Maharaja Surajmal Institute

C-4, Janakpuri, New Delhi

PRACTICAL FILE

BCA-372

Internet of Things Practical

Submitted to:

Dr. Menal Dahiya

Associate Professor

(Department of Computer Science)

Submitted by:

Name: Siddharth Kumar Pandey

Class: BCA 6A - Morning

Enroll no. : 03914902021

INDEX

Q1. Study and Install IDE of Arduino.

The Arduino Integrated Development Environment (IDE) is a cross-platform application written in Java that facilitates code writing, compiling, and uploading to Arduino boards. It provides a user-friendly interface for programming Arduino boards without the need for extensive knowledge of programming languages.

Key features of the Arduino IDE include:

Text editor for writing code

Syntax highlighting for different programming languages (primarily C/C++)

Built-in library management system for easy access to pre-written code

Serial monitor for debugging

Integrated compiler and uploader for Arduino boards

2. Download Arduino IDE:

You can download the Arduino IDE from the official Arduino website. Visit Arduino Software and choose the appropriate version for your operating system (Windows, Mac, or Linux).

3. Install Arduino IDE:

Once the download is complete, follow these steps to install the Arduino IDE:

Windows: Double-click the downloaded executable file and follow the installation wizard instructions.

Mac: Open the downloaded disk image file (.dmg), drag the Arduino application to the Applications folder, and then eject the disk image.

Linux: Extract the downloaded archive to a location of your choice, navigate to the extracted folder, and run the install.sh script in a terminal.

4. Launch Arduino IDE:

After successful installation, you can launch the Arduino IDE from your applications menu or by double-clicking its icon on the desktop.

Q2. Write the steps to add libraries in Arduino and setup of Arduino IDE for programming.

Adding Libraries in Arduino IDE:

Open Arduino IDE: Launch the Arduino IDE on your computer.

Navigate to Library Manager: Go to Sketch > Include Library > Manage Libraries....

Search for Libraries: In the Library Manager window, you can search for libraries by name. Type the name of the library you want to install in the search bar.

Install Library: Once you've found the library you need, click on it to select it. Then, click the "Install" button to download and install the library.

Library Installed: After installation, you'll see a green "Installed" label next to the library name. You can now close the Library Manager window.

Include Library in Sketch: To use the installed library in your sketch, go to Sketch > Include Library and select the library from the list. This will add the necessary #include statement to your sketch.

Setting up Arduino IDE for Programming:

Select Board: Connect your Arduino board to your computer via USB. Go to Tools > Board and select the appropriate Arduino board model you're using.

Select Port: If your Arduino board is connected to your computer, go to Tools > Port and select the port to which your Arduino is connected. If you're unsure, you can check which port your Arduino is connected to by disconnecting and reconnecting it, and observing which port disappears and reappears in the list.

Set Programmer (if required): Depending on your project requirements, you might need to set the programmer. This is usually necessary for programming bootloaders or burning firmware. Go to Tools > Programmer and select the appropriate programmer if needed.

Choose Processor (optional): For some boards, you may need to select the processor variant. Go to Tools > Processor and choose the correct processor variant if applicable.

Set Clock Speed (optional): If your project requires a specific clock speed, you can set it under Tools > CPU Frequency.

Verify/Compile Sketch: Write your Arduino code in the editor window. To verify or compile your sketch, click on the checkmark icon (✓) in the toolbar, or go to Sketch > Verify/Compile.

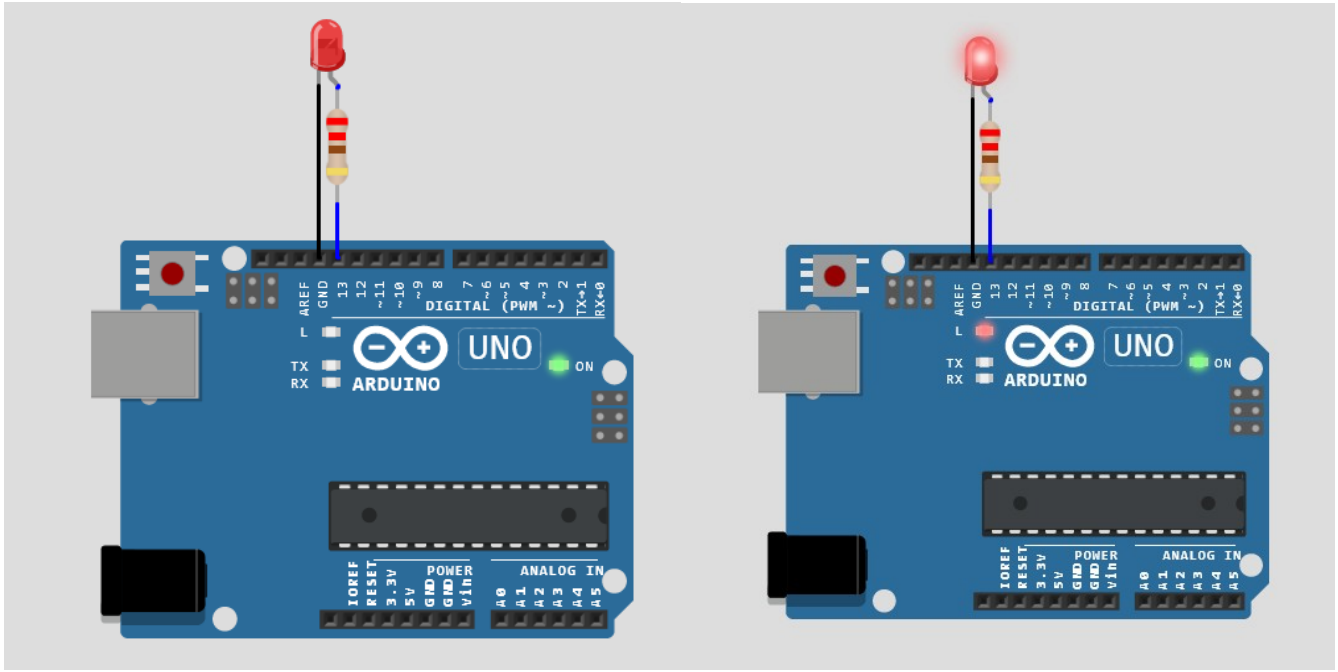
Upload Sketch: Once your sketch is verified without errors, you can upload it to your Arduino board by clicking the right arrow icon (→) in the toolbar, or by going to Sketch > Upload.

Monitor Serial Output (optional): If your sketch uses serial communication for debugging or interaction, you can open the serial monitor by clicking on the magnifying glass icon in the toolbar, or by going to Tools > Serial Monitor.

Q3. Write a Program using Arduino for Blink LED.

```
// Define the pin number for the LED
const int ledPin = 13;
void setup() {
  // Initialize the digital pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Turn the LED on (HIGH) for 1 second
  digitalWrite(ledPin, HIGH);
  delay(1000);

  // Turn the LED off (LOW) for 1 second
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

OUTPUT:

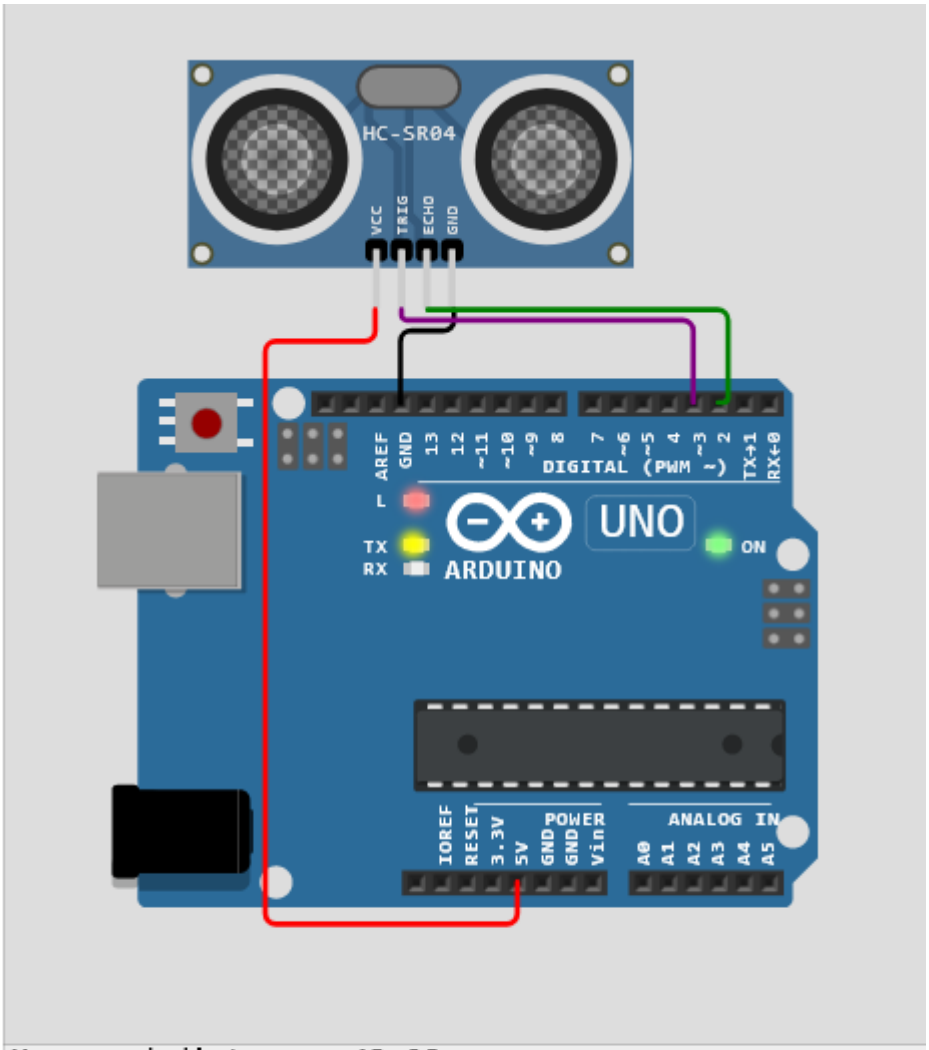


Q4. Write a Program using Arduino for Ultrasonic distance sensor.

```
// Define the pins for the ultrasonic sensor
const int trigPin = 7; // Trigger pin
const int echoPin = 6; // Echo pin

// Variables to store the distance and duration
long duration;
int distance;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set the trigPin as an output
  pinMode(trigPin, OUTPUT);
  // Set the echoPin as an input
  pinMode(echoPin, INPUT);
}

void loop() {
  // Clear the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Send a 10 microsecond pulse to trigger the sensor
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the duration of the pulse from the echoPin
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance = duration * 0.034 / 2;

  // Print the distance to the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Wait for a short delay before taking the next reading
```

```
  delay(1000);
}
```

OUTPUT:

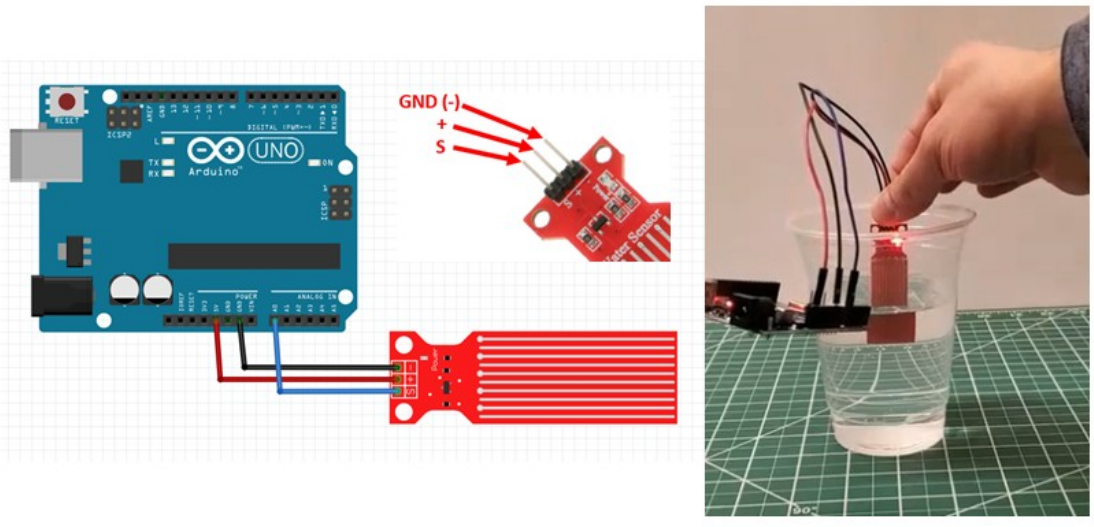



```
Measured distance: 43.98
Measured distance: 43.98
Measured distance: 43.98
Measured distance: 43.98
Measured distance: 43.98
Measured distance: 44.08
Measured distance: 44.08
```

Q5. Write a Program using Arduino for water level sensor.

```
// Define the pin for the water level sensor
const int waterLevelPin = 2;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set the waterLevelPin as an input
  pinMode(waterLevelPin, INPUT);
}

void loop() {
  // Read the water level sensor state
  int waterLevel = digitalRead(waterLevelPin);

  // Check if the water level is high or low
  if (waterLevel == HIGH) {
    Serial.println("Water level: High");
  } else {
    Serial.println("Water level: Low");
  }

  // Add a short delay before taking the next reading
  delay(1000);
}
```

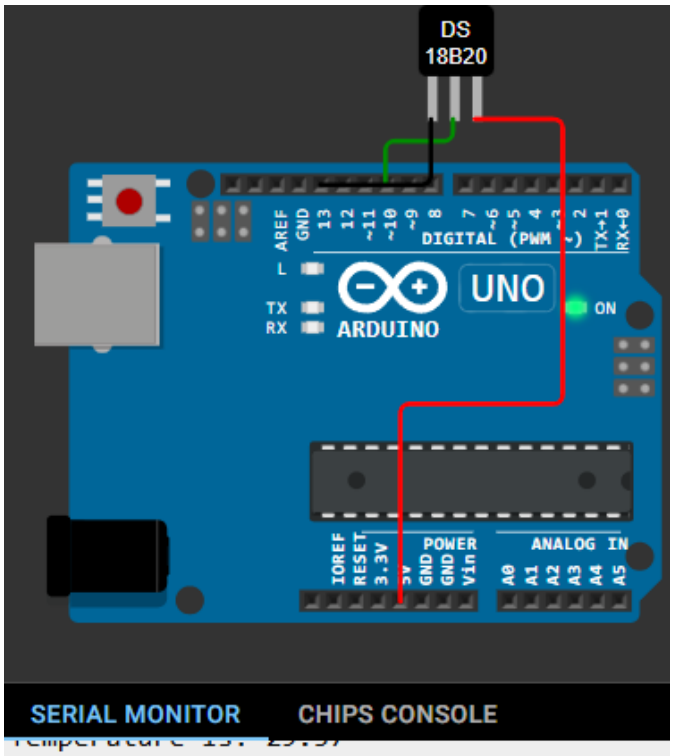
OUTPUT:



Q6. Write a Program for monitoring Temperature using Arduino and LM35 Temperature Sensors.

```
const int lm35Pin = A0;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);
}
void loop() {
  // Read the analog voltage from the LM35 sensor
  int sensorValue = analogRead(lm35Pin);
  // Convert the analog value to temperature in Celsius
  float temperature = (sensorValue * 5.0 / 1023.0) * 100.0;
  // Print the temperature to the Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  // Add a short delay before taking the next reading
  delay(1000);
}
```

OUTPUT:

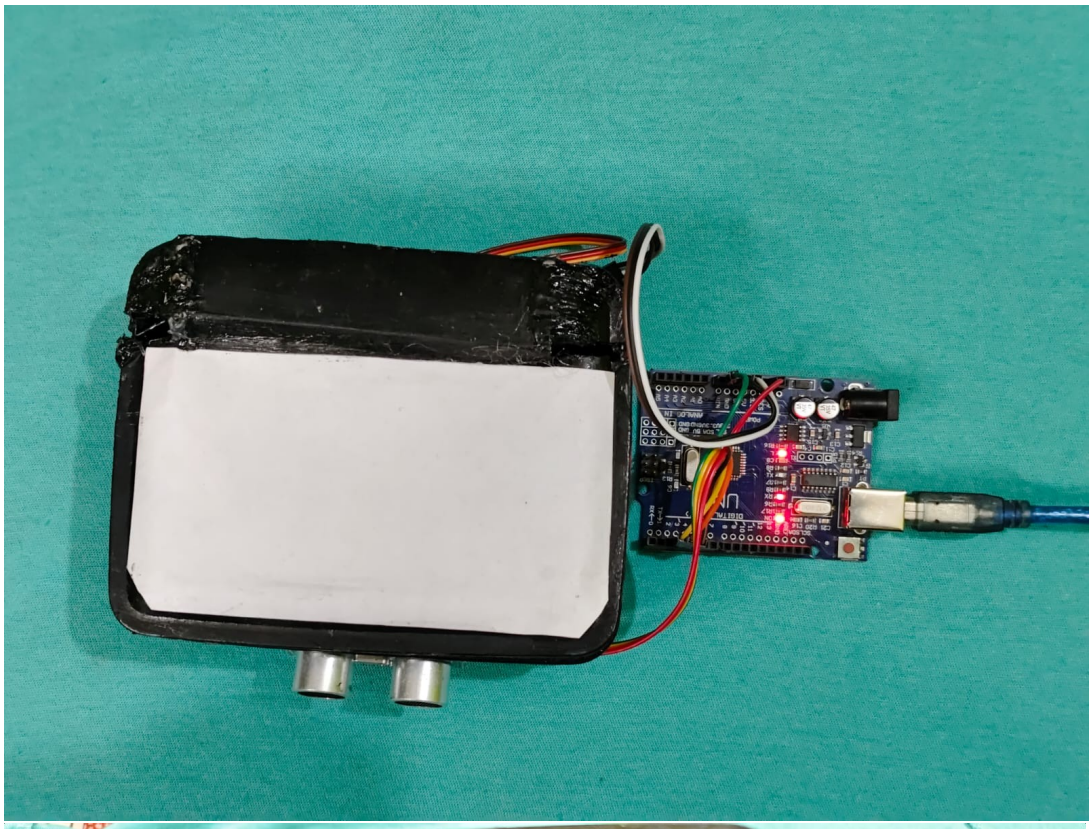


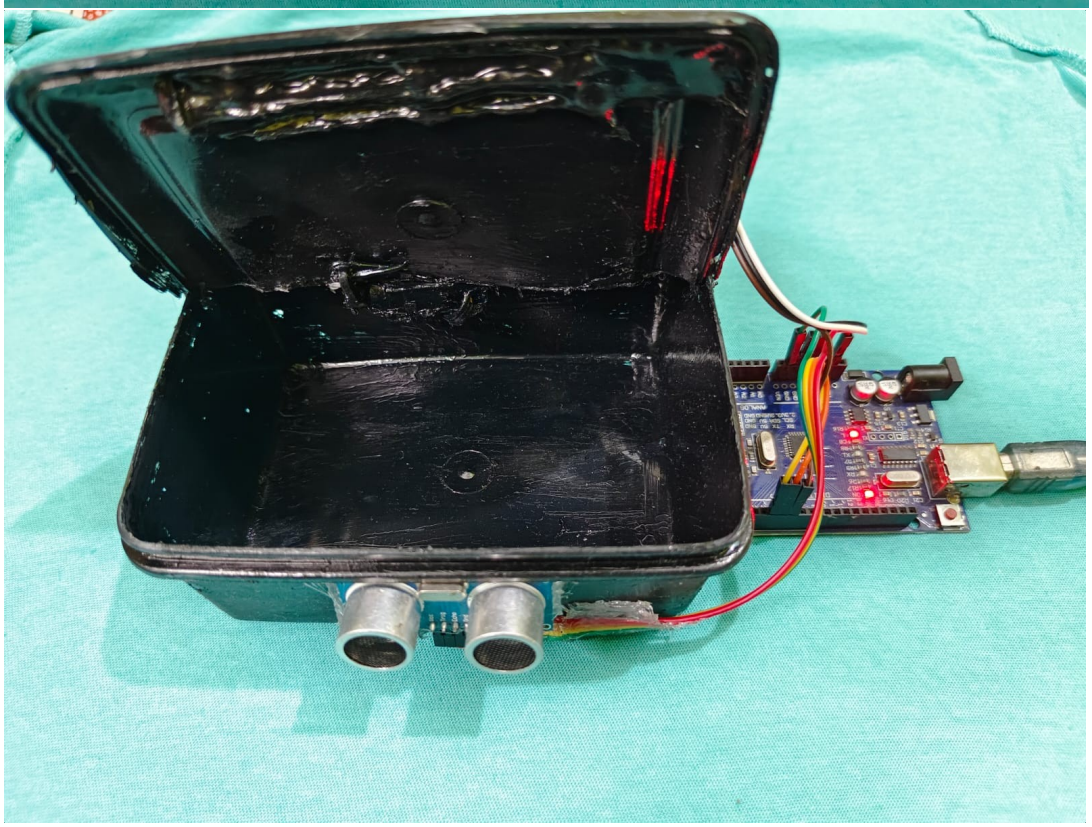
Q7. Write a program for Arduino by using Ultrasonic sensors and servo motor (HC-SR04), and make a smart dustbin.

```
#include <Servo.h>

// Define pins for ultrasonic sensor
const int trigPin = 7;
const int echoPin = 6;

// Define pins for servo motor
const int servoPin = 9;

// Define variables for servo motor and ultrasonic sensor
Servo servo;
long duration;
int distance;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize servo motor
  servo.attach(servoPin);

  // Set pins for ultrasonic sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Start with the lid closed
  servo.write(0);
}

void loop() {
  // Trigger ultrasonic sensor to measure distance
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure duration of pulse returned by ultrasonic sensor
```

```
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance based on duration
  distance = duration * 0.034 / 2;

  // Print distance to Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);

  // If distance is less than 20cm, open the lid
  if (distance < 20) {
    openLid();
  } else {
    closeLid();
  }

  // Delay before taking the next reading
  delay(1000);
}

void openLid() {
  // Open the lid of the dustbin
  servo.write(90);
  delay(1000); // Adjust delay as needed for smooth operation
}

void closeLid() {
  // Close the lid of the dustbin
  servo.write(0);
  delay(1000); // Adjust delay as needed for smooth operation
}
```

OUTPUT :

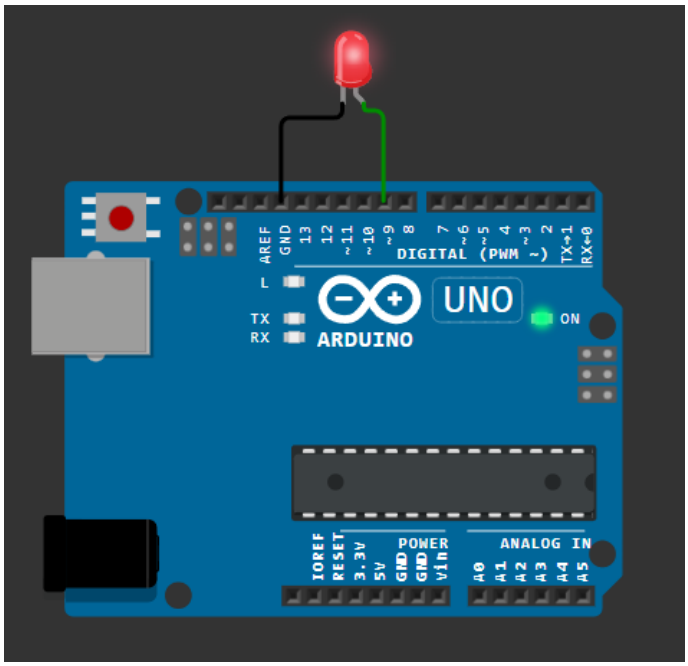




Q8. Write a program to shows how to fade an LED on pin 9 using the analog Write( ) function.

```
const int ledPin = 9;

void setup( ) {

}

void loop( ) {
  // Fade in
  for (int brightness = 0; brightness <= 255; brightness++) {
    analogWrite(ledPin, brightness);
    delay(10); // Adjust fade speed by changing delay value
  }
  // Fade out
  for (int brightness = 255; brightness >= 0; brightness--) {
    analogWrite(ledPin, brightness);
    delay(10); // Adjust fade speed by changing delay value
  }
}
```

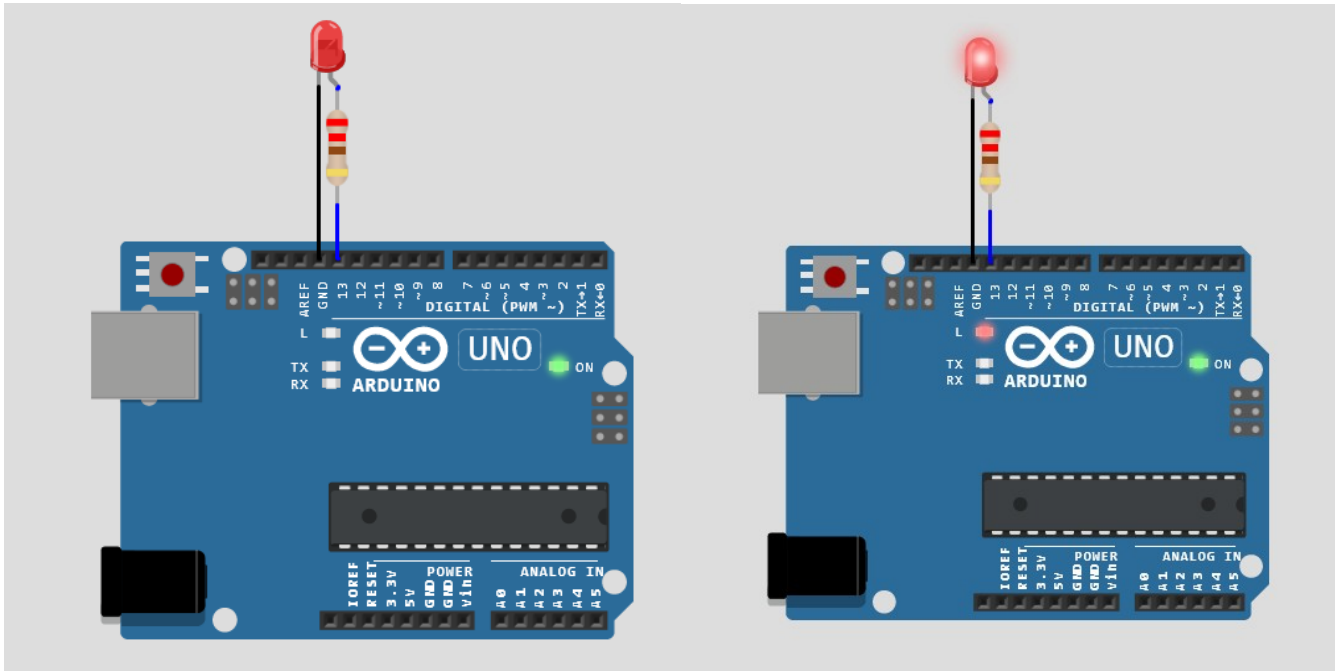
OUTPUT:



Q9. Write a program to control LED using loop.

```
// Define the pin for the LED
const int ledPin = 9;

void setup() {
  // Initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Turn the LED on
  digitalWrite(ledPin, HIGH);
  delay(1000); // Wait for 1 second

  // Turn the LED off
  digitalWrite(ledPin, LOW);
  delay(1000); // Wait for 1 second
}
```

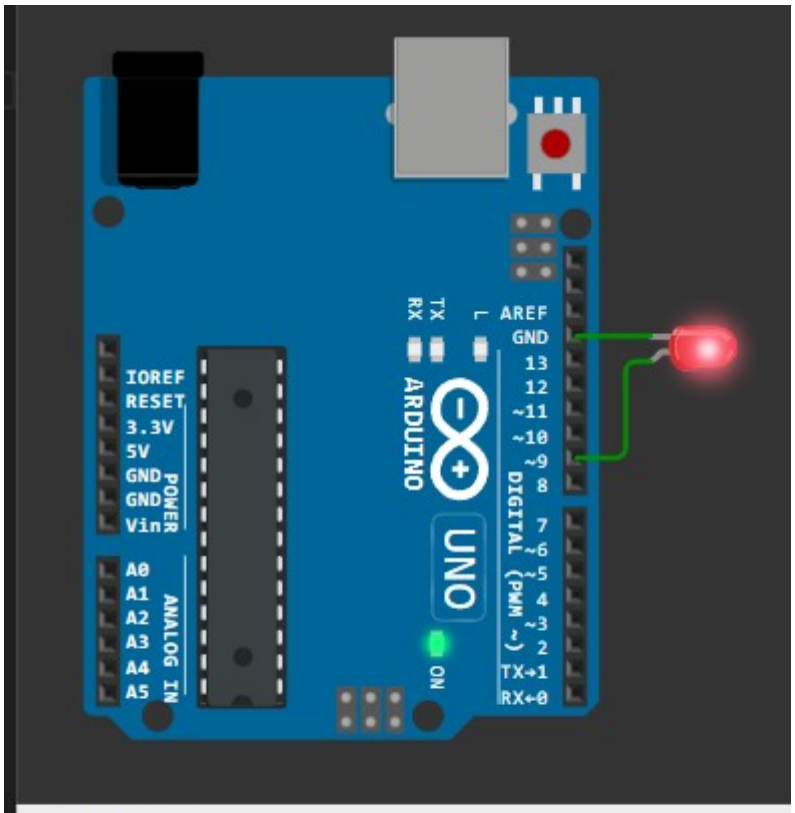
OUTPUT:



Q10. Write a program to control LED using Serial Number

```
// Define the pin for the LED
const int ledPin = 9;

void setup() {
  // Initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);

  // Initialize serial communication
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    // Read the incoming byte from Serial Monitor
    char command = Serial.read();

    // Process the command
    if (command == '1') {
      digitalWrite(ledPin, HIGH); // Turn the LED on
      Serial.println("LED is ON");
    } else if (command == '0') {
      digitalWrite(ledPin, LOW); // Turn the LED off
      Serial.println("LED is OFF");
    }
  }
}
```

OUTPUT:

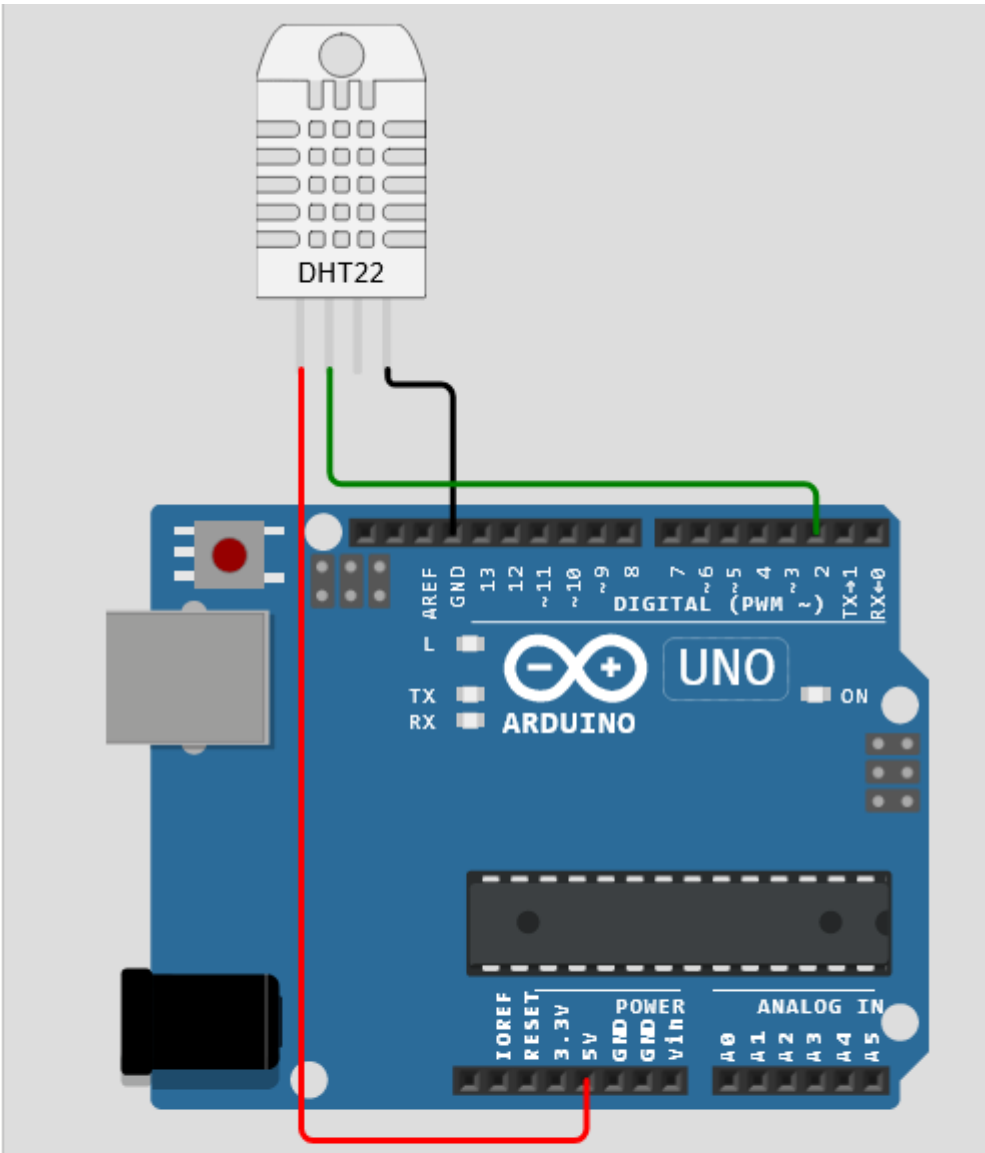



```
LED is ON
LED is OFF
LED is ON
```

Q11. Program to measure temperature and humidity

```
#include <DHT.h>
// Define the type of DHT sensor (DHT11 or DHT22)
#define DHTTYPE DHT11 // Change to DHT22 if you're using a DHT22 sensor

// Define the pin for the DHT sensor
const int dhtPin = 2; // Connect the signal pin of the DHT sensor to digital pin 2

// Initialize DHT sensor
DHT dht(dhtPin, DHTTYPE);

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize DHT sensor
  dht.begin();
}

void loop() {
  // Read temperature and humidity from DHT sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Check if any reading failed
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
  } else {
    // Print temperature and humidity to Serial Monitor
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" °C");
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
  }

  // Delay before taking the next reading
  delay(2000); // Adjust delay as needed
}
```

OUTPUT:

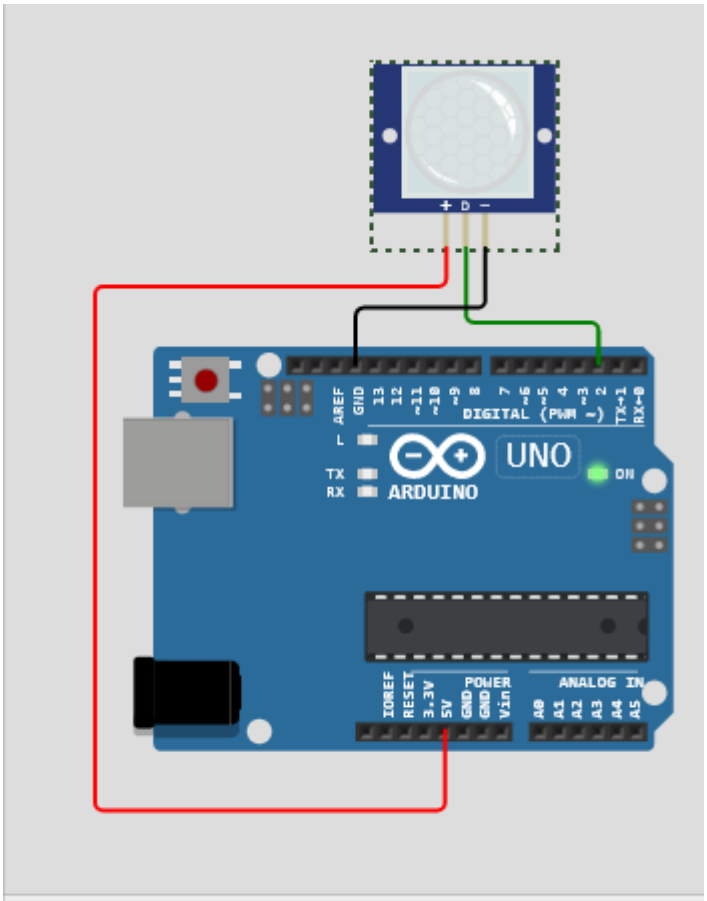



```
DHT22 example!
Humidity: 40.00%  Temperature: 24.00°C
Humidity: 40.00%  Temperature: 24.00°C
Humidity: 40.00%  Temperature: 24.00°C
Humidity: 40.00%  Temperature: 24.00°C
```

Q12. Program for motion detector or PIR sensor using Arduino.

```
// Define the pin for the PIR sensor
const int pirPin = 2; // Connect the output pin of the PIR sensor to digital pin 2

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize the PIR sensor pin as an input
  pinMode(pirPin, INPUT);

  // Allow time for the PIR sensor to stabilize
  delay(2000);

  Serial.println("Motion Detector Initialized");
}

void loop() {
  // Read the state of the PIR sensor
  int motionDetected = digitalRead(pirPin);

  // Check if motion is detected
  if (motionDetected == HIGH) {
    Serial.println("Motion detected!");
  }

  // Add a short delay before taking the next reading
  delay(500);
}
```

OUTPUT:



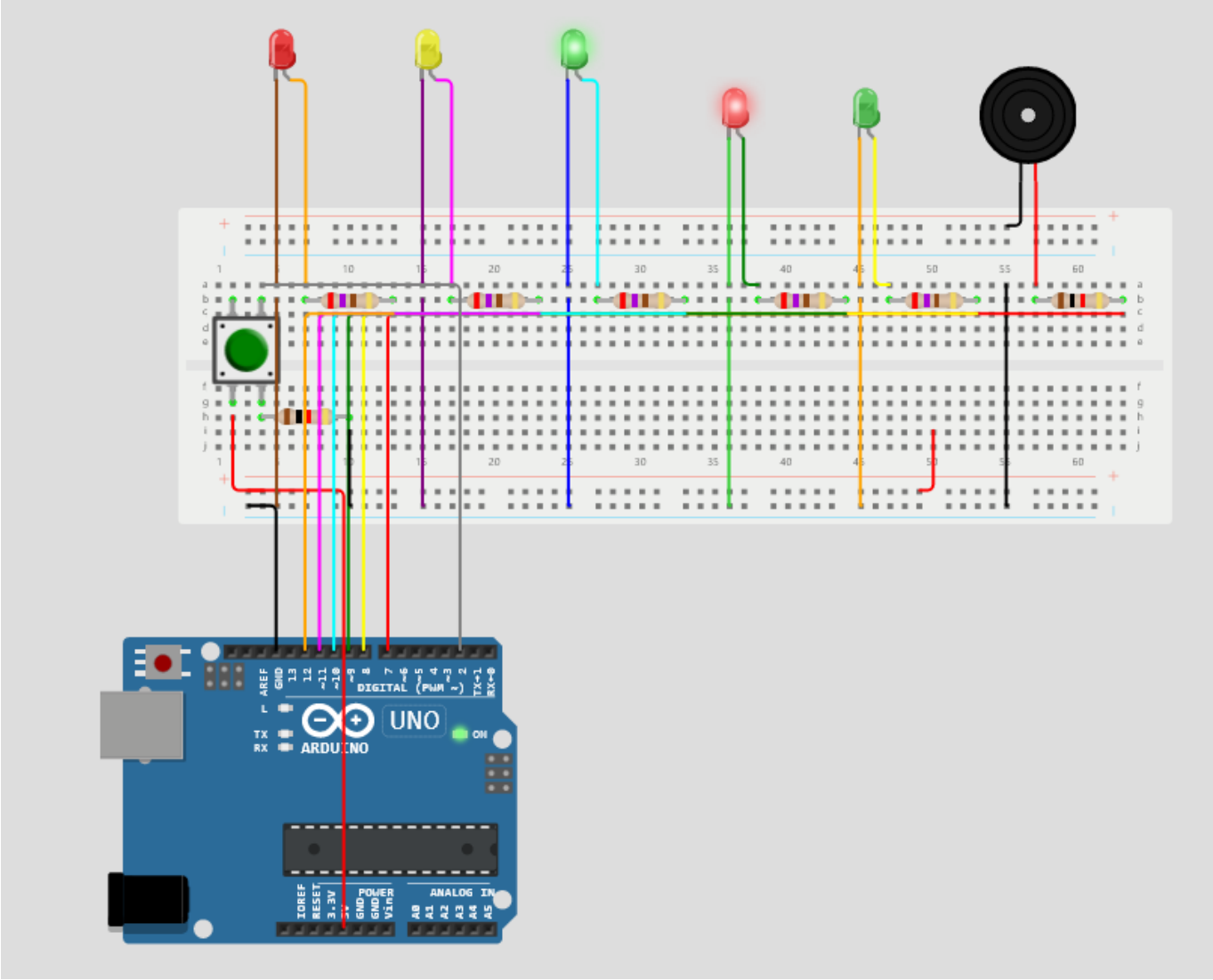

```
Motion detected!
Motion ended!
```

Q13. Program to create traffic light simulator for pedestrians.

```
// Define pin numbers for the traffic lights
const int redLight = 8;
const int yellowLight = 9;
const int greenLight = 10;
const int pedestrianRed = 11;
const int pedestrianGreen = 12;
const int buttonPin = 2;

// Define variables for timing
unsigned long previousMillis = 0;
const long interval = 2000; // 2 seconds

void setup() {
  // Initialize the pins for the traffic lights and pedestrian lights
  pinMode(redLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(greenLight, OUTPUT);
  pinMode(pedestrianRed, OUTPUT);
  pinMode(pedestrianGreen, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);

  // Initialize traffic lights
  digitalWrite(redLight, HIGH);
  digitalWrite(yellowLight, LOW);
  digitalWrite(greenLight, LOW);

  // Initialize pedestrian lights
  digitalWrite(pedestrianRed, HIGH);
  digitalWrite(pedestrianGreen, LOW);
}

void loop() {
  // Check if the button is pressed
  if (digitalRead(buttonPin) == LOW) {
    // Activate pedestrian lights
    digitalWrite(redLight, LOW);
    digitalWrite(yellowLight, LOW);
    digitalWrite(greenLight, LOW);
    digitalWrite(pedestrianRed, LOW);
    digitalWrite(pedestrianGreen, HIGH);
    delay(5000); // Pedestrians have 5 seconds to cross
```

```
    digitalWrite(pedestrianGreen, LOW);
    digitalWrite(pedestrianRed, HIGH);
    // Reset traffic lights
    digitalWrite(greenLight, HIGH);
    delay(2000); // Green light for vehicles
    digitalWrite(greenLight, LOW);
    digitalWrite(yellowLight, HIGH);
    delay(2000); // Yellow light for vehicles
    digitalWrite(yellowLight, LOW);
    digitalWrite(redLight, HIGH);
  }
}
```

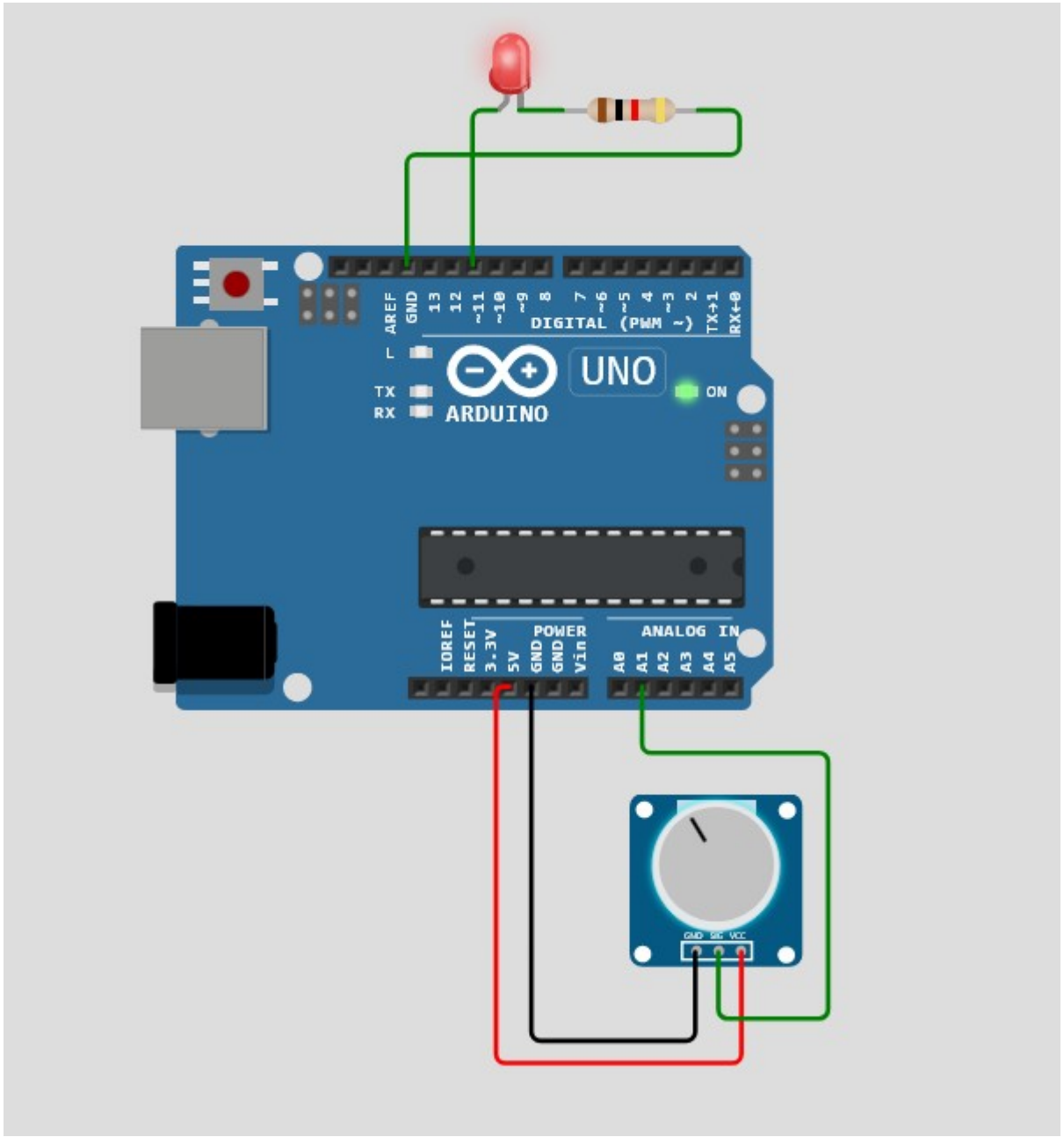
OUTPUT:



Q14. Program to create Dimmable LED using potentiometer.

```
// Define pin numbers for LED and potentiometer
const int ledPin = 9; // Connect the anode of the LED to pin 9
const int potentiometerPin = A0; // Connect the middle pin of the potentiometer to analog pin A0


void setup() {
  // Initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);

  // Initialize serial communication
  Serial.begin(9600);
}


void loop() {
  // Read the value from the potentiometer
  int potValue = analogRead(potentiometerPin);

  // Map the potentiometer value (0-1023) to the LED brightness (0-255)
  int brightness = map(potValue, 0, 1023, 0, 255);

  // Set the LED brightness using analogWrite
  analogWrite(ledPin, brightness);

  // Print the potentiometer value and LED brightness to the Serial Monitor
  Serial.print("Potentiometer Value: ");
  Serial.print(potValue);
  Serial.print(", LED Brightness: ");
  Serial.println(brightness);

  // Add a short delay to avoid flooding the Serial Monitor
  delay(100);
}
```

OUTPUT:

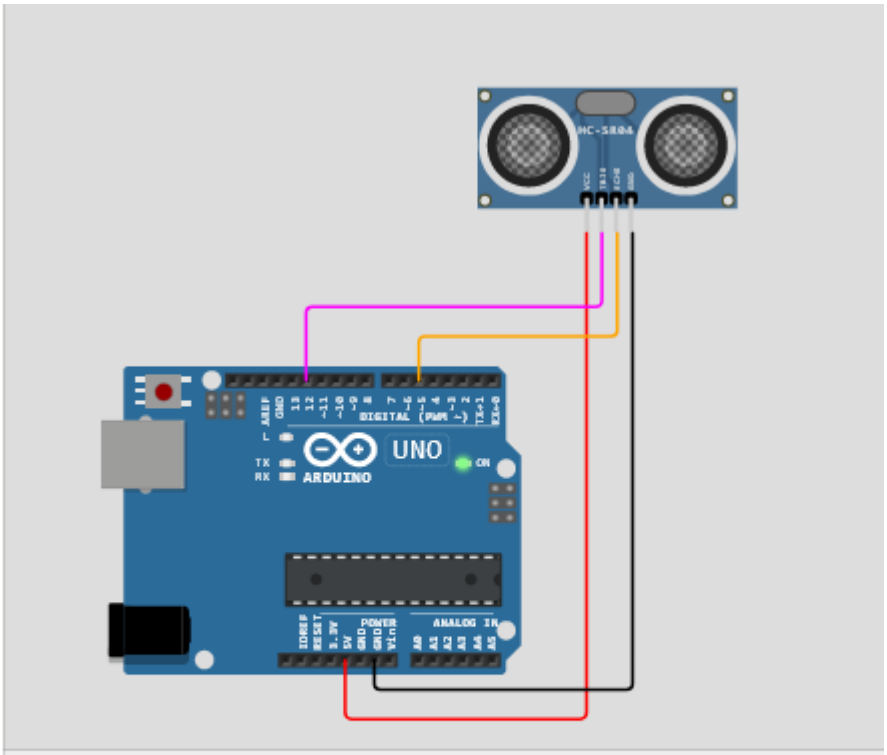

AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM ~)
L
TX
RX
ARDUINO
UNO
ON
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
GND SIG VCC

Q15. Program to measure speed of sound using ultrasonic sensor

```
// Define pins for the ultrasonic sensor
const int trigPin = 7; // Trigger pin
const int echoPin = 6; // Echo pin

// Define variables for distance and speed of sound
float distance;
float speedOfSound;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set the trigPin as an output
  pinMode(trigPin, OUTPUT);
  // Set the echoPin as an input
  pinMode(echoPin, INPUT);
}

void loop() {
  // Clear the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Send a 10 microsecond pulse to trigger the sensor
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the duration of the pulse from the echoPin
  float duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance = (duration * 0.034) / 2; // Speed of sound in air is approximately 0.034 cm/microsecond

  // Calculate the speed of sound based on the distance measured
  speedOfSound = distance / (duration / 1000000.0); // Convert microseconds to seconds

  // Print the distance and speed of sound to the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" cm, Speed of Sound: ");
```

```
  Serial.print(speedOfSound);
  Serial.println(" m/s");

  // Delay before taking the next reading
  delay(1000);
}
```

OUTPUT:




```
Speed of the sound in miles per hour: 58.30
Speed of the sound in miles per hour: 116.59
Speed of the sound in miles per hour: 233.19
```