Internet of Things

Lab Assignment

Submitted by: Abhi Jain

Enrollment: 0801CS181003

Solutions

1) ESP32 is the name of the chip that was developed by Espressif Systems. This provides Wi-Fi (and in some models) dual-mode Bluetooth connectivity to embedded devices. While ESP32 is technically just the chip, modules and development boards that contain this chip are often also referred to as "ESP32" by the manufacturer. The ESP32 chip has a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, with a clock rate of over 240 MHz.

Summary

Processors – The ESP32 uses a Tensilica Xtensa 32-bit LX6 microprocessor. This uses 1 or 2 cores (*all chips in the series are dual-core, except the ESP32-S0WD). The clock frequency reaches up to 240MHz and it performs up to 600 DMIPS (Dhrystone Million Instructions Per Second). Moreover, its low power consumption allows for ADC conversions, computation, and level thresholds, all while in deep sleep mode.

Wireless connectivity – The ESP32 enables connectivity to integrated Wi-Fi through the 802.11 b/g/n/e/i/. Moreover, dual-mode Bluetooth is made possible with the v4.2 BR/EDR and features Bluetooth Low Energy (BLE).

Memory – Internal memory for the ESP32 is as follows – ROM: 448 KiB (for booting/core functions), SRAM: 520 KiB (for data/instructions), RTC fast SRAM: 8 KiB (for data storage/main CPU during boot from sleep mode), RTC slow SRAM: 8 KiB (for co-processor access during sleep mode), and eFuse: 1 KiBit (256 bits used for the system (MAC address and chip configuration) and 768 bits reserved for customer applications). Moreover, two of the ESP32 chips – ESP32-D2WD and ESP32-PICO-D4 – have internally connected flash. The others are as follows: 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips), 2 MiB (ESP32-D2WD chip), and 4 MiB (ESP32-PICO-D4 SiP module).

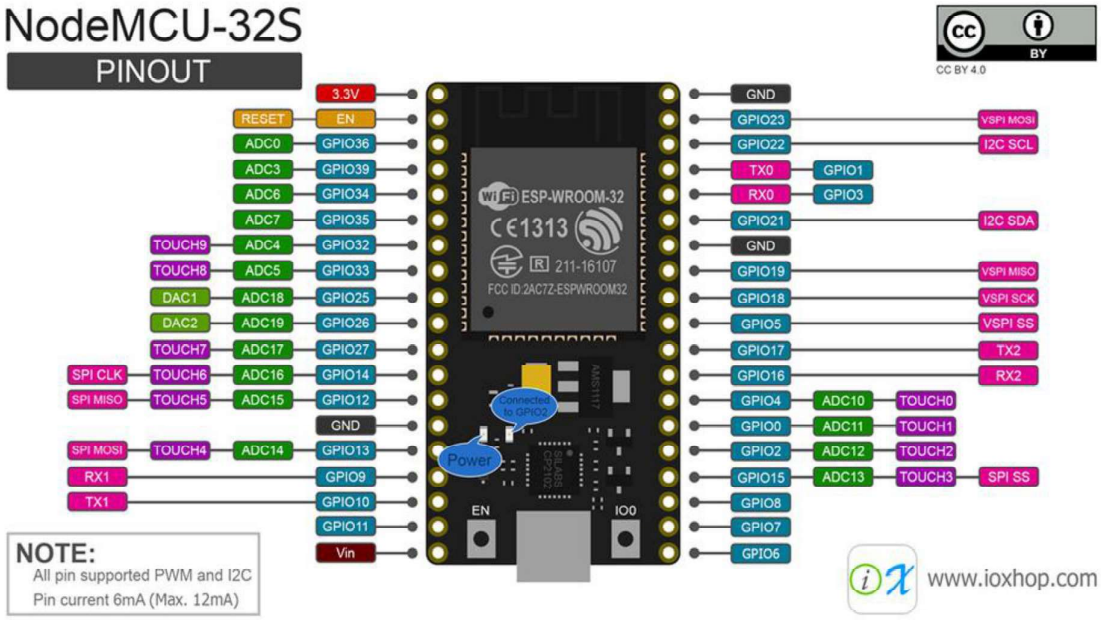
External Flash and SRAM – ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. It accesses the external QSPI flash and SRAM through high-speed caches.

Security – IEEE 802.11 standard security features are all supported, including WFA, WPA/WPA2 and WAPI. Moreover, ESP32 has a secure boot and flash encryption.

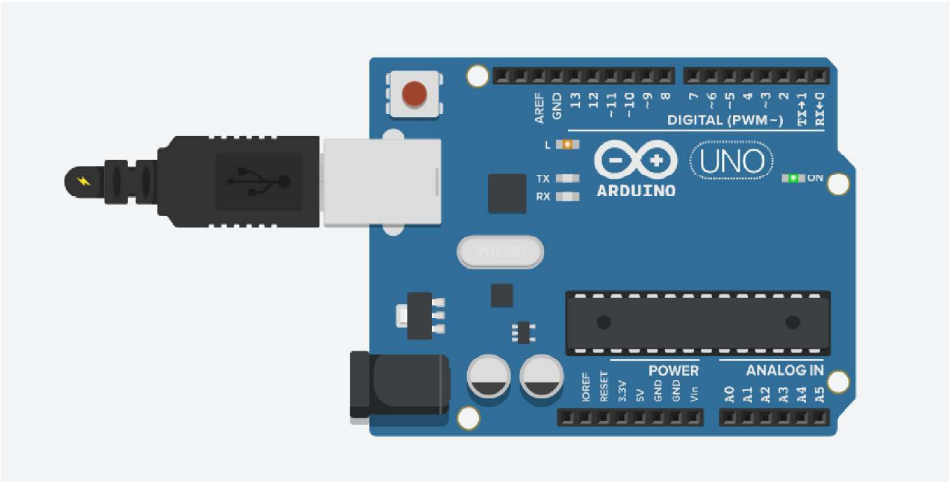
Inbuilt sensors

1) Hall Effect sensor
2) A total of 10 Touch sensors T0 to T9 are multiplexed with GPIO pins of ESP32.
3) Temperature sensor

Pinout



2) Inbuilt LED Blinking



Code

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Functions

- setup(): Used to initialize variables, pin modes, start using libraries, etc. It runs only once, after each powerup or reset of the Arduino board.
- loop(): Function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond.
- pinMode():Configures the specified pin to behave either as an input or an output
- digitalWrite(): Write a HIGH or a LOW value to a digital pin.
- delay(): Pauses the program for the amount of time (in milliseconds) specified as parameter.

3) Inbuilt temperature sensor

```
#ifdef __cplusplus
extern "C" {
#endif
uint8_t temprature_sens_read();
#ifdef __cplusplus
}
#endif
uint8_t temprature_sens_read();
void setup() {
  Serial.begin(115200);
}
void loop() {
  Serial.print(temprature_sens_read());
  Serial.println(" F");
  delay(5000);
}
```

Functions

- temprature_sens_read(): Used to read the inbuilt temperature sensor

ESP32 has on chip temperature sensor, this sensor is not usable to monitor external temperature, it is used to monitor its core temperature. Moreover, this sensor is no longer included in newer versions of ESP32.

4) Inbuilt hall sensor

```
void setup()
{
  Serial.begin(115200);
}
void loop() {
    int measurement = 0;
    measurement = hallRead();
    Serial.println(measurement);
    delay(1000);
}
```

Functions

- hallRead(): Reads and returns the value of the inbuilt hall effect sensor

When we bring a magnet in the field of hall effect sensor, the reading sharply rises

5) Touch pins

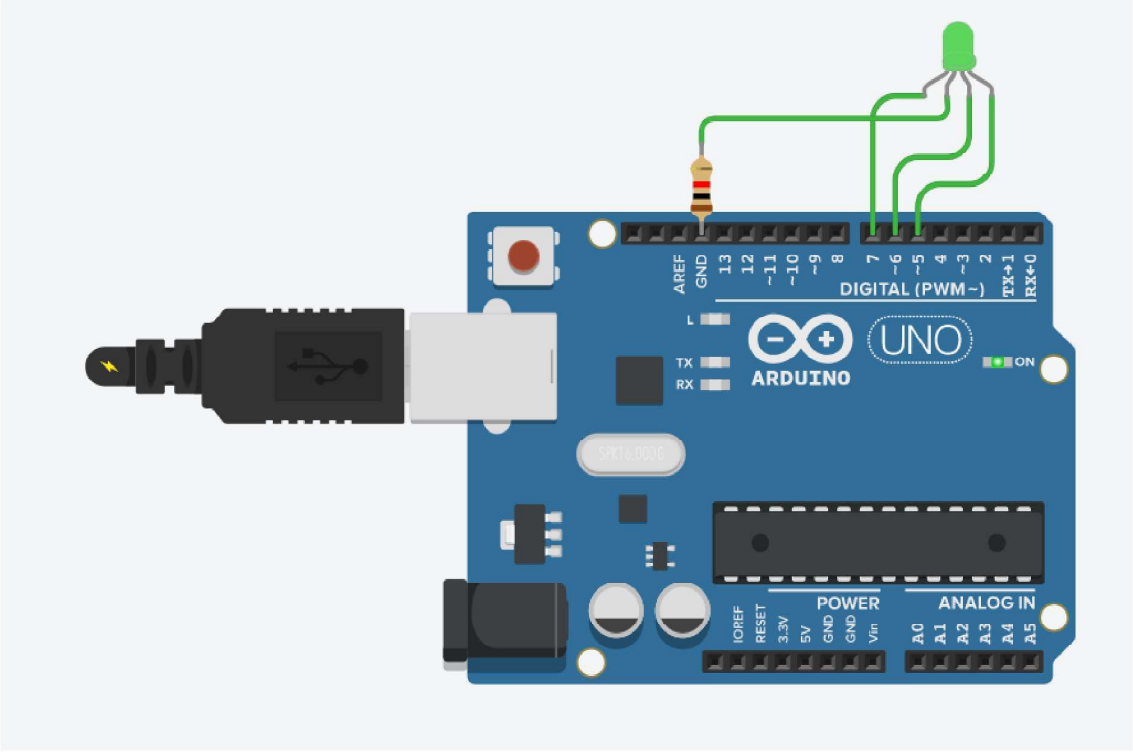
```
void setup()
{
  Serial.begin(115200);
  pinMode (ledPin, OUTPUT);
}
void loop()
{
  int touchValue = touchRead(4);
  if(touchValue < 20){
```

```
    digitalWrite(16, HIGH);
  }
  else{
    digitalWrite(16, LOW);
  }
  delay(500);
}
```

Functions

- touchRead(touch_sensor_pin_number): This function is used to read the value of touch sensor value associated with the touch pin. you only need to pass the name of touch pin to this function. For example, if you want to use touch pin four, you will simply use this function like this touchRead(4) .

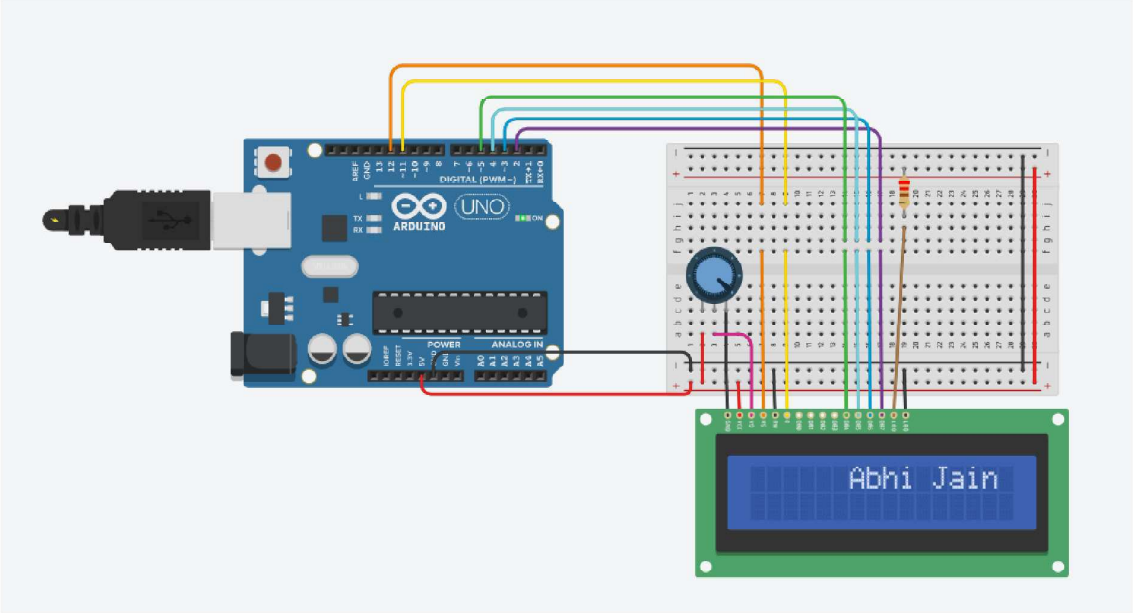
6) RGB LED




```
void setup()
{
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
}
```

```
void loop()
{
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  delay(1000);
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);
  digitalWrite(7, LOW);
  delay(1000);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
  delay(1000);
}
```

7) LCD Display



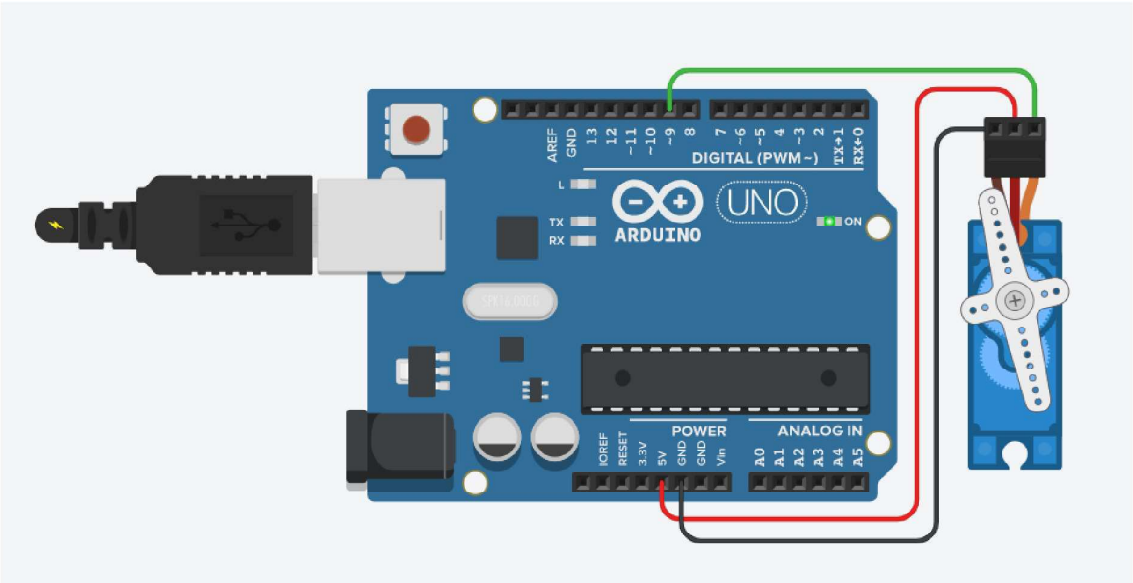

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
  lcd.begin(16, 2);
  lcd.print("Abhi Jain");
}

void loop() {
  for (int positionCounter = 0; positionCounter < 13; positionCounter++) {
```

```
    lcd.scrollDisplayLeft();
    delay(150);
  }
  for (int positionCounter = 0; positionCounter < 29; positionCounter++) {
    lcd.scrollDisplayRight();
    delay(150);
  }
  for (int positionCounter = 0; positionCounter < 16; positionCounter++) {
    lcd.scrollDisplayLeft();
    delay(150);
  }
  delay(1000);
}
```

Functions

- LiquidCrystal Library: This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).
- scrollDisplayLeft(): Scrolls the contents of the display (text and cursor) one space to the left.
- scrollDisplayRight(): Scrolls the contents of the display (text and cursor) one space to the right.

8) Servo Motor

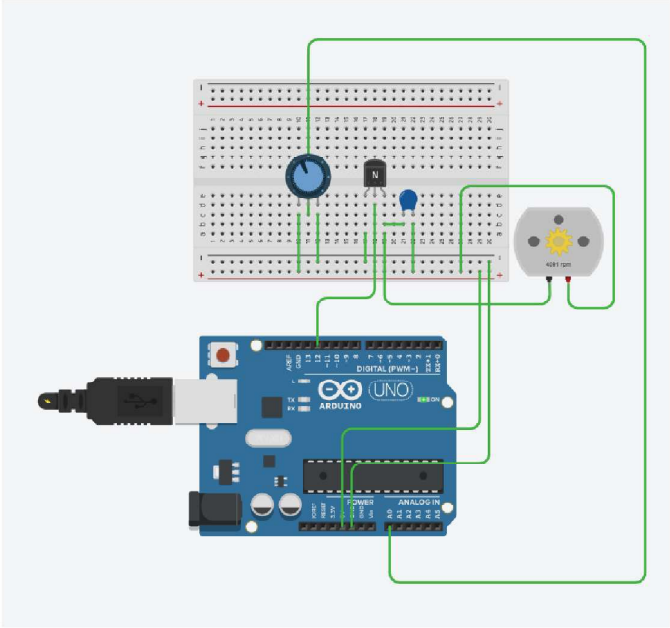


```
#include <Servo.h>
int pos = 0;
Servo servo;

void setup()
{
  servo.attach(9, 500, 2500);
}

void loop()
{
  for (pos = 0; pos <= 180; pos += 1) {
    servo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    servo.write(pos);
    delay(15);
  }
}
```

Functions

- Servo library: Allows Arduino/Genuino boards to control a variety of servo motors.
- write(): Writes a value to the servo, controlling the shaft accordingly

9) DC Motor



```
int c1 = 0, c2 = 0;
int buttonState = 0;

void setup()
{
  pinMode(12, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  c2= analogRead(A0);
  c1= 1024-c2;
  buttonState = digitalRead(2);
  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);
    digitalWrite(12, HIGH);
    delayMicroseconds(c1);
    digitalWrite(12, LOW);
    delayMicroseconds(c2);
  } else {
    digitalWrite(13, LOW);
    digitalWrite(12, LOW);
    delayMicroseconds(c1);
    digitalWrite(12, HIGH);
    delayMicroseconds(c2);
  }
}
```

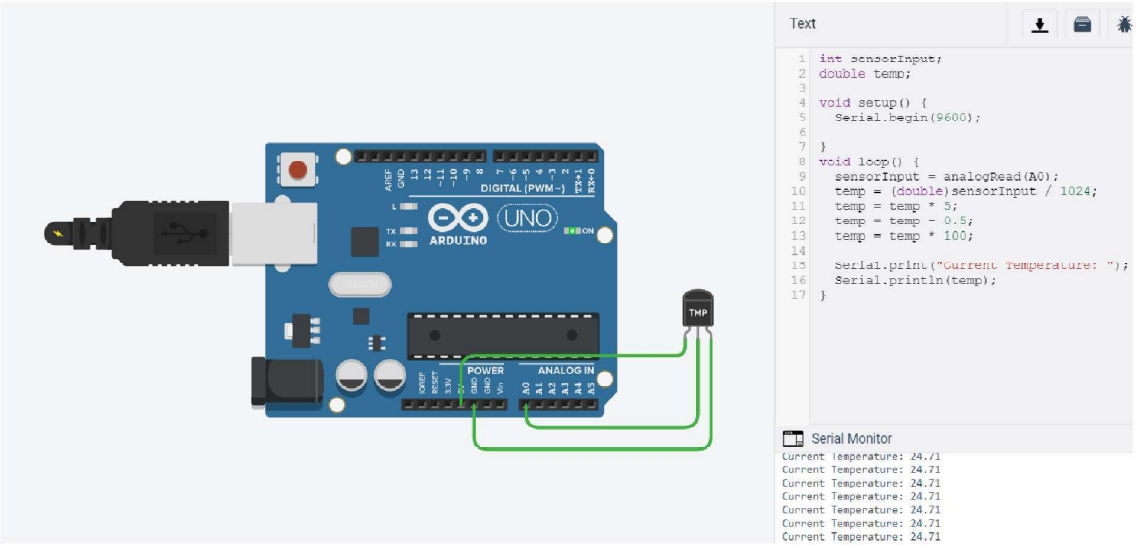
Functions

- analogRead(): Reads the value from the specified analog pin

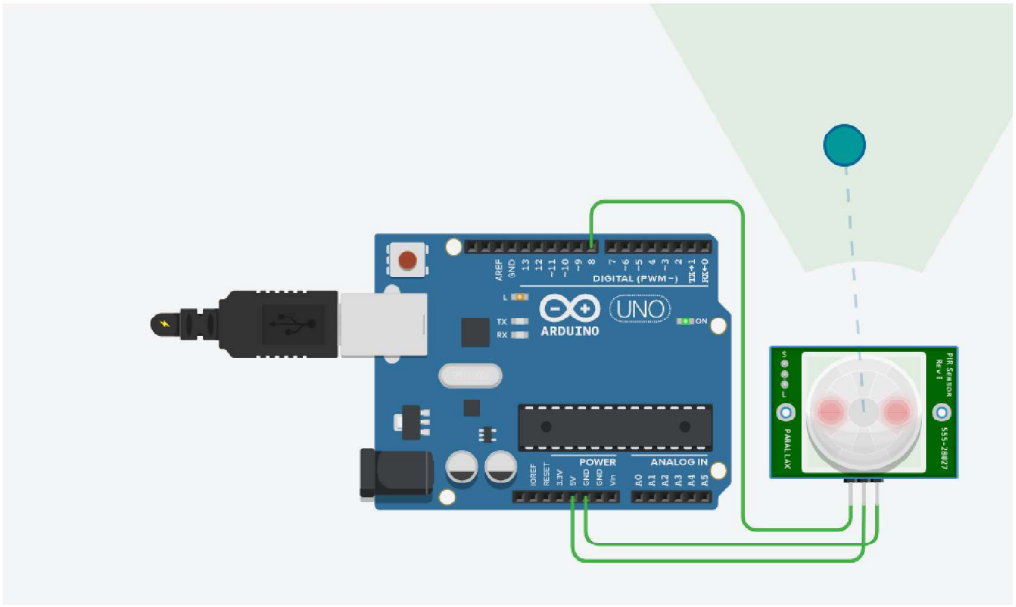
10) Temperature sensor

```
int sensorInput;
double temp;

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

}
void loop() {
```

```
  sensorInput = analogRead(A0);
  temp = (double)sensorInput / 1024;
  temp = temp * 5;
  temp = temp - 0.5;
  temp = temp * 100;

  Serial.print("Current Temperature: ");
  Serial.println(temp);
}
```

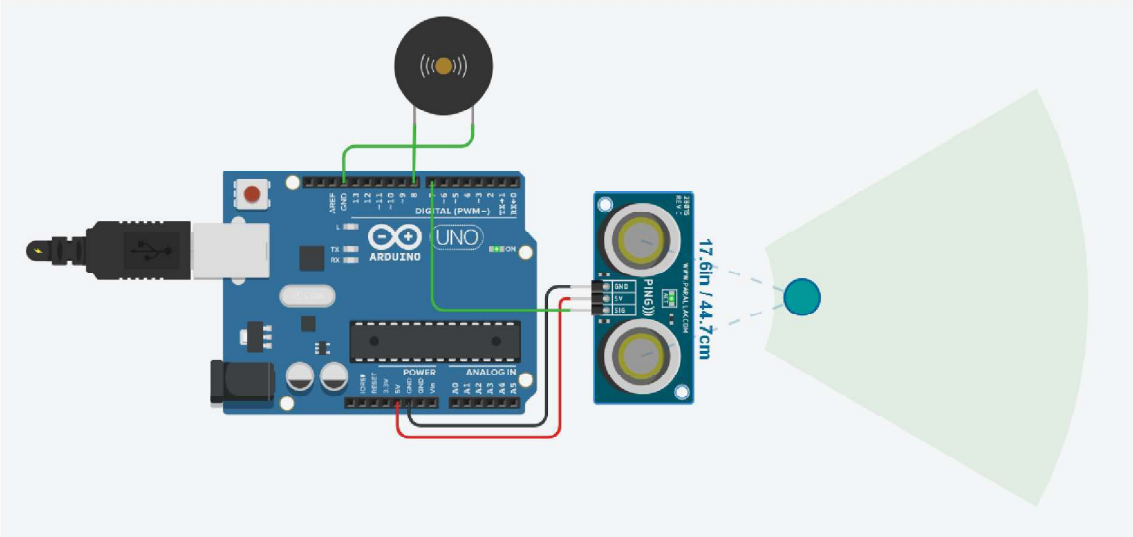


11) PIR



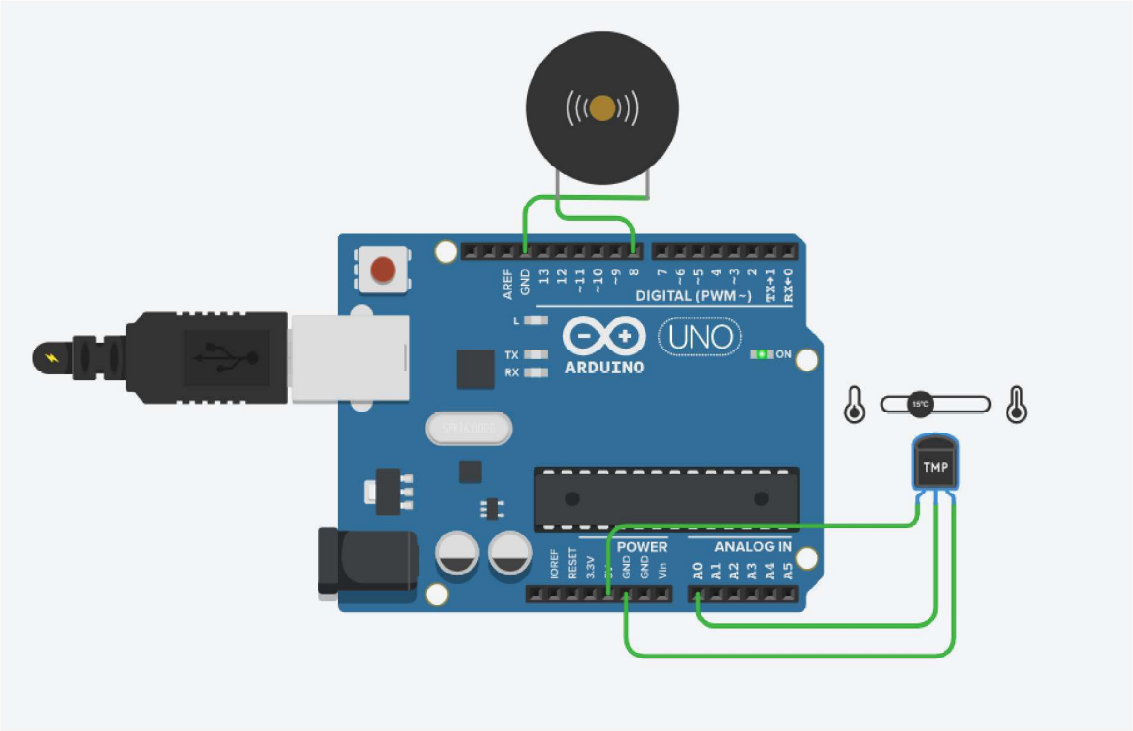
```
int val = 0;
int pirState = LOW;
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(8, INPUT);
}

void loop(){
  val = digitalRead(8);
  if (val == HIGH)
  {
    digitalWrite(13, HIGH);
    if (pirState == LOW) {
      pirState = HIGH;
    }
  }
  else
  {
    digitalWrite(13, LOW);
    if (pirState == HIGH){
      pirState = LOW;
    }
  }
}
```

12) Ultra-sonic



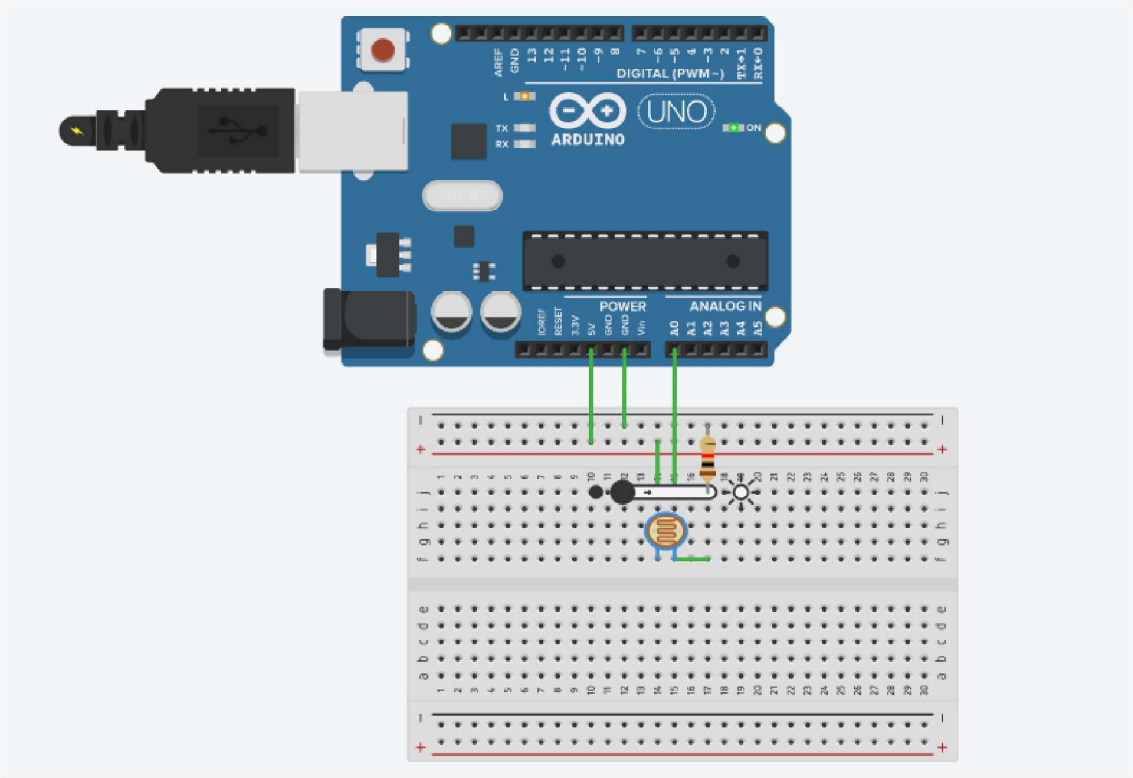
```
int cm = 0;
long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);
}

void setup()
{
  pinMode(8, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  cm = 0.01723 * readUltrasonicDistance(7, 7);
  if(cm < 50)
  {
       digitalWrite(8, HIGH);
  }
  else
  {
       digitalWrite(8, LOW);
  }
  Serial.print(cm);
  Serial.println("cm");
  delay(100);
}
```

13) Temperature Sensor



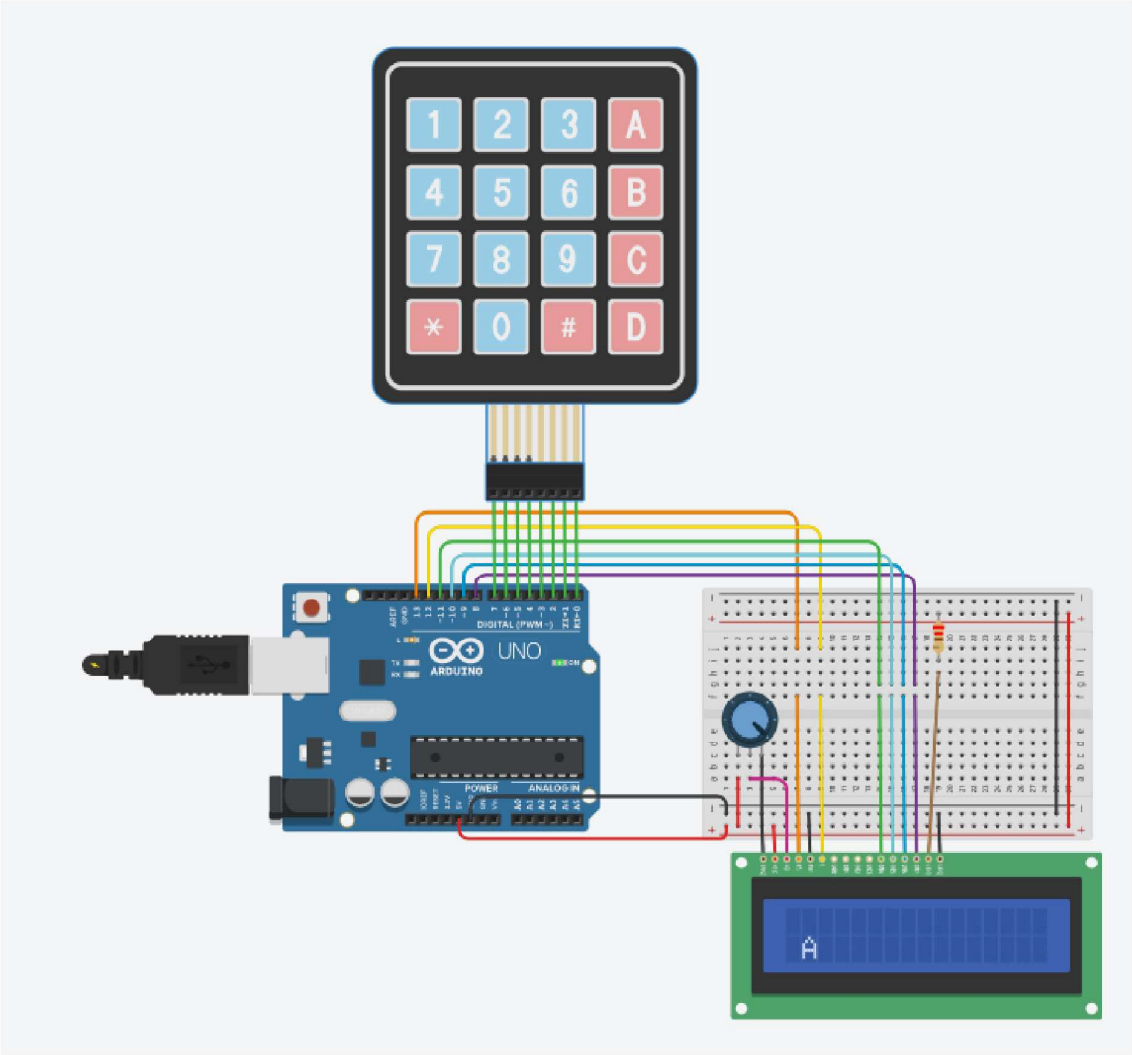

```
int sensorInput;
double temp;

void setup() {
  Serial.begin(9600);
  pinMode(8, OUTPUT);
}
void loop() {
  sensorInput = analogRead(A0);
  temp = (double)sensorInput / 1024;
  temp = temp * 5;
  temp = temp - 0.5;
  temp = temp * 100;
  Serial.print("Current Temperature: ");
  Serial.println(temp);
  if(temp < 20.00)
  {
        digitalWrite(8, HIGH);
  }
  else
  {
        digitalWrite(8, LOW);
  }
}
```

14) LDR Sensor




```
int sensorPin = A0;
int sensorValue = 0;

void setup()
{
        Serial.begin(9600);
        pinMode(13, OUTPUT);
}
void loop()
{
        sensorValue = analogRead(sensorPin);
        Serial.println(sensorValue);
        if(sensorValue < 200)
    {
        digitalWrite(13, HIGH);
    }
        else
    {
        digitalWrite(13, LOW);
    }
        delay(100);
}
```

15) Keypad

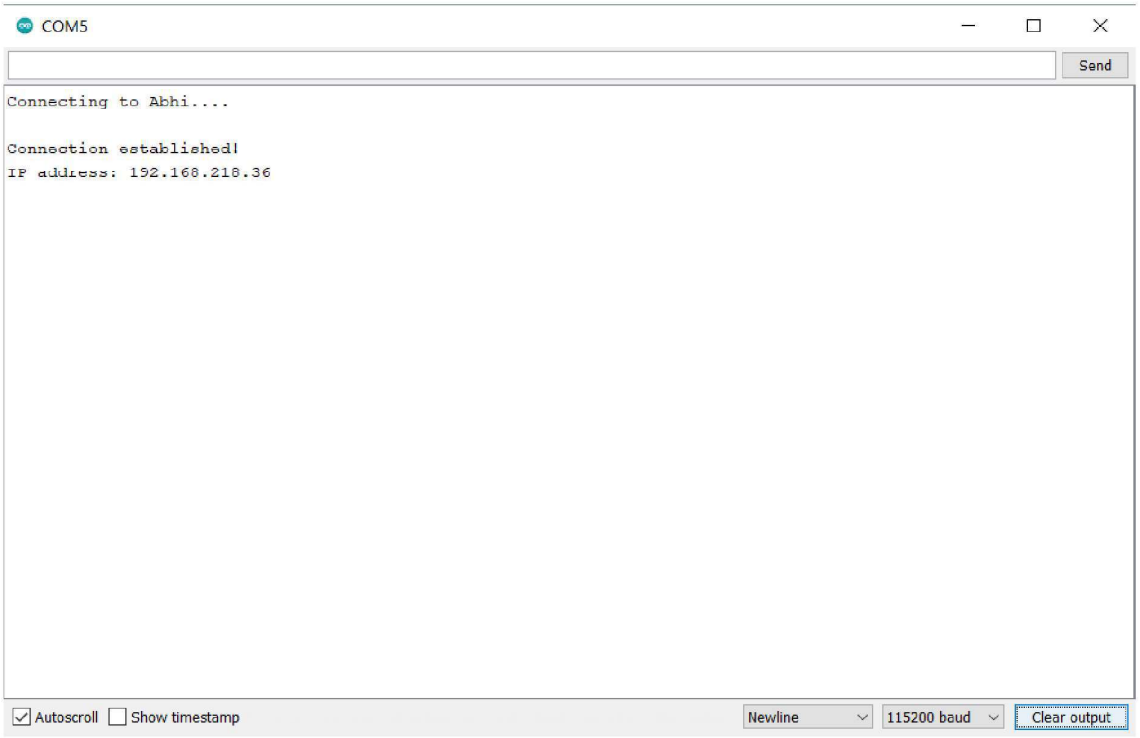



```
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <string.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

char keys[4][4] =
{
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
```

```
byte rowPins[4] = {7, 6, 5, 4};
byte colPins[4] = {3, 2, 1, 0};
Keypad mykeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
int i=0;

void setup() {
    lcd.begin(16,2);

}

void loop() {
  char key = mykeypad.getKey() ;
  if ((key != NO_KEY) && (i<4)){
       lcd.setCursor(1,1);
       lcd.print(key);
  }
}
```

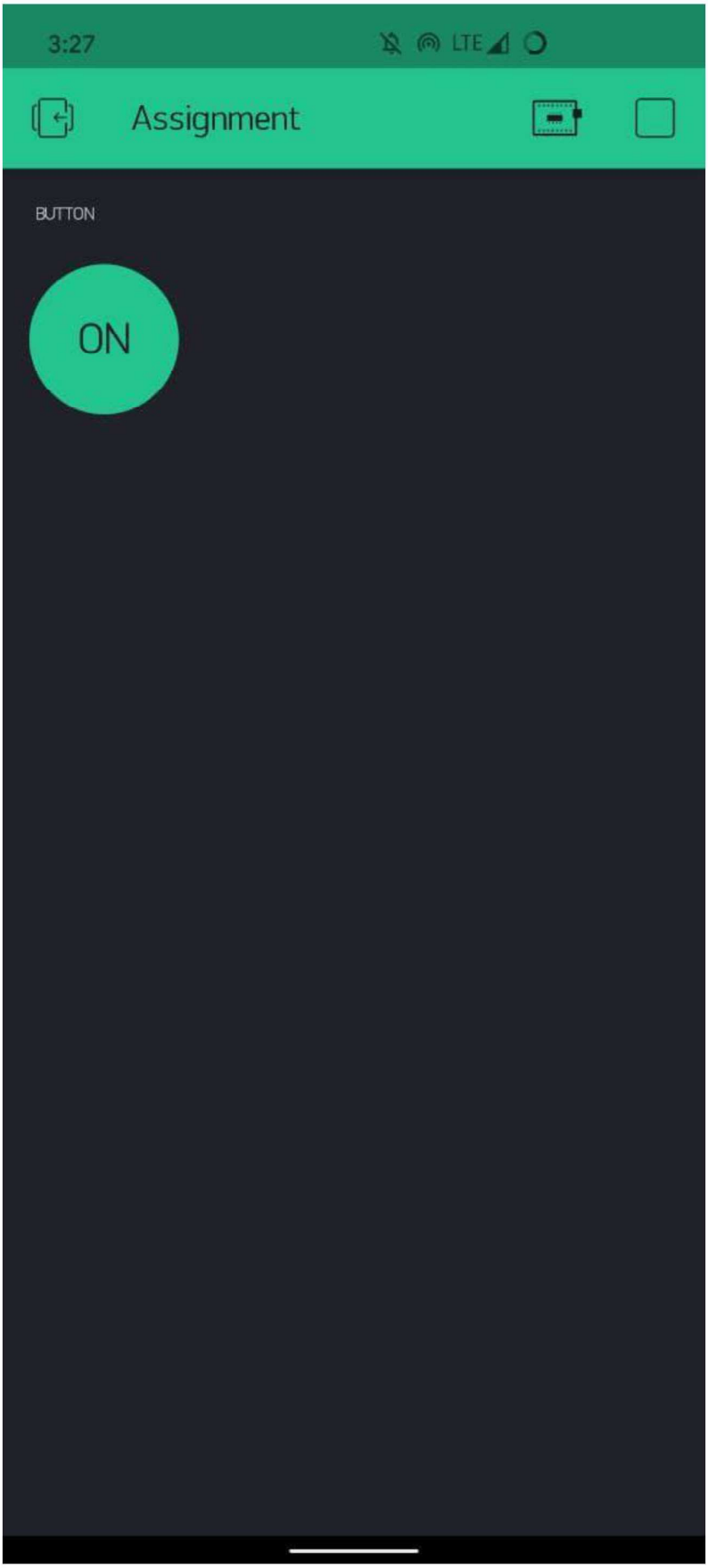
16) .

17) NodeMCU WiFi



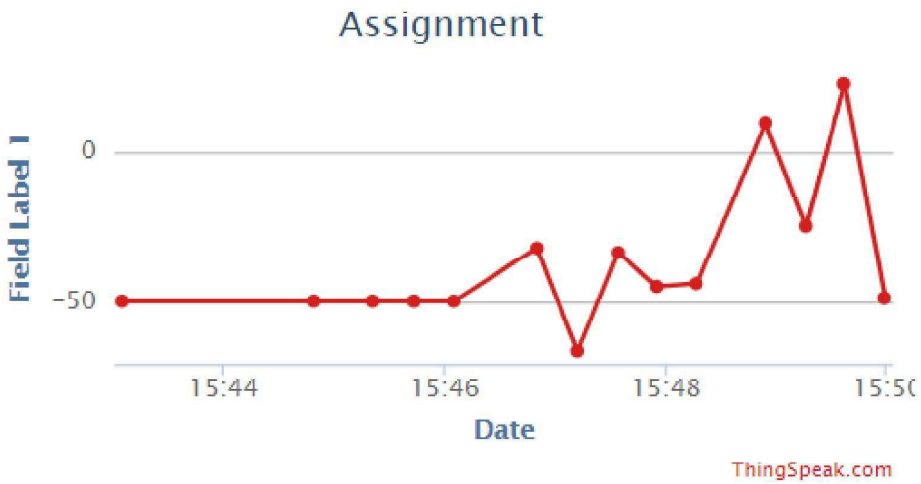
```
#include <ESP8266WiFi.h>
const char* ssid     = "Abhi";
const char* password = "abcd1234";

void setup() {
  Serial.begin(115200);
  delay(10);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to ");
  Serial.print(ssid);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print('.');
  }
  Serial.println('\n');
  Serial.println("Connection established!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() { }
```

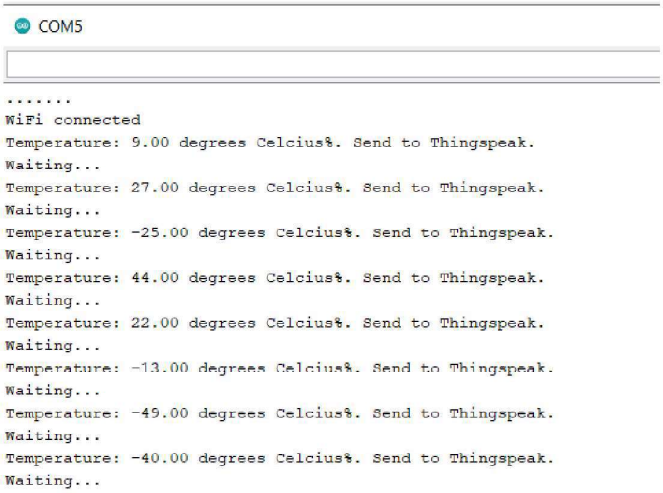
18) Blynk LED


3:27
LTE
Assignment
BUTTON
ON

```
#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[] = "4Y8Vp_xSMZt20-8xGUEVefLtj6BM78Wn";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Abhi";
char pass[] = "abcd1234";

void setup()
{
  // Debug console
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}

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

19) Thingspeak

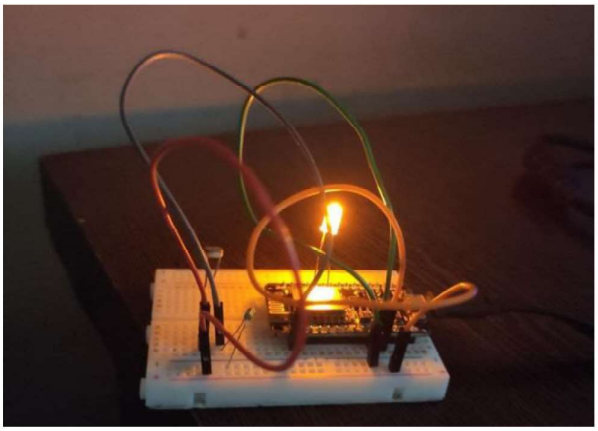


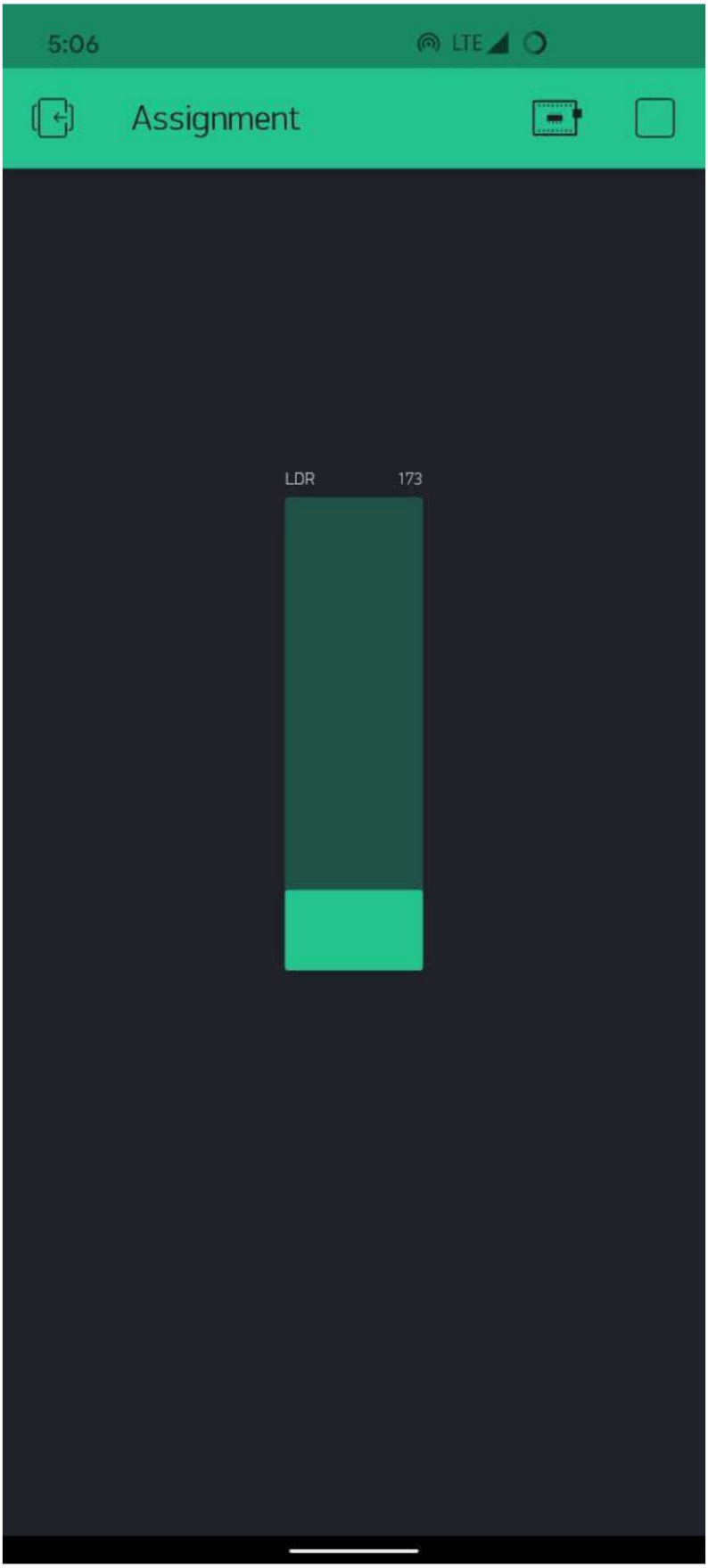



```
#include <ESP8266WiFi.h>
String apiKey = "1E2G6O5AMGRE08QD";

const char *ssid =  "Abhi";
const char *pass =  "abcd1234";
const char* server = "api.thingspeak.com";
double temp;

WiFiClient client;

void setup()
{
   Serial.begin(115200);
   delay(10);
   Serial.println("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, pass);
   while (WiFi.status() != WL_CONNECTED)
   {
     delay(500);
     Serial.print(".");
   }
   Serial.println("");
   Serial.println("WiFi connected");
}


void loop()
{
```

```
  temp = random(-50, 50);

  if (client.connect(server,80))
  {
     String postStr = apiKey;
     postStr +="&field1=";
     postStr += String(temp);
     postStr += "\r\n\r\n";

     client.print("POST /update HTTP/1.1\n");
     client.print("Host: api.thingspeak.com\n");
     client.print("Connection: close\n");
     client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
     client.print("Content-Type: application/x-www-form-
urlencoded\n");
     client.print("Content-Length: ");
     client.print(postStr.length());
     client.print("\n\n");
     client.print(postStr);

     Serial.print("Temperature: ");
     Serial.print(temp);
     Serial.print(" degrees Celcius");
     Serial.println("%. Send to Thingspeak.");
  }
  client.stop();
  Serial.println("Waiting...");
  delay(10000);
}
```

20) LDR, LED, Blynk




5:06
LTE
Assignment
LDR
173

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#define led D4
#define sensorPin A0
int sensorValue = 0;
char auth[] = "4Y8Vp_xSMZt20-8xGUEVefLtj6BM78Wn";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Abhi";
char pass[] = "abcd1234";

void setup()
{
  // Debug console
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  pinMode(led, OUTPUT);
}

void loop()
{
  Blynk.run();
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
  if(sensorValue < 400)
  {
    digitalWrite(led, HIGH);
  }
  else
  {
    digitalWrite(led, LOW);
  }
  Blynk.virtualWrite(V5, sensorValue);
  delay(100);
}
```