Internet of Things

Practical No.: 1

Aim:- Interfacing LED, BUZZER and RELAY with Arduino to turn it on/off.

Hardware Requirements:

- **ARDUINO UNO-** The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. ... The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.
- **LED-** A light-emitting diode (LED) is a semiconductor device that emits light when an electric current flows through it. When current passes through an LED, the electrons recombine with holes emitting light in the process. LED's allow the current to flow in the forward direction and blocks the current in the reverse direction. Light-emitting diodes are heavily doped p-n junctions. Based on the semiconductor material used and the amount of doping, an LED will emit a coloured light at a particular spectral wavelength when forward biased. As shown in the figure, an LED is encapsulated with a transparent cover so that emitted light can come out.
- **BUZZER/SOUNDER-** Piezo buzzers are simple devices that can generate basic beeps and tones. They work by using a piezo crystal, a special material that changes shape when voltage is applied to it. If the crystal pushes against a diaphragm, like a tiny speaker cone, it can generate a pressure wave which the human ear picks up as sound
- **RELAY-** A Relay is an electromechanical device that can be used to make or break an electrical connection. It consists of a flexible moving mechanical part which can be controlled electronically through an electromagnet, basically, a relay is just like a mechanical switch but you can control it with an electronic signal instead of manually turning it on or off. So relay is a switch which controls (open and close) circuits electromechanically. The main operation of this device is to make or break contact with the help of a signal without any human involvement in order to switch it ON or OFF. It is mainly used to control a high powered circuit using a low power signal. Generally, a DC signal is used to control the circuit which is driven by high voltage like **controlling AC home appliances with DC signals from microcontrollers.**
- **RESISTOR-** Resistors are the most commonly used components in electronic circuits and devices. The main purpose of a resistor is to maintain specified values of voltage and current in an electronic circuit. A Resistor works on the principle of Ohm's law and the law states that the voltage across the terminals of a resistor is directly proportional to the current flowing through it. The unit of resistance is Ohm. The Ohm symbol shows resistance in a circuit from the name Geog Ohm – a German physicist who invented it.
- **BREADBOARD-** A breadboard, or protoboard, is a construction base for prototyping of electronics. Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason, solderless breadboards are also popular with students and in technological education. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

- **Software used:** Tinker-cad, Arduino IDE

- **Applications and Uses of LEDs can be seen in:**

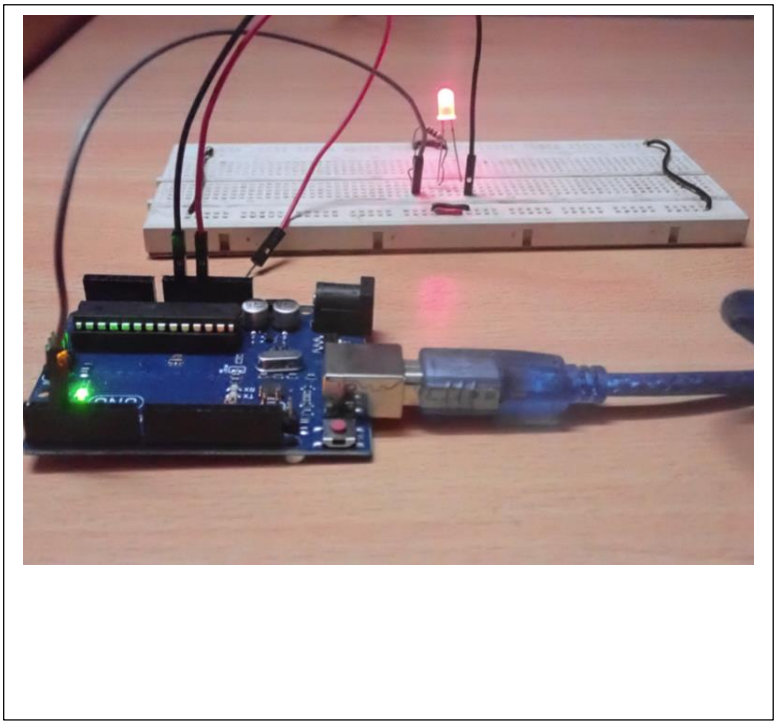
1. TV Backlighting
2. Smartphone Backlighting
3. LED displays
4. Automotive Lighting
5. Dimming of lights

The uses and functions of LEDs depend upon the place where it is used.

❖ **Interfacing LED with Arduino to turn it on/off:**

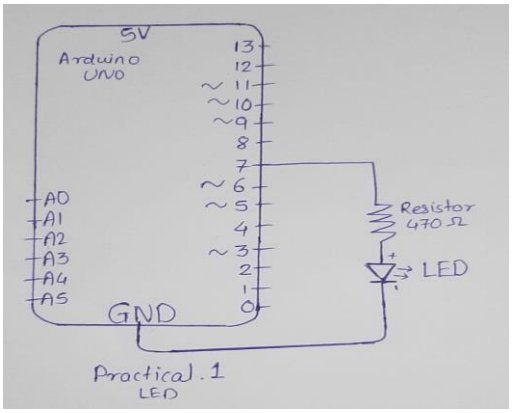
- **For this Practical we will only need:**

a) Arduino uno
b) LED
c) 470 Ohm resistor



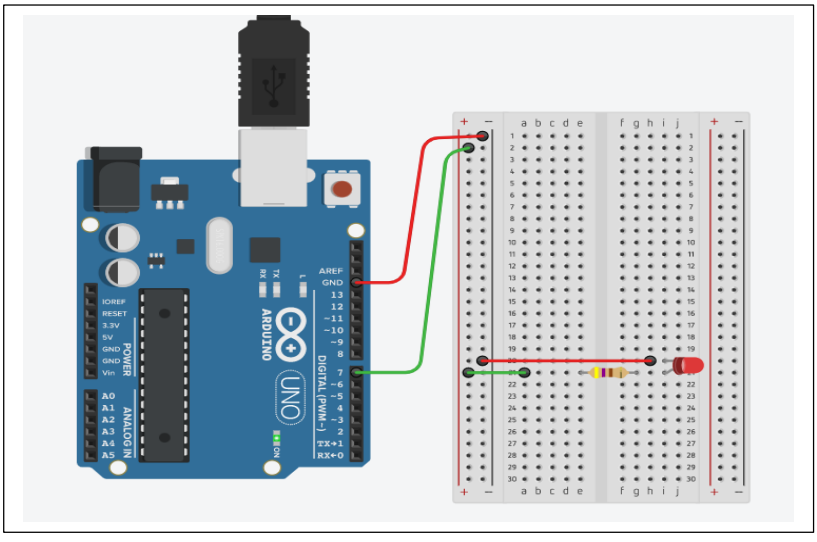
- **Circuit Schematic and Circuit:**

  - **Schematic**



- **Program:**

```
void setup(){
  pinMode(7, OUTPUT);
}
void loop(){
  digitalWrite(7, HIGH);
  delay(1000);
  digitalWrite(7, LOW);
  delay(1000);
}
```



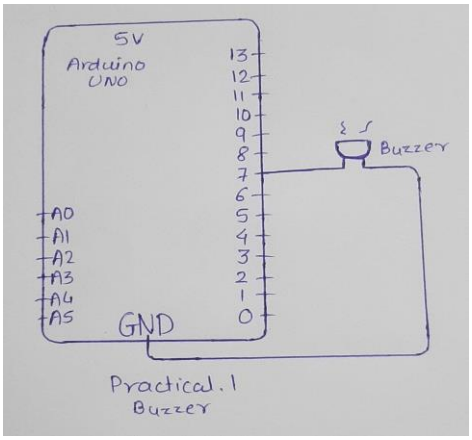
- Once we start Simulation the LED will start blinking at the interval of 1 second

❖ **Interfacing Buzzer with Arduino to turn it on/off:**

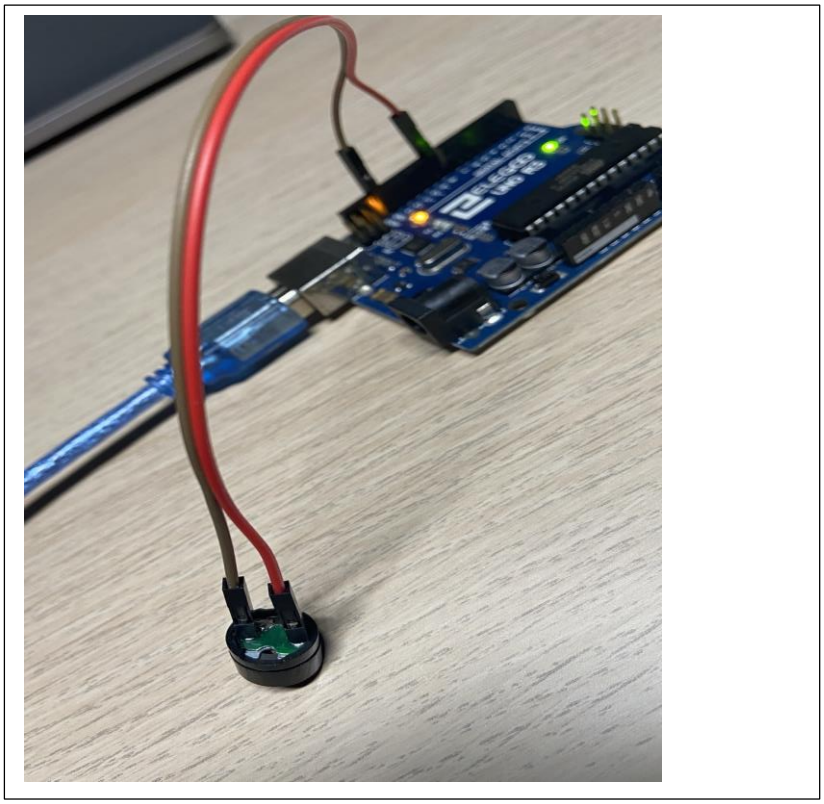
- **For this Practical we will only need:**

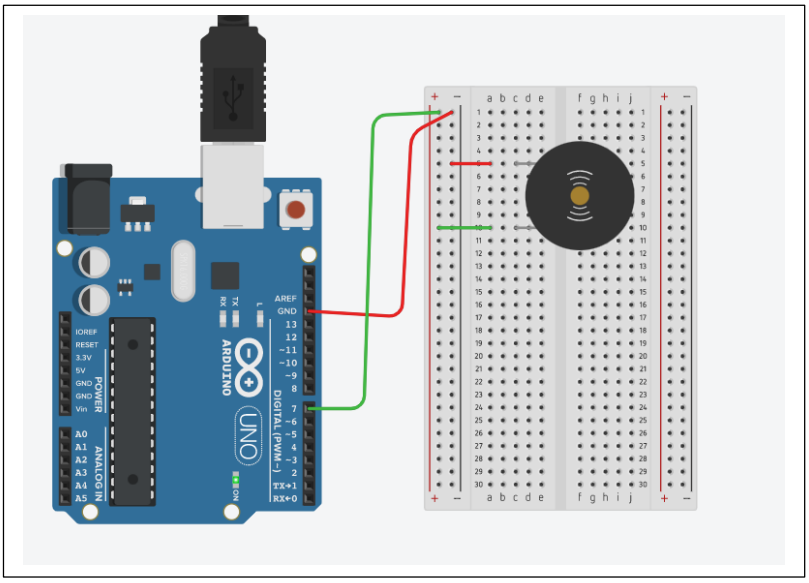
a) Arduino UNO
b) Buzzer

- **Circuit Schematic and Circuit:**
  - **Schematic**



  - **Circuit:**





- **Program:**

```
void setup()
{
  pinMode(7, OUTPUT);
}
void loop()
{
  digitalWrite(7, HIGH);
  delay(1000);
  digitalWrite(7, LOW);
  delay(1000);
}
```

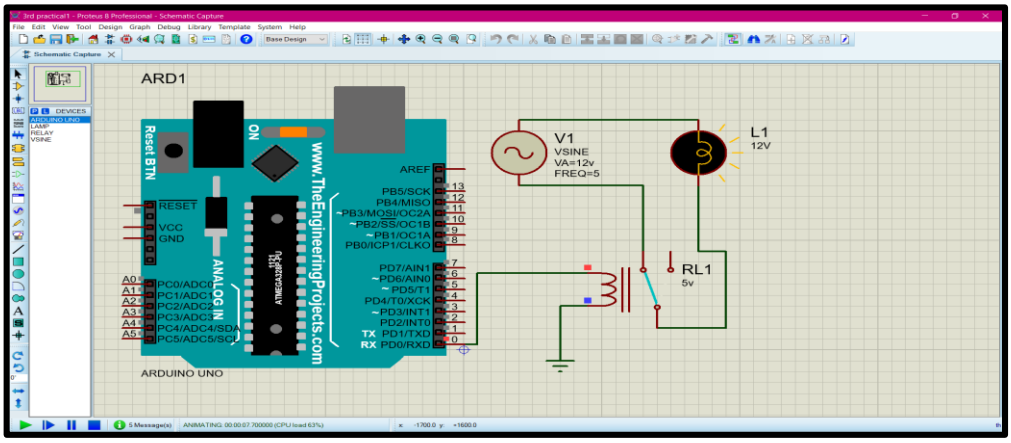
- **Output:**

❖ **Interfacing Buzzer with Arduino to turn it on/off:**

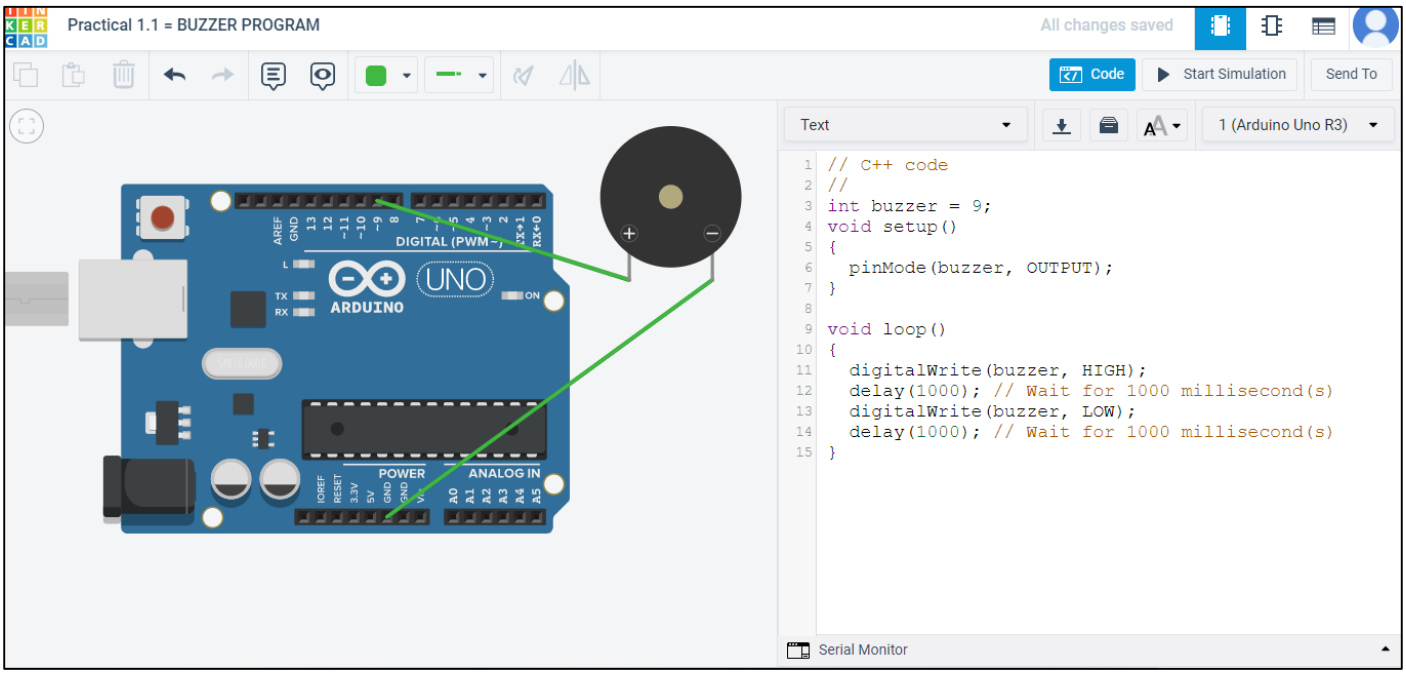
- **For this Practical we will only need:**

c) **Arduino UNO**
d) **Buzzer**

- **Circuit Schematic and Circuit:**
  - **Schematic:**



Arduino Code For RELAY



Conclusion: **Thus we successfully Interfaced LED, BUZZER and RELAY with Arduino to turn it on/off**

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 2

Aim :- Interfacing Switch with Arduino/NodeMCU/Raspberry Pi

Hardware Requirements: Arduino UNO, Pushbutton, Red LED, Resistor, breadboard, connecting wires.

Software used: Tinkercad, Arduino IDE

- **Switch or Pushbutton:**

  Push buttons or switches connect two points in a circuit when you press them.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.

A Pushbutton or Switch

The push button has 4 metal legs on its base (two legs on one side, and two legs on the opposite side). Unlike most other parts that connect to only one side of a breadboard, the push button has to connect to **both** sides of a breadboard. The two legs of the push button are connected to each other as shown in the picture.

- **Interfacing a switch with Arduino UNO:**

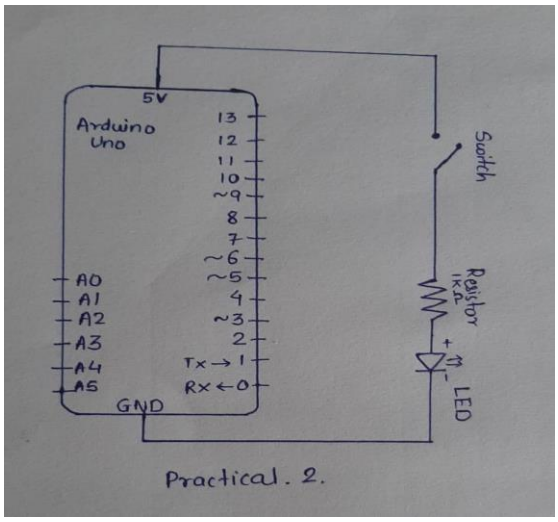
  In this example, we have interfaced a switch or pushbutton with Arduino UNO to turn on a LED when it is pressed.

  - **Steps:**
    1. Connect the LED on breadboard and then connect it's anode of the LED to pin 13 of Arduino board and cathode to ground.
    2. Connect the pushbutton's one leg to 5V supply and another leg to 10K ohm resistor.
    3. Connect one terminal of the resistor to Pin 7 and another to ground.
    4. Write the code and start simulation.

We can notice that when we press the pushbutton, the LED starts glowing and when we press the button again it stops glowing.

- **Circuit and Circuit Diagram:**

  - **Schematic Circuit Diagram**

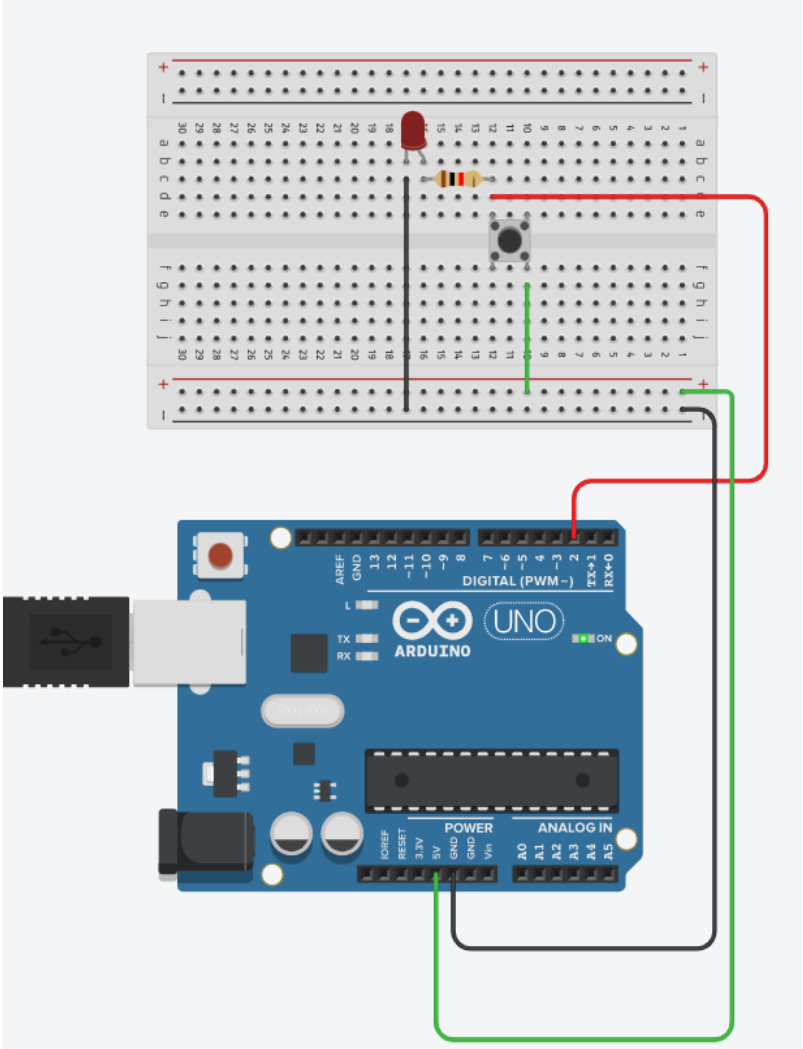


Program:

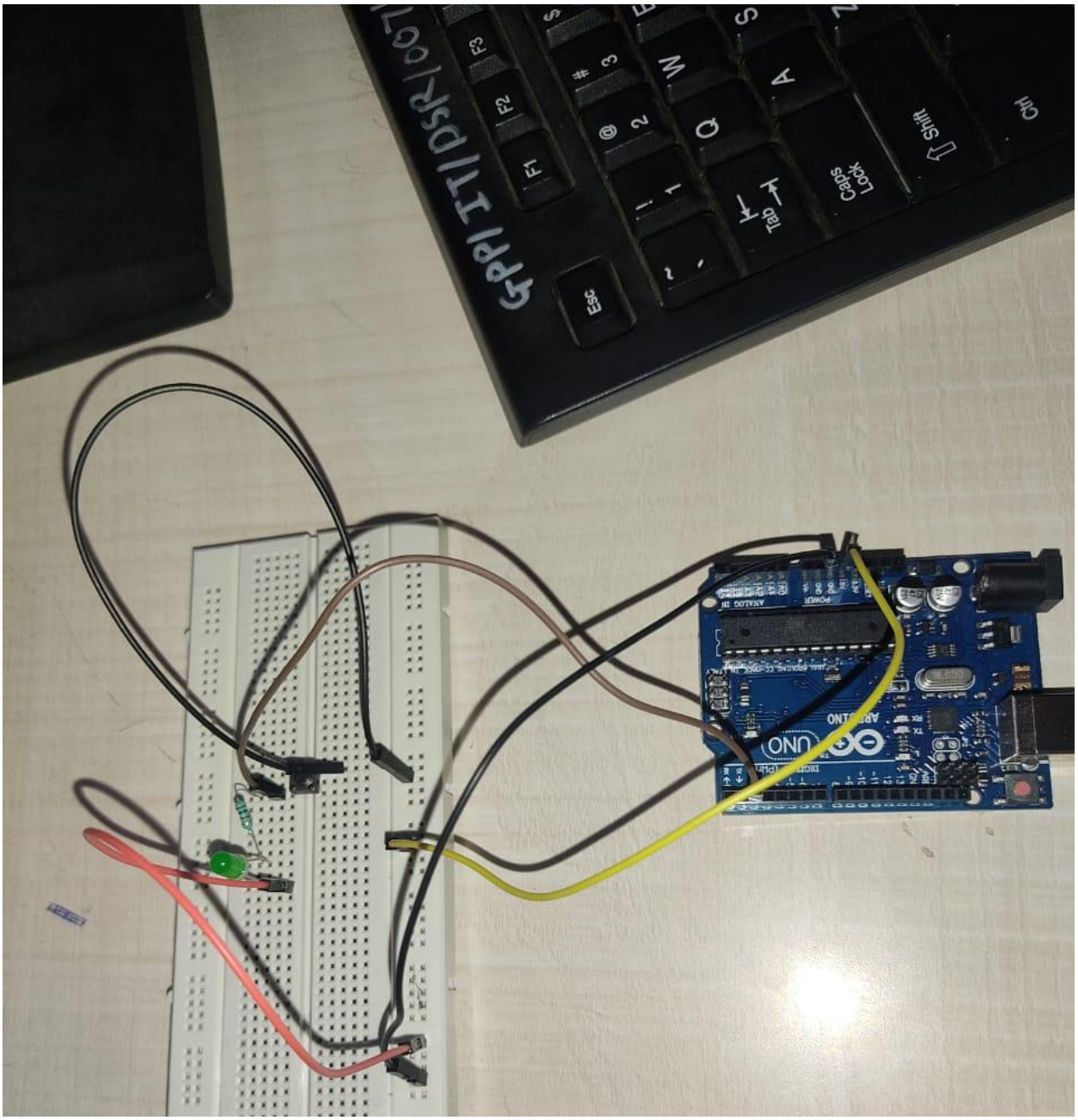
```
int buttonState = 0;
void setup()
{
  pinMode(7, OUTPUT);
  pinMode(2, INPUT);
}

void loop()
{
  buttonState = digitalRead(2);
  if(buttonState == HIGH)
  {
  digitalWrite(7, HIGH);
  }
  else
  {
  digitalWrite(7, LOW);
  }
}
```

- **Output:**

1. When Pushbutton is not pressed



➔ When pushbutton is pressed



CONSLUSION: We have successfully interfaced a switch

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 3

Aim :- Interfacing LDR with Arduino/NodeMCU/ Raspberrry pi to Sense Light Presence.

Hardware Requirements: Arduino UNO, Photoresistor, 1kΩ Resistor, Bulb, Breadboard, Connecting wires.

Software used: Tinkercad.

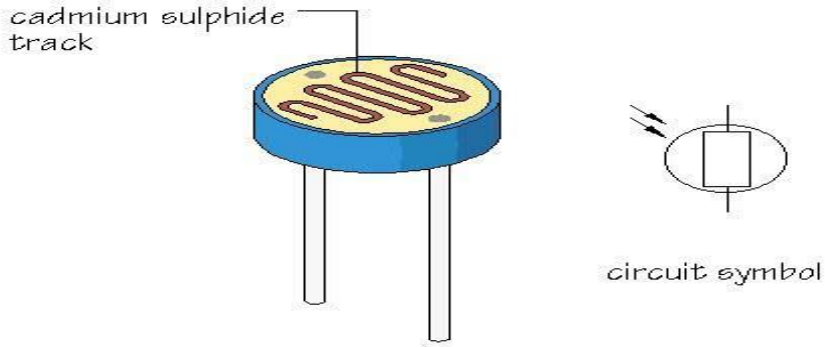
- **LDR (Light Dependent Resistor):**

An LDR or light dependent resistor is also known as photo resistor, photocell, photoconductor. It is a one type of resistor whose resistance varies depending on the amount of light falling on its surface. When the light falls on the resistor, then the resistance changes. These resistors are often used in many circuits where it is required to sense the presence of light. These resistors have a variety of functions and resistance. For instance, when the LDR is in darkness, then it can be used to turn ON a light or to turn OFF a light when it is in the light. A typical light dependent resistor has a resistance in the darkness of 1MOhm, and in the brightness a resistance of a couple of KOhm.

- **Working Principle of LDR:**

This resistor works on the principle of photo conductivity. It is nothing but, when the light falls on its surface, then the material conductivity reduces and also the electrons in the valence band of the device are excited to the conduction band. These photons in the incident light must have energy greater than the band gap of the semiconductor material. This makes the electrons to jump from the valence band to conduction.

These devices depend on the light, when light falls on the LDR then the resistance decreases, and increases in the dark.When a LDR is kept in the dark place, its resistance is high and, when the LDR is kept in the light its resistance will decrease.



- **Light Dependent Resistor Applications:**

Light dependent resistors have a low cost and simple structure. These resistors are frequently used as light sensors. These resistors are mainly used when there is a need to sense the absence and presence of the light such as burglar alarm circuits, alarm clock, light intensity meters, etc. LDR resistors mainly involves in various electrical and electronic projects.

- **Here are some of the applications of the LDR:**
  1. Security System Controlled by An Electronic Eye
  2. LDR Based light Intensity Control for Street Lights
  3. Lighting Switch from Sunset to Sunrise

- **Advantages and Disadvantages of LDR:**

  - **Advantages:**

1. The linearity of photoelectric conversion under strong illumination is poor.
2. Photoelectric relaxation process is long, and the relaxation phenomenon of photoconductivity will happen: that is, after illumination, the photoconductivity of semiconductor increases gradually with illumination time, and reaches the steady state value after a period of time. The photoconductivity decreases gradually after the illumination stops.
3. The frequency response (the ability of the device to detect the fast changing optical signals) is very low.

The internal photoelectric effect has nothing to do with the electrodes (only photodiodes do), that is, DC power supply can be used.

  - **Disadvantages:**

1. The linearity of photoelectric conversion under strong illumination is poor.
2. Photoelectric relaxation process is long, and the relaxation phenomenon of photoconductivity will happen: that is, after illumination, the photoconductivity of semiconductor increases gradually with illumination time, and reaches the steady state value after a period of time. The photoconductivity decreases gradually after the illumination stops.
3. The frequency response (the ability of the ldr light dependent resistor to detect the fast changing optical signals) is very low.

- **Circuit Schematic and Circuit:**

  - **Schematic:**

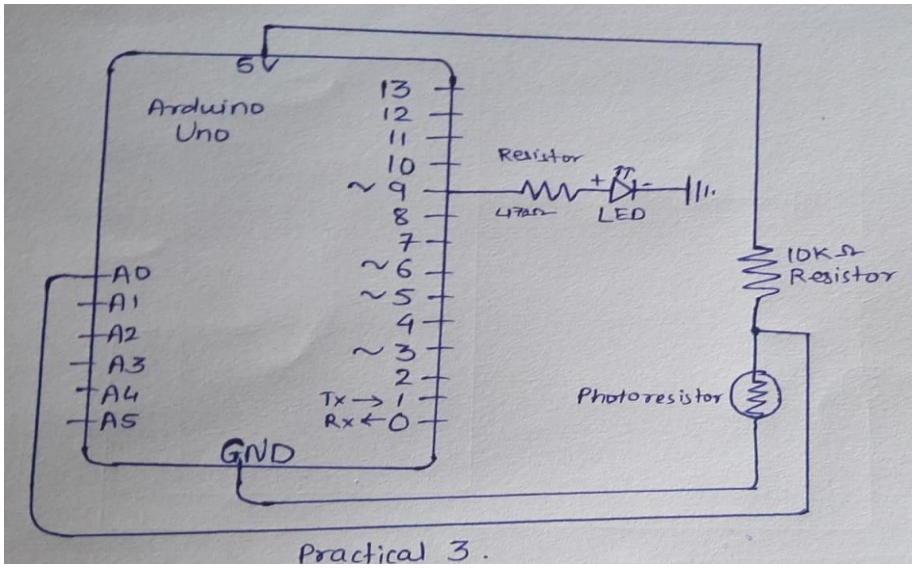


Fig.3.3. Circuit Schematic

- **Circuit:**

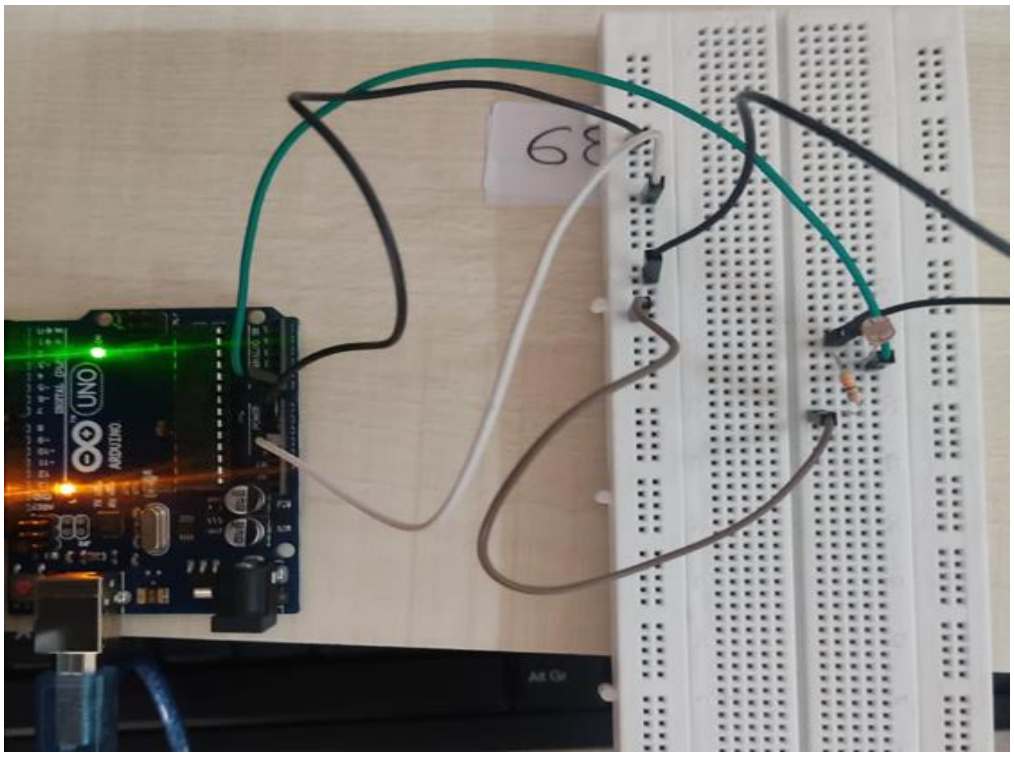
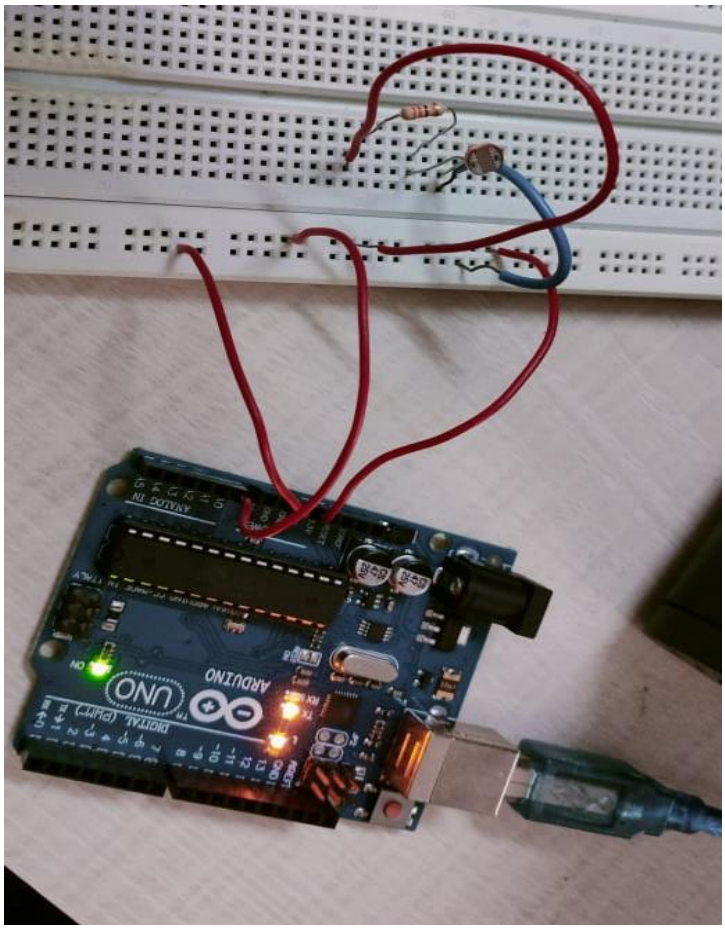


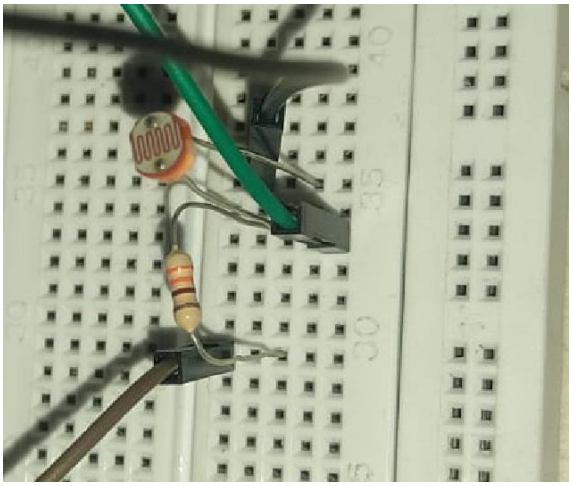
Fig.3.4. Circuit on Tinkercad

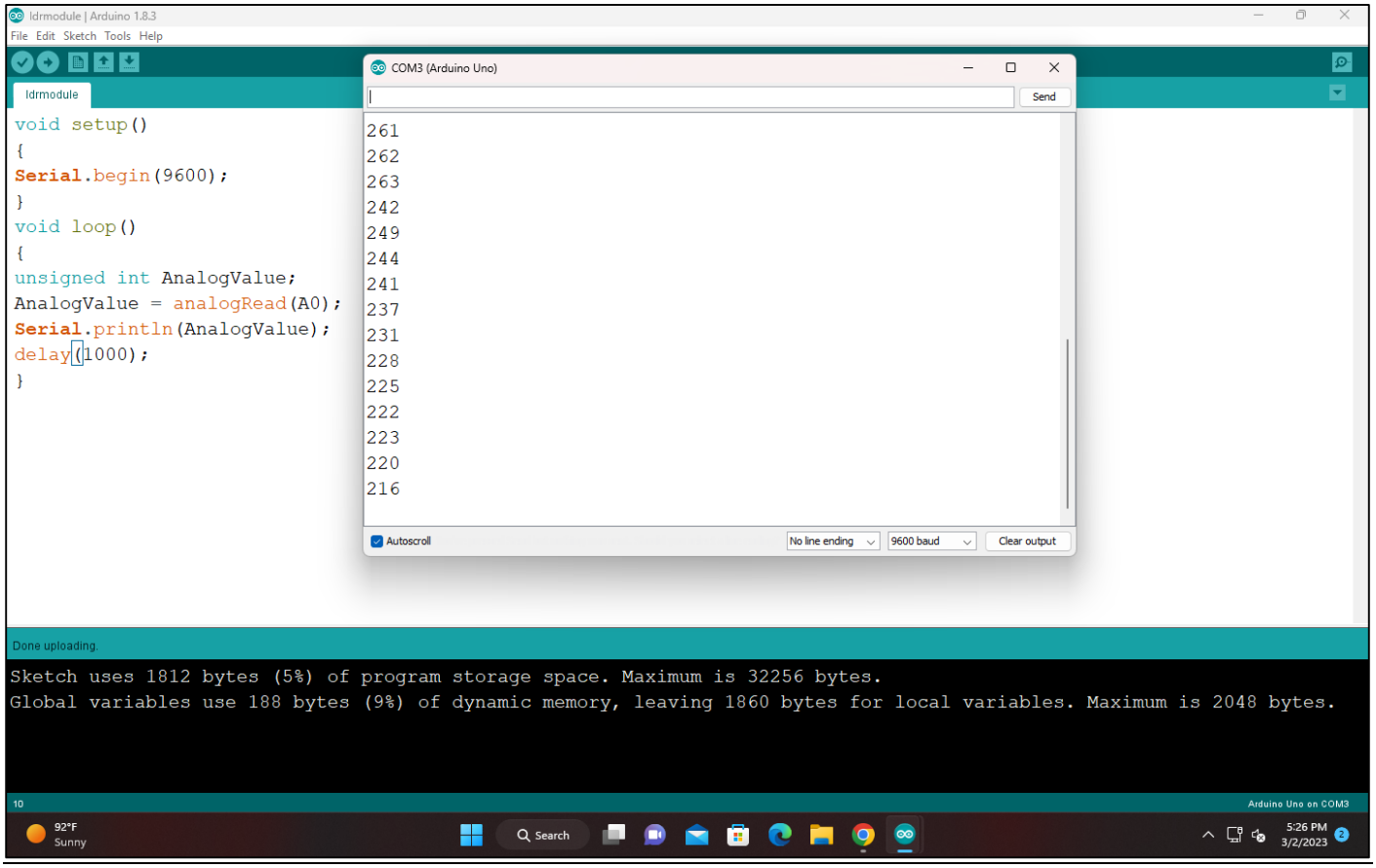
- **Program:**

```
int ldr = A0;
int value = 0;
void setup() {
Serial.begin(9600);
pinMode(9,OUTPUT);
}
void loop() {
value=analogRead(ldr);
Serial.println(value);
if(value<90){
   digitalWrite(9,LOW);
  }
  else{
   digitalWrite(9,HIGH);
  }
}
```



Output: Here the serial monitor shows the number 58. But as I increase the light on the photoresistor the number increases. The light here is under 90 hence the bulb isn't glowing.





- **Conclusion:** We successfully completed practical on Interfacing LDR with Arduino UNO to sense light.

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 4

Aim:- Interfacing Analog Temperature Sensor i.e., LM35 with Arduino to Sense Temperature

Hardware Requirements: Arduino UNO, Breadboard, Temperature Sensor [TMP36]/LM35.

Software used: Tinkercad.

- **LM35 Temperature Sensor: -**

LM35 is a temperature measuring device having an analog output voltage proportional to the temperature. It provides output voltage in centigrade (Celsius). It does not require any external calibration circuitry. The sensitivity of LM35 is 10 mV/degree Celsius. As temperature increases, output voltage also increases.

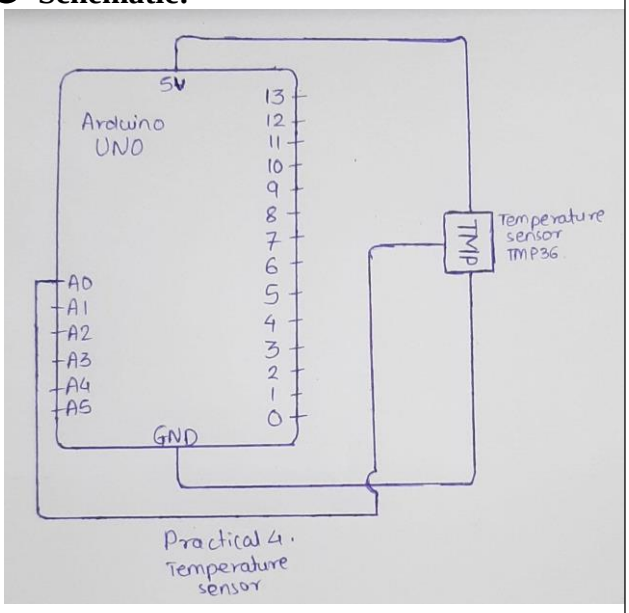
E.g., 250 mV means 25°C

- **Connecting LM35with Arduino Uno: -**

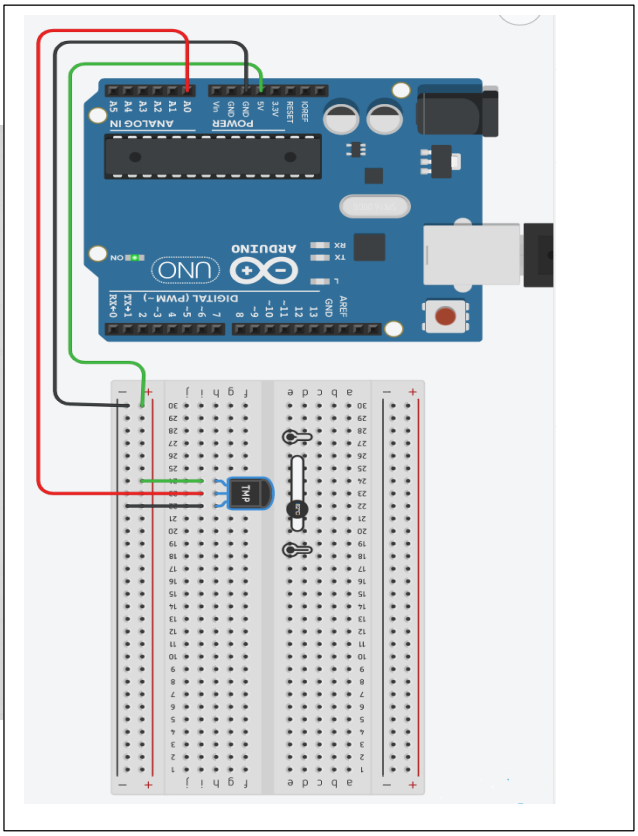
Connecting an LM35 to the Arduino is very easy as you only need to connect 3 pins. Start by connecting positive pin to 5V output of the Arduino and the Ground (GND) pin to the ground.

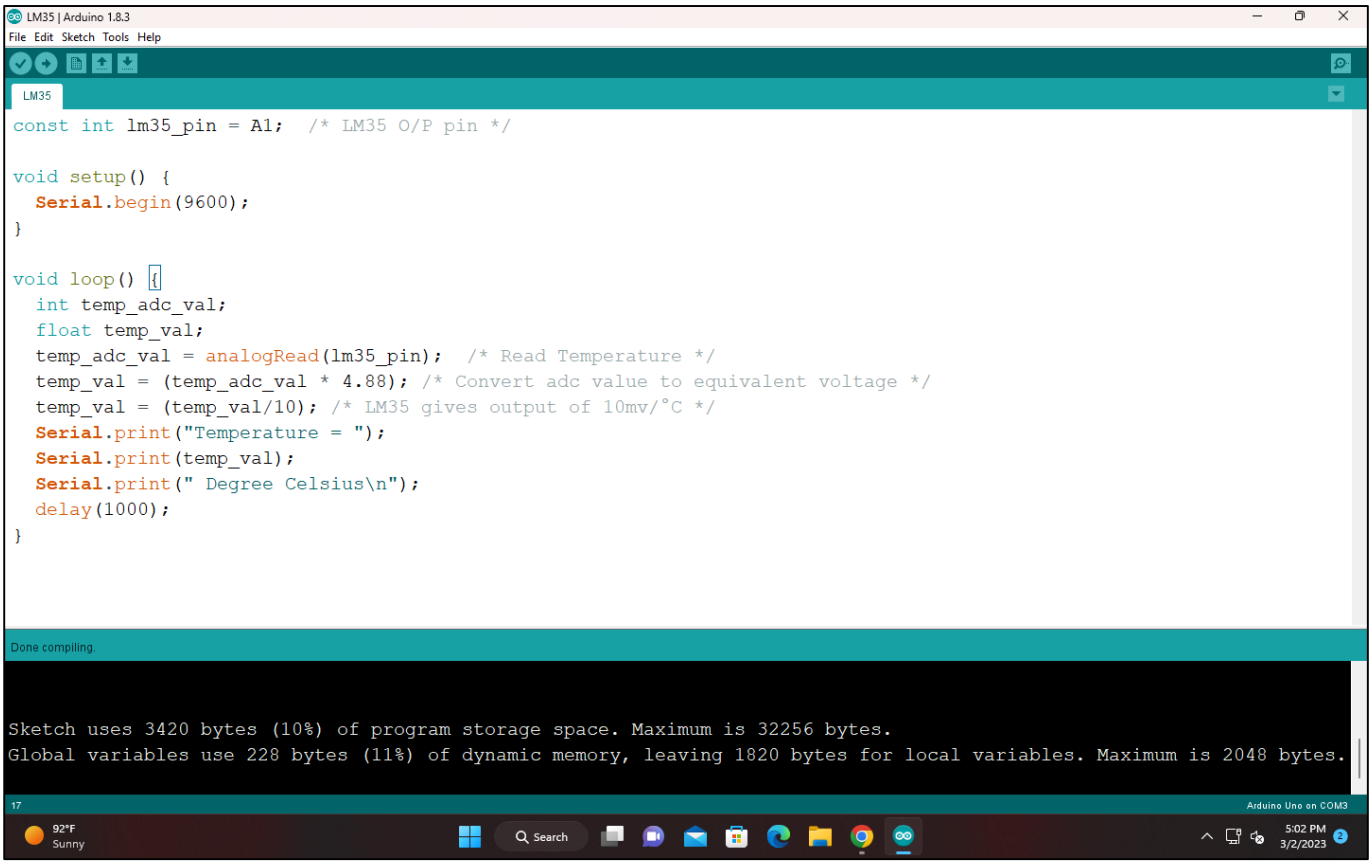
Next, connect the middle pin (V_{OUT}) to any of the analog inputs of the Arduino. In this case, I used the analog input pin A0.

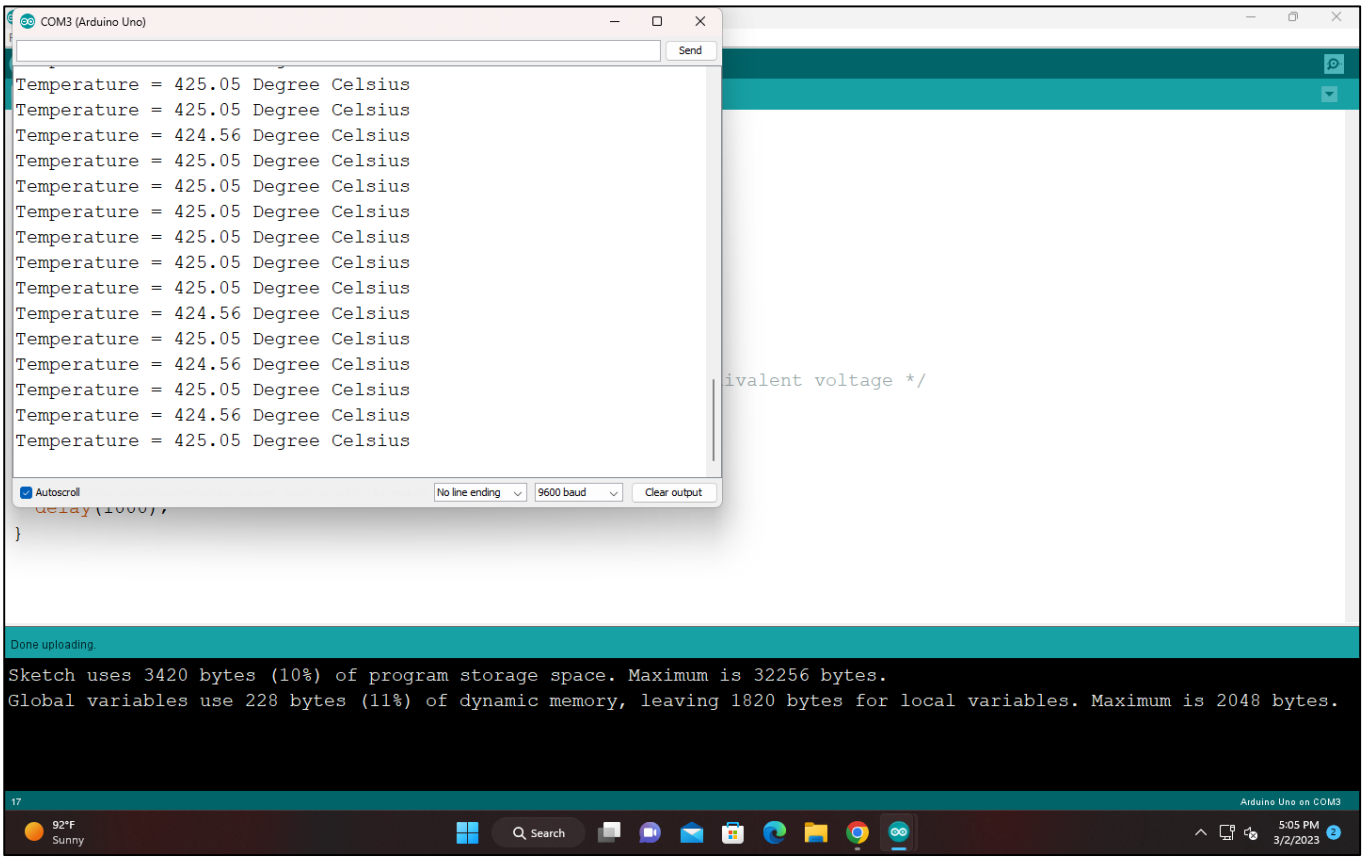
- **Circuit Schematic and Circuit:**
  - **Schematic:**



  - **Circuit:**





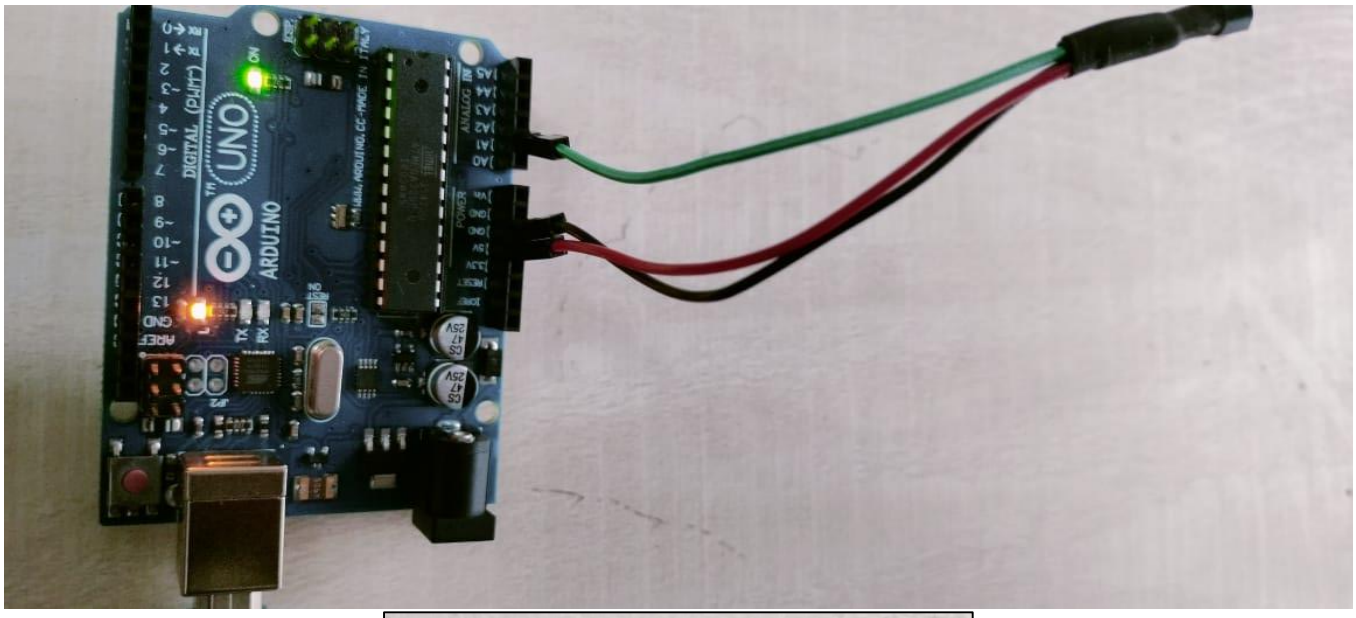


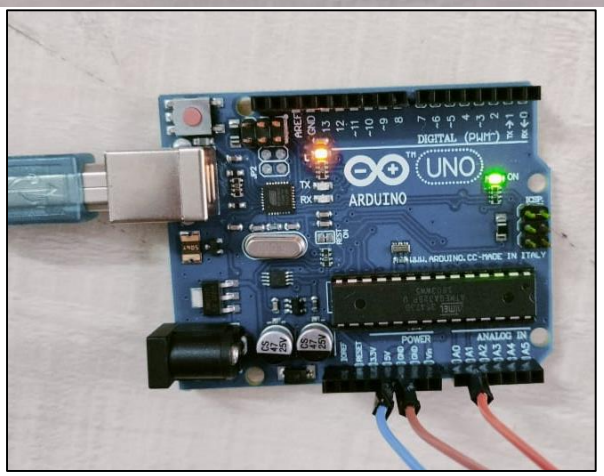
- **Explanation: -**

First, I defined to which pin of the Arduino the V_{OUT} pin of the sensor is connected. In this case, we used the analog pin A0. The statement #define can be used to give a name to a constant value. The compiler will

replace all references to this constant with the defined value when the program is compiled. So everywhere you mention sensorPin, the compiler will replace it with A0 when the program is compiled. In the setup section of the code, we begin serial communication at a baud rate of 9600. In the loop section of the code, we start by taking a reading from the sensor with the function analogRead(pin). Next, we use the formulas that I mentioned earlier in the article to convert the readding into voltage and then into temperature. Lastly, the results are printed in the Serial Monitor.

- **Output: -**





Conclusion: - Hence, we have successfully demonstrated the use of LM35 analog temperature sensor with Arduino UNO.

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 5

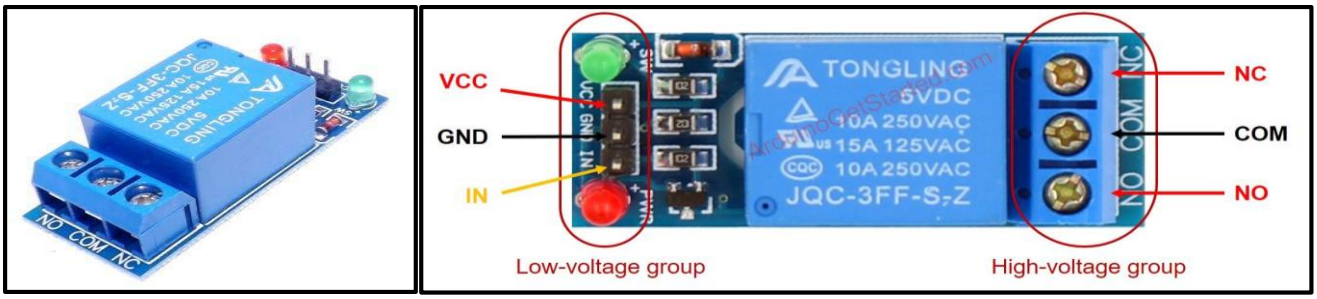
Aim :- Control action using Relay with Arduino/NodeMCU/Raspberry Pi to Turn it ON/OFF when Temperature increases or decreases.

Hardware Components: Arduino UNO, breadboard, bulb, resistor, temperature sensor, relay SPDT, power supply and connecting wires.

Software used: Tinkercad.

- **Relay:**

Relay is a programmable electrical switch which can be controlled by Arduino or any micro- controller. It is used to programmatically control on/off devices, which use the high voltage and/or high current. It is a bridge between Arduino and high voltage devices.



Relay has two groups of pins: Low voltage group and high voltage group.

- Pins in low voltage group are connected to Arduino, including three pins:
  - VCC pin: needs to be connected to VCC (5V).
  - IN pin: receives the control signal from the Arduino.
- Pins in high voltage group are connected to high voltage a device, including three pins (usually in screw terminal):
  - COM pin: It is the common pin. It is used in both normally open mode and normally closed mode.
  - NO pin: It is the normally open pin. It is used in normally open mode.
  - NC pin: It is the normally closed pin. It is used in normally closed mode.

  In practice, we usually do not use all of the pins in high voltage group, we use only two of them:
  - We use only COM and NO pin if we use normally open mode.
  - We use only COM and NC pin if we use normally closed mode.
- **How Relay works:**

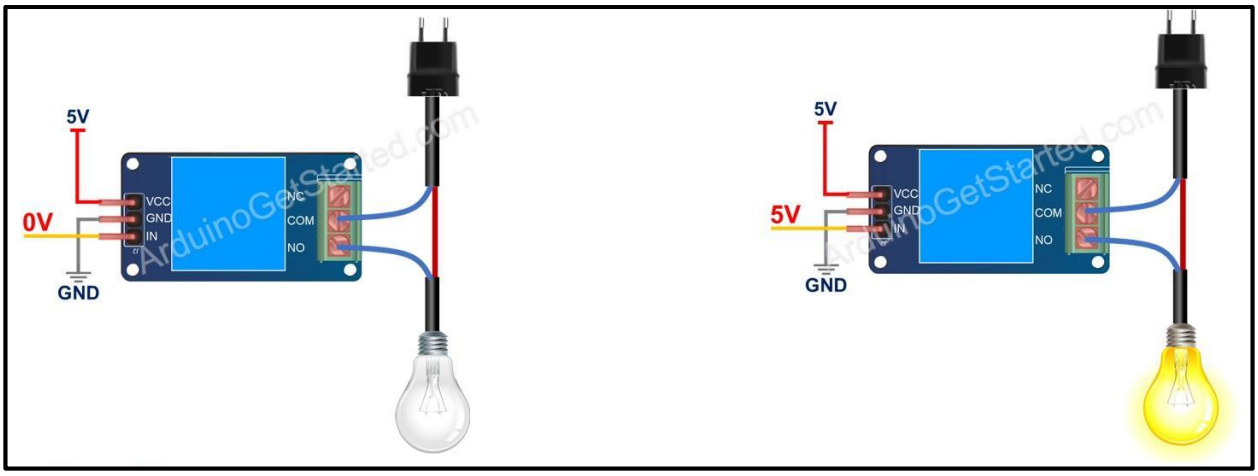
The relay can work with two modes: normally open mode and normally closed mode.

1. **Normally open mode:**

The "normally" means "if IN pin is connected to LOW (0V)".

To use this mode, we need to connect the high voltage device to COM pin and NO pin.

- If the IN pin is connected to LOW (0V), the switch is open. The device is off (inactive).
- If the IN pin is connected to HIGH (5V), the switch is closed. The device is on (active).



2. **Normally closed mode:**

To use this mode, we need to connect the high voltage device to COM pin and NC pin.

- If the IN pin is connected to LOW (0V), the switch is open. The device is off (inactive).
- If the IN pin is connected to HIGH (5V), the switch is closed. The device is on (active).

Arduino controls a high voltage device by controlling a relay.

Controlling a relay is simple. We just need:

- Connect an Arduino's pin to the IN pin of the relay.
- Control the relay by programming the pin to LOW or HIGH.

- **Temperature Sensor:**

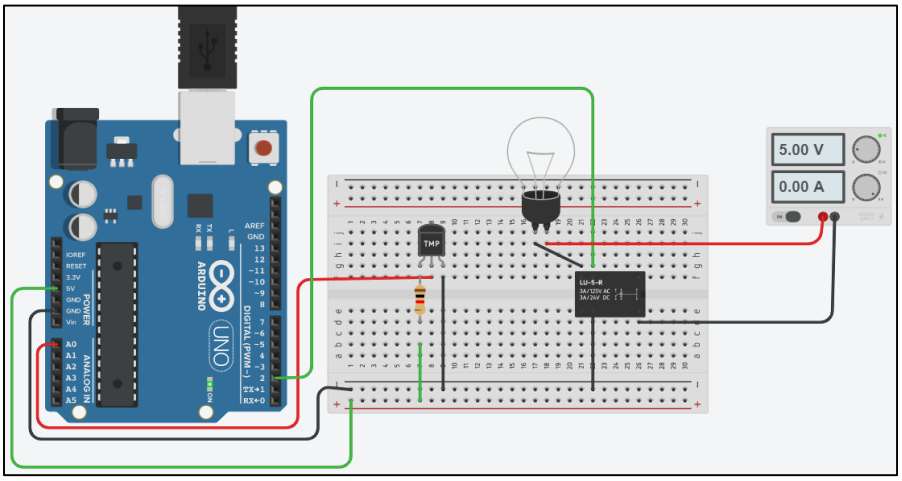
A temperature sensor is a device used to measure temperature. This can be air temperature, liquid temperature or the temperature of solid matter.

There are different types of temperature sensors available and they each use different technologies and principles to take the temperature measurement.

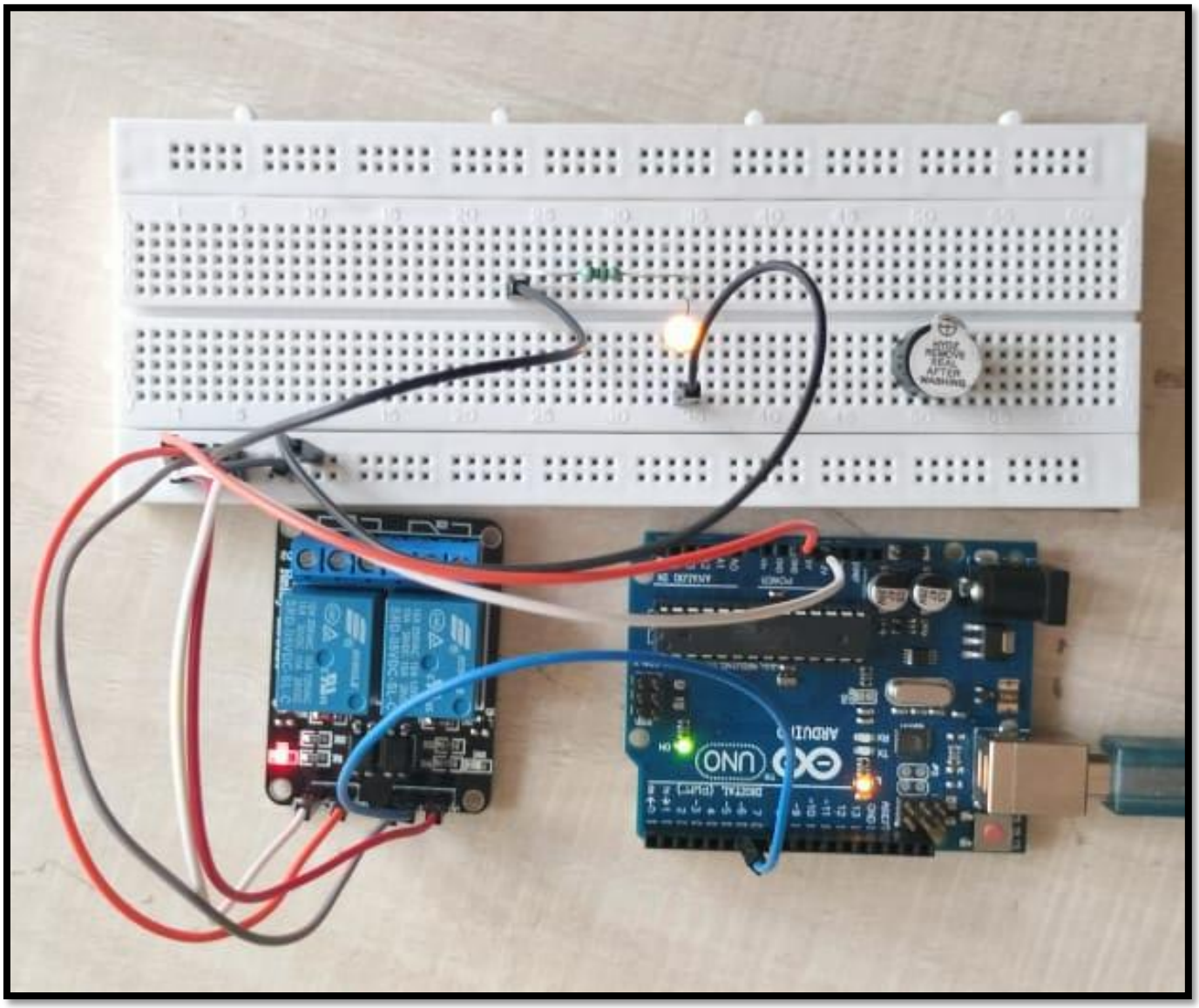
Temperature sensors are used to measure temperature in many different applications and industries. They are all around us; present in both everyday life and more industrial settings.

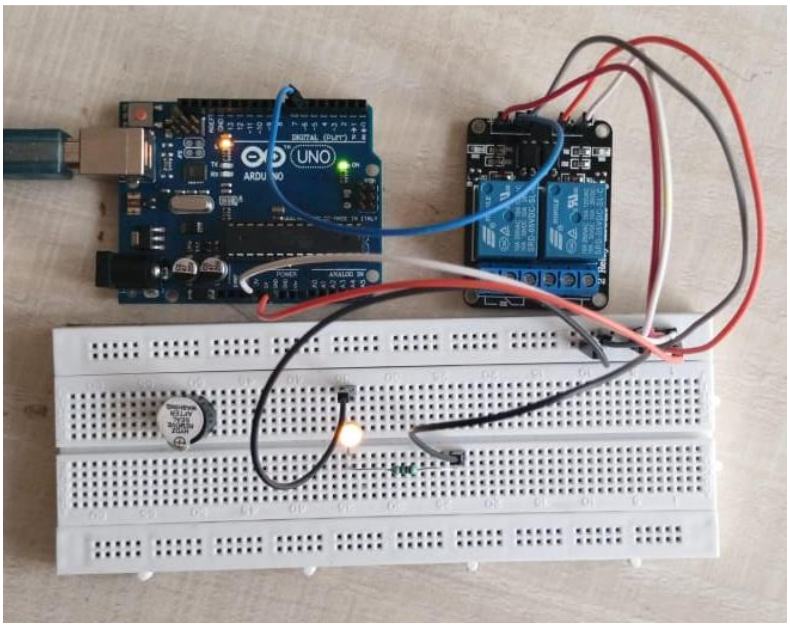
- **Control action using Relay with Arduino to Turn a bulb ON/OFF when Temperature increases or decreases.**
- **Circuit Schematic and Circuit:**
  - **Schematic**

- **Circuit:**



Actual Implementation on circuit





Code for actual hardware implementation

relay2 | Arduino 1.8.3

File Edit Sketch Tools Help

relay2

```
int RelayPin = 6;

void setup() {
  // Set RelayPin as an output pin
  pinMode(RelayPin, OUTPUT);
}

void loop() {
  // Let's turn on the relay...
  digitalWrite(RelayPin, LOW);
  delay(500);

  // Let's turn off the relay...
  digitalWrite(RelayPin, HIGH);
  delay(500);
}
```

Done uploading.

```
Sketch uses 936 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

15 Arduino Uno on COM3

83°F Sunny Search 12:21 PM 3/27/2023

CONCLUSION: Hence, we have successfully controlled action using Relay with Arduino to Turn a bulb ON/OFF when Temperature increases or decreases.

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 6

Aim :- Interfacing 12C LCD with Arduino/NodeMCU to display message.

Hardware Components: 12C LCD, 220 Ω Resistor, Breadboard, Connecting Wires and Arduino UNO.

Software used: Tinkercad.

- **12 C LCD:**
  - 12C_LCD is an easy to use display module, it can make display easier.
  - Using it can reduce the difficulty of make, so that makers can focus on the core of the work.
  - The Arduino library for 12C_LCD, user need just a few lines of the code can achieve complex graphics and text display features.
  - It can replace the serial monitor of Arduino in some place, we can get running information without a computer.
  - Version: 12C_LCD (with universal grove cable), 12C_LCD (with conversion grove cable)

- **Features:**
  - Only 2 Arduino pins are occupied (Use 12C interface).
  - Supports standard 12C mode (100bit/s) and fast 12C mode(400bit/s).
  - Compatible with multiple communication logic levels: 2.8~5VDC.
  - Arduino library supported, use a line of code to complete the display.
  - Integrate 7 sizes of ASCII fonts, 5 graphics functions.
  - Provide dedicated picture data convert software (Bitmap Converter).
  - Most of the complex operation is processed by 12C_LCD independent controller, saving user controller resources.
  - Supports cursor function, can set up 16 cursor flicker frequency.
  - Supports 128 level backlight lightness adjustment.
  - Support 64 level screen contrast adjustment.
  - Support device address modification.

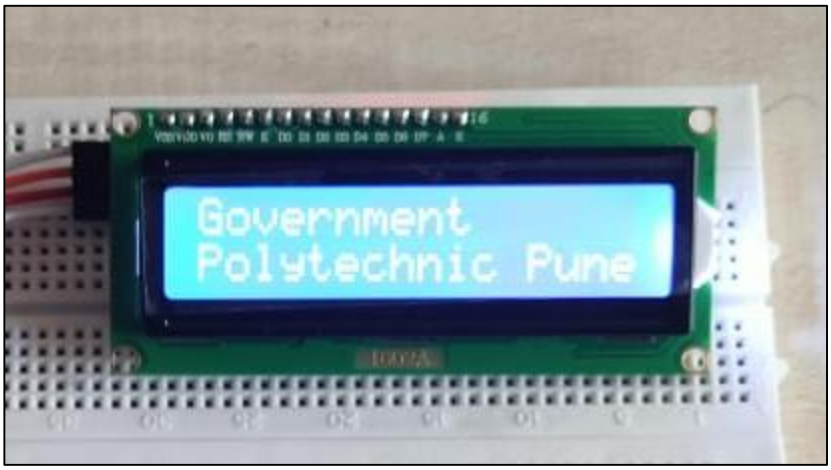


Fig. 6.1 LCD

- **Specification:**

Parameter	Value
Screen Type	Dual color LCD
Screen Resolution	28*64 pixels
Screen active area(L*W)	7.1*26.5mm
Individual pixel size	33*0.33mm
Communication mode	12C(100bit/s and 400bit/s)
Controller	STM8S005KBT6
Operating Frequency Weight	16MHz 0g

- **Pinout:**

LCD 12C uses 12C interface, so it has 4pins:

1. GND Pin: needs to be connected to GND(0V).
2. VCC Pin: the power supply for the LCD, needs to be connected to VCC(5V).
3. SDA Pin: 12C data signal.
4. SCL Pin: 12C clock signal

- **Circuit Schematic and Circuit:**
  - **Schematic**

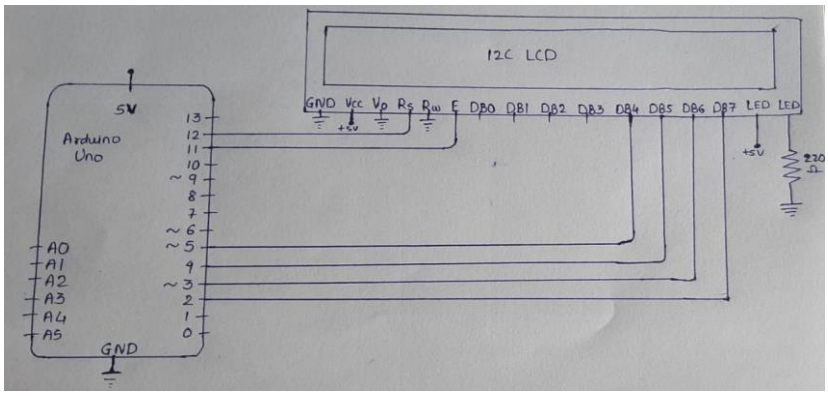
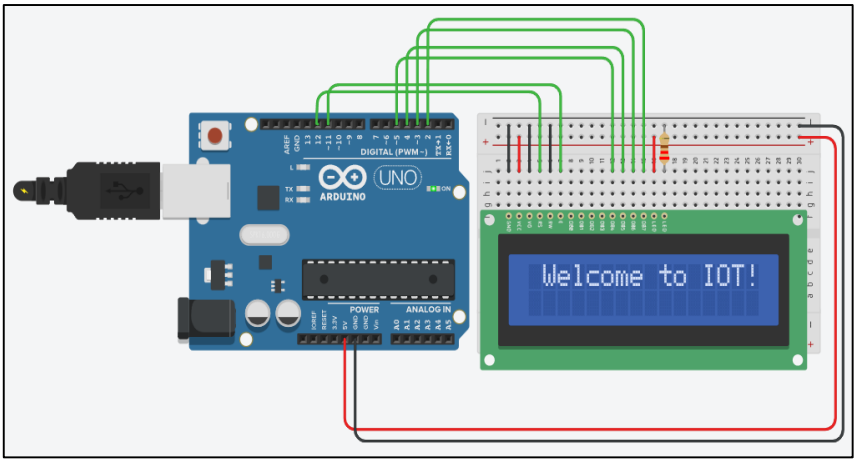


Fig. 6.2 Circuit Schematic

  - **Circuit:**



Actual implementation:

Code for actual implementation:

lcd-begin | Arduino 1.8.3

File Edit Sketch Tools Help

lcd-begin

```
#include <LiquidCrystal_I2C.h>
// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup()
{
  // initialize the LCD
  lcd.begin();
  // Turn on the backlight
  lcd.setBacklight(HIGH);

  lcd.setCursor(0,0);
  // Print a message to the LCD.
  lcd.print("Government");

  lcd.setCursor(0,1);
  lcd.print("Polytechnic Pune");
}
void loop()
{
  // Do nothing in the loop
```

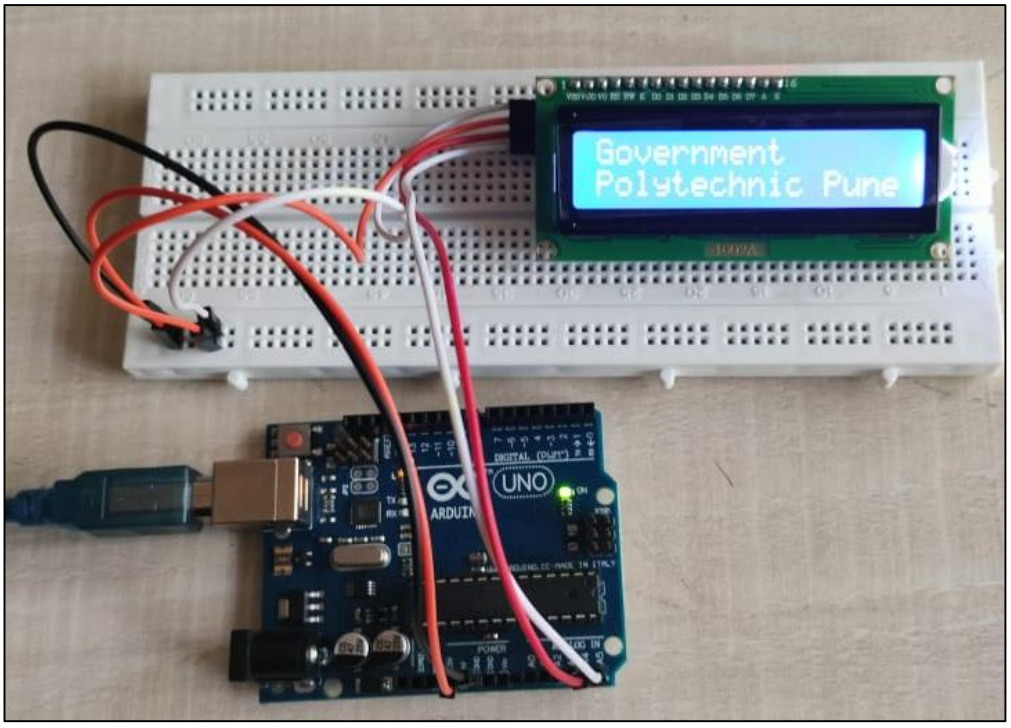
Done Saving.

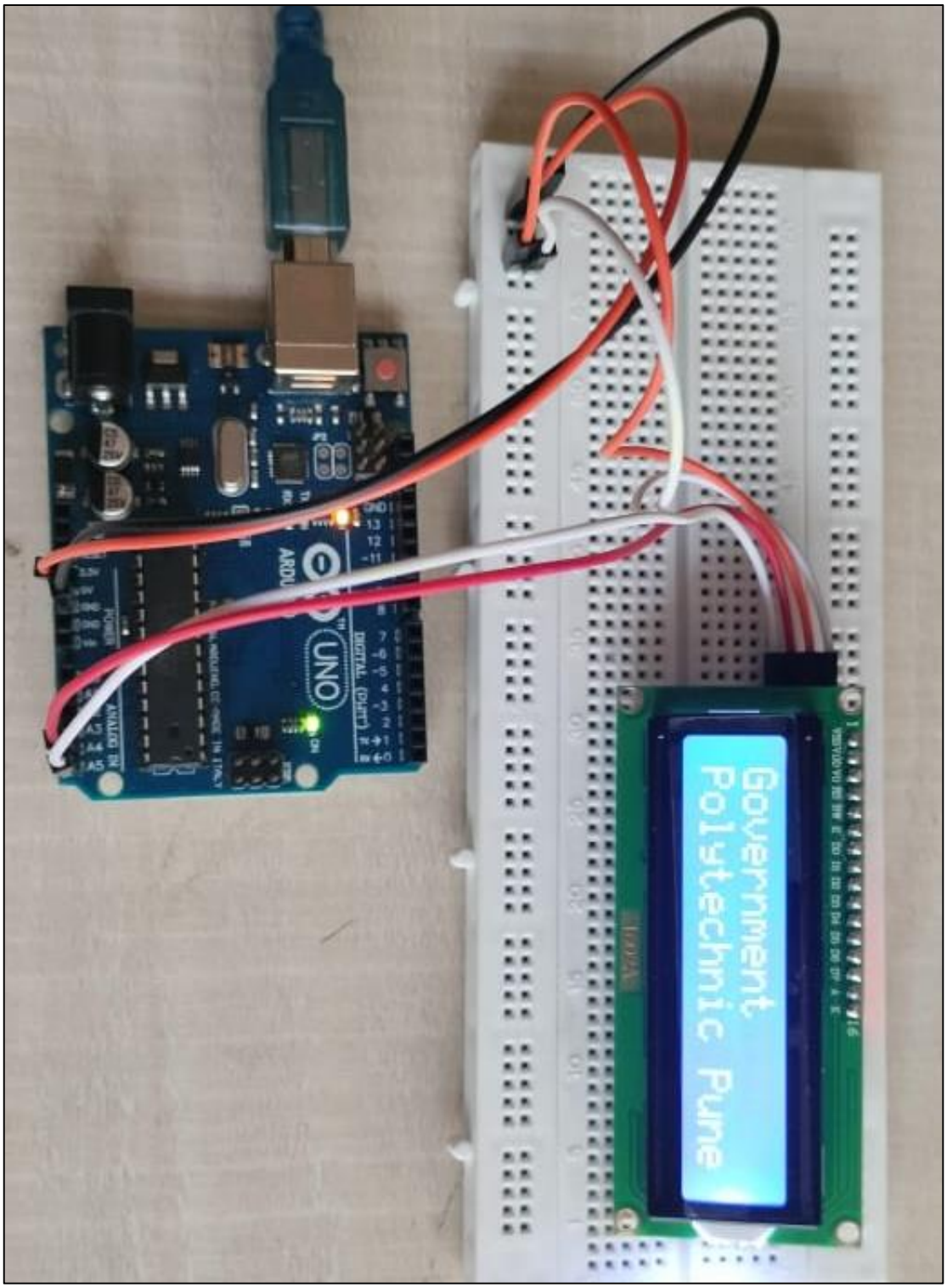
```
Sketch uses 3292 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 285 bytes (13%) of dynamic memory, leaving 1763 bytes for local variables. Maximum is 2048 bytes.
```

3 Arduino Uno on COM3

87°F Sunny Search 1:06 PM 3/27/2023

Output of actual implementation:





- **Conclusion**: We successfully completed practical on Interfacing I2C LCD with Arduino/NodeMCU to

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 7

Aim :- Interfacing DHT11 Sensor with Arduino/NodeMCU/Raspberry Pi to get Temperature and Humidity and display same on I2C LCD.

Hardware Components: Arduino Uno R3, LCD 16 x 2, Temperature Sensor [DHT11/TMP36], 250 kΩ Potentiometer, 1 kΩ Resistor, Breadboard, Connecting wires.

Software used: Tinkercad.

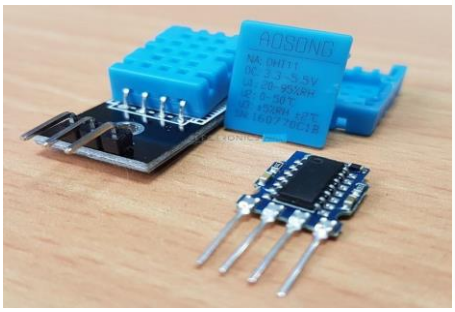
DHT11 Humidity Sensor

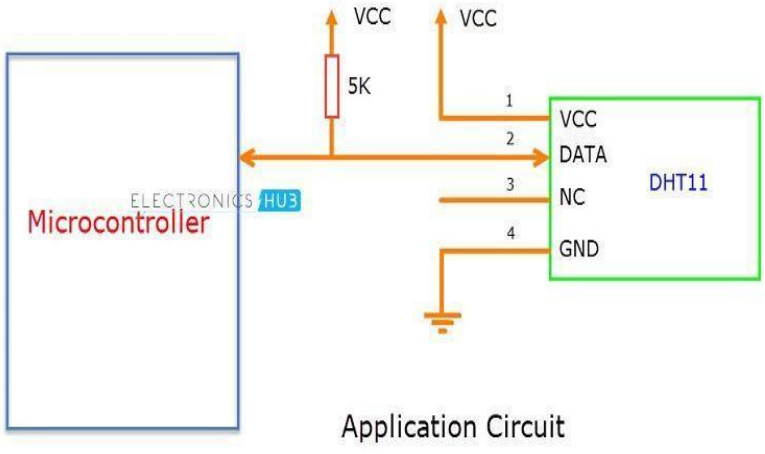
DHT11 is a Humidity and Temperature Sensor, which generates calibrated digital output. DHT11 can be interface with any microcontroller like Arduino, Raspberry Pi, etc. and get instantaneous results. DHT11 is a low cost humidity and temperature sensor which provides high reliability and long term stability.

Component Description

DHT11 Temperature and Humidity Sensor

DHT11 is a part of DHTXX series of Humidity sensors. The other sensor in this series is DHT22. Both these sensors are Relative Humidity (RH) Sensor. As a result, they will measure both the humidity and temperature.





The DHT11 Humidity and Temperature Sensor consists of 3 main components. A resistive type humidity sensor, an NTC (negative temperature coefficient) thermistor (to measure the temperature) and an 8-bit microcontroller, which converts the analog signals from both the sensors and sends out single digital signal.

This digital signal can be read by any microcontroller or microprocessor for further analysis.

DHT11 Humidity Sensor consists of 4 pins: VCC, Data Out, Not Connected (NC) and GND. The range of voltage for VCC pin is 3.5V to 5.5V. A 5V supply would do fine. The data from the Data Out pin is a serial digital data.

The following image shows a typical application circuit for DHT11 Humidity and Temperature Sensor. DHT11 Sensor can measure a humidity value in the range of 20 – 90% of Relative Humidity (RH) and a temperature in the range of 0 – 500C. The sampling period of the sensor is 1 second i.e.

All the DHT11 Sensors are accurately calibrated in the laboratory and the results are stored in the memory. A single wire communication can be established between any microcontroller like Arduino and the DHT11 Sensor.

Also, the length of the cable can be as long as 20 meters. The data from the sensor consists of integral and decimal parts for both Relative Humidity (RH) and temperature.

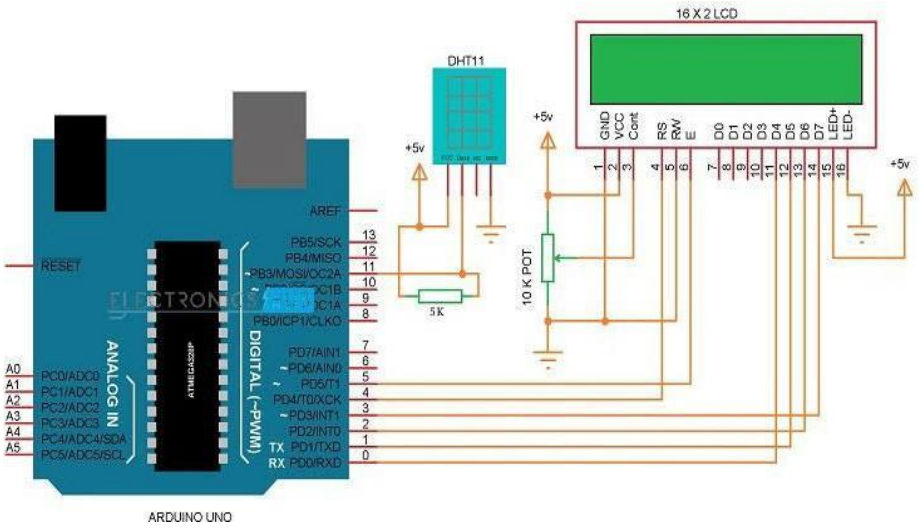
The data from the DHT11 sensor consists of 40 – bits and the format is as follow.

- Bit data for integral RH value,
- Bit data for decimal RH value,
- Bit data for integral Temperature value,
- Bit data for integral Temperature value and
- Bit data for checksum.

● Specification:

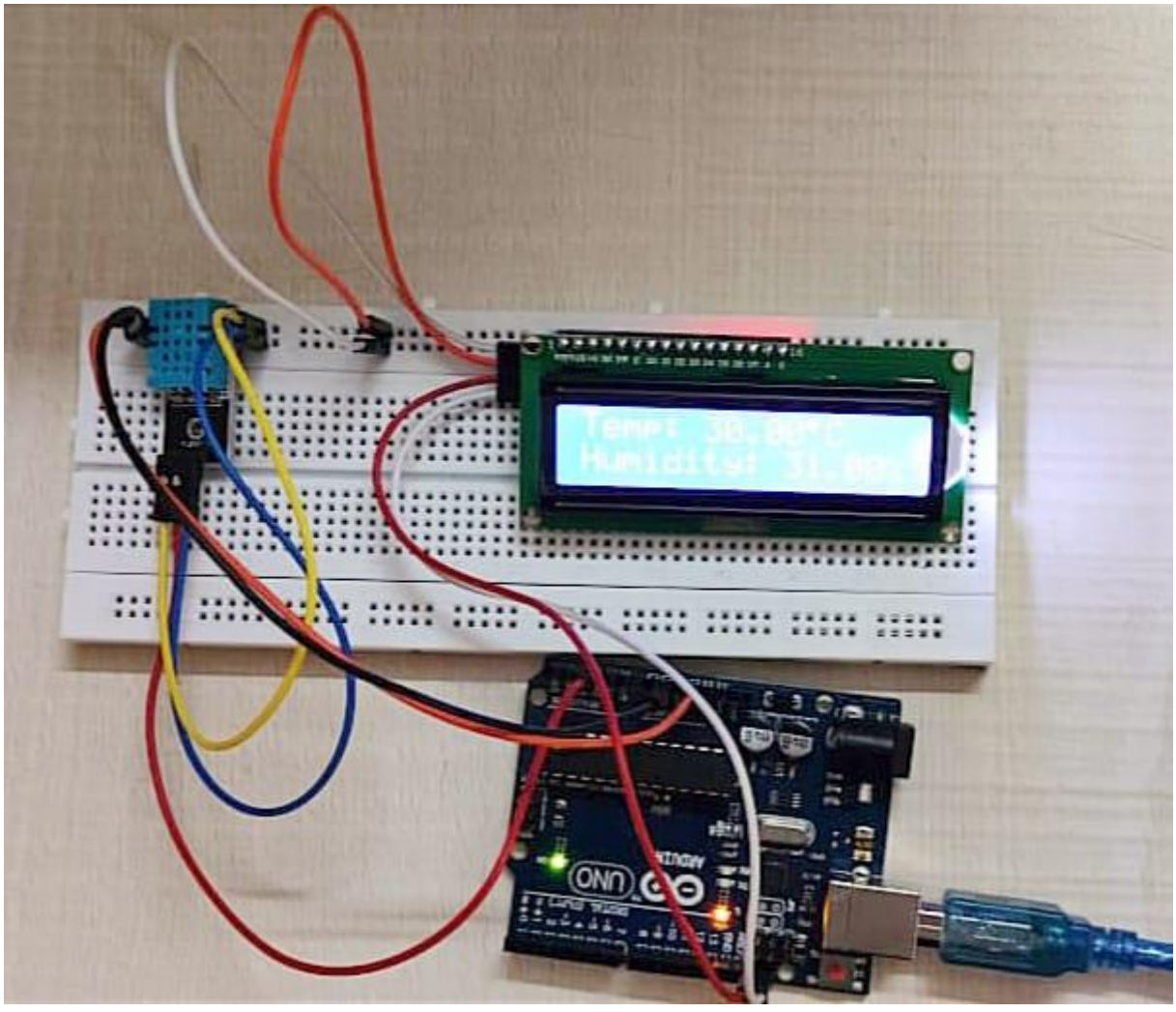
- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

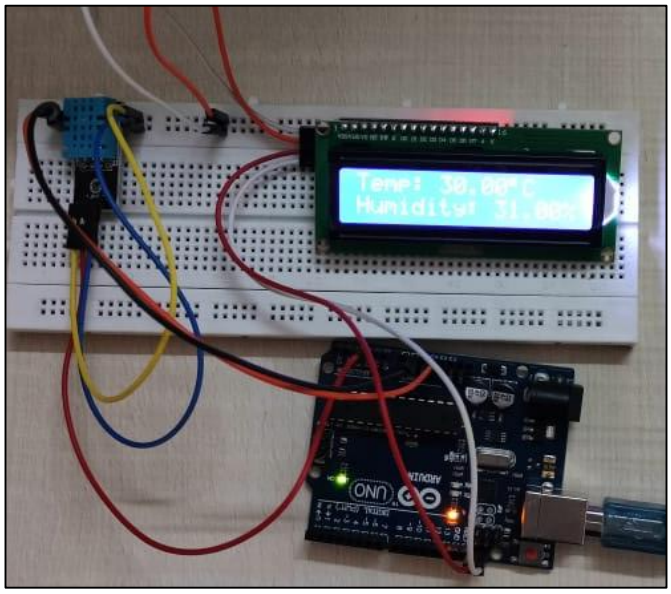
○ Circuit Schematic:



Actual implementation:

Output of actual implementation:





Code for actual implementation:

Practical-07 | Arduino 1.8.3

File Edit Sketch Tools Help

Practical-07 §

```
#include <LiquidCrystal_I2C.h>
#include <dht.h>
// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);
dht DHT;
#define DHT11_PIN A2
void setup()
{// initialize the LCD
  lcd.begin();
  // Turn on the backlight
  lcd.setBacklight(HIGH);
  int chk = DHT.read11(DHT11_PIN);
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(DHT.temperature);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0,1);
  lcd.print("Humidity: ");
  lcd.print(DHT.humidity);
  lcd.print("%");
  delay(300);
}
void loop()
{// Do nothing in the loop  }
```

Done compiling.

```
WARNING: Category '' in library SoftwareSerial is not valid. Setting to 'Uncategorized'
WARNING: Category '' in library Wire is not valid. Setting to 'Uncategorized'
Warning: platform.txt from core 'Arduino AVR Boards' contains deprecated recipe.ar.pattern="{compiler.path}{compiler.ar.cmd}" {compiler.ar.flags} {compiler.ar.extra_flags} "{build.path}/{archive_file}" "
Sketch uses 5794 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 298 bytes (14%) of dynamic memory, leaving 1750 bytes for local variables. Maximum is 2048 bytes.
```

25 Arduino Uno on COM4

Type here to search 34°C Sunny 4:43 PM 3/29/2023

Conclusion : We successfully completed practical on Interfacing DHT11 Sensor with Arduino/NodeMCU/Raspberry Pi to get Temperature and Humidity and display same on I2C LCD.

Knowledge	Skill	Presentation	Assessment	Total

Practical No.: 8

Aim :-Interfacing PIR Sensor with Arduino/NodeMCU/Raspberry Pi to Detect Motion.

Hardware Components: Arduino UNO, breadboard, resistor, potentiometer, buzzer, PIR sensor, LCD 16 x 2 and connecting wires.

Software used: Tinkercad.

PIR Sensor:

The PIR motion sensor is used to detect movement in particular area. PIR stand for "Passive Infrared". Basically, the PIR motion sensor measures infrared light which is coming from the objects within its field of range. So it can detect motion based on changes in infrared light in the environment which is coming from the human being.

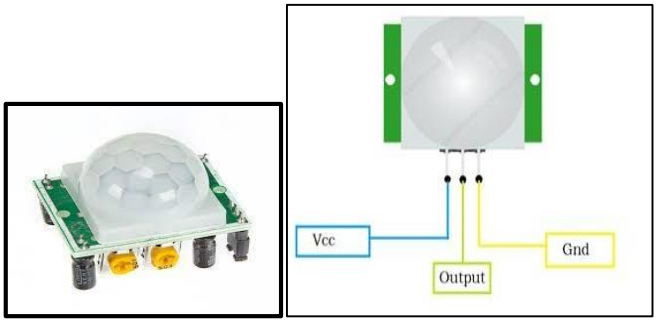


Fig.8.1 PIR Sensor

- **How PIR Sensor works:**

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a *positive differential* change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

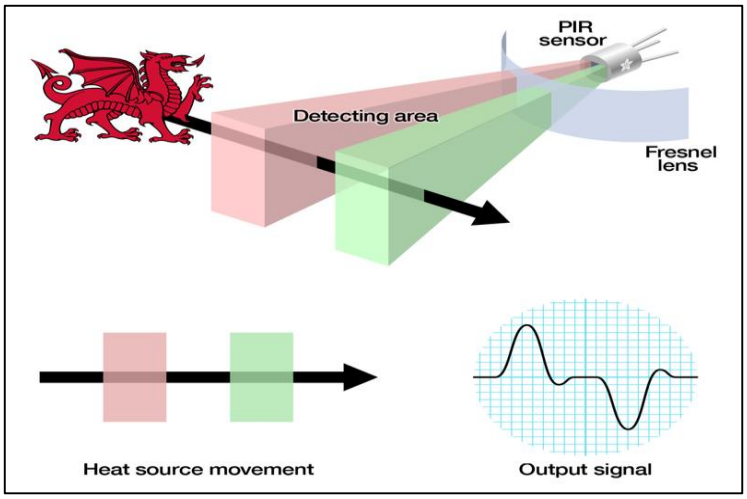


Fig 8.2 PIR Sensor Working

- **Connecting to PIR:**

Most PIR modules have a 3-pin connection at the side or bottom. The pinout may vary between modules so triple-check the pinout! It's often silkscreened on right next to the connection (at least, ours is!) One pin will be ground, another will be signal and the final one will be power. Power is usually 3-5VDC input but may be as high as 12V. Sometimes larger modules don't have direct output and instead just operate a relay in which case there is ground, power and the two switch connections.

The output of some relays may be 'open collector' - that means it requires a pullup resistor. If you're not getting a variable output be sure to try attaching a 10K pullup between the signal and power pins.

An easy way of prototyping with PIR sensors is to connect it to a breadboard since the connection port is 0.1" spacing. Some PIRs come with header on them already, the one's from adafruit have a straight 3-pin header on them for connecting a cable.

PIR connections - Connect the Gnd pin of sensor to the ground of Arduino. Vcc pin of the sensor to 5V of Arduino. And signal / output pin to digital pin 5 of Arduino board.

PIR terminals: Gnd, Vcc and signal pin. Gnd is considered as the negative pin and is connected to the ground of the system. Vcc basically powers up the pin typically 5V. Signal pin is the output pin.

2. Potentiometer:

A potentiometer is a type of position sensor. They are used to measure displacement in any direction. Linear potentiometers linearly measure displacement and rotary potentiometers measure rotational displacement. Potentiometers work by varying the position of a sliding contact across a uniform resistance. In a potentiometer, the entire input voltage is applied across the whole length of the resistor, and the output voltage is the voltage drop between the fixed and sliding contact as shown below.

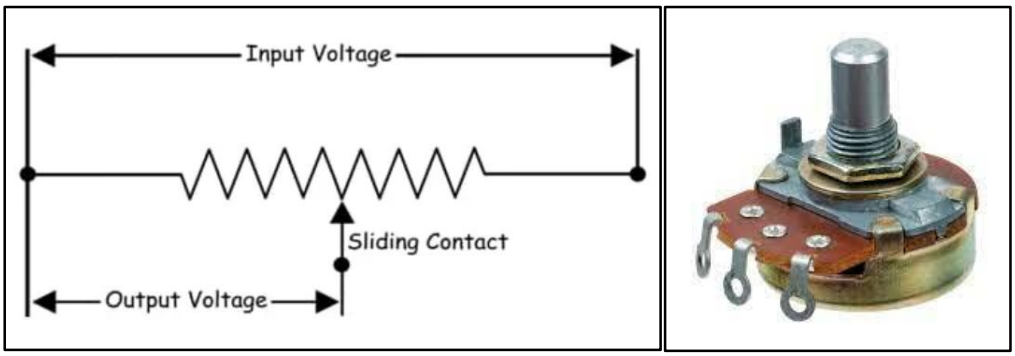


Fig. 8.4 Potentiometer

Interfacing PIR Sensor with Arduinoto Detect Motion.

- **Circuit Schematic and Circuit:**

Schematic

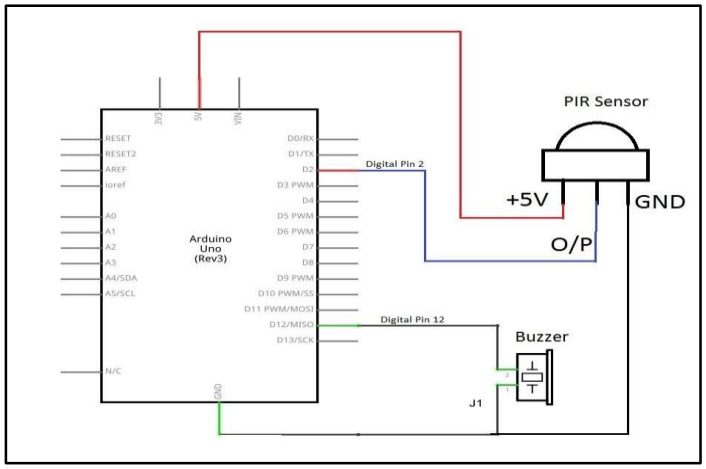


Fig.8.6 Schematic Circuit Diagram

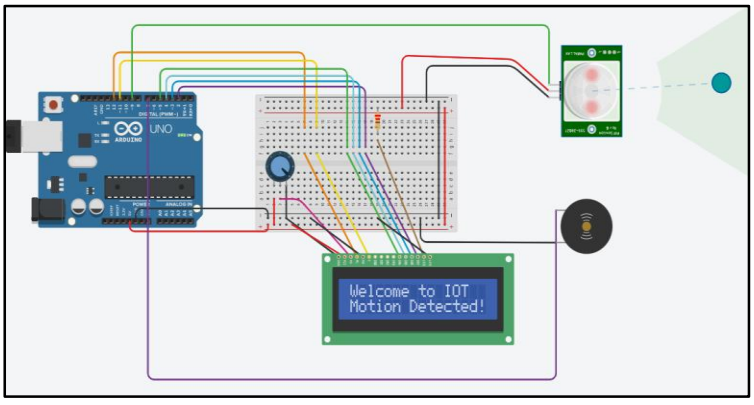
Actual Implementation of the Circuit:

Program:

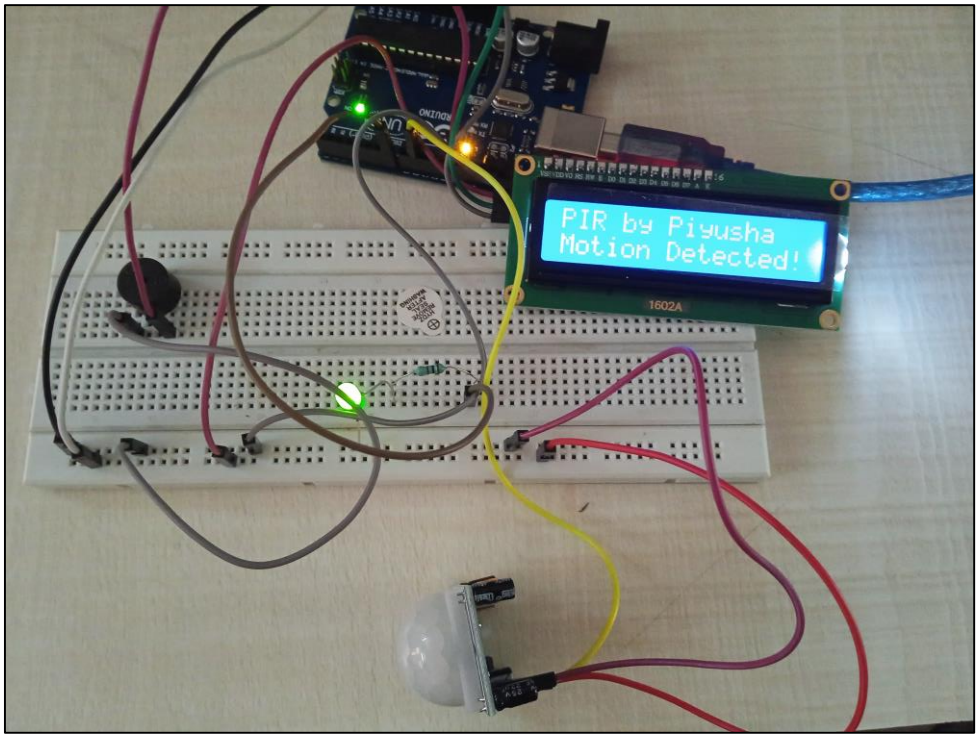
```
// C++ Code
//
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int sensor = 9;  // the pin that the sensor is atteched to
int state = LOW; // by default, no motion detected
int val = 0;
int buzzer= 7;  // the pin that the buzzer is atteched to
void setup()
{
pinMode(sensor, INPUT);      // initialize sensor as an
    input
pinMode(buzzer, OUTPUT); //initialize buzzer as an
    OUTPUT
lcd.begin(16, 2);
lcd.print("Welcome to IOT");
}

void loop()
{
val = digitalRead(sensor);   // read sensor value
  if (val == HIGH)
  {
  delay(100);              // delay 100 milliseconds
   if (state == LOW)
     {
lcd.setCursor(0, 1);
lcd.print("Motion Detected!");
digitalWrite(buzzer, HIGH);   // turn the LED/Buzz ON
    state = HIGH;       // update variable state to HIGH
      }
    }
  else
   {
    delay(200);           // delay 200 milliseconds
     if (state == HIGH)
     {
lcd.setCursor(0, 1);
lcd.print("Motion Stopped!");
digitalWrite(buzzer, LOW);   // turn the Buzzer ON
       state = LOW;       // update variable state to LOW
     }
   }
}
```

Output:

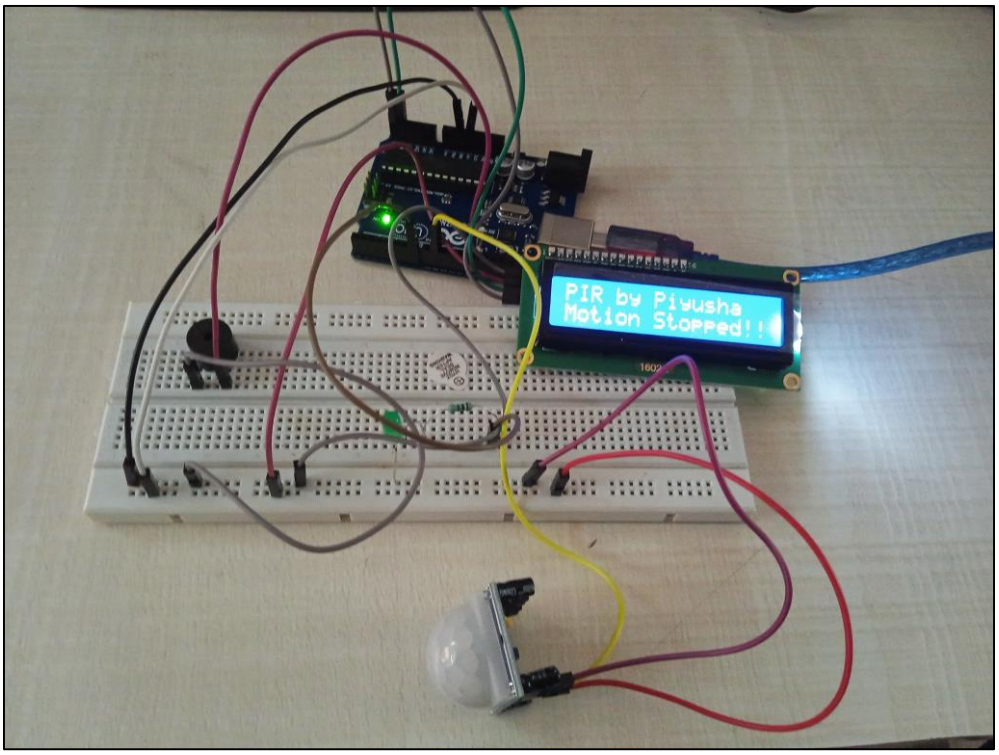
1. When motion is changed.



When the motion is changed and detected buzzer plays and led glows:



When the motion is stopped buzzer led stop



CONCLUSION: Hence, we have successfully interfaced PIR Sensor with Arduino to Detect Motion.

Knowledge	Skill	Presentation	Assessment	Total