

## **Practical 1**

**Aim:** To implement following applications on Arduino board:

- (a)** Create (any) delay routine for buzzing LED(s).
- (b)** Toggle LED(s) by switching.

### **Components Required:**

- LED (Generic)
- Resistor 220 ohm
- Jumper Wires (Generic)
- Arduino Uno
- Breadboard (Generic)

### **Theory Background:**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

#### ➤ **Power USB**

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection.

#### ➤ **Power (Barrel Jack)**

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack.

#### ➤ **Voltage Regulator**

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

➤ **Crystal Oscillator**

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz

➤ **Arduino Reset**

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET.

➤ **Analog pins**

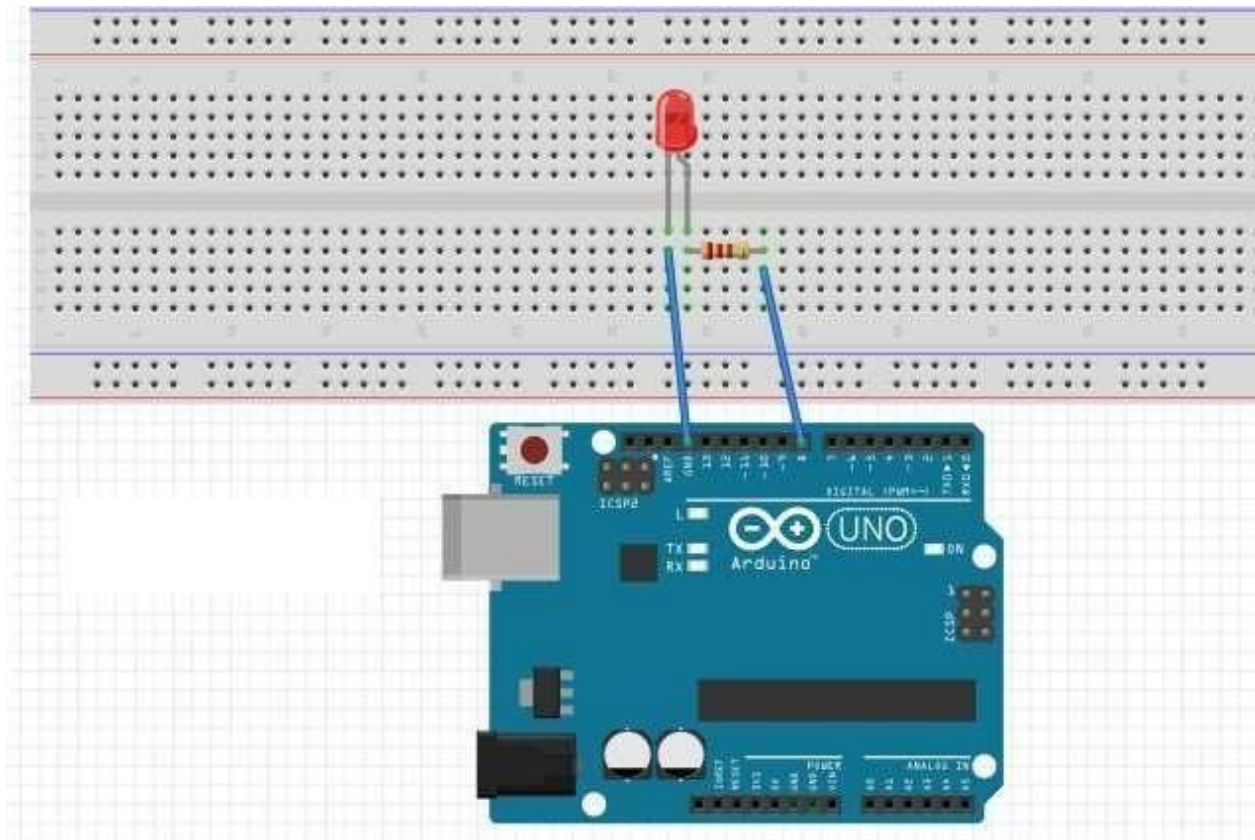
The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

➤ **Main microcontroller**

Each Arduino board has its own microcontroller. You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

➤ **AREF**

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

**Circuit Diagram:****Process Step:**

**Step 1:** Start the process

**Step 2:** Connect the components as shown in circuit diagram.

**Step 3:** Connect the Arduino board with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->Arduino Uno and the select port.

**Step 6:** Then enter the code in Arduino Software.

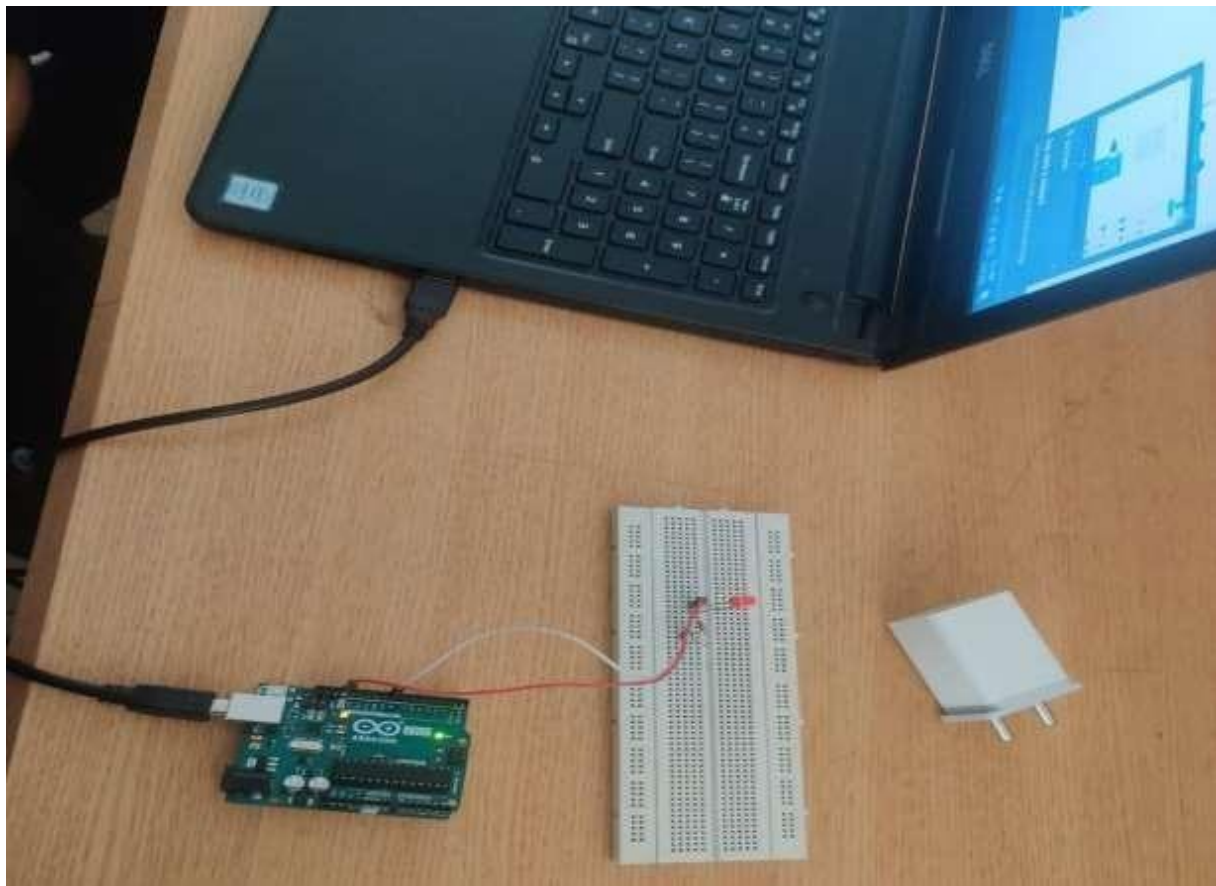
**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Arduino board and output will be displayed in the serial monitor.

**Step 9:** Stop the process.

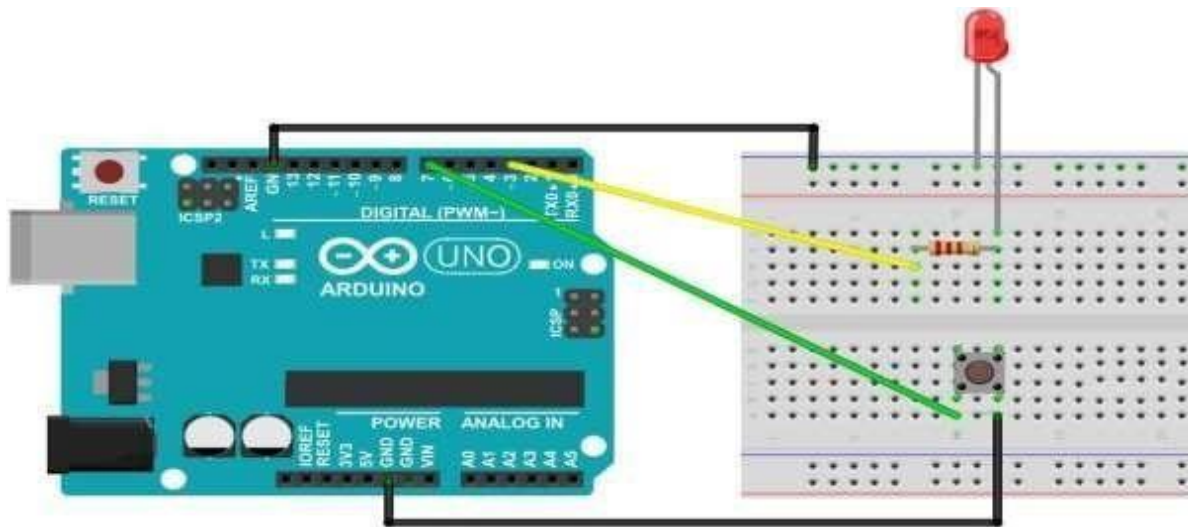
**Program code:****(a)**

```
int ledPin=8;
void setup()
{
  pinMode(ledPin,OUTPUT);
}
void loop()
{
  digitalWrite(ledPin,HIGH);
  delay(1000);
  digitalWrite(ledPin,LOW);
  delay(1000);
}
```

**Output:**

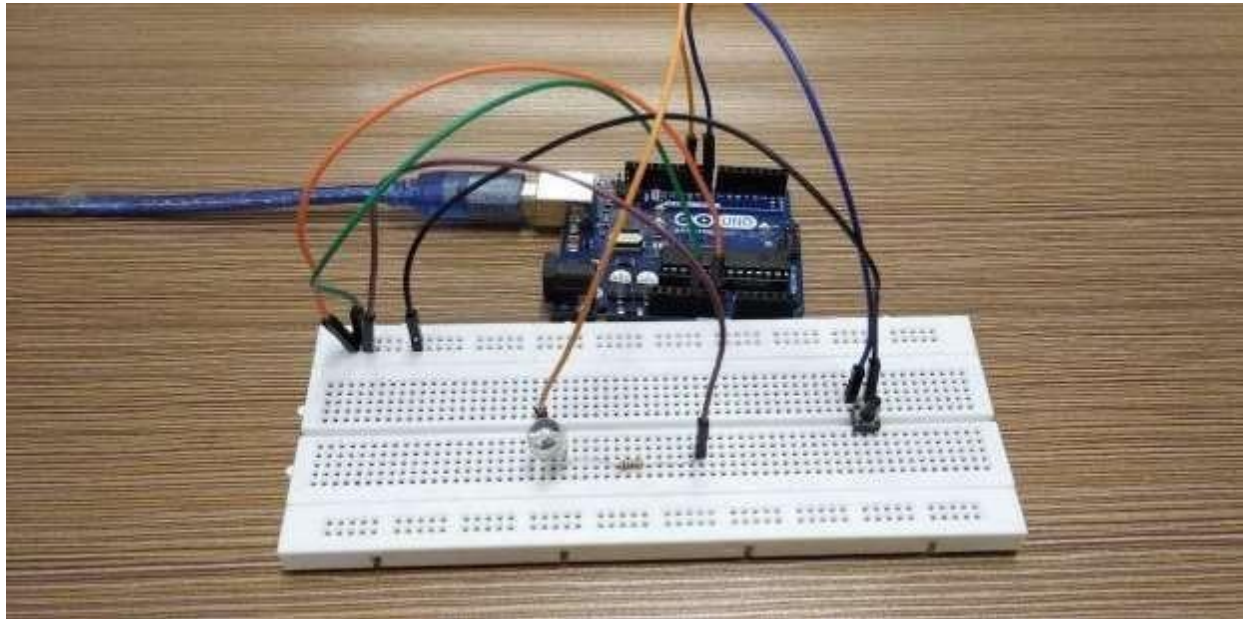
**Program code:****(b)**

```
const int BUTTON_PIN = 7;
const int LED_PIN = 3;
int ledState = LOW;
int lastButtonState;
int currentButtonState;
void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  currentButtonState = digitalRead(BUTTON_PIN);
}
void loop() {
  lastButtonState = currentButtonState;
  currentButtonState = digitalRead(BUTTON_PIN);
  if(lastButtonState == HIGH && currentButtonState == LOW) {
    Serial.println("The button is pressed");
    ledState = !ledState;
    digitalWrite(LED_PIN, ledState); }
}
```

**Circuit Diagram:**

---

**Output:**



**Observation:**

Learned about Arduino UNO. Also learned how to make an LED blink using Arduino IDE software.

## **Practical 2**

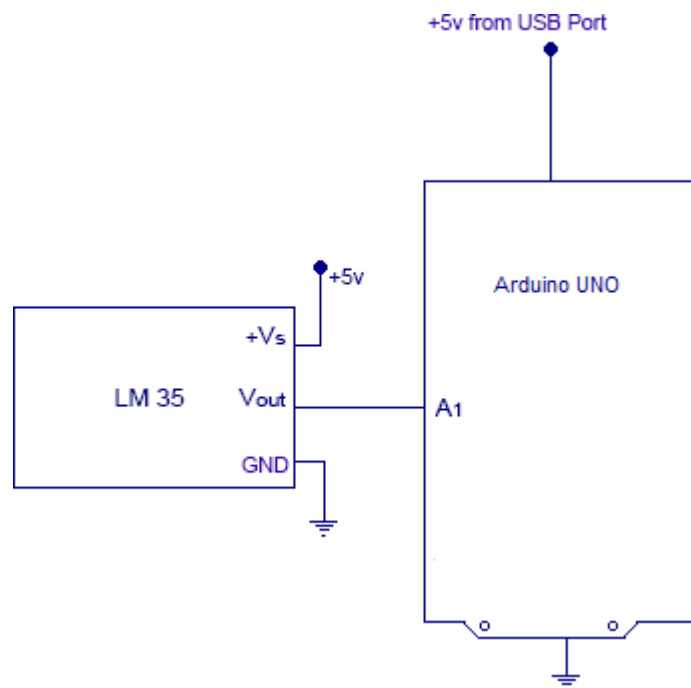
**Aim:** To interface a temperature sensor with Arduino and write a program to print temperature and humidity readings on LCD displays.

### **Components Required:**

- Temperature sensor LM 35
- Jumper wires and Bread Board
- Connectivity cable/ USB cable
- Arduino Uno
- 16\*2 LCD display

### **Theory Background:**

LM35 is an analog, linear temperature sensor whose output voltage varies linearly with change in temperature. LM35 is three terminal linear temperature sensor from National semiconductors. It can measure temperature from -55 degree Celsius to +150 degree Celsius. The voltage output of the LM35 increases 10mV per degree Celsius rise in temperature. LM35 can be operated from a 5V supply and the stand by current is less than 60uA. The pin diagram of LM35 and Arduino Uno is shown in the figure below.

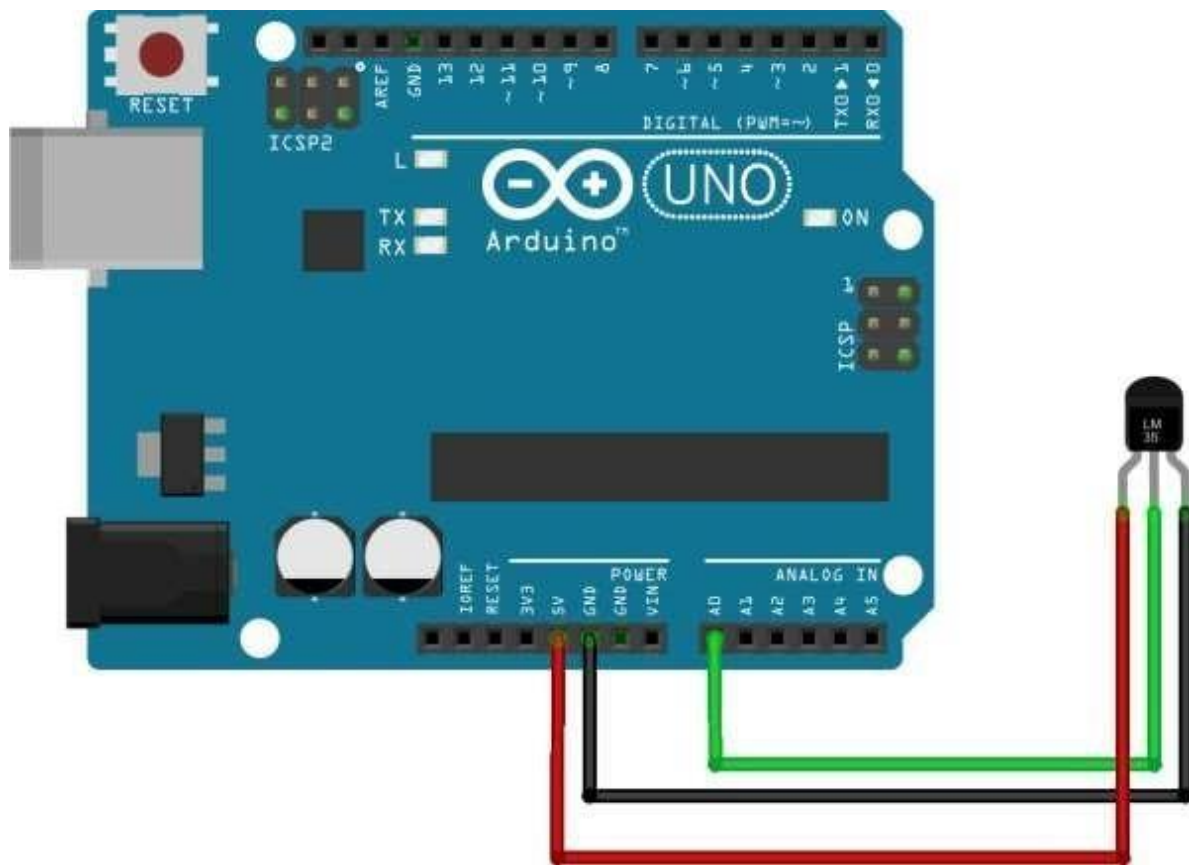




Lm35 Celsius/centigrade resolution is 10 mills volt. Means 10 mills volt represent one degree centigrade/Celsius. So if Lm35 outputs 100 mills volts the equivalent temperature in centigrade/Celsius will be  $100/10 = 10$  centigrade/Celsius. Arduino analog pin resolution is 1023 starting from 0. On +5 volts input it counts to 1023. So we need to perform following conversion while reading temprature.

```
val = analogRead(tempPin);  
float mv = ( val/1024.0)*5000;  
float cel = mv/10;
```

### Circuit Diagram:





**Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In Arduino board, connect VCC to power supply 5V and connect to ground as in PIN gnd and connect analog read A1 pin to read input using jumper wires.

**Step 3:** Connect the Arduino board with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->Arduino Uno and the select port.

**Step 6:** Then enter the code in Arduino Software.

**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Arduino board and output will be displayed in the serial monitor.

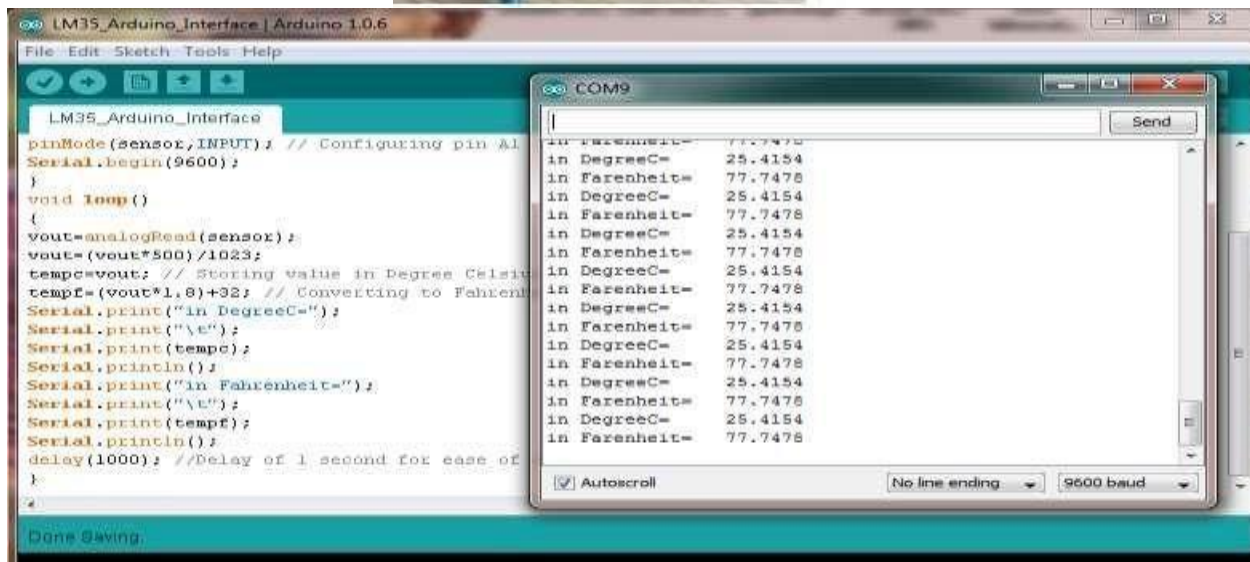
**Step 9:** Stop the process.

**Program code:**

**(a)**

```
const int sensor=A1;
float tempc;
float tempf;
float vout;
void setup(){
  pinMode(sensor, INPUT);
  Serial.begin(9600);
}
void loop() {
  vout=analogRead(sensor);
  float mv = (vout/1024.0)*5000;
  float tempc = mv/10;
  tempf=(vout*1.8)+32;
  Serial.print("in DegreeC=");
  Serial.print("\t");
  Serial.print(tempc);
  Serial.println();
  Serial.print("in Fahrenheit=");
```

```
Serial.print("\t");  
Serial.print(tempf);  
Serial.println();  
delay(1000);  
}
```

**Output:****Observation:**

Learned about how to make temperature sensor using Arduino Uno and print the result in the Arduino IDE.

## Practical 3

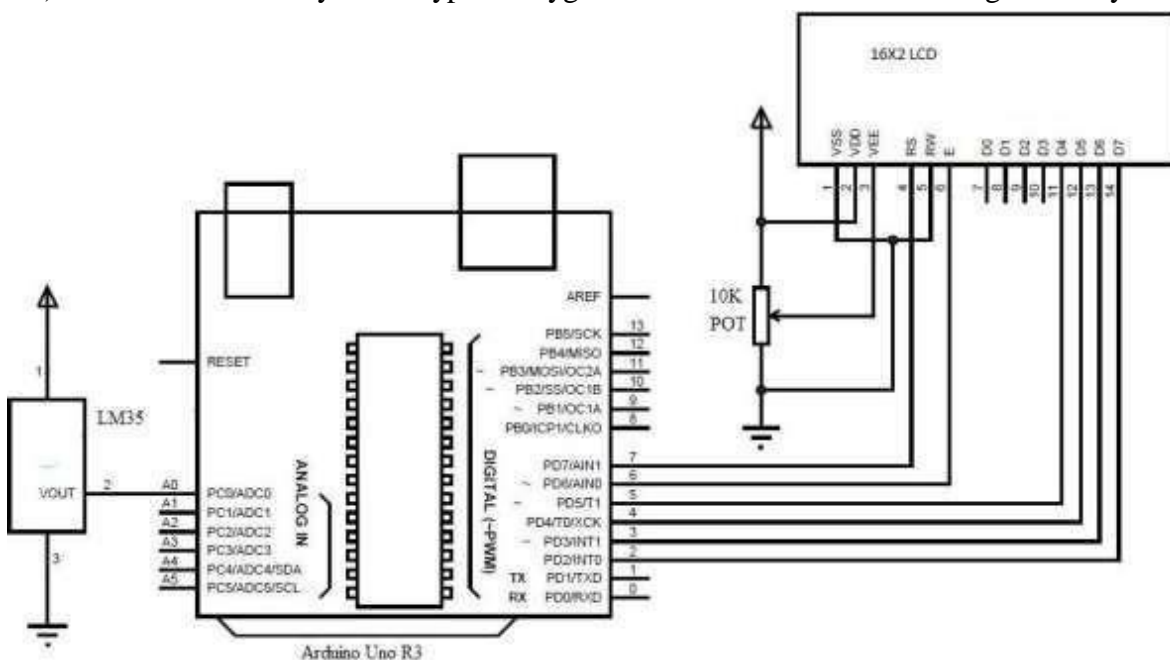
**Aim:** To implement an application which senses temperature and humidity if it is above the predefined threshold value then application should glow the LED for notification.

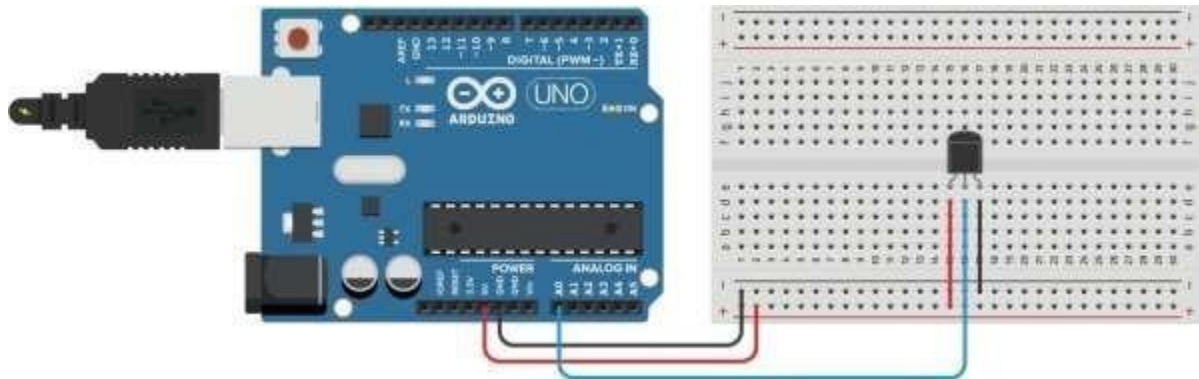
### Components Required:

- Arduino Uno
- LED (Generic)
- DGT11 Temperature and Humidity Sensor
- Jumper wires
- Resistor 330 ohm
- Breadboard

### Thory Background:

The resistance temperature sensor (Pt100) is also used for heating the humidity sensing element in order to recover or detect each deterioration mode. Humidity sensors work by detecting changes that alter electrical currents or temperature in the air. There are three basic types of humidity sensors: capacitive, resistive and thermal. All three types will monitor minute changes in the atmosphere in order to calculate the humidity in the air. A humidity monitoring device is called a hygrometer. Hygrometers may be designed for indoor or outdoor humidity monitoring use (or both). Below is a summary of the types of hygrometers available for measuring humidity.



**Circuit Diagram:****Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In Arduino board, connect VCC to power supply 5V and connect to ground as in PIN gnd and connect analog read A1 pin to read input using jumper wires.

**Step 3:** Connect the Arduino board with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->Arduino Uno and the select port.

**Step 6:** Then enter the code in Arduino Software.

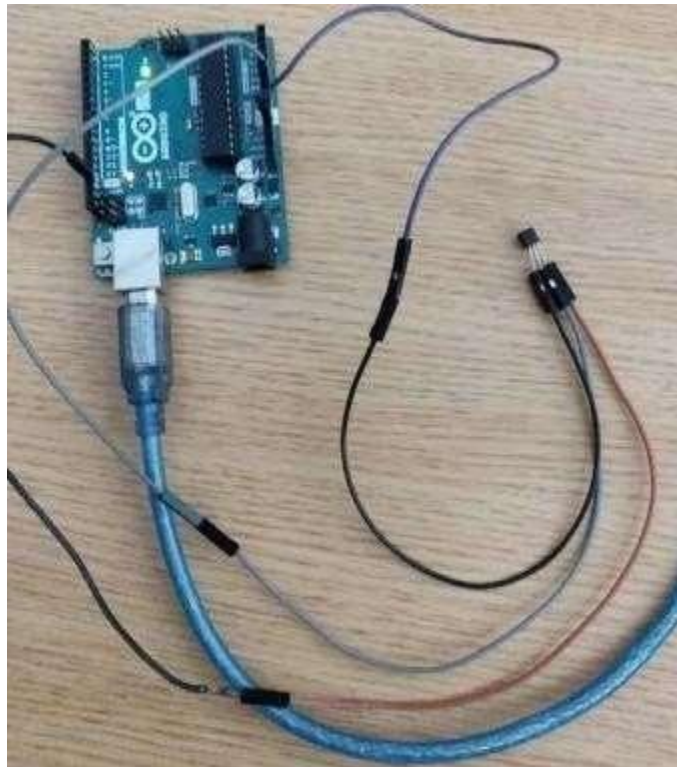
**Step 7:** Compile the code in Arduino Software.

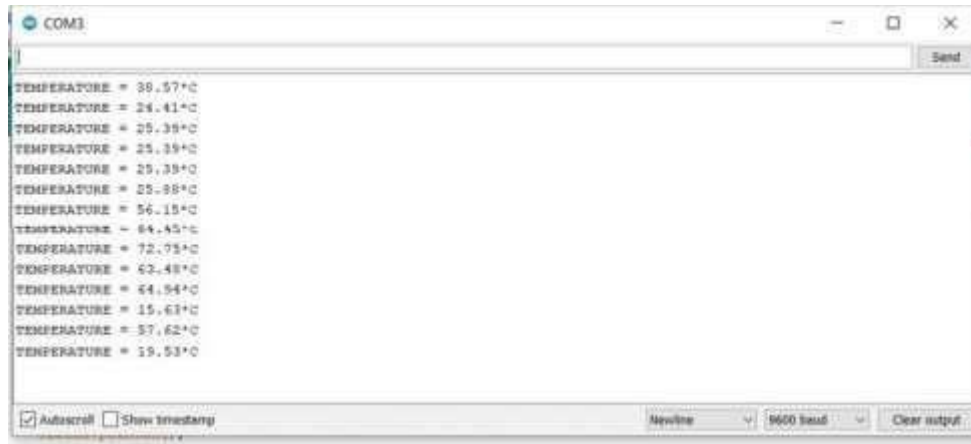
**Step 8:** Upload the code in Arduino board and output will be displayed in the serial monitor.

**Step 9:** Stop the process.

**Program Code:**

```
float temp;  
int tempPin = A0;  
void setup() {  
  Serial.begin(9600);  
  pinMode(tempPin,INPUT);  
}  
void loop() {  
  temp = analogRead(tempPin);  
  temp = temp * 0.48828125;  
  Serial.print("TEMPERATURE = ");  
  Serial.print(temp);  
  Serial.print("*C");  
  Serial.println();  
  delay(1000);  
}
```

**Output:**

**Observation:**

Learned about how to make temperature and humidity sensor that react to certain threshold using Arduino Uno and print the result in the Arduino IDE.

## **Practical 4**

**Aim:** To develop an application to send and receive data with NodeMCU using HTTP requests.

### **Components Required:**

- Arduino IDE
- Arduino\_JSON Library
- ESP8266
- NodeMCU
- MicroSD Card
- Jumper wires
- Breadboard

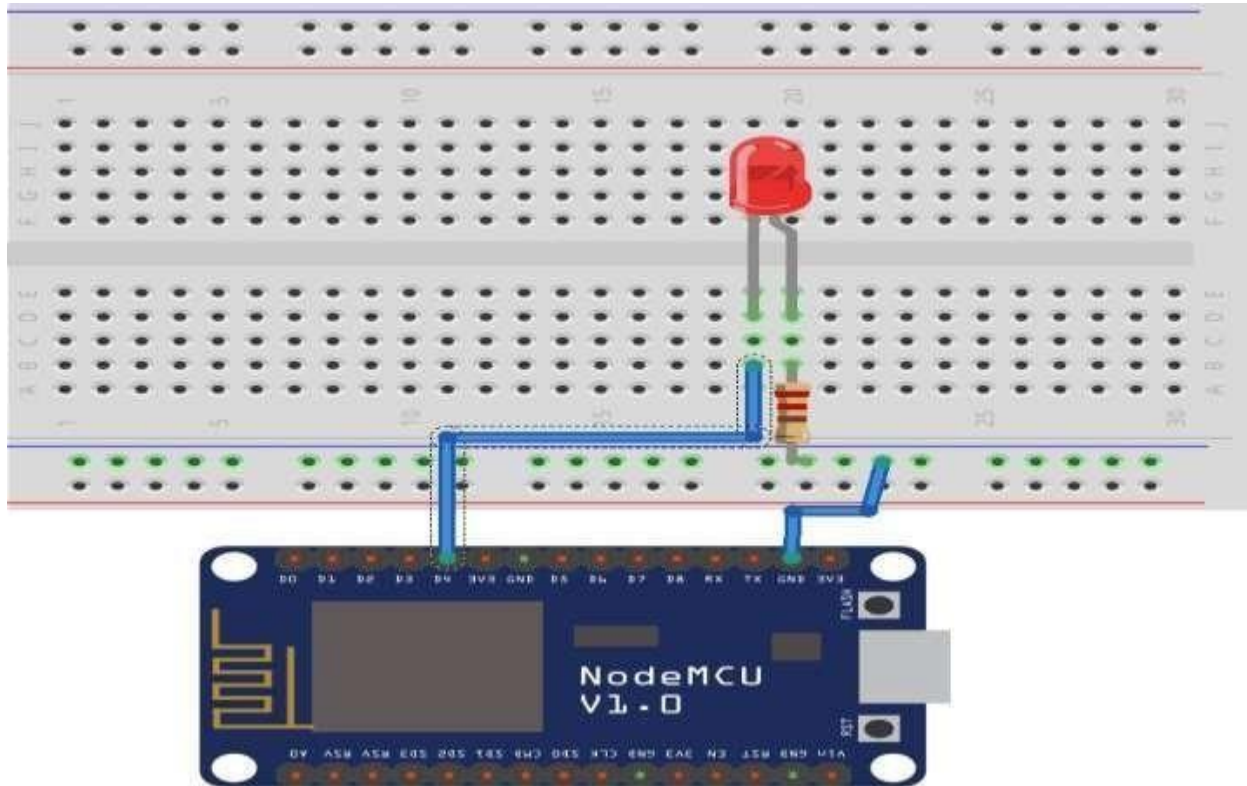
### **Theory Background:**

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added. The ESP8266 performs HTTP requests to Node-RED, but you can use these examples with other services like ThingSpeak, IFTTT.com (Web Hooks service), OpenWeatherMap.org, PHP server, etc. The ESP8266 will make an HTTP GET request to update a reading in a service. This type of request could also be used to filter a value, request a value or return a JSON object.

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). Strictly speaking, the term "NodeMCU" refers to the firmware rather than the associated development kits.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented. The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards.



**Circuit Diagram:****Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In NodeMCU, connect VCC to power supply 5V and other connections using jumper wires.

**Step 3:** Connect the NodeMCU with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->NodeMCU and the select port.

**Step 6:** Then enter the code in Arduino Software.

**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Arduino board.

**Step 9:** Stop the process.

---

**Program Code:**

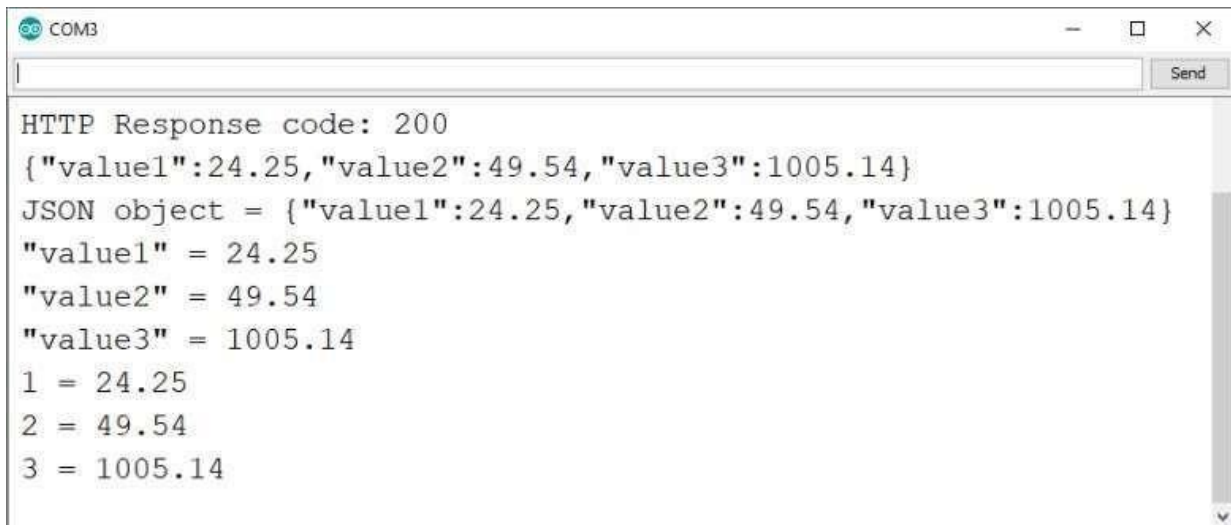
```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
const char* ssid = "clg";
const char* password = "123";

//Your Domain name with URL path or IP address with path
String serverName = "http://192.168.1.106:1880/get-sensor";

unsigned long lastTime = 0;
unsigned long timerDelay = 5000;
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());

  Serial.println("Timer set to 5 seconds (timerDelay variable), it will take 5 seconds before
publishing the first reading.");
}
void loop() {
  if ((millis() - lastTime) > timerDelay) {
    if(WiFi.status()== WL_CONNECTED){
      WiFiClient client;
      HTTPClient http;
      String serverPath = serverName + "?temperature=24.37";
      http.begin(client, serverPath.c_str());
      int httpResponseCode = http.GET();
      if (httpResponseCode>0) {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        String payload = http.getString();
        Serial.println(payload);
      }
    }
  }
}
```

```
    }  
    else {  
        Serial.print("Error code: ");  
        Serial.println(httpResponseCode);  
    }  
    http.end();  
}  
else {  
    Serial.println("WiFi Disconnected");  
}  
lastTime = millis();  
}  
}
```

**Output:**

```
HTTP Response code: 200  
{"value1":24.25,"value2":49.54,"value3":1005.14}  
JSON object = {"value1":24.25,"value2":49.54,"value3":1005.14}  
"value1" = 24.25  
"value2" = 49.54  
"value3" = 1005.14  
1 = 24.25  
2 = 49.54  
3 = 1005.14
```

**Observations:**

Learned about how to develop an application that send and receive data with NodeMCU using HTTP requests.

## **Practical 5**

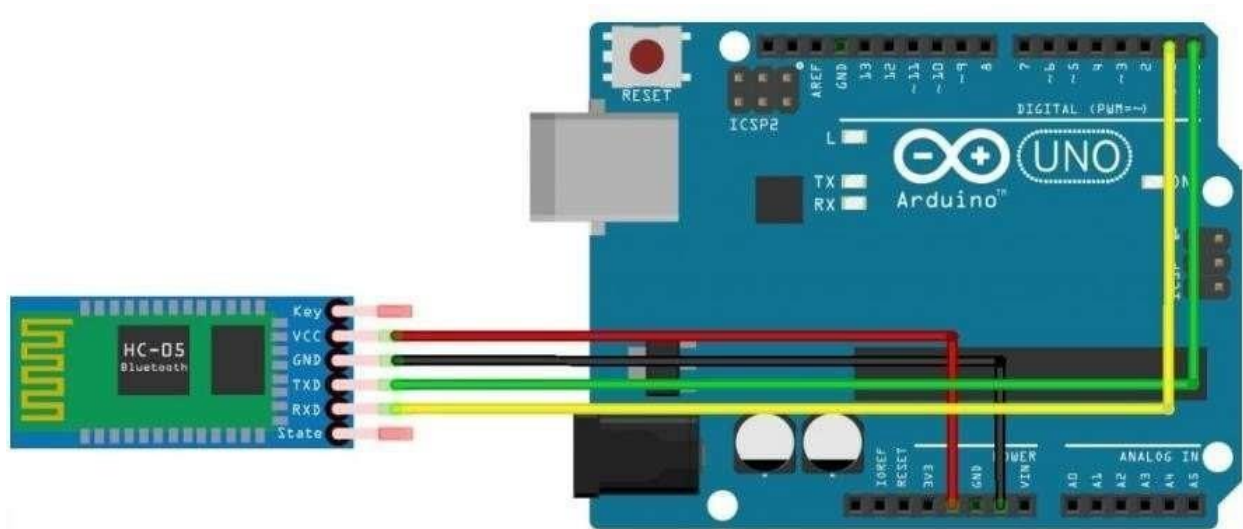
**Aim:** To interface Bluetooth with Arduino and write a program to send sensor data to smartphones using Bluetooth.

### **Components Required:**

- Arduino UNO
- Android Smartphone that has Bluetooth.
- HC-05 Bluetooth Module
- Android Studio
- USB cable
- DHT-11 temperature and humidity sensor

### **Theory Background:**

Bluetooth Low Energy (BLE) is a version of Bluetooth and it is present as a smaller, highly optimized version of the classic Bluetooth. It is also known as Smart Bluetooth. The BLE was designed keeping in mind the lowest possible power consumption specifically for low cost, low bandwidth, low power and low complexity. ESP32 has inbuilt BLE capabilities but for other microcontrollers like Arduino, nRF24L01 can be used. This RF module can be also used as BLE module to send the data to other Bluetooth device like smartphones, computer etc. BLE uses the same 2.4 GHz ISM band with baud rate from 250Kbps to 2Mbps which is allowed in many countries and can be applied to industrial and medical applications. The Band starts at 2400 MHz to 2483.5 MHz and it is divided into 40 channels. Three of these channels are known as 'Advertising' and are used by devices to send advertising packets with information about themselves so other BLE devices can connect. These channels were initially selected at the lower upper of the band and middle of the band to avoid interference which can possibly interfere with a number of channels.

**Circuit Diagram:****Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In Arduino UNO, connect VCC to power supply 5V and other connections using jumper wires.

**Step 3:** Connect the Arduino UNO with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->Arduino UNO and the select port.

**Step 6:** Then enter the code in Arduino Software.

**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Arduino board.

**Step 9:** Create sample app for temperature sensing using Android Studio

**Step 10:** Enable the Bluetooth in Android device and connect it with Arduino UNO

**Step 11:** Stop the process.

**Program Code:**

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();}

void loop()
{ char c;
if(Serial.available())
{
  c = Serial.read();
  if(c=='t')
  readSensor();
  }}
void readSensor() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.print(" *C ");
}
```

**Output:****Observations:**

Learned about how to develop an application that connect to smartphone using bluetooth and send data to the application using Arduino UNO.



## **Practical 6**

**Aim:** To develop an application that measures the room temperature and post the temperature value on Thingspeak cloud platform.

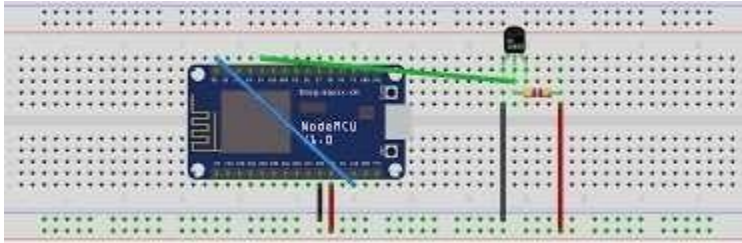
**Components Required:**

- NodeMCU
- Temperature Sensor
- Jumper wires
- Breadboard

**Theory Background:**

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added. The ESP8266 performs HTTP requests to Node-RED, but you can use these examples with other services like ThingSpeak, IFTTT.com (Web Hooks service), OpenWeatherMap.org, PHP server, etc. The ESP8266 will make an HTTP GET request to update a reading in a service. This type of request could also be used to filter a value, request a value or return a JSON object.

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). Strictly speaking, the term "NodeMCU" refers to the firmware rather than the associated development kits. The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented. The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards.

**Circuit Diagram:****Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In NodeMCU, connect VCC to power supply 5V and other connections using jumper wires.

**Step 3:** Connect the NodeMCU with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->NodeMCU and the select port.

**Step 6:** Then enter the code in Arduino Software.

**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Thingspeak.

**Step 9:** Stop the process.

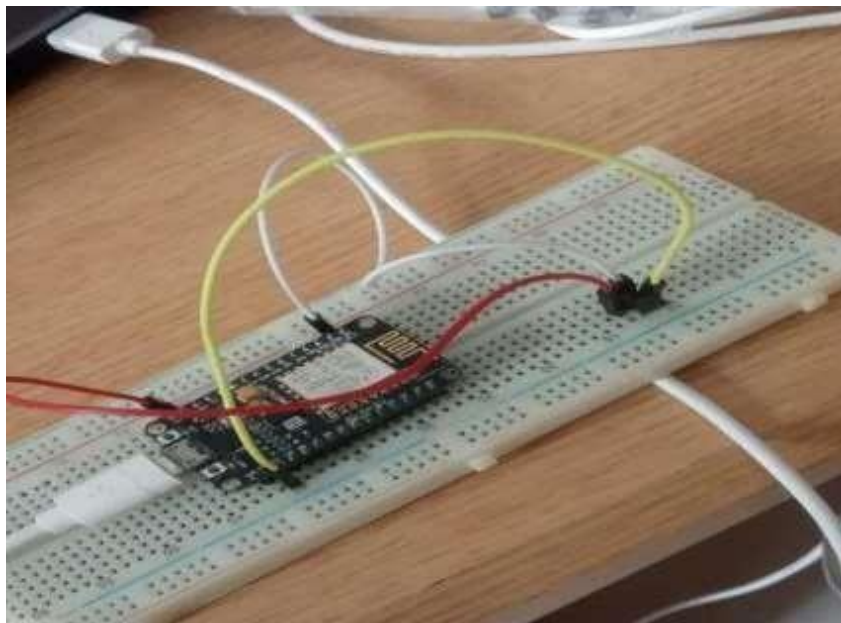
**Program Code:**

```
#include <ESP8266WiFi.h>
const char* ssid = "ram";
const char* password = "meet2602";
int LM35 = A0; //Analog channel A0 as used to measure temperature
WiFiServer server(80);
void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println();
```

---

```
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
server.begin();
Serial.println("Server started");
Serial.print("Use this URL to connect: ");
Serial.print("http://"); //URL IP to be typed in mobile/desktop browser
Serial.print(WiFi.localIP());
Serial.println("/");
}
void loop() {
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  Serial.println("new client");
  while (!client.available()) {
    delay(1);
  }
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();
  float temperatureC;
  float temperatureF;
  int value = LOW;
  if (request.indexOf("/Tem=ON") != -1) {
    float reading = analogRead(LM35); //Analog pin reading output voltage by Lm35
    float temperatureC = LM35 / 3.1; //Finding the true centigrade/celsius temperature
    Serial.println("CENTI TEMP= ");
    Serial.print(temperatureC); //Print centigrade temperature on Serial Monitor
    float temperatureF = ((temperatureC ) * 9.0 / 5.0) + 32.0;
    Serial.println("FARE TEMP= ");
    Serial.print(temperatureF); //Print farenheight temperature on Serial Monitor
```

```
value = HIGH;
}
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.print("Celcius temperature =");
client.print(temperatureC);
client.println("Farenheight temperature =");
client.print(temperatureF);
if (value == HIGH) {
  client.println("Updated");
} else {
  client.print("Not Updated");
}
client.println("<br><br>");
client.println("<a href=\"/Tem=ON\"><button>Update Temperature</button></a></br>");
client.println("</html>");
delay(1);
Serial.println("Client disonnected");
Serial.println("");
}
```

**Output:**



Learned about how to develop an application that measure room temperature and send data to Thingspeak.

## **Practical 7**

**Aim:** To develop an application for measuring the distance using ultrasonic sensors and post distance value on Thingspeak cloud platform.

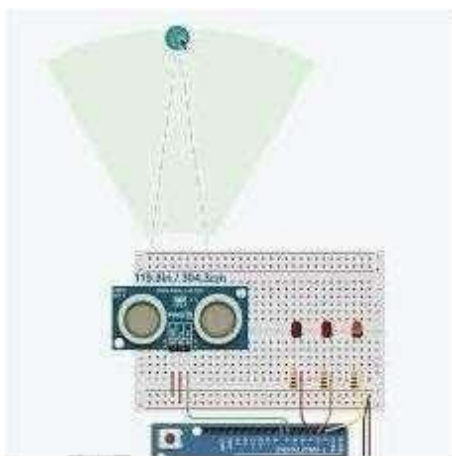
### **Components Required:**

- Arduino Uno
- Distance Sensor
- Jumper wires
- Breadboard

### **Theory Background:**

Ultrasonic sensors (sometimes called ultrasonic transducers), measure the distance to or the presence of a target object by sending a sound pulse, above the range of human hearing (ultrasonic), toward the target and then measuring the time it takes the sound echo to return. After inserting your network credentials, channel number and API key, upload the code to your board. Open the Serial Monitor at a baud rate of 115200, and press the on-board RST button. After 30 seconds, it should connect to Wi-Fi and start publishing the readings to ThingSpeak. Ultrasonic transducers operate at frequencies in the range of 30–500 kHz for air-coupled applications. As the ultrasonic frequency increases, the rate of attenuation increases. Thus, low- frequency sensors (30–80 kHz) are more effective for long range, while high-frequency sensors are more effective for short range.

### **Circuit Diagram:**



**Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In Arduino UNO, connect VCC to power supply 5V and other connections using jumper wires.

**Step 3:** Connect the Arduino UNO with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->Arduino UNO and the select port.

**Step 6:** Then enter the code in Arduino Software.

**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Arduino board.

**Step 9:** Create account on Thingspeak

**Step 10:** Connect it with Arduino UNO for data transfer

**Step 11:** Stop the process.

**Program Code:**

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
char ssid[] = "clgcgpit"; //SSID
char pass[] = "123456789"; //Password
WiFiClient client;
unsigned long myChannelField = 123456; // Channel ID
const int ChannelField = 1; // Which To Field Write
String value = "";
void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client); }
void loop(){
  if (Serial.available() > 0) {
```

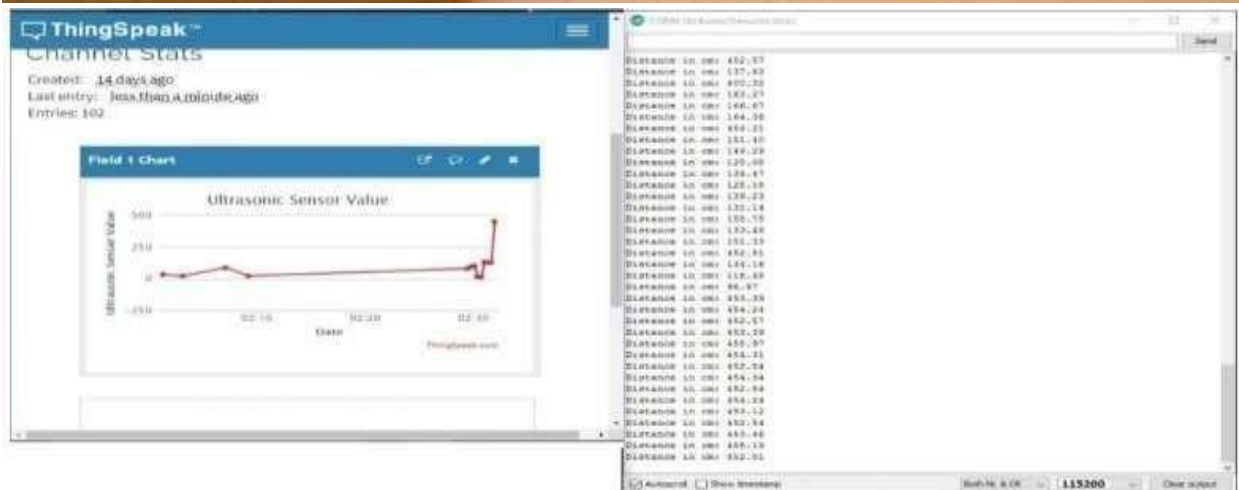
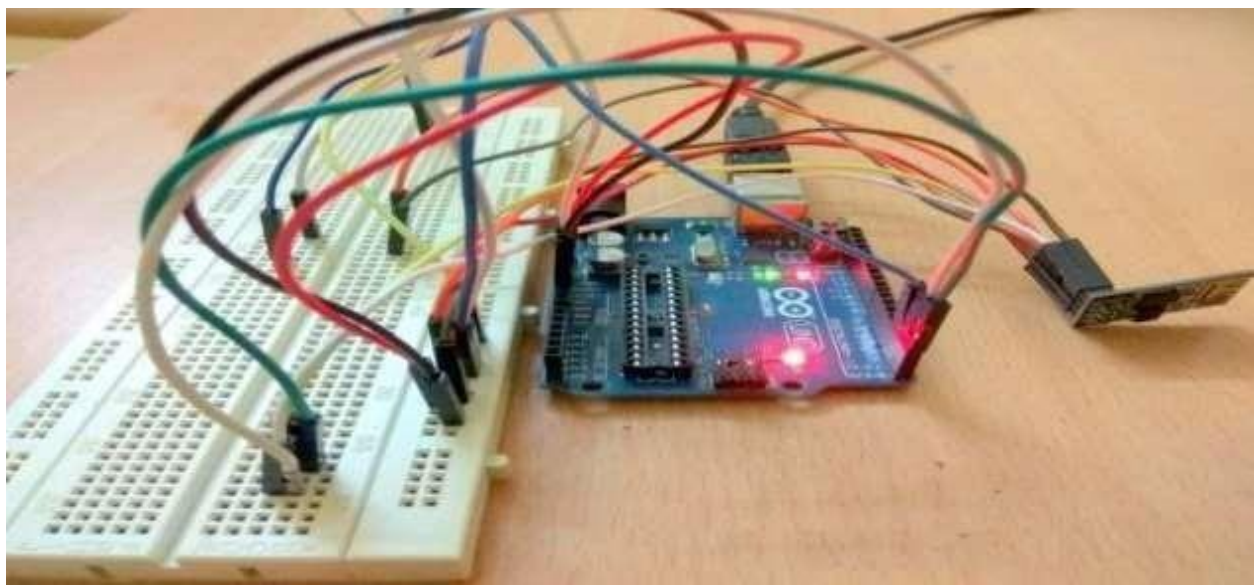


```

while (Serial.available() > 0) {
  int inChar = Serial.read();
  value += (char)inChar; } }
if (WiFi.status() != WL_CONNECTED) {
  while (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(ssid, pass);
    delay(5000); } }
value = "";

```

### Output:



### Observations:

Learned about how to develop a circuit that measure distance and send data to Thingspeak.

## **Practical 8**

**Aim:** To develop an application that measures the moisture of soil and post the sensed data over Google Firebase cloud platform.

### **Components Required:**

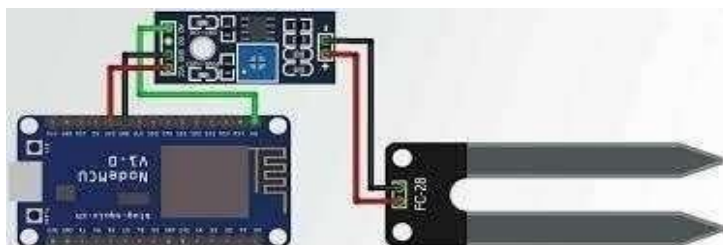
- NodeMCU
- DHT11 Temperature and Humidity Sensor
- Jumper wires
- Breadboard

### **Theory Background:**

Firebase is Google's mobile development platform that empowers you to quickly build and grow your app. It's built so that you're able to easily pull in Google Cloud products as your team or infrastructure needs grow. Here is a circuit diagram for connecting DHT11 Humidity Temperature Sensor with NodeMCU ESP8266 Board. Connect the positive terminal of DHT11 to 3.3V & negative terminal to GND. Connect the digital output pin to D2/GPIO4 of NodeMCU. Earlier when there was no development of IoT, the remote monitoring of sensor data was limited. The device used to be placed near the observer to check the data on display devices. Since the advancement of IoT, the limitations have been removed and data monitoring remotely has been possible. Not only the monitoring of data remotely but also monitoring the data on a real-time basis has become possible.

So we are basically focusing on IoT Based Remote Data Monitoring System. In some of our earlier projects, we used the IoT platforms like Thingspeak, Adafruit.io & Webpage to monitor data remotely using Nodemcu ESP8266. But in this project, we will Send the Real-Time Sensor Data to Google Firebase with ESP8266 & DHT11 Humidity Temperature Sensor. The data will be read using DHT11 Sensor and sent to Google Firebase Console Database.

### **Circuit Diagram:**



**Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram. In NodeMCU, connect VCC to power supply 5V and other connections using jumper wires.

**Step 3:** Connect the NodeMCU with USB cable to the system.

**Step 4:** Start ->Arduino IDE

**Step 5:** Select tools -> select board ->NodeMCU and the select port.

**Step 6:** Then enter the code in Arduino Software.

**Step 7:** Compile the code in Arduino Software.

**Step 8:** Upload the code in Arduino board.

**Step 9:** Create account on Firebase

**Step 10:** Connect it with NodeMCU for data transfer

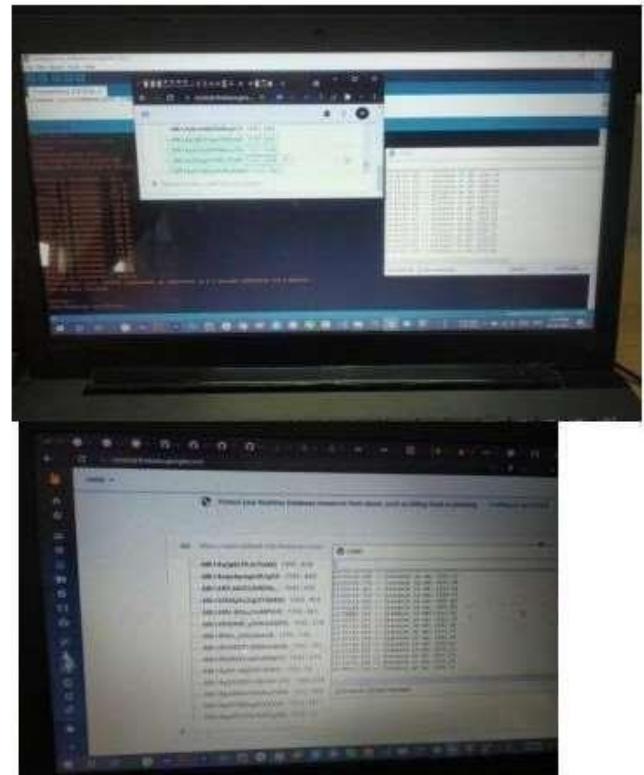
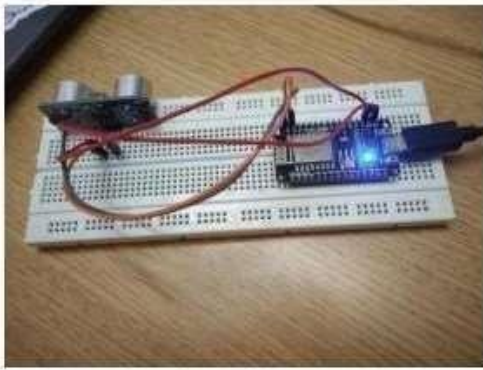
**Step 11:** Stop the process.

**Program code:**

```
#include
<ESP8266WiFi.h>
#include
<FirebaseArduino.h>
#define FIREBASE_HOST "test.firebaseio.com" //Replace with your firebase realtime dbhost
#define FIREBASE_AUTH "REALTIME DB SECRET" //Replace with your firebase realtime
dbsecret key
#define WIFI_SSID "SSID" //Replace with your wifi name
#define WIFI_PASSWORD "PASS" //Replace with your wifi passwordconst
int trigger = 16;
const int echo =5;long
T;float distanceCM;
void setup() {
  Serial.begin(115200);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
```

---

```
WiFi.mode(WIFI_STA);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to ");
Serial.print(WIFI_SSID);
while (WiFi.status() != WL_CONNECTED) { Serial.print(".");
delay(500);
}
Serial.println();
Serial.print("Connected");
Serial.print("IP Address: ");
Serial.println(WiFi.localIP()); //prints local IP address
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); // connect to the firebase
}
void loop(){
digitalWrite(trigger, LOW); delay(1);
digitalWrite(trigger, HIGH);
delayMicroseconds(10);
digitalWrite(trigger, LOW);
T = pulseIn(echo, HIGH);
distanceCM = T * 0.034;
distanceCM = distanceCM / 2;
Serial.print("Distance in cm: ");
Serial.println(distanceCM);
Firebase.pushFloat("/", distanceCM);
if (Firebase.failed()) {
Serial.print("pushing /logs failed:");
Serial.println(Firebase.error()); return;
}
}
```

**Output:****Observations:**

Learned about how to develop a circuit that measure distance and send data to Google Firebase.

## **Practical 9**

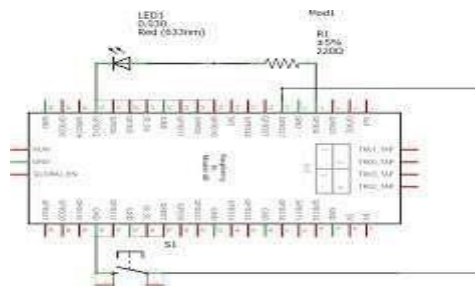
**Aim:** To develop an application for controlling LED with a switch using Raspberry Pi.

### **Components Required:**

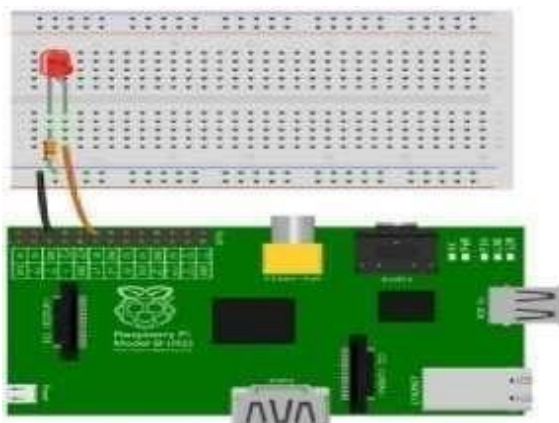
- LED
- Raspberry Pi
- Jumper wires
- Breadboard

### **Theory Background:**

Controlling an LED is the first step you can take to discover how GPIOs work and what to do with them. After this you'll be able to take other steps to build cool projects with your Pi. A 5mm LED is used as an output device. The anode of the LED (long lead) is connected to Physical Pin 18 (GPIO24) of Raspberry Pi. The cathode of the LED (short lead) is connected to one terminal of a 100Ω Resistor. The other terminal of the Resistor is connected to GND.



### **Circuit Diagram:**



**Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram.

**Step 3:** Connect the Raspberry Pi with USB cable to the system.

**Step 4:** Install Raspberry Pi image in your system

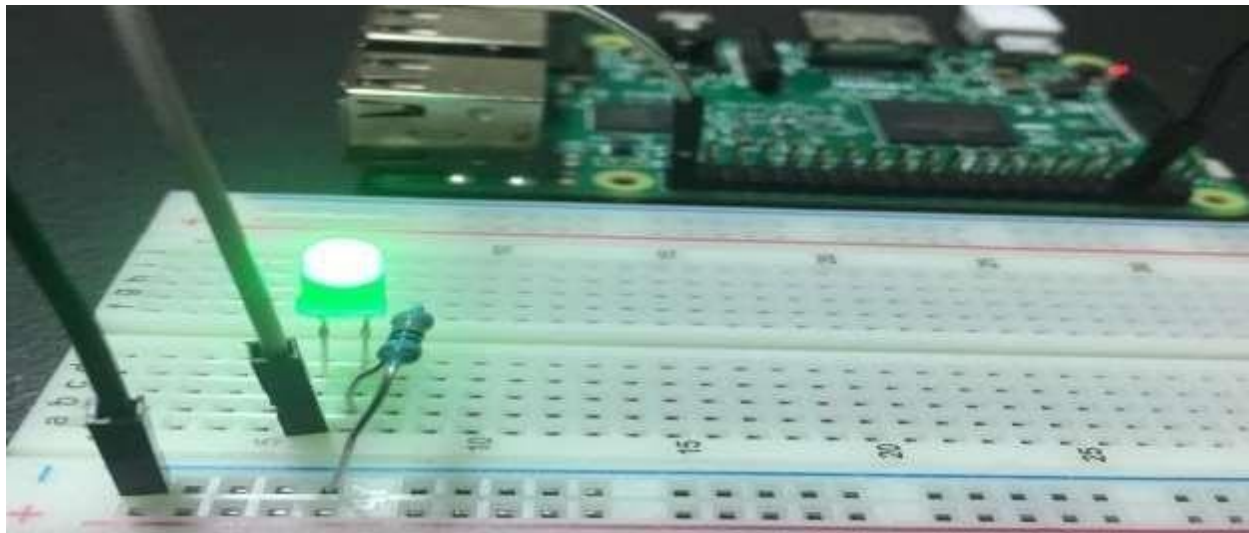
**Step 5:** Write the program in the notepad

**Step 6:** Save the file and go back to the console

**Step 7:** Stop the process.

**Program Code:**

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
print "LED ON"
GPIO.output(18,GPIO.HIGH)
time.sleep(1)
print "LED OFF"
GPIO.output(18,GPIO.LOW)
```

**Output:****Observations:**

Learned about how to develop a Raspberry Pi circuit to control a LED using a switch.



## **Practical 10**

**Aim:** To develop an application of motion detection using Raspberry Pi.

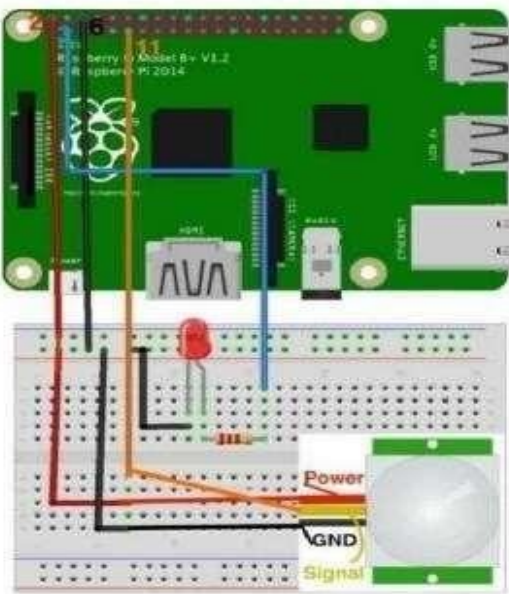
### **Components Required:**

- Raspberry Pi
- HC-SR501 PIR motion sensor
- LED
- Jumper wires
- Breadboard

### **Theory Background:**

A motion sensor, or motion detector, is an electronic device that uses a sensor to detect nearby people or objects. Motion sensors are an important component of any security system. When a sensor detects motion, it will send an alert to your security system, and with newer systems, right to your mobile phone. Connect the VCC and GND pins of the PIR Motion Sensor to +5V and GND pins of the Raspberry Pi. Connect the DATA Pin of the PIR Sensor to GPIO23 i.e. Physical Pin 16 of the Raspberry Pi. A 5V Buzzer is connected to GPIO24 i.e. Physical Pin 18 of the Raspberry Pi. The other pin of the buzzer is connected to GND. The most common type of active motion detector uses ultrasonic sensor technology; these motion sensors emit sound waves to detect the presence of objects. There are also microwave sensors (which emit microwave radiation), and tomographic sensors (which transmit and receive radio waves).

### **Circuit Diagram:**



**Process steps:**

**Step 1:** Start the process.

**Step 2:** Connect the components as shown in circuit diagram.

**Step 3:** Connect the Raspberry Pi with USB cable to the system.

**Step 4:** Install Raspberry Pi image in your system

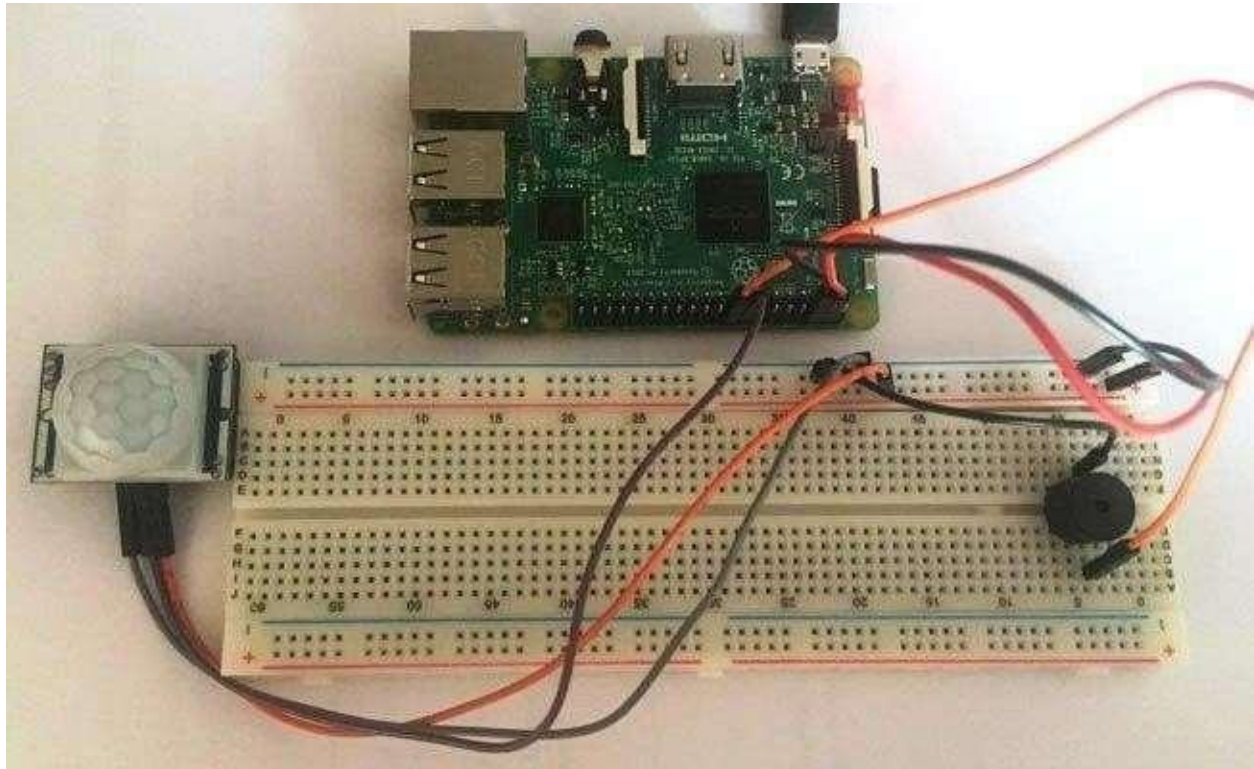
**Step 5:** Write the program in the notepad

**Step 6:** Save the file and go back to the console

**Step 7:** Stop the process.

**Program Code:**

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.IN)
GPIO.setup(3,GPIO.IN)
while True:
    i=GPIO.input(11)
    if i==0:
        print "No intrudes",i
        GPIO.output(3,0)
        time.sleep(0.1)
    elif i==1:
        print "intrudes detected",i
        GPIO.output(3,1)
        time.sleep(0.1)
```

**Output:****Observations:**

Learned about how to develop an application for motion sensing using Raspberry Pi.