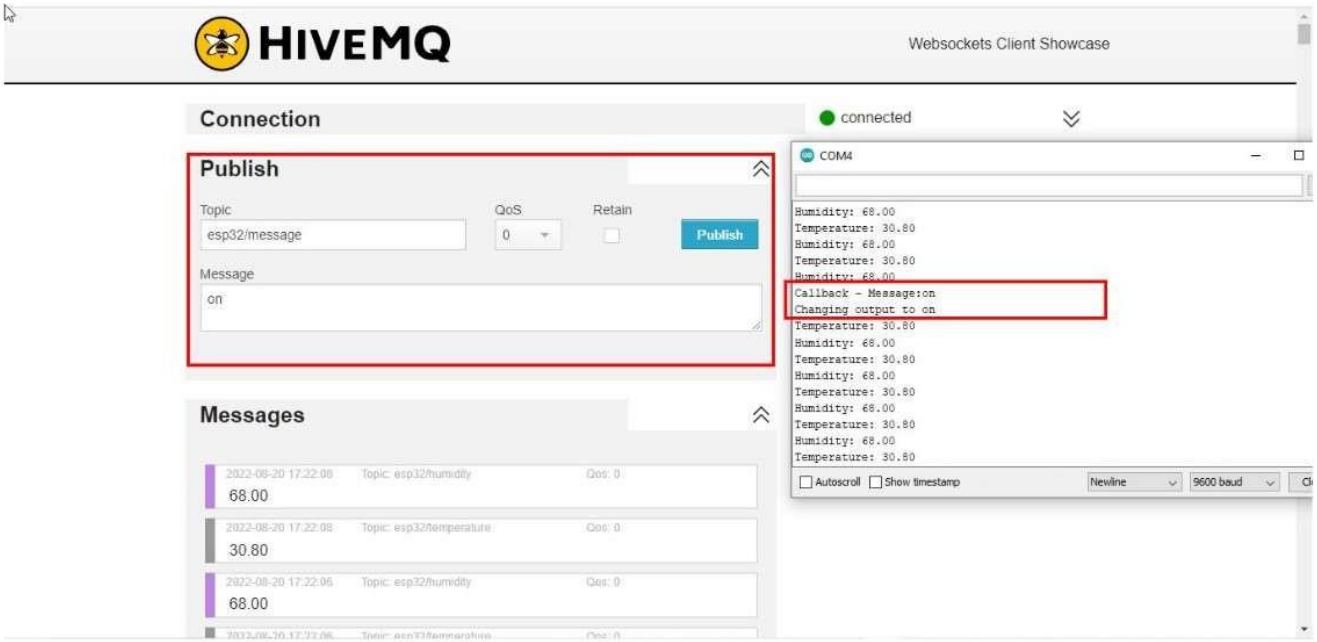
```
#include <Arduino.h>
#include  <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);
#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
const char* ssid = "Types your SSID";
const char* password = "Your Password";
char *mqttServer = "broker.hivemq.com";
int mqttPort = 1883;
void setupMQTT() {
mqttClient.setServer(mqttServer, mqttPort);
mqttClient.setCallback(callback);
}
void reconnect() {
Serial.println("Connecting to MQTT Broker...");
while (!mqttClient.connected()) {
Serial.println("Reconnecting to MQTT Broker..");
String clientId = "ESP32Client-";
clientId += String(random(0xffff), HEX);
if (mqttClient.connect(clientId.c_str())) {
Serial.println("Connected.");
// subscribe to topic
mqttClient.subscribe("esp32/message");
}
}
}
void setup() {
Serial.begin(9600);
Serial.println("DHT11 test!");
dht.begin();
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("Connected to Wi-Fi");
pinMode(2,  OUTPUT);
setupMQTT();
}
void loop() {
if (!mqttClient.connected())
```

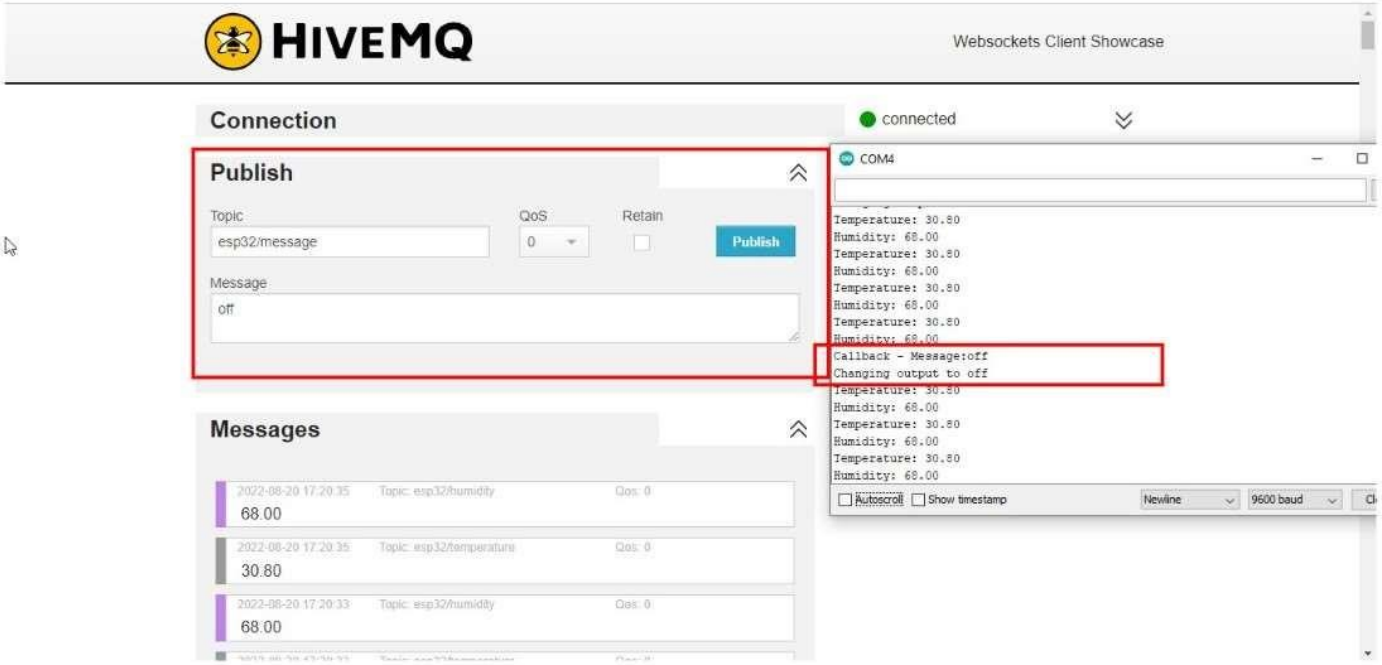
```
reconnect();
mqttClient.loop();
long now = millis();
long previous_time = 0;
if (now - previous_time > 1000) {
previous_time = now;
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
char tempString[8];
dtostrf(t, 1, 2, tempString);
Serial.print("Temperature: ");
Serial.println(tempString);
mqttClient.publish("esp32/temperature", tempString);
char humString[8];
dtostrf(h, 1, 2, humString);
Serial.print("Humidity: ");
Serial.println(humString);
mqttClient.publish("esp32/humidity", humString);
delay(2000);
}
}
void callback(char* topic, byte* message, unsigned int length) {
Serial.print("Callback - ");
Serial.print("Message:");
String messageTemp;
for (int i = 0; i < length; i++) {
Serial.print((char)message[i]);
messageTemp += (char)message[i];
}
Serial.println();
if (String(topic) == "esp32/message") {
Serial.print("Changing output to ");
if(messageTemp == "on"){
Serial.println("on");
digitalWrite(2, HIGH);
}
else if(messageTemp == "off"){
Serial.println("off");
digitalWrite(2, LOW);
}
}
}
```

To Experience the Outcome :

Uplink : To view the Sensor Data Downlink: To On & Off the relay







Result:

Conclusion:

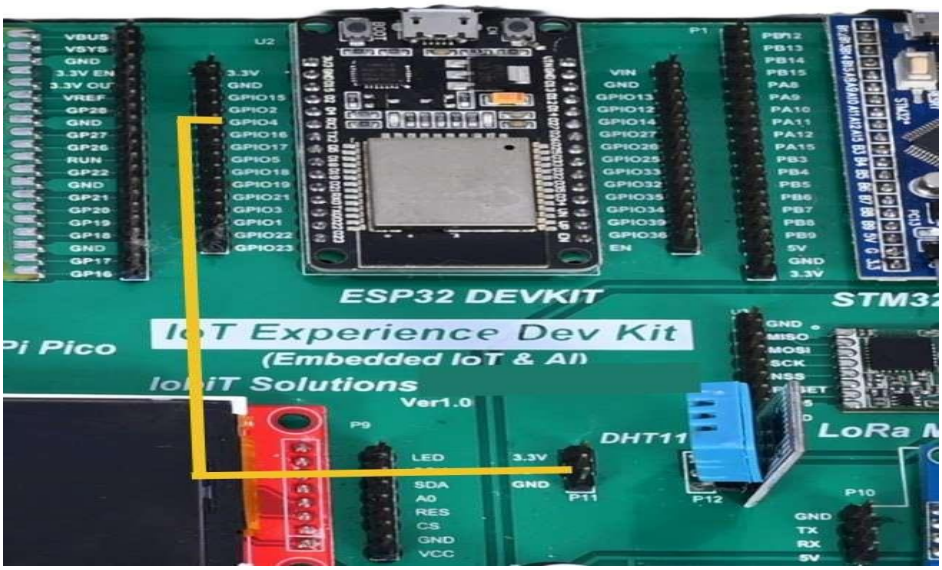
Viva Questions:

Date:

Experiment No: 4

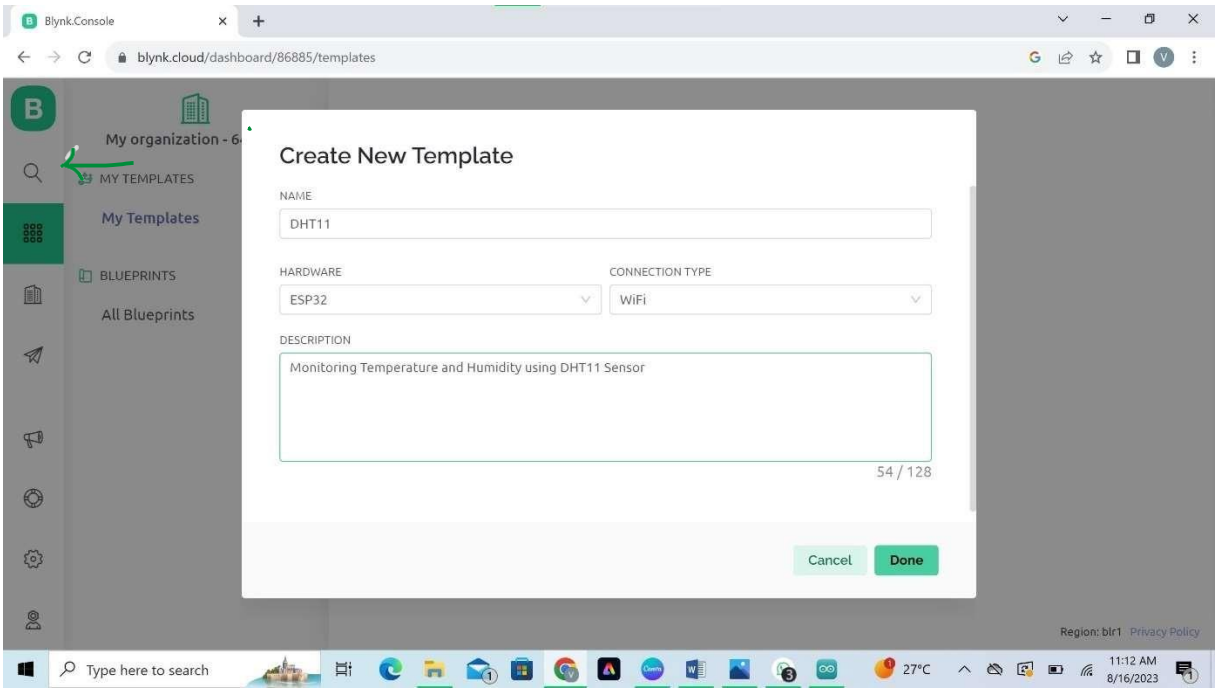
1. Visualization of Diverse Sensors data using Dashboard.

Hardware Connection

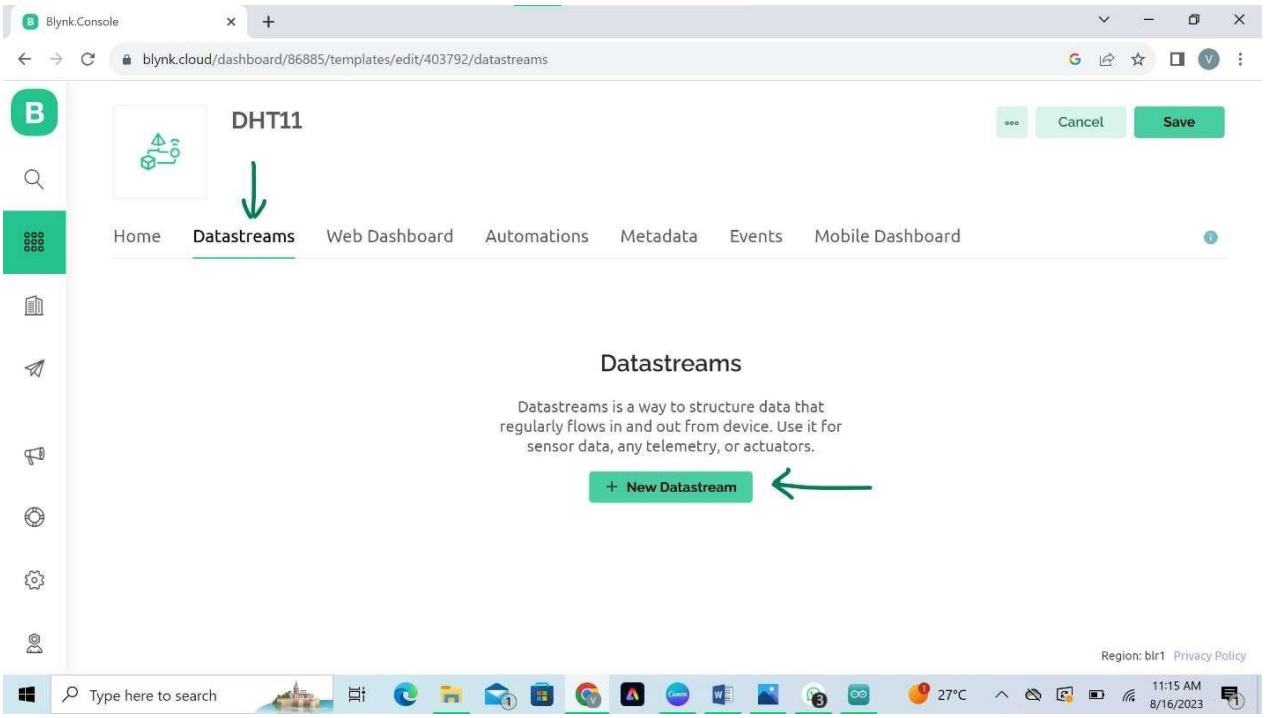


Creating a Blynk IoT Environment :

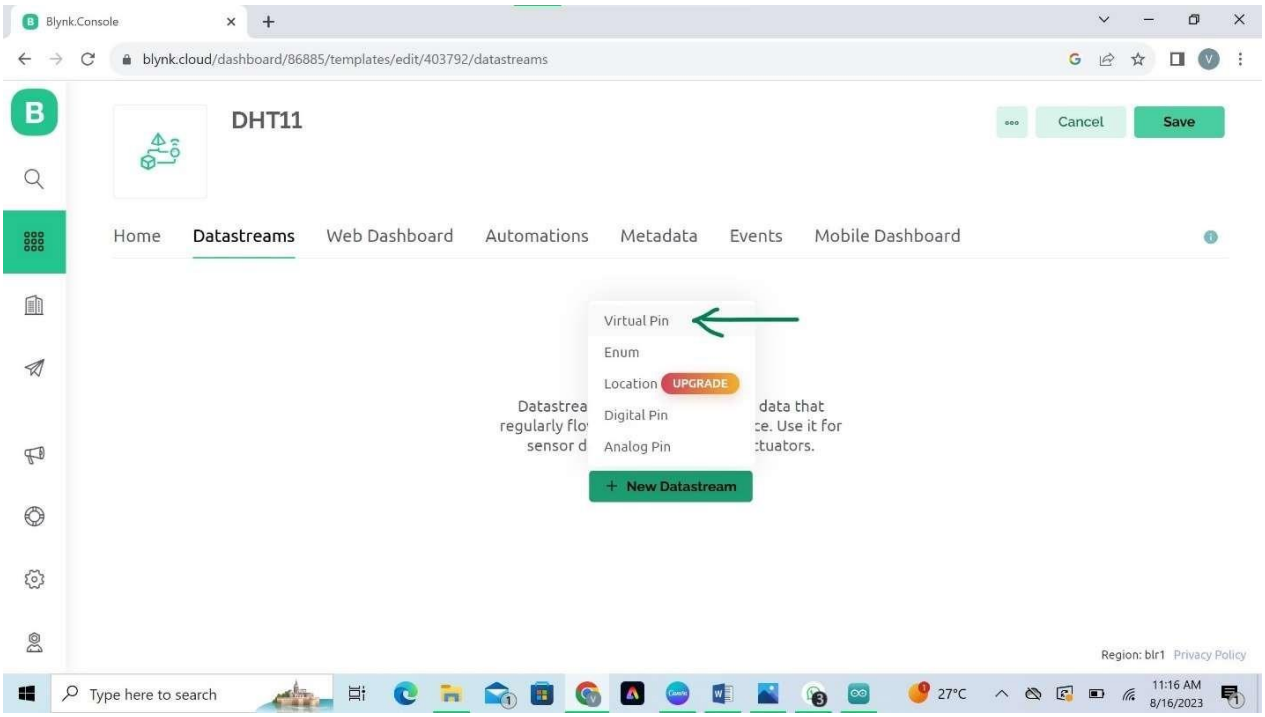
1. Open the URL https://blynk.cloud/
2. Signup in to this platform as shown below.
3. Create Template in Blynk IoT Cloud



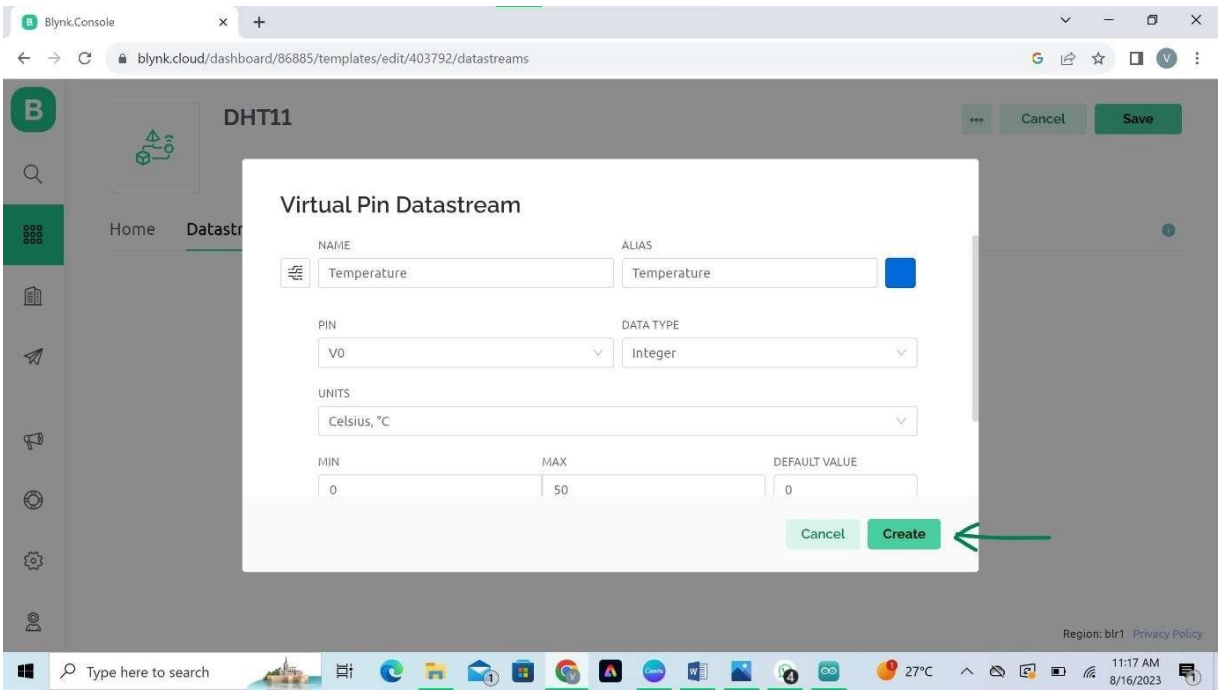
Enter Template Name(DHT11) > Select Hardware(ESP32) > Select Connection (WiFi) >Done



Select Template > Select DataStreams > + New Datastreams



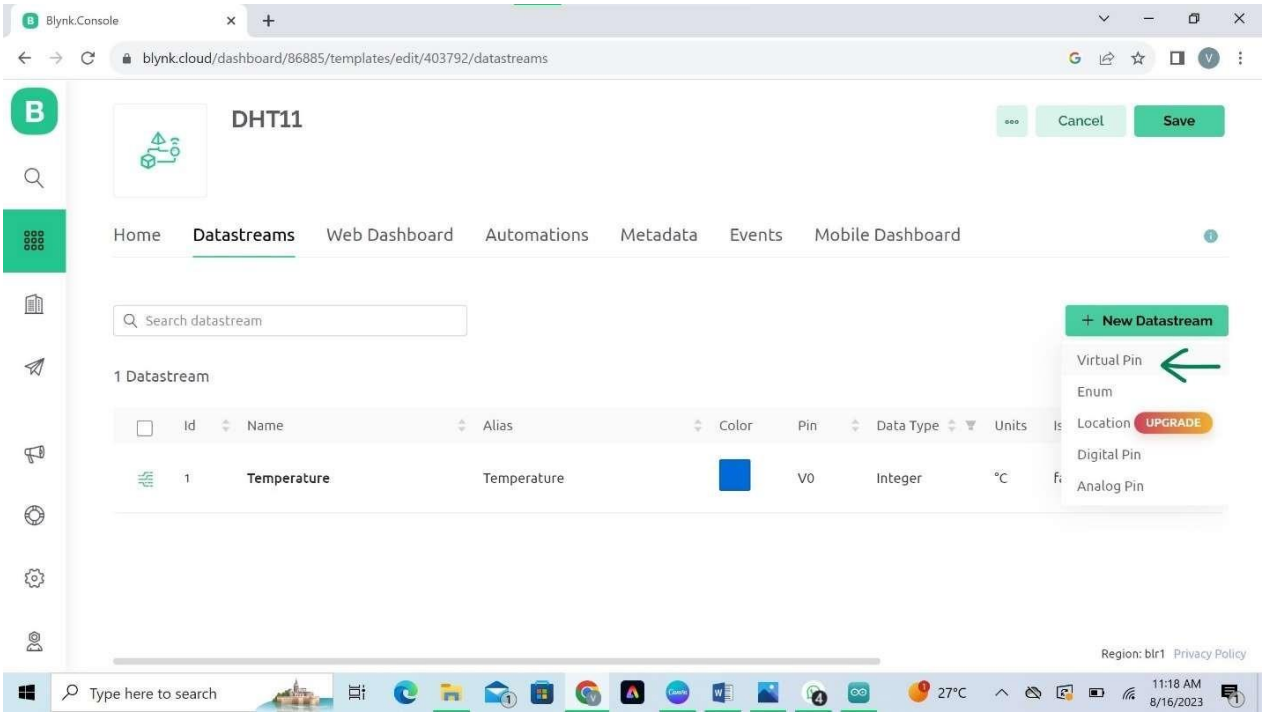
New Datastream > Select Virtual Pin >



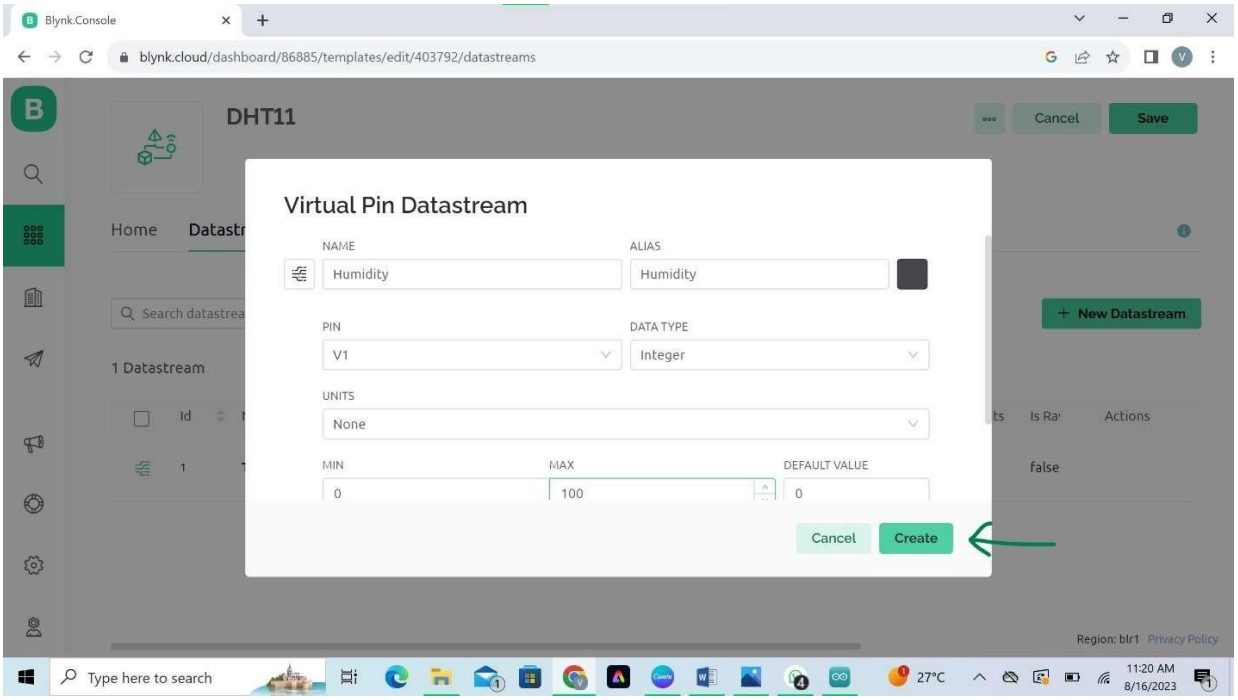
We have to create Two Datastreams because DHT11 Sensor will measure Both the Temperature and Humidity Values.

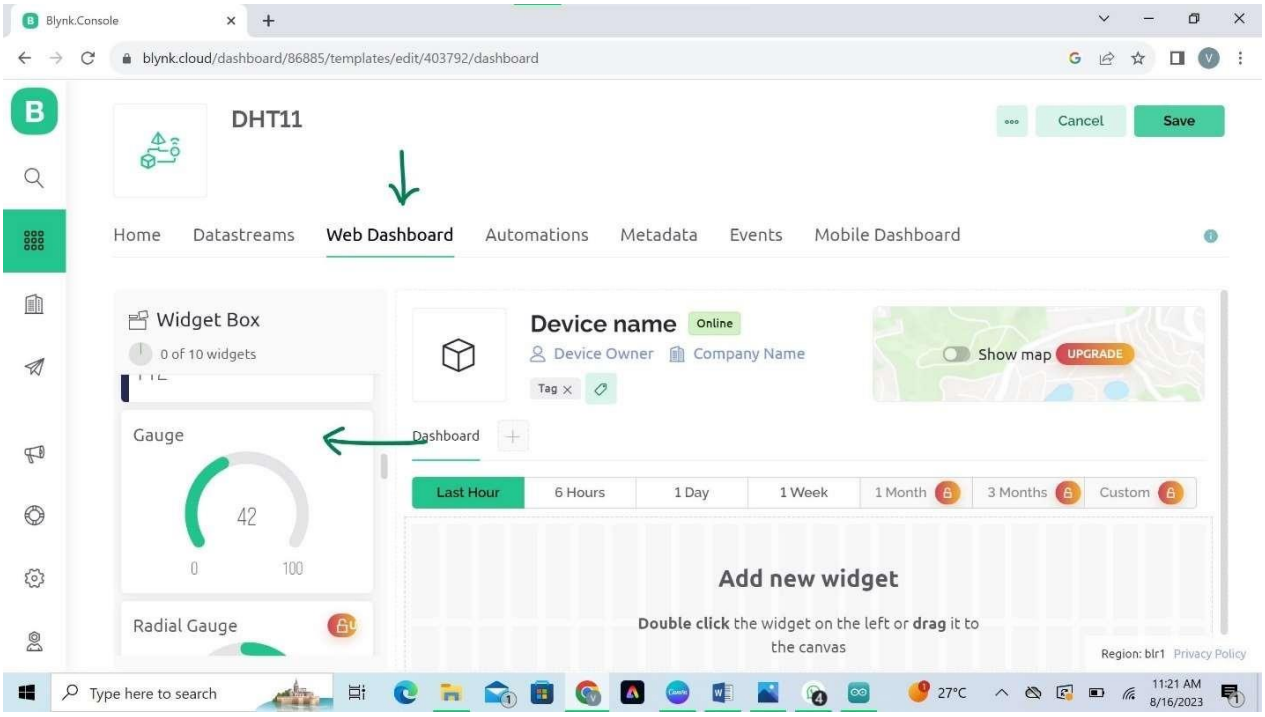
Enter Name(Temperature) > Select Pin (V0) > Select Unit(Celsius) > Select Range (Min & Max) > Click on Create. Now we created Temperature Datastream.

Now we have to Datastream for Humidity.



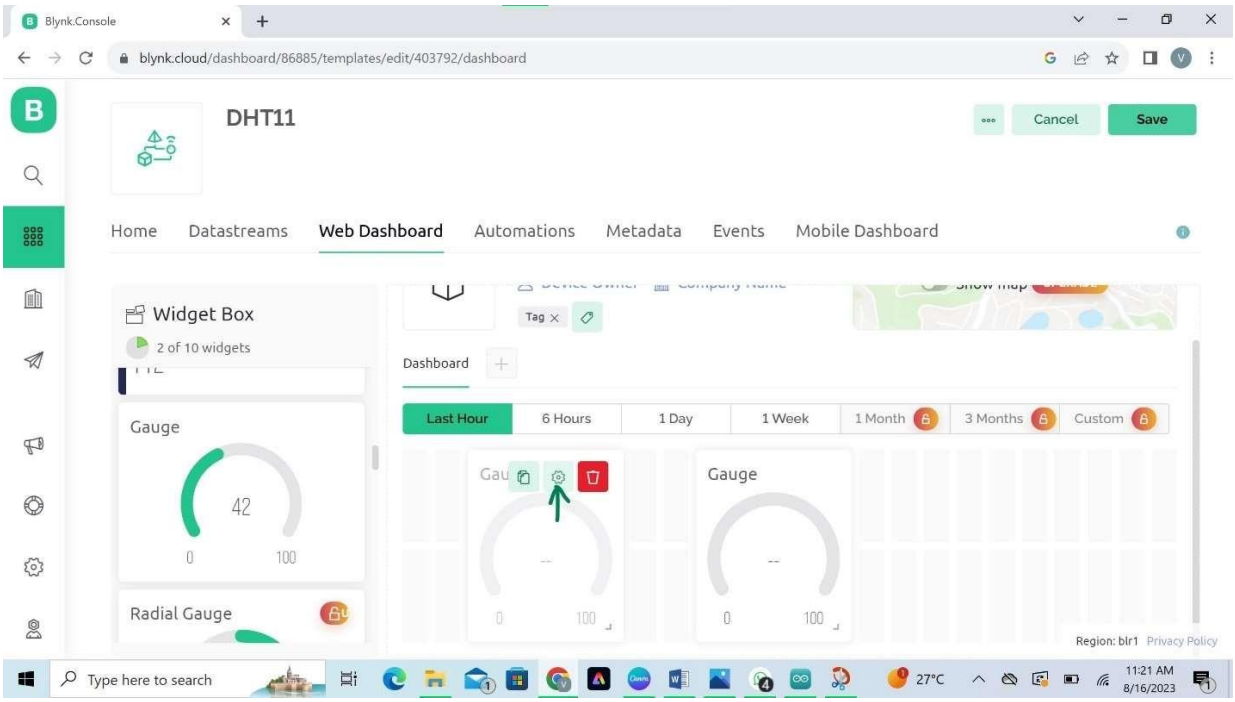
Click on +New Datastream > Select Virtual Pin > Enter Name(Humidity) > Select Pin (V1) > Select Unit(%) > Select Range (Min & Max) > Click on Create. Now we created Humidity DataStream.



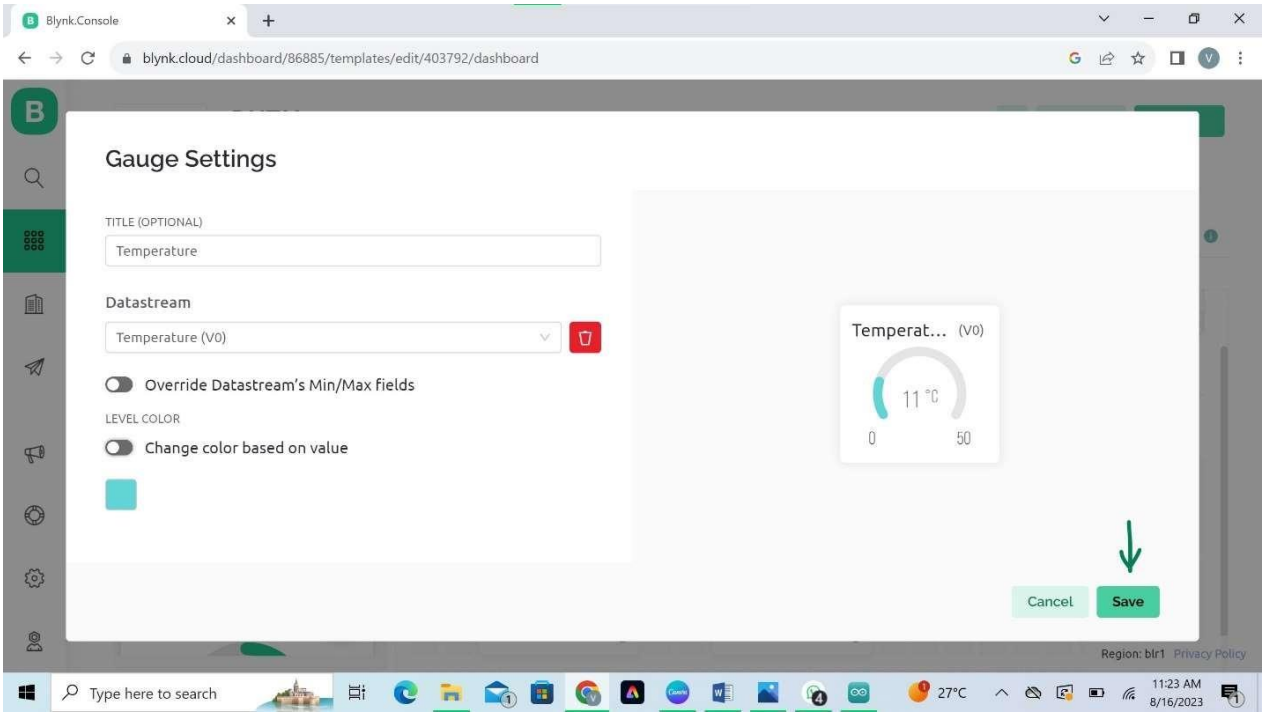


Click on Web Dashboard > Scroll down the Widget Box and Select the Gauge and drag that Gauge Widget to the Dashboard area as shown below.

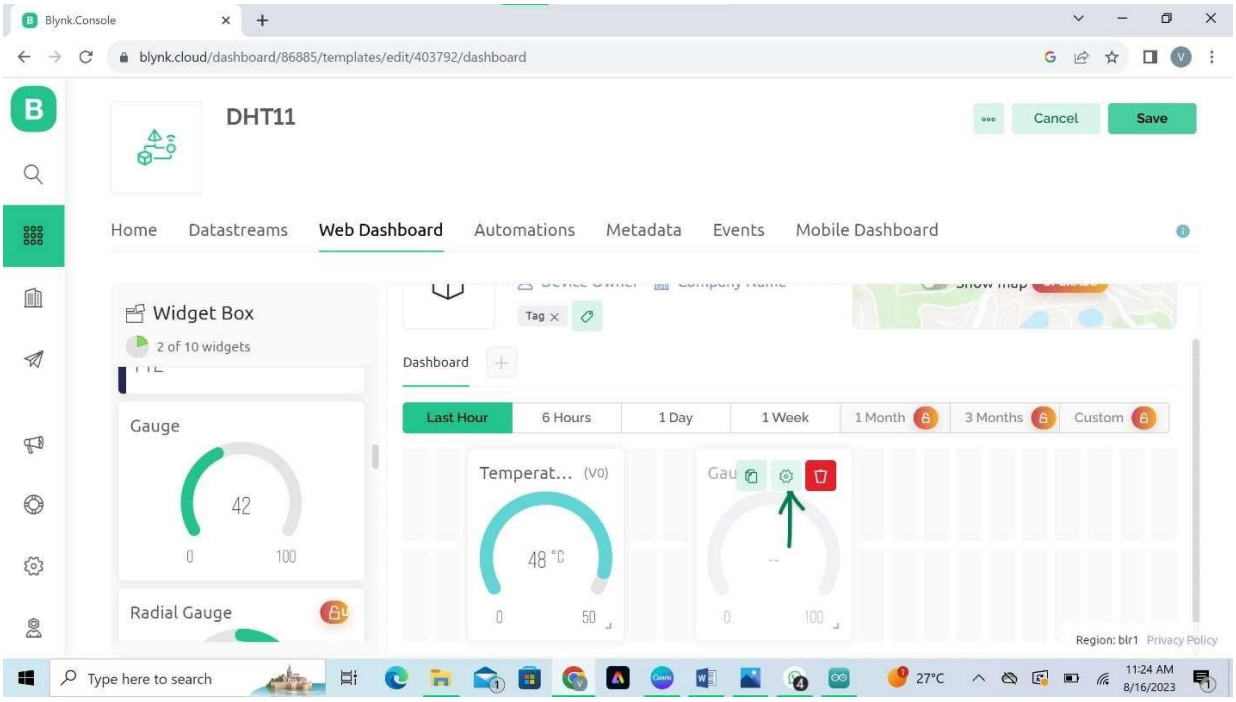
We Need Two Widgets One for Temperature and One for Humidity.



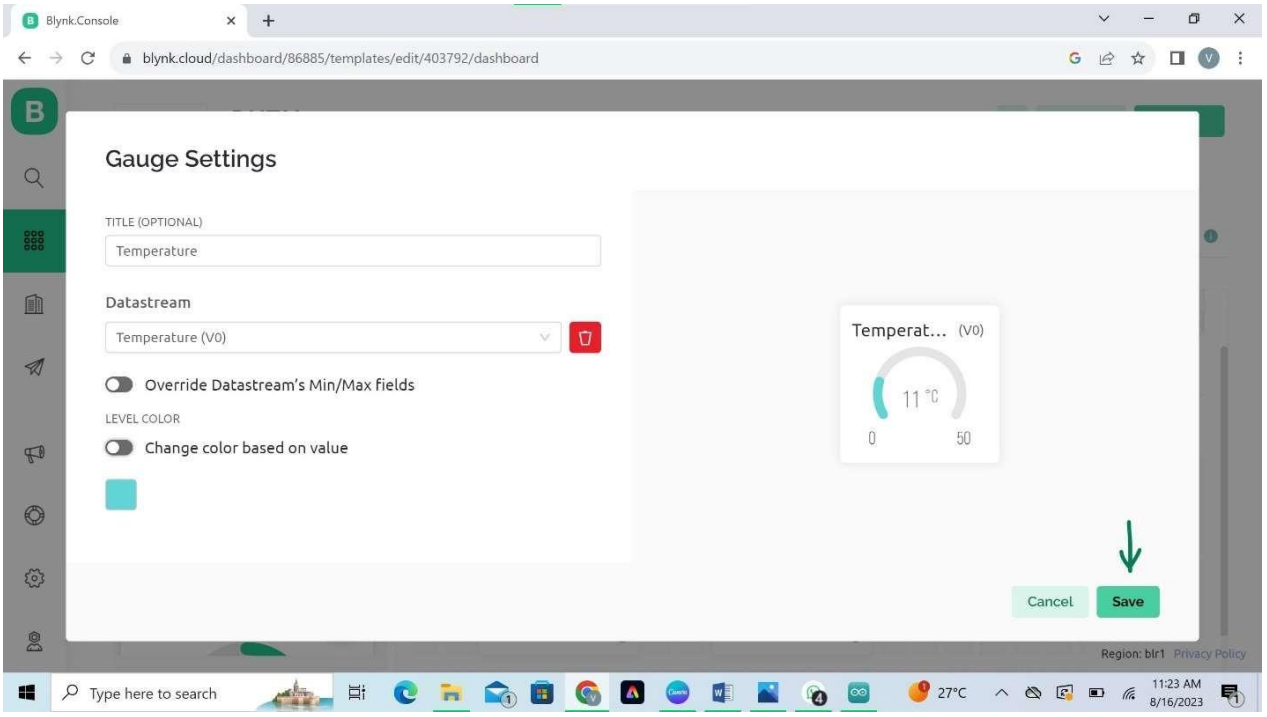
Click on the Settings Icon on the Widget .



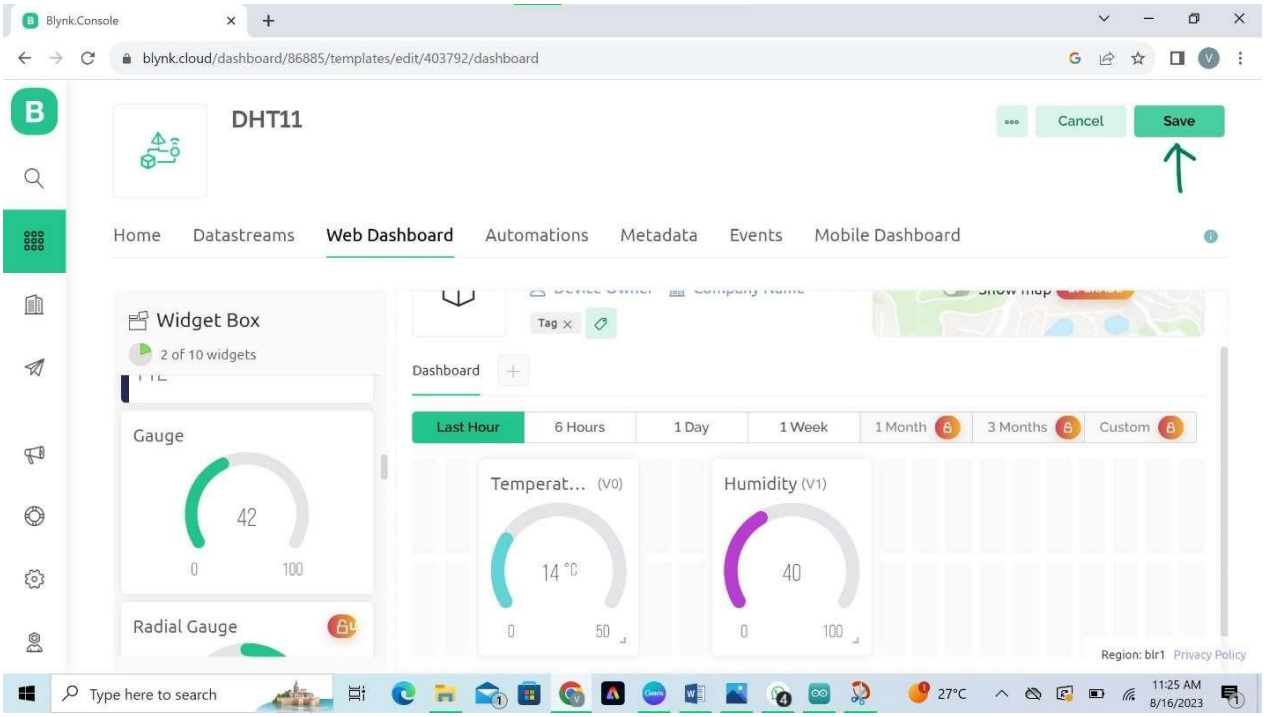
Enter the Title Name (Temperature) > Select Datastream(Temperature) > Save.



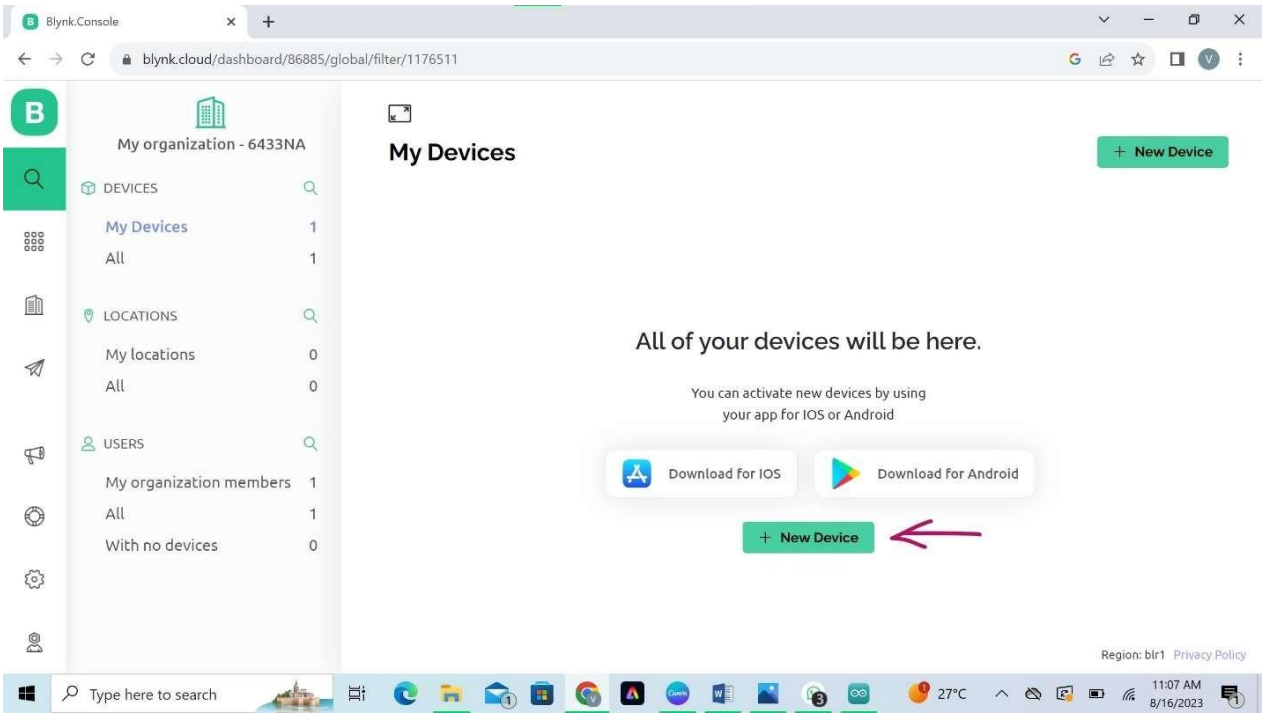
Click on the Settings Icon on the Widget .



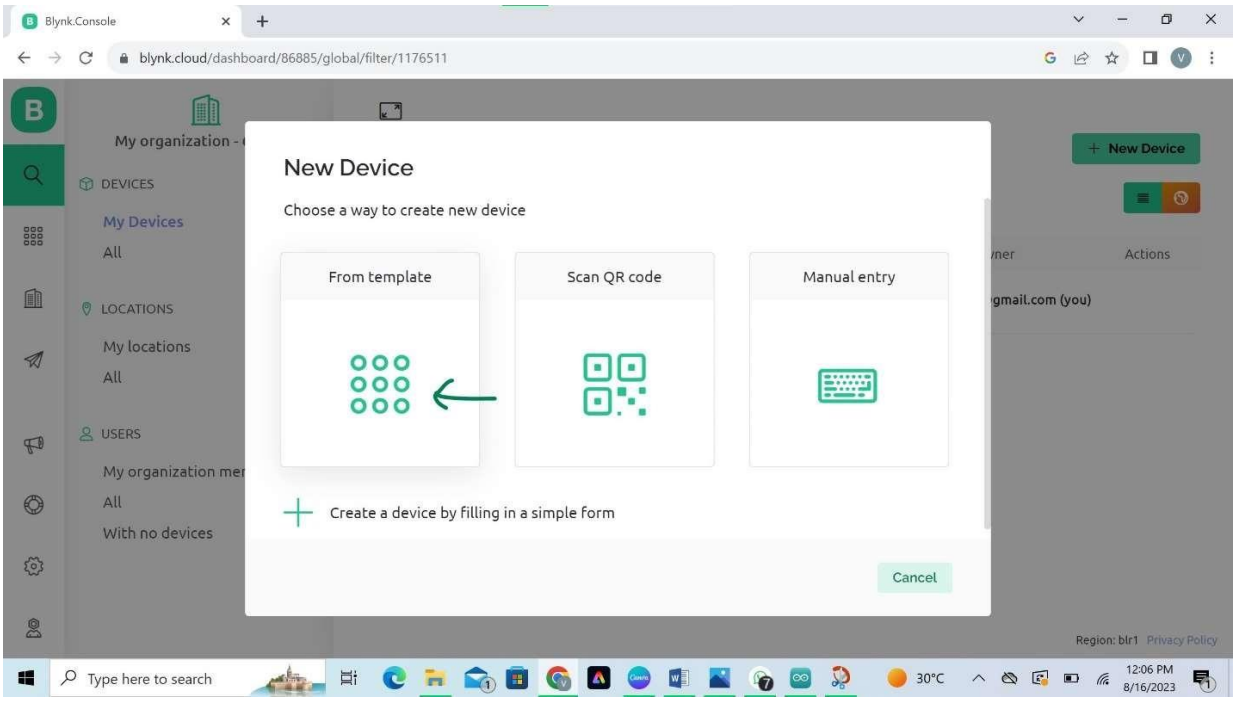
Enter the Title Name (Humidity) > Select Datastream(Humidity) > Save.

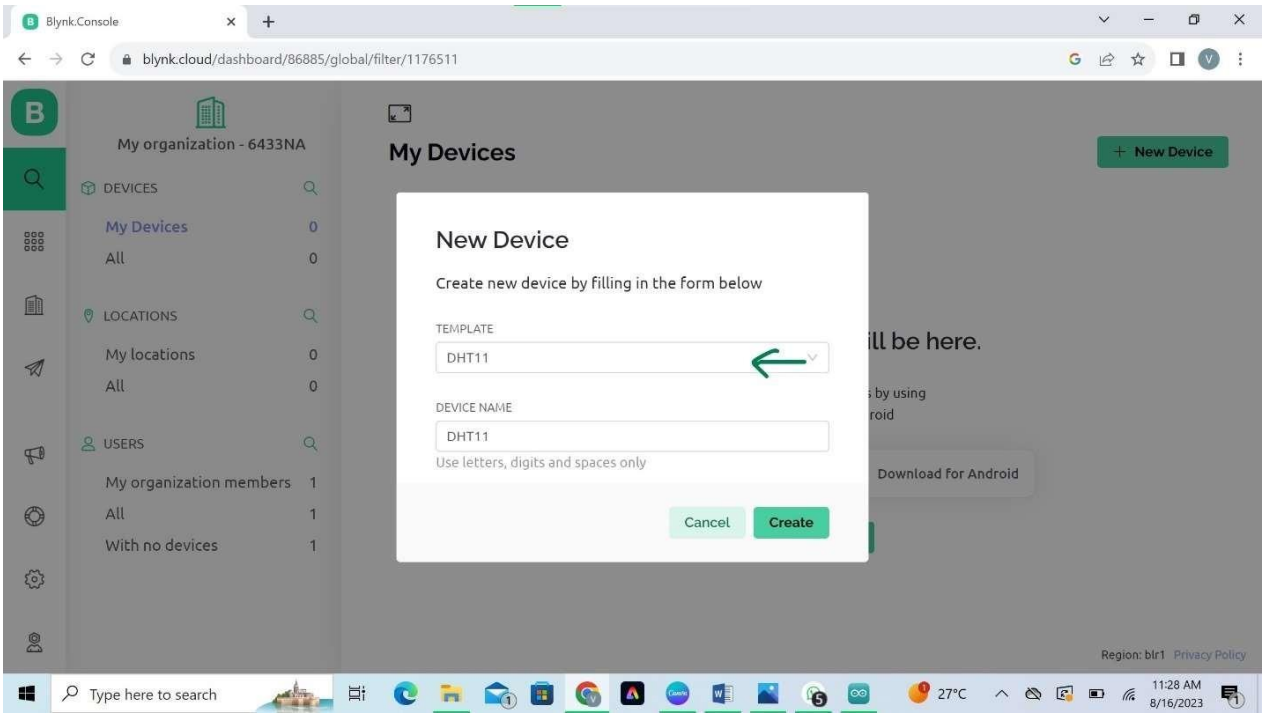


Click on Save.

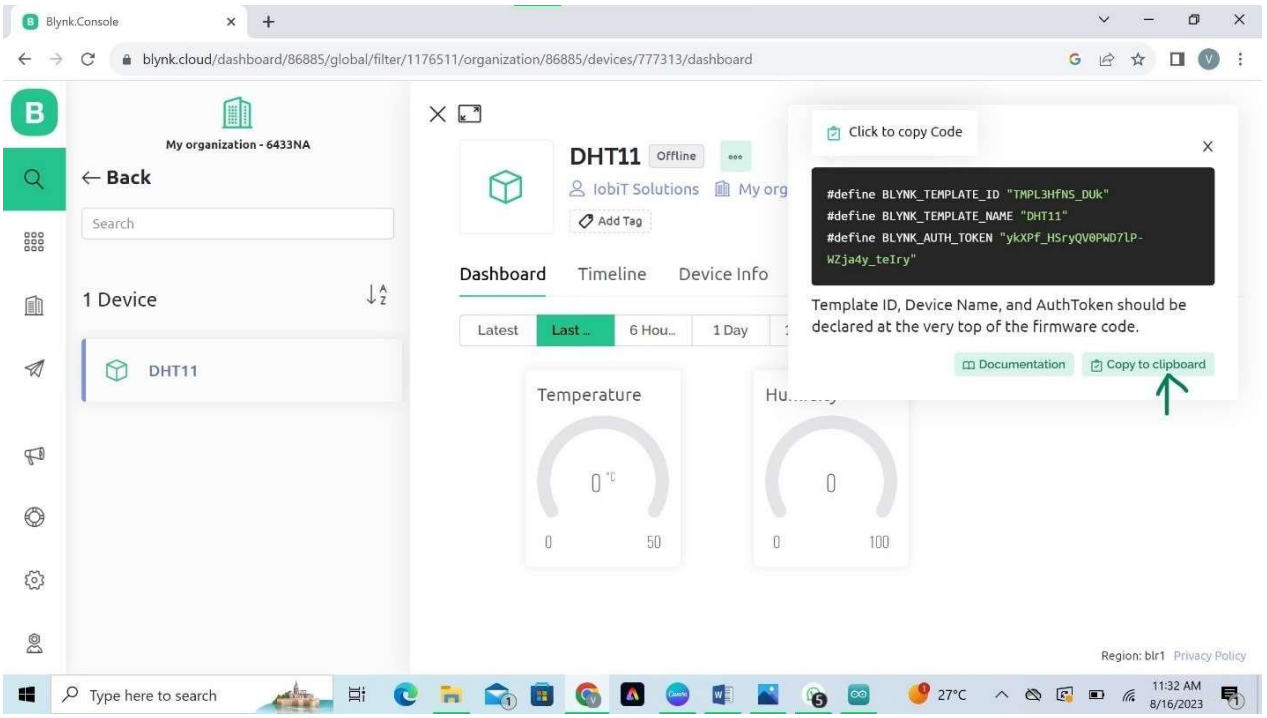


Then Click on Search Icon > Click on +New Device > Click on From template.





Select the Template name (Created template DHT11) > Click on Create.

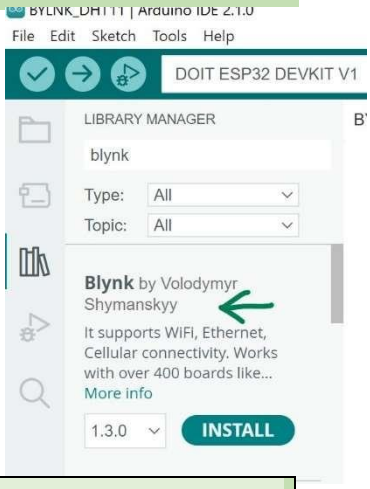


After Creating Device > then Click on that Copy to Clipboard > Later we have to Paste this in Code.

Code :

(Copy below Code and Paste it on your Arduino IDE and upload)

Library Installation



Paste the Code we copied from Blynk Cloud Platform and Replace it on top first three lines of below Code.

15th and 16th line Enter your WiFi Name and Password.

```
#define BLYNK_TEMPLATE_ID "TMPL3HfNS_DUk"
#define BLYNK_TEMPLATE_NAME "DHT11"
#define BLYNK_AUTH_TOKEN "ykXPf_HSryQV0PWD7lP-WZja4y_teIry"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>


#include "DHT.h"

char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "IobiT Solutions"; // type your wifi name
char pass[] = "12345678";  // type your wifi password

#define DHTPIN 4         // Mention the digital pin where you connected
#define DHTTYPE DHT11     // DHT 11
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;

void sendSensor(){
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

    Blynk.virtualWrite(V0, t);
    Blynk.virtualWrite(V1, h);

    Serial.print("Temperature : ");
    Serial.print(t);
```

```
  Serial.print(" Humidity : ");
  Serial.println(h);
 }


void setup(){
 Serial.begin(115200);
 Blynk.begin(auth, ssid, pass);
 dht.begin();
 timer.setInterval(1000L, sendSensor);
}

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

Result:

Conclusion:

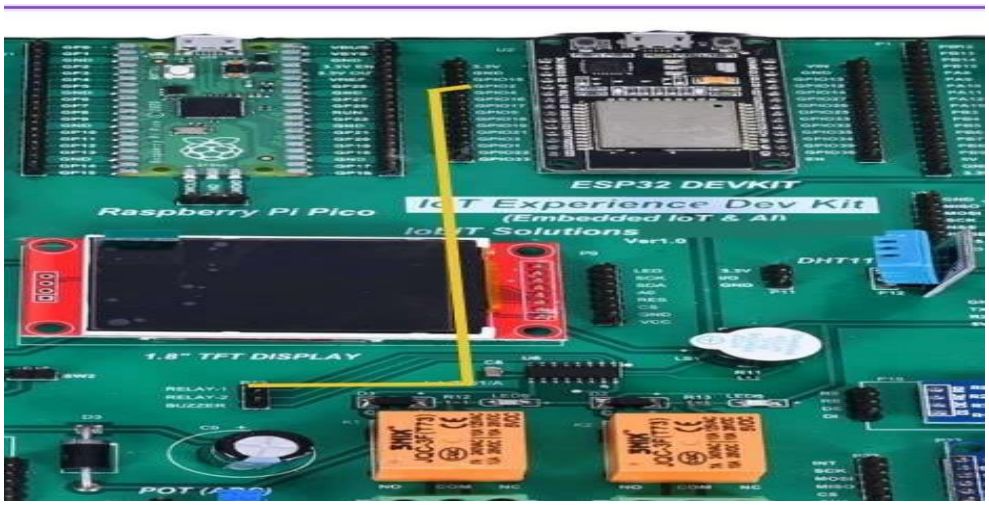
Viva Questions:

Date:

Experiment No: 5

1. Device Control using mobile Apps or through Web pages.

Hardware Connection

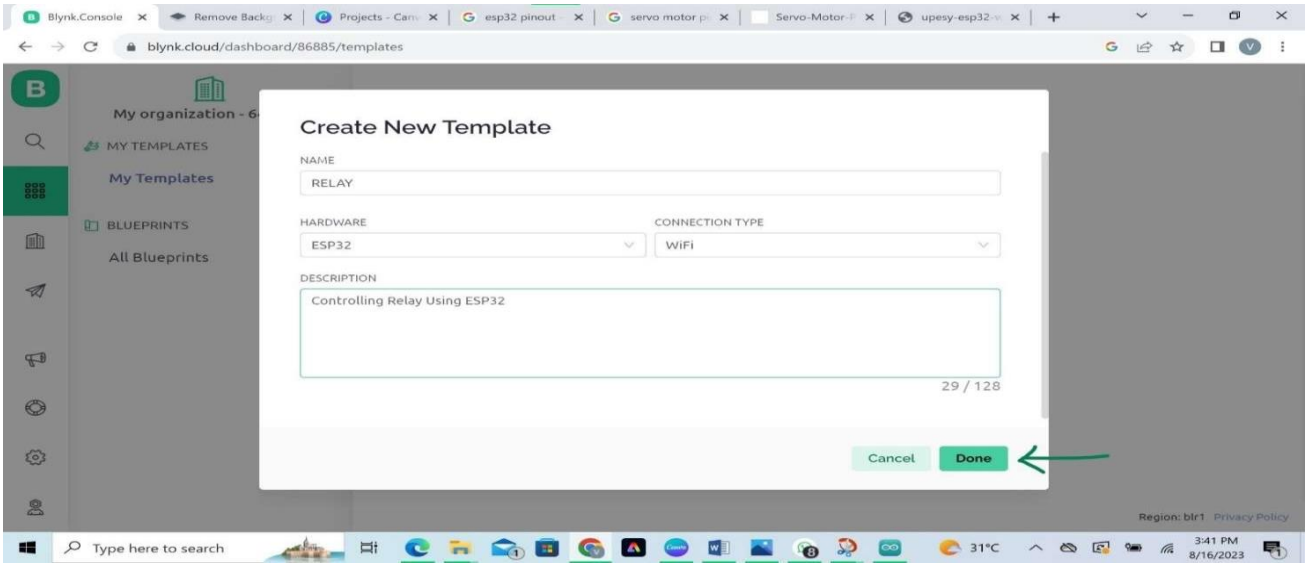


Creating a Blynk IoT Environment :

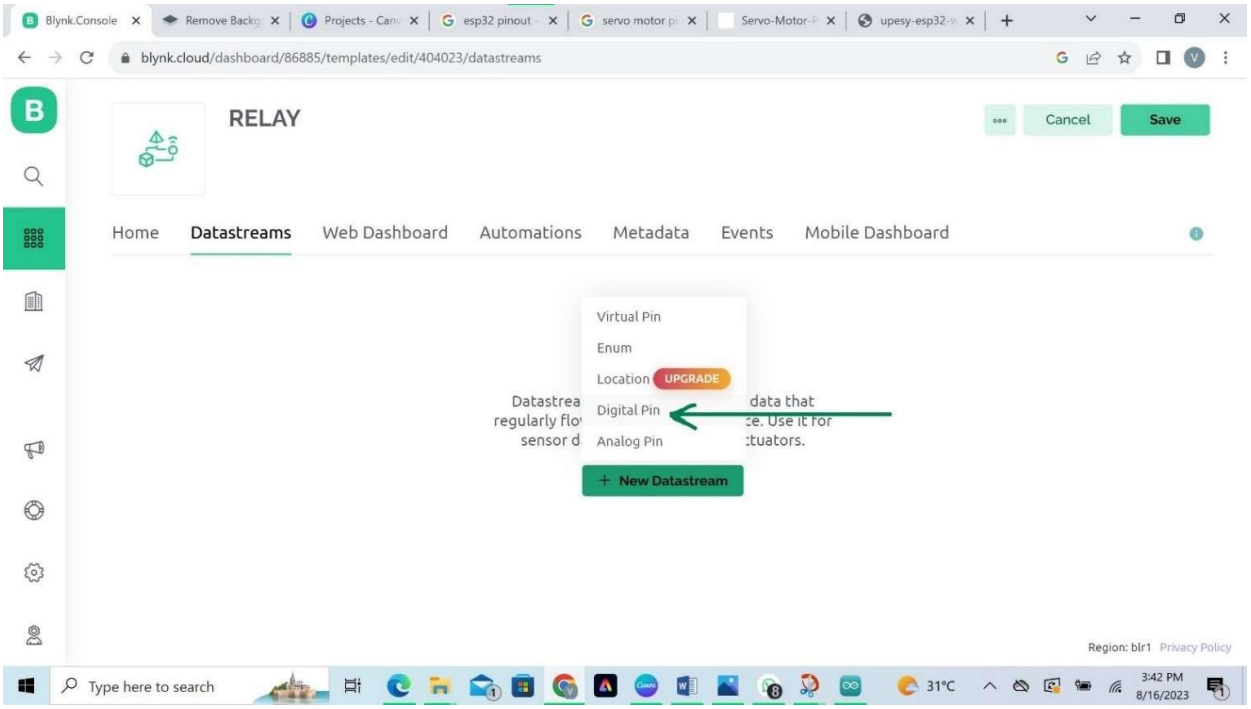
Open the URL https://blynk.cloud/

1. Create Template in Blynk IoT Cloud

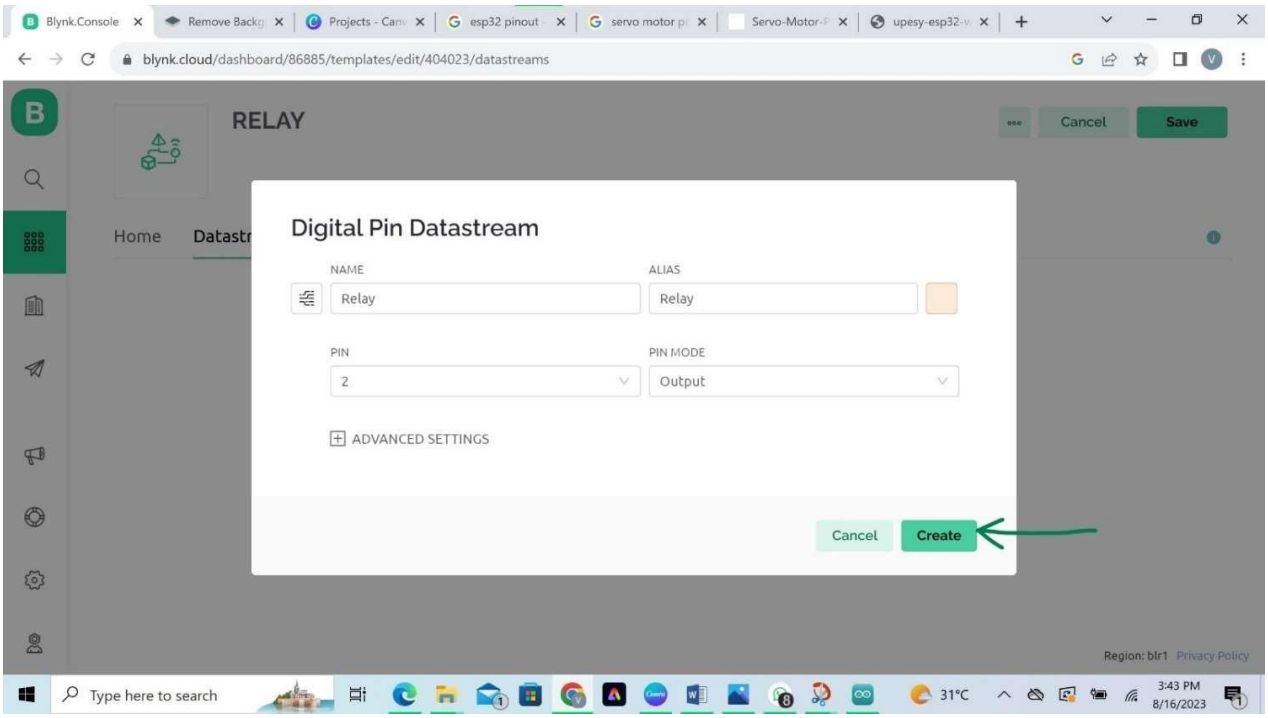
Click on Template > New Template



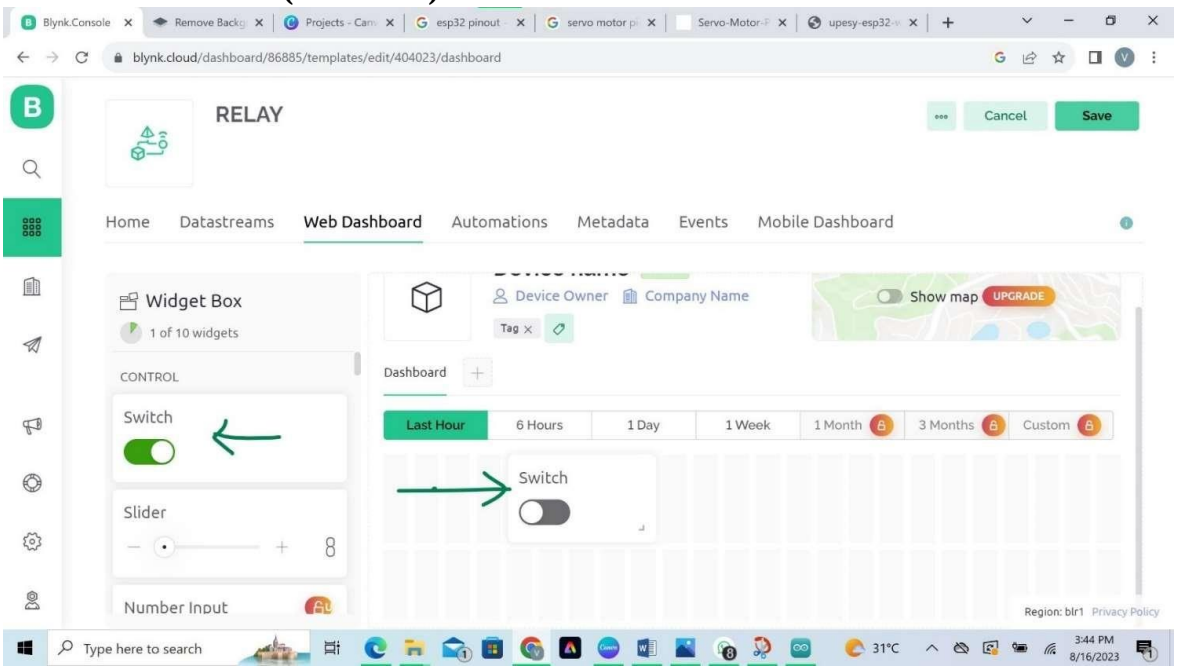
Enter the Name (Relay) > Select Hardware (ESP32) > Select Connection (WiFi) > Click Done.



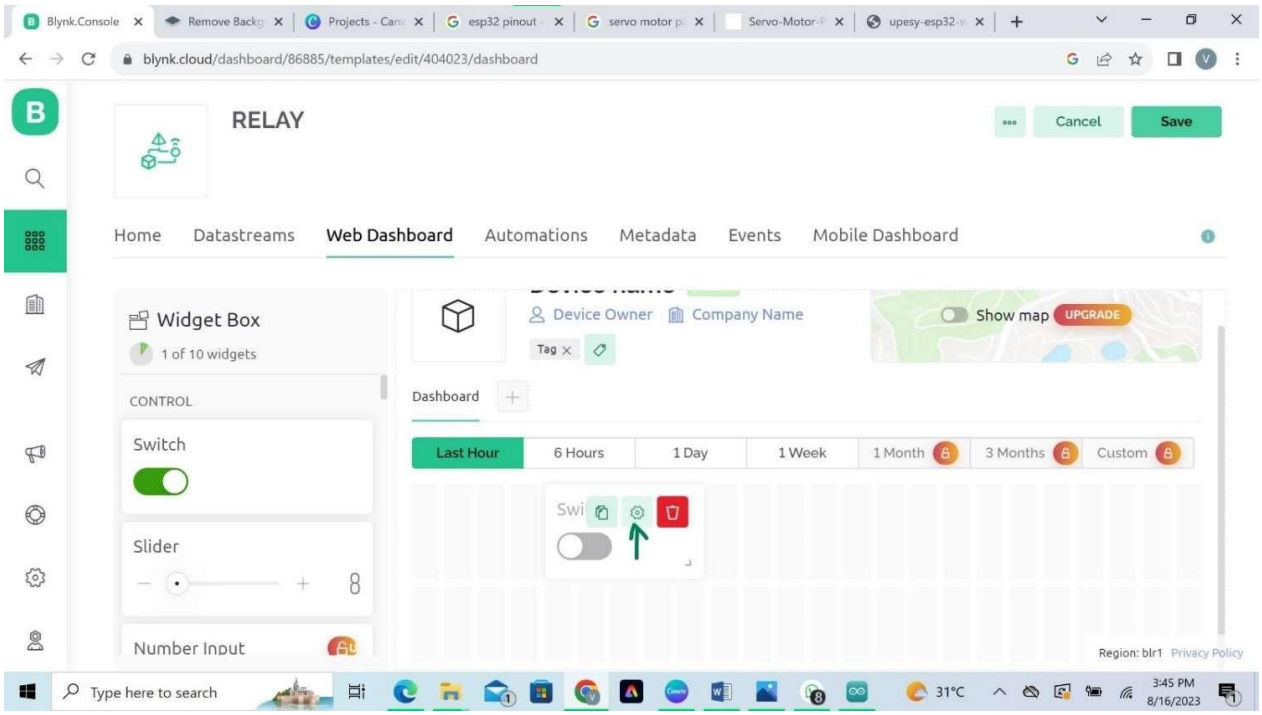
Go to Datastream > Click on New Datastream > Select Digital Pin.



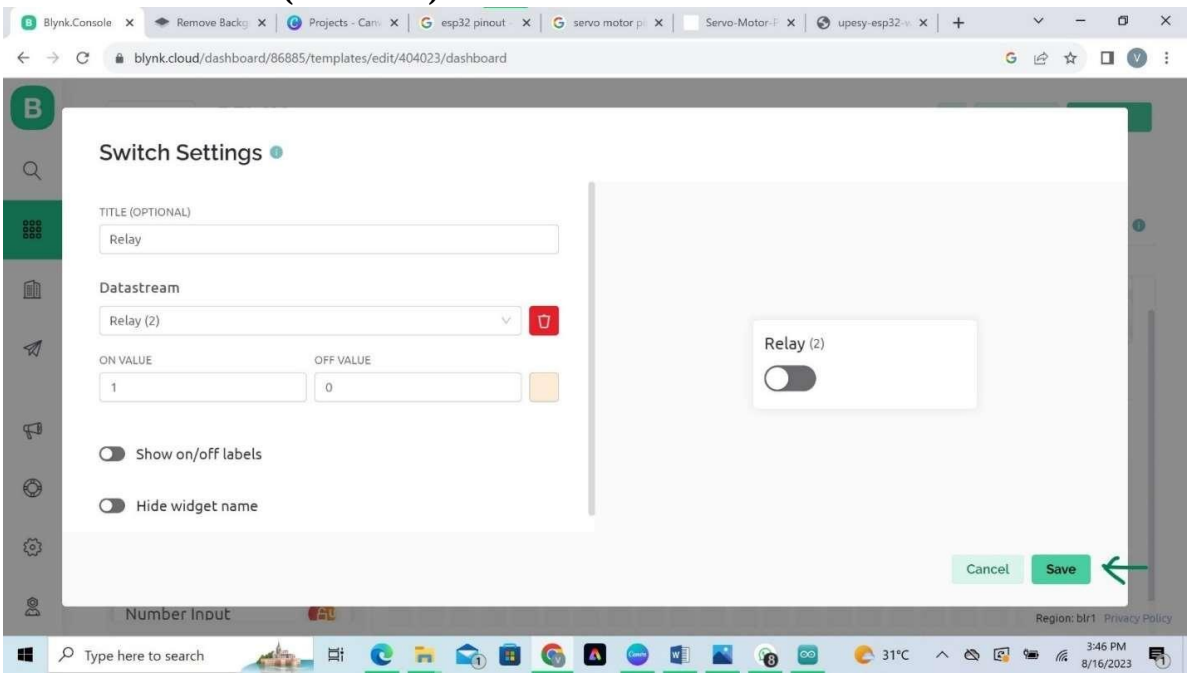
Enter the Name(Relay) > Enter Pin Number (Enter Same Pin Mentioned in Program) >SelectPinMode(OUTPUT) > Click on Create.

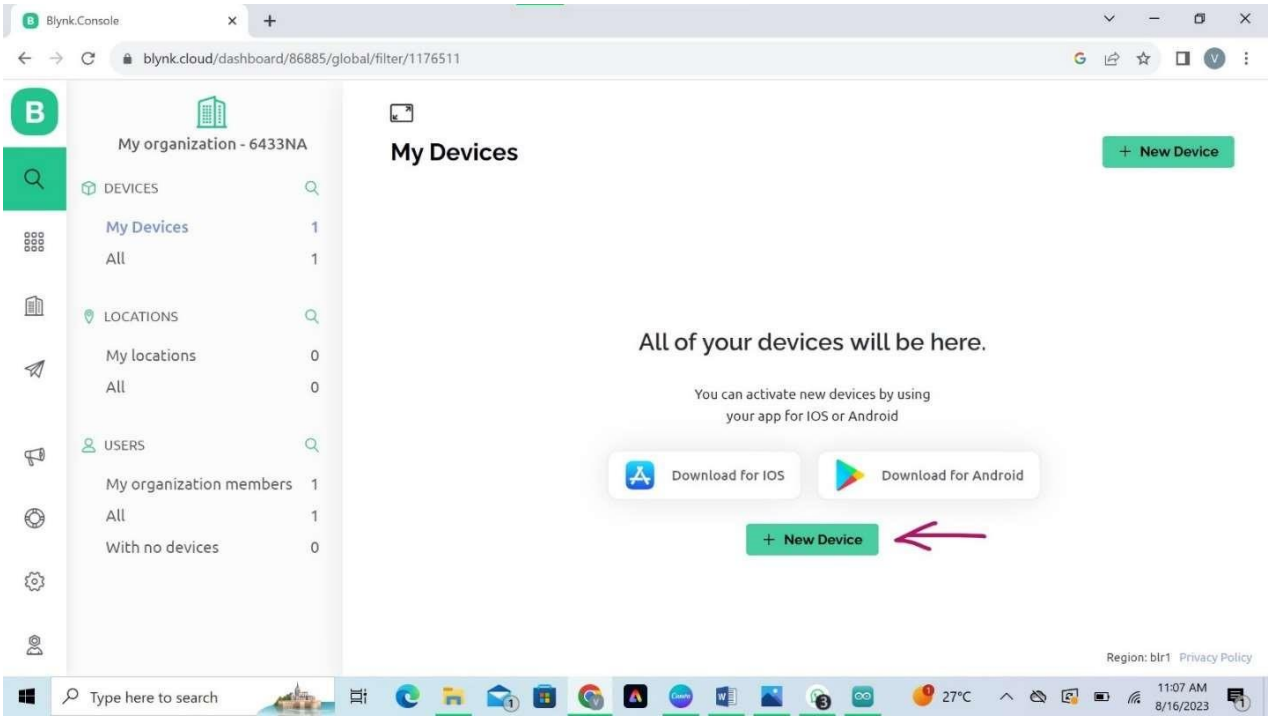


Click on Web Dashboard > Select Switch widget in Widget Box > Drag and drop to Dashboard asshown.

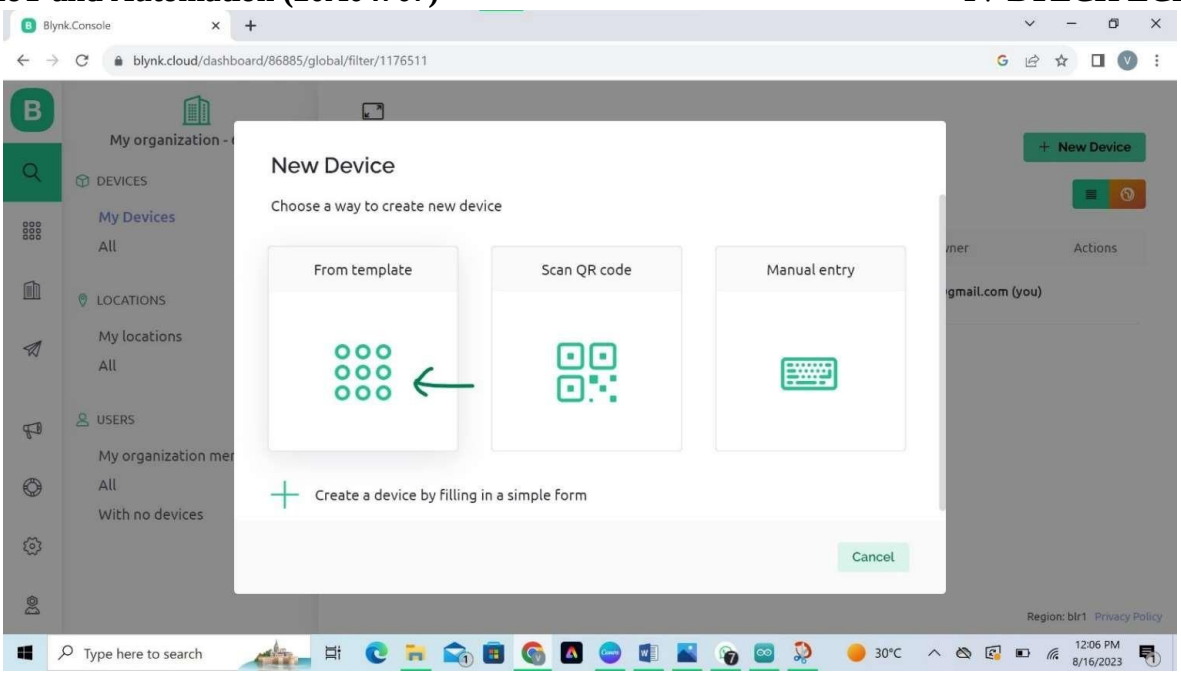


Click on Settings of Switch widget > Enter the Title > Select Datastream > Enter ON & OFF Value >Click Save.

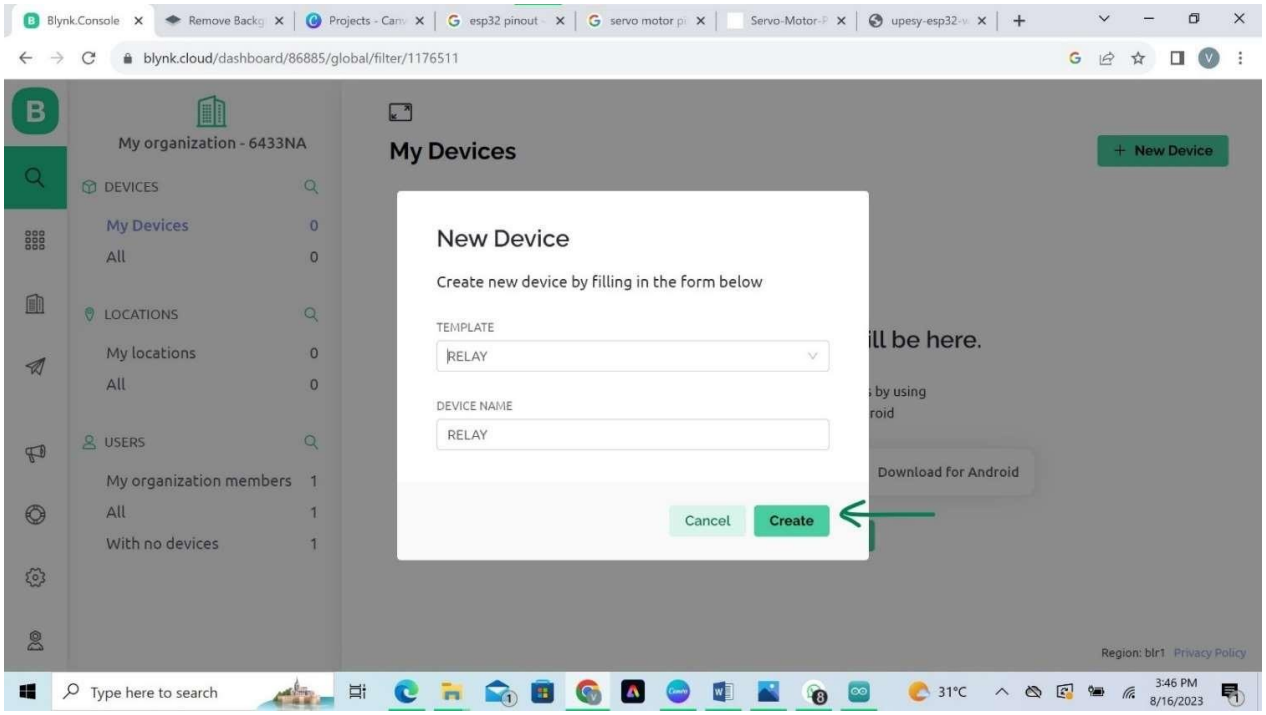




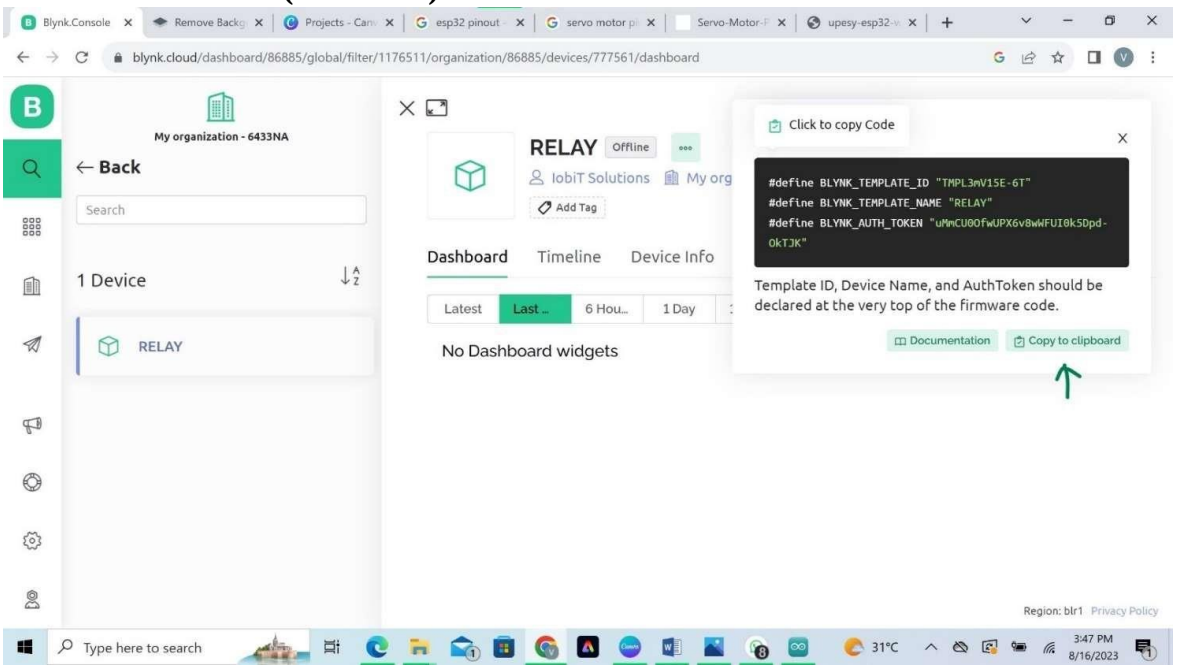
Click on Search icon > Click on New Device as shown above.



Click on From Template as shown above.



Select the Template you Created > Click on Create.



After Creating you will get Three line code. Just copy those lines and Paste it on Code.

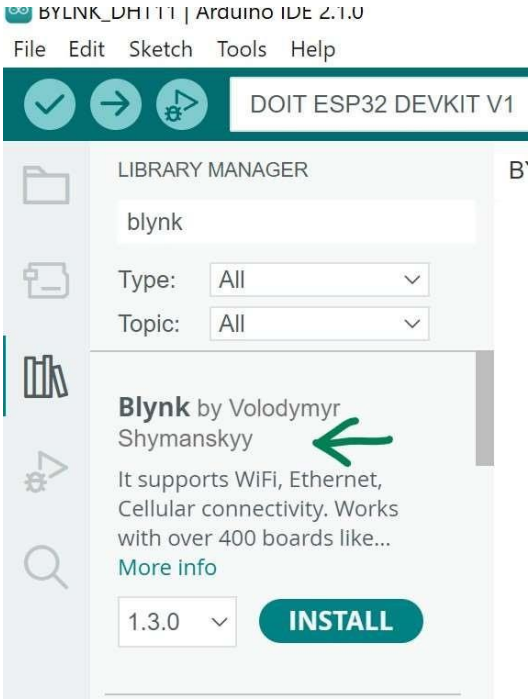
Code :

(Copy below Code and Paste it on your Arduino IDE and upload)

Library Installation

Paste the Code we copied from Blynk Cloud Platform and Replace it on top first three lines of below Code.

15th and 16th line Enter your WiFi Name and Password.



```
#define BLYNK_TEMPLATE_ID "TMPL3mV15E-6T"
#define BLYNK_TEMPLATE_NAME "RELAY"
#define BLYNK_AUTH_TOKEN "uMmCU0OfwUPX6v8wWFUI0k5Dpd-OkTJK"


#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>


char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "IobiT Solutions"; // Enter your Wifi Username
char pass[] = "IobiT@2023";  // Enter your Wifi password

int ledpin = 2;
void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  pinMode(ledpin,OUTPUT);
}

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

Result:

Conclusion:

Viva Questions:

PLC PROGRAM EXECUTION

New ⇨ Create a blank document.

◈ Method 1: Click on "New" under the "File" function.

◈ Method 2: Click on the icon, , on the tool bar.

◈ Method 3: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔N〕.

Open ⇨ Open the old documents in the drive.

◈ Method 1: Click on "Open" under the "File" function.

◈ Method 2: Click on the icon, , on the tool bar.

◈ Method 3: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔O〕.

Save ⇨ Save the file contents into the drive.

◈ Method 1: Click on "Save" under the "File" function.

◈ Method 2: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔S〕.

Save as… ⇨ Save the current file with a different name.

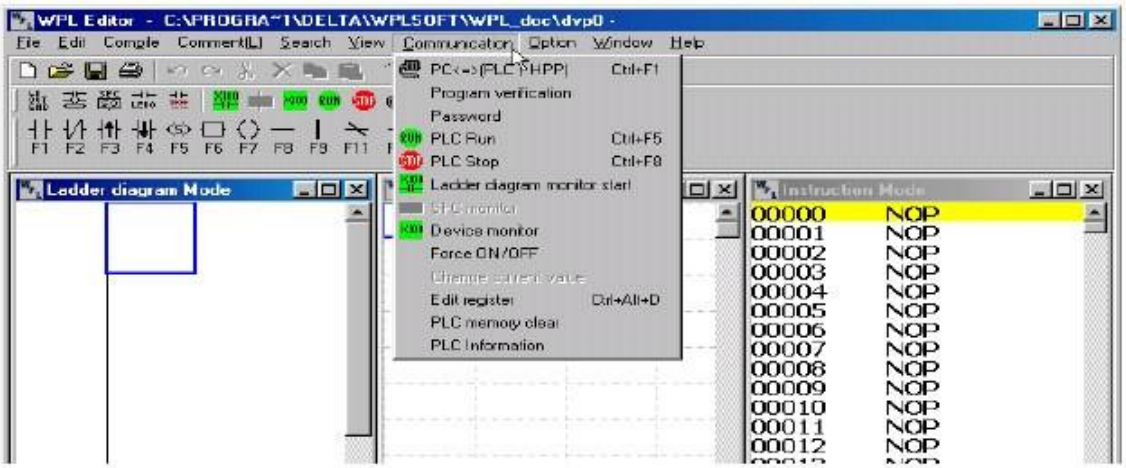
◈ Method 1: Click on "Save as" under the "File" function.

◈ Method 2: Click on the icon, , on the tool bar.

Close ⇨ Close the current file.

◈ Method: Click on "Close" under the "File" function.

COMMUNICATION with PC



- PC<=> (PLC | HPP) ⇨ The communication between PC and PLC or PC and HPP are meant for the readout or write-in of programs.
  - ◈ Method 1: Click on "PC 〈 = 〉 (PLC | HPP)" under the "Communication" function.
  - ◈ Method 2: Click on the icon,  , on the tool bar.
  - ◈ Method 3: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔F1〕.
- Program verification ⇨ Verify whether the programs within PLC are the same as those in the process of editing.
  - ◈ Method: Click on "Program verification" under the "Communication" function.
- Password ⇨ Setup or remove the PLC password.
  - ◈ Method: Click on the "Password" under the "Communication" function.
- PLC Run ⇨ Execute the PLC.
  - ◈ Method 1: Click on "PLC Run" under the "Communication" function.
  - ◈ Method 2: Click on the icon,  , on the tool bar.
  - ◈ Method 3: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔F5〕.
- PLC Stop ⇨ Stop the execution of PLC.
  - ◈ Method 1: Click on "PLC Stop" under the "Communication" function.
  - ◈ Method 2: Click on the icon,  , on the tool bar.
  - ◈ Method 3: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔F8〕.

- Ladder diagram monitor start ⇨ Switch to the monitor mode of the ladder diagram. (Only effective under the ladder diagram mode)
  - ◈ Method 1: Click on "Ladder diagram monitor start" or "Ladder diagram monitor stop" under the "Communication" function.
  - ◈ Click on the icon,  , on the tool bar.
- SFC monitor ⇨ Switch to the monitor mode of the SFC editing mode. (Only effective under the SFC editing)
  - ◈ Method 1: Click on "SFC monitor start" or "SFC monitor stop" under the "Communication" function.
  - ◈ Click on the icon,  , on the tool bar.

  Device monitor ⇨ Switch to the device monitor window to get to know the status and numeric values of the device to be monitored.
  - ◈ Method 1: Click on "Device monitor" under the "Communication" function.
  - ◈ Click on the icon,  , on the tool bar.
- Force ON/OFF ⇨ Force devices (Y, M, S, T and C) to be set as ON or OFF. (only effective under the ladder diagram mode or the device monitor mode)
  - ◈ Method 1: Click on "Force ON/OFF" under the "Communication" function.
  - ◈ Method 2: Place the editing box upon the device, and press the right button on the mouse to select "Force ON" or "Force OFF" function.

  Change current value ⇨ Change the current value of the designated device register (T, C and D). (Only effective under the ladder diagram monitor mode or the device monitor mode)
  - ◈ Method 1: Click on "Change current value" under the "Communication" function.
  - ◈ Method 2: Place the editing box upon the device, and press the right button on the mouse to select "Change current value" function.

  Edit register ⇨ Proceed with functions such as read, write, print, file readout, and save the file within internal registers (T, C and D) of the PLC.
  - ◈ Method 1: Click on "Edit register" under the "Communication" function.
  - ◈ Method 2: Make use of the speedy key-in function, and simply type in the compound buttons 〔Ctrl〕+〔Alt〕+〔D〕.

4.3 Editing Example

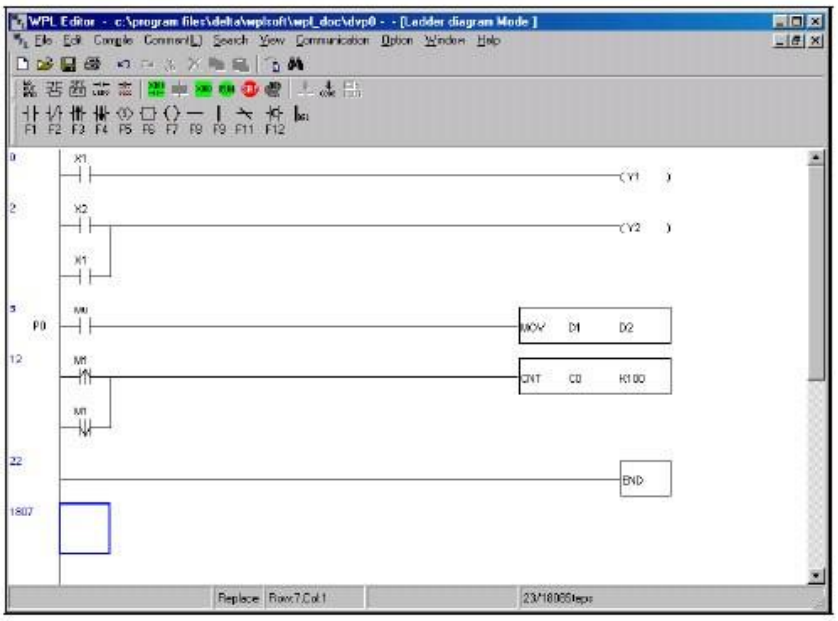
■ The Ladder Diagram

■ The Editing Operation Procedure of the Ladder Diagram:

Procedure	The Ladder Symbols	Location of the Cursor	Input through the clicking on the Function Keys		Input through the Keyboard
1	⊣ ⊢	Row: 0, Line: 1	F1 *Footnote 1	Device name: X Device number: 1	LD X1 ↵
2	—( )	Row: 0, Line: 2	F7	Device name: Y Device number: 1	OUT Y1 ↵
3	⊣ ⊢	Row: 1, Line: 1	F1	Device name: X Device number: 2	LD X2 ↵
4	|	Row: 1, Line: 2	F9		F9
5	—( )	Row: 1, Line: 2	F7	Device name: Y Device number: 2	OUT Y2 ↵
6	⊣ ⊢	Row: 2, Line: 1	F1	Device name: X Device number: 1	LD X1 ↵
7	⊣ ⊢	Row: 3, Line: 1	F1	Device name: M Device number: 0	LD M0 ↵
8	—▭	Row: 3, Line: 2	F6 *Footnote 2	Application command MOV Operand 1: D Device value: 1 Operand 2: D Device value: 2	MOV D1 D2 ↵
Procedure	The Ladder Symbols	Location of the Cursor	Input through the clicking on the Function Keys		Input through the Keyboard

9		Row: 4, Line: 0		Double click the mouse to input P0	P0 ↵
10	⊣↑⊢	Row: 4, Line: 1	F3	Device name: M Device number: 1	LDP M1 ↵
11	|	Row: 4, Line: 2	F9		F9
12	—□	Row: 4, Line: 2	F6	Counting command CNT Operand 1: C Device value: 0 Operand 2: K Device value: 100	CNT C0 K100 ↵
13	⊣↓⊢	Row: 5, Line: 1	F4	Device name: M Device number: 1	LDF M1 ↵
14	—□	Row: 6, Line: 1	F6	Application command END	END ↵

- After the input is completed, the Ladder Diagram could be converted to the command code and the SFC diagram through compiling, and will look like what follows:



*Footnote 1: Basic command input

Date:

Experiment No: 6

1. LOGIC GATE SIMULATION

STATIC APPLICATION PANELS

The Static application panels (SAP) are Simulation modules of various applications that usually come across in Industrial Environment. Such as Motor Control, level controlling, Process Controlling, Industrial automation etc.

For each SAP dedicated panels are designed. Terminals to connect Input and Output are brought outside the panels. Input like switch or sensor will be defined using push buttons or toggle switches and outputs like Motor, Relay coil, contractor will be defined through LED's. Attractive stickers are designed with different colors to differentiate input and outputs with designations.

CONNECTION BETWEEN SAP & PLC

Each SAP module requires Digital input and digital output. In the Main Panel of PLC the digital I/O's arebrought out to the panels and clearly designated for Eg.

Digital input as DI-0, DI-1, DI-2 ……..

Digital output as DO-0, DO-1, DO-2,……

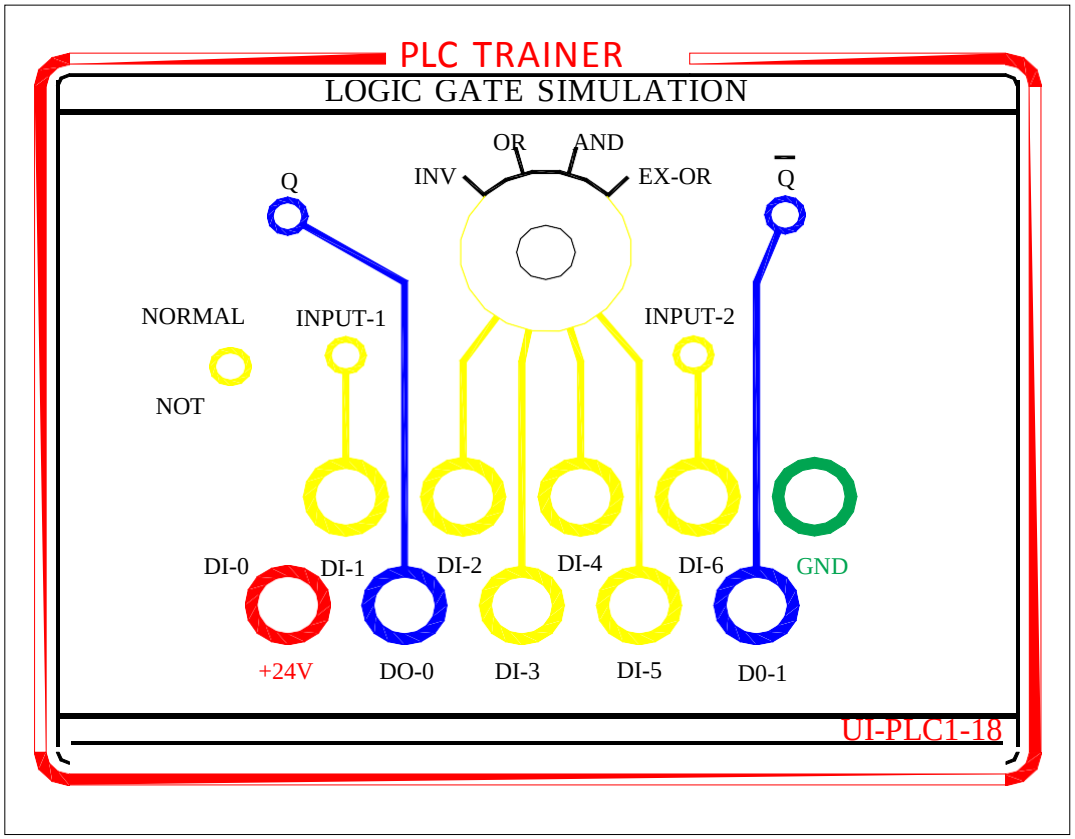
On all the SAP modules the required input and output are brought out to the terminals and designatedaccordingly. Match the inputs to inputs and outputs to outputs.

PLC PANEL

STATIC APPLICATION PANEL

1. LOGIC GATE SIMULATION

Logic Gate Panel:

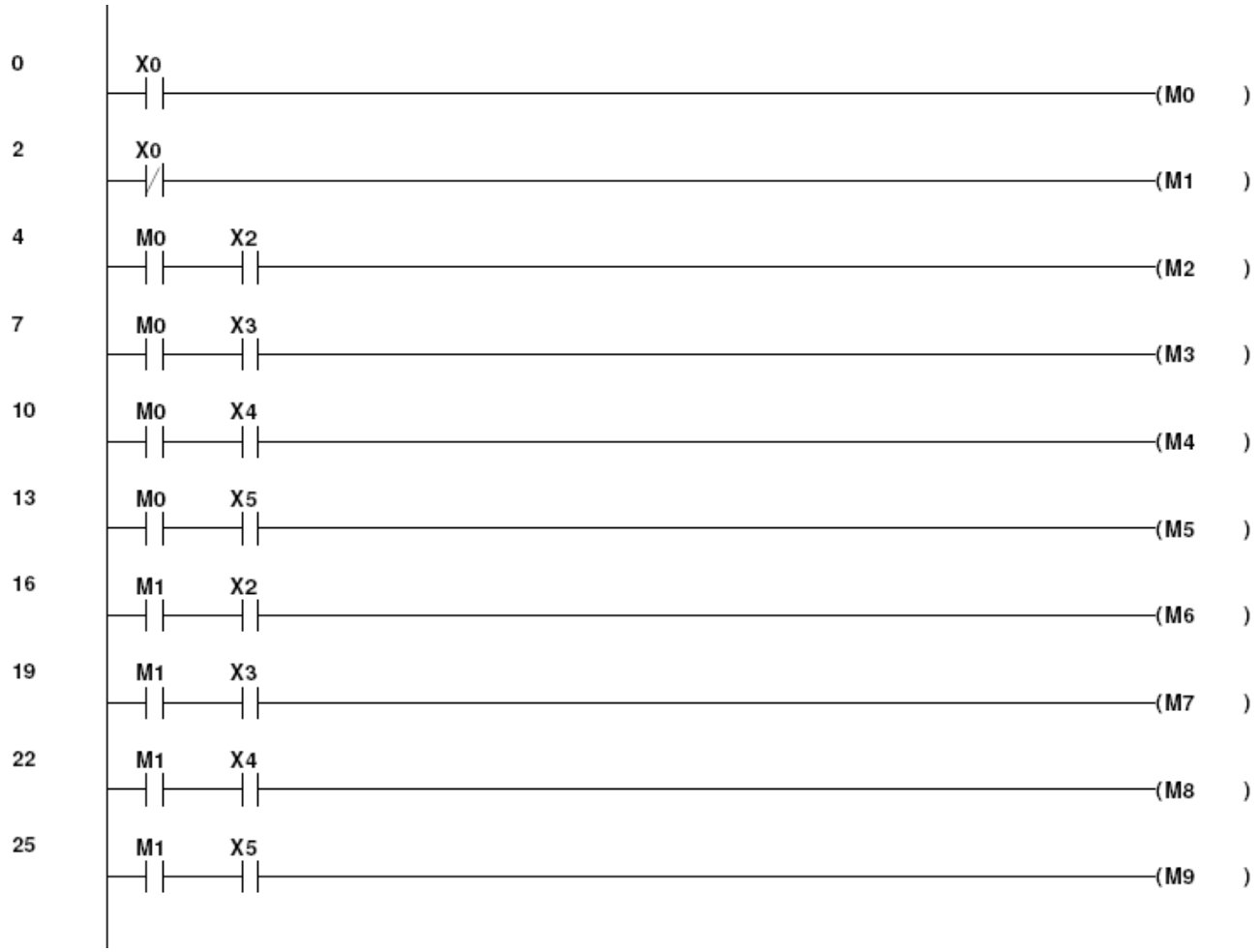


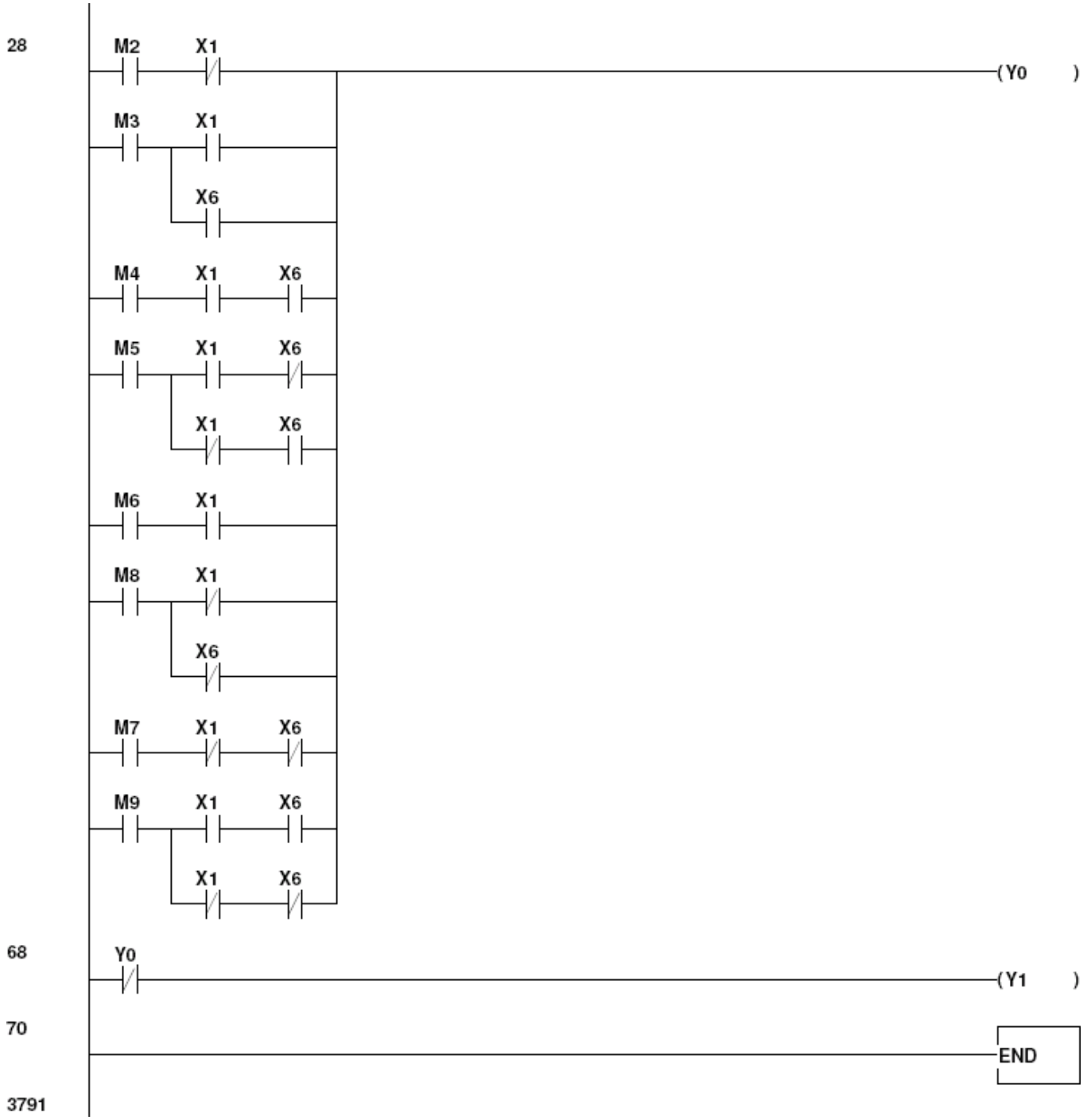
Definition : Logic Gate Simulation panel is an experimental module where the program is written in LD to simulate the LOGIC GATES, such as INV, OR, NOR, AND, EX-OR for any two given inputs. Two LED's are given to indicate the status of output Q and invQ. If the LED's in ON it indicates 1 if LED is off it indicates 0.

Digital inputs DI -0 (X0) to DI-6 (X6) are used as two switches and DO-0 (Y0) andDO-1(Y1) is used as output to LED.

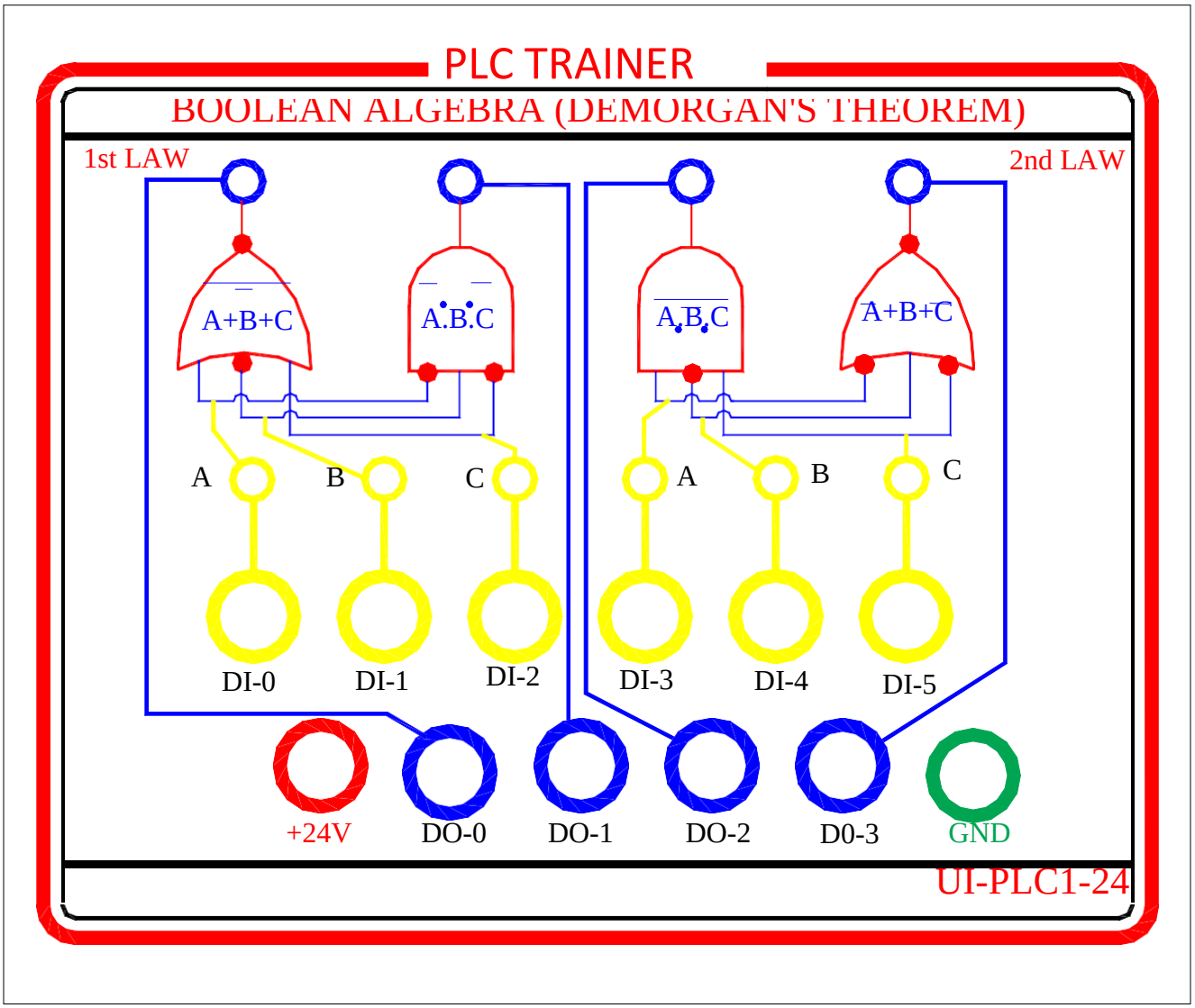
CONNECTION: Connect the Digital inputs from the PLC panel to the SAP panel using patch cards. Connect DI-0 and DI-1 on the PLC trainer to DI-0 and DI-1 on the application panel respectively. And DO-0 on the PLC trainer to DO-0 on the application panel. Connect 24V power supply and Ground to the respective terminals matching the terminal color.

LOGIC GATE SAMPLE LADDER PROGRAM




28
M2 X1 (Y0)
M3 X1
X6
M4 X1 X6
M5 X1 X6
X1 X6
M6 X1
M8 X1
X6
M7 X1 X6
M9 X1 X6
X1 X6
68
Y0 (Y1)
70
END
3791

2. BOOLEAN ALGEBRA

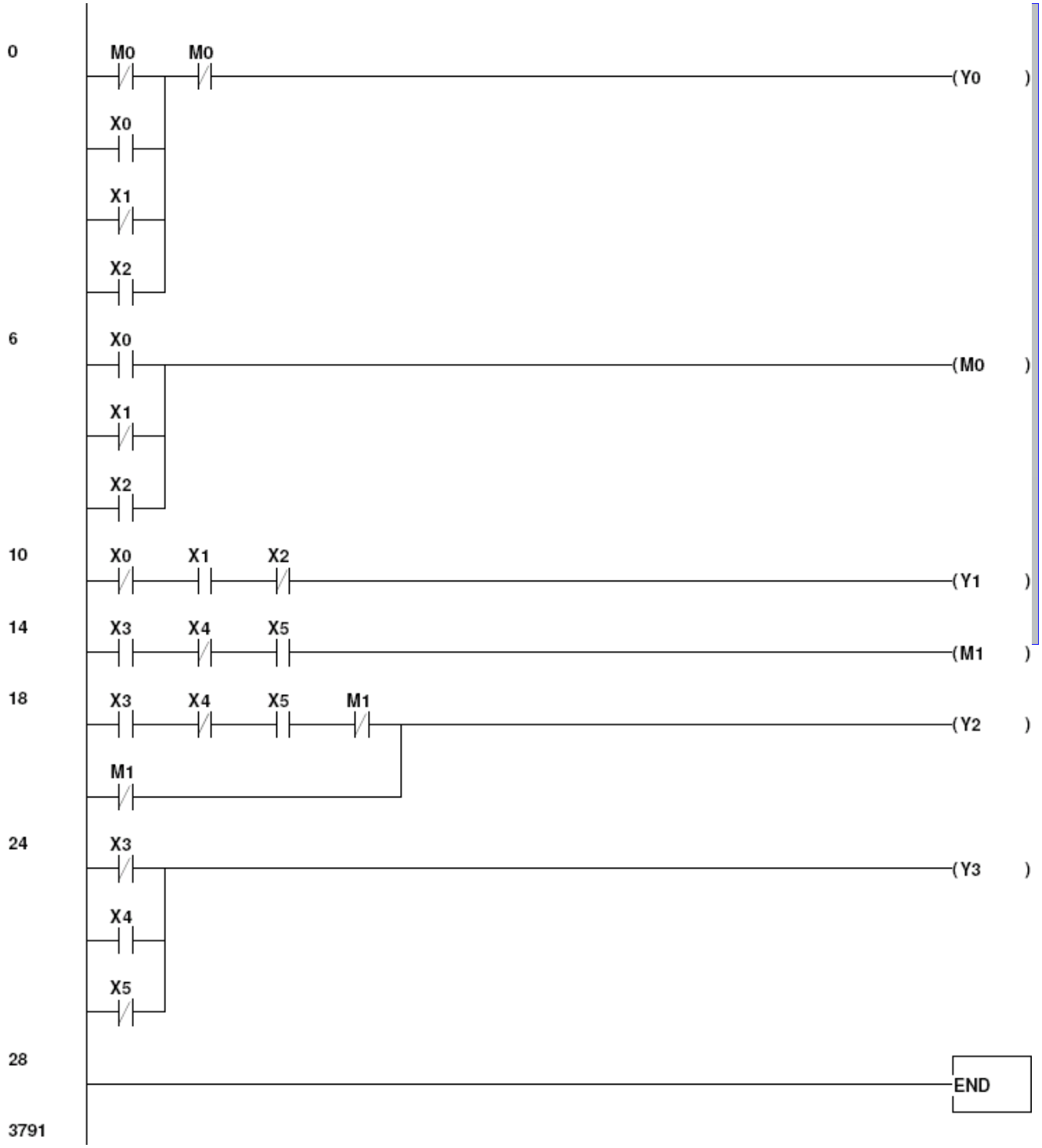


Definition : Boolean Algebra (Demorgan's Theorem) panel is an experimental module where the program is written inLD to prove Demorgan's theorem that LHS of an equation is equal to RHS for three inputs. Two LED's are given to indicate the status of LHS and RHS of the equation. If the LED's in ON it indicates 1 if LED is off it indicates 0.

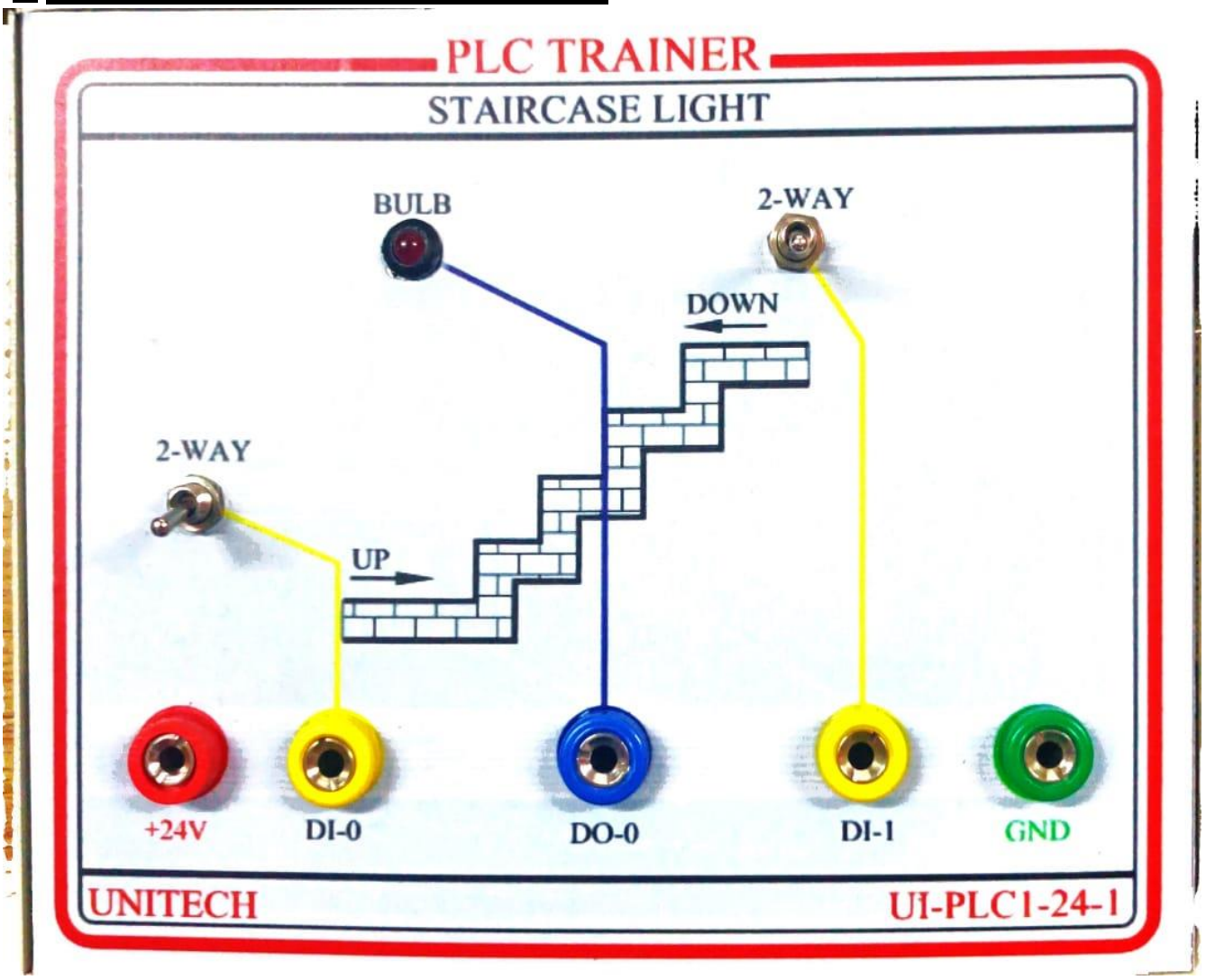
Digital inputs DI -0 (X0) to DI-5 (X5) are used as two switches and DO-0 (Y0) toDO-3(Y3) are used as output to LED.

CONNECTION: Connect the Digital inputs from the PLC panel to the SAP panel using patch cards. Connect DI-0 and DI-1 on the PLC trainer to DI-0 and DI-1 on the application panel respectively. And DO-0 on the PLC trainer to DO-0 on the application panel. Connect 24V power supply and Ground to the respective terminals matching the terminal color.

LADDER PROGRAM FOR BOOLEAN ALGEBRA



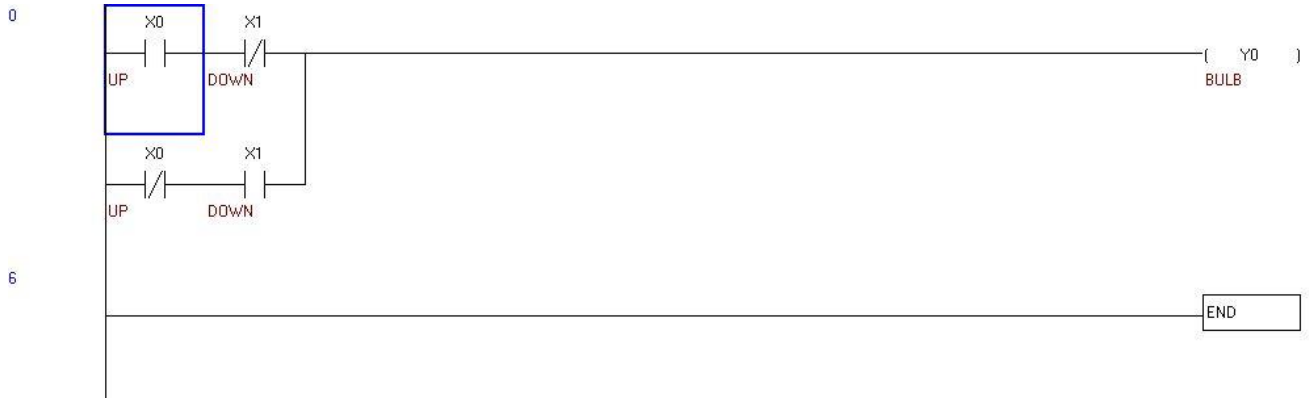
3. STAIR CASE CONTROL SIMULATION



Digital inputs DI -0 (X0) to DI-1 (X1) are used as two switches and DO-0 (Y0) are used as output to LED.

` **CONNECTION**: Connect the Digital inputs from the PLC panel to the SAP panel using patch cards. Connect DI-0 and DI-1 on the PLC trainer to DI-0 and DI-1 on the application panel respectively. And DO-0 on the PLC trainer to DO-0 on the application panel. Connect 24V power supply and Ground to the respective terminals matching the terminal color.

Ladder Diagram for Stair case



Result:

Conclusion:

Viva Questions: