Practical Robotics Projects with Arduino

(CSE 4571)

Lab Assignment No – 03

Light & Sound Show

Submission Date: _____

Branch: CSE		**Section:19**
Name	**Registration No.**	**Signature**
Subrat Samal	2241019269	

Department of Computer Science and Engineering
Institute of Technical Education and Research (Faculty of Engineering)
Siksha 'O' Anusandhan (Deemed to be University)
Bhubaneswar, Odisha-751030.

Aim:

Light & Sound Show: To design and implement an intelligent lighting system using Arduino Uno that integrates analog voltage control, potentiometers, photoresistors, RGB LED color mixing, and audio synchronization, thereby enabling adaptive brightness, color variation, and sound response based on user input and ambient light conditions.

Objectives:

1) **Develop a comprehensive understanding of analog voltage and potentiometers.**

1.1 Read analog voltage using the "analogRead" command and convert the analog voltage to a voltage reading in volts

- ✔ Generate an analog voltage by building a voltage divider circuit with two resistors (330 ohm and 100 ohm) and connect the circuit to the 5V supply pin of the Arduino.
- ✔ Acquire knowledge of analog voltage reading by implementing the "analogRead" command on the Arduino platform.
- ✔ Utilize the "analogRead" command to measure and display the analog voltage readings on the Arduino Serial Monitor.

1.2 Read and interpret the voltage signals produced by a potentiometer.

- ✔ Demonstrate knowledge of the function and operation of potentiometers in electronic circuits.
- ✔ Write an Arduino sketch to read the analog value from the potentiometer connected to pin A0 and displays the value on the Serial Monitor.

1.3 Read the analog value from the potentiometer and converts it to a voltage reading in volts using Arduino and serial monitor

- ✔ Read the analog value from the potentiometer and converts it to a voltage reading in volts using the formula V = (ADC value * Vref) / 1023, where Vref is the reference voltage (5V in this case) and 1023 is the maximum value of the ADC. The voltage reading is then displayed on the Serial Monitor.

1.4 Controlling LED Brightness with Potentiometer and Serial Monitoring using Arduino

1.4.1 Controlling LED brightness with potentiometer without using map() function.

- Read the analog value from the potentiometer, calculates the proportional

brightness value, sets the brightness of the LED, and displays the values on the serial monitor.

1.4.2 Controlling LED brightness with potentiometer using map() function.

- Read the analog value from a potentiometer connected to pin A0 of an Arduino board and maps this value to a range of LED brightness from 0 to 255. The output of the LED is controlled by pin 9. The serial monitor is used to display the potentiometer value and the corresponding LED brightness.

1.5 Optimizing Electronic Control with Potentiometers: A Practical Approach to Controlling Speaker Volume and LED Brightness with Arduino

- ✔ Utilize the serial monitor in the Arduino sketches to accurately monitor the changes in analog values and ensure reliable control while adjusting the volume of a speaker and the brightness of an LED using potentiometer controls. The circuit connection for this sketch involves connecting a potentiometer for controlling speaker volume and another potentiometer for controlling LED brightness.

2) Understand the function and operation of an LDR/Photoresistor in electronic circuits and its ability to measure changes in light intensity.

2.1 Comprehend the relationship between light intensity and the resistance of photoresistors and how they can be used to measure changes in light intensity.

- ✔ Develop a comprehensive understanding of the photoresistor and its ability to measure the brightness of the room, where the resistance of the photoresistor is inversely proportional to the brightness of the light.
- ✔ Hook up a photoresistor in series with a fixed resistor to obtain a measurable change in voltage across the series resistor.
- ✔ Develop proficiency in programming the Arduino to read and interpret the analog signals from the photoresistor through the serial monitor.

2.2 Demonstrate knowledge of the practical application of photoresistors in electronic circuits by implementing a project involving street light control using a red and green LED.

- ✔ Read the voltage signal via an analog pin on the Arduino and use the information to control electronic components such as LEDs.
- ✔ Implement a practical application of a photoresistor by programming an Arduino to control a red and green LED based on the brightness of light, where the green LED is turned on when the light is on and the red LED is turned on when the light is off.
- ✔ Test and troubleshoot the photoresistor circuit and Arduino sketch to ensure accurate and reliable operation.

3) Create an audible signal that changes tone based on the brightness of the light in the room by utilizing a photoresistor to measure light and a passive buzzer to produce the audible signal.

3.1 Effectively implement the photoresistor and passive buzzer in an electronic circuit, ensuring accurate and reliable measurement of the light and creation of the audible signal.

- ✔ Use the serial monitor to monitor the changes in the analog values of the photoresistor and ensure that the audible signal is proportional to the brightness of the room.

3.2 Implementation of a Photoresistor-Based Audible Signal with control Button

- ✔ Using a button to start/stop the tone generation loop in an Arduino sketch. To start/stop the sound, you can add a conditional statement that checks for a certain condition, such as a button press, and sets a variable to start/stop the loop that generates the tone.

4) Exploring RGB Mixing with Arduino: Connecting, Programming, and Controlling Primary and Intermediate Colors using PWM Signals

4.1 Develop the ability to experiment with RGB mixing and create other colors by varying the levels of Red, Green, and Blue channels.

- ✔ Learn the fundamental concepts of RGB LEDs and how they work to produce different colors.
- ✔ Understand the differences between common anode and common cathode RGB LEDs and learn how to connect a common cathode RGB LED to an Arduino.
- ✔ Program an Arduino to get the primary colors by mixing different combinations of Red, Green, and Blue channels using PWM signals.

4.2 Achieve the ability to connect an RGB LED and program it to display primary and intermediate colors based on user input.

- ✔ Develop a program that allows user for input to control the primary and intermediate colors displayed by the RGB LED.
- ✔ Create examples of primary colors, such as Cyan, Magenta, and Yellow, and explore the possibilities of mixing in-between colors.
- ✔ Test the program and RGB LED thoroughly to ensure accurate and reliable color mixing.

5) Implementation of an Arduino Ambient Light Mood Lamp with Start/Stop Button Control (Create a mood lamp that changes color and tone based on the ambient light level in the room with Start/Stop Button Control)

- ✔ Write an Arduino sketch that reads the analog value from the LDR, uses this value to adjust the color and tone of the RGB LED and the pitch of the buzzer, and displays the light level and other relevant data on the serial monitor. The start/stop button allows the user to control the sound and light of the lamp. When the button is pressed, the flags for stopSound and stopLight are set to true, and the sound and light are stopped.

Pre-Lab Questionnaire:

1) What is the purpose of the "analogRead" command in Arduino?
2) How can you convert analog voltage to voltage reading in volts using Arduino?
3) Can you use the "analogRead" command to measure digital signals?
4) What is the maximum analog voltage that can be read by the Arduino?
5) What is the range of values that the "analogRead" command can output on the Arduino platform?
6) What is the function of the map() function in Arduino?
7) What is the relationship between light intensity and the resistance of a photoresistor?
8) How can a photoresistor be used to measure changes in light intensity?
9) What is the resistance range of a typical photoresistor in bright light conditions?

Components/Equipment Required:

Sl. No.	Name of the Component / Equipment	Specification	Quantity
1)	Arduino UNO R3	16MHz	1
2)	Arduino UNO cable	USB Type A to B	1
3)	Trimmer Potentiometer	10k, Preset	2
4)	LDR/Photoresistor	5mm	1
5)	RGB LED (4 pin)	5mm, Common Cathode	1
6)	Resistors (carbon type)	¼ watt (100Ω)	1
		¼ watt (330Ω)	3
		¼ watt (10kΩ)	1
7)	LED	Any two different colour of your choice	2
8)	Buzzer	5V, small	1
9)	Breadboard	840 Tie points	1
10)	Digital Multimeter	----------	1
11)	Jumper Wire	----------	As per requirement

Objective 1

Develop a comprehensive understanding of analog voltage and potentiometers.

1.1 Read analog voltage using the "analogRead" command and convert the analog voltage to a voltage reading in volts

- ✔ Generate an analog voltage by building a voltage divider circuit with two resistors (330 ohm and 100 ohm) and connect the circuit to the 5V supply pin of the Arduino.
- ✔ Acquire knowledge of analog voltage reading by implementing the "analogRead" command on the Arduino platform.
- ✔ Utilize the "analogRead" command to measure and display the analog voltage readings on the Arduino Serial Monitor.

1.2 Read and interpret the voltage signals produced by a potentiometer.

- ✔ Demonstrate knowledge of the function and operation of potentiometers in electronic circuits.
- ✔ Write an Arduino sketch to read the analog value from the potentiometer connected to pin A0 and displays the value on the Serial Monitor.

1.3 Read the analog value from the potentiometer and converts it to a voltage reading

in volts using Arduino and serial monitor

- ✔ Read the analog value from the potentiometer and converts it to a voltage reading in volts using the formula V = (ADC value * Vref) / 1023, where Vref is the reference voltage (5V in this case) and 1023 is the maximum value of the ADC. The voltage reading is then displayed on the Serial Monitor.

1.4 Controlling LED Brightness with Potentiometer and Serial Monitoring using Arduino

1.4.1 Controlling LED brightness with potentiometer without using map() function.

- Read the analog value from the potentiometer, calculates the proportional brightness value, sets the brightness of the LED, and displays the values on the serial monitor.

1.4.2 Controlling LED brightness with potentiometer using map() function.

- Read the analog value from a potentiometer connected to pin A0 of an Arduino board and maps this value to a range of LED brightness from 0 to 255. The output of the LED is controlled by pin 9. The serial monitor is used to display the potentiometer value and the corresponding LED brightness.

1.5 Optimizing Electronic Control with Potentiometers: A Practical Approach to Controlling Speaker Volume and LED Brightness with Arduino

- ✔ Utilize the serial monitor in the Arduino sketches to accurately monitor the changes in analog values and ensure reliable control while adjusting the volume of a speaker and the brightness of an LED using potentiometer controls. The circuit connection for this sketch involves connecting a potentiometer for controlling speaker volume and another potentiometer for controlling LED brightness.

Circuit / Schematic Diagram

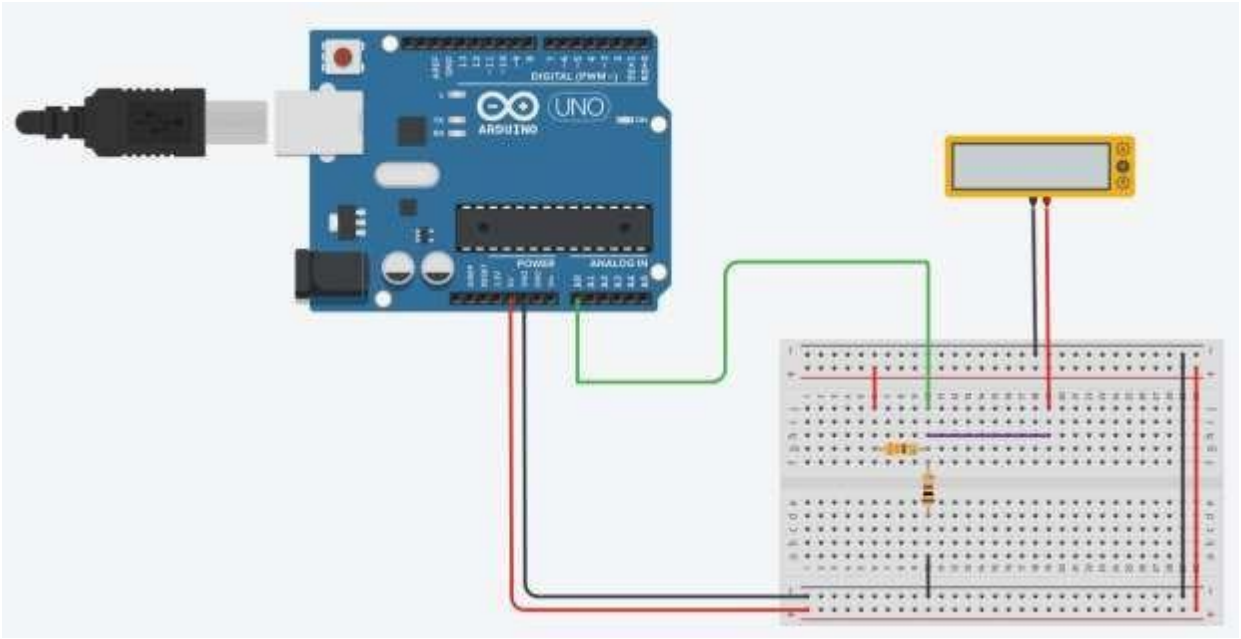

Figure 1.1: Schematic of reading analog voltage using the "analogRead" command and convert the analog voltage to a voltage reading in volts

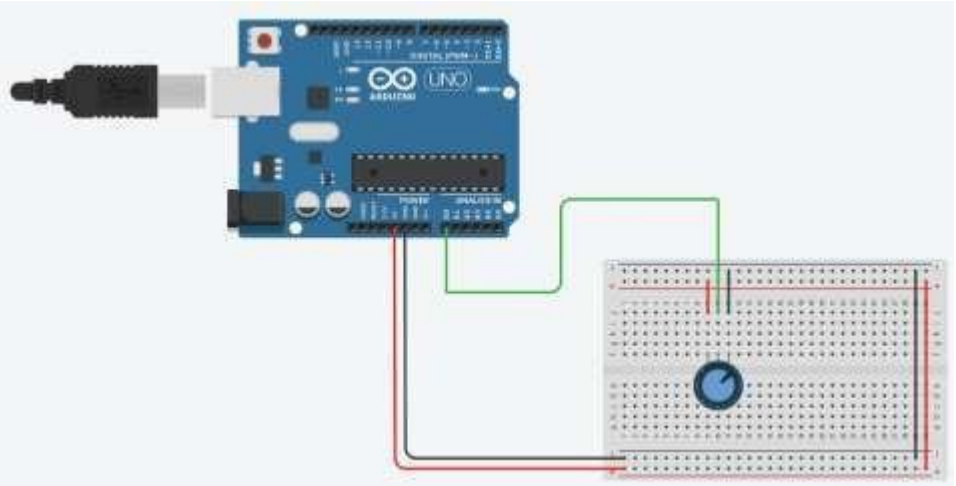

Figure 1.2: Schematic of reading and interpreting the voltage signals produced by a potentiometer.

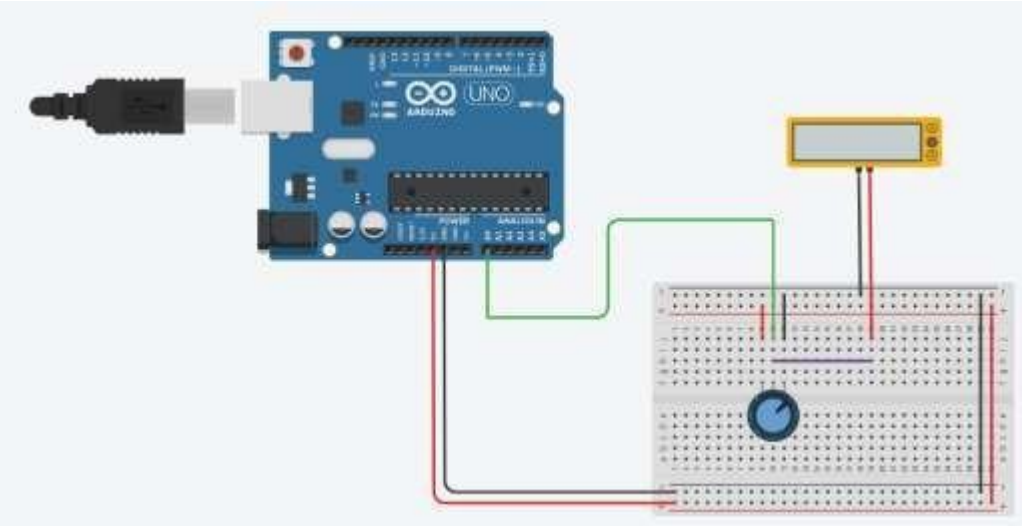

Figure 1.3: Schematic of reading the analog value from the potentiometer and converts it to a voltage reading in volts using Arduino and serial monitor

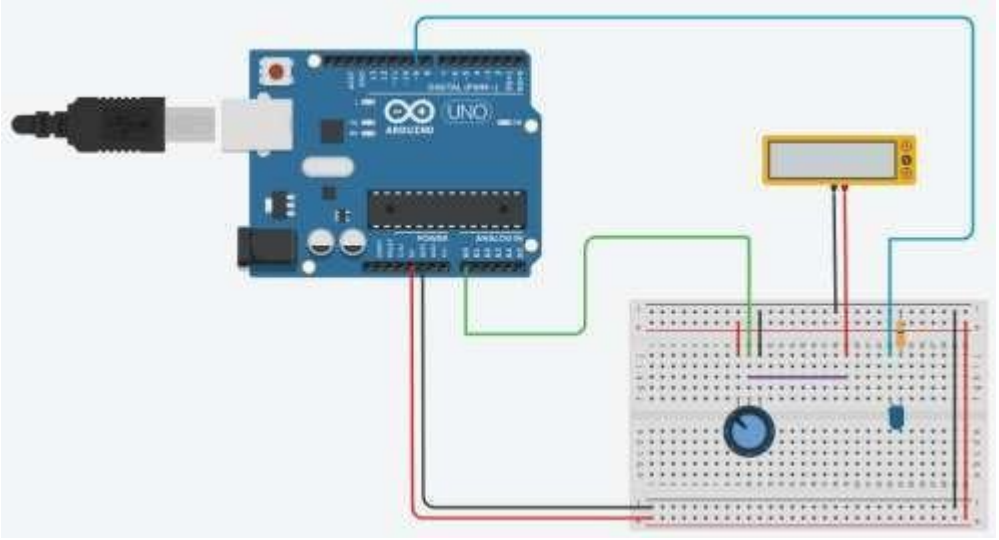

Figure 1.4: Schematic of controlling LED Brightness with Potentiometer and Serial Monitoring using Arduino

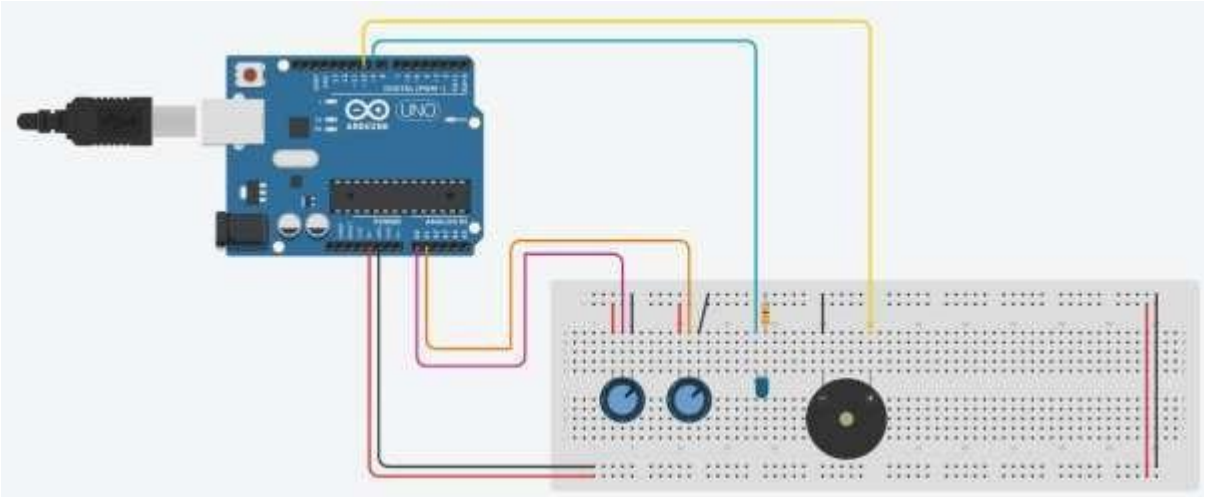

Figure 1.5: Schematic of optimizing Electronic Control with Potentiometers: A Practical Approach to Controlling Speaker Volume and LED Brightness with Arduino

Code

1.1 Read analog voltage using the "analogRead" command and convert the analog voltage to a voltage reading in volts

```
const int analogPin = A0;
const float R1 = 330.0;
const float R2 = 100.0;
const float Vcc = 5.0;
int sensorValue = 0;
float voltage = 0.0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  sensorValue = analogRead(analogPin);
  float inputVoltage = (sensorValue * Vcc) / 1023.0;
    voltage = inputVoltage * ((R1 + R2) / R2);
    Serial.print("Analog Reading: ");
  Serial.print(sensorValue);
  Serial.print(" -> Voltage: ");
  Serial.print(voltage, 3);
  Serial.println(" V");
  delay(1000);
}
```

1.2 Read and interpret the voltage signals produced by a potentiometer.

```
const int potPin = A0;
const float Vref = 5.0;
void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
  int adcValue = analogRead(potPin);
  float voltage = (adcValue * Vref) / 1023.0;
  Serial.print("ADC Value: ");
  Serial.print(adcValue);
  Serial.print("  Voltage: ");
  Serial.print(voltage, 2);
  Serial.println(" V");
  delay(500);
}
```

1.3 Read the analog value from the potentiometer and converts it to a voltage reading in volts using Arduino and serial monitor

```
const int potPin = A0;
const float Vref = 5.0;

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

void loop() {
  int adcValue = analogRead(potPin);
  float voltage = (adcValue * Vref) / 1023.0;
  Serial.print("ADC Value: ");
  Serial.print(adcValue);
  Serial.print("  Voltage: ");
  Serial.print(voltage, 2);
  Serial.println(" V");
  delay(500);
}
```

1.4 Controlling LED Brightness with Potentiometer and Serial Monitoring using Arduino

1.4.1: Controlling LED brightness with potentiometer without using map() function.

```
int potPin = A0;
int ledPin = 9;
int potValue = 0;
int ledBrightness = 0;

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

void loop() {
  potValue = analogRead(potPin);
  ledBrightness = potValue / 4;
  analogWrite(ledPin, ledBrightness);
  Serial.print("Potentiometer Value: ");
  Serial.print(potValue);
```

```
    Serial.print(" LED Brightness: ");
    Serial.println(ledBrightness);
    delay(100);
   }
```

1.4.2: Controlling LED brightness with potentiometer using map() function.

```
int potPin = A0;
int ledPin = 9;
int potValue = 0;
int ledBrightness = 0;

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

void loop() {
 potValue = analogRead(potPin);
 ledBrightness = map(potValue, 0, 1023, 0, 255);
 analogWrite(ledPin, ledBrightness);
 Serial.print("Potentiometer Value: ");
 Serial.print(potValue);
 Serial.print(" LED Brightness: ");
 Serial.println(ledBrightness);
 delay(100);
}
```

1.5 Optimizing Electronic Control with Potentiometers: A Practical Approach to Controlling Speaker Volume and LED Brightness with Arduino

```
int potLed = A0;
int potSpeaker = A1;
int ledPin = 9;
int speakerPin = 10;
int potLedValue;
int potSpeakerValue;
int ledOutput;
int speakerOutput;
void setup() {
 pinMode(ledPin, OUTPUT);
 pinMode(speakerPin, OUTPUT);}
void loop() {
 potLedValue = analogRead(potLed);
 potSpeakerValue = analogRead(potSpeaker);
 ledOutput = map(potLedValue, 0, 1023, 0, 255);
 speakerOutput = map(potSpeakerValue, 0, 1023, 0, 255);
 analogWrite(ledPin, ledOutput);
 analogWrite(speakerPin, speakerOutput);
 delay(10);}
```

Observation

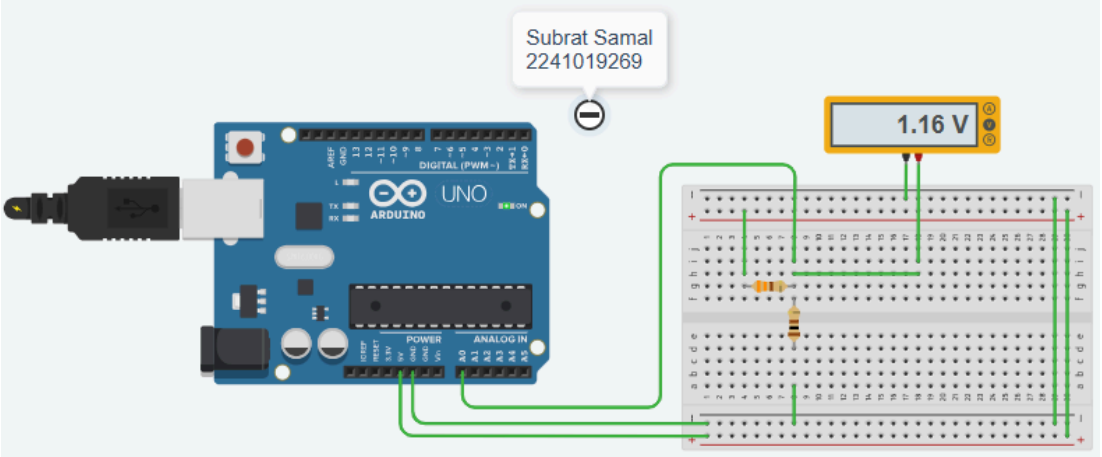

Figure 1.1.1: Simulation based reading analog voltage using the "analogRead" command and convert the analog voltage to a voltage reading in volts

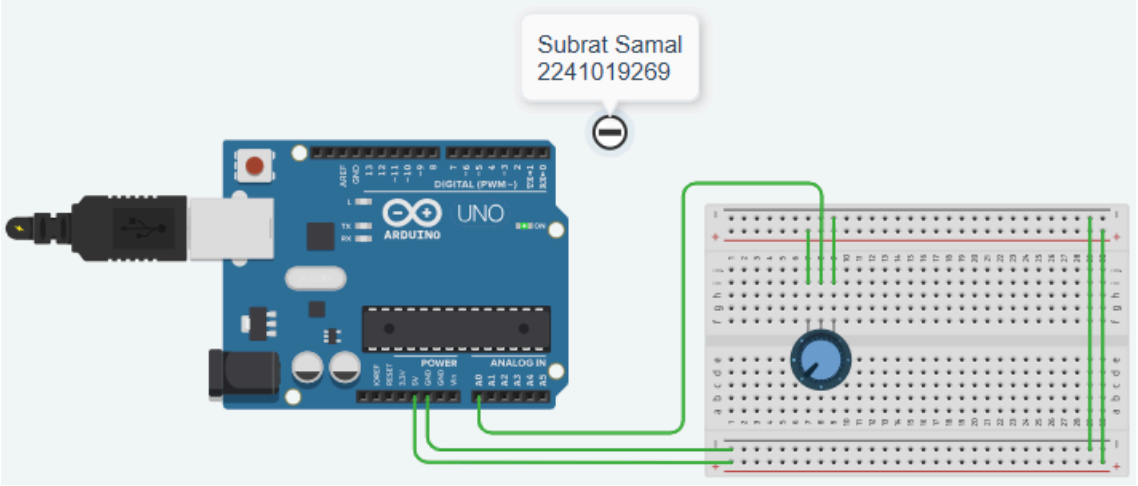

Figure 1.2.1: Simulation based reading and interpreting the voltage signals produced by potentiometer.

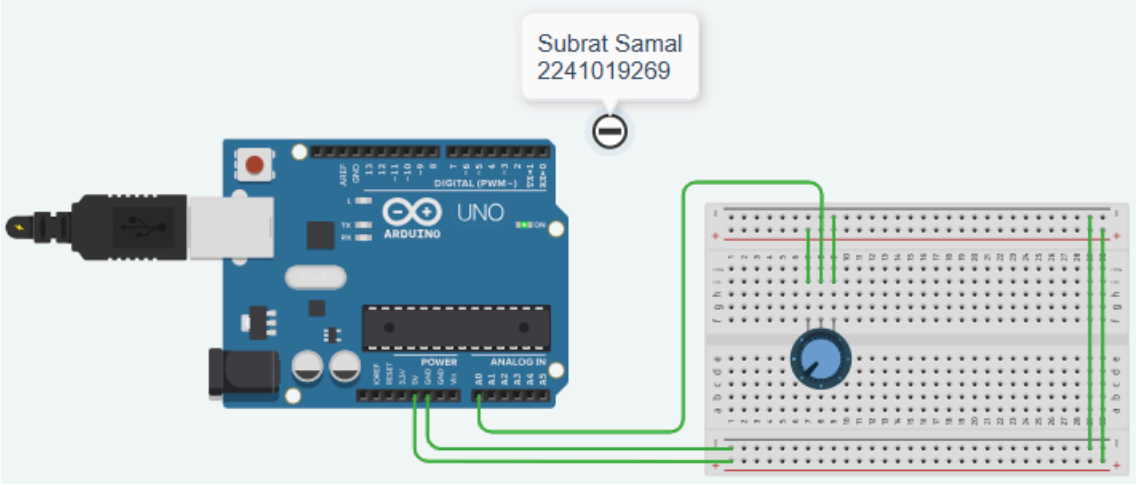

Figure 1.3.1: Simulation based reading and interpreting the voltage signals produced by a potentiometer.

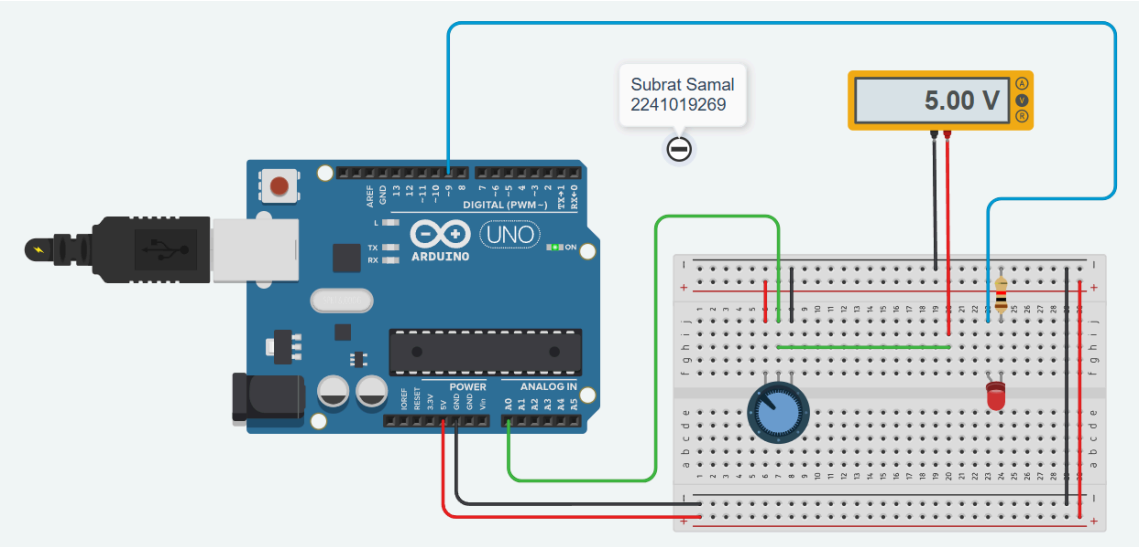

Figure 1.4.1: Simulation based controlling LED Brightness with Potentiometer and Serial Monitoring using Arduino

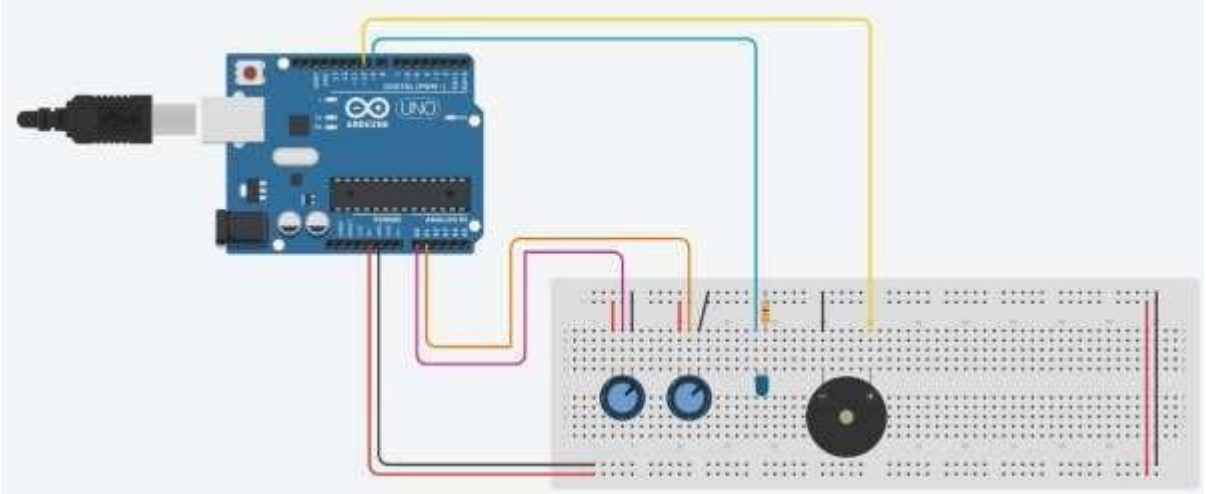

Figure 1.5.1: Simulation based optimizing Electronic Control with Potentiometers: A Practical Approach to Controlling Speaker Volume and LED Brightness with Arduino

Figure 1.5.2: Hardware Implementation based optimizing Electronic Control with Potentiometers: A

Objective 2

Understand the function and operation of an LDR/Photoresistor in electronic circuits and its ability to measure changes in light intensity.

2.1 Comprehend the relationship between light intensity and the resistance of photoresistors and how they can be used to measure changes in light intensity.

- ✔ Develop a comprehensive understanding of the photoresistor and its ability to measure the brightness of the room, where the resistance of the photoresistor is inversely proportional to the brightness of the light.
- ✔ Hook up a photoresistor in series with a fixed resistor to obtain a measurable change in voltage across the series resistor.
- ✔ Develop proficiency in programming the Arduino to read and interpret the analog signals from the photoresistor through the serial monitor.

2.2 Demonstrate knowledge of the practical application of photoresistors in electronic circuits by implementing a project involving street light control using a red and green LED.

- ✔ Read the voltage signal via an analog pin on the Arduino and use the information to control electronic components such as LEDs.
- ✔ Implement a practical application of a photoresistor by programming an Arduino to control a red and green LED based on the brightness of light, where the green LED is turned on when the light is on and the red LED is turned on when the light is off.
- ✔ Test and troubleshoot the photoresistor circuit and Arduino sketch to ensure accurate and reliable operation.

Circuit / Schematic Diagram

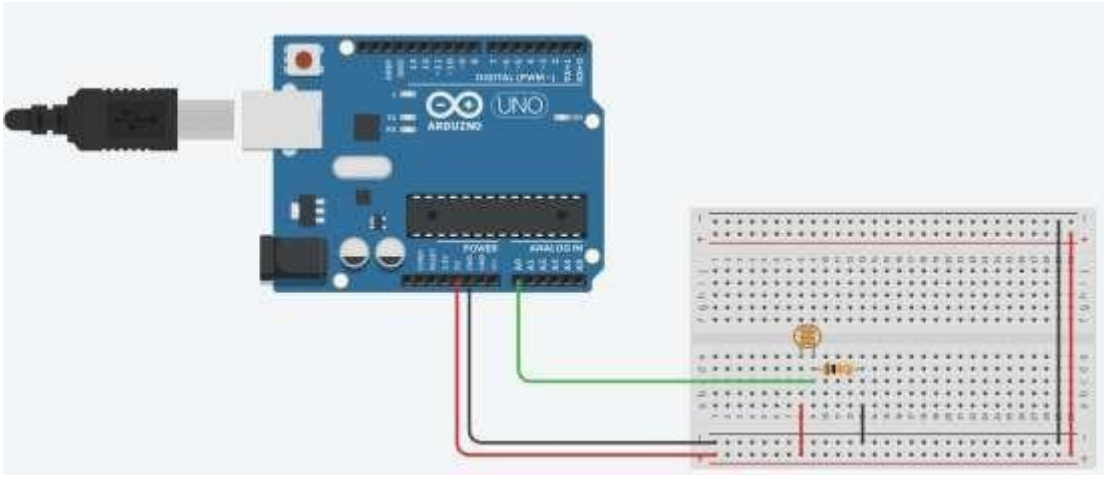

Figure 2.1: Schematic of comprehending the relationship between light intensity and the resistance of photoresistors and how they can be used to measure changes in light intensity.

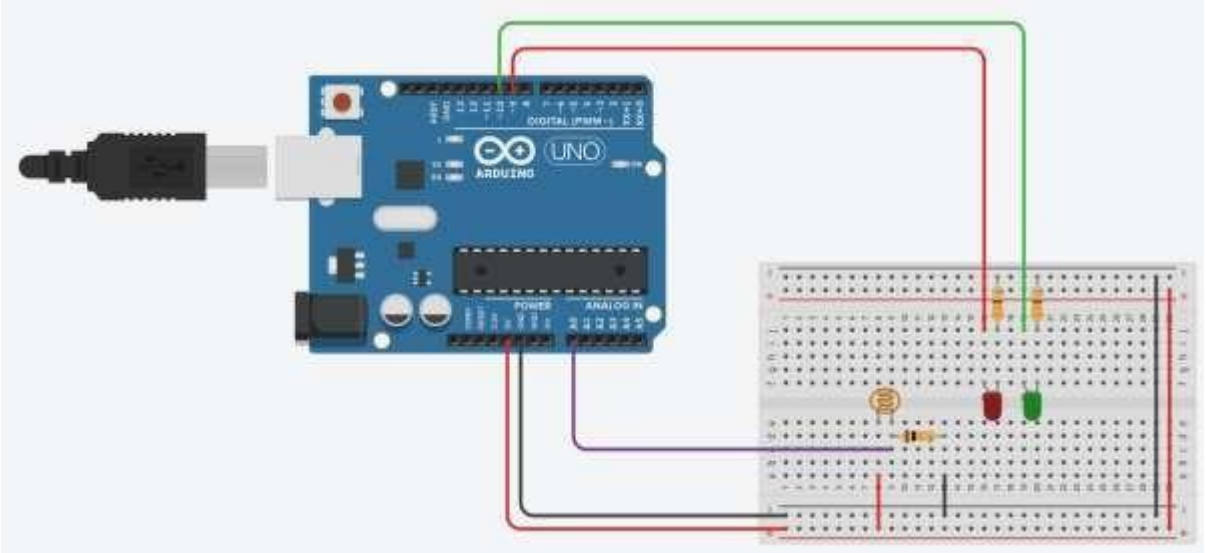

Figure 2.2: Schematic of demonstrating knowledge of the practical application of photoresistors in electronic circuits by implementing a project involving street light control using a red and green LED.

Code

2.1 Comprehend the relationship between light intensity and the resistance of photoresistors and how they can be used to measure changes in light intensity.

```
int sensorPin = A0;
int sensorValue = 0;
float voltage = 0;
float resistance = 0;
float knownResistor = 220.0;

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

void loop() {
  sensorValue = analogRead(sensorPin);
  voltage = sensorValue * (5.0 / 1023.0);
  resistance = (5.0 * knownResistor / voltage) - knownResistor;
  Serial.print("Light Intensity (approx): ");
  Serial.print(1023 - sensorValue);
  Serial.print(" | Resistance (ohms): ");
  Serial.println(resistance);
  delay(1000);
}
```

2.2 Demonstrate knowledge of the practical application of photoresistors in electronic circuits by implementing a project involving street light control using a red and green LED.

```
int ldrPin = A0;
int redLED = 9;
int greenLED = 10;
int ldrValue = 0;
int threshold = 150;

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

void loop() {
  ldrValue = analogRead(ldrPin);
  Serial.println(ldrValue);

  if (ldrValue < threshold) {
    digitalWrite(redLED, HIGH);
    digitalWrite(greenLED, LOW);
  } else {
    digitalWrite(redLED, LOW);
    digitalWrite(greenLED, HIGH);
  }

  delay(500);
}
```

Observation

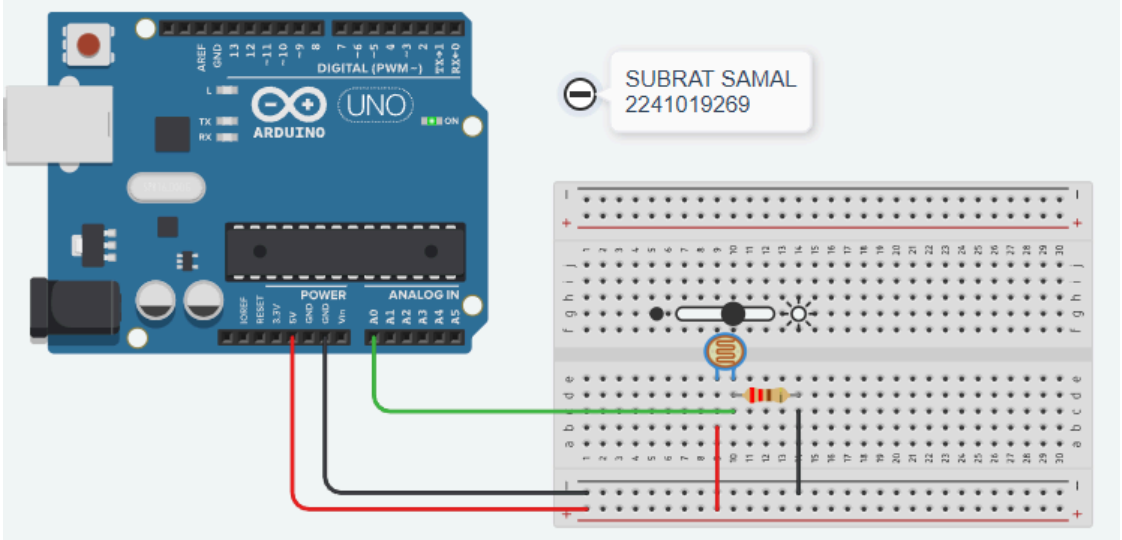

Figure 2.1.1: Simulation based comprehending the relationship between light intensity and the resistance of photoresistors and how they can be used to measure changes in light intensity.

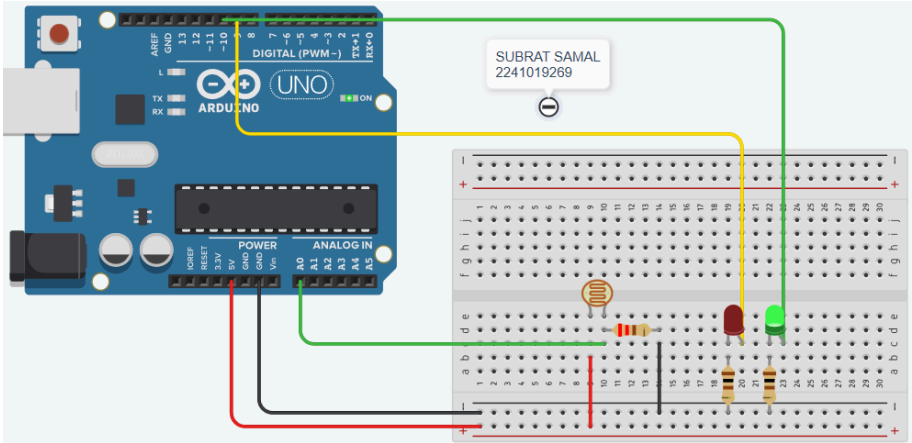

Figure 2.1.1: Simulation based demonstrating knowledge of the practical application of photoresistors in electronic circuits by implementing a project involving street light control using a red and green LED

Figure 2.1.2: Hardware Implementation based demonstrating knowledge of the practical application of photoresistors in electronic circuits by implementing a project involving street light control using a red and green LED.

Objective 3

Create an audible signal that changes tone based on the brightness of the light in the room by utilizing a photoresistor to measure light and a passive buzzer to produce the audible signal.

3.1 Effectively implement the photoresistor and passive buzzer in an electronic circuit, ensuring accurate and reliable measurement of the light and creation of the audible signal.

- ✔ Use the serial monitor to monitor the changes in the analog values of the photoresistor and ensure that the audible signal is proportional to the brightness of the room.

3.2 Implementation of a Photoresistor-Based Audible Signal with control Button

- ✔ Using a button to start/stop the tone generation loop in an Arduino sketch. To start/stop the sound, you can add a conditional statement that checks for a certain condition, such as a button press, and sets a variable to start/stop the loop that generates the tone.

Circuit / Schematic Diagram

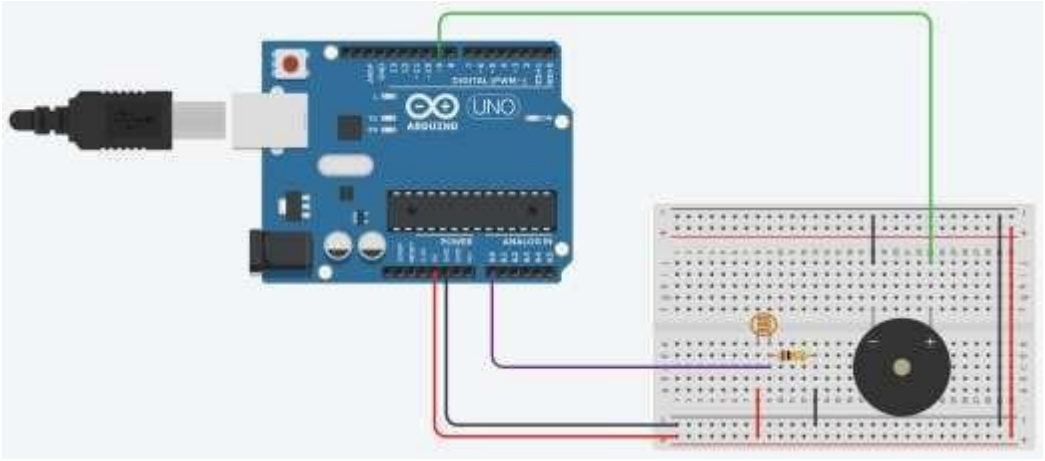

Figure 3.1: Schematic of effectively implement the photoresistor and passive buzzer in an electronic circuit, ensuring accurate and reliable measurement of the light and creation of the audible signal.

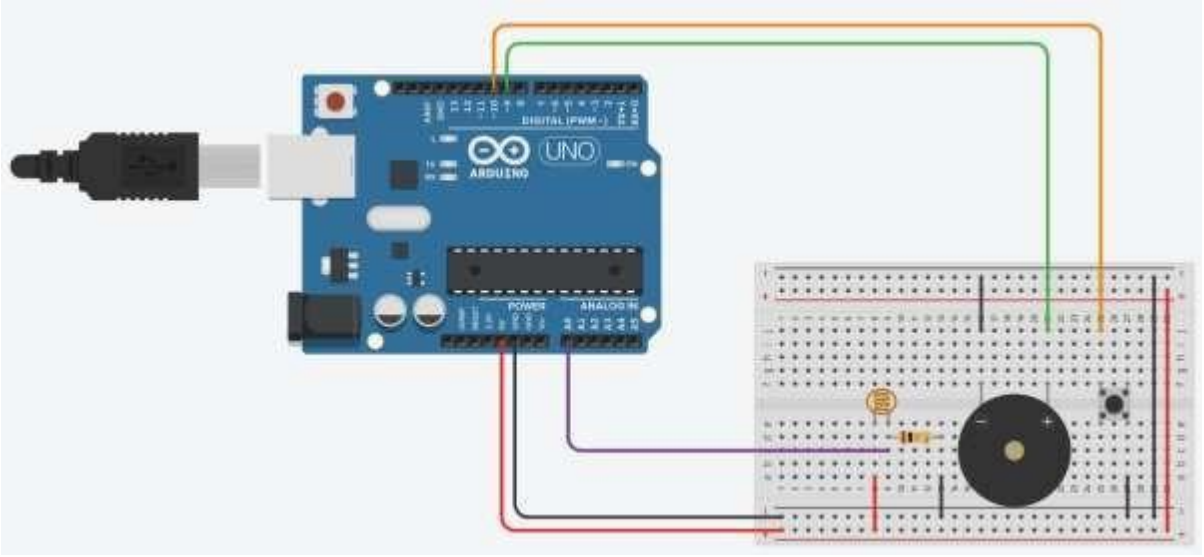

Figure 3.2: Schematic of implementation of a Photoresistor-Based Audible Signal with control Button

Code

3.1 Effectively implement the photoresistor and passive buzzer in an electronic circuit, ensuring accurate and reliable measurement of the light and creation of the audible signal.

```
int photoresistorPin = A0;
```

```
int buzzerPin = 9;
int lightLevel = 0;
int buzzerTone = 0;

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

void loop() {
  lightLevel = analogRead(photoresistorPin);
  buzzerTone = map(lightLevel, 0, 1023, 1000, 200);
  tone(buzzerPin, buzzerTone);
  Serial.print("Light Level: ");
  Serial.println(lightLevel);
  delay(100);
}
```

3.2 Implementation of a Photoresistor-Based Audible Signal with control Button

```
int ldrPin = A0;
int buzzerPin = 9;
int buttonPin = 7;

int ldrValue = 0;
int threshold = 600;
int buttonState = 0;

void setup() {
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  ldrValue = analogRead(ldrPin);
  buttonState = digitalRead(buttonPin);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);

  if (ldrValue < threshold && buttonState == LOW) {
    tone(buzzerPin, 1000);
  } else {
    noTone(buzzerPin);
  }

  delay(200);
}
```

Observation

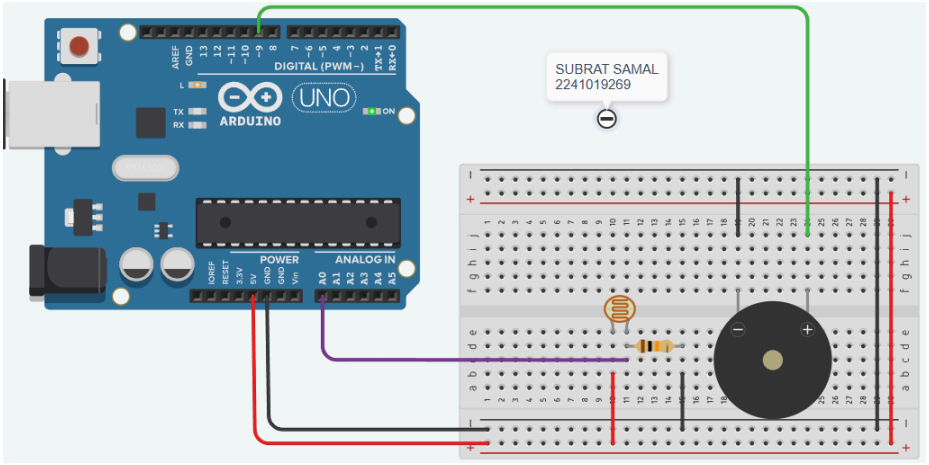

Figure 3.1.1: Simulation based effectively implement the photoresistor and passive buzzer in an electronic circuit, ensuring accurate and reliable measurement of the light and creation of the audible signal.

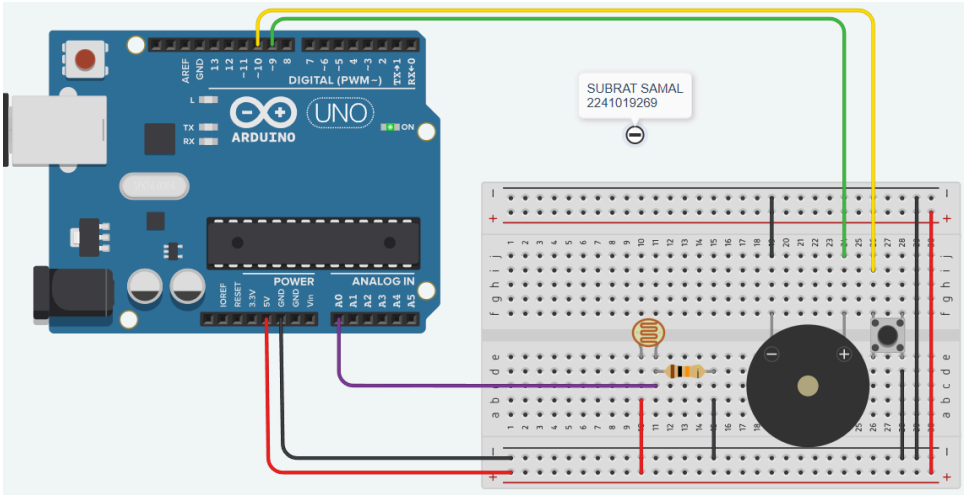

Figure 3.2.1: Simulation based implementation of a Photoresistor-Based Audible Signal with control Button.

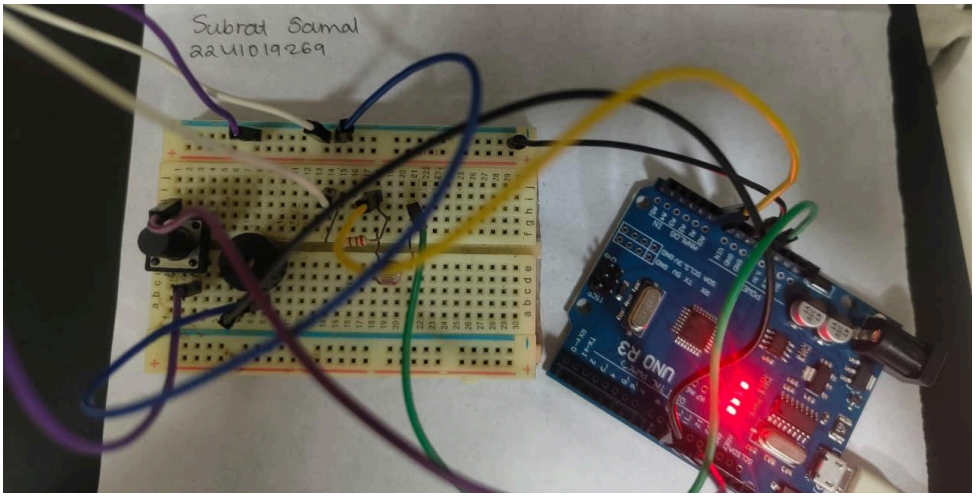

Figure 3.2.2: Hardware Implementation based implementation of a Photoresistor-Based Audible Signal with control Button.

Objective 4

Exploring RGB Mixing with Arduino: Connecting, Programming, and Controlling Primary and Intermediate Colors using PWM Signals

4.1 Develop the ability to experiment with RGB mixing and create other colors by varying the levels of Red, Green, and Blue channels.

- ✔ Learn the fundamental concepts of RGB LEDs and how they work to produce different colors.
- ✔ Understand the differences between common anode and common cathode RGB LEDs and learn how to connect a common cathode RGB LED to an Arduino.
- ✔ Program an Arduino to get the primary colors by mixing different combinations of Red, Green, and Blue channels using PWM signals.

4.2 Achieve the ability to connect an RGB LED and program it to display primary and intermediate colors based on user input.

- ✔ Develop a program that allows user for input to control the primary and intermediate colors displayed by the RGB LED.
- ✔ Create examples of primary colors, such as Cyan, Magenta, and Yellow, and explore the possibilities of mixing in-between colors.
- ✔ Test the program and RGB LED thoroughly to ensure accurate and reliable color mixing.

Circuit / Schematic Diagram

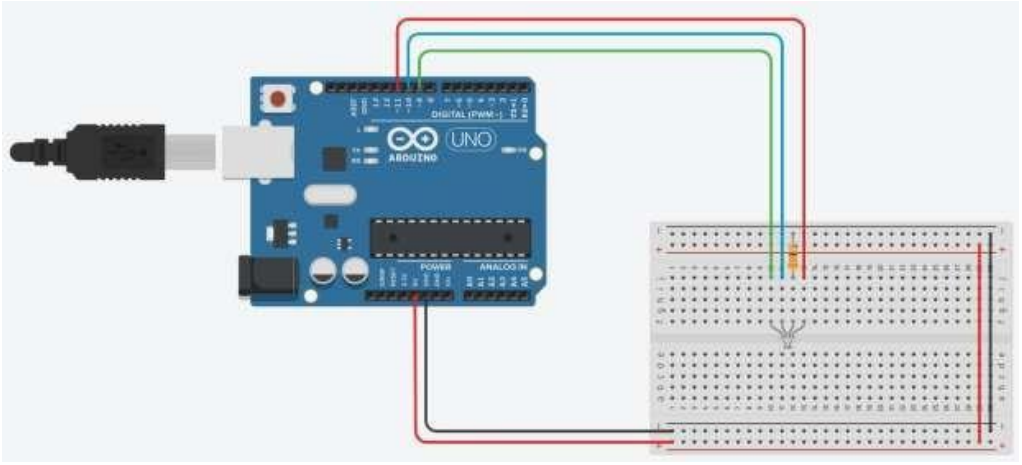

Figure 4.1: Schematic of developing the ability to experiment with RGB mixing and create other colors by varying the levels of Red, Green, and Blue channels

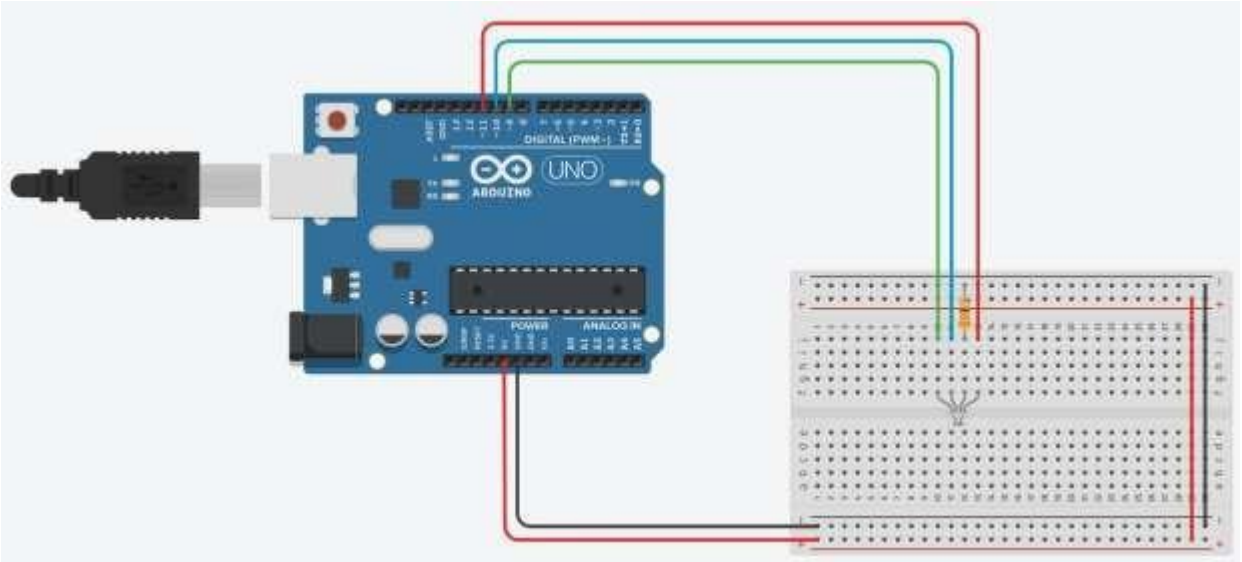

Figure 4.2: Schematic of achieving the ability to connect an RGB LED and program it to display primary and intermediate colors based on user input.

Code

4.1: Develop the ability to experiment with RGB mixing and create other colors by varying the levels of Red, Green, and Blue channels.

```
int red = 9;
int green = 10;
int blue = 11;
void setup() {
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);
}
void loop() {
  analogWrite(red, 255);
  analogWrite(green, 0);
  analogWrite(blue, 0);
  delay(1000);
  analogWrite(red, 0);
  analogWrite(green, 255);
  analogWrite(blue, 0);
  delay(1000);
  analogWrite(red, 0);
  analogWrite(green, 0);
  analogWrite(blue, 255);
  delay(1000);
  analogWrite(red, 255);
  analogWrite(green, 255);
  analogWrite(blue, 0);
  delay(1000);
  analogWrite(red, 0);
  analogWrite(green, 255);
```

```
  analogWrite(blue, 255);
  delay(1000);
  analogWrite(red, 255);
  analogWrite(green, 0);
  analogWrite(blue, 255);
  delay(1000);

  analogWrite(red, 255);
  analogWrite(green, 255);
  analogWrite(blue, 255);
  delay(1000);
}
```

4.2: Achieve the ability to connect an RGB LED and program it to display primary and intermediate colors based on user input.

```
int red = 9;
int green = 10;
int blue = 11;
String color;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);
  Serial.begin(9600);
  Serial.println("Enter color: red, green, blue, yellow, cyan, magenta, white, off");
}

void loop() {
  if (Serial.available()) {
    color = Serial.readStringUntil('\n');
    color.trim();

    if (color == "red") {
      analogWrite(red, 255); analogWrite(green, 0); analogWrite(blue, 0);
    } else if (color == "green") {
      analogWrite(red, 0); analogWrite(green, 255); analogWrite(blue, 0);
    } else if (color == "blue") {
      analogWrite(red, 0); analogWrite(green, 0); analogWrite(blue, 255);
    } else if (color == "yellow") {
      analogWrite(red, 255); analogWrite(green, 255); analogWrite(blue, 0);
    } else if (color == "cyan") {
      analogWrite(red, 0); analogWrite(green, 255); analogWrite(blue, 255);
    } else if (color == "magenta") {
      analogWrite(red, 255); analogWrite(green, 0); analogWrite(blue, 255);
    } else if (color == "white") {
      analogWrite(red, 255); analogWrite(green, 255); analogWrite(blue, 255);
    } else if (color == "off") {
```

```
      analogWrite(red, 0); analogWrite(green, 0); analogWrite(blue, 0);
    }
  }
}
```

Observation

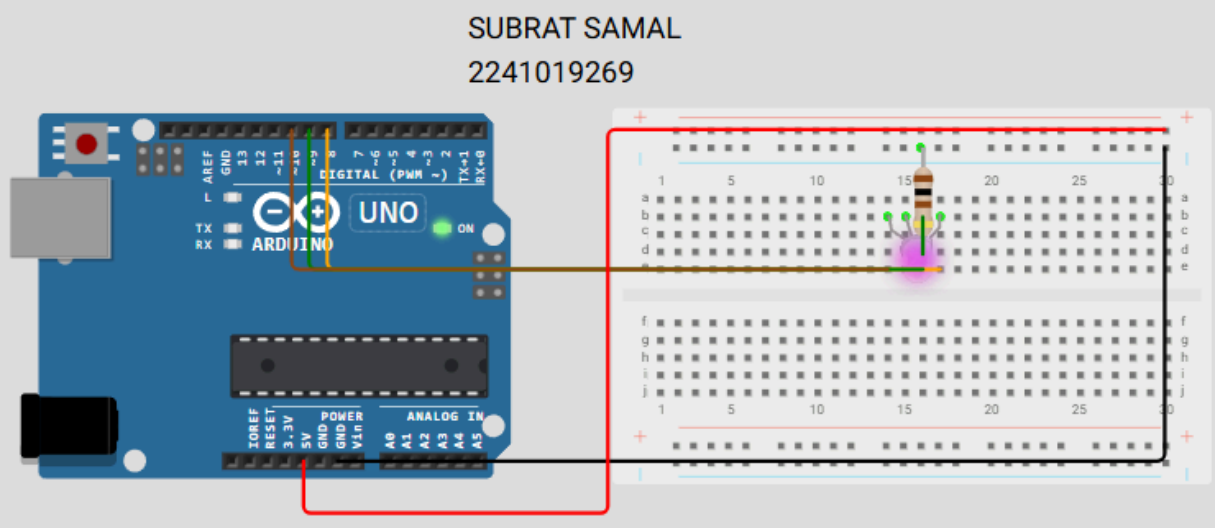

Figure 4.1.1: Simulation based developing the ability to experiment with RGB mixing and create other colors by varying the levels of Red, Green, and Blue channels and display primary and intermediate colors based on user input.

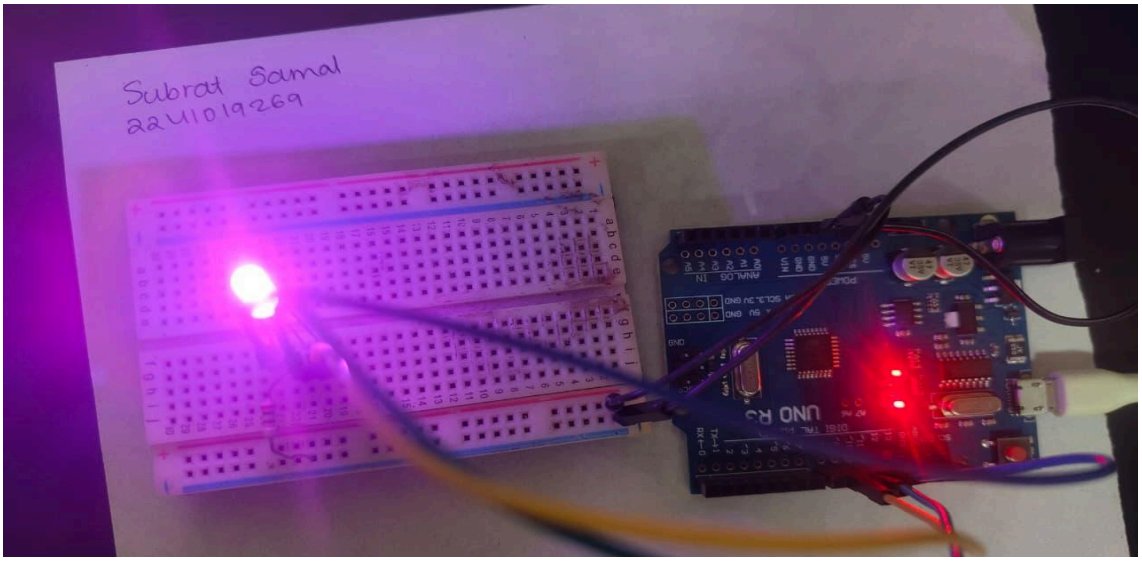

Figure 4.1.2: Hardware Implementation based developing the ability to experiment with RGB mixing and create other colors by varying the levels of Red, Green, and Blue channels and display primary and intermediate colors based on user input.

Objective 5

Implementation of an Arduino Ambient Light Mood Lamp with Start/Stop Button Control (Create a mood lamp that changes color and tone based on the ambient light level in the room with Start/Stop Button Control)

- ✔ Write an Arduino sketch that reads the analog value from the LDR, uses this value to adjust the color and tone of the RGB LED and the pitch of the buzzer, and displays the light level and other relevant data on the serial monitor. The start/stop button allows the user to control the sound and light of the lamp. When the button is pressed, the flags for stopSound and stopLight are set to true, and the sound and light are stopped.

Circuit / Schematic Diagram

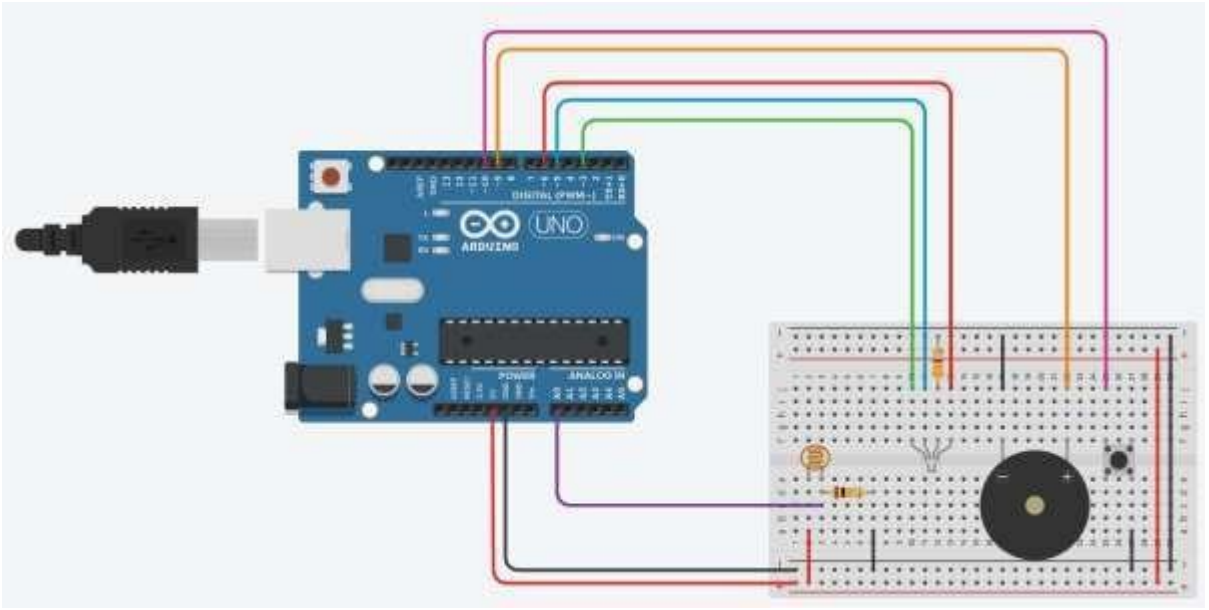

Figure 5.1: Schematic of implementation of an Arduino Ambient Light Mood Lamp with Start/Stop Button Control

Code

Implementation of an Arduino Ambient Light Mood Lamp with Start/Stop Button Control (Create a mood lamp that changes color and tone based on the ambient light level in the room with Start/Stop Button Control)

```
int red = 6;
int green = 3;
int blue = 5;
int ldr = A0;
int button = 10;
int buzzer = 9;
bool active = false;
int lastButtonState = HIGH;
unsigned long lastDebounce = 0;
```

```
unsigned long debounceDelay = 50;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);
  pinMode(button, INPUT_PULLUP);
  pinMode(buzzer, OUTPUT);
}

void loop() {
  int reading = digitalRead(button);
  if (reading != lastButtonState) lastDebounce = millis();
  if ((millis() - lastDebounce) > debounceDelay) {
    if (reading == LOW && lastButtonState == HIGH) active = !active;
  }
  lastButtonState = reading;

  if (active) {
    int lightLevel = analogRead(ldr);
    int brightness = map(lightLevel, 0, 1023, 0, 255);
    int r = map(brightness, 0, 255, 0, 255);
    int g = 255 - brightness;
    int b = brightness / 2;
    analogWrite(red, r);
    analogWrite(green, g);
    analogWrite(blue, b);
    int toneFreq = map(lightLevel, 0, 1023, 200, 1000);
    tone(buzzer, toneFreq, 100);
    delay(100);
  } else {
    analogWrite(red, 0);
    analogWrite(green, 0);
    analogWrite(blue, 0);
    noTone(buzzer);
  }
}
```

Observation

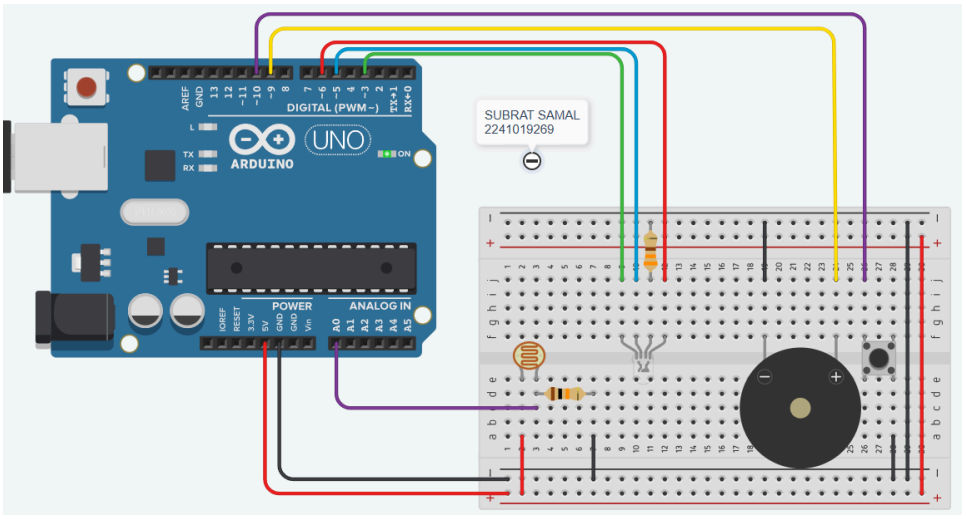

Figure 5.1.1: Simulation based implementation of an Arduino Ambient Light Mood Lamp with Start/Stop Button Control

Figure 5.1.2: Hardware Implementation based implementation of an Arduino Ambient Light Mood Lamp with Start/Stop Button Control

Conclusion:

Precautions:

Answers to Post-Lab Questions

1) If the "analogRead" command outputs a value of 512, what is the corresponding voltage reading in volts?

2) How does the value of the analog voltage reading change as the resistance of the second resistor in the voltage divider circuit is increased?

3) How do you calibrate a potentiometer reading in an Arduino sketch?

4) If the analogRead() function on the Arduino returns a value of 1023, what is the corresponding voltage reading in volts assuming a 5V reference voltage?

5) What is the maximum value of the ADC in Arduino?

6) If the ADC value of a potentiometer is 256 and the reference voltage is 5V, what is the voltage reading in volts?

7) If the reference voltage is changed to 3.3V, what is the voltage reading in volts for an ADC value of 750?

8) If the ADC value of a potentiometer is 1023, what is the voltage reading in volts for a reference voltage of 5V?

9) What is the formula to convert ADC value to voltage reading in volts?

10) What is the resolution of the analog-to-digital converter (ADC) in the Arduino board?

11) How does the Arduino handle noisy analog signals?

12) How can you adjust the range of LED brightness values using the map() function?

(Signature of the Faculty)

Date: _____

(Signature of the Student)

Name: _____

Registration No.: _____

Branch: _____

Section _____