Roll No. __45__ Exam Seat No.

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R. C. Marg,
Chembur, Mumbai – 400074. Contact No. 02261532532

Since 1962

CERTIFICATE

Certified that Mr. ________Pushkar Prasad Sane________________________

of ___________________________FYMCA/A___________________________ has

satisfactorily completed a course of the necessary experiments in

________Internet of Things__________ under my supervision

in the Institute of Technology in the academic year 20 23 – 2024.

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology, Collector Colony, Chembur, Mumbai, Maharashtra 400047
Department of M.C.A

INDEX

Sr. No.	**Experiment**	**CO Mapped**
1	Introduction to Basic Components: such as Arduino, Breadboard, Programming the Arduino and Basic electronic components such as LED, resistors, battery etc.	CO1
2	Working with Switches, LEDs, and More: such as interfacing LED's, Switches/push buttons and Speakers/Buzzer, LCD/ Seven Segment Display with Arduino	CO2, CO3
3	Programs based on interfacing LEDs, Potentiometer, photoresistor with Arduino Programs using PWM pins of Arduino, Programs using Serial Monitor of Arduino	CO2, CO3
4	Programs based on interfacing LEDs, Servo Motor, Potentiometer with arduino.	CO3, CO4
5	Programs based on interfacing DHT11/TMP36 temperature sensor Programs based on interfacing Passive infrared sensors (PIR), Ultrasonic of Arduino.	CO2, CO3
6	IoT with Cloud: Interfacing IoT device with Cloud	CO3, CO4

7	Programs based on interfacing LDR with Arduino.	CO3, CO4
8	Programs based on interfacing & segment as a counter with Arduino.	CO3, CO4
9	Module: Mini Project	ALL

Final Grade	**Instructor Signature**

Name of Student: Pushkar Sane			
Roll Number: 45		**Lab Assignment Number: 1**	
Title of Lab Assignment: Introduction to Basic Components: Such as Arduino, Breadboard, Programming the Arduino and Basic electronic components such as LED, Resistors, Battery etc.			
DOP: 09-01-2024		**DOS: 16-01-2024**	
CO Mapped: **CO1**	**PO Mapped:** **PO1, PO2, ,PO5, PO7, PSO1**		**Signature:**

Practical No. 1

Aim: Introduction to Basic Components such as Arduino, Breadboard, Programming the Arduino and Basic electronic components such as LED, Resistors, Battery etc.

Theory:

IOT :

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

An IoT system consists of sensors/devices which “talk” to the cloud through some kind of connectivity. Once the data gets to the cloud, software processes it and then might decide to perform an action, such as sending an alert or automatically adjusting the sensors/devices without the need for the user.

But if user input is needed or if the user simply wants to check in on the system, the user interface allows them to do so. Any adjustments or actions that the user makes are then sent in the opposite direction through the system: from the user interface, to the cloud, and back to the sensors/devices to make some kind of change.

Basic Components:

1. **Arduino:**

   Arduino acts as the brain of the system and processes the data from the sensor. Arduino is an open source hardware platform that is readily available for hobbyists & enthusiasts across the globe to build projects. It comes with an ATMEGA microcontroller that processes the data and facilitates the proper working of the IoT system. And the beauty is that the Arduino can be programmed ‘n’ number of times making it possible for you to build various types of IoT projects just by changing a simple code. You need to use C++ language for Arduino programming. Also IDE software is needed for Arduino based IoT projects. And you need to use the ESP-8266 WiFi module to establish the WiFi communication between the Arduino and cloud platform.

It 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 Conclusion:

Hence, we explored the Basic Components and electronic components in IOT.

Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. 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. You can tell your board what to do by sending a set of instructions to the microcontroller on the board.

2. **Breadboard:**

A breadboard is a solderless device for temporary prototypes with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connects the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

It is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used when slicing bread. In the 1970s the solderless breadboard became available and nowadays the term “breadboard” is commonly used to refer to these.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. Older breadboard types did not have this property. A stripboard and similar prototyping printed circuit boards, which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused.

Basic Electronic Components:

1. **LED**

In the simplest terms, a light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. By connecting lighting systems to IoT, it becomes possible to control lighting through user’s devices. LED bulbs produce more light than traditional bulbs, but because they do so more efficiently, less of the energy radiates from the bulb as heat. When they were first introduced, they produced more of a “cool” white light; today they offer the same natural-looking “warm” light as incandescents. IoT-based LED lighting has the potential to offer important features and capabilities such as: Ease of monitoring LED luminaire status and the life cycle of LED lighting elements, Generation of analytics to improve lighting system performance, Tracking of user behavior to help optimize the user experience in a home or building, or to attract attention to product displays in retail spaces, Personalization of lighting experiences by connecting, for example, to a profile on a user’s smart device.

2. **Resistors**

   Resistors are electronic components, which offer resistance against the current flow, or speaking at a deeper level, against the electrons flow. Resistors, denoted by R, are passive components, which means that they don't generate any electricity at all, but rather reduce voltage and current by dissipating power in the form of heat. The unit of resistance is ohms (Ω) and resistors are usually built using carbon or metal wire. You will also find the resistors being color-coded in order to help convey the value of resistance they offer.

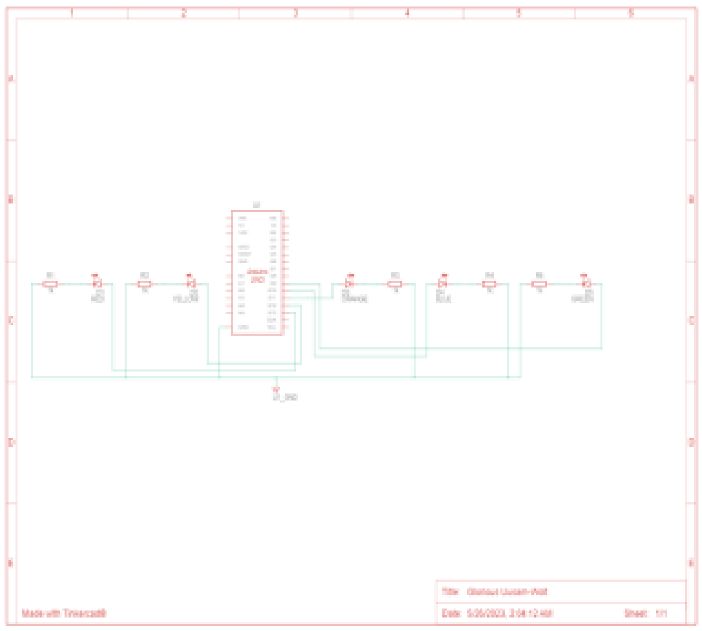
3. **Battery**

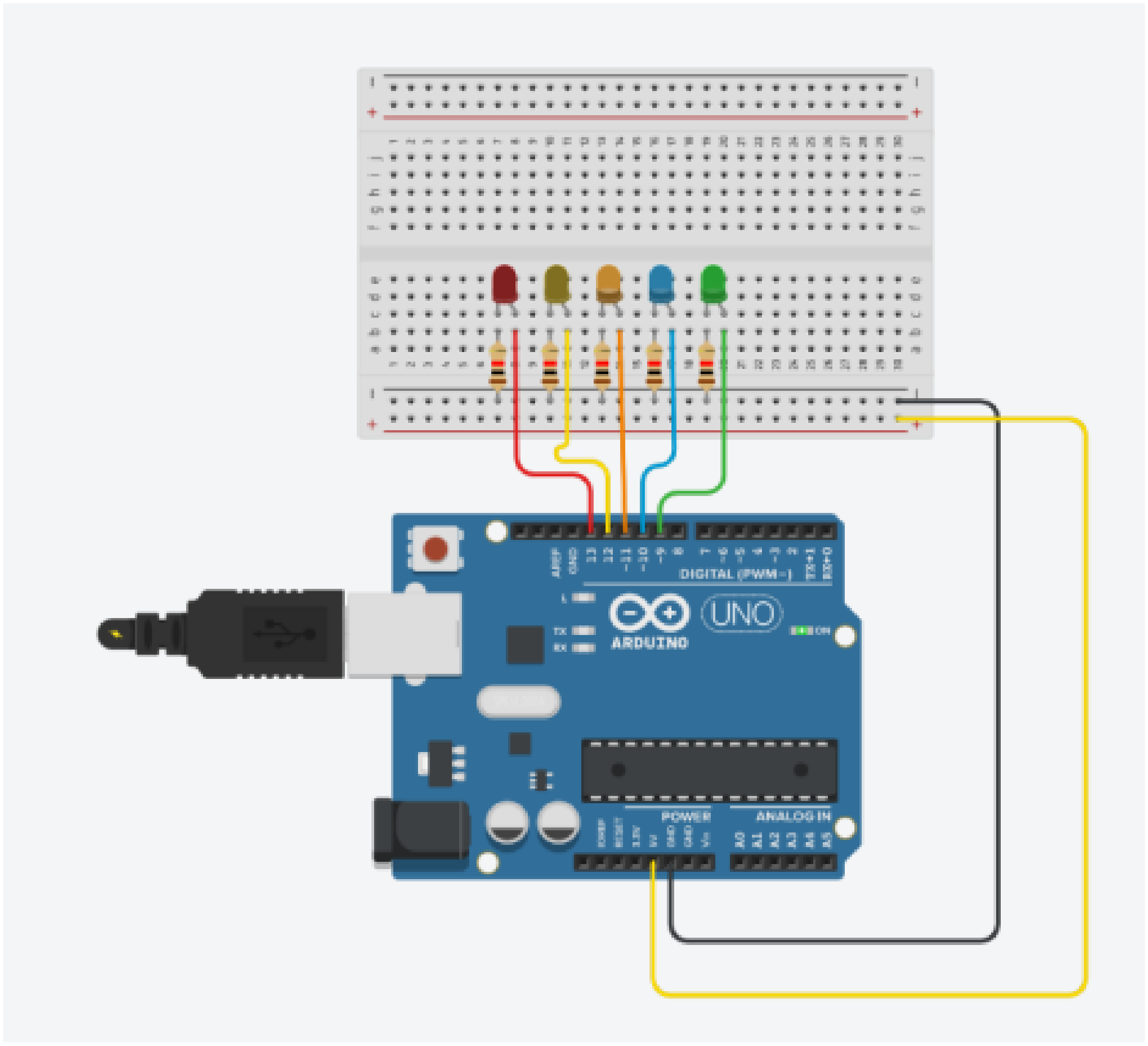
   A battery is a power source consisting of one or more electrochemical cells with external connections for powering IOT devices. When a battery is supplying electric power, its positive terminal is the cathode and its negative terminal is the anode. The terminal marked negative is the source of electrons that will flow through an external electric circuit to the positive terminal.

Code:

```
// C++ code
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
}
void loop() {
  for(int i=13;i>7;i--){
    digitalWrite(i,HIGH);
    delay(30);
    digitalWrite(i,LOW);
    delay(30);
  }
  for(inti=7;i<=13;i++){
    digitalWrite(i,HIGH);
    delay(30);
    digitalWrite(i,LOW);
    delay(30);
  }
}
```

Circuit Diagram:


RED
YELLOW
Arduino UNO
ORANGE
BLUE
GREEN
Made with Tinkercad®
Title: Glorious Uusam-Wolt
Date: 5/26/2023, 2:04:12 AM
Sheet: 1/1



Conclusion:

Hence, we explored the Basic Components and electronic components in IOT.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 2**	
Title of Lab Assignment: Working with Switches, LEDs, and More: Such as interfacing LED's, Switches/push buttons and Speakers/Buzzer, LCD/ Seven Segment Display with Arduino.		
DOP: 16-01-2024	**DOS: 23-01-2024**	
CO Mapped: **CO2, CO3**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

Practical No. 2

Aim: Working with Switches, LEDs, and More: Such as interfacing LED's, Switches/push buttons and Speakers/Buzzer, LCD/ Seven Segment Display with Arduino.

Theory:

LCD:

These LCDs are ideal for displaying text/characters only, hence the name 'Character LCD'. The display has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row. Each rectangle contains a grid of 5×8 pixels. If you look closely, you can actually see the little rectangles for each character on the display and the pixels that make up a character. Each of these rectangles is a grid of 5×8 pixels. Although they display only text, they do come in many sizes and colors: for example, 16×1, 16×4, 20×4, with white text on blue background, with black text on green and many more.

Buzzer:

A "piezo buzzer" is basically a tiny speaker that you can connect directly to an Arduino. "Piezoelectricity" is an effect where certain crystals will change shape when you apply electricity to them. By applying an electric signal at the right frequency, the crystal can make sound. If your buzzer has a sticker on top of it, pull the sticker off. From the Arduino, you can make sounds with a buzzer by using tone. You have to tell it which pin the buzzer is on, what frequency (in Hertz, Hz) you want, and how long (in milliseconds) you want it to keep making the tone.

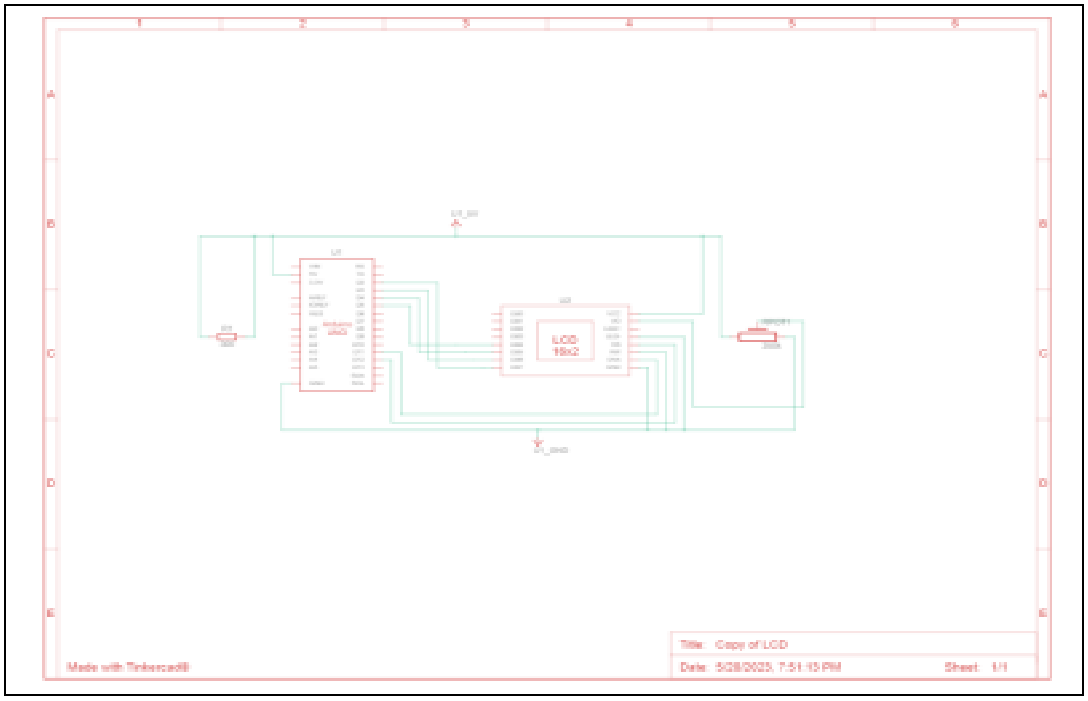
Push Button:

The buttons are similar to switches that create and break electrical connections in the circuits. The button plays a transition between ON and OFF state. A single press turns the state ON, while another press turns the state OFF. It means that the button connects the two points in a circuit when we press them.

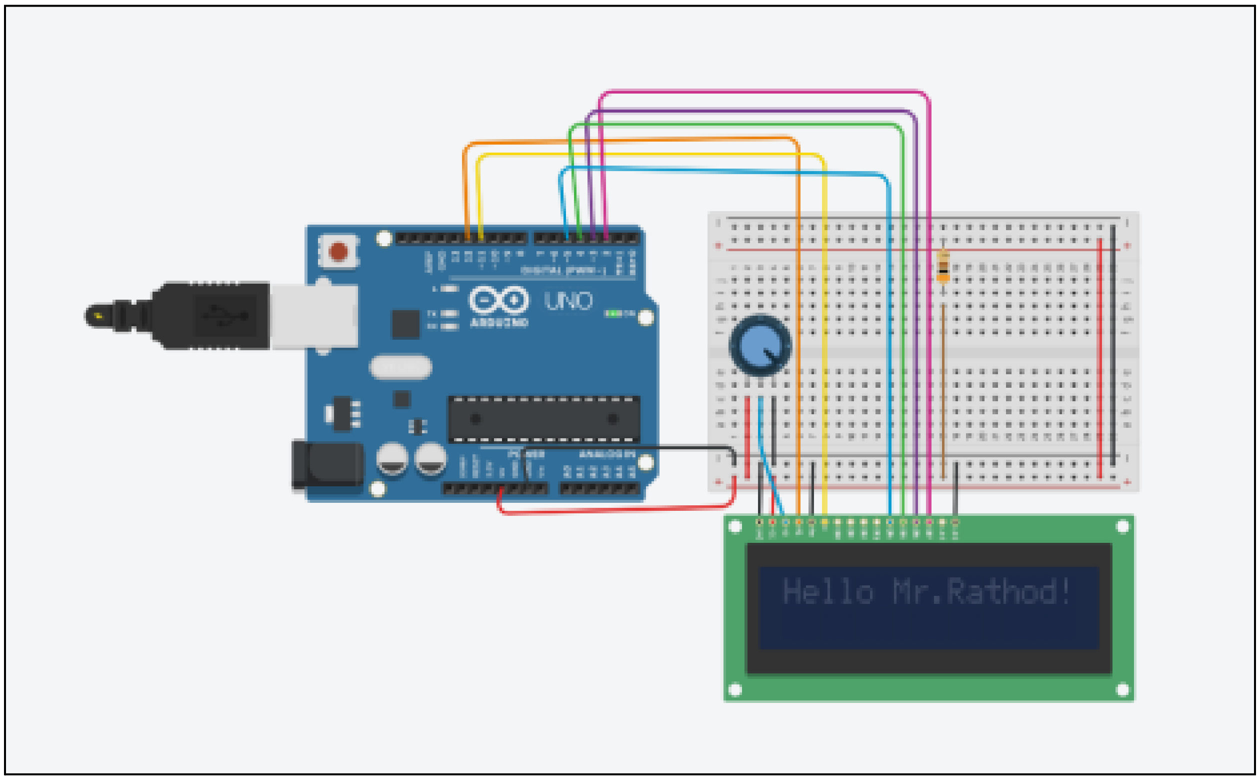
Code:

```
// C++ code
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
   // set up the LCD's number of columns and rows:
   lcd.begin(16, 2);
   // Print a message to the LCD.
   lcd.print("Hello Mr.Rathod!!");
}
void loop() {
   // set the cursor to column 0, line 1
   // (note: line 1 is the second row, since counting begins with 0):
   lcd.setCursor(0, 1);
   // print the number of seconds since reset:
   //lcd.print(millis() / 1000);
}
```

Circuit Diagram:



Output:



Conclusion:

Hence, we have seen the working of LCD with Arduino.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 3**	
Title of Lab Assignment: Programs based on interfacing LEDs, Potentiometer, photoresistor with arduino programs using PWM pins of Arduino, Programs using Serial monitor of Arduino.		
DOP: 23-01-2024	**DOS: 30-01-2024**	
CO Mapped: **CO2, CO3**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

Practical No. 3

Aim: Programs based on interfacing LEDs, Potentiometer, photoresistor with arduino programs using PWM pins of Arduino, Programs using Serial monitor of Arduino.

Theory:

LDR (Light Dependent Resistor):

An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. Light Dependent Resistors (LDR) are also called photoresistors.

They are made of high resistance semiconductor material. When light hits the device, the photons give electrons energy. This makes them jump into the conductive band and thereby conduct electricity.

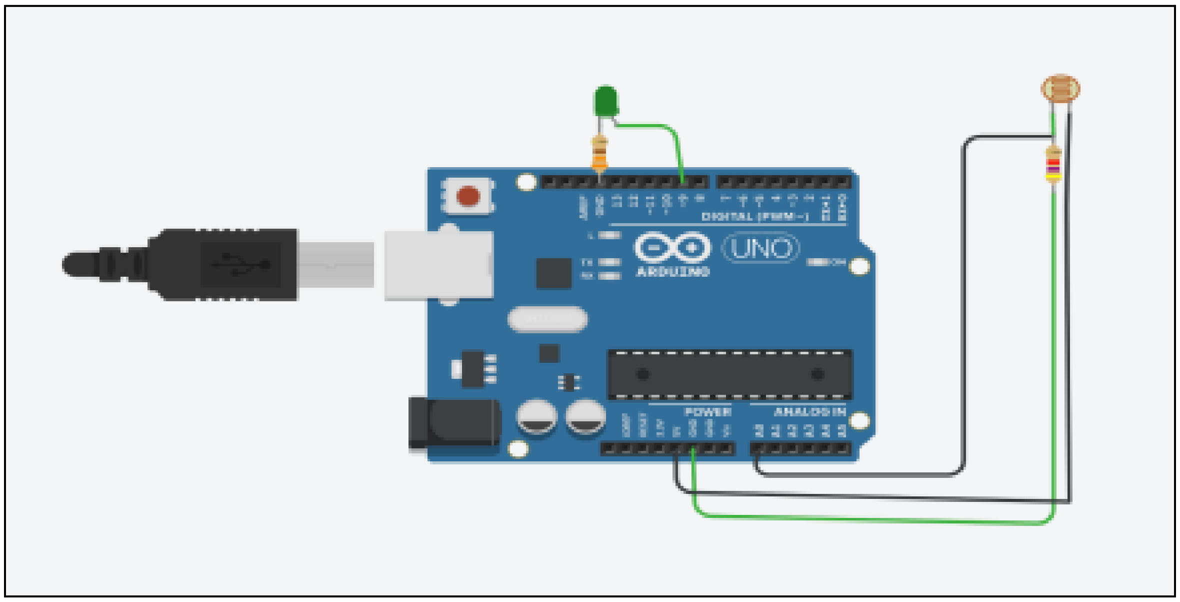
Light dependent resistors, LDRs or photoresistors are often used in circuits where it is necessary to detect the presence or the level of light.

They can be described by a variety of names from light dependent resistor, LDR, photoresistor, or even photocell, photocell or photoconductor.

A Light Sensor is something that a robot can use to detect the current ambient light level - i.e. how bright/dark it is.

There are a range of different types of light sensors, including 'Photoresistors', 'Photodiodes', and 'Phototransistors'.

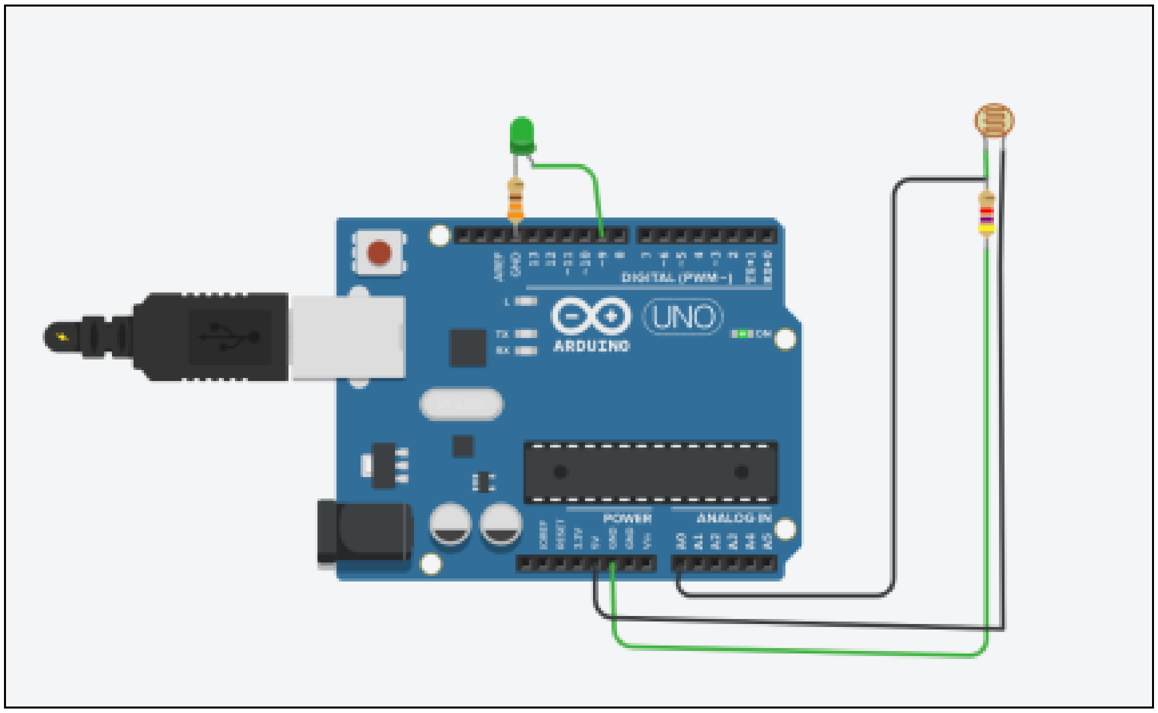
Circuit Diagram:



Code:

```
// C++ code
int photosensor = 0;
void setup() {
    pinMode(A0,INPUT);
    Serial.begin(9600);
    pinMode(9,OUTPUT);
}

void loop() {
    photosensor = analogRead(A0);
    Serial.println(photosensor);
    analogWrite(9,map(photosensor,0,1023,0,255));
    delay(20);
}
```

Output:



Conclusion:

From this practical, I have learned and implemented a photoresistor with an arduino.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 4**	
Title of Lab Assignment: Programs based on interfacing LEDs, Servo Motor, Potentiometer with arduino.		
DOP: 30-01-2024	**DOS: 06-02-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

Practical No. 4

Aim: Programs based on interfacing LEDs, Servo Motor, Potentiometer with arduino.

Theory:

Servo Motor:

A servomotor is a linear actuator or rotary actuator that allows for precise control of linear or angular position, acceleration, and velocity. It consists of a motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servo motors.

Servo motor applications are also commonly seen in remote-controlled toy cars for controlling the direction of motion, and it is also very widely used as the motor which moves the tray of a CD or DVD player. Besides these, there are hundreds of servo motor applications we see in our daily life.

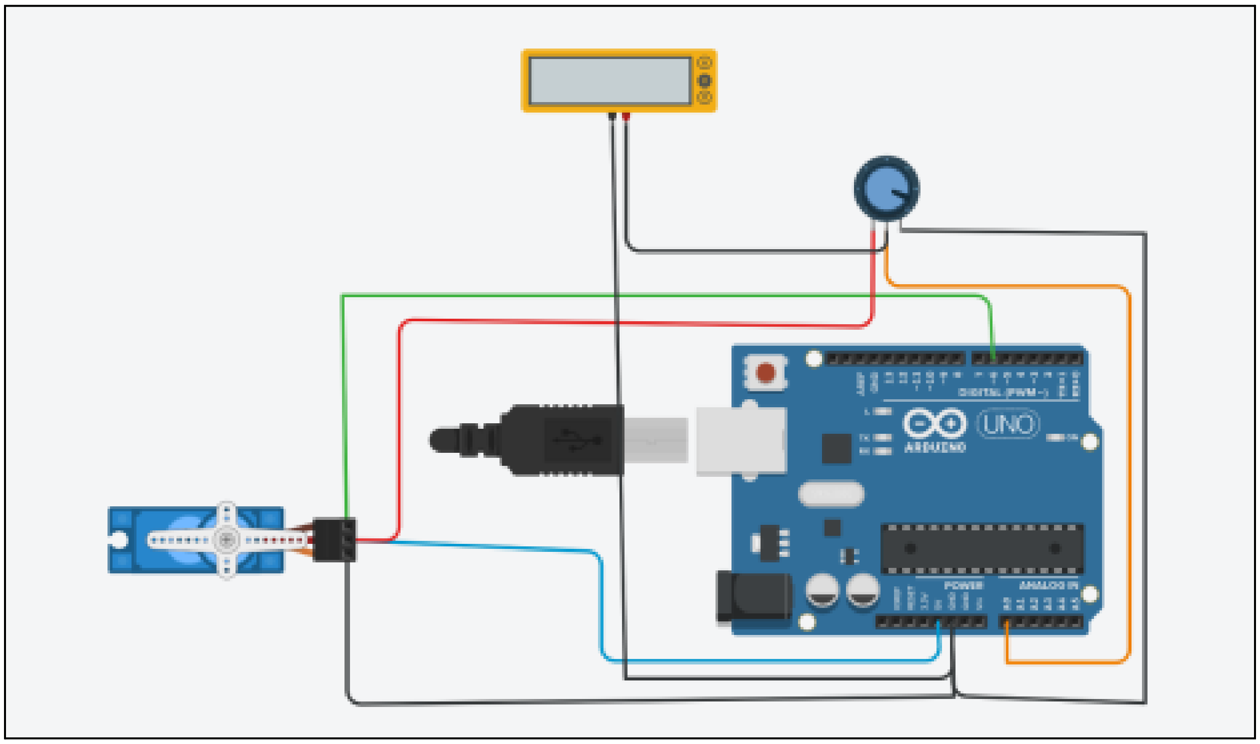
Potentiometer:

A potentiometer is a manually adjustable variable resistor with 3 terminals. Two terminals are connected to both ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element. The position of the wiper determines the output voltage of the potentiometer. The potentiometer essentially functions as a variable voltage divider.

The resistive
element can be seen as two resistors in series(potentiometer resistance), where the wiper position determines the resistance ratio of the first resistor to the second resistor. In order to use the potentiometer as a rheostat or variable resistor, it should have only two terminals with one end and the wiper.

A potentiometer is also commonly known as a potmeter or pot. The most common The form of the potmeter is the single turn rotary potmeter. This type of pot is often used in audio volume control (logarithmic taper) as well as many other applications. Different materials are used to construct potentiometers, including carbon composition, cermet, wirewound, conductive plastic or metal film.

Circuit Diagram:

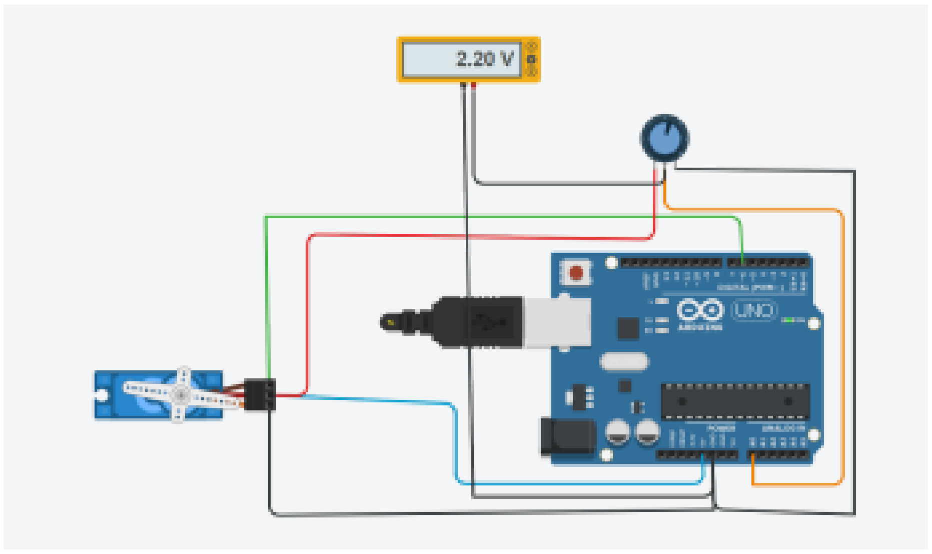


Code:

```
#include<Servo.h>
Servo myservo;
const int p = A0;
void setup() {
   pinMode(p,INPUT);
   myservo.attach(6);
   Serial.begin(9600);
}
void loop() {
   int pValue = analogRead(p);
   delay(100);
   myservo.write(pValue);
   delay(500);
   Serial.println("Value:" + pValue);
```

```
    /*for(int ang = 0; ang < 180; ang++) {
        myservo.write(ang);
        delay(10);
    }
    for(int ang = 180; ang > 0; ang--) {
        myservo.write(ang);
        delay(10);
    }*/
}
```

Output:



Conclusion: Successfully Learned and implemented Servo motor in Arduino.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 5**	
Title of Lab Assignment: Programs based on interfacing DHT11/TMP36 temperature sensor Programs based on interfacing Passive infrared sensors (PIR), Ultrasonic of Arduino.		
DOP: 06-02-2024	**DOS: 13-02-2024**	
CO Mapped: **CO2, CO3**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

Practical No. 5

Aim: Programs based on interfacing DHT11/TMP36 temperature sensor Programs based on interfacing Passive infrared sensors (PIR), Ultrasonic of Arduino.

Theory:

DHT11

The DHT11 humidity and temperature sensor make it really easy to add humidity and temperature data to your DIY electronics projects. It’s perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems. The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed). It’s fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, the sensor readings can be up to 2 seconds old.

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet. The operation is not affected by sunlight or black material, Although acoustically, soft materials like cloth can be difficult to detect. It comes complete with an ultrasonic transmitter and receiver module.

The DHT11 is a commonly used Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers. The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

How to use DHT11 Sensor:

The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. If you are trying to interface DHT11 with Arduino then there are ready-made libraries for it which will give you a quick start. If you are trying to interface it with some other MCU then the datasheet given below will come in handy. The output given out by the data pin will be in the order of 8 bit humidity integer data + 8 bit the Humidity decimal data +8 bit temperature integer data + 8 bit fractional temperature data +8 bit parity bit.

Applications:

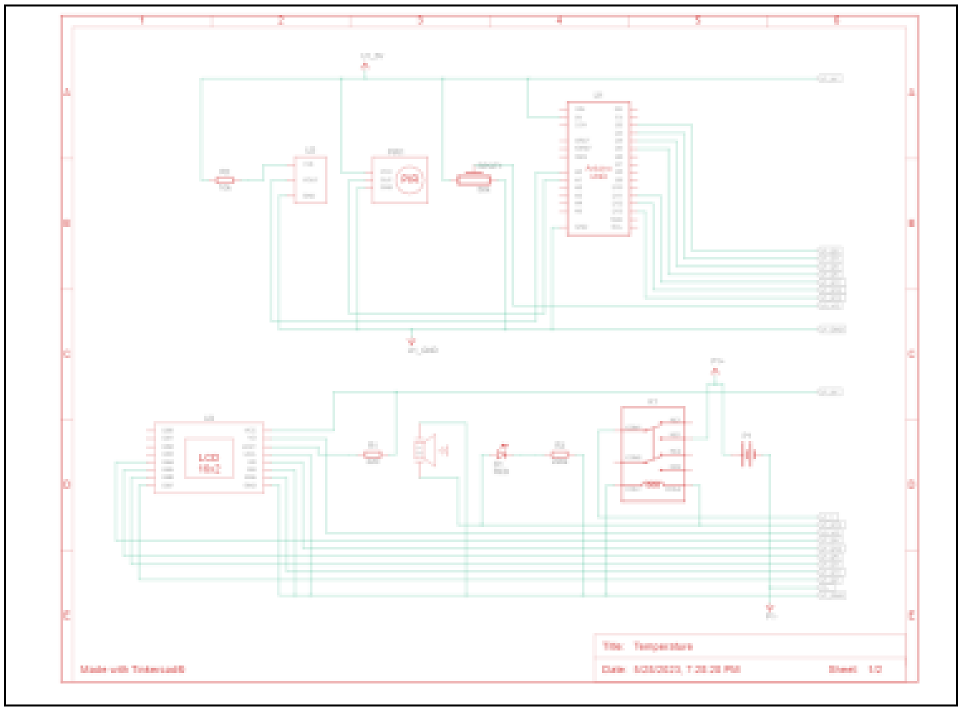
1. Measure temperature and humidity
2. Local Weather station
3. Automatic climate control
4. Environment monitoring

Code:

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
int val;
int cel = 0;
int tempin = A0;
void setup() {
   lcd.begin(16,2);
   pinMode(13,OUTPUT);
   Serial.begin(9600);
}
void loop() {
// For ultrasonic
   val = 30;
   cel = analogRead(A1);
   if(cel >= 100) {
      digitalWrite(13, HIGH);
      lcd.setCursor(0,1);
      lcd.print("Alert!!! Someone ");
   } else {
      digitalWrite(13, LOW);
      lcd.print(" Wait ");
   }
```

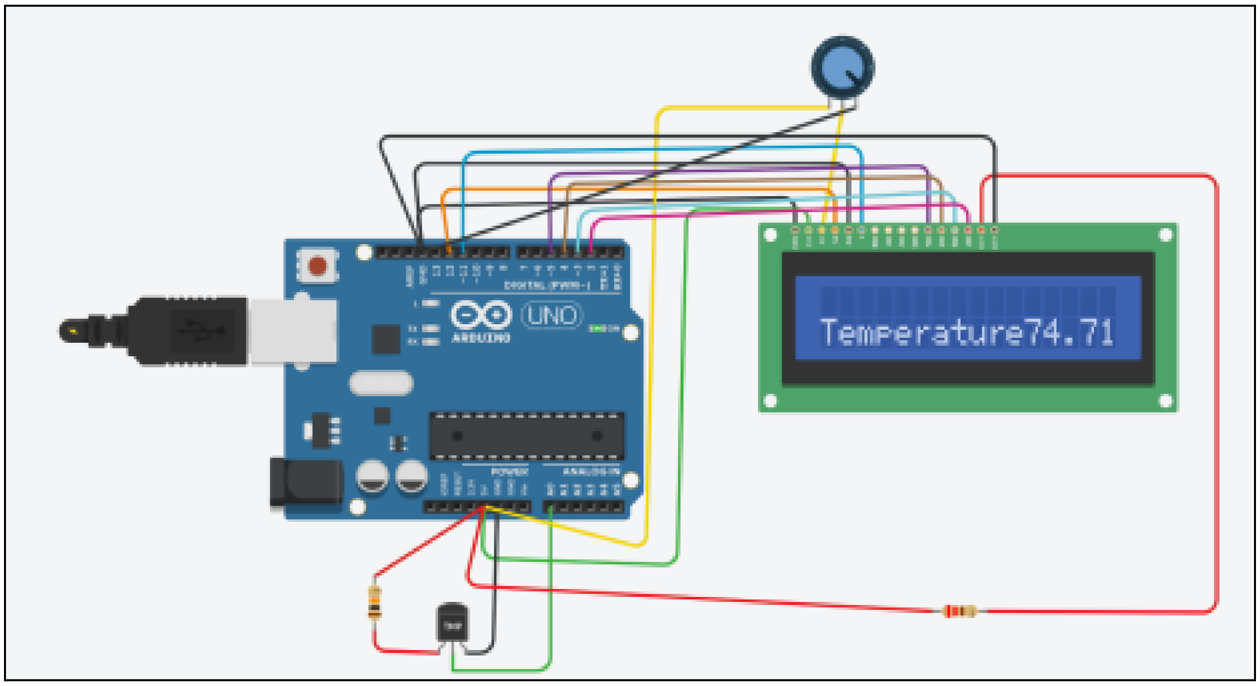
```
  delay(100);
  lcd.clear();
  //For DHT11
  val = analogRead(tempin);
  float mv = ((val/1024.0)*5000);
  float cell = mv/10;
  float fh = cell*(9/5)+32;
  lcd.print("Temperature:");
  lcd.print(cell);
  Serial.println("Temperature");
  Serial.println(cell);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
  lcd.clear();
}
```

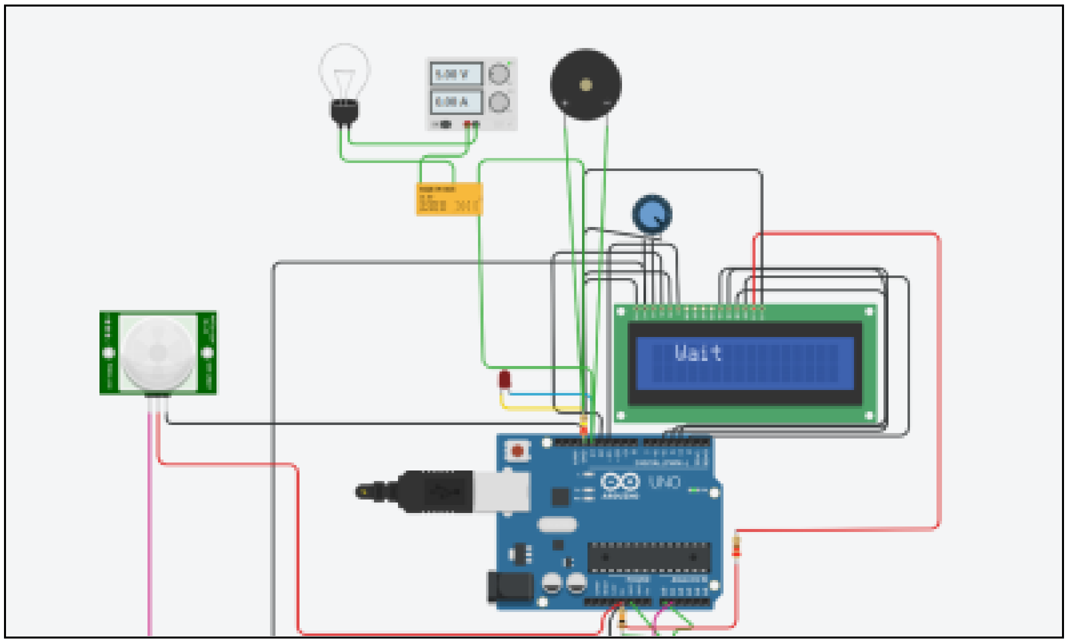
Circuit Diagram:

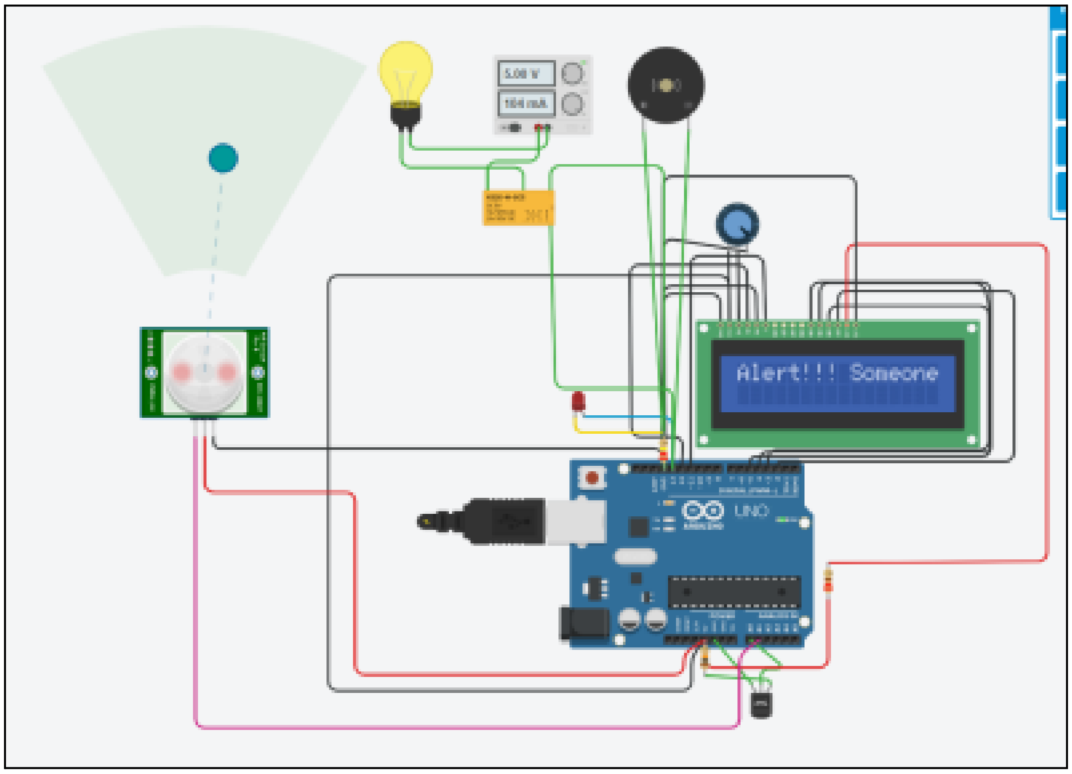


Output:

For DHT11:







Conclusion:

Hence, we have explored programs based on interfacing DHT11 temperature sensor Programs based on interfacing Passive infrared sensors (PIR), Ultrasonic of Arduino.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 6**	
Title of Lab Assignment: IOT with Cloud: Interfacing IOT device with cloud.		
DOP: 13-02-2024	**DOS: 20-02-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

Practical No. 6

Aim: IOT with Cloud : Interfacing IOT device with cloud.

Theory:

Internet-of-Things can benefit from the scalability, performance and pay-as you-go nature of cloud computing infrastructures. Indeed, as IoT applications produce large volumes of data and comprise multiple computational components (e.g., data processing and analytics algorithms), their integration with cloud computing infrastructures could provide them with opportunities for cost-effective on-demand scaling.

An IoT cloud is a massive network that supports IoT devices and applications. This includes the underlying infrastructure, servers and storage, needed for real-time operations and processing.

IoT/cloud infrastructures and related services can be classified to the following models:

1. **Infrastructure-as-a-Service (IaaS) IoT/Clouds:**
   These services provide the means for accessing sensors and actuators in the cloud. The associated business model involves the IoT/Cloud provider to act either as data or sensor provider.

2. **Platform-as-a-Service (PaaS) IoT/Clouds:**
   This is the most widespread model for IoT/cloud services, given that it is the model provided by all publicIoT/cloud infrastructures outlined above.

3. **Software-as-a-Service (SaaS) IoT/Clouds:**
   SaaS IoT services are the ones enabling their uses to access complete IoT-based software applications through the cloud, on- demand and in a pay-as-you-go fashion. As soon as sensors and IoT devices are not visible, SaaS IoT applications resemble very much conventional cloud-based SaaS applications.

ThingsSpeak:

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.

Code:

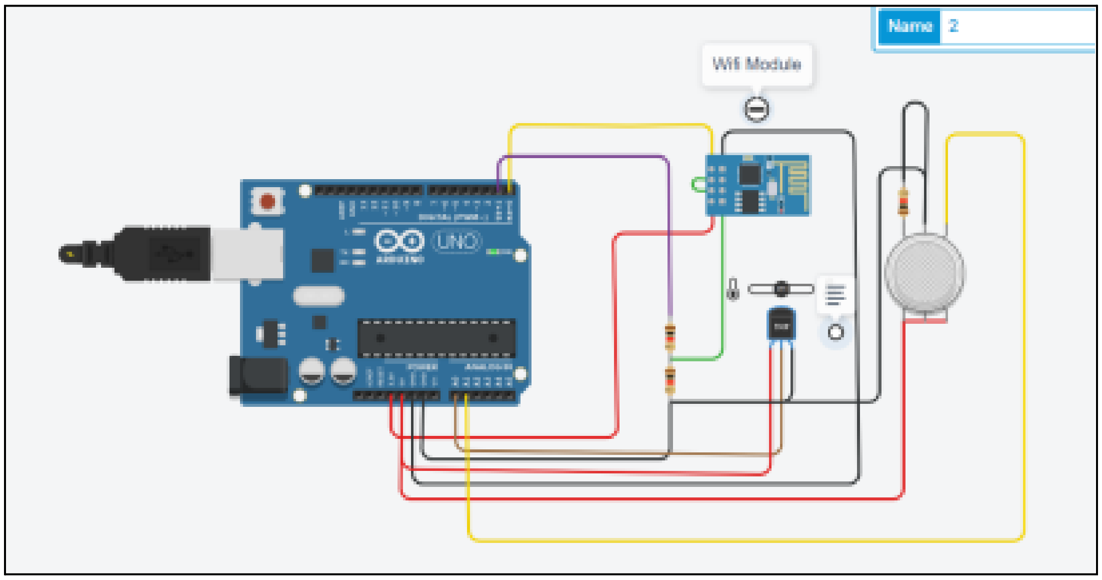
```
float val, voltage, temp;
String ssid = "Simulator Wifi"; // SSID to connect to
String password = ""; //virtual wifi has no password
String host = "api.thingspeak.com"; // Open Weather Map API
const int httpPort = 80;
String url = "/update?api_key=ZXP27Y8OGRHR7ISL&field1="; //Replace
XXXXXXXXXXXXXXXX by your ThingSpeak Channel API Key

void setupESP8266(void) {
  // Start our ESP8266 Serial Communication
  Serial.begin(115200); // Serial connection over USB to computer
   Serial.println("AT"); // Serial connection on Tx / Rx port to ESP8266 delay(10); // Wait a little
for the ESP to respond
  if (Serial.find("OK"))
    Serial.println("ESP8266 OK!!!");
    // Connect to Simulator Wifi
    Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
    delay(10); // Wait a little for the ESP to respond
  if (Serial.find("OK"))
    Serial.println("Connected to WiFi!!!");
```

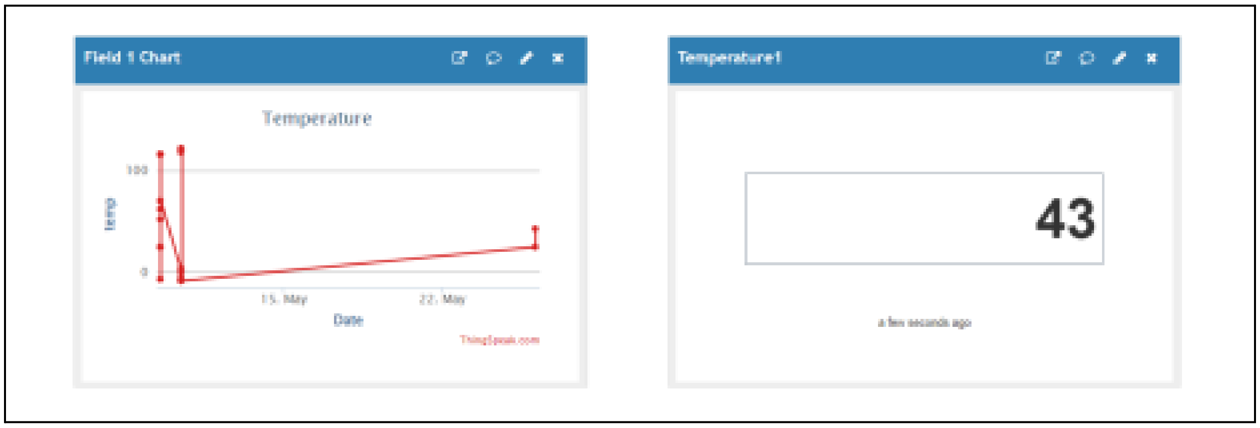
```
        // Open TCP connection to the host:
        //ESP8266 connects to the server as a TCP client.
          Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\",\" + httpPort); delay(50); // Wait a little
for the ESP to respond
    if (Serial.find("OK"))
        Serial.println("ESP8266 Connected to server!!!") ;
}

void anydata(void) {
    //val = analogRead(A1);
    val=analogRead(A0);
    voltage=val*0.0048828125;
    temp = (voltage - 0.5) * 100.0;
    // Construct our HTTP call
    String httpPacket = "GET " + url + String(temp) + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
    int length = httpPacket.length();
    // Send our message length
    Serial.print("AT+CIPSEND=");
    Serial.println(length);
    delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return -1;
    // Send our http request
    Serial.print(httpPacket);
    delay(10); // Wait a little for the ESP to respond
    if (Serial.find("SEND OK\r\n"))
        Serial.println("ESP8266 sends data to the server");
    }

void setup() {
    pinMode(A0, INPUT);
    //pinMode(A1,INPUT);
    setupESP8266();
}
```

```
void loop() {
  anydata();
  delay(4000); // delay changed for faster analytics
}
```

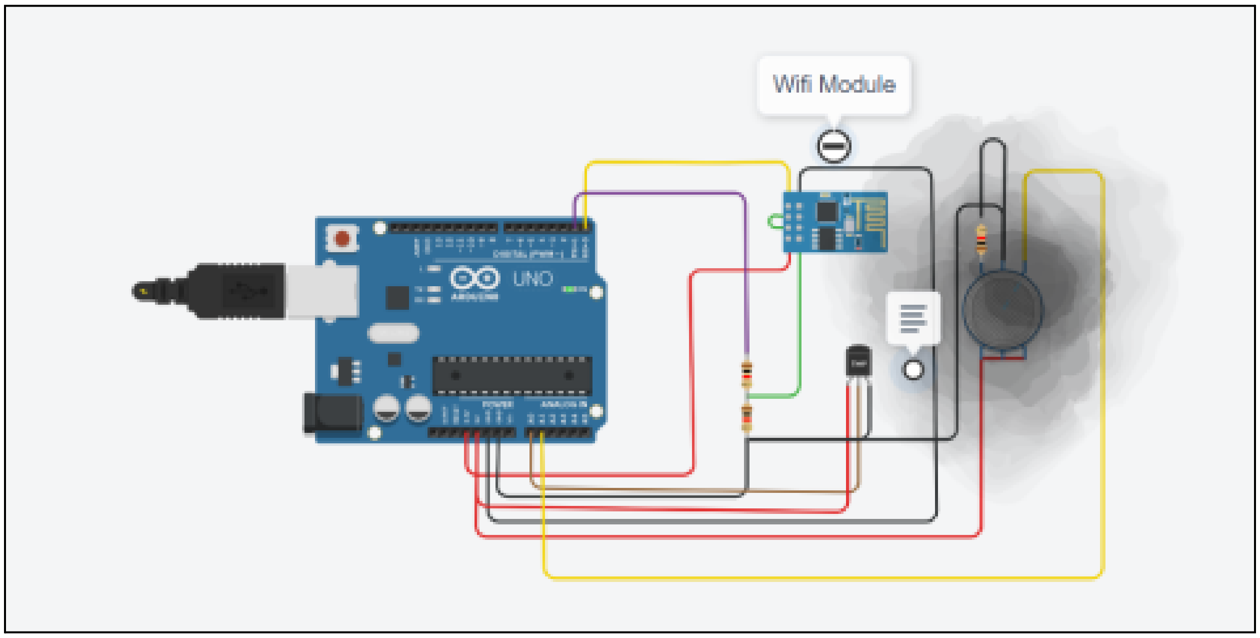
Output:

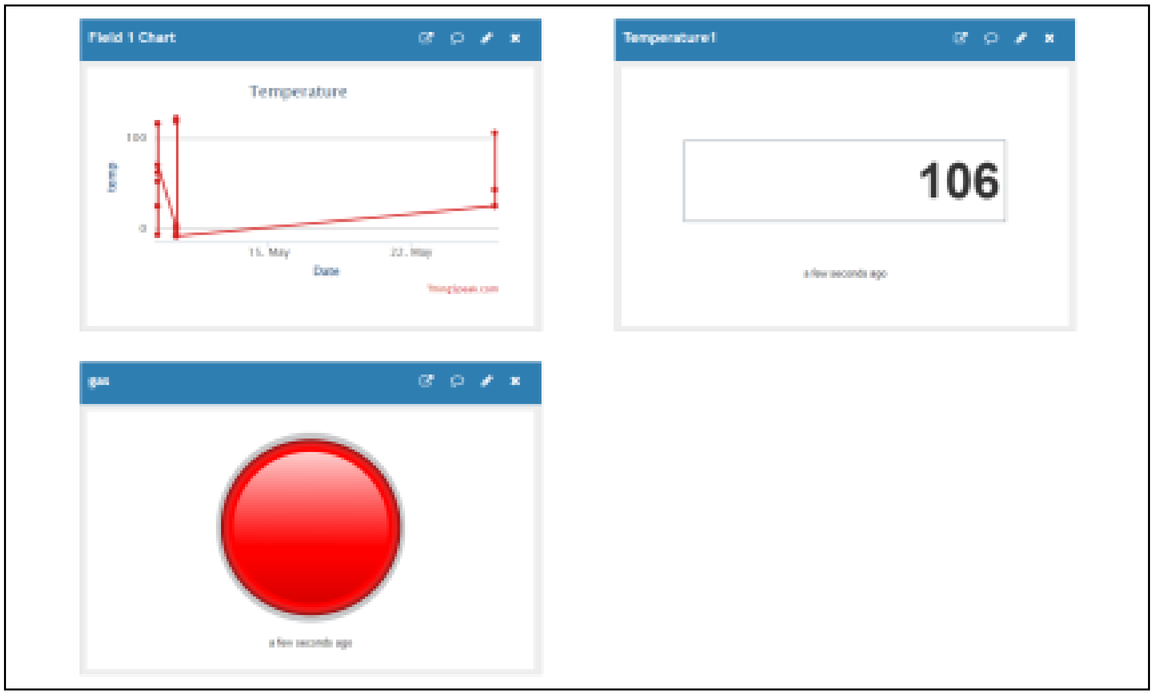
For Temperature Sensor:





For Gas Sensor:





Conclusion:

Hence, we have successfully implemented IoT devices with Cloud.

Name of Student: Pushkar Sane			
Roll Number: 45		**Lab Assignment Number: 7**	
Title of Lab Assignment: Programs Based on Interfacing LDR with Arduino			
DOP: 20-02-2024		**DOS: 27-02-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**		**Signature:**

Practical No. 7

Aim: Programs Based on Interfacing LDR with Arduino.

Theory:

LDR (Light Dependent Resistor):

An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits.

Light Dependent Resistors (LDR) are also called photoresistors.

They are made of high resistance semiconductor material. When light hits the device, the photons give electrons energy. This makes them jump into the conductive band and thereby conduct electricity.

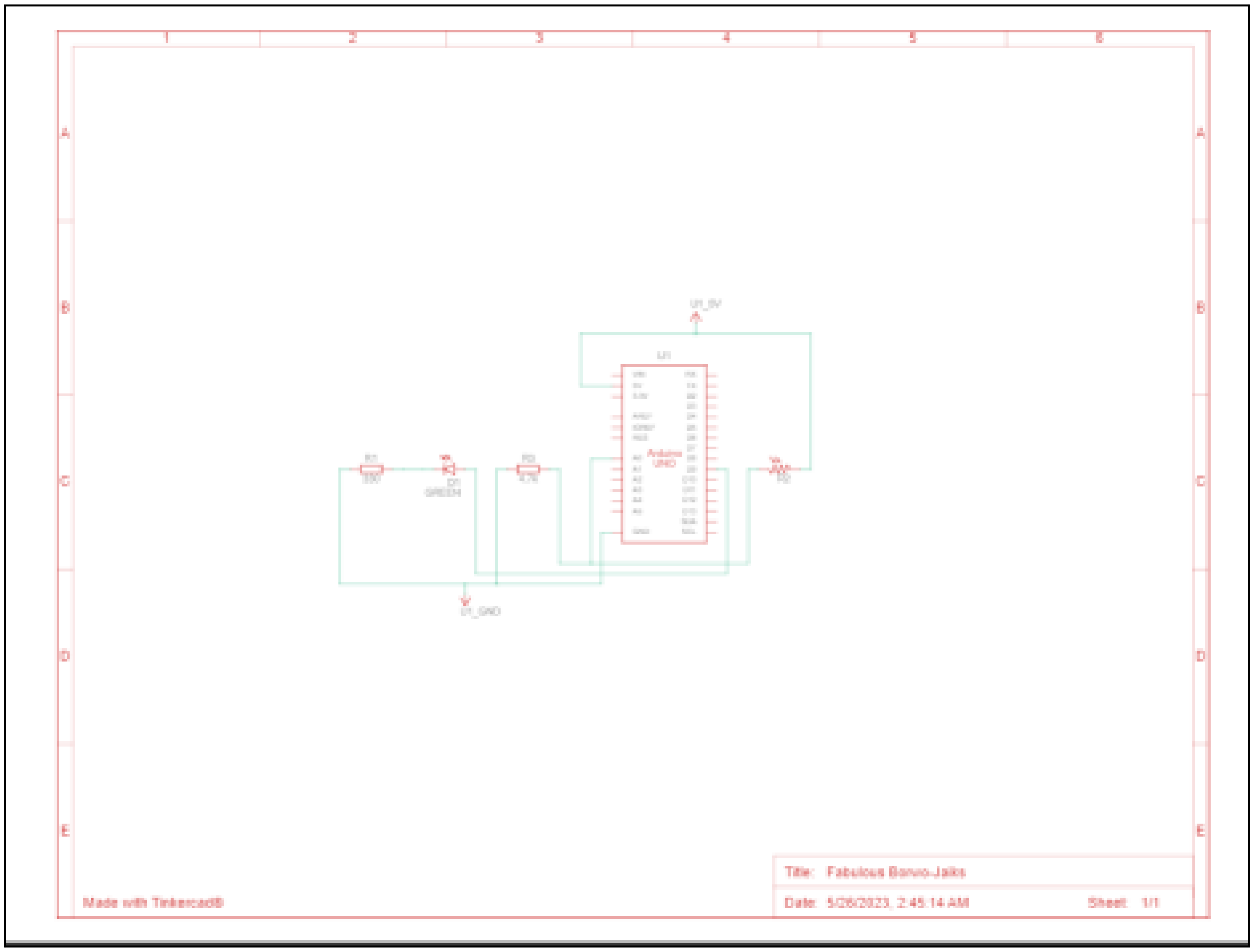
Light dependent resistors, LDRs or photoresistors are often used in circuits where it is necessary to detect the presence or the level of light.

They can be described by a variety of names from light dependent resistor, LDR, photoresistor, or even photocell, photocell or photoconductor.

A Light Sensor is something that a robot can use to detect the current ambient light level - i.e. how bright/dark it is.

There are a range of different types of light sensors, including 'Photoresistors', 'Photodiodes', and 'Phototransistors'.

Circuit Diagram:

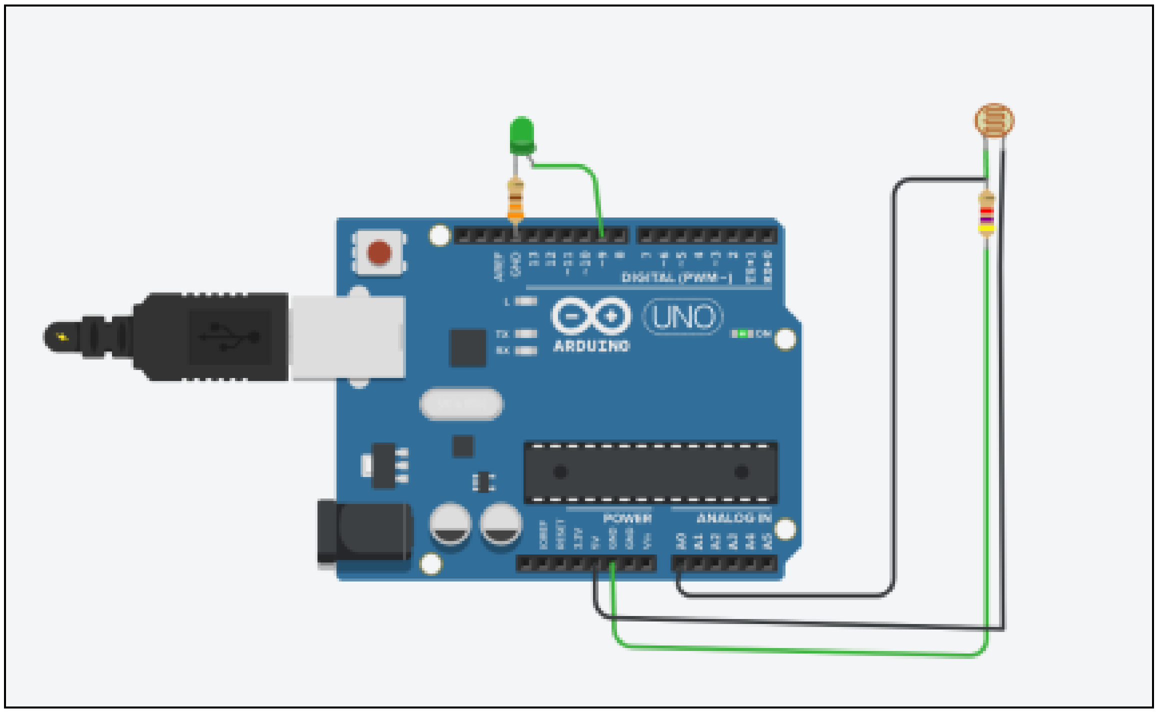


Source Code:

```
// C++ code
int photosensor = 0;
void setup() {
   pinMode(A0,INPUT);
   Serial.begin(9600);
   pinMode(9,OUTPUT);
}
void loop() {
   photosensor = analogRead(A0);
   Serial.println(photosensor);
   analogWrite(9,map(photosensor,0,1023,0,255));
```

```
    delay(20);
}
```

Output:



Conclusion: From this practical, I have learned and implemented a photoresistor with an arduino.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 8**	
Title of Lab Assignment: Programs based on interfacing & segment as a counter with Arduino.		
DOP: 05-03-2024	**DOS: 12-03-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

Practical No. 8

Aim: Programs based on interfacing & segment as a counter with Arduino.

Theory:

Seven-Segment Display

A seven-segment display is a form of electronic display device for displaying decimal numerals that are an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The 7-segment display, also written as “seven segment display”, consists of seven LEDs (hence its name) arranged in a rectangular fashion. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point, (DP) when two or more 7-segment displays are connected together to display numbers greater than ten.

LED based 7-segment displays are very popular amongst Electronics hobbyists as They are easy to use and easy to understand. In most practical applications, 7- segment displays are driven by a suitable decoder/driver IC such as the CMOS 4511 or TTL 7447 from a 4-bit BCD input. Today, LED based 7-segment displays have been largely replaced by liquid crystal displays (LCDs) which consume less current.

Seven-segment LED circuit configuration

LED display devices have two kinds of circuit: common anode and common cathode.

- Common anode: when the common pin is positive.
- Common cathode: when the common pin is negative.

Segment displays are available in various colors (Red, Blue, and Green) and sizes (0.56 to 6.5 inches). Sometimes two to four 7-segment displays are packed together to form a big display. Few of the 7-segment displays have 8 LEDs. It is in the form of an additional circular LED on board.The circular LED indicates decimal point in numeral.

Applications of Seven Segment Displays:

Common applications of seven segment displays are in:

- Digital clocks
- Clock radios
- Radio frequency indicators
- Wrist Watches
- Speedometers
- Motor-vehicle odometers
- Calculators

Code:

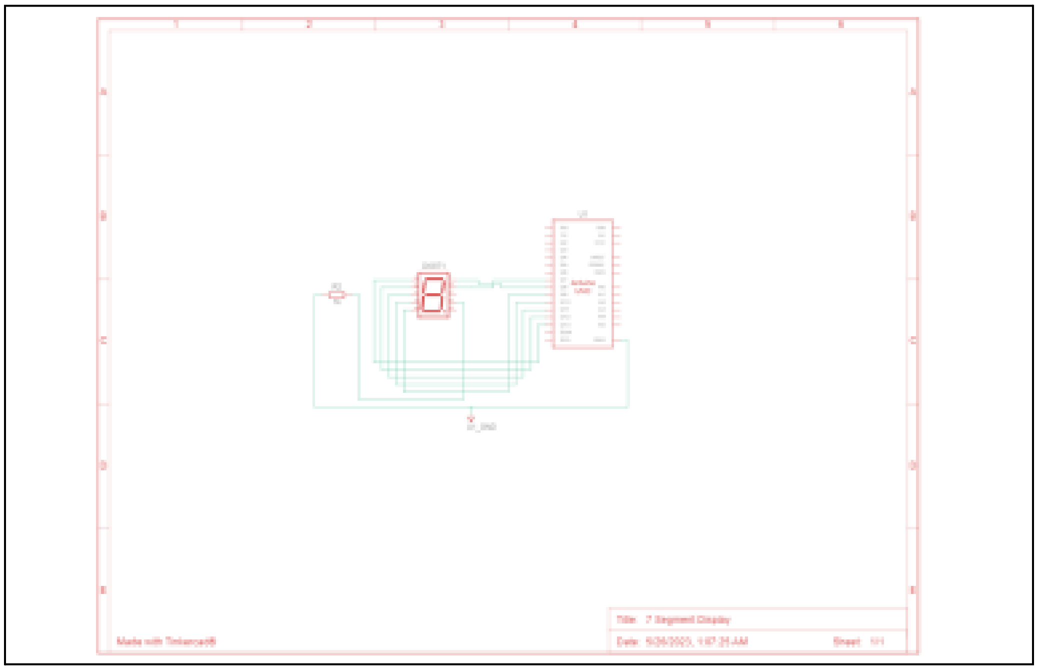
```
// C++ code
//
const int A= 13;
const int B = 12;
const int C = 11;
const int D = 10;
const int E = 9;
const int F = 8;
const int G = 7;
void setup() {
   pinMode(A, OUTPUT);
   pinMode(B, OUTPUT);
   pinMode(C, OUTPUT);
   pinMode(D, OUTPUT);
   pinMode(E, OUTPUT);
   pinMode(F, OUTPUT);
   pinMode(G, OUTPUT);
}

void zero (void){
   digitalWrite(A,HIGH);
   digitalWrite(B,HIGH);
```

```
    digitalWrite(C,HIGH);
    digitalWrite(D,HIGH);
    digitalWrite(E,HIGH);
    digitalWrite(F,HIGH);
    digitalWrite(G,LOW);
}

voidone (void){
    digitalWrite(A,LOW);
    digitalWrite(B,HIGH);
    digitalWrite(C,HIGH);
    digitalWrite(D,LOW);
    digitalWrite(E,LOW);
    digitalWrite(F,LOW);
    digitalWrite(G,LOW);
}

voidtwo (void){
    digitalWrite(A,HIGH);
    digitalWrite(B,HIGH);
    digitalWrite(C,LOW);
    digitalWrite(D,HIGH);
    digitalWrite(E,HIGH);
    digitalWrite(F,LOW);
    digitalWrite(G,HIGH);
}

voidthree (void){
    digitalWrite(A,HIGH);
    digitalWrite(B,HIGH);
    digitalWrite(C,HIGH);
    digitalWrite(D,HIGH);
    digitalWrite(E,LOW);
```

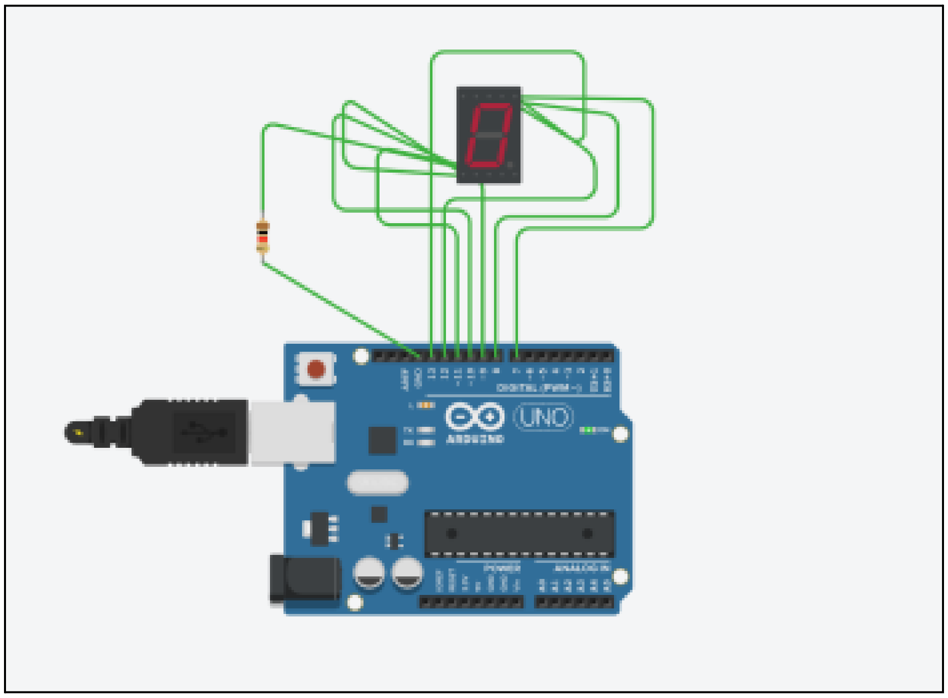
```
  digitalWrite(F,LOW);
  digitalWrite(G,HIGH);
}

voidFour (void){
  digitalWrite(A,LOW);
  digitalWrite(B,HIGH);
  digitalWrite(C,HIGH);
  digitalWrite(D,LOW);
  digitalWrite(E,LOW);
  digitalWrite(F,HIGH);
  digitalWrite(G,HIGH);
}

voidnine (void){
  digitalWrite(A,HIGH);
  digitalWrite(B,HIGH);
  digitalWrite(C,HIGH);
  digitalWrite(D,HIGH);
  digitalWrite(E,LOW);
  digitalWrite(F,HIGH);
  digitalWrite(G,HIGH);
}

voidloop() {
  zero();
  delay(500);
  one();
  delay(500);
  two();
  delay(500);
  three();
  delay(500);
```

```
    Four();
    delay(500);
    nine();
    delay(500);//Waitfor1000millisecond(s)
}
```

Circuit Diagram:



Output:



Conclusion: Hence, we have successfully implemented programs based on interfacing & segment as a counter with Arduino.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: Mini Project**	
Title of Lab Assignment: Mini Project (Arduino Based Plant monitoring system.)		
DOP: 05-03-2024	**DOS: 12-03-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO1, PO2, PO5, PO7, PSO1**	**Signature:**

IOT Mini-Project

Topic: Arduino Based Plant monitoring system.

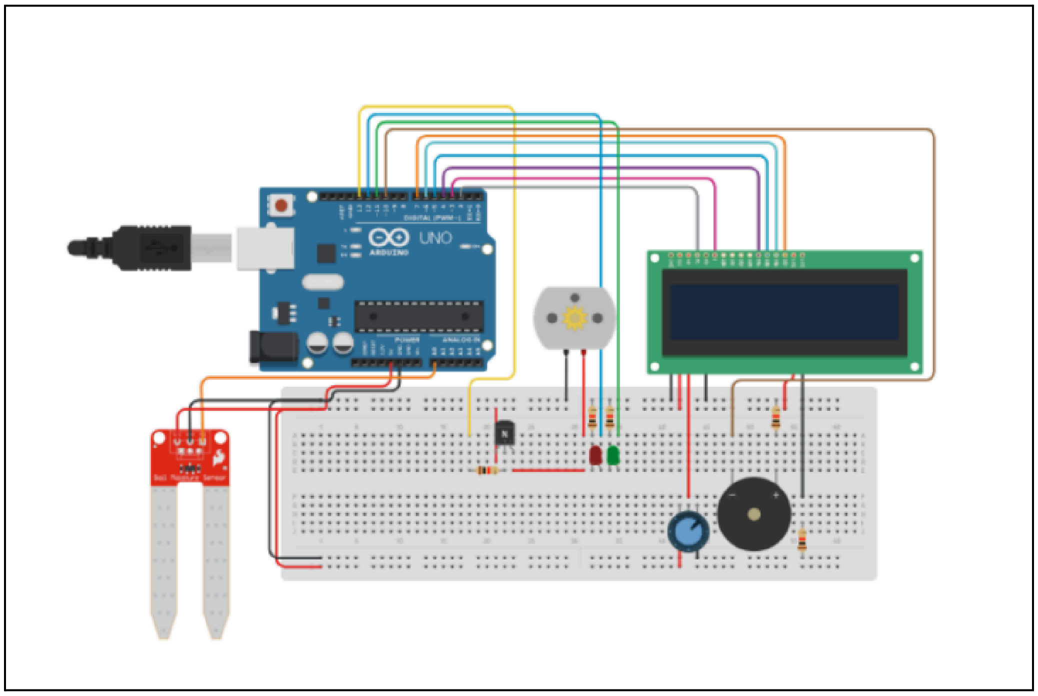
Theory:

An Arduino-based plant monitoring system is a project aimed at providing real-time data about the environment and conditions surrounding a plant. By using sensors connected to an Arduino board, this system can measure parameters such as soil moisture, temperature, light intensity, and humidity. This data can then be used to monitor the health of plants and optimize their growth conditions, helping gardeners and farmers to ensure that their plants are thriving.

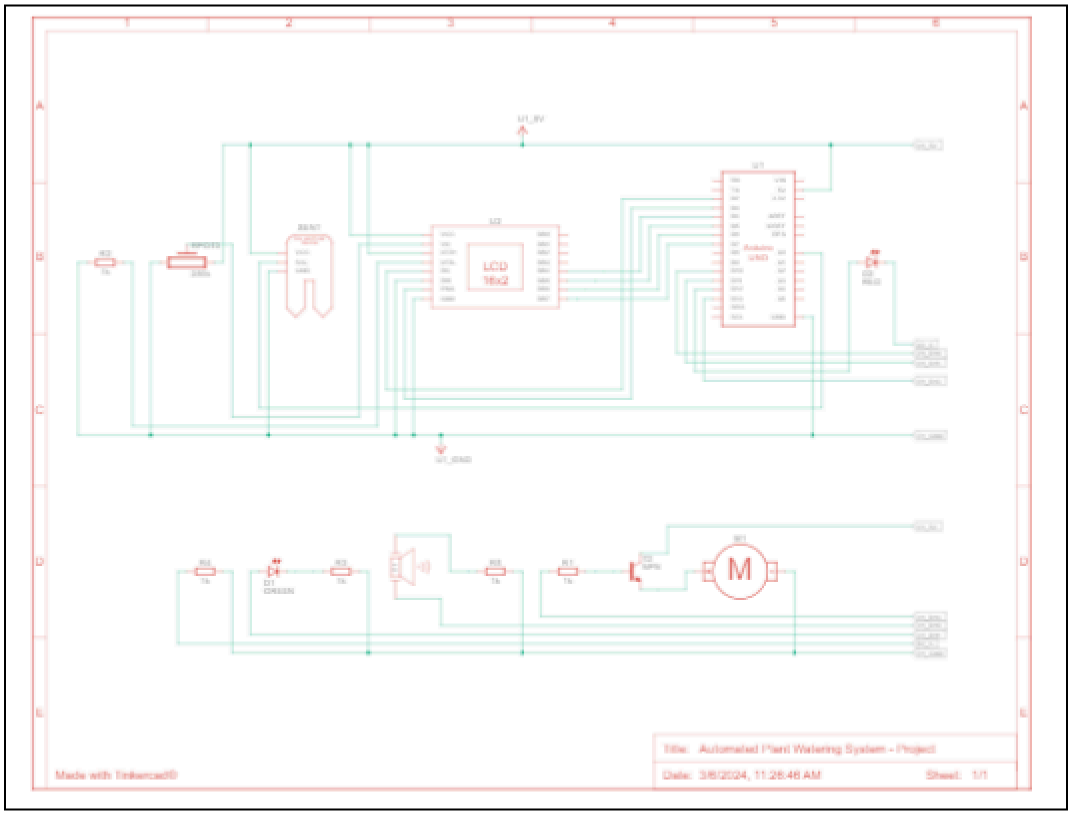
Project Objective:

- Monitor Soil Moisture: Measure the moisture level in the soil to ensure plants are adequately watered.
- Measure Temperature and Humidity: Monitor the environmental conditions to ensure they are within the optimal range for plant growth.
- Monitor Light Intensity: Measure the amount of light available to the plants to ensure they are receiving sufficient light for photosynthesis.
- Real-time Data Display: Display the collected data in real-time, allowing users to easily monitor the conditions.
- Alert System: Implement an alert system to notify users when certain conditions (e.g., soil too dry) are met, so they can take action.
- Data Logging: Log the collected data over time to analyze trends and optimize plant care strategies.
- Remote Monitoring: Enable users to monitor the plant conditions remotely, using a web or mobile interface.
- Automation: Implement automation features to control watering systems or adjust light sources based on sensor readings.
- Energy Efficiency: Design the system to be energy-efficient, using low-power components and sleep modes when not in use.
- Expandability: Design the system with expandability in mind, allowing users to add more sensors or features in the future.

Circuit Diagram:



Schematic Diagram:


U1_5V
U1_GND
LCD 16x2
Arduino Uno
M
Title: Automated Plant Watering System – Project
Date: 3/6/2024, 11:28:48 AM
Sheet: 1/1
Made with Tinkercad®

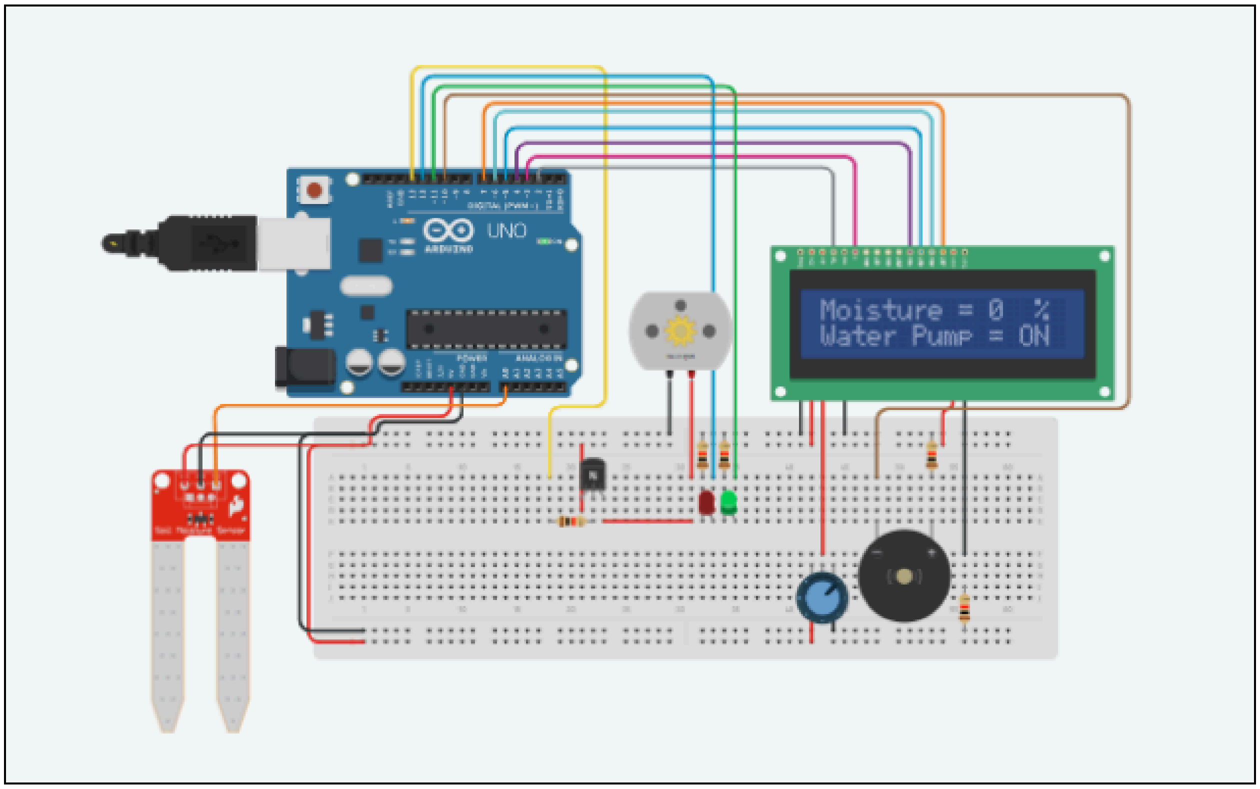
Code:

```
#include <LiquidCrystal.h>
// Include the LiquidCrystal library, which provides functions to control the LCD display.
// Declaring variables
const int BuzzerPin = 10; // Pin connected to the piezo buzzer
const int LedRed = 12; // Pin connected to the red LED
const int LedGreen = 11; // Pin connected to the green LED
const int SoilMoistureSensor = A0; // Pin connected to the soil moisture sensor
const int WaterPump = 13; // Pin connected to the water pump relay
LiquidCrystal lcd(2, 3, 4, 5, 6, 7); // Create an instance of the LiquidCrystal class to control the
LCD
void setup() {
  pinMode(WaterPump, OUTPUT); // Set the water pump pin as output
  pinMode(LedRed, OUTPUT); // Set the red LED pin as output
  pinMode(LedGreen, OUTPUT); // Set the green LED pin as output
  pinMode(BuzzerPin, OUTPUT); // Set the piezo buzzer pin as output
  Serial.begin(9600); // Initialize serial communication at 9600 bps
  lcd.begin(16, 2); // Initialize the LCD with 16 columns and 2 rows
  pinMode(BuzzerPin, OUTPUT); // Set the piezo pin as output
  lcd.clear(); // Clear the LCD display
  lcd.setCursor(0, 0); // Set the cursor to the first column and first row
  String message1 = "Automated Plant";
  String message2 = "Watering System";
   // Display "Automated Plant" on the first row of the LCD with a delay of 100ms between
characters
  for (int i = 0; i < message1.length(); i++) {
    lcd.print(message1.charAt(i));
    delay(100);
  }
  lcd.setCursor(0, 1); // Set the cursor to the first column and second row
    // Display "Watering System" on the second row of the LCD with a delay of 100ms between
characters
    for (int i = 0; i < message2.length(); i++) {
```

```
    lcd.print(message2.charAt(i));
    delay(100);
  }
  delay(2500); // Delay for 2.5 seconds to display the messages
  lcd.clear(); // Clear the LCD display again
  lcd.setCursor(0, 0); // Set the cursor to the first column and first row
  lcd.print("Moisture ="); // Display "Moisture =" on the first row
  lcd.setCursor(0, 1); // Set the cursor to the first column and second row
  lcd.print("Water Pump ="); // Display "Water Pump =" on the second row
}

void loop(){
    int Sensor = analogRead(SoilMoistureSensor); // Read the value from the soil moisture
sensor
   int mappedValue = map(Sensor, 0, 876, 0, 99); // Map the sensor value to a range from 0 to
99
  lcd.setCursor(11, 0); // Set the cursor to the 12th column and first row
  lcd.print(mappedValue); // Display the mapped value (moisture percentage) on the first row
  lcd.setCursor(14, 0); // Set the cursor to the 15th column and first row
  lcd.print("%"); // Display the percentage symbol on the first row
  lcd.setCursor(13, 1); // Set the cursor to the 14th column and second row
  // Control the water pump and LEDs based on the moisture percentage
  if (mappedValue < 50) {
    digitalWrite(WaterPump, HIGH); // Turn on the water pump
    digitalWrite(LedGreen, HIGH); // Turn on the green LED
    digitalWrite(LedRed, LOW); // Turn off the red LED
    lcd.print("ON "); // Display "ON" on the second row
    playSound(); // Play a sound using the piezo buzzer
  }
  else {
    digitalWrite(WaterPump, LOW); // Turn off the water pump
    digitalWrite(LedGreen, LOW); // Turn off the green LED
    digitalWrite(LedRed, HIGH); // Turn on the red LED
```

```
        lcd.print("OFF"); // Display "OFF" on the second row
    }
}
void playSound() {
    tone(BuzzerPin, 87, 100); // Play a tone of 87 Hz for 100 milliseconds on the piezo buzzer
    delay(1000); // Delay for 1 second
}
```

Output:



Conclusion:

Successfully implemented the IOT Mini Project on Arduino Based Plant monitoring system.