Experiment 1

Student Name: Manvendra Singh Chauhan **UID: 22BCS12287**
Branch: Computer Science and Engineering **Section/Group: 634 A**
Semester: 5th **Date of Performance: 18/07/24**
Subject Name: IoT Architecture Lab **Subject Code: 22CSP-329**

1. Aim:

To connect Arduino Uno controller to a computer system/laptop and complete the essential software setup.

2. Objective:

1. To study hardware and software related to IoT
2. To understand the function of Arduino Uno and other controllers.

3. Equipment Used:

1. Laptop
2. Arduino Uno
3. Connecting Cable

4. Arduino Board:

- An Arduino is a microcontroller-based kit commonly used for communications and controlling various devices.
- The Arduino UNO board is the most popular model in the Arduino family and is considered the best choice for beginners in electronics and coding.
- While some boards may look slightly different, most Arduinos share the same essential components.
- The Arduino includes two types of memory: program memory and data memory.

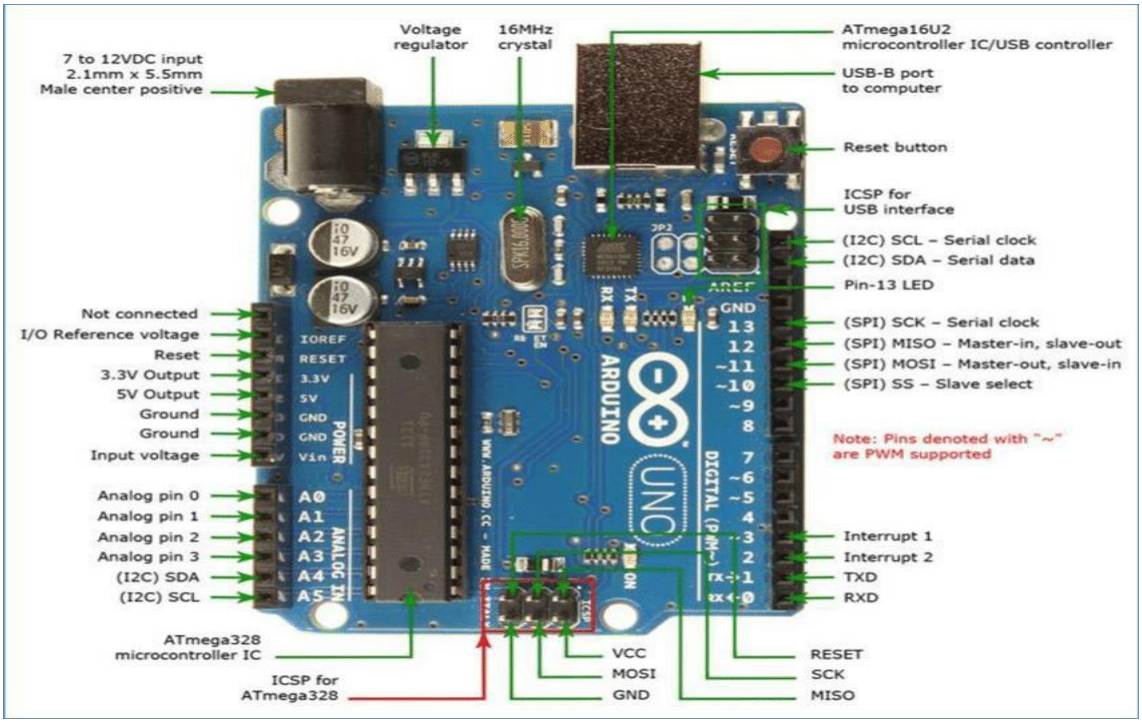


Fig: 1 Arduino Uno Board

5. Procedure-

1. Gather Materials:Arduino Uno board USB cable (Type A to B) Computer or laptop
2. Install Arduino IDE:Go to the Arduino Software page.
3. Download the appropriate version of the Arduino IDE
4. Connect Arduino to Computer
5. Plug the USB cable into the Arduino Uno board.
6. Connect the other end of the USB cable to an available USB port on your computer.
7. The power LED on the Arduino should light up, indicating it is receiving power.
8. Launch Arduino IDE
9. Open the Arduino IDE on your computer.
10. If this is the first time you are using the Arduino IDE, you may need to complete an initial setup.
11. Select Arduino Board
12. Go to Tools > Board.Select Arduino/Genuino Uno from the list of available boards.

13. Select Port Go to Tools > Port.
14. Select the port that corresponds to your Arduino Uno. This will usually be labeled with Arduino/Genuino Uno.
15. Install Drivers (if needed): Click the checkmark icon in the Arduino IDE to compile the sketch. If the sketch compiles successfully, click the right-arrow icon to upload it to the Arduino Uno.

1. Click on the button to Agree

2. Install the components

3. Select the folder

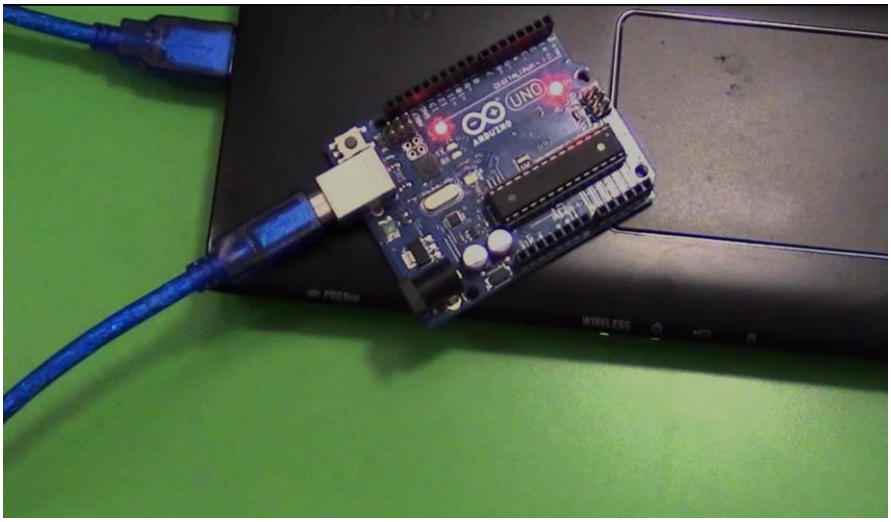
4. Wait for the Program to finish

6. Connections-

USB Cable: Use a USB cable (Type A to B).
Connect to Arduino: Plug the Type B end into the Arduino Uno.
Connect to Computer: Plug the Type A end into an available USB port on your computer.
Power Indicator: The power LED on the Arduino should light up, indicating a successful connection.
Data Transfer: This connection powers the Arduino and enables data transfer for uploading programs from the Arduino IDE and serial communication.

7. Result-

The Arduino Uno successfully connects to the computer via the USB cable, powers up, and allows for data transfer. The power LED lights up, and programs can be uploaded from the Arduino IDE.



8. Conclusion-

By following the setup procedure, the Arduino Uno is ready for programming and communication with the computer, enabling various projects and experiments. This connection verifies that the board is functioning correctly and is prepared for further use.

Experiment 2

Student Name: **UID:**
Branch: CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT LAB **Subject Code: 22CSP-329**

1. **Aim:** To Design LCD interfacing on WOKWI or Any other IoT Simulation Platform
2. **Objective:**
   - Learn about IoT based simulations.
   - Testing and model in IoT based simulation platform.
3. **Input/Equipment Used:**
   Laptop
   Tinker Cad
   Arduino Uno
   LCD
   Connecting Wire
   Resistor
4. **Theory:**

   Tinkercad - https://www.tinkercad.com is an excellent tool that allows you to simulate Arduino-based systems (and a lot more). You can (perhaps you SHOULD) simulate all exercises and even your own designs before trying them on real hardware. It also allows you to do programming using blocks. You can download / copy-paste the generated code later into Arduino IDE to program the real Arduino board, rather than having to write it from scratch.

5. **Procedure:**

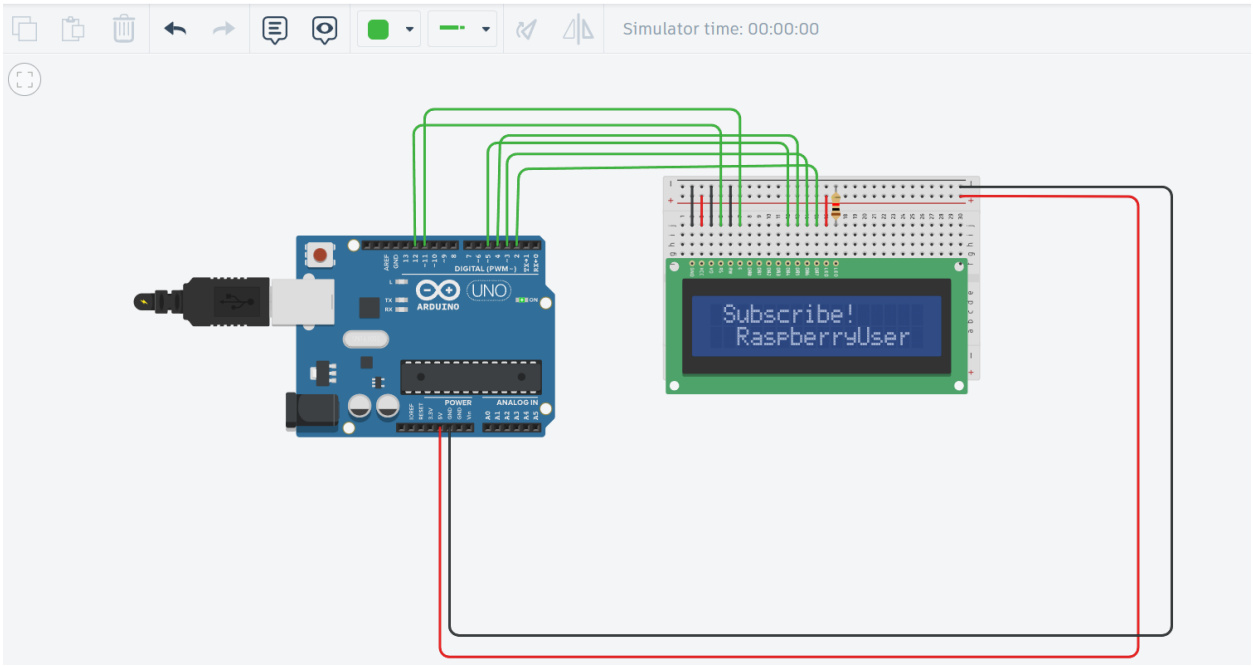
- Go to Tinkercad (tinkercad.com) and log in to your account.
- Click on "Create new circuit" to start a new project.
- Search for components like Arduino boards and LCD displays in the components panel.
- Drag and drop an Arduino board (e.g., Arduino Uno) onto the workspace.
- Search for a 16x2 LCD display and add it to the workspace.
- Connect the components by dragging wires from the output pin of one component to the input pin of another.
- Double-click on the Arduino board to open the code editor.
- Write or paste your Arduino code to initialize the LCD and display text.
- Run the code

6. **Code:**

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
lcd.begin(16, 2);
}
void loop()
{
 lcd.setCursor(0,0);
 lcd.print(" Subscribe!");
```

```
lcd.setCursor(2,1);
lcd.print("RaspberryUser");
}
```

7. Result:



8. Conclusions:

In conclusion, the provided Arduino code effectively demonstrates how to simultaneously scroll two different messages on a 16x2 LCD display using the Liquid Crystal library.

Experiment 3

Student Name: Zatch **UID:**
Branch: CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT **Subject Code: 22CSP-329**

1. **Aim:** To Develop a smart traffic light management system with the help of IOT.
2. **Objective:**
   1. Learn about interfacing.
   2. Learn about IoT programming
3. **Input/Equipment Used:**
   1 × Breadboard
   1 × Arduino Uno R3
   3 × LEDs (Red, Yellow, Green)
   3 × 220Ω Resistor
   3 × Jumper
4. **Theory:**
   Nowadays, everyone prefers a personal vehicle. Hence, the number of vehicles on the road is increasing continuously, which results in traffic jams. Traffic light controller helps to manage the traffic and to maintain proper traffic management. These systems are placed at the intersections of the road or at the crossings to avoid congestions and accidents. The systems indicate to the driver by using different colors of light. Therefore it is simple to avoid congestion at the intersections.

5. Procedure:

- Place Arduino Uno on the breadboard.
- Connect LEDs (Red, Yellow, Green) to digital pins of Arduino via resistors
- Open Arduino IDE and start a new sketch
- Write code to initialize pins for LEDs and sensors, and define traffic light control logic based on sensor inputs
- Connect Arduino Uno to your computer via USB cable
- Compile and upload the code to the Arduino board using Arduino IDE
- Observe the behavior of LEDs (simulating traffic lights) based on sensor inputs or predefined timings
- Verify all wires and components are securely connected

Connections:

- This is the circuit diagram for the traffic light controller by using Arduino.

• Connect LEDs on the breadboard as Red, Yellow, Green, respectively.

• Connect the negative terminal of the LED and connect the 220 Ohm resistor in series.

• Connect these negative terminals to the ground.

• Connect the positive terminal of the LEDs to the pins 2 to 10, respectively.

• Power the breadboard by using 5V and GND on the Arduino

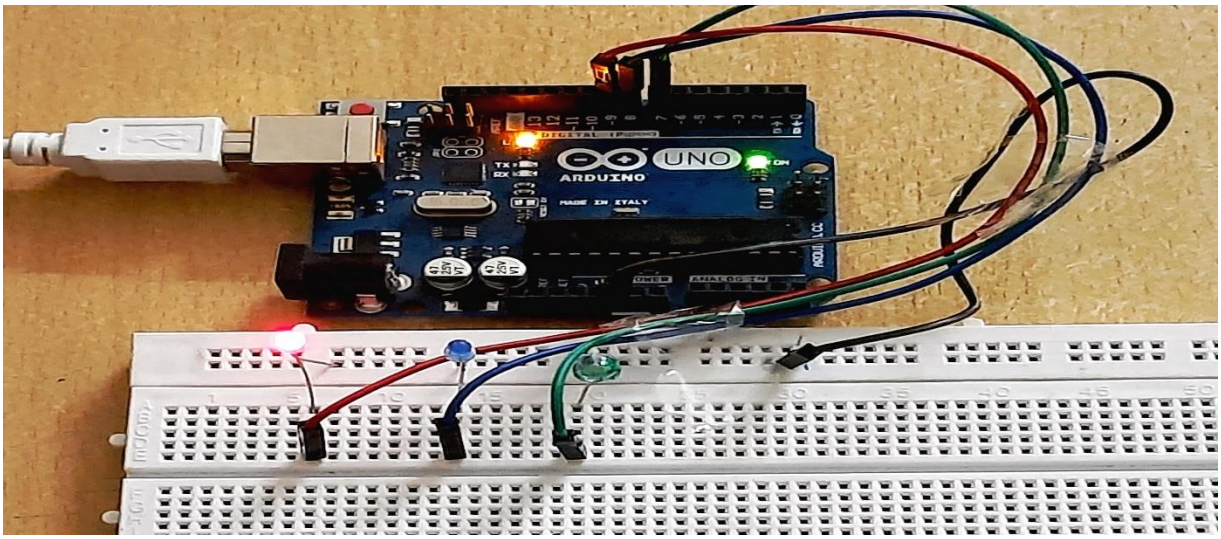
6. Code:

```
void setup() {
// configure the output pins
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
} void loop()
{
digitalWrite(2,1); //enables the 1st set of signals
digitalWrite(7,1);
digitalWrite(10,1);
digitalWrite(4,0);
digitalWrite(3,0);
digitalWrite(6,0);
digitalWrite(8,0);
digitalWrite(9,0);
digitalWrite(5,0);
delay(5000);
```

```
digitalWrite(3,1); //enables the yellow lights
digitalWrite(6,1);
digitalWrite(2,0);
digitalWrite(7,0);
delay(1000);
digitalWrite(4,1); //enables the 2nd set of signals
digitalWrite(5,1);
digitalWrite(10,1);
digitalWrite(2,0);
digitalWrite(3,0);
digitalWrite(6,0);
digitalWrite(8,0);
digitalWrite(9,0);
digitalWrite(7,0);
delay(5000);
digitalWrite(9,1); //enables the yellow lights
digitalWrite(6,1);
digitalWrite(10,0);
digitalWrite(5,0);
digitalWrite(4,0);
delay(1000);
digitalWrite(8,1); //enables the 3rd set of signals
digitalWrite(4,1);
digitalWrite(7,1);
digitalWrite(2,0);
```

```
digitalWrite(3,0);
digitalWrite(5,0);
digitalWrite(6,0);
digitalWrite(9,0);
digitalWrite(10,0);
delay(5000);
digitalWrite(9,1); //enables the yellow lights
digitalWrite(3,1);
digitalWrite(7,0);
digitalWrite(8,0);
digitalWrite(4,0);
delay(1000);
}
```

7. Result:



8. Conclusions:

You should see your LEDs turn on and off. If the required output is not seen, make sure you have assembled the circuit correctly and verified and uploaded the code to your board. This traffic light controller includes a crosswalk signaling system. The traffic light controller in this system can be implemented practically and expanded further.

Experiment 4

Student Name: Zatch **UID:**

Branch: BE-CSE **Section/Group:**

Semester: 5th **Date of Performance:**

Subject Name: IOT LAB **Subject Code: 22CSP-329**

1. **Aim:** To Formulate distance using ultrasonic sensor.

2. **Objective:** The objective of this experiment is to measure the distance to an object using an ultrasonic sensor by emitting sound waves and calculating the time taken for the echo to return.

3. **Input/Equipment Used:**

   a. 1×ArduinoUnoR3
   b. 1×Ultrasonic Sensor (HC-SR04)
   c. 16×2 LCD 12C Display
   d. Jumper Wires

4. **Procedure:**

   i. **Connect Sensor:** Attach VCC, GND, Trig, and Echo pins of the ultrasonic sensor to the microcontroller.
   ii. **Trigger Pulse:** Send a 10 µs pulse to the Trig pin to emit an ultrasonic wave.
   iii. **Measure Echo:** Capture the time between the Echo pin going HIGH and LOW.
   iv. **Calculate Distance:** Use the formula Distance=Time×3432\text{Distance} = \frac{\text{Time} \times 343}{2}Distance=2Time×343 to determine the distance.
   v. **Display Result:** Output the calculated distance to the desired interface.
   vi. **Repeat:** Continuously repeat the measurement for real-time distance monitoring.

5. **Connections:**

Step 1: Power Connections

- Connect the **VCC pin** of the ultrasonic sensor to the **5V pin** on the microcontroller.
- Connect the **GND pin** of the sensor to the **GND pin** on the microcontroller.

Step 2: Trigger Pin Connection

- Connect the **Trig pin** of the sensor to a digital output pin on the microcontroller (e.g., **D2**).

Step 3: Echo Pin Connection

- Connect the **Echo pin** of the sensor to a digital input pin on the microcontroller (e.g., **D3**).

Step 4: Verify Connections

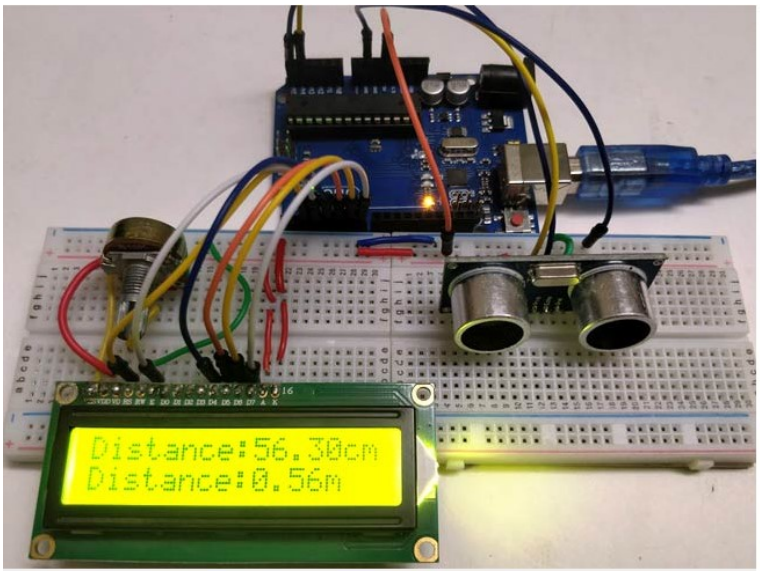
- Double-check all connections to ensure they are secure and correct according to your microcontroller's pin configuration.

6. CODE:

```
#include <LiquidCrystal.h>

#define TRIG_PIN 18  // Trigger pin connected to digital pin 18
#define ECHO_PIN 19  // Echo pin connected to digital pin 19
#define LED_PIN LED_BUILTIN  // Built-in LED

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);  // Initialize the LCD with the appropriate pins

void setup() {
  Serial.begin(115200);
  lcd.begin(16, 2);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);

  lcd.print("Ultra sonic");
  lcd.setCursor(0, 1);
  lcd.print("Distance Meter");
  delay(2000);
  lcd.clear();
  lcd.print(" Circuit Digest");
  delay(2000);
  lcd.clear();
}

float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  float duration = pulseIn(ECHO_PIN, HIGH);
  float distanceCM = duration * 0.034 / 2;
  return distanceCM;
}
```

```
void loop() {
  float distanceCM = readDistanceCM();
  bool isNearby = distanceCM < 100;

  // LED indication
  digitalWrite(LED_PIN, isNearby);

  // Serial Monitor output
  Serial.print("Target Distance: ");
  Serial.print(distanceCM);
  Serial.println(" cm");

  // LCD display output
  lcd.clear();
  lcd.print("Distance: ");
  lcd.print(distanceCM);
  lcd.print(" cm");

  lcd.setCursor(0, 1);
  lcd.print("Distance: ");
  lcd.print(distanceCM / 100);
  lcd.print(" m");

  delay(1000);  // Wait 1 second before the next measurement
  }
```

7. **Result:**



8. **Conclusion:**The experiment effectively demonstrates distance measurement using an ultrasonic sensor. By integrating an LCD display and Serial Monitor output, it provides clear and immediate feedback on distance in both centimeters and meters. The built-in LED serves as a proximity indicator, highlighting when an object is within 100 cm. This setup showcases the ability to measure and display distance in real-time, making it suitable for various applications in monitoring and automation.

Experiment 5

Student Name: Zatch **UID:**
Branch: CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT LAB **Subject Code: 22CSP-329**

1. **Aim:** To Design a weather station by checking Air quality of an environment with the help of IOT.
2. **Objective:**
   - Accurately measure various pollutants and gases in the environment using the MQ135 sensor.
   - Continuously collect data from the air quality sensor and potentially other sensors, and store it for analysis.
3. **Input/Equipment Used:**
   1. Arduino Uno R3
   2. MQ 135 Air Quality Sensor Module
   3. Male to Female Jumper Wire
   4. Software: Arduino IDE
4. **Theory:**

   The MQ135 sensor detects air quality by measuring the concentration of various gases. The Arduino Uno processes this data and can communicate with IoT platforms for remote monitoring. The IoT integration allows data to be sent to a cloud server, where it can be analyzed and visualized.
5. **Procedure:**
   - Connect the MQ135 sensor to the Arduino Uno using male-to-female jumper wires.
   - Install the Arduino IDE on your computer.

- Open the Arduino IDE and write the code to read data from the MQ135 sensor and send it to a serial monitor.
- Connect the Arduino to your computer using a USB cable
- Select the correct board and port in the Arduino IDE. Upload the code to the Arduino.
- Open the Serial Monitor in the Arduino IDE to view real-time data from the MQ135 sensor.

Connections:

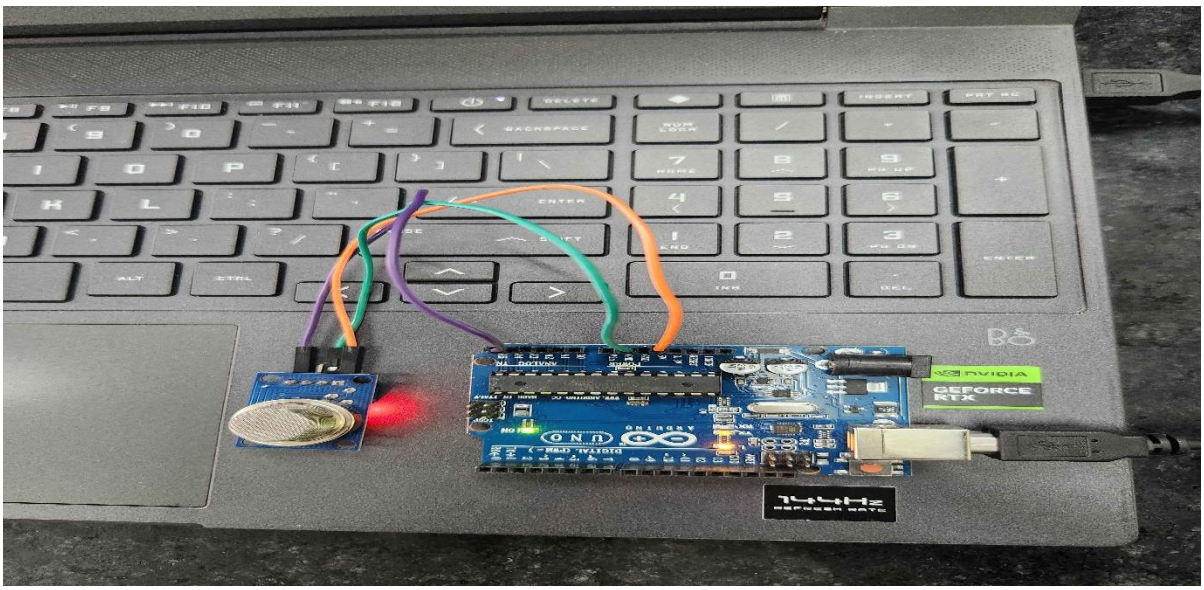
- Connect the VCC pin of the MQ135 to the 5V pin on the Arduino.
- Connect the GND pin of the MQ135 to one of the GND pins on the Arduino.
- Connect the Analog Output (AO) pin of the MQ135 to the Analog pin A0 on the Arduino. This pin will read the analog value from the sensor.
- Connect the Digital Output (DO) pin to a digital pin 2 on the Arduino.

6. **Code:**

```
int sensorValue;
int digitalValue;
void setup()
{
 Serial.begin(9600); // sets the serial port to 9600
 pinMode(13, OUTPUT);
 pinMode(2, INPUT);
}
void loop()
{
```

```
sensorValue = analogRead(0); // read analog input pin 0
digitalValue = digitalRead(2);
if (sensorValue > 400)
{
digitalWrite(13, HIGH);
}
else
 digitalWrite(13, LOW);
Serial.println(sensorValue, DEC); // prints the value read
Serial.println(digitalValue, DEC);
delay(1000); // wait 100ms for next reading
}
```

7. **Result:**



8. Conclusions:

The setup successfully demonstrates how to monitor air quality using an Arduino Uno and an MQ135 sensor. Integrating IoT capabilities can further enhance the system by allowing remote monitoring and data analysis.

Experiment 6

Student Name: Zatch **UID:**
Branch: CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT LAB **Subject Code: 22CSP-329**

1. **Aim:** To investigate the real-time relationship between humidity and temperature in an IoT system.
2. **Objective:**
   - To interface the DHT11 sensor with an Arduino board.
   - To display the temperature and humidity data on the serial monitor.
   - To understand how temperature and humidity are correlated in real-time.
3. **Input/Equipment Used:**
   1. Arduino Board
   2. Breadboard
   3. Jumper Wires
   4. DH11 Temperature and Humidity Sensor
4. **Theory:**

   DHT11 Module features a temperature & humidity sensor complex with a calibrated digital signal output. The exclusive digital-signal-acquisition technique and temperature & humidity sensing technology ensure high reliability and excellent long-term stability. This sensor includes an NTC for temperature measurement and a resistive-type humidity measurement component for humidity measurement. These are connected to a high-performance 8-bit microcontroller, offering excellent quality, fast response, antiinterference ability, and cost-effectiveness.

5. Procedure:

- Assemble the circuit and connect to required pin.
- Install the Arduino IDE on your computer.
- Open the Arduino IDE
- Go to Sketch > Include Library > Manage Libraries.
- Search for Adafruit DHT sensor library and Adafruit Unified Sensor Driver.
- Install both libraries.
- Open a new sketch in Arduino IDE and write the code.
- Connect the Arduino to your computer using a USB cable
- Select the correct board and port in the Arduino IDE. Upload the code to the Arduino.
- Once the code is uploaded, open the Serial Monitor from the Arduino IDE (Tools > Serial Monitor).
- The real-time temperature and humidity data will be displayed.

Connections:

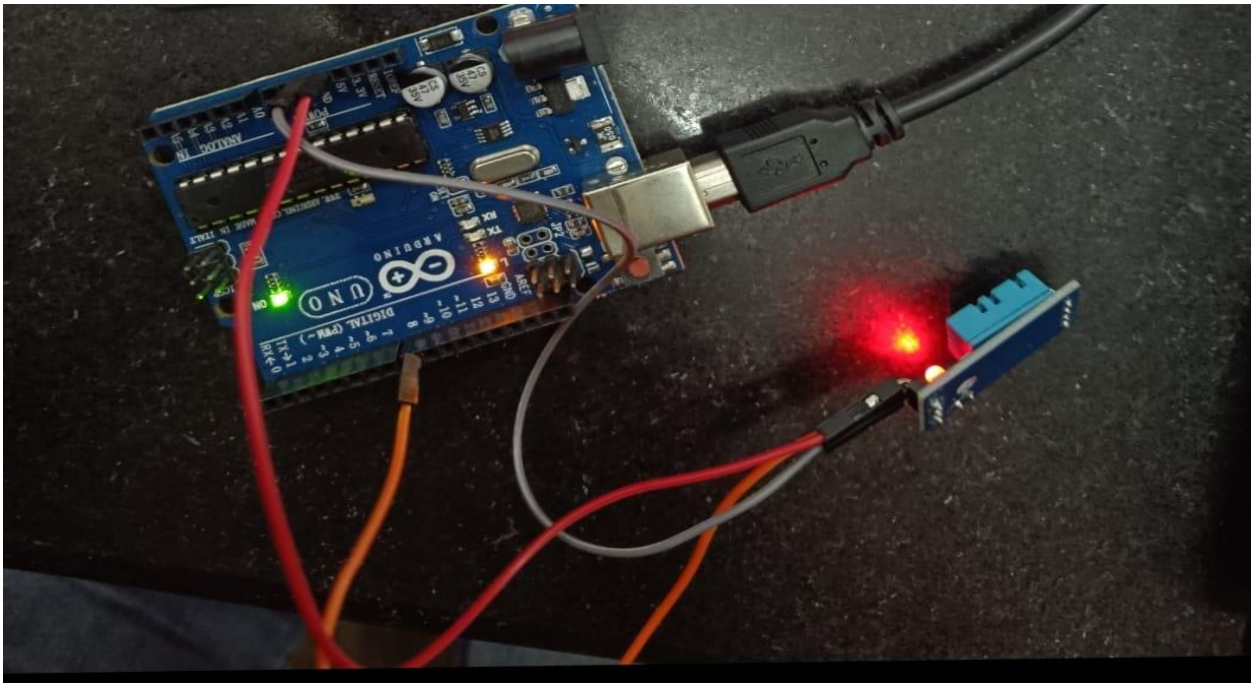
- Connect the VCC pin of the DHT11 sensor to the 5V pin of the Arduino.
- Connect the GND pin of the DHT11 to the GND pin of the Arduino.
- Connect the DATA pin of the DHT11 sensor to the Digital Pin 2 of the Arduino.
- add a 10k ohm pull-up resistor between the VCC and DATA pin for signal stability.

6. **Code:**

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTTYPE DHT11
#define DHTPIN 2

DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;

void setup() {

  Serial.begin(9600);

  dht.begin();

  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  delayMS = sensor.min_delay / 1000;
}

void loop() {
  sensors_event_t event;
```

```
 dht.temperature().getEvent(&event);
 Serial.print(F("Temperature: "));
 Serial.print(event.temperature);
 Serial.println(F("°C"));
 dht.humidity().getEvent(&event);
 Serial.print(F("Humidity: "));
 Serial.print(event.relative_humidity);
 Serial.println(F("%"));
 delay(delayMS);
}
```

7. Result:



8. Conclusions:

By using the DHT11 sensor and Arduino, we successfully captured real-time temperature and humidity data. The relationship between temperature and humidity can be analyzed through the serial output. This project demonstrates how IoT devices can monitor environmental conditions in real time for various applications such as weather stations, smart homes, and agricultural monitoring.

Experiment 7

Student Name: Zatch **UID:**
Branch: CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT LAB **Subject Code: 22CSP-329**

1. **Aim:**

   To Assemble and Controlling of actuators using Arduino Uno.

2. **Objective:**

   The goal of this project is to assemble and control various actuators (LED, motor, and buzzer) using an Arduino Uno. This project will involve connecting the components to the Arduino and writing a program to control their behavior, such as blinking the LED, turning the motor on and off, and sounding the buzzer.

3. **Input/Equipment Used:**

   1. 1x Arduino
   2. 1x LED
   3. 1x Motor
   4. 1x Buzzer

4. **Theory:**

   Servo Motors: Servomotors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the board. The signal pin is typically yellow or orange and should be connected to PWM pin on the board.

5. Procedure:

- Place the Arduino Uno on a working surface or connect it to a breadboard for easier wiring
- Connect the LED to the breadboard. Attach the longer leg (anode) of the LED to pin 8 of the Arduino and the shorter leg (cathode) to ground (GND) through a 220 ohm resistor.
- If using a motor driver (L298N or similar), connect the control pins of the driver to pin 9 of the Arduino, the power pins to a 5V external power supply, and the motor output pins to the motor.
- Connect the Buzzer to pin 10 of the Arduino, with the other terminal connected to ground.
- Install the Arduino IDE on your computer.
- Open the Arduino IDE
- Open a new sketch in Arduino IDE and write the code.
- Connect the Arduino to your computer using a USB cable
- Select the correct board and port in the Arduino IDE. Upload the code to the Arduino.
- Once the code is uploaded, the LED will blink, the motor will spin, and the buzzer will sound simultaneously for 1 second and then turn off for 1 second in a loop.

Connections:

- Connect the LED anode to pin 8 and cathode to GND
- Connect motor driver: Motor + and - terminals connected to the motor, and input pins connected to Arduino Pin 9.

- Connect one terminal of Buzzer to pin 10 and other to ground GND

6. Code:

```
const int ledPin = 8;       // Pin for LED
const int motorPin = 9;     // Pin for Motor
const int buzzerPin = 10;   // Pin for Buzzer

void setup() {

  pinMode(ledPin, OUTPUT);
  pinMode(motorPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // Turn on LED
  digitalWrite(ledPin, HIGH);
  // Turn on Motor
  digitalWrite(motorPin, HIGH);
  // Turn on Buzzer
  digitalWrite(buzzerPin, HIGH);
  delay(1000);  // Wait for 1 second
  // Turn off LED
  digitalWrite(ledPin, LOW);
  // Turn off Motor
  digitalWrite(motorPin, LOW);
```

```
  // Turn off Buzzer
  digitalWrite(buzzerPin, LOW);
  delay(1000);  // Wait for 1 second
}
```

7. Result:

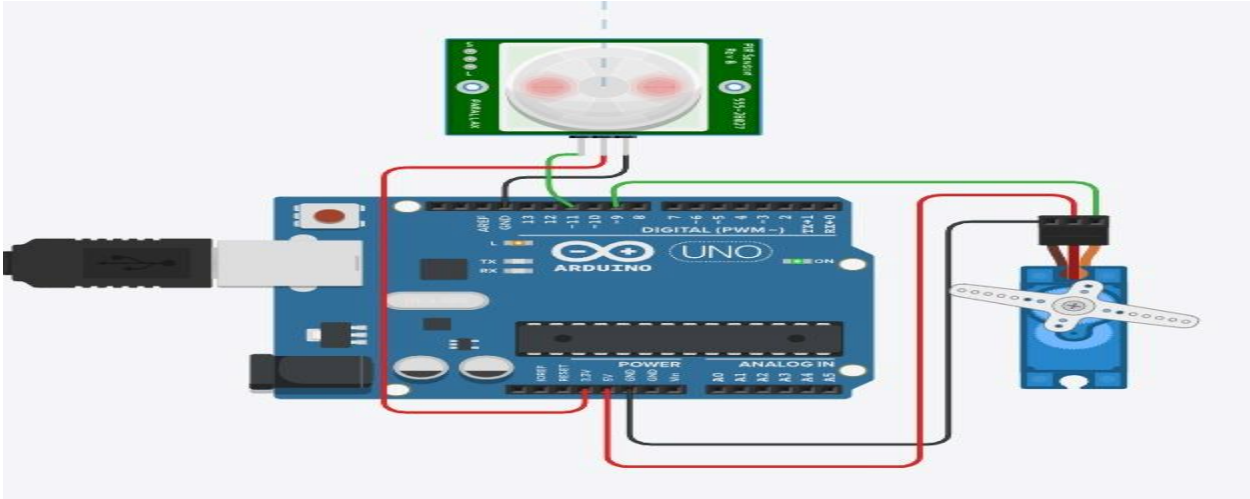


Figure 1: Output

8. Conclusions:

we successfully controlled multiple actuators (LED, motor, and buzzer) using an Arduino Uno. By setting the appropriate pin modes and controlling the output signals with code, we were able to blink the LED, control the motor, and sound the buzzer in a synchronized pattern. This experiment demonstrates how Arduino can be used to control various electronic components in automation or robotics applications.

Experiment 8

Student Name: Zatch **UID:**
Branch: BE-CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT Lab **Subject Code: 22CSP-329**

1. Aim:

To Create a Smart door lock system-using RFID.

2. Objective:

- Prevent unauthorized access by only allowing registered RFID tags.
- Logs access history for auditing who enters and exits.
- Deactivate lost or stolen RFID tags to prevent misuse.
- Keyless entry using RFID cards, tags, or smartphones with NFC.
- Automatic unlocking when an authorized RFID tag is detected.
- Simplified access management—easy to add or remove users.

3. Equipment Used:

- Arduino Uno R3 board
- RFID RC522
- SG-90 Servo Motor
- 16×2 LCD

4. Procedure:

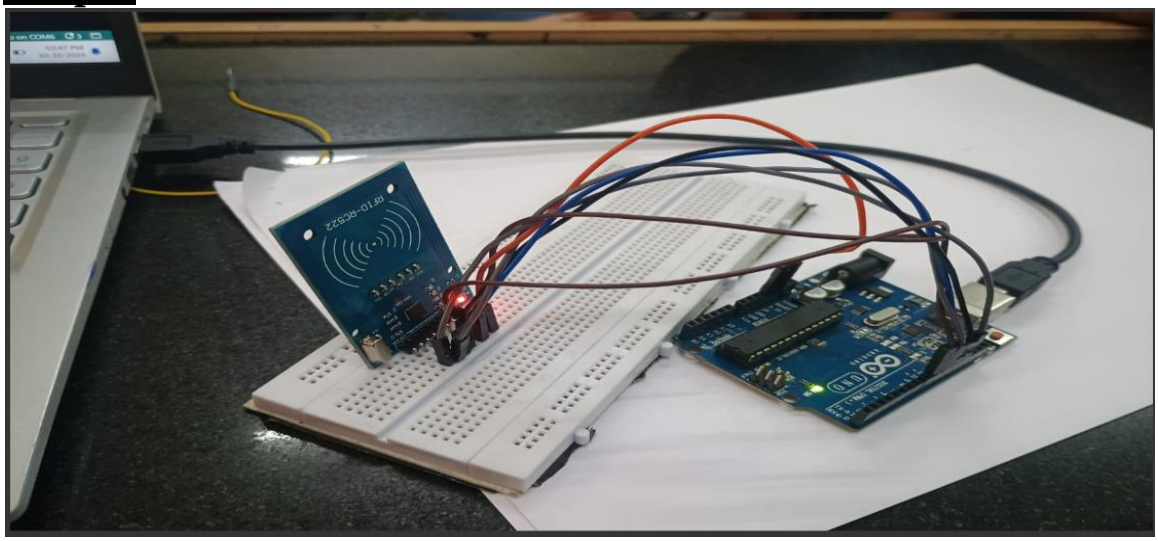
- D0 – D7: Pin number 7-14 are data bus lines that are used to send data from Arduino which you want to display on LCD. With these 8 data lines, data can be transferred either in an 8-bit format or in a 4-bit format.
- Contrast Select (VEE): It will help to control the contrast of PIXELS according to the 16X2
- LCD light.

- RS: This pin is known as a register select pin. It helps to toggle the command/data register.
- R/W: The signal on this pin will decide whether it is going to read from LCD or write on it.
- EN: Enable pin will help to transfer the instruction from the data pins and another command pin
- to the LCD. It acts as permission to internal registers.
- VSS: It's a ground pin for common grounds.
- VCC: The power pin will use for voltage input to the 16X2 LCD.

5. Code:

```
//Arduino Code - RC522 Read RFID Tag UID
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 7
MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class
MFRC522::MIFARE_Key key;
void setup() {
  Serial.begin(9600);
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init RC522
}
void loop() {
  // Reset the loop if no new card present on the sensor/reader. This saves the entire process
when idle.
  if ( ! rfid.PICC_IsNewCardPresent())
    return;
  // Verify if the NUID has been readed
  if ( ! rfid.PICC_ReadCardSerial())
  return;
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  Serial.print(F("RFID Tag UID:"));
  printHex(rfid.uid.uidByte, rfid.uid.size);
  Serial.println("");
  rfid.PICC_HaltA(); // Halt PICC
}
//Routine to dump a byte array as hex values to Serial.
void printHex(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}
```

7. Output:



```
    // Dump debug info about the card; PICC_HaltA() is automatically called
    mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}

void ShowReaderDetails() {
    // Get the MFRC522 software version
    byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
    Serial.print(F("MFRC522 Software Version: 0x"));
    Serial.print(v, HEX);
    if (v == 0x91)
        Serial.print(F(" = v1.0"));
    else if (v == 0x92)
        Serial.print(F(" = v2.0"));
    else
        Serial.print(F(" (unknown)"));
    Serial.println("");
    // When 0x00 or 0xFF is returned, communication probably failed
    if ((v == 0x00) || (v == 0xFF)) {
        Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));
    }
}
```

8. Learning Outcome:

- Learn the principles of RFID operation, including how RFID tags and readers communicate.
- Understand the differences between active and passive RFID systems.
- Gain experience in working with microcontrollers (such as Arduino, Raspberry Pi) to control the RFID system.
- Learn how to interface RFID readers with microcontrollers for real-time access control.
- Learn how to design and build circuits that integrate RFID readers, door locks (e.g., electronic solenoids), and other components like buzzers or LEDs.

Experiment 9

Student Name: Zatch **UID:**
Branch: CSE **Section/Group:**
Semester: 5th **Date of Performance:**
Subject Name: IOT LAB **Subject Code: 22CSP-329**

1. **Aim:**

   Case studies of IoT in Healthcare and propose any IoT model for the healthcare sector

2. **Objective:**

   The goal of this project is to assemble and control various actuators (LED, motor, and buzzer) using an Arduino Uno. This project will involve connecting the components to the Arduino and writing a program to control their behavior, such as blinking the LED, turning the motor on and off, and sounding the buzzer.

3. **Input/Equipment Used:**

   1.Arduino Uno board

   2. Soil moisture sensor

   3. Breadboard

   4. Jumper wires

   5. USB cable for Arduino

   6. Computer with Arduino IDE installed

4. **Theory:**

   Soil moisture sensors measure the volumetric water content in soil. They typically consist of two probes that allow the current to pass through the

soil. More water in the soil results in lower resistance and higher conductivity between the probes.

5. Procedure:

- Setup the Arduino and Sensor
- Write the Arduino Code
- Upload the Code: Connect the Arduino to your computer using the USB cable and upload the code to the Arduino board.
- Open Serial Monitor: After uploading the code, open the Serial Monitor from the Arduino IDE (Tools > Serial Monitor) to see the sensor readings.
- Take Readings: Observe the sensor values as you place the sensor in different soil conditions: Dry Soil: Note the sensor value in dry soil.
- Wet Soil: Note the sensor value in wet soil. Record Data: Record the sensor values for different soil moisture levels.
- Interpret Sensor Values: Higher sensor values typically indicate lower moisture levels (dry soil), while lower sensor values indicate higher moisture levels (wet soil).

Connections:

- Connect the VCC pin of the soil moisture sensor to the 5V pin on the Arduino.
- Connect the GND pin of the soil moisture sensor to a GND pin on the Arduino.

- Connect the A0 pin of the soil moisture sensor to an analog input pin (e.g., A0) on the Arduino.

6. **Code:**

```
// Define the pin connected to the soil moisture sensor
const int sensorPin = A0;
void setup()
{
// Initialize serial communication at 9600 bits per second
Serial.begin(9600);
}
void loop()
{
// Read the value from the soil moisture sensor
int sensorValue = analogRead(sensorPin);
// Print the sensor value to the serial monitor
Serial.print("Soil Moisture Value: ");
Serial.println(sensorValue);
// Wait for a second before taking another reading
delay(1000);
}
```

7. **Result:**

Output lga lena

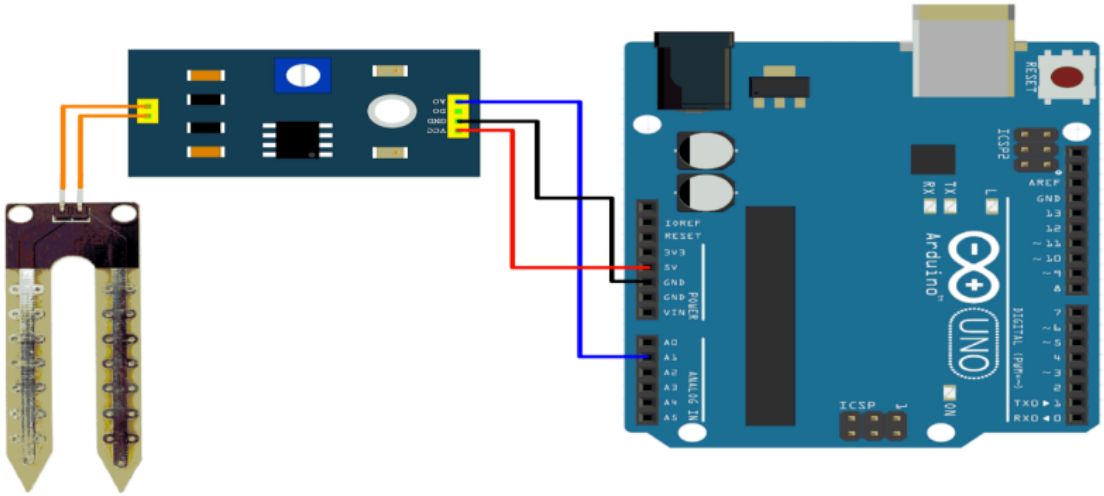


Figure 1: Output

8. Conclusions:

By completing this experiment, you have successfully interfaced a soil moisture sensor with an Arduino board, measured soil moisture levels, and interpreted the sensor readings. This knowledge can be applied to automate irrigation systems in agriculture.