Assignment 1:

Traffic Signal (basics of Arduino, LED, Resistor)

Using the data from the sensors, it will decide how dense the traffic is in each lane, and based on that, it will controlthe traffic signals, which will then take advantage of any traffic signals. LEDs were used in the creation of the system's traffic signals. Red, Yellow and green LEDs are present on each signal.

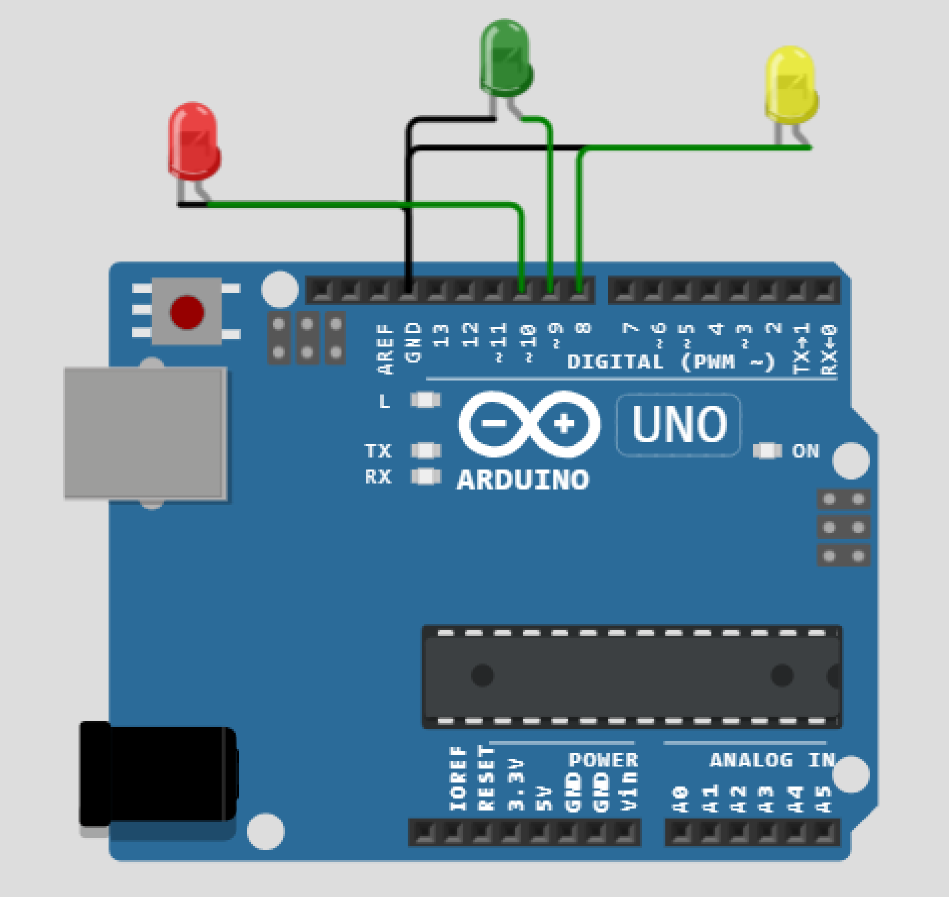
Software Used:

WOKWI

Components Used

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
LED	3	LED (Green), LED (Yellow) and LED (Red)
Resistor	3	1 Resistor - LED (Green), 1 Resistor - LED (Yellow) and 1 Resistor - LED (Red)

Circuit Diagram



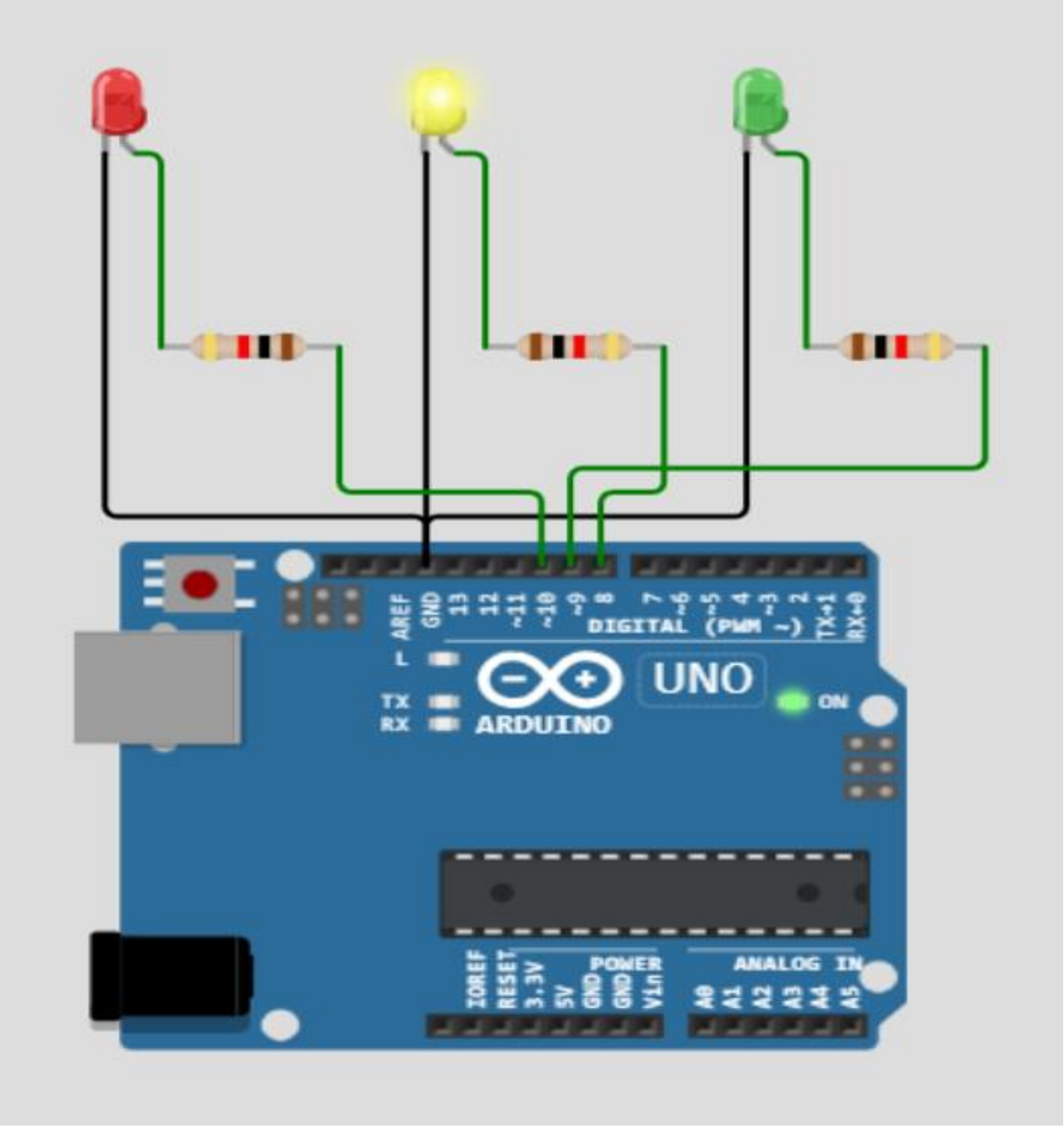
Component Connection

Name	Quantity	Pin	Connection
LED (Red)	1	Cathode (negative pin)	Ground
		Anode (positive pin)	A10
LED (Green)	1	Cathode (negative pin)	Ground
		Anode (positive pin)	A9
LED (Yellow)	1	Cathode (negative pin)	Ground
		Anode (positive pin)	A8

Code

```
// Define the pins to which the traffic signal LEDs are connected
const int redPin = 10;
const int greenPin = 9;   // Green LED connected to pin 9
const int yellowPin = 8;  // Yellow LED connected to pin 8

// Define the duration of each phase in milliseconds
const int redDuration = 5000;     // 5000 milliseconds (5 seconds)
const int yellowDuration = 2000;  // 2000 milliseconds (2 seconds)
const int greenDuration = 5000;   // 5000 milliseconds (5 seconds)


void setup() {
   // Set the LED pins as outputs
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop() {

// Red phase
  digitalWrite(redPin, HIGH);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, LOW);
  delay(redDuration);

  // Yellow phase
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, HIGH);
  digitalWrite(greenPin, LOW);
  delay(yellowDuration);
  // Green phase
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, HIGH);
  delay(greenDuration);

}
```

OUTPUT



Assignment 2:
Visitors count using PIR motion sensor.

It counts visitors in a room, as they enter through a narrow corridor containing two PIRs, passing from PIR1 toPIR2 and increase/decreases the people count when enter/exit. It provides output using LCD.

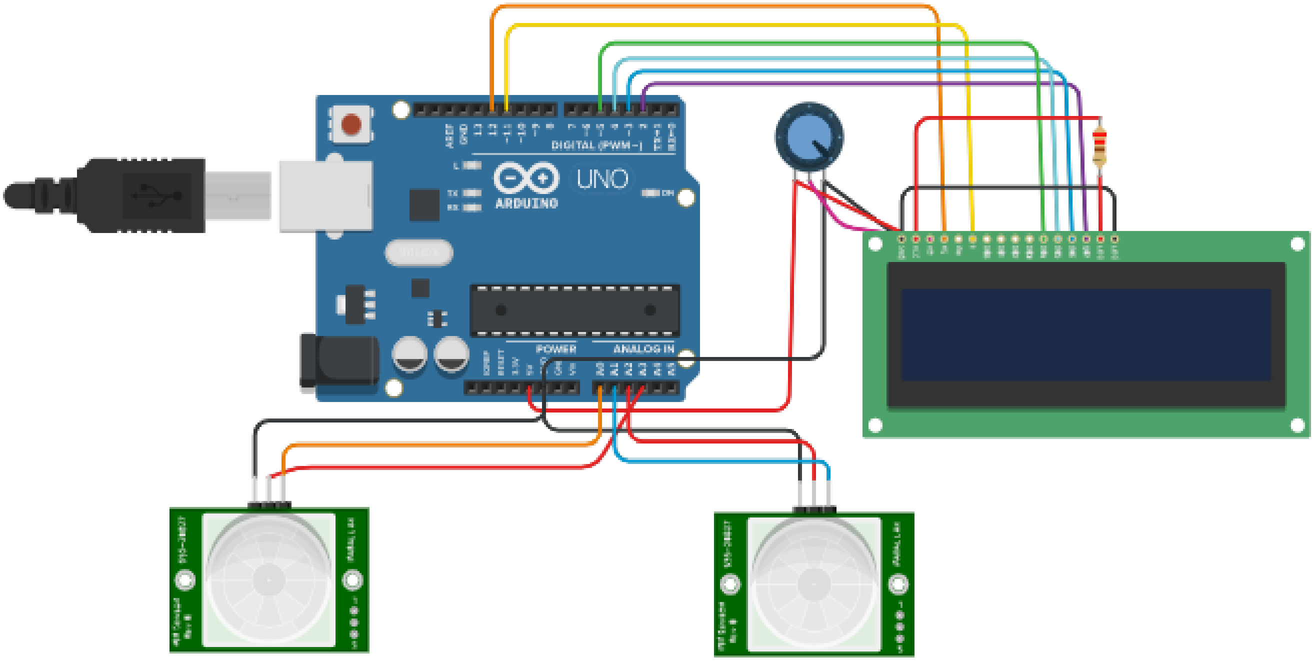
Software Used:

TINKERCAD

Components

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
PIR1	1	Counting People In
PIR2	1	Counting People Out
LCD 16 x 2	1	To display the people count
250 kΩ Potentiometer	1	Internal Resistor
220 Ω Resistor	1	To regulate the voltage

Circuit Design



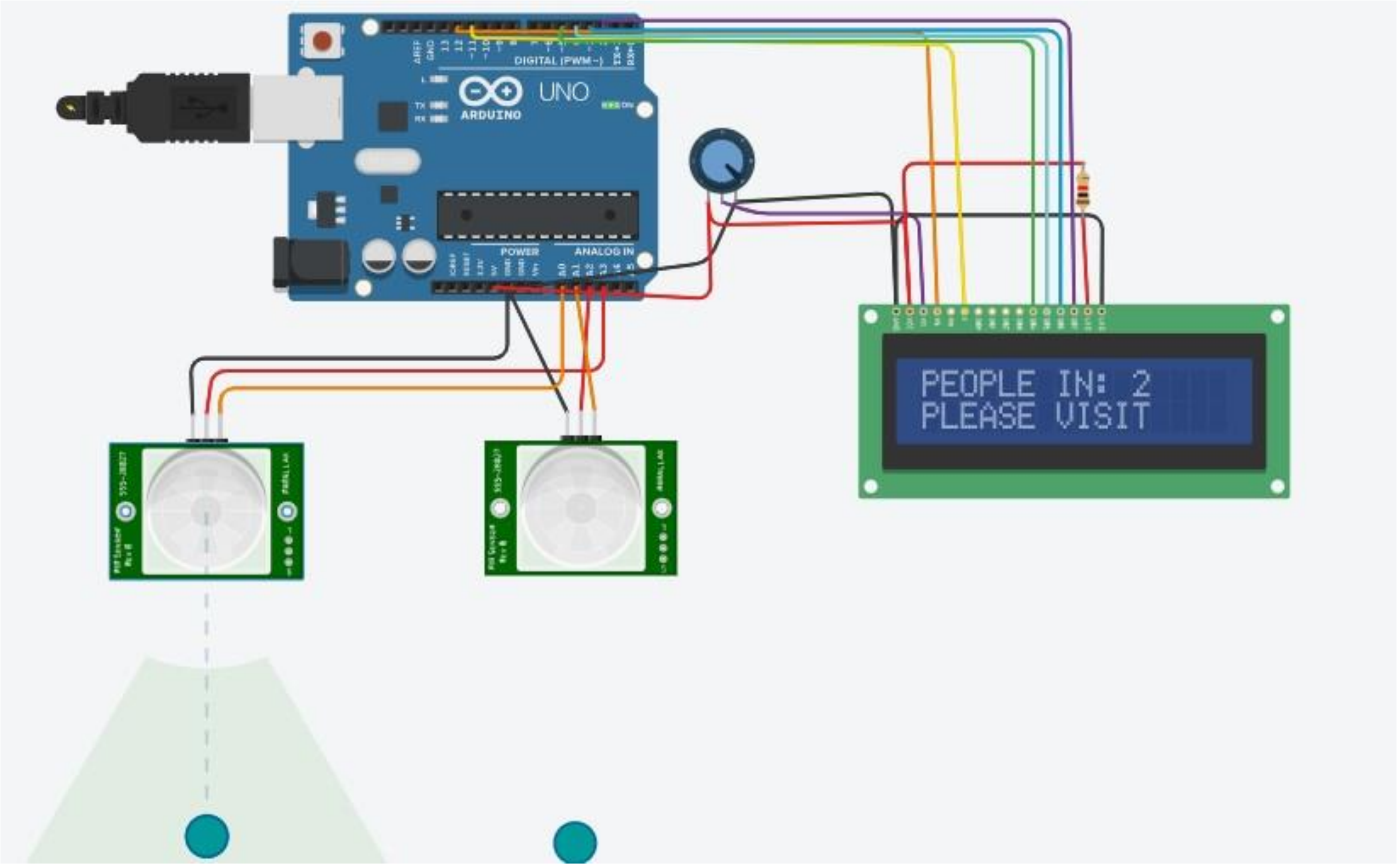
Component Connection

Name	Quantity	Pin	Connection
PIR1	1	Ground	GND
		Power	A3
		Signal	A0
PIR2	1	Ground	GND
		Power	A2
		Signal	A1
LCD 16 x 2	1	GND	LED
		VCC	LED
		VO	Wiper
		RS	D12
		E	D11
		D84	D5
		D85	D4
		D86	D3
		D87	D2
250 kΩ Potentiometer	1	Terminal1	5V
		Wiper	VO
		Terminal2	GND
220 Ω Resistor	1	Terminal1	LED
		Terminal2	VCC

Code

```
#include <LiquidCrystal.h>
int in = 15;  //A1
int inpr = 16; //A2
int out = 14; //A0
int outpr = 17; //A3
int ppl = 0;
LiquidCrystallcd(12, 11, 5, 4, 3, 2);
bool pi = 0;
bool po = 0;
void setup() {
pinMode(15, INPUT);
pinMode(14, INPUT);
pinMode(16, OUTPUT);
pinMode(17, OUTPUT);
lcd.begin(16, 2);
}
void loop() {
lcd.clear();
digitalWrite(outpr, HIGH);
digitalWrite(inpr, HIGH);
pi = digitalRead(in);
  po = digitalRead(out);
  if (pi == 1){
    ppl--;
delay(500);
  }
  else if (po == 1){
    ppl++ ;
delay(500);
  }
  ppl = constrain(ppl, 0, 50);
lcd.setCursor(0, 0);
lcd.print("PEOPLE IN:");

lcd.setCursor(11, 0);

lcd.print(ppl);

  if (ppl >= 20){

lcd.setCursor(0, 1);

lcd.print("PLEASE WAIT");

delay(1000);

  }

  if (ppl <= 19){

lcd.setCursor(0, 1);

lcd.print("PLEASE VISIT");

delay(1000);

  }

}
```

OUTPUT



```
PEOPLE IN: 2
PLEASE VISIT
```

Assignment 3:
Rain Drop Sensor.

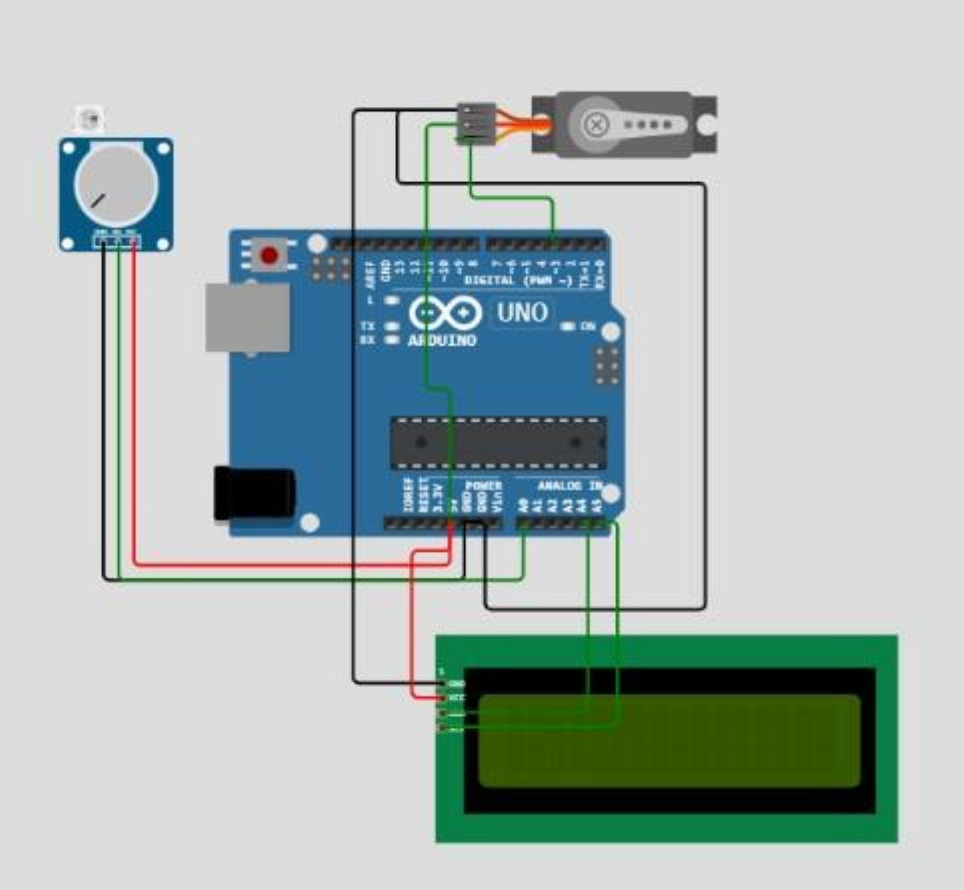
The raindrop sensor measures the moisture via analog output pins and it provides a digital output based on the threshold value. The modules are equipped with a sensor that detects raindrops and outputs a digital signal to the Arduino.

Software Used : WokWi

Components Used

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
250 kΩ Potentiometer	1	To read analog input
Servo	1	Servopin

Circuit Design



Component Connection

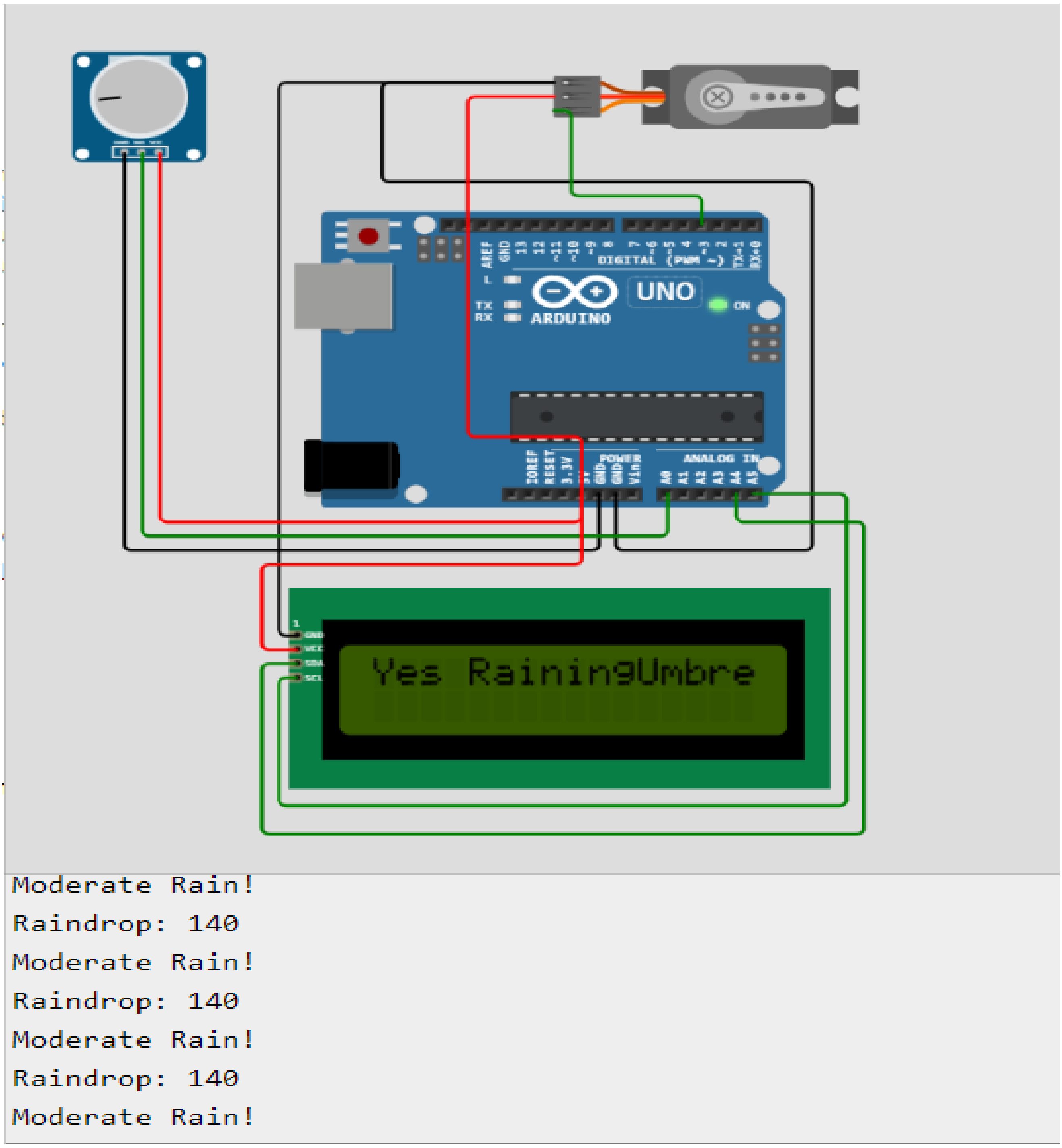
Name	Quantity	Pin	Connection
LCD (I2C)	1	GND	Arduino GND
		VCC	Arduino 5V
		SDA	Arduino A4
		SCL	Arduino A5
250 kΩ Potentiometer	1	GND	Arduino GND
		SIG	Arduino A0
		VCC	Arduino 5V
Servo	1	GND	Arduino GND & LCD (i2C) GND
		V+	Arduino 5V
		PWM	Arduino 3

Code

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const int raindropPin = A0;      // Analog pin for the simulated raindrop sensor
const int servoPin = 3;          // Digital pin for the servo motor
const int lcdColumns = 20;       // Number of columns in your LCD
const int lcdRows = 4;           // Number of rows in your LCD

Servo umbrellaServo;             // Create a servo object for the umbrella
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);  // Change the address if needed

const int rainThreshold = 400;    // Adjust the raindrop value

void setup() {
   Serial.begin(9600);
   lcd.begin(lcdColumns, lcdRows);  // Initialize the LCD screen
   lcd.print("Display Rain Fall Details");
   delay(2000);
   lcd.clear();
}

void loop()
```

```
{
    int raindropValue = analogRead(raindropPin);
    lcd.setCursor(0, 0);

if (raindropValue >0)
{
                                lcd.print("Yes Raining");
}
      else
{     lcd.print("No Rain");
}

Serial.print("Raindrop: ");
Serial.println(raindropValue);

if (raindropValue == 0)
      {
        Serial.println("No Rain!");
        lcd.print("Umbrella: Not Required ");
      }
      else if (raindropValue < rainThreshold)
      {
Serial.println("Moderate Rain!");
            lcd.print("Umbrella: Open ");
      }
      else
      {
          Serial.println("Heavy Rain!");
        lcd.print("Umbrella: Open ");

      }

    delay(1000); // Adjust delay based on your needs
}
```

Output



```
Yes RainingUmbre
```

```
Moderate Rain!
Raindrop: 140
Moderate Rain!
Raindrop: 140
Moderate Rain!
Raindrop: 140
Moderate Rain!
```

Assignment 4:

Moisture Sensor.

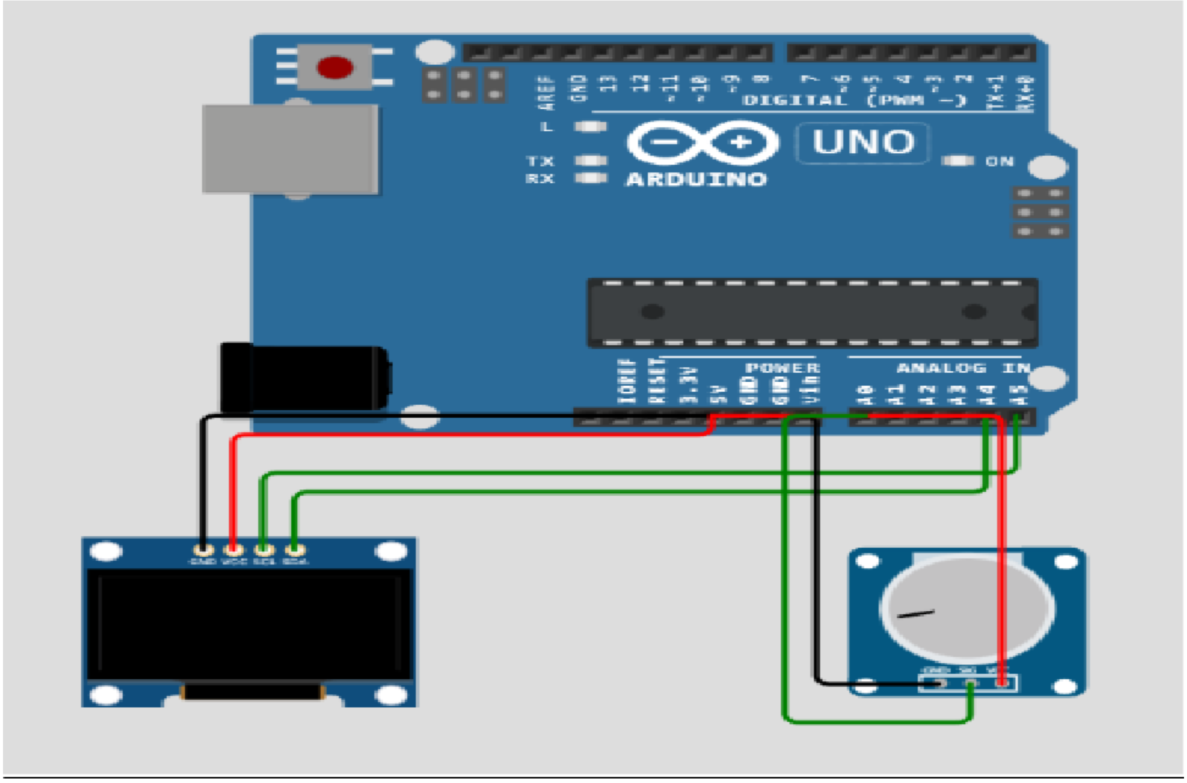
Soil Moisture can measure the moisture content in the soil based on the change in resistance between the two conducting plates. The resistance between the two conducting plates varies in an inverse manner with the amount of moisture present in the soil. To focus on a moisture sensor is a device used to measure the moisture levelin soil or other materials.

Software Used: WokWi

Components Used

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
250 kΩ Potentiometer	1	To read analog input
oled1	1	SSD1306 OLED Display

Design



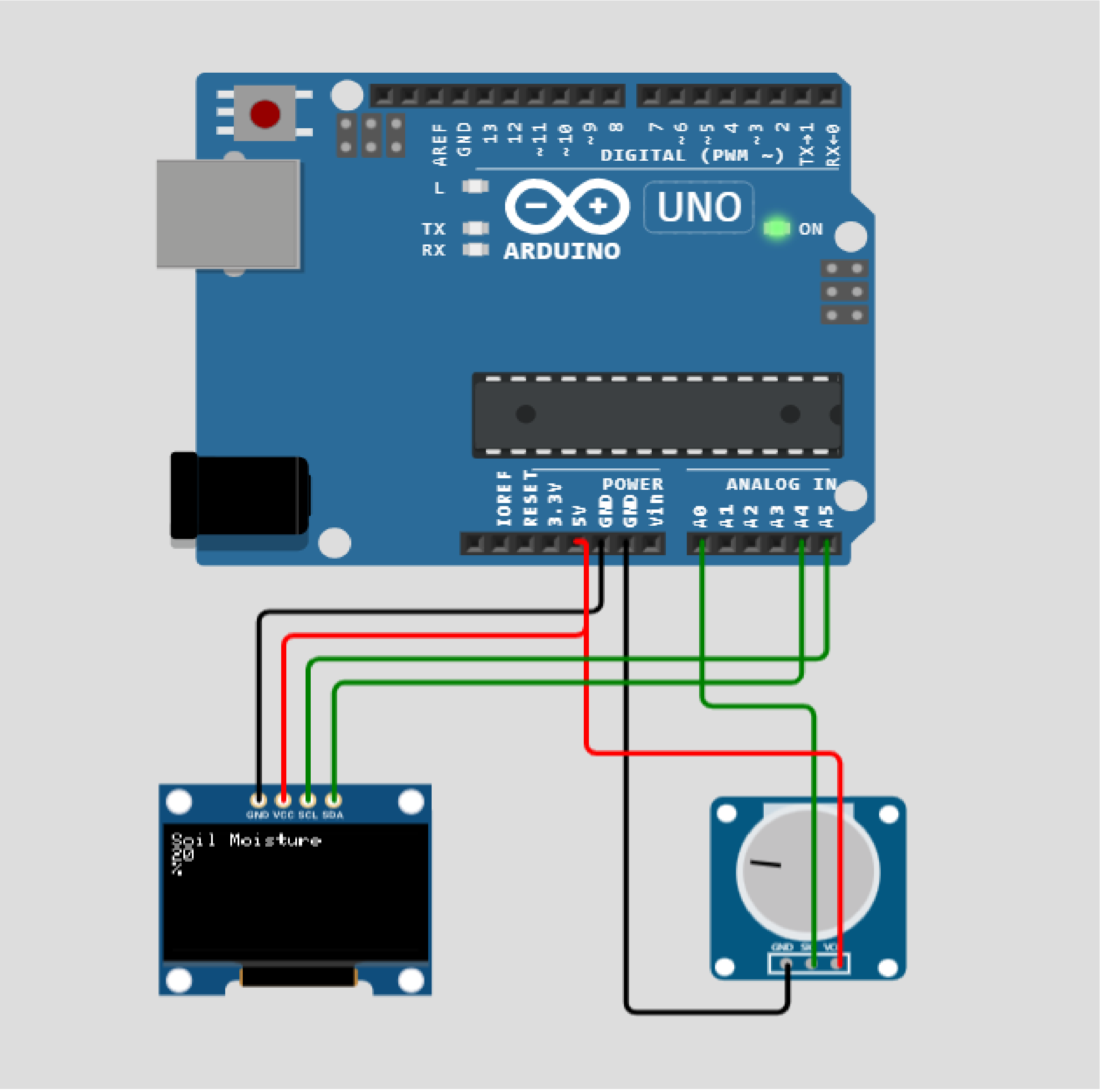
Component Connection

Name	Quantity	Pin	Connection
250 kΩ Potentiometer	1	GND	Arduino GND
		SIG	Arduino A0
		VCC	Arduino 5V
OLED	1	GND	Arduino GND
		VCC	Arduino 5V
		SCL	Arduino A5
		SDA	Arduino A4

Code

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
#define sensor A0

Adafruit_SSD1306 display(OLED_RESET);

void setup() {
  // put your setup code here, to run once:
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

}

void loop() {
  // put your main code here, to run repeatedly:
  int value = analogRead(sensor);
  int percent = map(value, 1024, 0,0,25);
  display.setTextSize(0.5);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println("Soil Moisture");
  display.println(percent);
  display.print("%");
  display.display();
  display.clearDisplay();
}
```

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
ANALOG IN
A0
A1
A2
A3
A4
A5
GND VCC SCL SDA
Soil Moisture
%
GND SIG VCC

Assignment 5:

Room Temperature Detection

Temperature sensor LM35 and Arduino Uno are the hardware used interfaced with computer, and the temperature is controlled in the room. Temperature is displayed on LCD display employing A1 pin of hardware with the help of analog pin utilizing pulse width modulation (PWM).

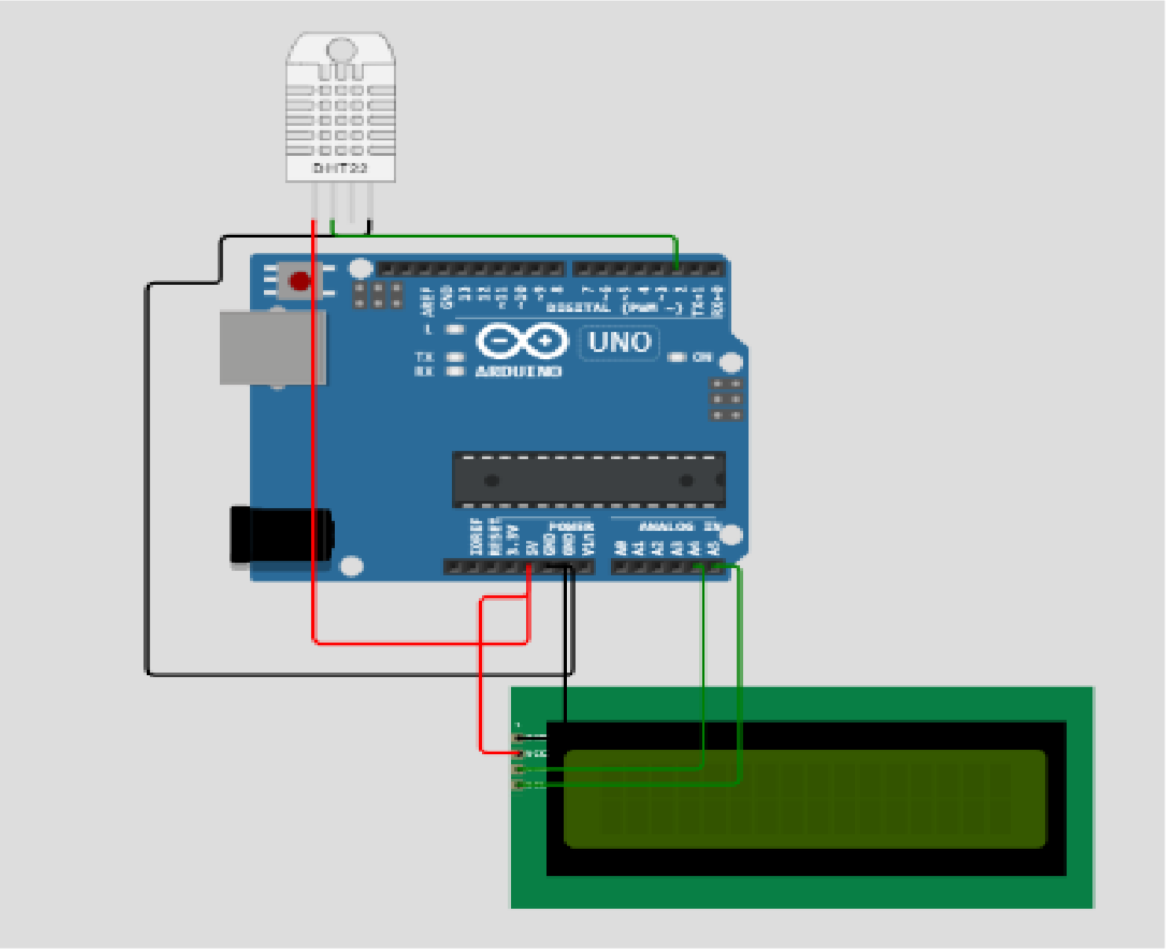
Software Used:

WokWi

Components Used

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
DHT22	1	Digital temperature and humidity sensor
LCD(I2C)	1	Liquid Crystal Display to display the room temperature

Circuit Design



Component Connection

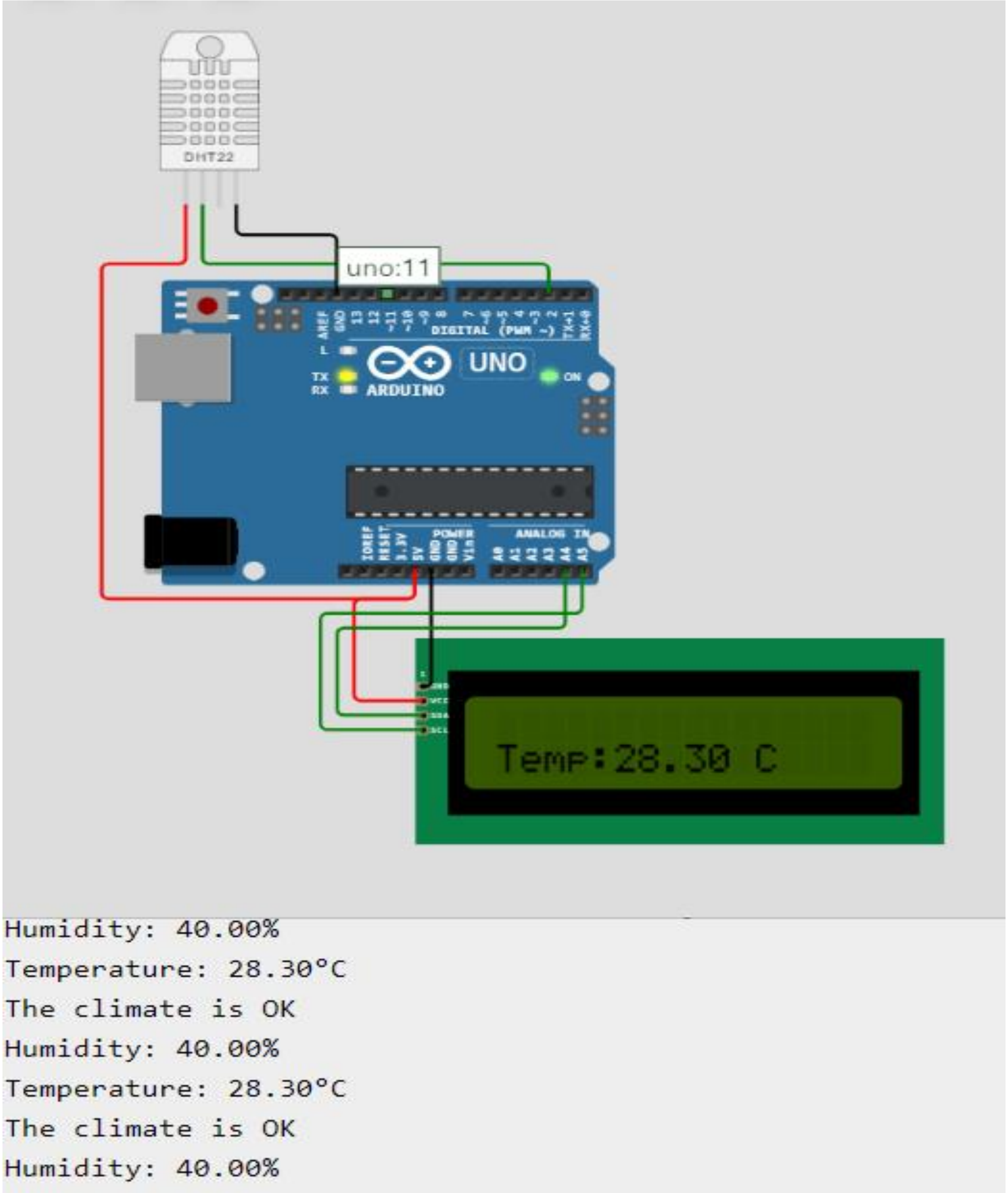
Name	Quantity	Pin	Connection
DHT22	1	GND	Arduino GND
		SDA	Arduino 2
		NC	---
		VCC	Arduino 5V
LCD(I2C)	1	GND	Arduino GND
		VCC	Arduino 5V
		SCL	Arduino A5
		SDA	Arduino A4

Code

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const int dhtPin = 2;            // Digital pin for the DHT22 sensor
const int lcdColumns = 20;        // Number of columns in your LCD
const int lcdRows = 4;           // Number of rows in your LCD

DHT dht(dhtPin, DHT22);           // Create a DHT object for the DHT22 sensor
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);  // Change the address if needed

const int temperatureThreshold = 30;  // Adjust the temperature threshold based on your needs

void setup() {
   Serial.begin(9600);
   dht.begin();  // Initialize the DHT22 sensor
   lcd.begin(lcdColumns, lcdRows);  // Initialize the LCD screen
   lcd.print("Room Temperature Sensing");
   delay(2000);
   lcd.clear();
}

void loop() {

   float humidity = dht.readHumidity();
   float temperatureC = dht.readTemperature();
```

```
    lcd.setCursor(0, 0);
    lcd.setCursor(0, 1);
    lcd.print("Temp:");
    lcd.print(temperatureC);
    lcd.print(" C");

    String msg = temperatureC > 35 ? "HOT" : temperatureC < 15 ? "COLD" : "OK";

    Serial.print("Temperature: ");
    Serial.print(temperatureC);
    Serial.println("°C");
    delay(500);

    Serial.print("The climate is ");
    Serial.println(msg);
   delay(500);

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println("%");
  delay(1000);

}
```

OUTPUT



```
Humidity: 40.00%
Temperature: 28.30°C
The climate is OK
Humidity: 40.00%
Temperature: 28.30°C
The climate is OK
Humidity: 40.00%
```

Assignment 6:
Piano Sensor (basics of Arduino, LED, Buzzer, Pushbutton)

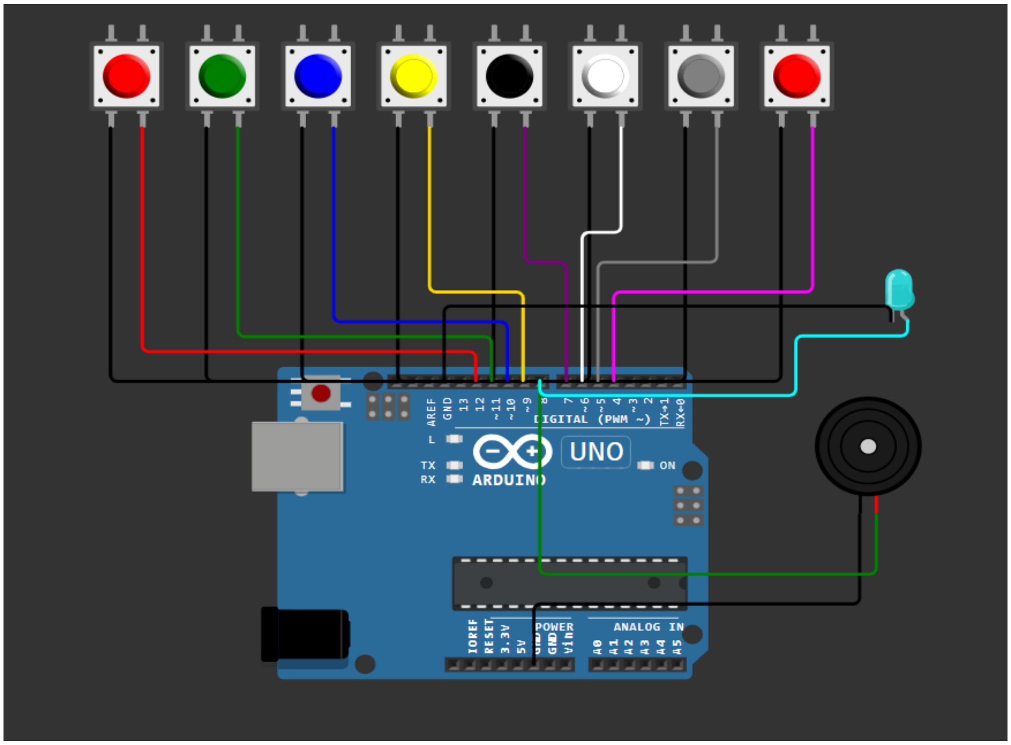
Software Used:

WOKWI

Components

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
Pushbutton	8	Allow us to power the circuit or make any particular connectiononly when we press the button
LED	1	LED(Purple)
Buzzer	1	An efficient component to include the features of sound in our system or project

Circuit Design



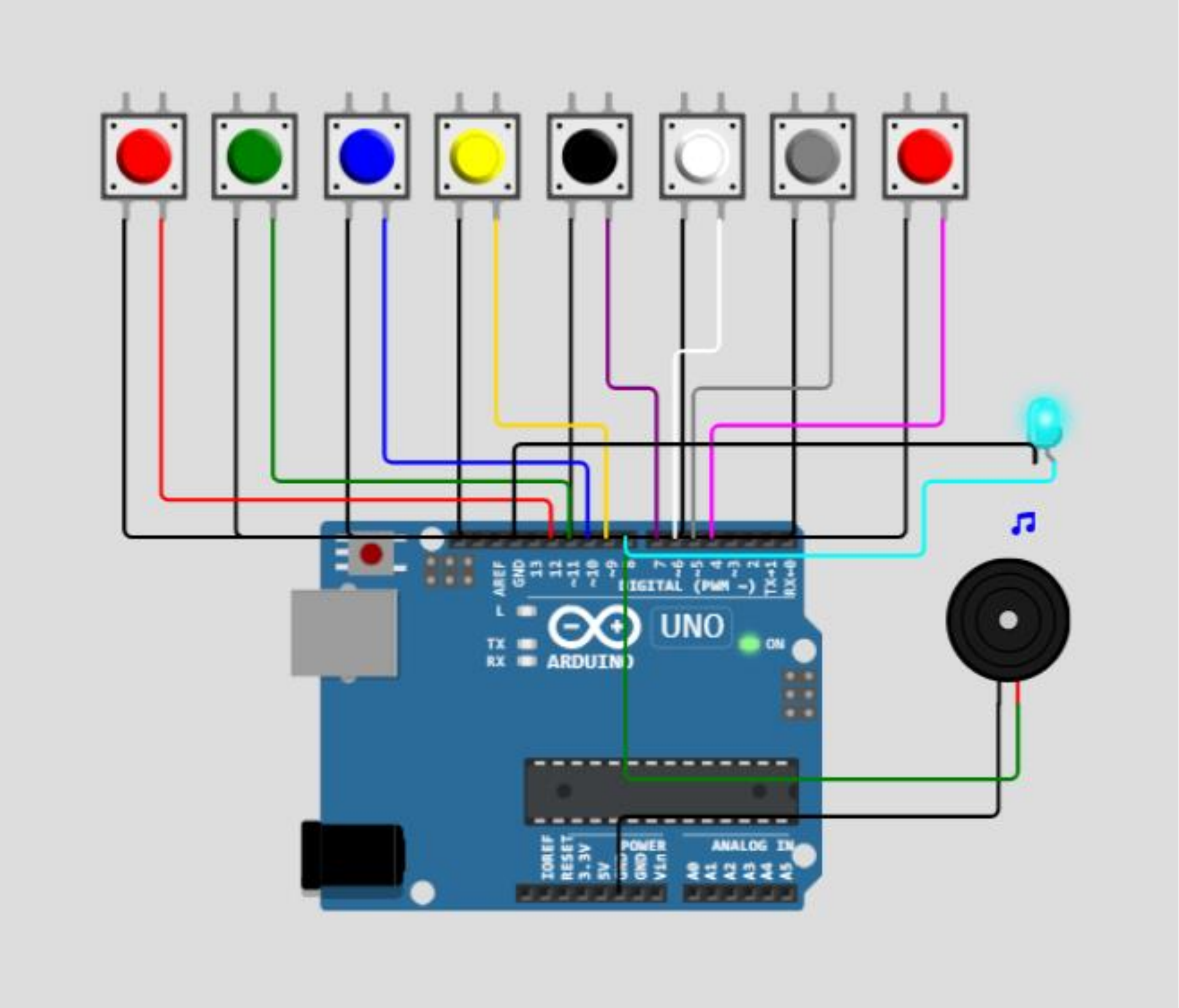
Component Connection

Name	Quantity	Pin	Connection
PushButton1	1	btn1:2.r	GND
		btn1:1.r	12
PushButton2	1	btn2:2.r	GND
		btn2:1.r	11
PushButton3	1	btn3:2.r	GND
		btn3:1.r	10
PushButton4	1	btn4:2.r	GND
		btn4:1.r	9
PushButton5	1	btn5:2.r	GND
		btn5:1.r	7
PushButton6	1	btn6:2.r	GND
		btn6:1.r	6
PushButton7	1	btn7:2.r	GND
		btn7:1.r	5
PushButton 8	1	btn8:2.r	GND
		btn8:1.r	4
LED	1	Cathode(negative pin)	GND
		Anode(positive pin)	8
Buzzer	1	bz1:1	GND
		bz1:2	8

Code

```
#include "pitches.h"

#define SPEAKER_PIN 8

const uint8_t buttonPins[] = { 12, 11, 10, 9, 7, 6, 5, 4 };
const int LEDPIN = 8;
const int buttonTones[] = {
  NOTE_C4, NOTE_D4, NOTE_E4, NOTE_F4,
  NOTE_G4, NOTE_A4, NOTE_B4, NOTE_C5
};
const int numTones = sizeof(buttonPins) / sizeof(buttonPins[0]);

void setup() {
  for (uint8_t i = 0; i < numTones; i++) {
    pinMode(buttonPins[i], INPUT_PULLUP);
  }
  pinMode(SPEAKER_PIN, OUTPUT);
  pinMode(LEDPIN, OUTPUT);
}

void loop() {
  int pitch = 0;
  for (uint8_t i = 0; i < numTones; i++) {
    if (digitalRead(buttonPins[i]) == LOW) {
      pitch = buttonTones[i];
    }
  }
  if (pitch) {
    tone(SPEAKER_PIN, pitch);
    pinMode(LEDPIN,HIGH);
  }
else {
    noTone(SPEAKER_PIN);
  }
}
```

Output


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

Assignment 7:
IR Sensor (basics of Arduino, LED, IR, Logic Analyzer)

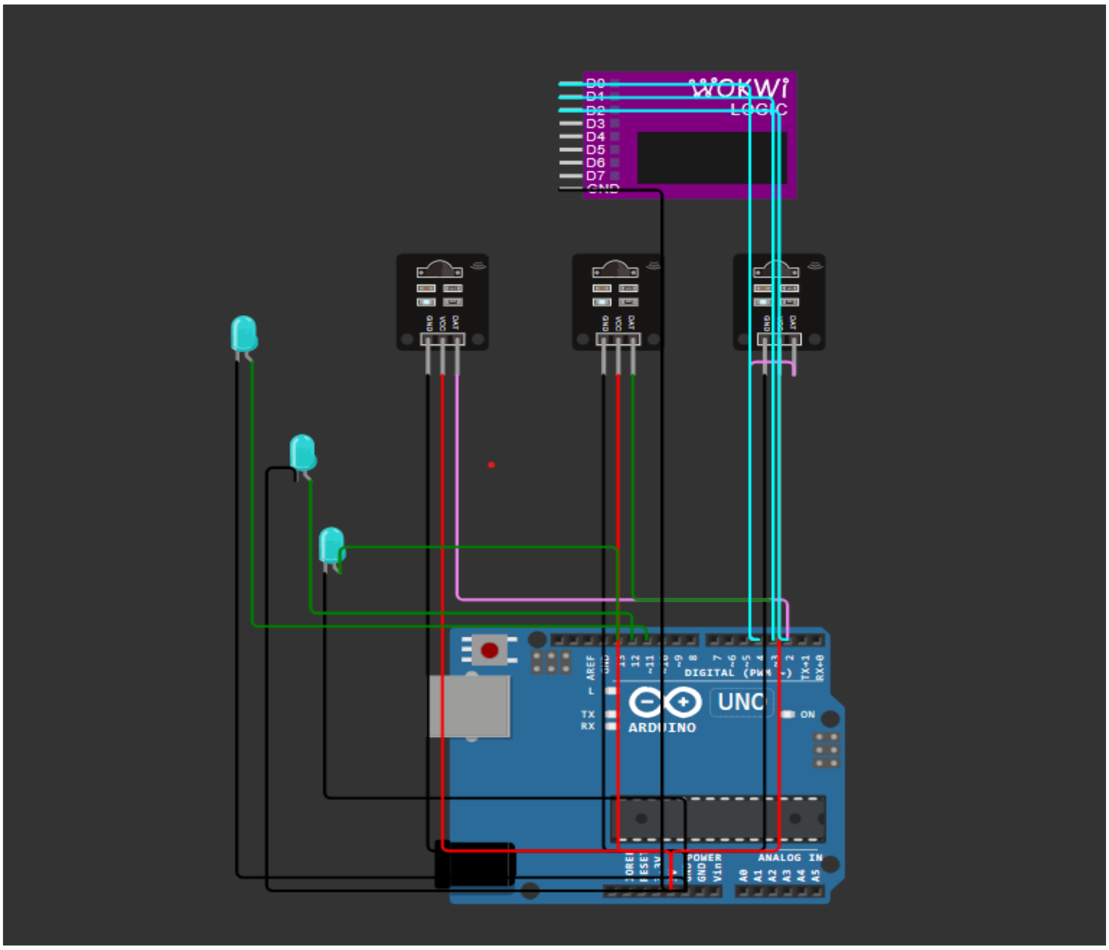
Software Used:

WOKWI

Components

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
IR	3	Measures and detects infrared radiation in its surrounding environment.
LED	3	LED(Purple), LED(Blue), LED(Orange)
LOGIC ANALYZER (8 CHANNELS)	1	Collected at the same time they signal analysis, such as I2C, UART, sampling and analysis.

Circuit Design



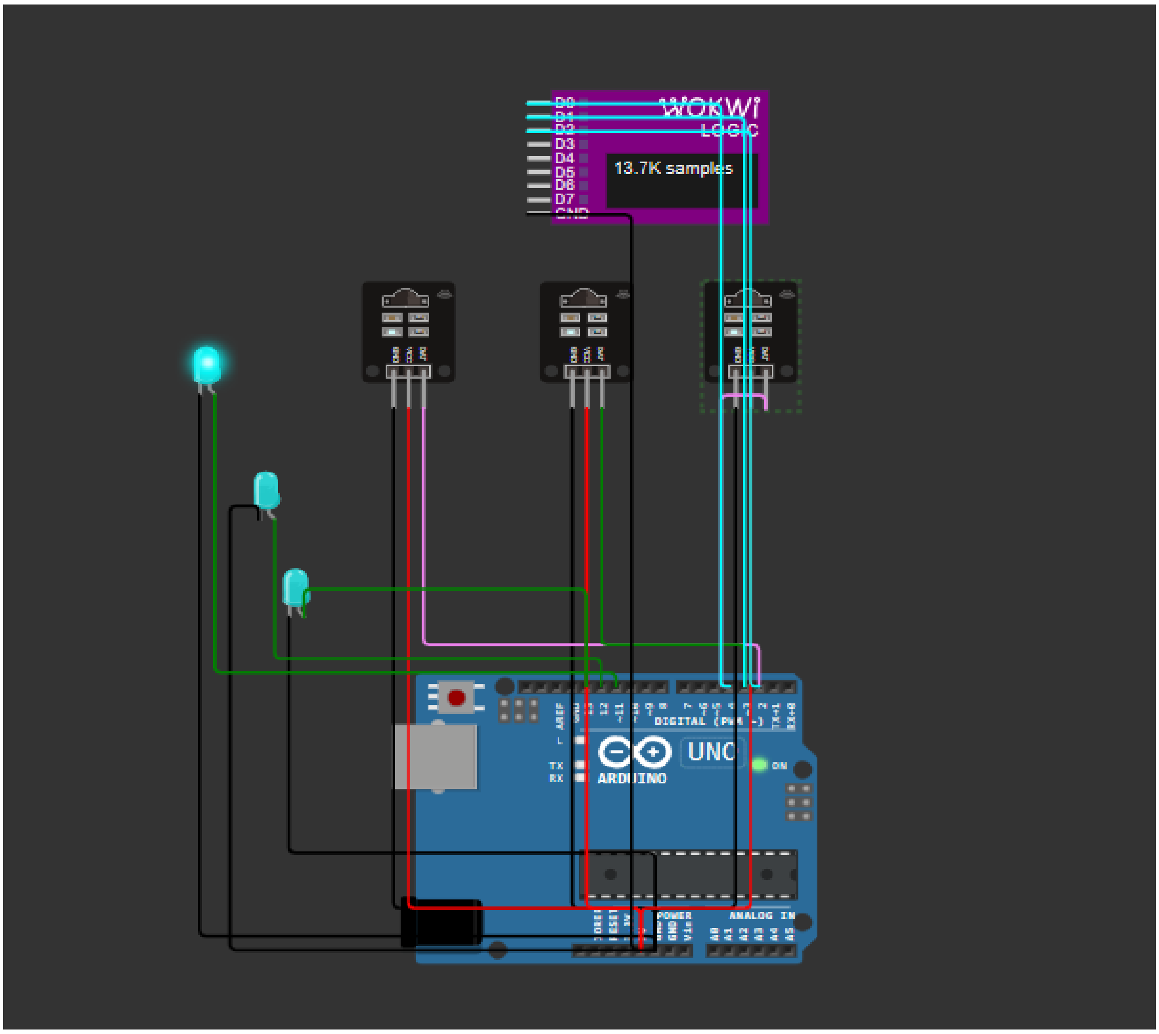
Component Connection

Name	Quantity	Pin	Connection
IR1	1	GND	GND
		VCC	5V
		DAT	2
IR2	1	GND	GND
		VCC	5V
		DAT	3
IR3	1	GND	GND
		VCC	5V
		DAT	4
LED1	1	Anode (positive pin)	13
		Cathode (negative pin)	GND
LED2	1	Anode (positive pin)	12
		Cathode (negative pin)	GND
LED3	1	Anode (positive pin)	11
		Cathode (negative pin)	GND
LOGIC ANALYZER (8 CHANNELS)	1	D0	4
		D1	3
		D2	2
		GND	GND

Code

```
int LED1 = 13;
int LED2 = 12;
int LED3 = 11;
int IR1 = 2;
int IR2 = 3;
int IR3 = 4;
int val1 = 0;
int val2 = 0;
int val3 = 0;

void setup() {
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(IR3, INPUT);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  Serial.begin(9600);

}

void loop() {
  val1 = digitalRead(IR1);
  val2 = digitalRead(IR2);
  val3 = digitalRead(IR3);

  if (val1 == 0) {
    digitalWrite(LED1, HIGH);

  }
  else {
    digitalWrite(LED1, LOW);
  }
  if (val2 == 0) {
    digitalWrite(LED2, HIGH);
  }
  else {
    digitalWrite(LED2, LOW);
  }
  if (val3 == 0) {
    digitalWrite(LED3, HIGH);
  }
  else {
    digitalWrite(LED3, LOW);
}
delay(200);

}
```

Output


WOKWI
LOGIC
D0
D1
D2
D3
D4
D5
D6
D7
GND
13.7K samples
UNO
ARDUINO
DIGITAL (PWM ~)
POWER
ANALOG IN
ON
TX
RX

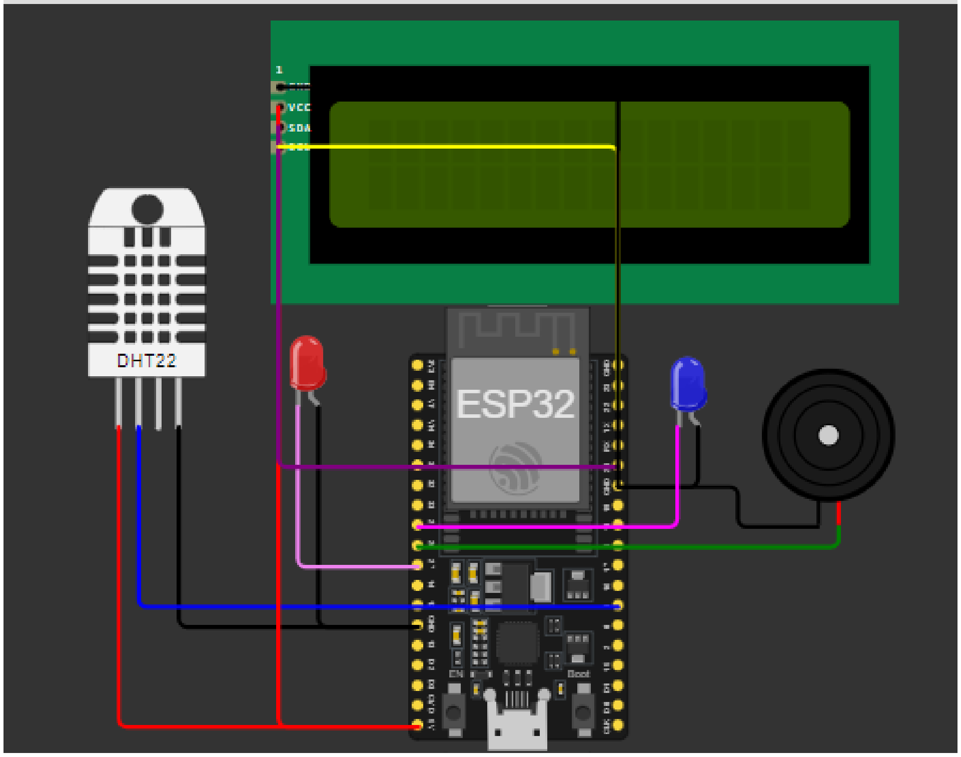
Assignment 10
Fire Alarm Systems (basics of ESP32, LED, Buzzer, LCD1602,DHT22)

Software Used: WOKWI

Components

Component Name	Quantity	Description
ESP32	1	The ESP32 is a popular Wi-Fi and Bluetooth-enabled microcontroller, widely used for IoT Projects.
LCD1602	1	An LCD with 2 lines, 16 characters per line.
LED	2	LED(Blue),LED(Red)
Buzzer	1	An efficient component to include the features of sound in our system or project
DHT22	1	Digital Humidity and Temperature sensor.

Circuit Design



Component Connection

Name	Quantity	Pin	Connection
LCD1602	1	GND	GND
		VCC	5V
		SDA	21
		SCL	22
DHT22	1	GND	GND
		VCC	5V
		SDA	4
LED(Blue)	1	Cathode(negative pin)	GND
		Anode(positive pin)	25
LED(Red)	1	Cathode(negative pin)	GND
		Anode(positive pin)	27
Buzzer	1	bz1:1	GND
		bz1:2	26

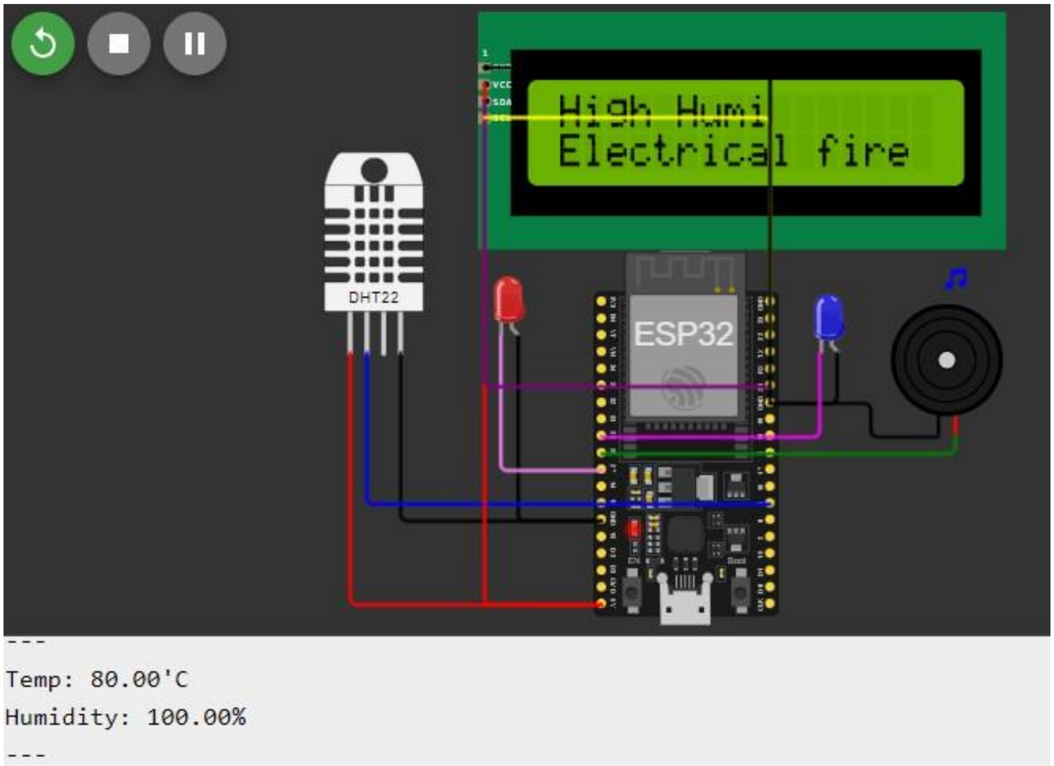
Code

```
#include <DHT.h>

#include <LiquidCrystal_I2C.h>

#define DHTPIN 4

#define DHTTYPE DHT22

#define LED_PIN 27 //red pin

#define LED1_PIN 25 //blue pin
```

```
#define BUZZER_PIN 26

#define TEMP_LOW 23

#define TEMP_HIGH 25

#define HUMI_LOW 40

#define HUMI_HIGH 60

#define I2C_ADDR    0x27

#define LCD_COLUMNS 16

#define LCD_LINES   2

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLUMNS, LCD_LINES);

void setup() {

  Serial.begin(115200);

  pinMode(LED_PIN, OUTPUT);

  pinMode(LED1_PIN, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);

    // Init

  lcd.init();

  lcd.backlight();

    // Print something

  lcd.setCursor(4, 0);

  lcd.print("T&H Fire");

  lcd.setCursor(2, 1);

  lcd.print("Alarm System");

  delay(1000);

  lcd.clear();

}

void loop() {

  float temp = dht.readTemperature();
  float humi = dht.readHumidity();
  Serial.print("Temp: ");
  Serial.print(temp);
  Serial.println("'C");
  Serial.print("Humidity: ");
  Serial.print(humi);
  Serial.println("%");
  Serial.println("---");
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.print("'C");
  lcd.setCursor(0, 1);
  lcd.print("Humi: ");
  lcd.print(humi);
  lcd.print(" %");
  delay(1000);
  if(temp > TEMP_HIGH)
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("High Temp");
    lcd.setCursor(0, 1);
    lcd.print("Warming");
    delay(1000);
    lcd.clear();
    if (humi < HUMI_LOW)
    {
      digitalWrite(LED_PIN, HIGH); // Turn on Red LED

      tone(BUZZER_PIN, 1000); // Turn on buzzer

      lcd.clear();

      lcd.setCursor(0, 0);

      lcd.print("LowHumi");

      lcd.setCursor(0, 1);

      lcd.print("Fire Detected");

      delay(1000);

      digitalWrite(LED_PIN, LOW); // Turn off Red LED

      noTone(BUZZER_PIN); // Turn off buzzer

      lcd.clear();

    }

    else if (humi > HUMI_HIGH)

    {

      digitalWrite(LED_PIN, HIGH); // Turn on Red LED

      tone(BUZZER_PIN, 1000); // Turn on buzzer

      lcd.clear();

      lcd.setCursor(0, 0);

      lcd.print("High Humi");

      lcd.setCursor(0, 1);

      lcd.print("Electrical fire");

      delay(1000);

      digitalWrite(LED_PIN, LOW); // Turn off Red LED

      noTone(BUZZER_PIN); // Turn off buzzer

      lcd.clear();

    }

  }

  else if(temp < TEMP_LOW)
```

```
{
  digitalWrite(LED1_PIN, HIGH); // Turn on Blue LED
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Low Temp");
  lcd.setCursor(0, 1);
  lcd.print("Beware Cold");
  delay(1000);
  digitalWrite(LED1_PIN, LOW); // Turn off Blue LED
  lcd.clear();
  if(humi < HUMI_LOW)
  {
    digitalWrite(LED1_PIN, HIGH); // Turn on Blue LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Low Humidity");
    lcd.setCursor(0, 1);
    lcd.print("Drink Water");
    delay(1000);
    digitalWrite(LED1_PIN, LOW); // Turn off Blue LED
    lcd.clear();
  }
  else if(humi > HUMI_HIGH)
  {
    digitalWrite(LED1_PIN, HIGH); // Turn on Blue LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("High Humi");
    lcd.setCursor(0, 1);
    lcd.print("Sloppy Floor");
    delay(1000);
    digitalWrite(LED1_PIN, LOW); // Turn off Blue LED
    lcd.clear();
  }
}
else if(temp > TEMP_LOW & temp < TEMP_HIGH)
{
  if(humi < HUMI_LOW)
  {
    digitalWrite(LED1_PIN, HIGH); // Turn on Blue LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Low Humidity");
    lcd.setCursor(0, 1);
    lcd.print("Drink Water");
    delay(1000);
    digitalWrite(LED1_PIN, LOW); // Turn off Blue LED
    lcd.clear();
  }
  else if(humi > HUMI_HIGH)
  {
    digitalWrite(LED1_PIN, HIGH); // Turn on Blue LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("High Humi");
    lcd.setCursor(0, 1);
    lcd.print("Sloppy Floor");

    delay(1000);

    digitalWrite(LED1_PIN, LOW); // Turn off Blue LED

    lcd.clear();

   }

  }

}
```

Output



```
---
Temp: 80.00'C
Humidity: 100.00%
---
```

Assignment 11

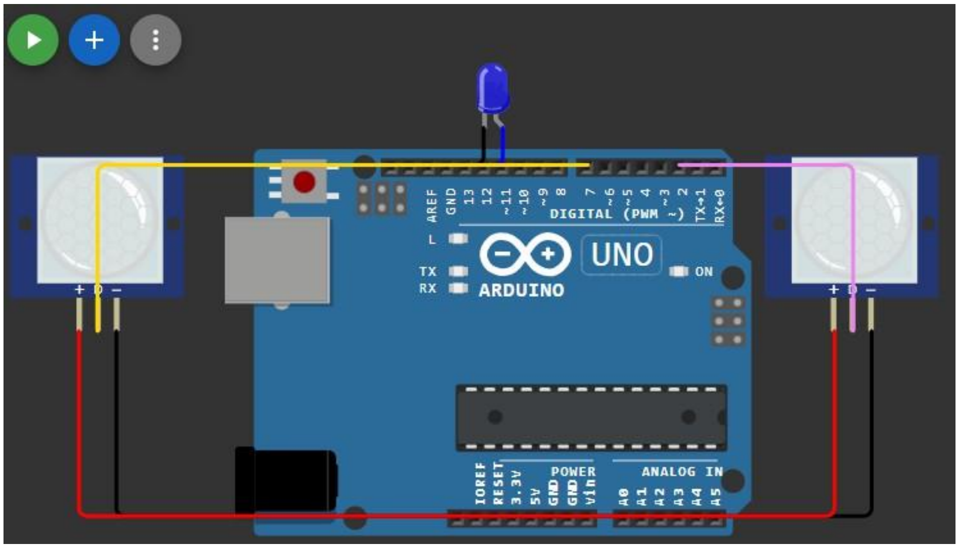
Automated Room Lighting System (basics of Arduino Uno, LED, PIR motion sensor)

Software Used :WOKWI

Components

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
LED	1	LED(Blue)
PIR motion sensor	2	an electronic sensor that measures infrared (IR) light radiating from objects in its field of view.

Circuit Design



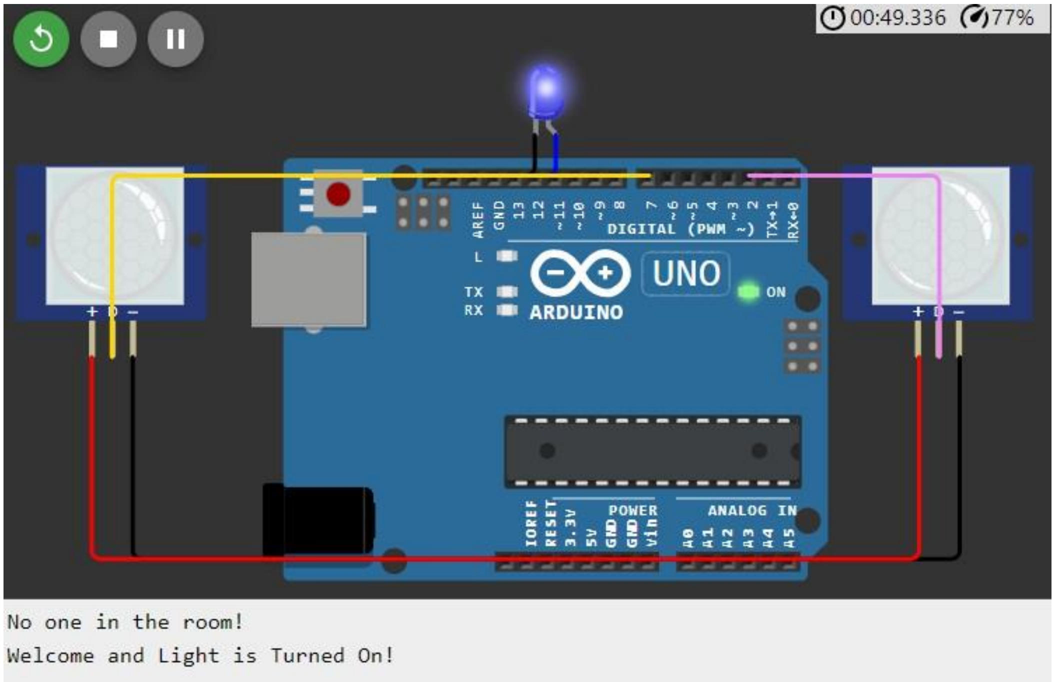
Component Connection

Name	Quantity	Pin	Connection
PIR motion sensor 1	1	GND	GND
		VCC	5V
		OUT	2
PIR motion sensor 2	1	GND	GND
		VCC	5V
		OUT	7
LED(Blue)	1	Cathode(negative pin)	GND
		Anode(positive pin)	12

Code

```
is int ledPin = 12;
int inputPin = 2;
int outPin = 7;
int pirState = LOW;
int val = 0;
int val2 = 0;
void setup() {
  pinMode(ledPin, OUTPUT);      // declare LED as output
  pinMode(inputPin, INPUT);
  pinMode(outPin, INPUT);    // declare sensor as input
  Serial.begin(9600);
  Serial.println("No one in the room!");
}
void loop() {
  val = digitalRead(inputPin);
  val2 = digitalRead(outPin);  // read input value
  if (val == HIGH) {            // check if the input is HIGH
    digitalWrite(ledPin, HIGH);  // turn LED ON
    if (pirState == LOW) {
      // we have just turned on
      Serial.println("Welcome and Light is Turned On!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  }
  if(val2 == HIGH){
    digitalWrite(ledPin,LOW);
  }
}
```

<u>Output</u>



```
No one in the room!
Welcome and Light is Turned On!
```

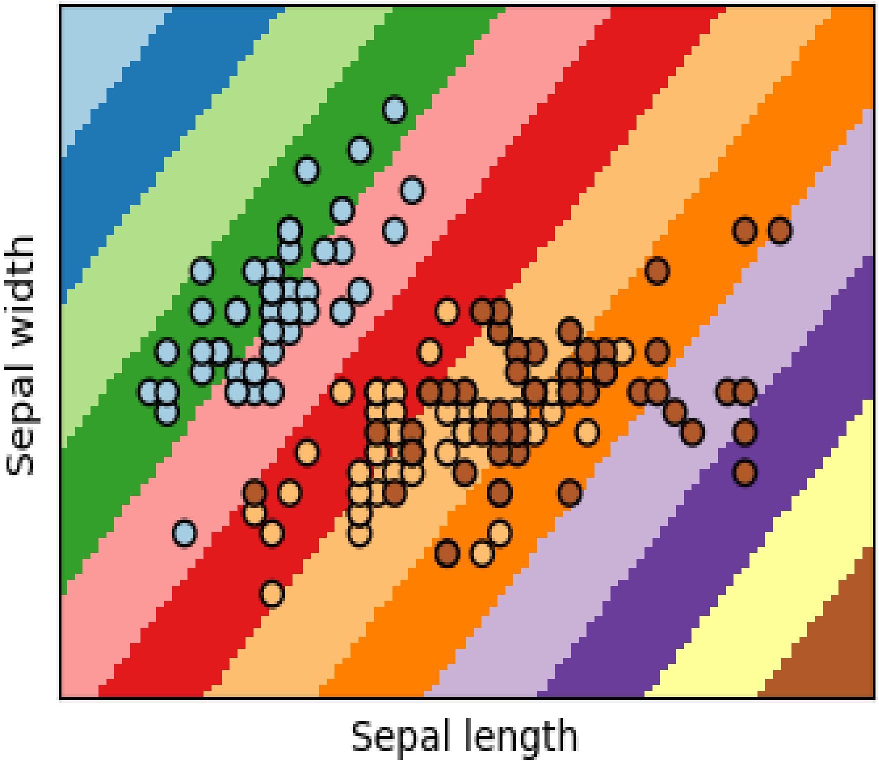
Assignment 13
Build and Apply Linear and Logistic Regression Models

Software Used: Google Colab, Iris Dataset

Code

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.linear_model import LinearRegression
iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target

linreg = LinearRegression()
linreg.fit(X, Y)
_, ax = plt.subplots(figsize=(4, 3))
DecisionBoundaryDisplay.from_estimator(
linreg,
 X,
cmap=plt.cm.Paired,
ax=ax,
response_method="predict",
plot_method="pcolormesh",
 shading="auto",
 xlabel="Sepal length",
ylabel="Sepal width",
 eps=0.5,
)

plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors="k", cmap=plt.cm.Paired)
plt.xticks(())
plt.yticks(())
plt.show()
```

Output



Assignment 13

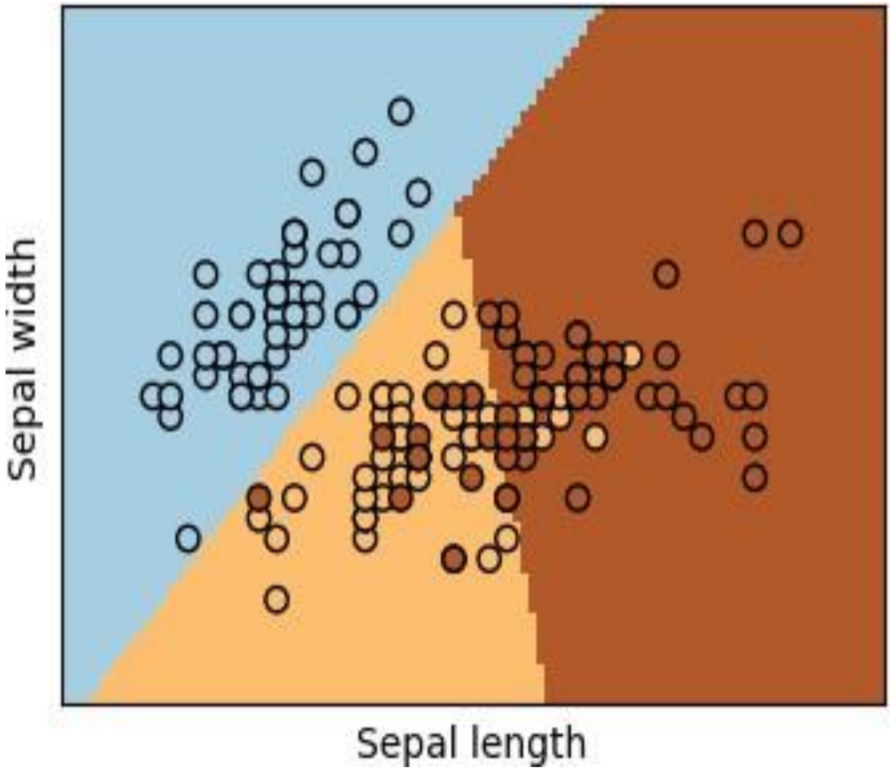
Build and Apply Linear and Logistic Regression Models

Software Used: Google Colab, Iris Dataset

Code

```
# Code source: Gaël Varoquaux

# Modified for documentation by Jaques Grobler # License: BSD 3 clause

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.inspection

import DecisionBoundaryDisplay

from sklearn.linear_model import LogisticRegression

# import some data to play with iris = datasets.load_iris()

X = iris.data[:, :2] # we only take the first two features.

Y = iris.target

# Create an instance of Logistic Regression Classifier and fit the data. logreg = LogisticRegression(C=1e5)

logreg.fit(X, Y)

_, ax = plt.subplots(figsize=(4, 3)) DecisionBoundaryDisplay.from_estimator(

logreg, X,

cmap=plt.cm.Paired, ax=ax,

response_method="predict", plot_method="pcolormesh", shading="auto", xlabel="Sepal length", ylabel="Sepal width", eps=0.5,

)

# Plot also the training points

plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors="k", cmap=plt.cm.Paired) plt.xticks(())

plt.yticks(())

plt.show()
```

Output



Assignment 14

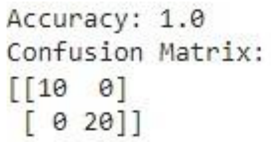
Perform Data Analysis with Machine Learning Methods

Software Used: Google Colab, Iris Dataset

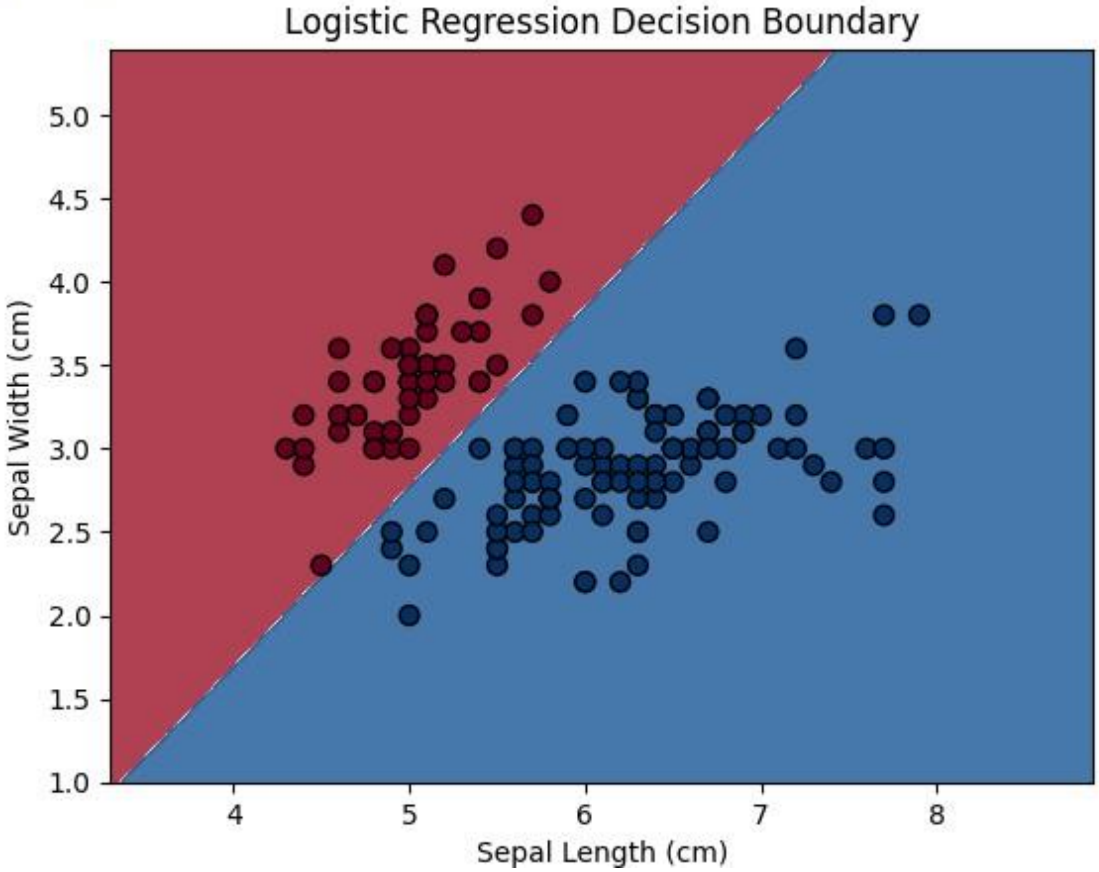
Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
 from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.datasets import load_iris
# Load Iris dataset iris = load_iris()
X = iris.data[:, :2] # Use only the first two features for simplicity
y = (iris.target != 0).astype(int) # Binary classification: setosa (0) vs. versicolor/virginica (1)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train the logistic regression model model = LogisticRegression() model.fit(X_train, y_train)
# Make predictions on the test set y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred) conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy}') print(f'Confusion Matrix:\n{conf_matrix}')
# Plot the decision boundary
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1 xx, yy = np.meshgrid(np.arange(x_min, x_max,
0.01), np.arange(y_min, y_max, 0.01))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()]) Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.RdBu, alpha=0.8) plt.scatter(X[:, 0], X[:, 1], c=y,
edgecolors='k', cmap=plt.cm.RdBu, marker='o', s=50) plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Logistic Regression Decision Boundary')
plt.show()
```

Output



```
Accuracy: 1.0
Confusion Matrix:
[[10  0]
 [ 0 20]]
```



Assignment 15

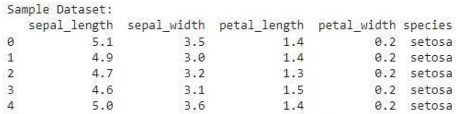
Perform Graphical Data Analysis

Software Used: Google Colab, Iris Dataset

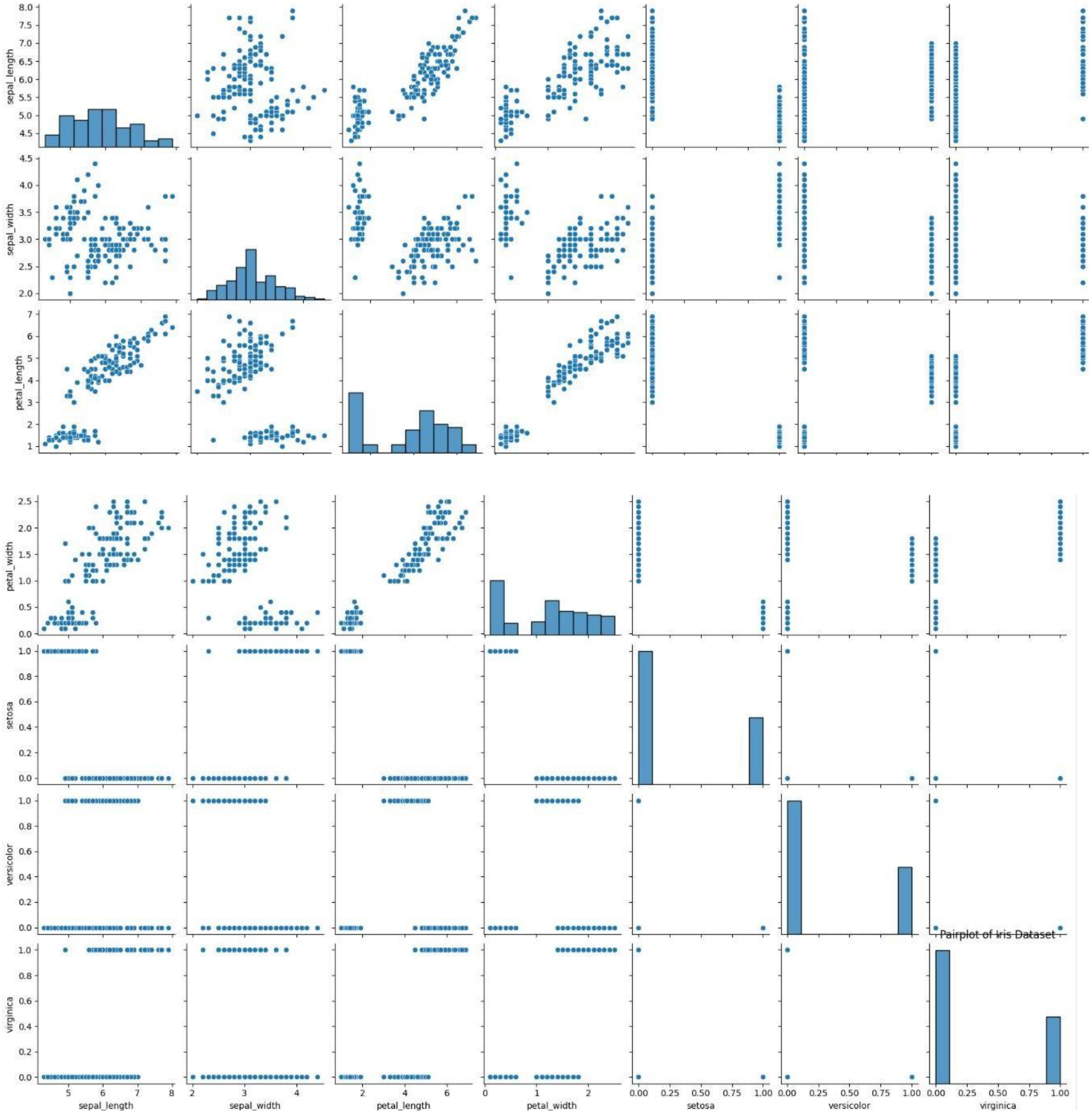
Code

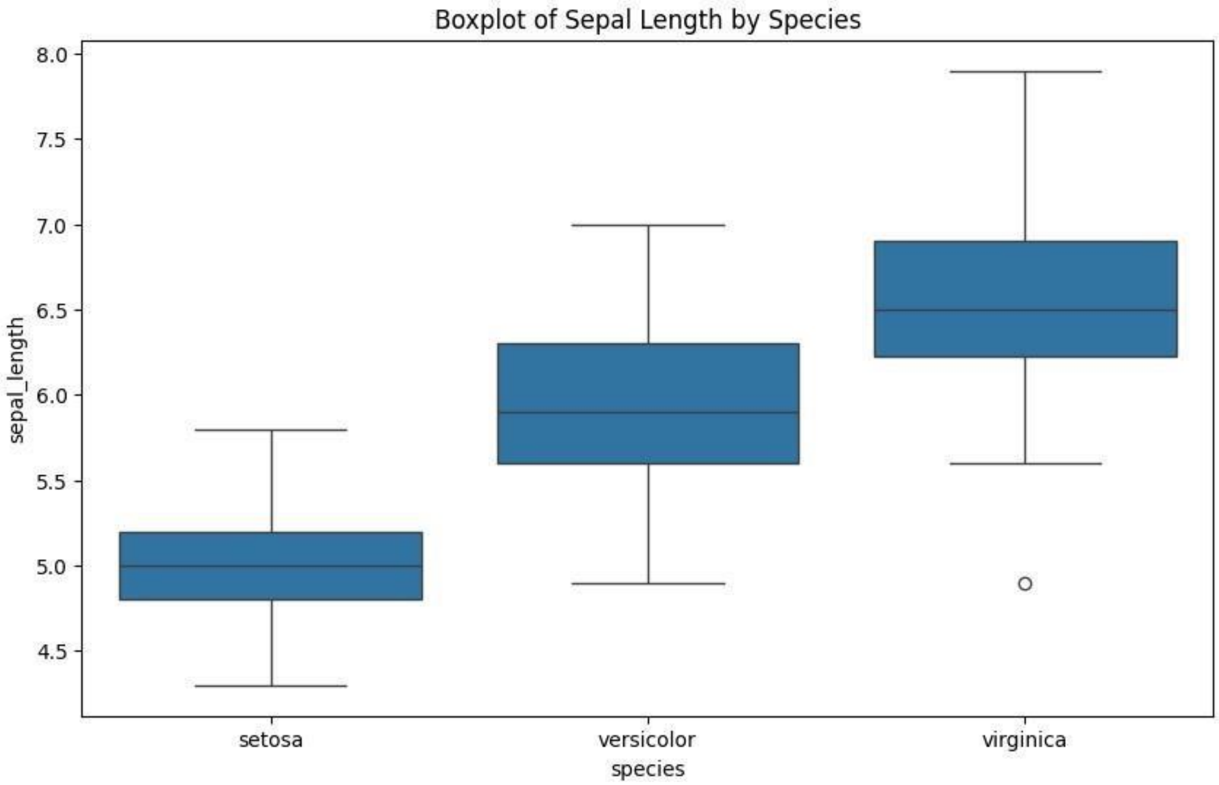
```
# Import necessary libraries import numpy as np

import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.preprocessing import OneHotEncoder

from sklearn.metrics import accuracy_score, confusion_matrix # Load a sample dataset (Iris dataset)

iris = sns.load_dataset('iris')

# Display the first few rows of the dataset print("Sample Dataset:") print(iris.head())

# One-hot encoding for the 'species' column one_hot_encoder = OneHotEncoder()

species_encoded = one_hot_encoder.fit_transform(iris[['species']]) # Create a DataFrame from the encoded species column species_encoded_df = pd.DataFrame(species_encoded.toarray(), columns=one_hot_encoder.categories_[0])

# Concatenate the original DataFrame with the encoded species DataFrame iris_encoded = pd.concat([iris.drop(columns=['species']), species_encoded_df], axis=1)

# Pairplot to visualize relationships between numerical features sns.pairplot(iris_encoded)

plt.title('Pairplot of Iris Dataset') plt.show()

# Boxplot to visualize distribution and identify outliers plt.figure(figsize=(10, 6))

sns.boxplot(x='species', y='sepal_length', data=iris) plt.title('Boxplot of Sepal Length by Species') plt.show()

# Violin plot to compare the distribution of petal length for each species plt.figure(figsize=(10, 6))

sns.violinplot(x='species', y='petal_length', data=iris) plt.title('Violin Plot of Petal Length by Species') plt.show()

# Heatmap to visualize the correlation matrix correlation_matrix = iris_encoded.corr() plt.figure(figsize=(8, 6))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Heatmap of Iris Dataset')

plt.show()
```

Output

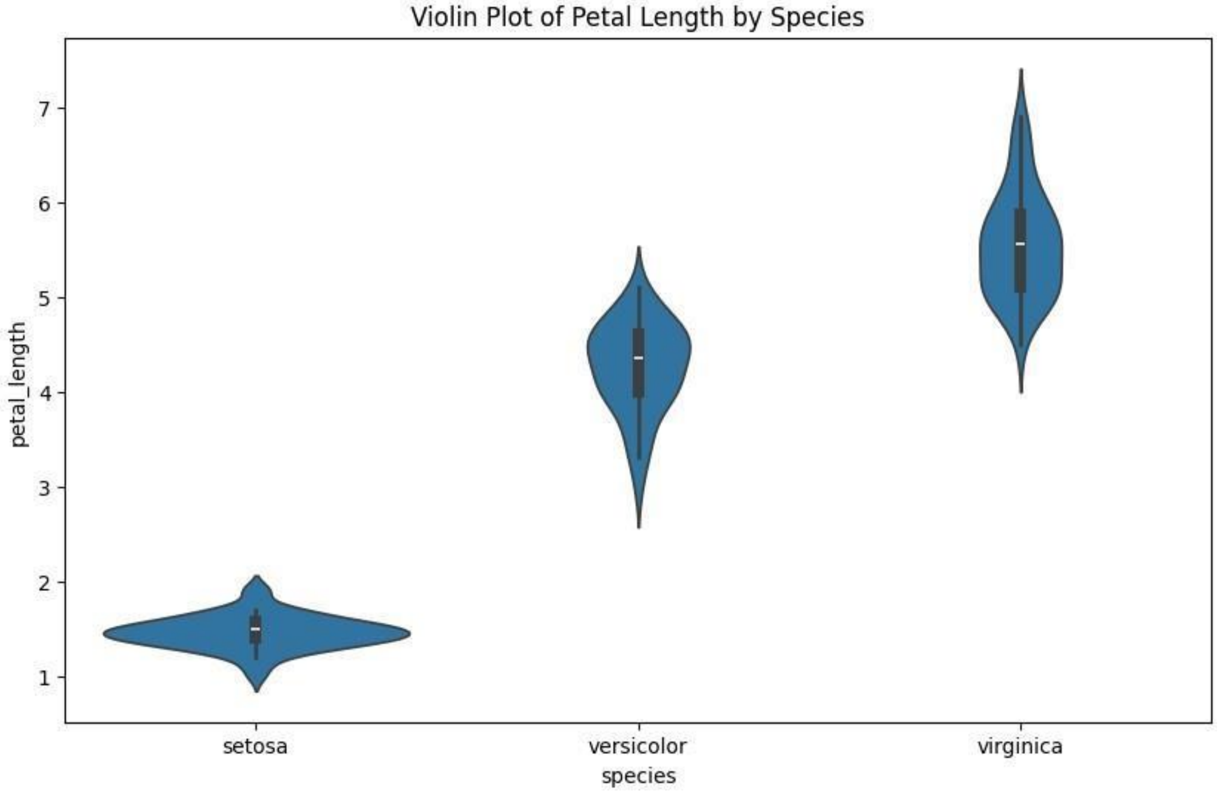


```
Sample Dataset:
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

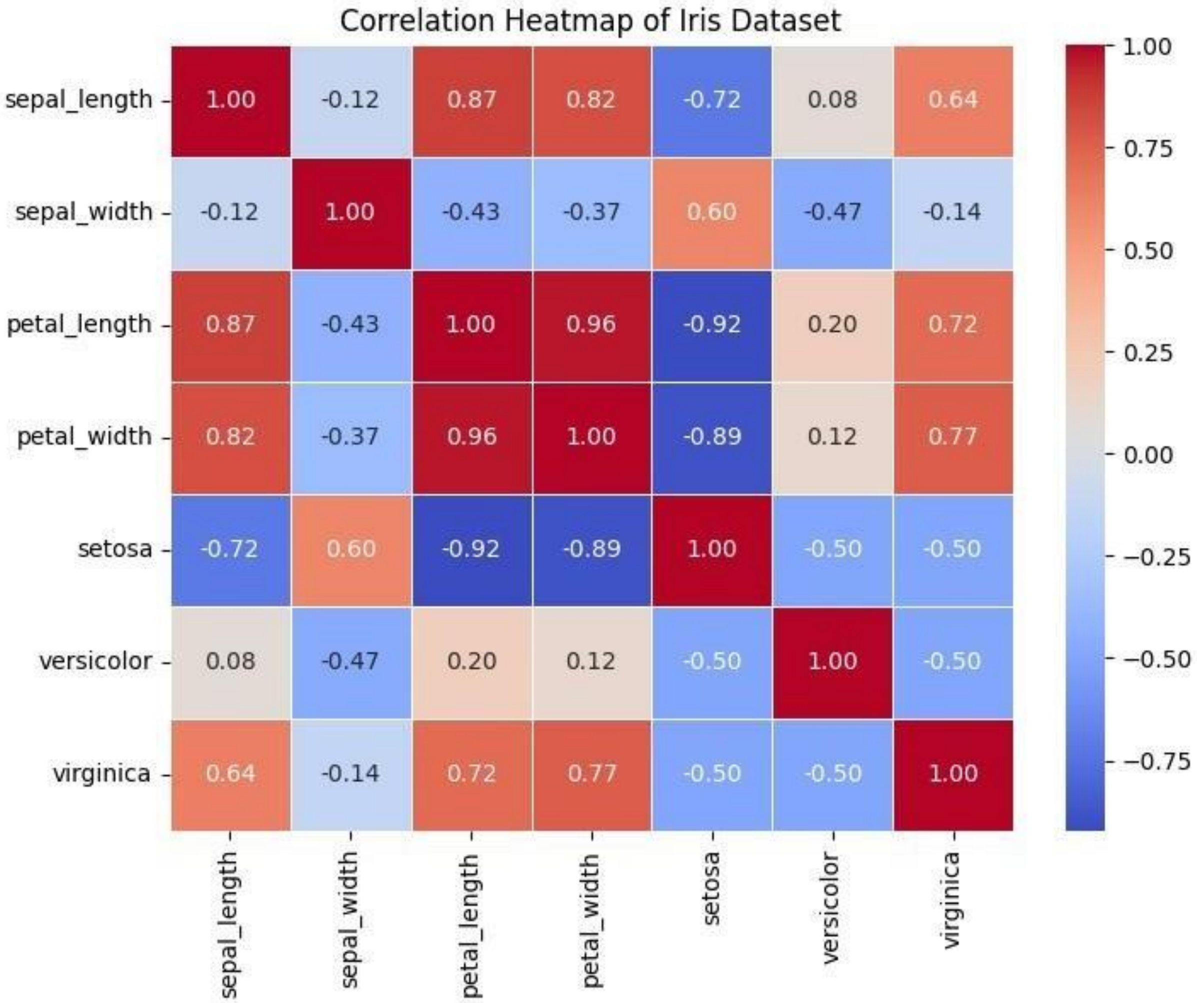




Boxplot of Sepal Length by Species



Violin Plot of Petal Length by Species


Correlation Heatmap of Iris Dataset
sepal_length
sepal_width
petal_length
petal_width
setosa
versicolor
virginica
1.00 -0.12 0.87 0.82 -0.72 0.08 0.64
-0.12 1.00 -0.43 -0.37 0.60 -0.47 -0.14
0.87 -0.43 1.00 0.96 -0.92 0.20 0.72
0.82 -0.37 0.96 1.00 -0.89 0.12 0.77
-0.72 0.60 -0.92 -0.89 1.00 -0.50 -0.50
0.08 -0.47 0.20 0.12 -0.50 1.00 -0.50
0.64 -0.14 0.72 0.77 -0.50 -0.50 1.00
sepal_length
sepal_width
petal_length
petal_width
setosa
versicolor
virginica
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

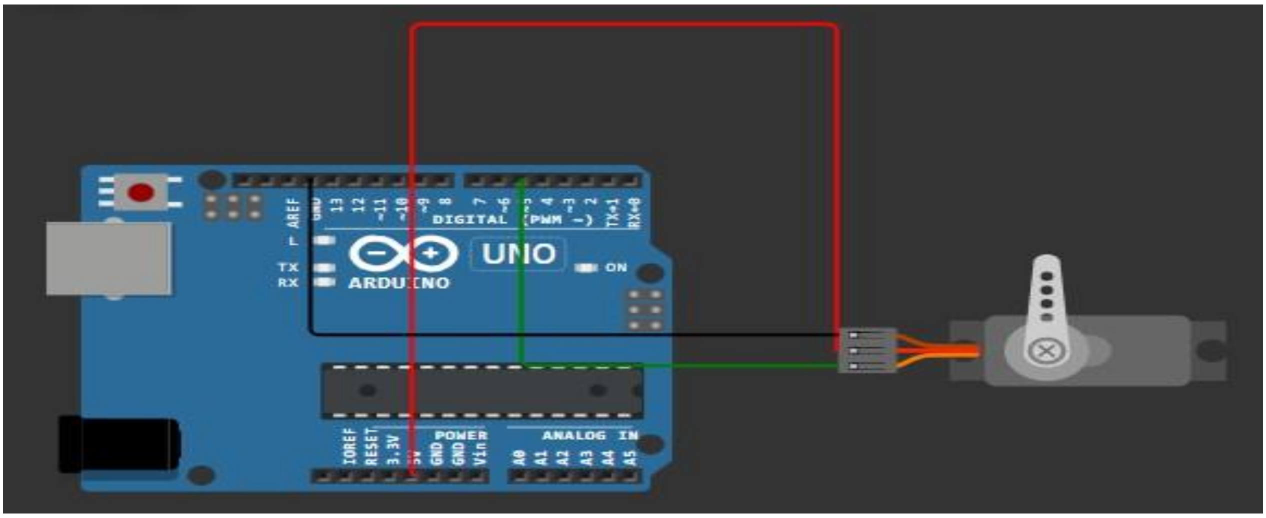
Assignment 8
Servo Moto (basics of Arduino, Servo)

Software Used: WOKWI

Components

Component Name	Quantity	Description
Arduino Uno R3	1	Micro Controller Board
Servo	1	Any motor-driven system with a feedback element built in.

Circuit Design



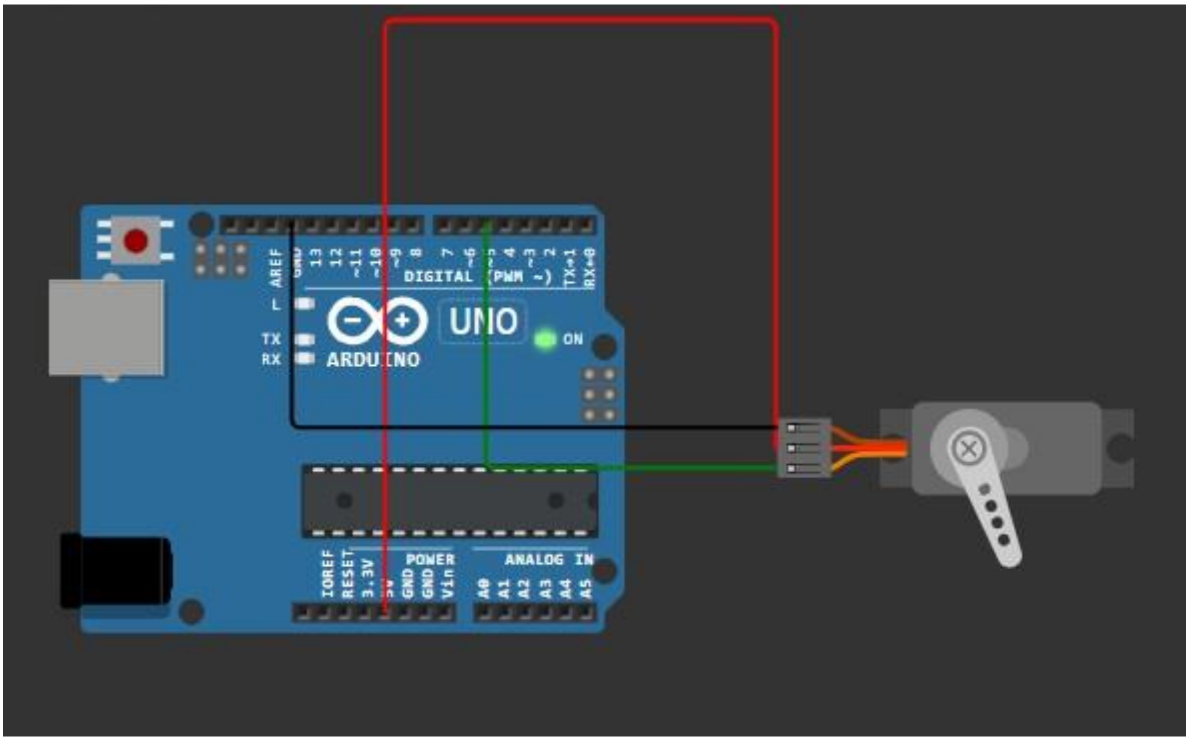
Component Design

Name	Quantity	Pin	Connection
Servo	1	V+	5V
		PWM	5
		GND	GND

Code

```
#include <Servo.h>
const int servoPin = 5;
Servo servo;
void setup() {
  servo.attach(servoPin, 500, 2400);
}
int pos = 0;
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    servo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    servo.write(pos);
    delay(15);
  }
}
```

Output



Assignment 9:

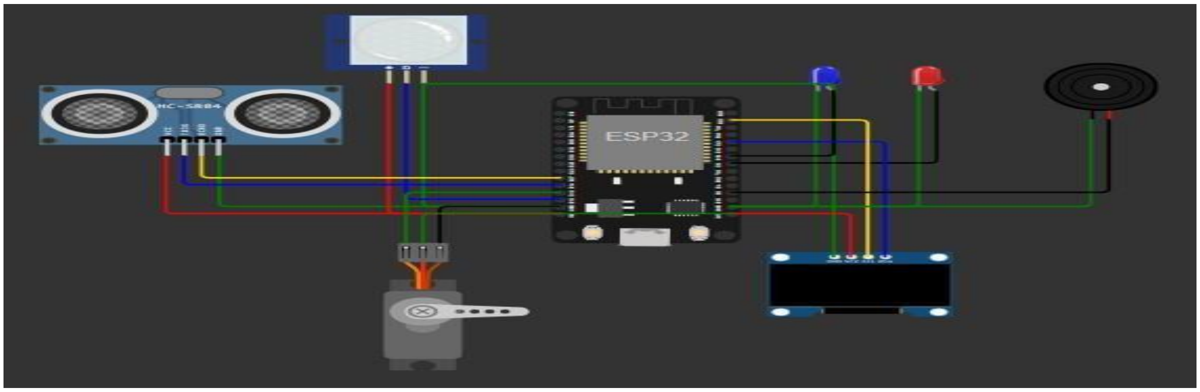
Smart Hand Sanitizer (basics of Arduino, Servo, HC-SR04 Ultrasonic Distance Sensor, Buzzer, LED, ESP32 Simulation, SSD1306 OLED display)

Software Used:WOKWI

Components

Component Name	Quantity	Description
PIR Motion Sensor	1	an electronic sensor that measures infrared (IR) light radiating from objects in its field of view
Servo	1	Any motor-driven system with a feedback element built in
HC-SR04 Ultrasonic Distance Sensor	1	High-Conductance Ultrasonic Sensor consists of a transmitter and receiver.
Buzzer	1	An efficient component to include the features of sound in our system or project
LED	2	LED(Red), LED(Blue)
ESP32 Simulation	1	a popular Wi-Fi and Bluetooth-enabled microcontroller, widely used for IoT Projects
SSD1306 OLED display	1	an OLED that is controlled by the SSD1306 micro-chip driver, which acts as a bridge between the display matrix and the microcontroller.

Circuit Design



Component Connection

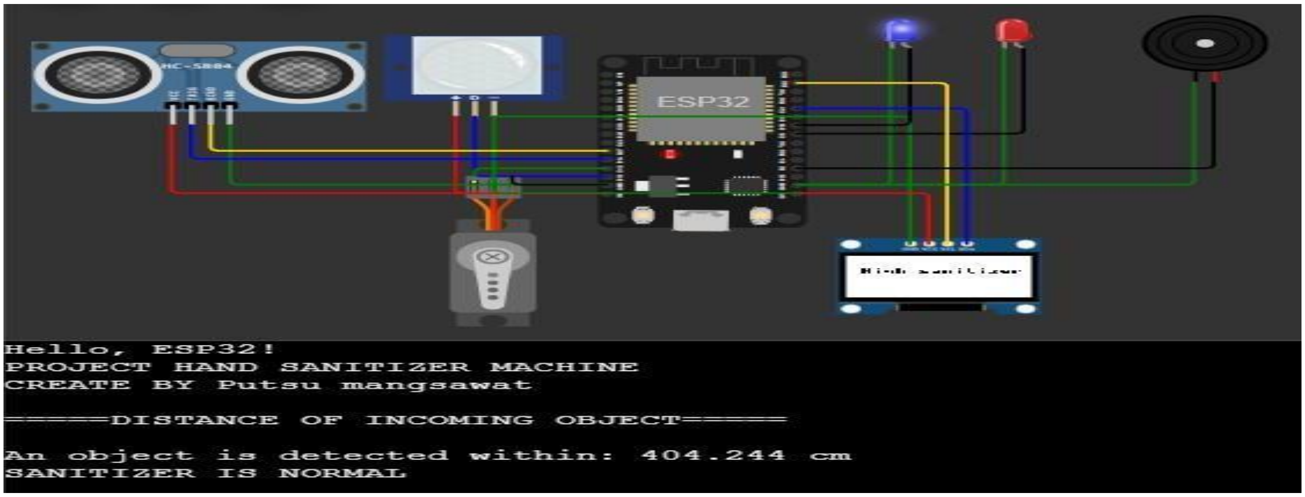
Name	Quantity	Pin	Connection
PIR	1	GND	GND
		VCC	5V
		OUT	13
OLED	1	GND	GND
		VCC	5V
		SCL	22
		SDA	21
Servo	1	GND	GND
		V+	5V
		PWM	12
Buzzer	1	bz 1:1	GND
		bz 1:2	2
LED1	1	Anode (positive pin)	18
		Cathode (negative pin)	GND
LED2	1	Anode (positive pin)	5
		Cathode (negative pin)	GND
HC-SR04 Ultrasonic Distance Sensor	1	VCC	5V
		TRIG	14
		ECHO	27
		GND	GND

Code

```
print("Hello, ESP32!")
print("PROJECT HAND SANITIZER MACHINE")
print("CREATE BY Putsu mangsawat")

# Import all libraries import
ultrasensor_library import
oled_library
from machine import Pin, SoftI2C, PWM
from utime import sleep

# Declare Pin
led_blue = Pin(18, Pin.OUT)
led_red = Pin(5, Pin.OUT)
TRIG = Pin(14, Pin.IN) ECHO
= Pin(27, Pin.OUT)
Buzzer_Pin = Pin(2, Pin.OUT) pir
= Pin(13, Pin.IN)
servoPin = Pin(12, Pin.OUT)
pin_oled = SoftI2C(scl=Pin(22), sda=Pin(21))

# Let's create a name for our OLED screen #
name = library name, class name
screen = oled_library.SSD1306_I2C(width=128, height=64, i2c=pin_oled)

# Declare object name for sensors with libraries
sensor_detected = ultrasensor_library.HCSR04(trigger_pin=TRIG, echo_pin=ECHO)

# Declare object name for servo motor
servo_motor = PWM(servoPin, freq=50)

# Function to move servo to the left def
move_servo_left():
  # Duty cycle for left position (adjust according to your servo)
  servo_motor.duty(40)  # Example duty cycle, adjust as needed
  sleep(1)  # Adjust time as needed
  # Duty cycle for stopping the servo
  servo_motor.duty(0)

# Main program while
True:
  # Ultrasensor part
  print('\n=====DISTANCE OF INCOMING OBJECT=====\n')
  distance_in_cm = sensor_detected.distance_cm()
  print('An object is detected within:', distance_in_cm, 'cm')

  # Buzzer part
  if distance_in_cm > 20: print("SANITIZER
    IS NORMAL")
    screen.fill(1)
```

```
        screen.text("High sanitizer level:/", 10, 20, 0) screen.show()
        led_blue.on()
        sleep(0.5)
        led_blue.off()
        sleep(0.5)

    elif 10 <= distance_in_cm < 20:
        print("SANITIZER IS LOW")
        tone_buzzer = PWM(Buzzer_Pin, freq=1000, duty=50) sleep(0.05)
        tone_buzzer = PWM(Buzzer_Pin, freq=1000, duty=0) sleep(0.05)
        screen.fill(1)
        screen.text("Low sanitizer level:/", 10, 20, 0) screen.show()
        led_red.on()  # Turn on the red LED when water level is low
        sleep(0.2)
        led_red.off()
        sleep(0.2)

    motion = pir.value()
    if motion == 1:
        print("\n\tPlace a hand\n") screen.fill(1)
        screen.text("Place a hand:/", 10, 20, 0)
        screen.text("!!!", 40, 40, 0)
        screen.show()  led_blue.on()
        sleep(2)
        led_blue.off()

        # Move servo to the left when motion is detected move_servo_left()

        sleep(2)
```

Output



```
Hello, ESP32!
PROJECT HAND SANITIZER MACHINE
CREATE BY Putsu mangsawat

======DISTANCE OF INCOMING OBJECT======

An object is detected within: 404.244 cm
SANITIZER IS NORMAL
```