

Name: Yash Walke
Class: MSc CS Part I

Academic Year: 2021 - 2022
Roll No. : 15

Advanced Embedded Systems

Index

Sr. No.	Title	Page No.
1	Switching ON and OFF LED using Push button with Arduino UNO.	2
2	Using 7-segment display with Arduino UNO to display from 0 to 9 after specific interval of time.	5
3	Configuring a digital object counter device using 7-segment display with Arduino UNO and IR proximity sensor.	10
4	Print message on LCD display with Arduino UNO.	17
5	Use 4 x 4 keypad to give the input in Arduino UNO serial monitor.	21
6	Interfacing of buzzer with Arduino UNO.	24
7	Interfacing of ultrasonic sensor with Arduino UNO.	27
8	Interfacing of servo motor with Arduino UNO.	31
9	Interfacing of DHT11 with Arduino UNO to read temperature and humidity which is then printed on serial monitor.	34
10	Interfacing of LED with Node MCU and controlling it remotely with Blynk application on mobile.	37

Practical 1

Aim: Switching ON and OFF LED using Push button with Arduino UNO.

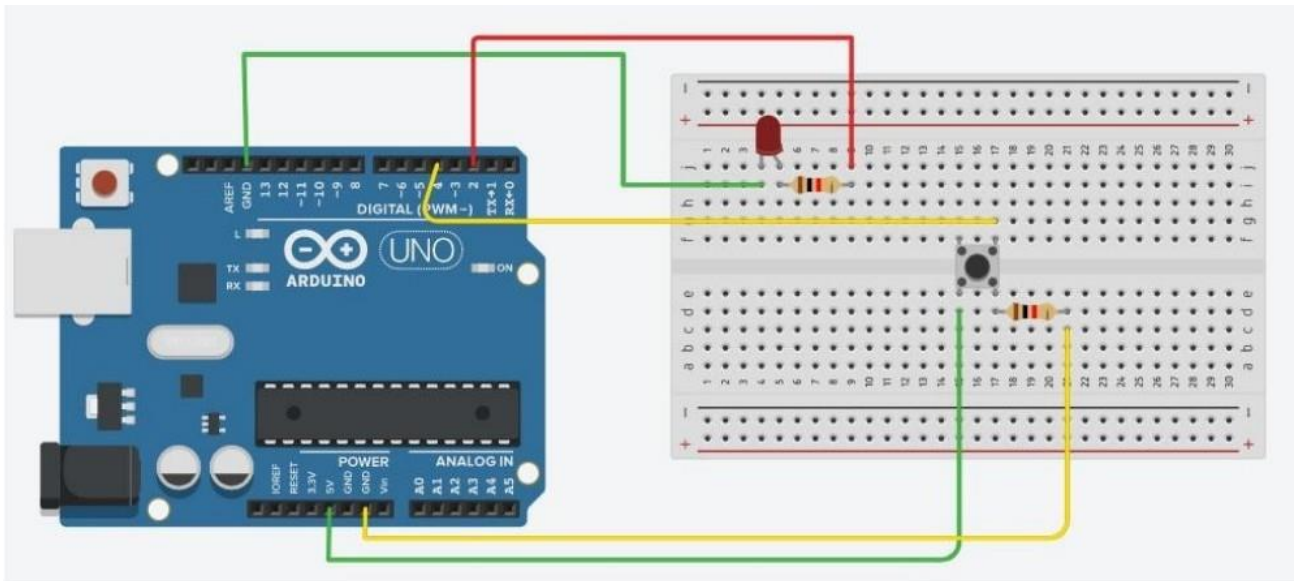
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **Pushbutton:**
 - The pushbutton is a component that connects two points in a circuit when you press it.
- **LED:**
 - A light-emitting diode (LED) is a semiconductor device that produces light from electricity. LEDs last a long time and do not break easily.

Required Components:

- Arduino Uno (1x)
- Breadboard (1x)
- LED (1x)
- Resistors (2x)
- Push Button (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to Push Button	5 V	Pin 1 (Using breadboard)
	GND	Pin 2 (Using breadboard and through resistance)
	Pin 4	Pin 3 (Using breadboard)
Arduino to LED	Pin 2	Positive end (Using breadboard and through resistance)
	GND	Negative end (Using breadboard)

Source Code:

```
const int ledpin = 2;
const int btnpin = 4;
int btnstate = 0;

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

void loop() {
  // put your main code here, to run repeatedly:
  btnstate = digitalRead(btnpin);
  if (btnstate == HIGH)
  {
    digitalWrite(ledpin, HIGH);
    Serial.println("LED ON");
  }
  else
  {
    digitalWrite(ledpin, LOW);
    Serial.println("LED OFF");
  }
}
```

Practical 2

Aim: Using 7-segment display with Arduino UNO to display from 0 to 9 after specific interval of time.

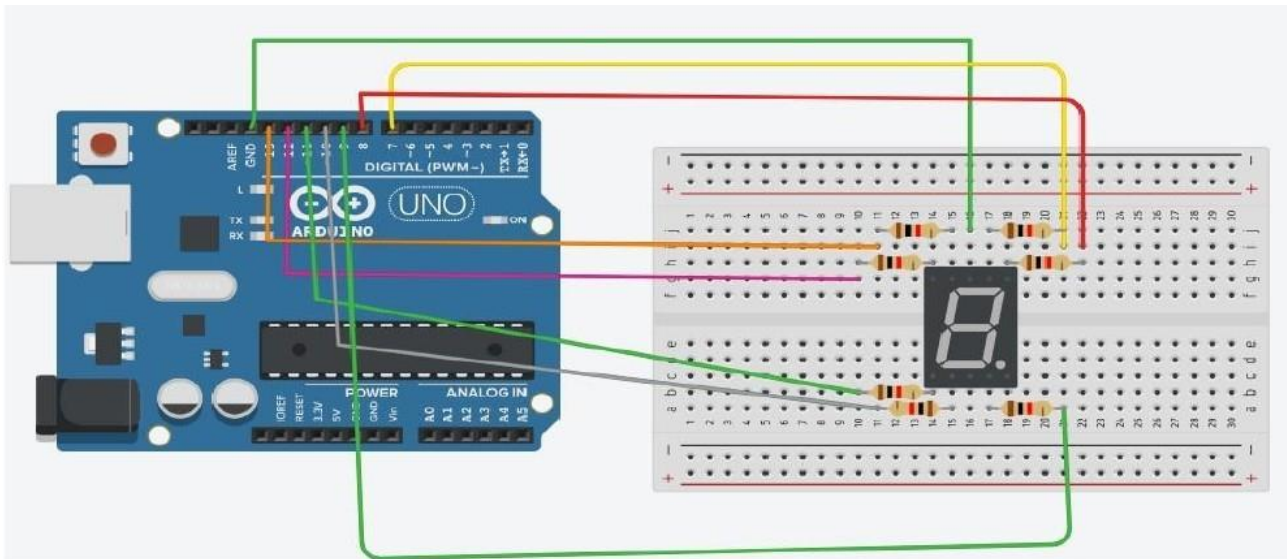
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **Seven-segment:**
 - The seven-segment display has seven LEDs arranged in the shape of number eight.
- **Resistors:**
 - It's a passive two-terminal electrical component that implements electrical resistance as a circuit element.
 - In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.

Required Components:

- Arduino Uno (1x)
- Breadboard (1x)
- Seven-segment Display (1x)
- Resistors (7x)
- Jump wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to 7-segment Display	GND	COM or GND
	13	Pin f (through resistance)
	12	Pin g (through resistance)
	11	Pin e (through resistance)
	10	Pin d (through resistance)
	9	Pin c (through resistance)
	8	Pin b (through resistance)
	7	Pin a (through resistance)

Source Code:

```
int f = 13;
int g = 12;
int e = 11;
int d = 10;
int c = 9;
int b = 8;
int a = 7;
int delay_ms = 1000;
int count = 0x00;
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
}

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

  digitalWrite(a, 1);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 1);
  digitalWrite(e, 1);
  digitalWrite(f, 1);
  digitalWrite(g, 0);

  delay(delay_ms); //0

  digitalWrite(a, 0);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 0);
  digitalWrite(e, 0);
  digitalWrite(f, 0);
  digitalWrite(g, 0);

  delay(delay_ms); //1

  digitalWrite(a, 1);
```

```
digitalWrite(b, 1);  
digitalWrite(c, 0);  
digitalWrite(d, 1);  
digitalWrite(e, 1);  
digitalWrite(f, 0);  
digitalWrite(g, 1);
```

```
delay(delay_ms); //2
```

```
digitalWrite(a, 1);  
digitalWrite(b, 1);  
digitalWrite(c, 1);  
digitalWrite(d, 1);  
digitalWrite(e, 0);  
digitalWrite(f, 0);  
digitalWrite(g, 1);
```

```
delay(delay_ms); //3
```

```
digitalWrite(a, 0);  
digitalWrite(b, 1);  
digitalWrite(c, 1);  
digitalWrite(d, 0);  
digitalWrite(e, 0);  
digitalWrite(f, 1);  
digitalWrite(g, 1);
```

```
delay(delay_ms); //4
```

```
digitalWrite(a, 1);  
digitalWrite(b, 0);  
digitalWrite(c, 1);  
digitalWrite(d, 1);  
digitalWrite(e, 0);  
digitalWrite(f, 1);  
digitalWrite(g, 1);
```

```
delay(delay_ms); //5
```

```
digitalWrite(a, 0);  
digitalWrite(b, 0);  
digitalWrite(c, 1);  
digitalWrite(d, 1);  
digitalWrite(e, 1);  
digitalWrite(f, 1);  
digitalWrite(g, 1);
```



```
    delay(delay_ms); //6

    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 0);
    digitalWrite(g, 0);

    delay(delay_ms); //7

    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 1);
    digitalWrite(f, 1);
    digitalWrite(g, 1);

    delay(delay_ms); //8

    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 1);
    digitalWrite(g, 1);

    delay(delay_ms); //9
}
```

Practical 3

Aim: Configuring a digital object counter device using 7-segment display with Arduino UNO and IR proximity sensor.

Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **Seven-segment:**
 - The seven-segment display has seven LEDs arranged in the shape of number eight.
- **Resistors:**
 - It's a passive two-terminal electrical component that implements electrical resistance as a circuit element.
 - In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.

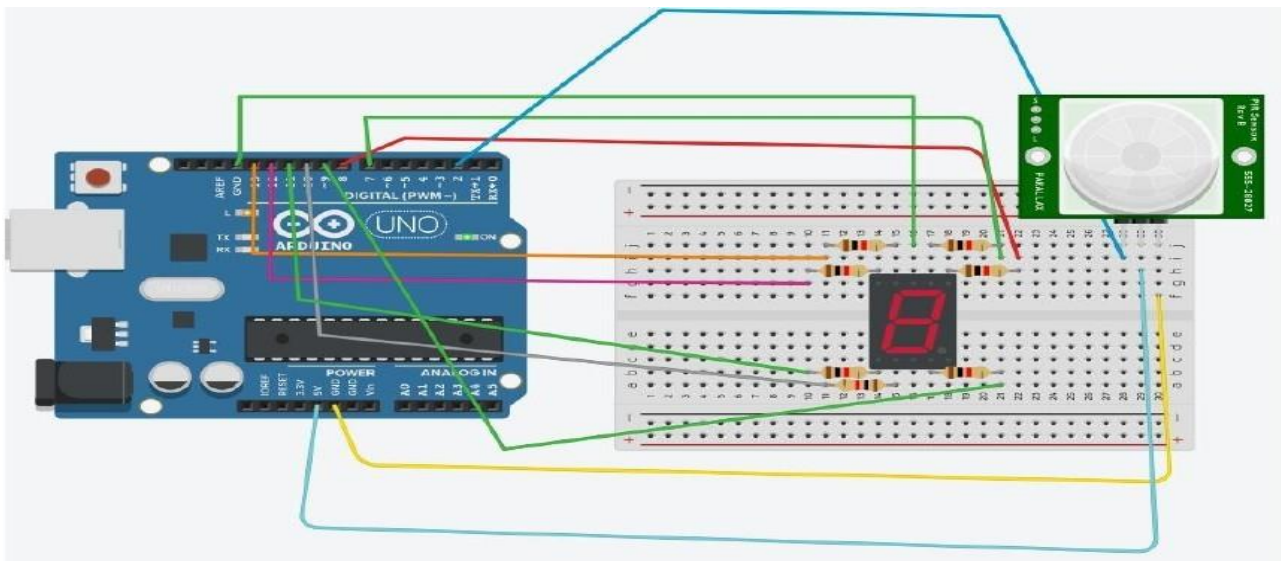
- **Infrared Sensor:**

- Infrared (IR) communication is a widely used and easy to implement wireless technology that has many useful applications.
- The most prominent examples in day to day life are TV/video remote controls, motion sensors, and infrared thermometers.

Required Components:

- Arduino Uno (1x)
- Breadboard (1x)
- Seven-segment Display (1x)
- Resistors (7x)
- IR Proximity Sensor (1x)
- Jump wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to 7-segment Display	GND	COM or GND
	13	Pin f (through resistance)
	12	Pin g (through resistance)
	11	Pin e (through resistance)
	10	Pin d (through resistance)
	9	Pin c (through resistance)
	8	Pin b (through resistance)
	7	Pin a (through resistance)
Arduino to IR Proximity Sensor	5 V	V _{CC}
	GND	GND
	2	O/P

Source Code:

```
int f = 13;
int g = 12;
int e = 11;
int d = 10;
int c = 9;
int b = 8;
int a = 7;
int delay_ms = 1000;
int irsensorpin = 5;
int irsensorstate = 0;
int p = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
}
```

```

pinMode(7, OUTPUT);
pinMode(5, INPUT);
}

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

    irsensorstate = digitalRead(irsensorpin);
    if(irsensorstate == HIGH)
    {
        p++;
    }

    if(p == 0)
    {
        digitalWrite(a, 1);
        digitalWrite(b, 1);
        digitalWrite(c, 1);
        digitalWrite(d, 1);
        digitalWrite(e, 1);
        digitalWrite(f, 1);
        digitalWrite(g, 0);

        delay(delay_ms); //0
    }

    if(p == 1)
    {
        digitalWrite(a, 0);
        digitalWrite(b, 1);
        digitalWrite(c, 1);
        digitalWrite(d, 0);
        digitalWrite(e, 0);
        digitalWrite(f, 0);
        digitalWrite(g, 0);

        delay(delay_ms); //1
    }

    if(p == 2)
    {
        digitalWrite(a, 1);
        digitalWrite(b, 1);
        digitalWrite(c, 0);
        digitalWrite(d, 1);
        digitalWrite(e, 1);
    }
}

```

```

        digitalWrite(f, 0);
        digitalWrite(g, 1);

        delay(delay_ms); //2
    }

    if(p == 3)
    {
        digitalWrite(a, 1);
        digitalWrite(b, 1);
        digitalWrite(c, 1);
        digitalWrite(d, 1);
        digitalWrite(e, 0);
        digitalWrite(f, 0);
        digitalWrite(g, 1);

        delay(delay_ms); //3
    }

    if(p == 4)
    {
        digitalWrite(a, 0);
        digitalWrite(b, 1);
        digitalWrite(c, 1);
        digitalWrite(d, 0);
        digitalWrite(e, 0);
        digitalWrite(f, 1);
        digitalWrite(g, 1);

        delay(delay_ms); //4
    }

    if(p == 5)
    {
        digitalWrite(a, 1);
        digitalWrite(b, 0);
        digitalWrite(c, 1);
        digitalWrite(d, 1);
        digitalWrite(e, 0);
        digitalWrite(f, 1);
        digitalWrite(g, 1);

        delay(delay_ms); //5
    }

    if(p == 6)

```

```

{
    digitalWrite(a, 0);
    digitalWrite(b, 0);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 1);
    digitalWrite(f, 1);
    digitalWrite(g, 1);

    delay(delay_ms); //6
}

if(p == 7)
{
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 0);
    digitalWrite(g, 0);

    delay(delay_ms); //7
}

if(p == 8)
{
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 1);
    digitalWrite(f, 1);
    digitalWrite(g, 1);

    delay(delay_ms); //8
}

if(p == 9)
{
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 1);

```

```
        digitalWrite(g, 1);  
        delay(delay_ms); //9  
    }  
}
```


Practical 4

Aim: Print message on LCD display with Arduino UNO.

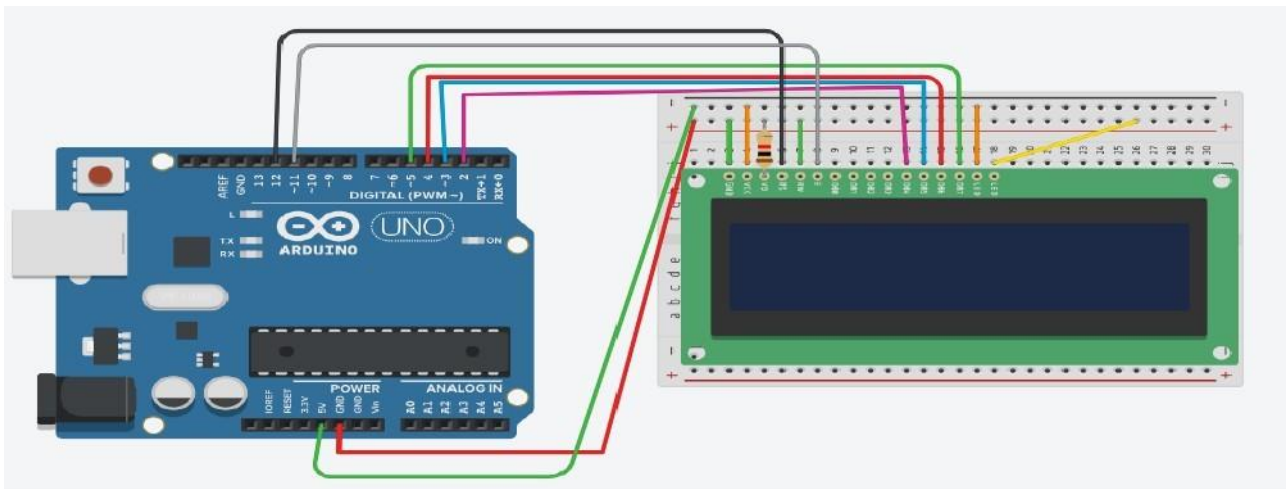
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **LCD:**
 - A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizer's.
 - Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in colour or monochrome.
 - It is 16 x 2 LCD display. That is it has 16 columns and 2 rows.

Required Components:

- Arduino Uno (1x)
- Breadboard (1x)
- LCD 16 x 2 (1x)
- Resistors (?x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to LCD display	12	RS
	11	Enable
	5	D4
	4	D5
	3	D6
	2	D7
	GND	R/W
	GND	V _{ss}
	5V	V _{cc}

Source Code:

Note: Install the *LiquidCrystal* library by navigating to Tools > Manage Libraries or by using the shortcut Ctrl + Shift + I.

```
#include<LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
    // put your setup code here, to run once:
    lcd.begin(16, 2);
    lcd.print("Hello, world!");
}

void loop() {
    // put your main code here, to run repeatedly:
    lcd.setCursor(13, 0);
    lcd.print("Ok");
    lcd.setCursor(5, 1);
```

```
for(int thisChar = 0; thisChar < 10; thisChar++)  
{  
    lcd.print(thisChar);  
    delay(500);  
}  
}
```

Practical 5

Aim: Use 4 x 4 keypad to give the input in Arduino UNO serial monitor.

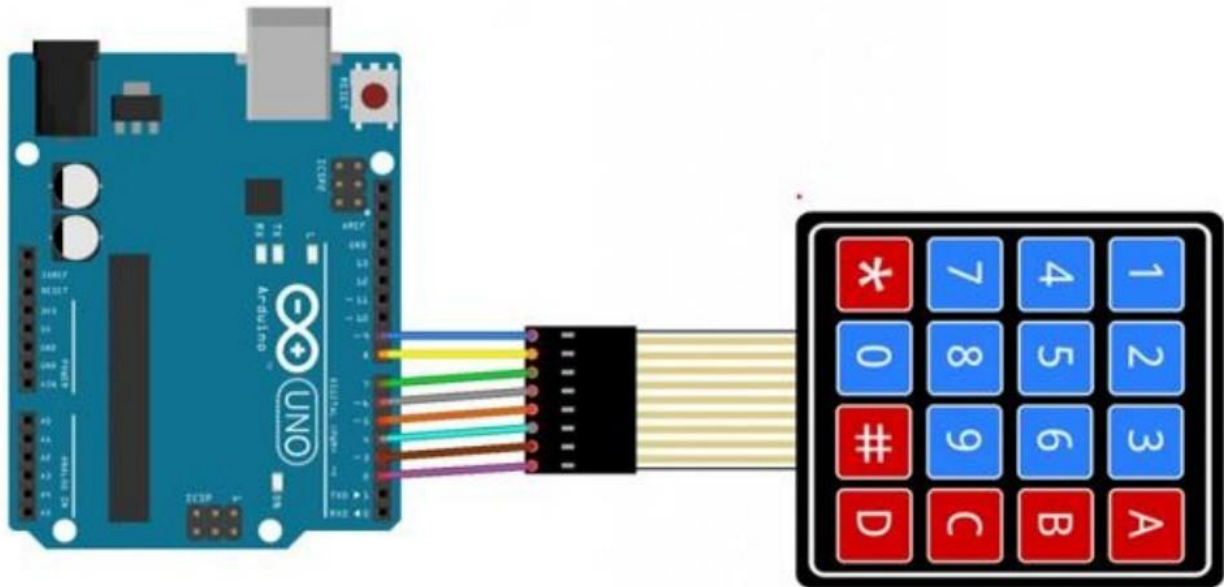
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Keypad :**
 - The buttons on a keypad are arranged in rows and columns. A 3X4 keypad has 4 rows and 3 columns, and a 4X4 keypad has 4 rows and 4 columns.
 - Keypad 4x4 is used for loading numerics into the microcontroller.
 - It consists of 16 buttons arranged in a form of an array containing four lines and four columns.
 - It is connected to the development system by regular IDC 10 female connector plugged in some development system's port.

Required Components:

- Arduino Uno (1x)
- 4 x 4 Keypad (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to 4 x 4 Keypad	9	R1
	8	R2
	7	R3
	6	R4
	5	C1
	4	C2
	3	C3
	2	C4

Source Code:

Note: Install the *Keypad* and *KeyMatrix* library by navigating to Tools > Manage Libraries or by using the shortcut Ctrl + Shift + I.

```
#include<Keypad.h>
#include<Key.h>
const byte rows = 4;
const byte cols = 4;
char keys[rows][cols] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};

byte colPins[rows] = {5, 4, 3, 2}; //Connect to the row pinouts of
keypad.
byte rowPins[cols] = {9, 8, 7, 6}; //Connect to the row pinouts of
keypad.
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, rows,
cols);

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

void loop() {
    // put your main code here, to run repeatedly:
    char key = keypad.getKey();
    if(key)
    {
        Serial.println(key);
    }
}
```

Practical 6

Aim: Interfacing of buzzer with Arduino UNO.

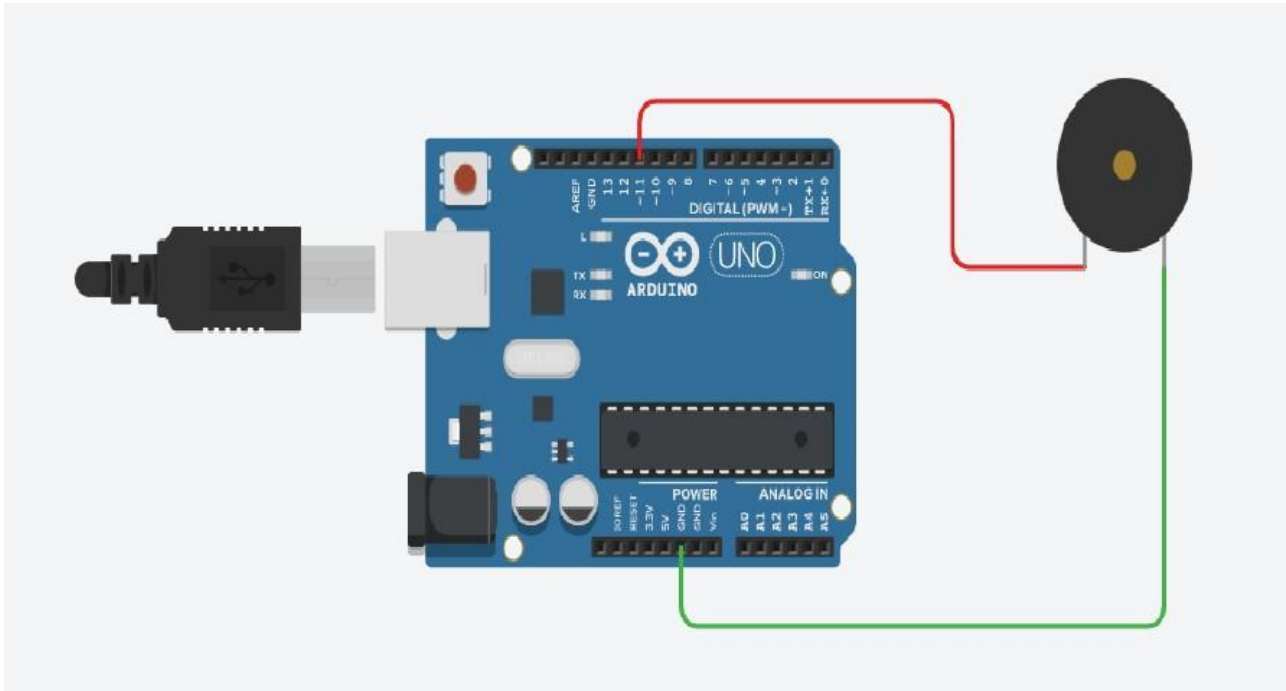
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Buzzer:**
 - A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short).
 - Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

Required Components:

- Arduino Uno (1x)
- Buzzer (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to Buzzer	11	+ve pin
	GND	-ve pin

Source Code:

1)

```
int buzzer = 11;

void setup() {
    // put your setup code here, to run once:

}

void loop() {
    // put your main code here, to run repeatedly:
    tone(buzzer, 450);
    delay(500);
    noTone(buzzer);
    delay(500);
}
```

2)

```
int buzzer = 11;

void setup() {
    // put your setup code here, to run once:

}

void loop() {
    // put your main code here, to run repeatedly:
    int i = 0;
    do
    {
        i++;
        tone(buzzer, 450);
        delay(200);
        noTone(buzzer);
        delay(200);
    }
    while(i < 3);
    delay(3000);
}
```

Practical 7

Aim: Interfacing of ultrasonic sensor with Arduino UNO.

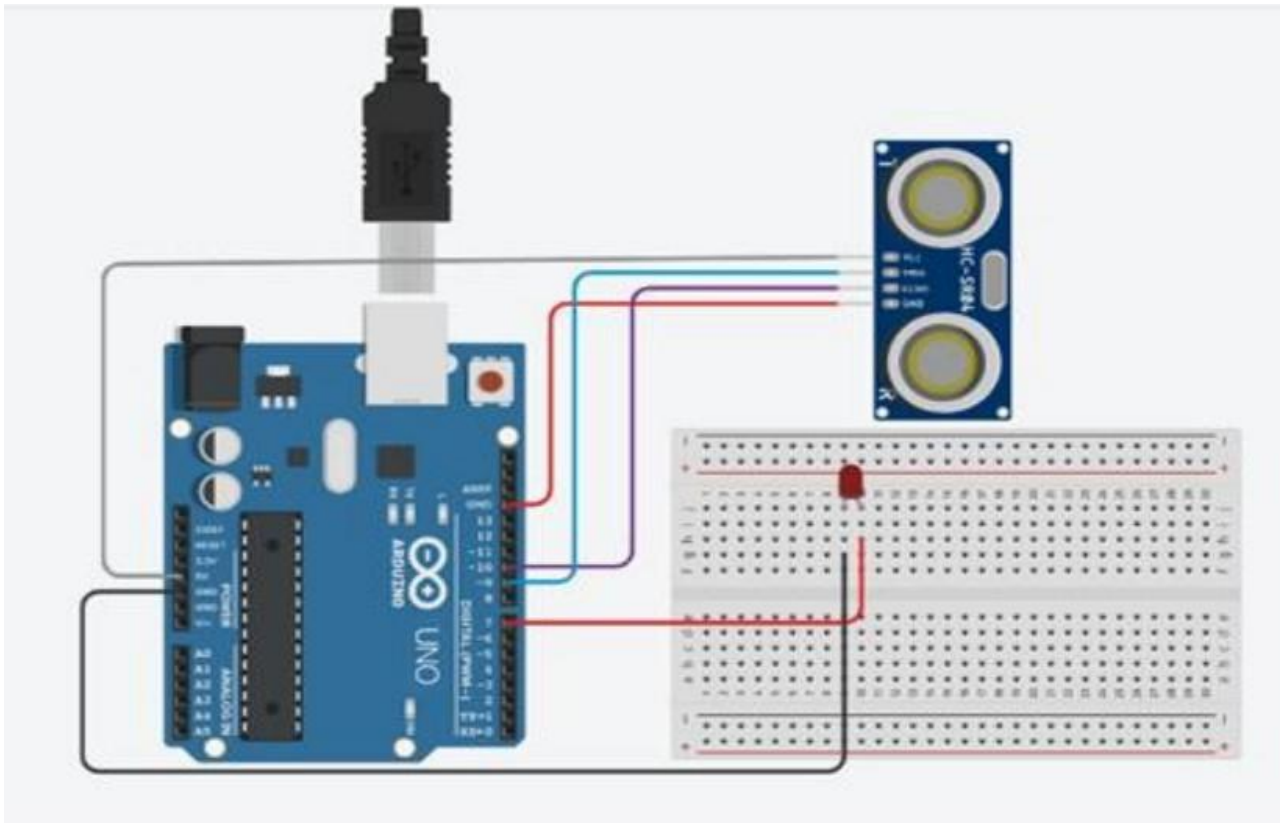
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **LED:**
 - A light-emitting diode (LED) is a semiconductor device that produces light from electricity. LEDs last a long time and do not break easily.
- **Ultrasonic Sensor:**
 - Ultrasonic distance sensor determines the distance to an object by measuring the time taken by the sound to reflect back from that object.
 - A typical ultrasonic distance sensor consists of two membranes. One membrane produces sound, another catches reflected echo. Basically they are speaker and microphone.

Required Components:

- Arduino Uno (1x)
- Breadboard (1x)
- LED (1x)
- Ultrasonic Sensor (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to Ultrasonic Sensor	5 V	VCC
	GND	GND
	9	Trig
	10	Echo
Arduino to LED	7	+ve end
	GND	-ve end

Source Code:

```
int trigpin = 9;
int echopin = 10;
int led = 7;

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

void loop() {
  // put your main code here, to run repeatedly:
  long duration, distance;
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigpin, LOW);
  duration = pulseIn(echopin, HIGH);
  distance = (duration / 2) / 29.1;
  Serial.print(distance);
  Serial.println("CM");
  delay(10);
  if(distance <= 10)
  {
    digitalWrite(led, HIGH);
  }
  else if(distance > 10)
  {
    digitalWrite(led, LOW);
  }
}
```

Practical 8

Aim: Interfacing of servo motor with Arduino UNO.

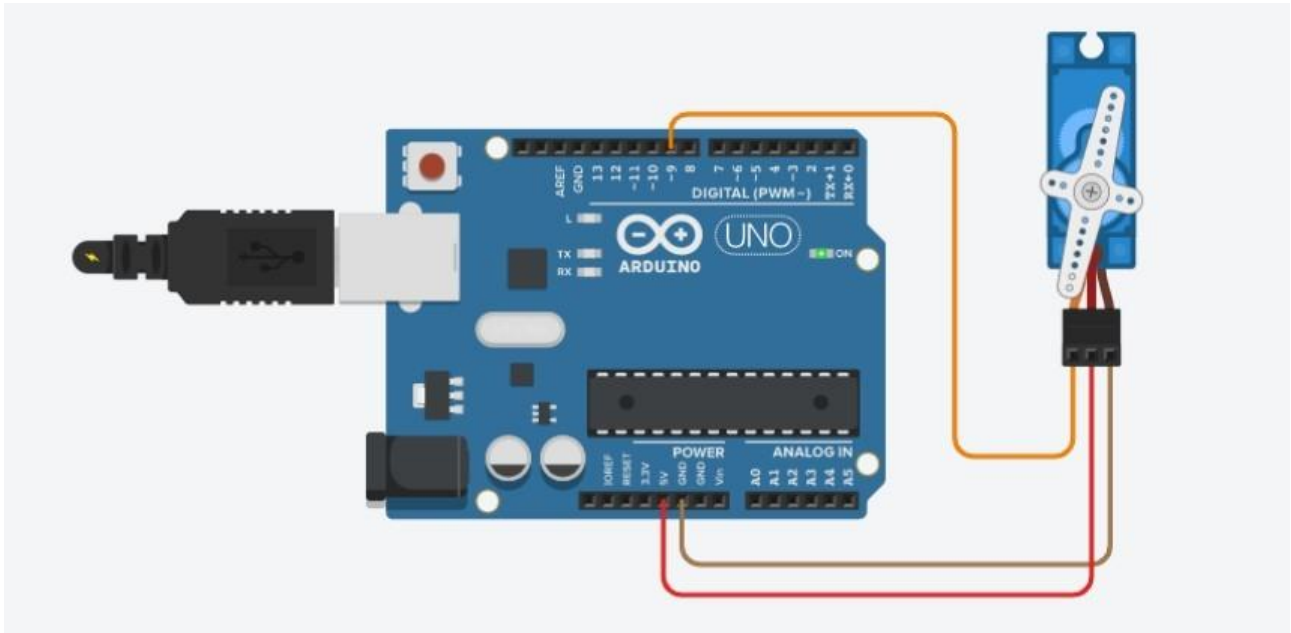
Description:

- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Servo Motor:**
 - A servo motor is an electrical device which can push or rotate an object with great precision.
 - If you want to rotate an object at some specific angles or distance, then you use servo motor.

Required Components:

- Arduino Uno (1x)
- Servo Motor (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to Servo Motor	9	Orange pin
	5 V	Red pin
	GND	Brown pin

Source Code:

Note: Install the Servo library by navigating to Tools > Manage Libraries or by using the shortcut Ctrl + Shift + I.

```
#include<Servo.h>
//Create Servo object to control a Servo
Servo myservo;
// 12 servo objects can be created on most boards.
int pos;
void setup() {
    // put your setup code here, to run once:
    myservo.attach(9); //Variable to store the servo position.
}

void loop() {
    // put your main code here, to run repeatedly:
    for(pos = 0; pos <= 180; pos += 1)
    {
        // Goes from 0 to 180 degrees with a step of 1
        myservo.write(pos); //Tell servo to go to position 'pos'.
        delay(15); //Wait 15ms for the servo to reach the position
        'pos'.
    }

    for(pos = 180; pos >= 0; pos -= 1)
    {
        // Goes from 180 to 0 degrees with a step of -1
        myservo.write(pos); //Tell servo to go to position 'pos'.
        delay(15); //Wait 15ms for the servo to reach the position
        'pos'.
    }
}
```

Practical 9

Aim: Interfacing of DHT11 with Arduino UNO to read temperature and humidity which is then printed on serial monitor.

Description:

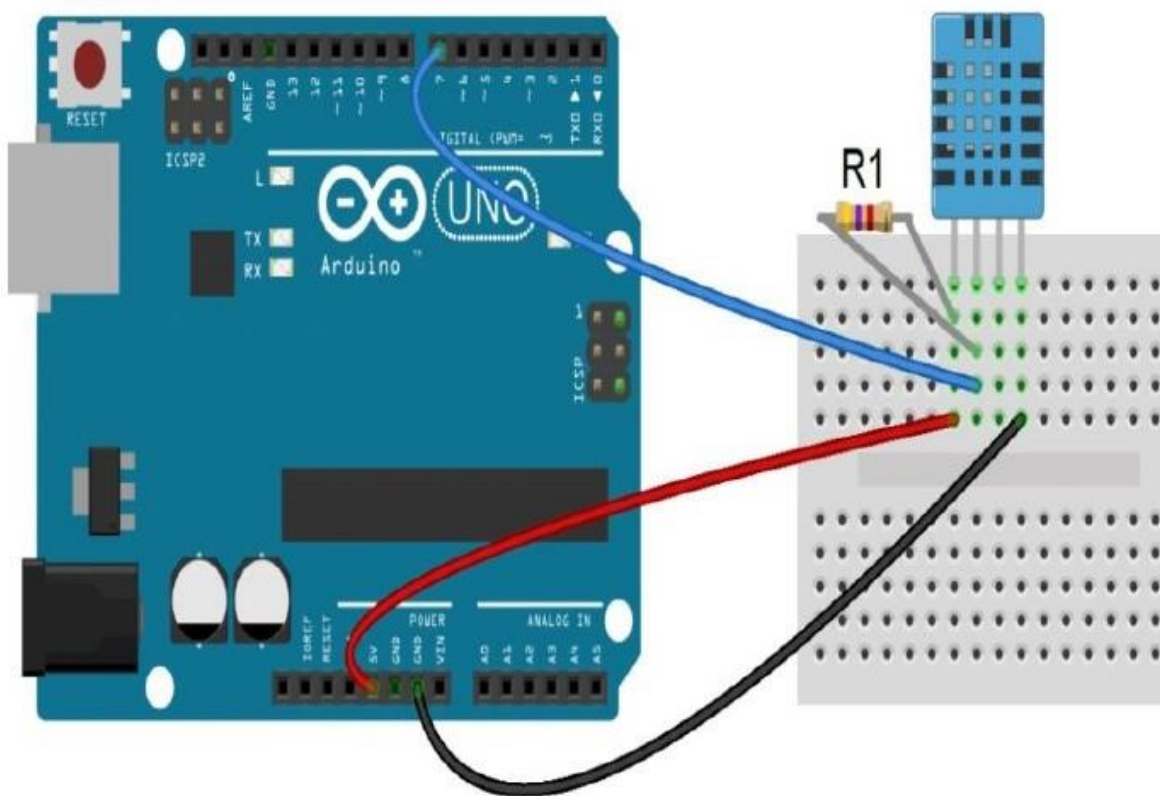
- **Arduino:**
 - Arduino is an open-source platform used for building electronics projects.
 - Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board.
 - Arduino UNO has 14 digital pins and 6 analog pins.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **Resistors:**
 - It's a passive two-terminal electrical component that implements electrical resistance as a circuit element.
 - In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.
- **DHT11:**
 - DHT11 is a low cost digital sensor for sensing temperature and humidity.
 - This can be easily interfaced with any microcontroller like Arduino, Raspberry Pi etc to measure humidity and temperature instantaneously.

- This sensor is used various applications such as measuring humidity and temperature values in heating, ventilation and AC systems.
- Offices, cars, green houses use this sensor for measuring humidity values and safety measure. This can be used for smart gardening.

Required Components:

- Arduino Uno (1x)
- Breadboard (1x)
- Resistor (1x)
- DHT11 Sensor (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
Arduino to DHT11	5 V	V _{CC}
	GND	GND
	7	SDA
Resistor to DHT11	V _{CC}	SDA

Source Code:

Note: Install the *DHTLib* library by navigating to Tools > Manage Libraries or by using the shortcut Ctrl + Shift + I.

```
#include<dht.h>
#define DHT11_PIN 7
dht DHT;

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

void loop() {
    // put your main code here, to run repeatedly:
    int chk = DHT.read11(DHT11_PIN);
    Serial.print("Temperature = ");
    Serial.println(DGT.temperature);
    Serial.print("Humidity = ");
    Serial.println(DHT.humidity);
    delay(1000);
}
```

Practical 10

Aim: Interfacing of LED with Node MCU and controlling it remotely with Blynk application on mobile.

Description:

- **NodeMCU:**
 - NodeMCU is low-cost, open source IOT platform.
 - It initially included firmware which runs on the ESP8266 Wi-Fi SoC.
 - Arduino UNO does not have inbuilt Wi-Fi module. It provides access to the GPIO.
 - It has 10 digital pins and only 1 analog pin.
 - It can also be programmed directly using Arduino IDE.
 - It consumes ten times of power than Arduino UNO.
- **Blynk App:**
 - Blynk is a platform with IOS and android apps to control Arduino, Raspberry Pi and so on.
 - It's a digital dashboard where we can build a graphic interface for our project by simply dragging and dropping widgets.
- **Breadboard:**
 - It is a way of constructing electronics without having to use a soldering iron.
 - Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
- **LED:**
 - A light-emitting diode (LED) is a semiconductor device that produces light from electricity. LEDs last a long time and do not break easily.

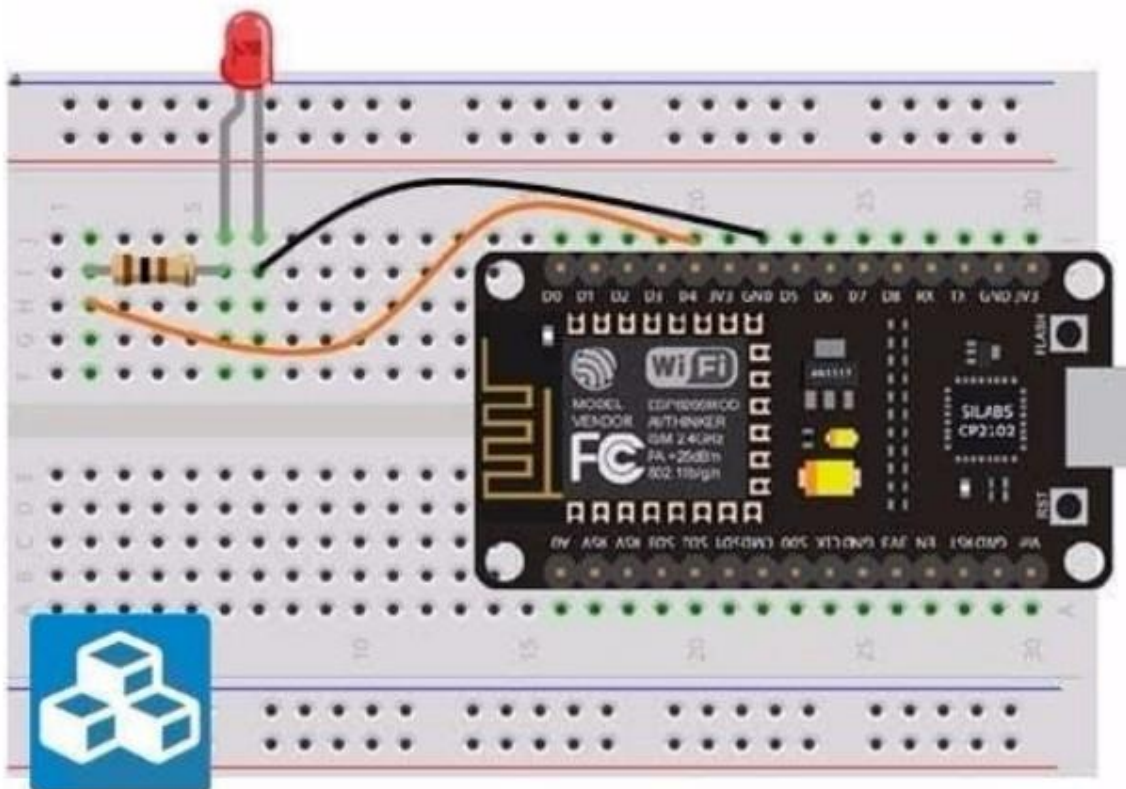
- **Resistors:**

- It's a passive two-terminal electrical component that implements electrical resistance as a circuit element.
- In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.

Required Components:

- NodeMCU (1x)
- Breadboard (1x)
- LED (1x)
- Resistor (1x)
- Jump Wires

Circuit Diagram:



Connections:

Components	Connections	
	From	To
NodeMCU to LED	D5	+ve pin
	GND	-ve pin

Blynk Iot App Configuration:

To control LED connected to NodeMCU remotely we need to install Blynk IoT app in our mobile as follows –

1. Download Blynk IoT app from play store or app store.

3:08

📶 78%



Sign Up

Log In

—

2. After downloading, create an account by clicking on Sign Up.

3:09 78%

← Sign Up

Simply fill in your email address and we will
send an account activation link.

EMAIL

your.email@email.com

Enter email

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

Continue

3. Enter your Email ID and then you will receive a link in your inbox where you need to enter a password for your account.

Set a password

Create a password which is hard to guess.

PASSWORD

password

Enter new password

VERY WEAK

- Make it longer than 8 symbols
- Use uncommon words
- Use non-standard uPPercaSing,
- Use creatif spelllllllling
- Use non-obvi0u\$ number\$ & symbo1s

Continue

4. After this, you need to enter your first name to finish the setup of your account.

User Profile

1 of 1

Fill in profile information

FIRST NAME

Enter...

Enter first name

Next

Blynk

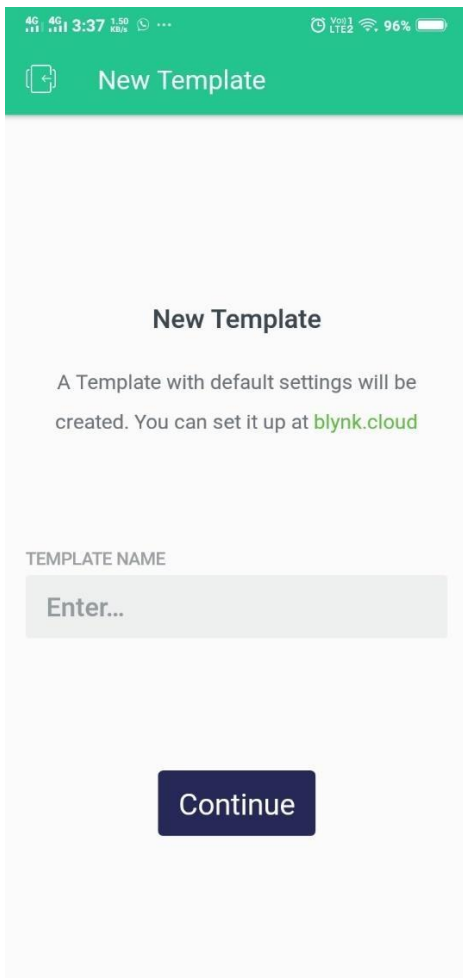
No devices yet

You don't have any devices connected yet

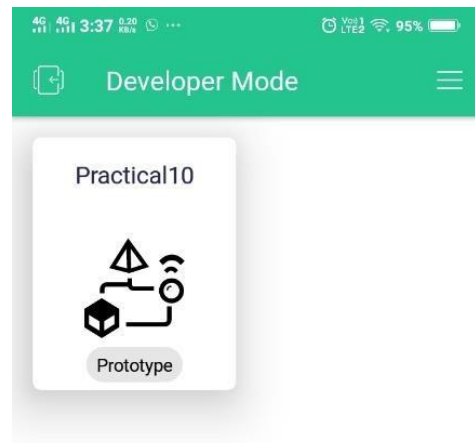
Add new device

Developer Mode

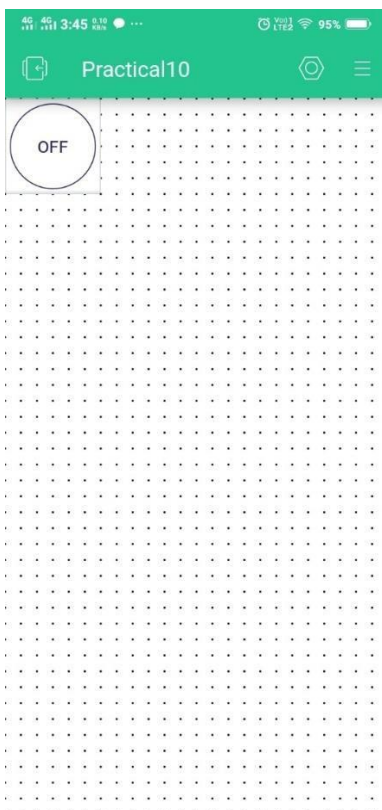
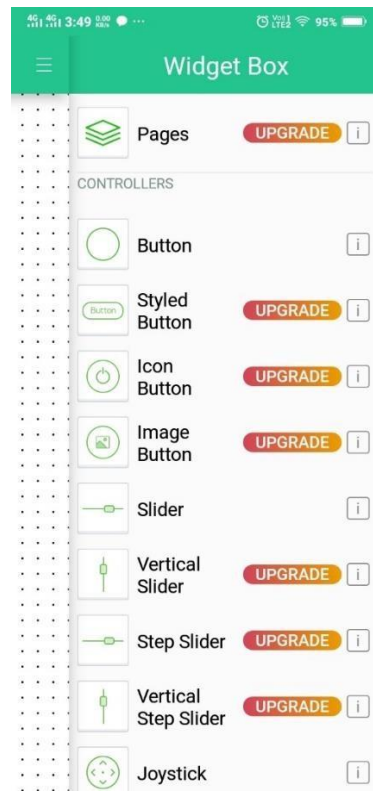
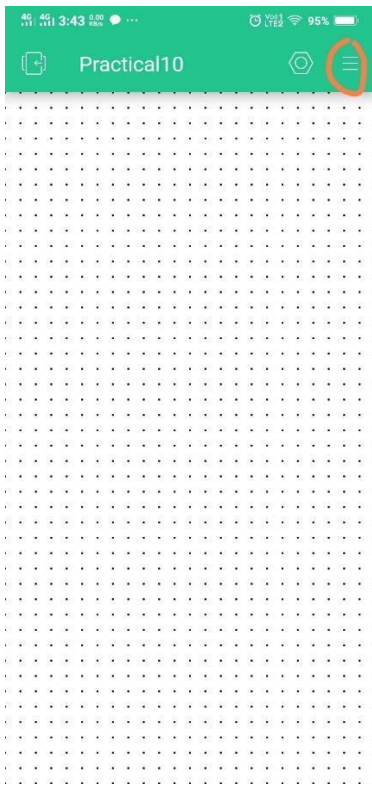
5. Click on Developer mode. Then you will be asked to enter a name for the template. Give a suitable template name and then click on continue.



The screenshot shows the 'New Template' screen of the Blynk app. The status bar at the top indicates 4G LTE, 3:37, 5:50 AM, and 96% battery. The app header is green with a back arrow and the text 'New Template'. The main content area has a light gray background. It features the title 'New Template' in bold, followed by the text 'A Template with default settings will be created. You can set it up at blynk.cloud'. Below this is a text input field labeled 'TEMPLATE NAME' with the placeholder text 'Enter...'. At the bottom, there is a dark blue button labeled 'Continue'.



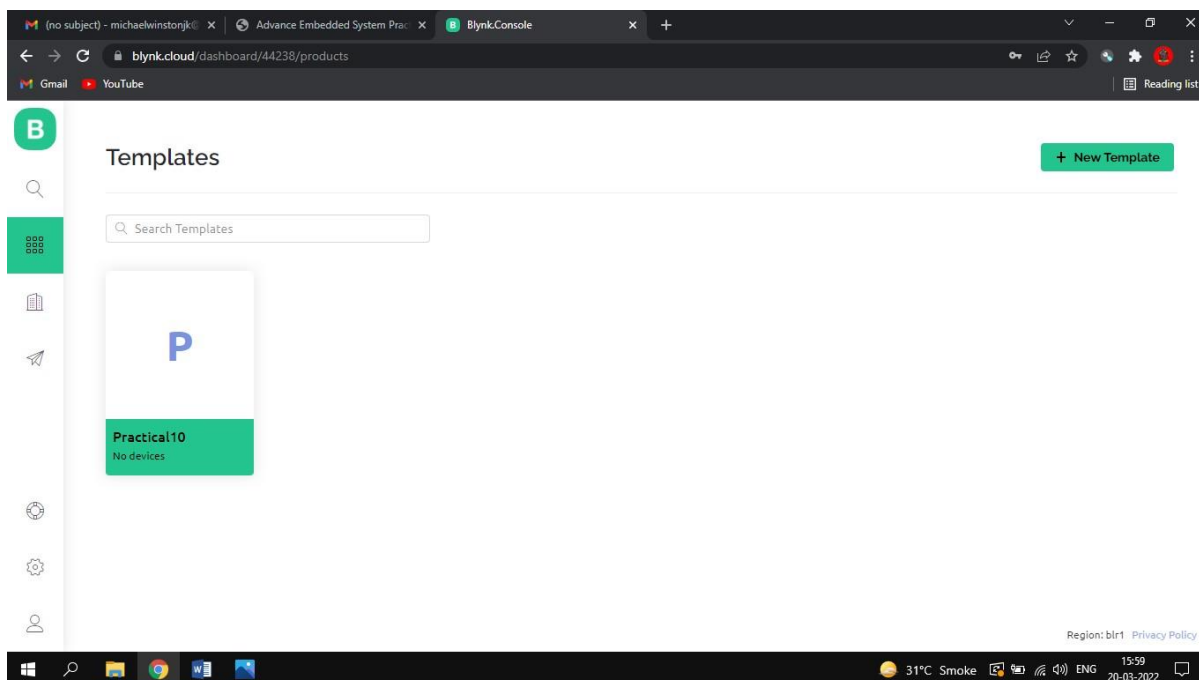
6. Now click on Prototype of Practical 10 a canvas will get opened. Then click on the circled part shown on the image and after that under controllers click on Button. A button with OFF written on it will be created on the canvas.



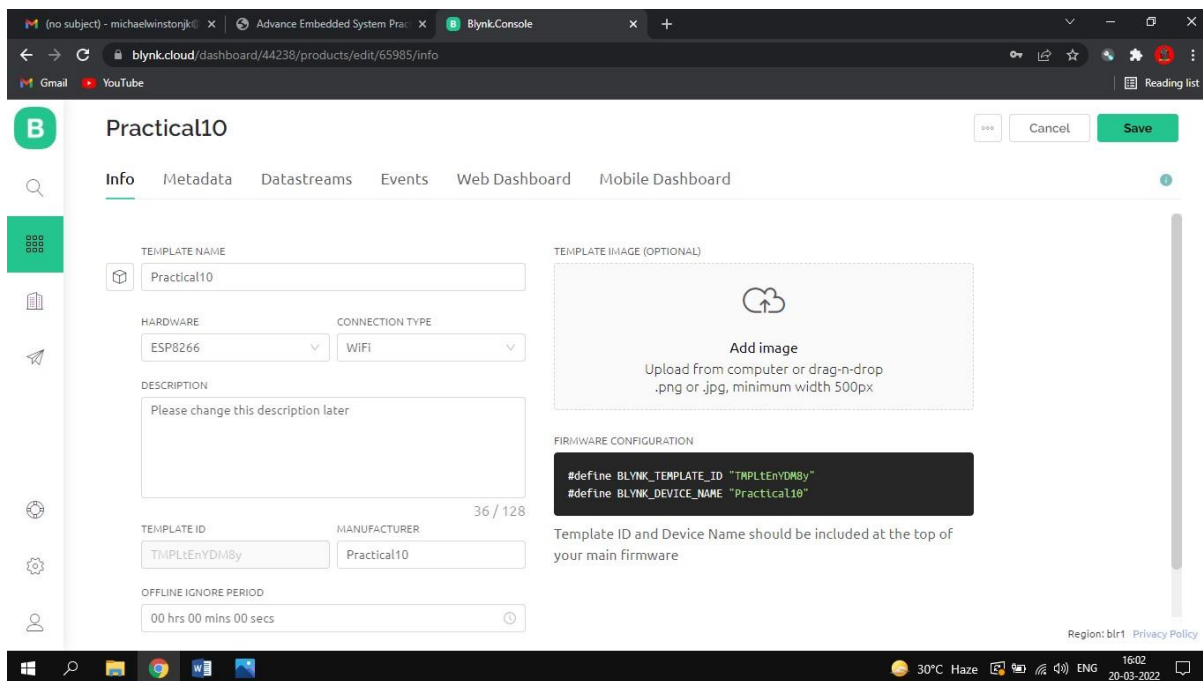
7. Now click on the button created it will open Button Settings. Enter title for the button and select the mode of the button to **SWITCH**.

The screenshot shows the 'Button Settings' screen in a mobile app. At the top, there's a green header with a back arrow, the title 'Button Settings', and an info icon. Below this is a section for 'LED' with a grey button labeled 'LED'. The next section is 'TITLE ALIGNMENT' with two icons: a left-aligned one (selected) and a right-aligned one. The 'DATASTREAM' section has a dropdown menu labeled 'Select Data Stream' with a checkmark icon. The 'MODE' section has two buttons: 'PUSH' (grey) and 'SWITCH' (dark blue, selected). The 'ON/OFF LABELS' section has two buttons: 'OFF' (grey) and 'ON' (grey). The 'FONT SIZE' section has a slider with a dot in the middle, labeled 'Medium'. The 'DESIGN' section has a 'TEXT' label and a blue semi-circle icon.

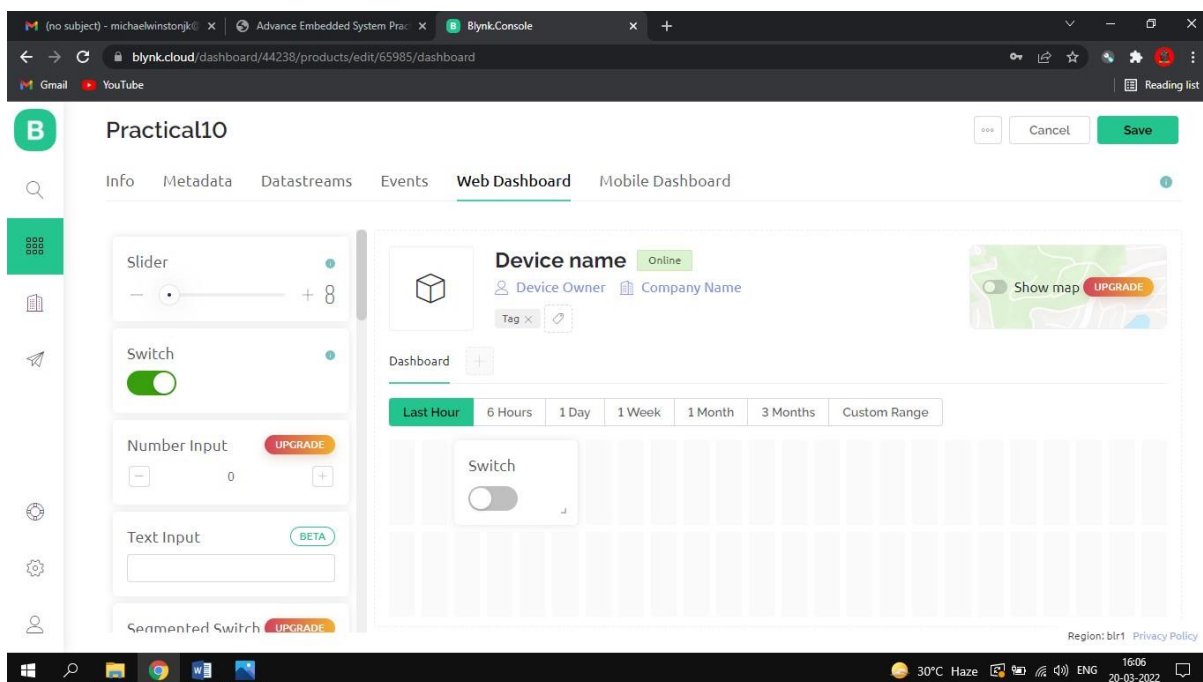
8. Now, Login to the desktop site of blynk i.e blynk cloud and then go to Templates section there you can see the template which we created on our mobile is visible click on it.



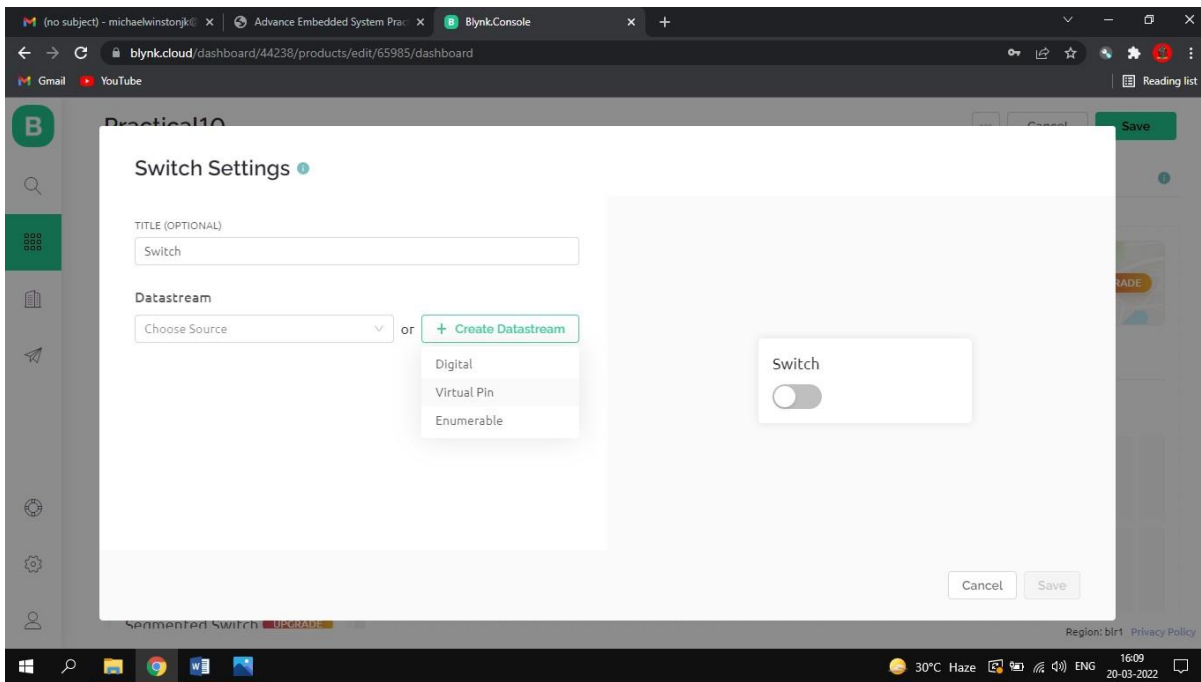
9. Now click on edit and change the **HARDWARE** to **esp8266** by selecting esp8266 from the list and then click on save.



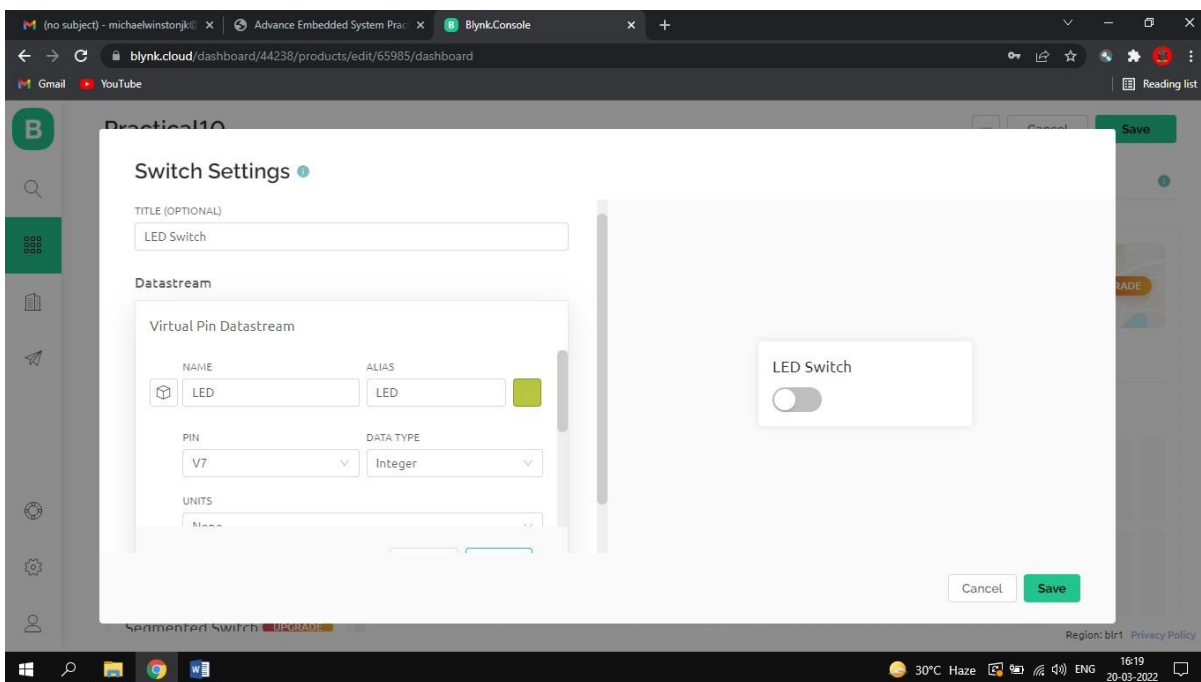
10. Now head towards the **Web Dashboard** of our template and then click on edit. After that drag and drop **Switch** widget from the widgets present on left of the page to the canvas.



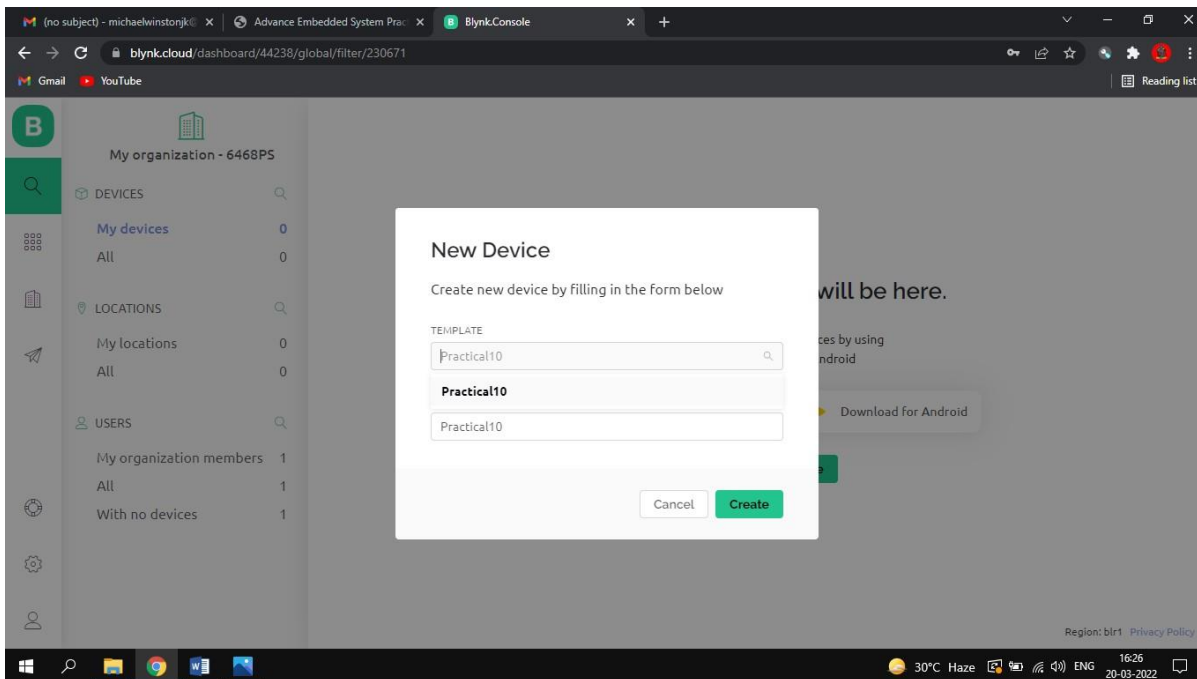
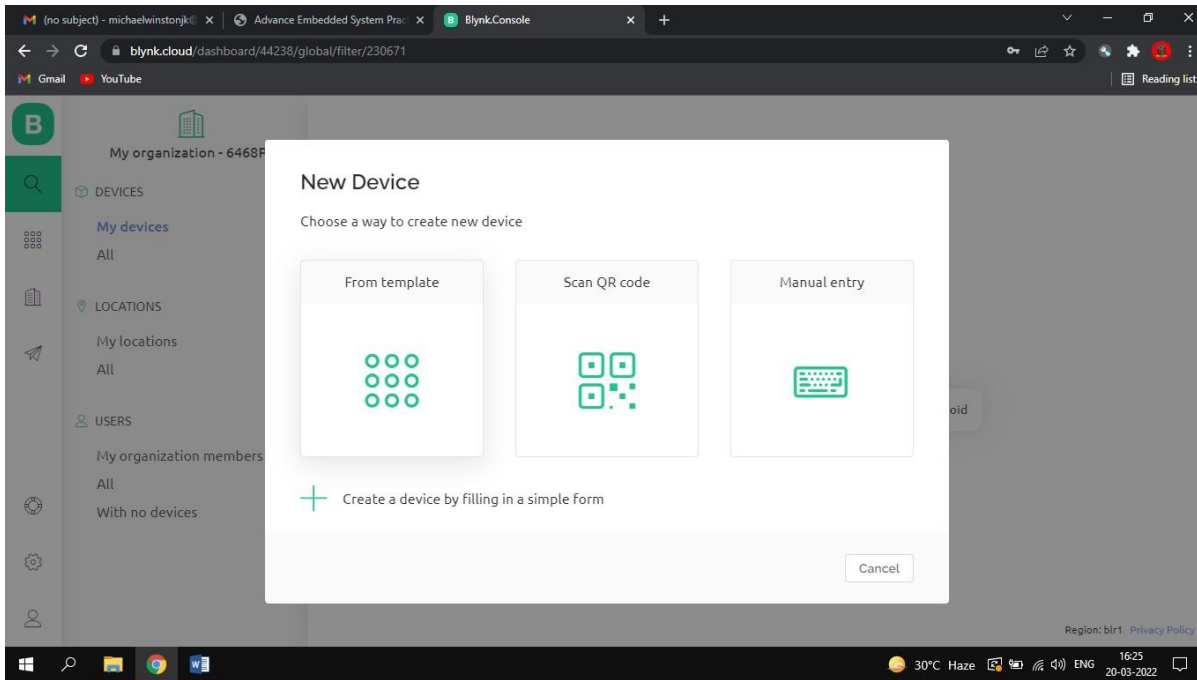
11. Now click on settings button of the switch widget it will open Switch Settings in that click on **Create Datastream** select **Virtual Pin**.



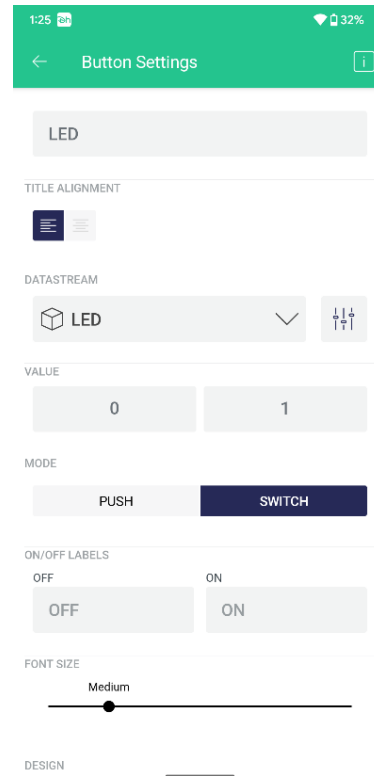
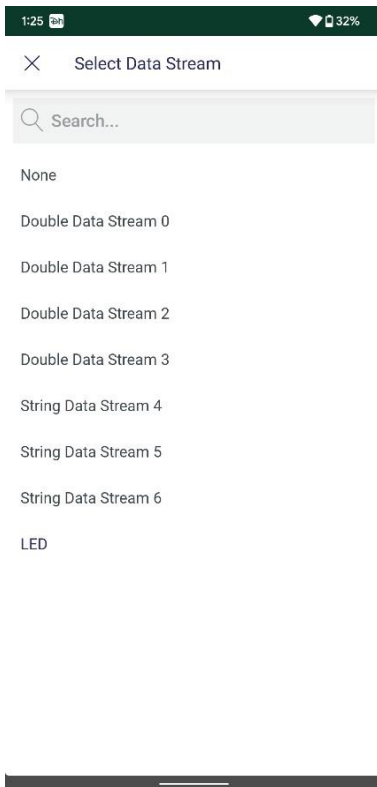
12. Now enter Title and name in Virtual Pin Destination and then click on create and after that click on save.



13. Now click on search on top left side and then click on **My devices**. Then click on **New Device** and click on **From Template** after that select template to Practical 10 or any suitable name given by you and then click on **Create**.



14. After that go to Blynk Iot App and click on the LED Button created on the template. Then click on Select DataStream. A list of Datastream appears in that select 'LED'.



With this Blynk Iot app configuration is done.

Uploading Firmware:

1. Download following zip file:-

https://github.com/blynkkk/blynk-library/releases/download/v0.6.1/Blynk_Release_v0.6.1.zip

2. Extract this file → we can see 2 folder **tools** and **libraries**

3. The content of tools will be copied and pasted on C:\Program Files (x86)\Arduino\tools

4. The content of libraries will be copied and pasted on C : \ Program Files (x86) \ Arduino \ libraries

Arduino IDE Setup:

Open arduino IDE and perform following configuration –

1. File → Preferences → In **Additional Boards Manager** text box enter:
https://arduino.esp8266.com/stable/package_esp8266com_index.json →
OK
2. Tools → Boards → Board Manager → Search for **esp8266 by ESP8266 community 2.6.3** → Install
3. Tool → Board → Select NodeMCU
4. Tools → Select COM port for communication

Code:

```
#define
BLYNK
_PRINT
Serial
#include
<ESP8266
WiFi.h>
#include <BlynkSimpleEsp8266_SSL.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon) → Auth
Tokens → Copy allchar auth[] =
"YourAuthToken";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName"; // Wi-Fi Name char pass[] = "YourPassword"; //
Wi-Fi Password
void setup()
{
  // Debug
  console
  Serial.begin
  (9600);
  Blynk.begin
  (auth, ssid,
  pass);
}
void loop()
{
  Blynk.run();
}
```

Note: - 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.

Now compile and Run the code.

Output:

Click the button from Blynk Iot app to switch ON and OFF the LED. We can test from remotely operating.