

1. Write a Embedded C Program to Create a Weather Reporting System that provides real- time environmental data to users.

Code:

```
#include <DHT.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {

    dht.begin();

    lcd.init();

    lcd.backlight();

    lcd.setCursor(0, 0);

    lcd.print("Weather Report");

}

void loop() {

    float temp = dht.readTemperature();

    float hum = dht.readHumidity();

    if (isnan(temp) || isnan(hum)) {

        lcd.setCursor(0, 1);

        lcd.print("Error Reading");

        return;

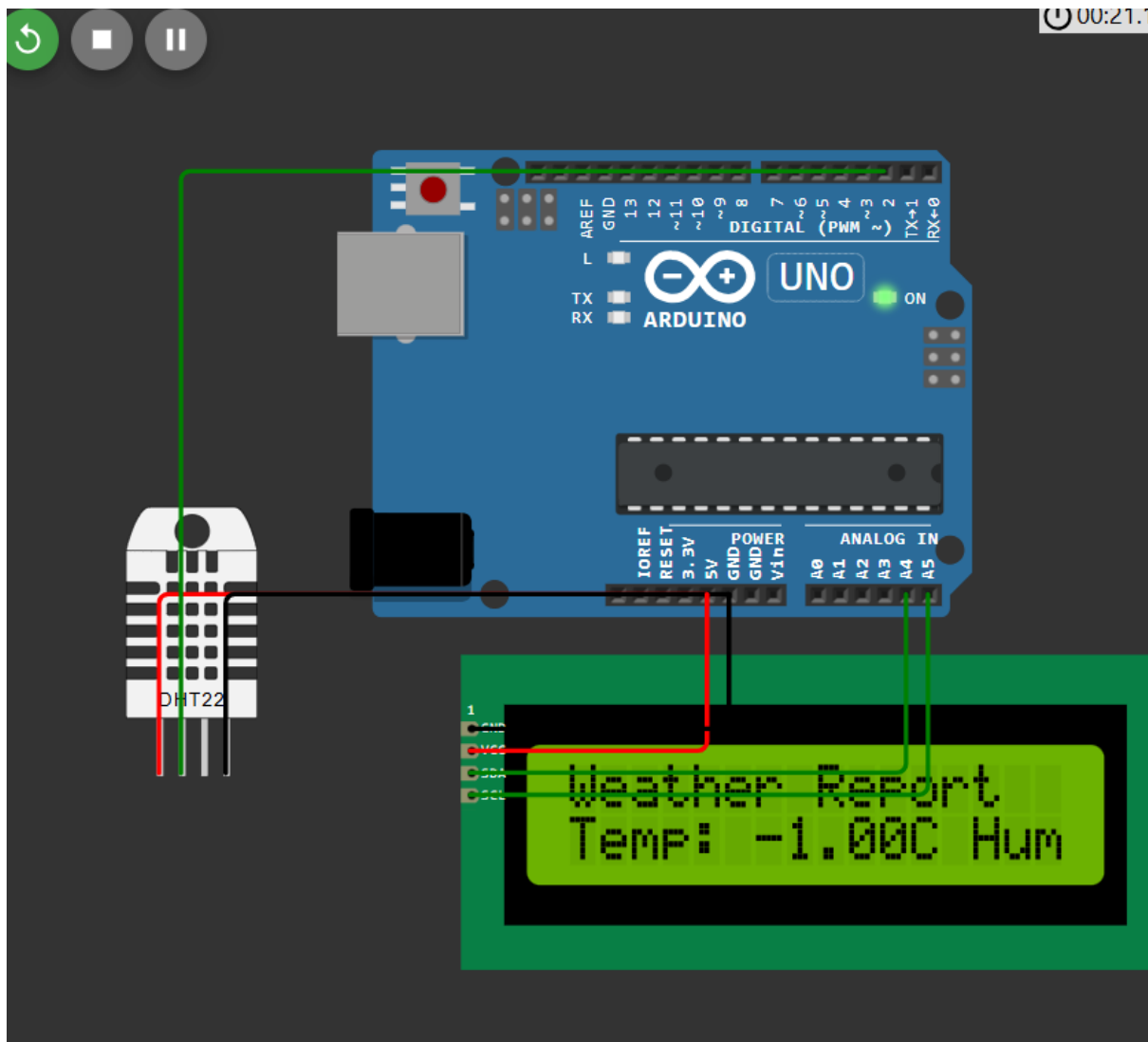
    }
```

```

lcd.setCursor(0, 1);
lcd.print("Temp: ");
lcd.print(temp);
lcd.print("C ");
lcd.print("Hum: ");
lcd.print(hum);
lcd.print("%");
delay(2000);
}

```

OP:



2. Write a Embedded C Program to Create a Home Automation System that simplifies daily routines (Any 2 Devices) by controlling devices remotely.

Code:

```
#define LED1 2
#define LED2 3

void setup() {
    // Initialize the LEDs as outputs
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);

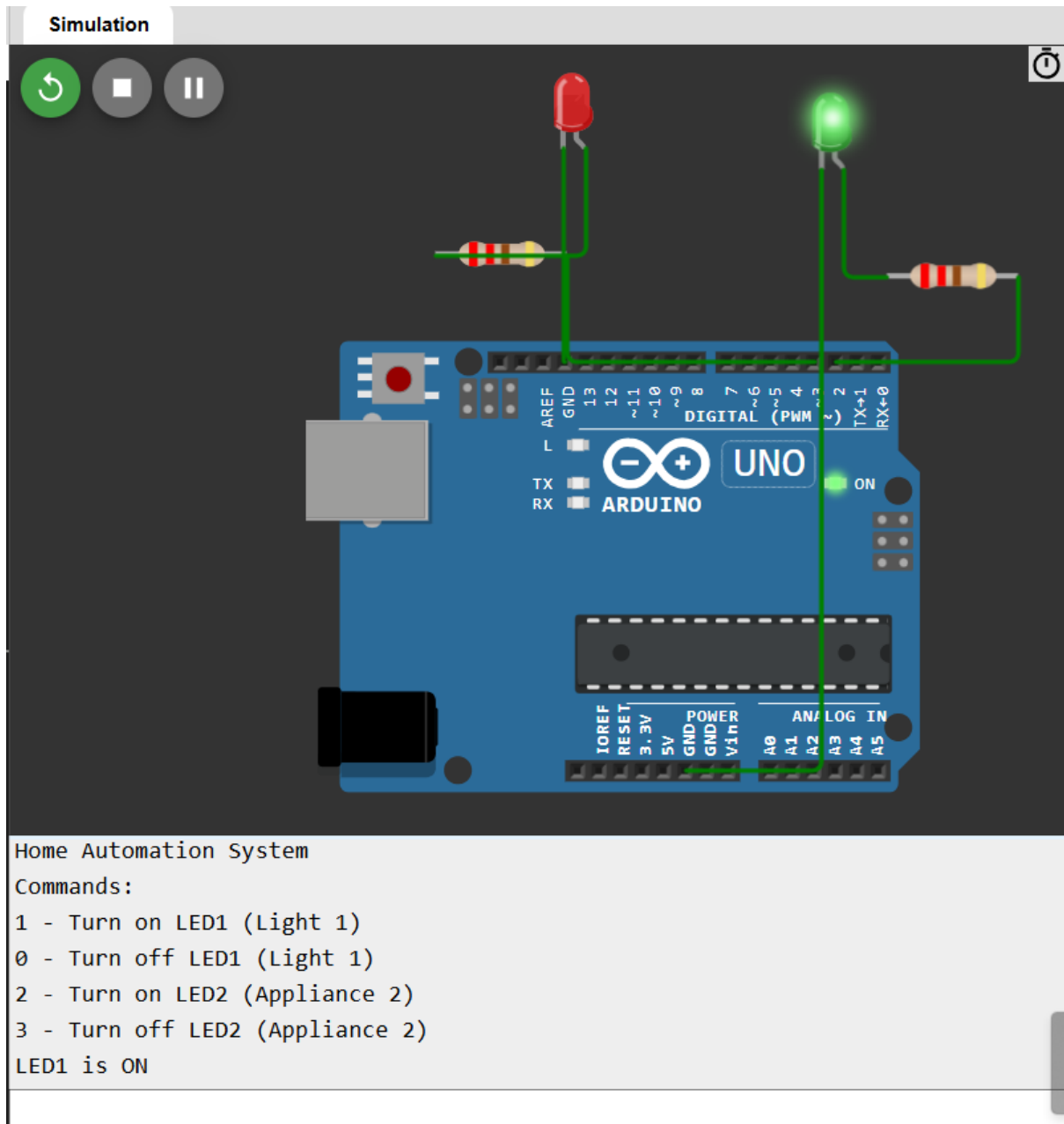
    // Start serial communication
    Serial.begin(9600);
    Serial.println("Home Automation System");
    Serial.println("Commands: ");
    Serial.println("1 - Turn on LED1 (Light 1)");
    Serial.println("0 - Turn off LED1 (Light 1)");
    Serial.println("2 - Turn on LED2 (Appliance 2)");
    Serial.println("3 - Turn off LED2 (Appliance 2)");
}

void loop() {
    // Check if data is available on Serial
    if (Serial.available()) {
        char command = Serial.read(); // Read the incoming command

        // Control LED1 (Light 1)
        if (command == '1') {
            digitalWrite(LED1, HIGH); // Turn on LED1
```

```
    Serial.println("LED1 is ON");
}
if (command == '0') {
    digitalWrite(LED1, LOW); // Turn off LED1
    Serial.println("LED1 is OFF");
}

// Control LED2 (Appliance 2)
if (command == '2') {
    digitalWrite(LED2, HIGH); // Turn on LED2
    Serial.println("LED2 is ON");
}
if (command == '3') {
    digitalWrite(LED2, LOW); // Turn off LED2
    Serial.println("LED2 is OFF");
}
}
}
Op:
```



3. Write a Embedded C Program to Create an Air Pollution Monitoring System that tracks air quality levels in real-time to ensure a healthier environment.

Code:

```
#include <Wire.h>

#include <Adafruit_SSD1306.h>

#include <Adafruit_GFX.h>
```

```

#define SSD1306_I2C_ADDRESS 0x3C // I2C address for OLED display

#define POT_PIN A0 // Analog pin for potentiometer
#define BUZZER_PIN 8
#define LED_PIN 9

// OLED settings
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1 // No reset pin needed

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

void setup() {
    Serial.begin(115200);

    // Set up Buzzer and LED pins
    pinMode(BUZZER_PIN, OUTPUT);
    pinMode(LED_PIN, OUTPUT);

    // Initialize OLED
    if(!display.begin(SSD1306_I2C_ADDRESS, OLED_RESET)) {
        Serial.println(F("OLED allocation failed"));
        for (;;)
    }

    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE);
    display.setTextSize(2); // Increase text size for better visibility
    display.setCursor(0, 0);
    display.print("Air Pollution Monitor");

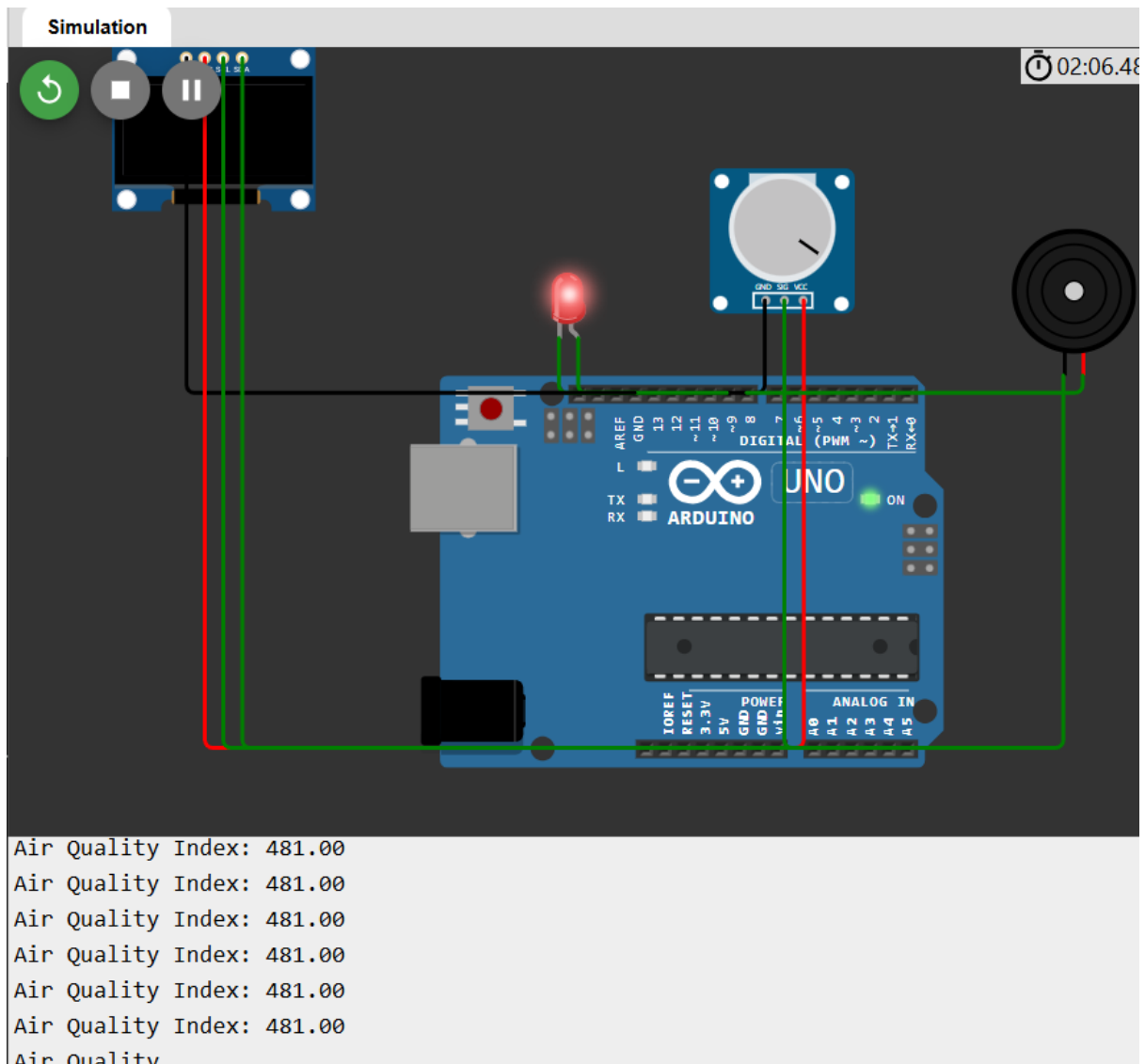
```

```
display.display();  
delay(2000);  
}  
  
void loop() {  
  int sensorValue = analogRead(POT_PIN);  
  float airQualityIndex = map(sensorValue, 0, 1023, 0, 500);  
  
  Serial.print("Air Quality Index: ");  
  Serial.println(airQualityIndex);  
  
  display.clearDisplay();  
  display.setCursor(0, 0);  
  display.print("Air Quality Index:");  
  display.setCursor(0, 20);  
  display.print(airQualityIndex);  
  display.print(" ppm");  
  
  if (airQualityIndex > 300) {  
    display.setCursor(0, 40);  
    display.print("Warning: Poor Air Quality!");  
    digitalWrite(BUZZER_PIN, HIGH);  
    digitalWrite(LED_PIN, HIGH);  
  } else {  
    display.setCursor(0, 40);  
    display.print("Air Quality is Good");  
    digitalWrite(BUZZER_PIN, LOW);  
    digitalWrite(LED_PIN, LOW);  
  }  
}
```

```

display.display();
delay(1000);
}

```



4. Write a Embedded C Program to Create an IoT-based Smart Irrigation System for Agriculture that automates watering based on weather and soil conditions

Code:

```
#include <DHT.h> // Include the DHT sensor library
```

```
// Define pins
```



```
#define SOIL_MOISTURE_PIN A0 // Analog pin for soil moisture sensor
(Potentiometer)

#define DHT_PIN 2 // Digital pin for DHT11 sensor (simulated)

#define RELAY_PIN 1 // Digital pin for relay (water pump)


// DHT sensor setup
DHT dht(DHT_PIN, DHT11); // DHT11 sensor on the specified pin


// Variables
int soilMoistureValue = 0;

float temperature = 30.0; // Simulate temperature of 30°C
float humidity = 0.0;
bool isWateringRequired = false;


void setup() {
    Serial.begin(115200);
    pinMode(RELAY_PIN, OUTPUT);
    digitalWrite(RELAY_PIN, LOW); // Ensure relay is off at startup


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


void loop() {
    // Read soil moisture (Potentiometer value)
    soilMoistureValue = analogRead(SOIL_MOISTURE_PIN);

    Serial.print("Soil Moisture: ");
    Serial.println(soilMoistureValue);


    // Simulate temperature (30°C)
    temperature = 35.0; // Manually set temperature to 30°C for testing
```

```

// Print simulated temperature and humidity
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.print(" °C | Humidity: ");
humidity = dht.readHumidity(); // Read humidity from DHT11
Serial.print(humidity);
Serial.println(" %");

// Logic for automatic irrigation: if soil is dry and temperature is high, water the plants
if (soilMoistureValue < 400 && temperature > 30.0) {
    isWateringRequired = true;
} else {
    isWateringRequired = false;
}

// Control water pump (Relay)
if (isWateringRequired) {
    Serial.println("Watering plants...");
    digitalWrite(RELAY_PIN, HIGH); // Turn on water pump
} else {
    Serial.println("No need to water.");
    digitalWrite(RELAY_PIN, LOW); // Turn off water pump
}

delay(5000); // Wait before next reading
}

```

Op:

Simulation

```

Watering plants...
Soil Moisture: 0
Temperature: 35.00 °C | Humidity: 15.40 %
Watering plants...
Soil Moisture: 0
Temperature: 35.00 °C | Humidity: 15.40 %
Watering plants...

```

- Write a Embedded C Program to Create a Smart Alarm Clock that adjusts to your schedule and environment, waking you up intelligently.

Code:

```
#define BUZZER_PIN 8    // Digital pin for buzzer
```

```
#define LED_PIN 9      // Digital pin for LED
```

```
int airQualityIndex = 0; // Default value of air quality index
```

```

void setup() {
    Serial.begin(115200); // Start serial communication for debugging

    // Set up Buzzer and LED pins
    pinMode(BUZZER_PIN, OUTPUT);
    pinMode(LED_PIN, OUTPUT);

    // Print initial message to Serial Monitor
    Serial.println("Air Pollution Monitoring System Initialized");
    Serial.println("Enter Air Quality Index (0-500): ");
}

void loop() {
    // Check if data is available in Serial Monitor
    if (Serial.available() > 0) {
        // Read the entered value
        airQualityIndex = Serial.parseInt();

        // Ensure that air quality index stays within the range (0 - 500)
        if (airQualityIndex < 0) airQualityIndex = 0;
        if (airQualityIndex > 500) airQualityIndex = 500;

        // Print the entered air quality index to the Serial Monitor
        Serial.print("Air Quality Index: ");
        Serial.print(airQualityIndex);
        Serial.println(" ppm");
    }

    // Logic to determine if air quality is good or poor
    if (airQualityIndex > 300) {

```

```

Serial.println("Warning: Poor Air Quality!");

digitalWrite(BUZZER_PIN, HIGH); // Turn on the buzzer

digitalWrite(LED_PIN, HIGH); // Turn on the LED
} else {
Serial.println("Air Quality is Good");

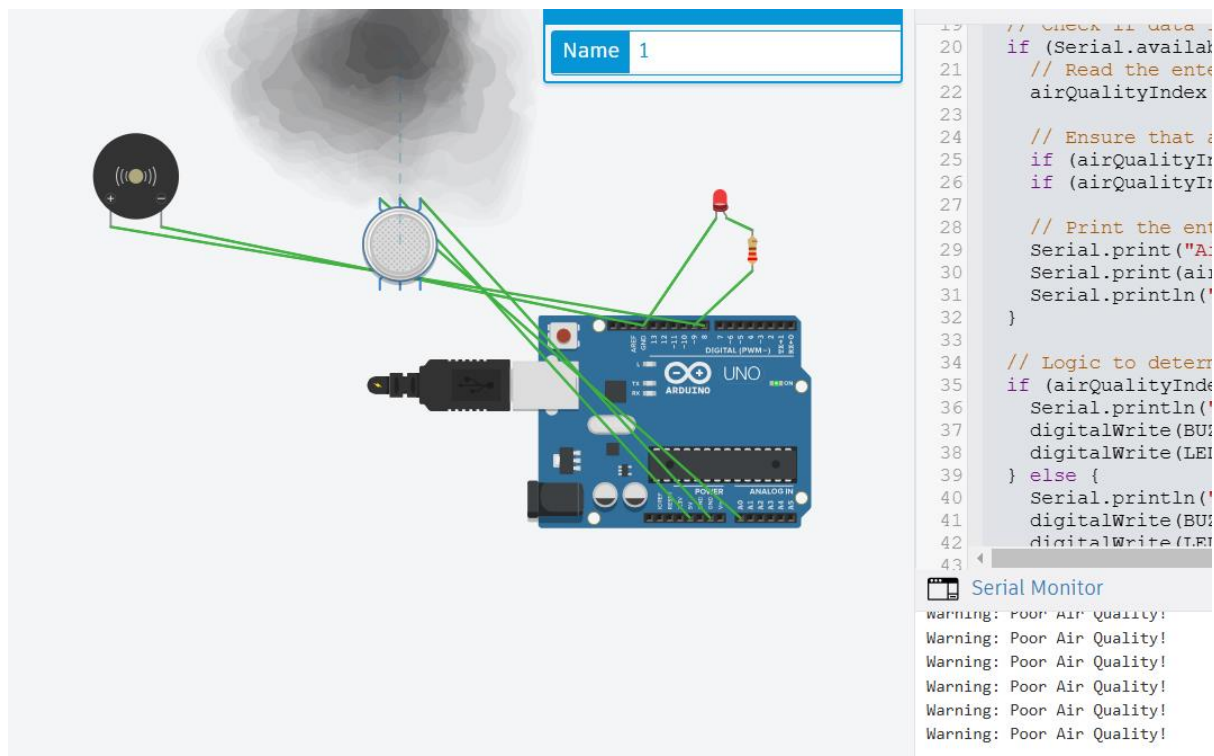
digitalWrite(BUZZER_PIN, LOW); // Turn off the buzzer

digitalWrite(LED_PIN, LOW); // Turn off the LED
}

delay(1000); // Wait for 1 second before checking again
}

```

Op:



G.CHARITHA

2211CS020162

AIML-BETA

