**Simulation of multiple LED blinking control over the control action**

EXPERIMENT NO (HW/SW): 1

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

Aim of this experiment is to regulate the blinking frequency, pattern, and synchronization of several LEDs according to predefined input conditions using Arduino board.

**SOFTWARE USED:**

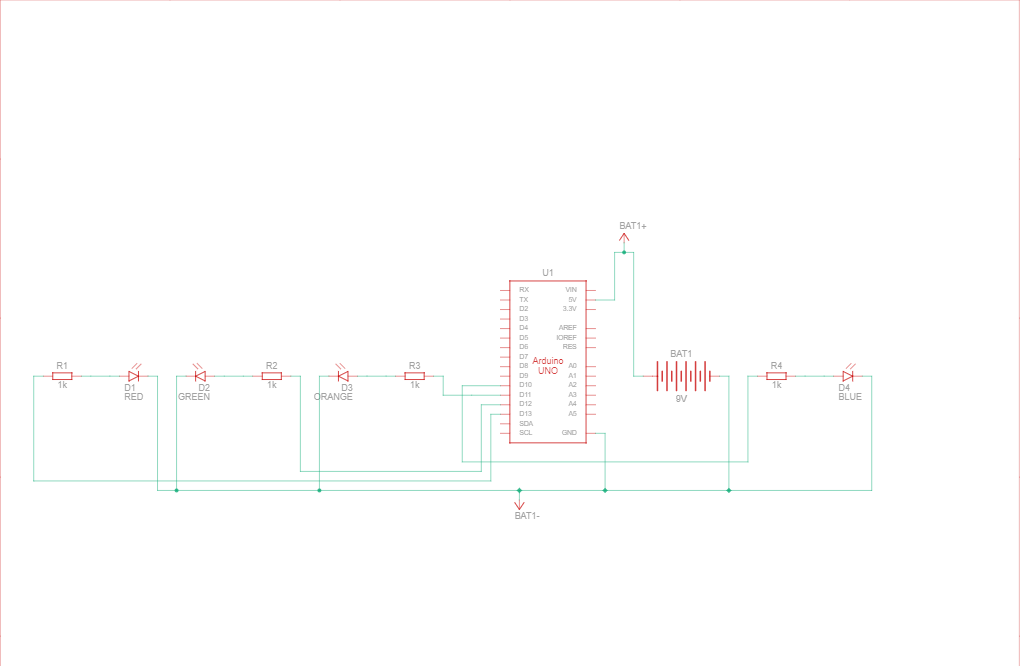
Tinkercad

**DESIGN AND THEORY:**

Case 1:

The below circuit is designed in a such way that one light glow then switches off and then the next one glows and switches off (one after the other) with 1000 millisecond interval. The time interval (S) can be changed by changing the code.

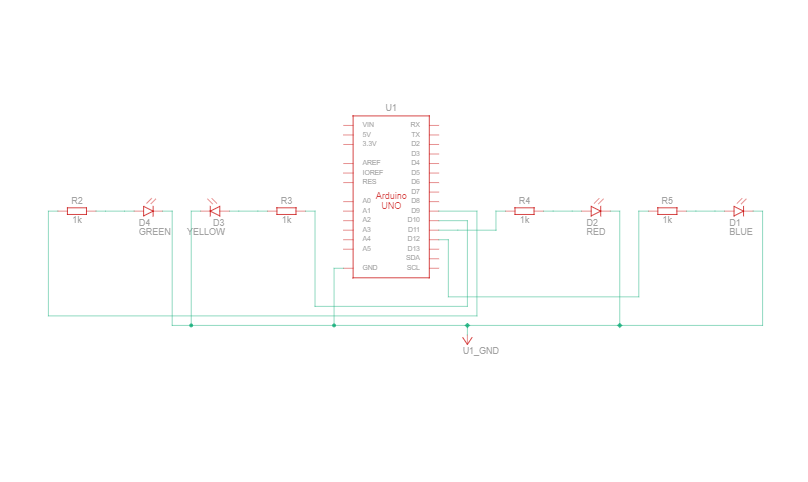
Circuit diagram:



Case 2:

This circuit is designed in such a way that all the lights glow and switch simultaneously. The inputs to the LEDs are set high for a certain amount of time and turned to low which leads to the switching on and off of the LEDs. In this case S is 1000 millisecond.

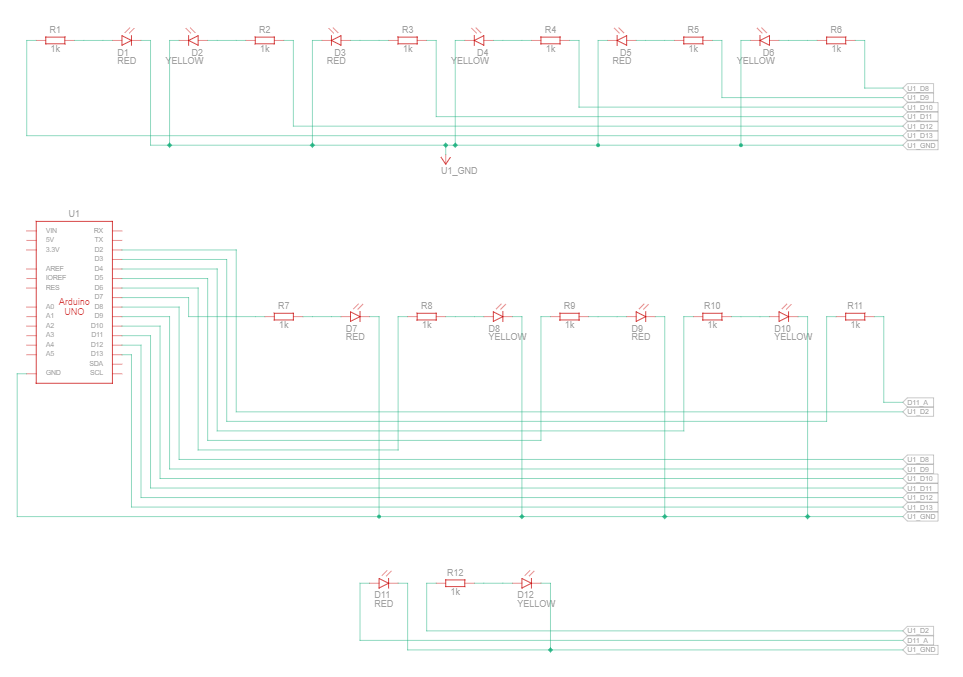
Circuit diagram:



Case 3:

This circuit is connected in such a way that the odd lights glow all at once and switch off and then the even lights glow all at once and switch off in the time interval of 1000 millisecond(S)

Circuit diagram:

****

OBSERVATION:

Case 1:

As you can see from the figure 1.1 the led glows one by one at a time interval of 1000 millisecond(S).

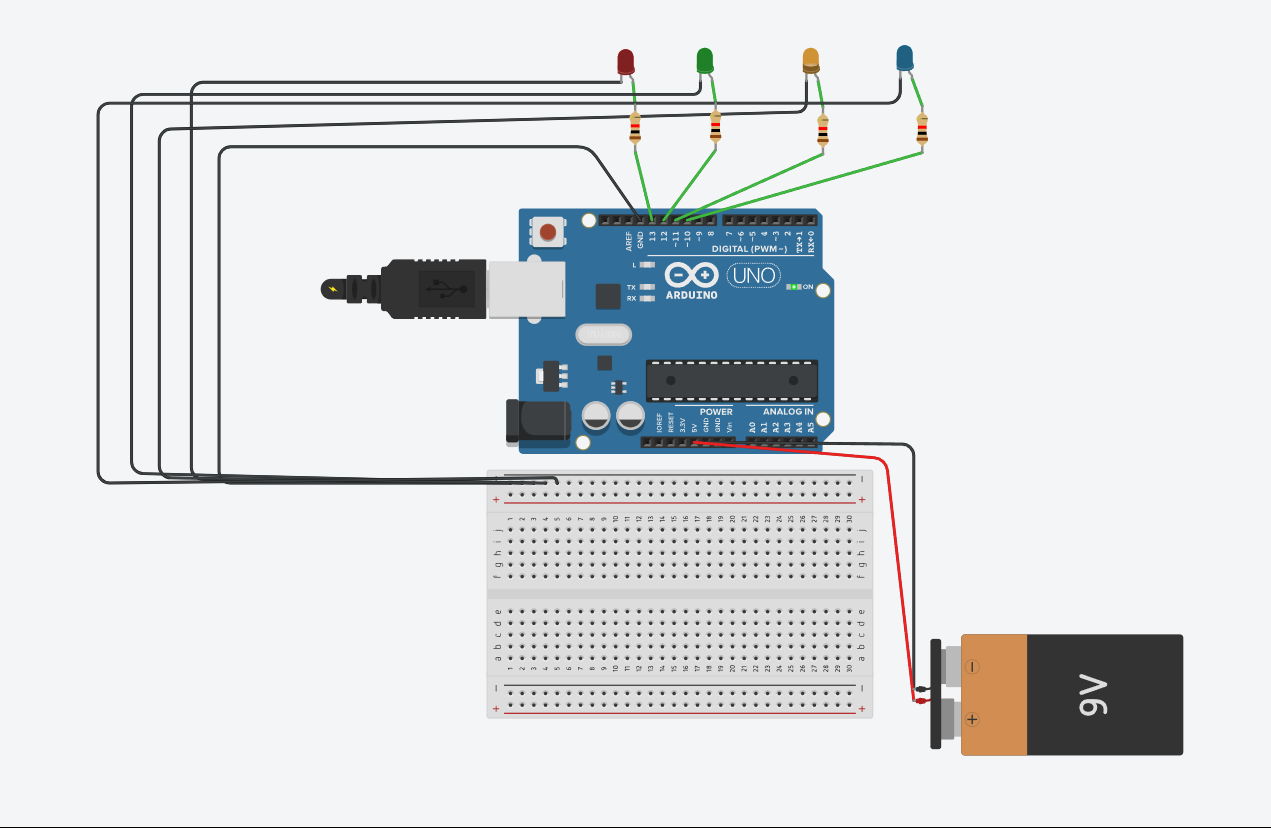


Figure 1.1

Code:

void setup()

{

pinMode(13, OUTPUT);

pinMode(12, OUTPUT);

pinMode(11, OUTPUT);

pinMode(10, OUTPUT);

pinMode(9, OUTPUT);

}

void loop()

{

digitalWrite(13, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(13, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(11, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(11, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(10, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(10, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(9, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(9, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

Case 2:

As you can see from the figure 1.2 all the LEDS are glowing simultaneously and turning off at a time interval of 1000 millisecond.

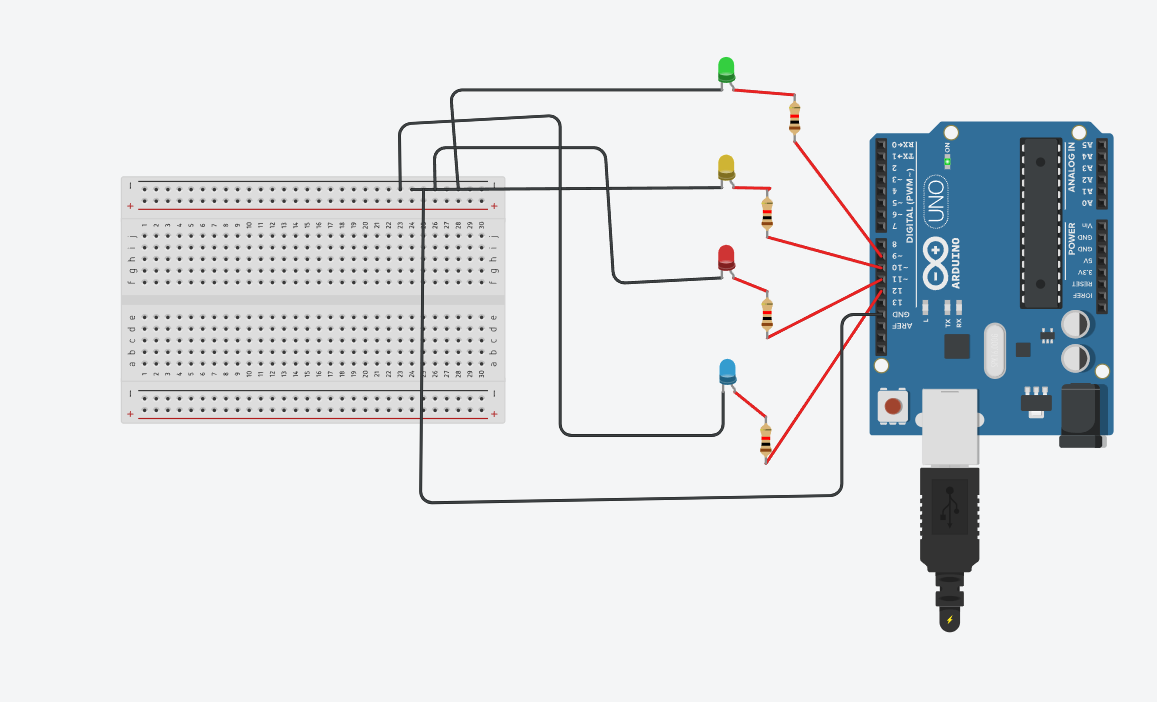
****

Figure 1.2

Code:

void setup()

{

pinMode(12, OUTPUT);

pinMode(11, OUTPUT);

pinMode(10, OUTPUT);

pinMode(9, OUTPUT);

}

void loop()

{

digitalWrite(12, HIGH);

digitalWrite(11, HIGH);

digitalWrite(10, HIGH);

digitalWrite(9, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, LOW);

digitalWrite(11, LOW);

digitalWrite(10, LOW);

digitalWrite(9, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

Case 3:

As you can see from the figure 1.3 all the even LEDS are glowing simultaneously and turned off. Then all the odd LEDS are glowing simultaneously and turned off at a time interval of 1000 millisecond.

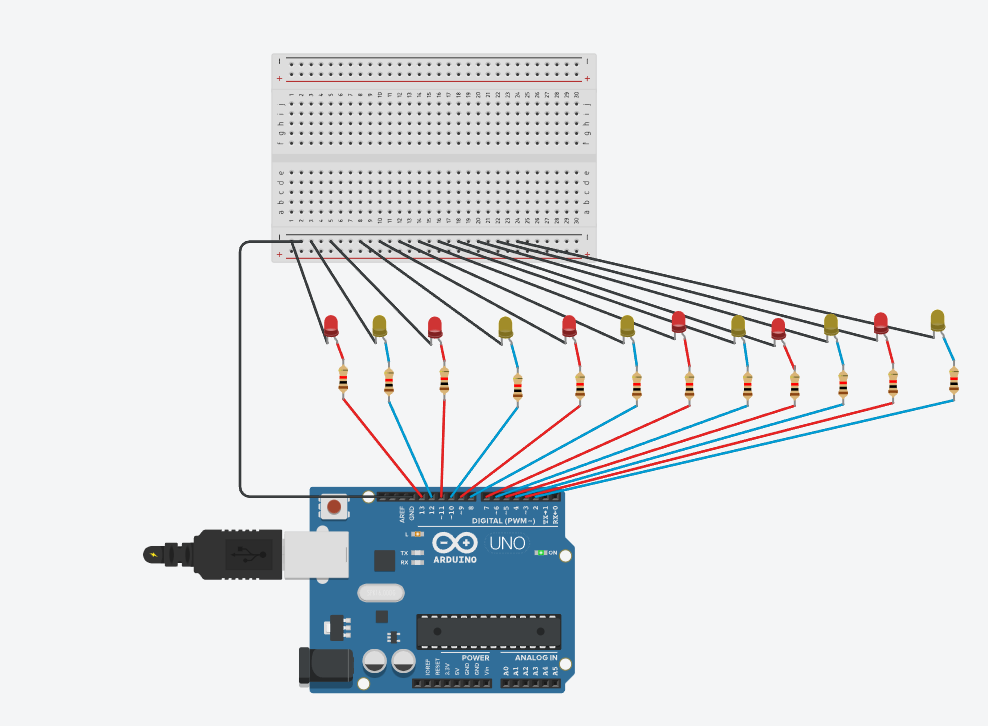


Figure 1.3

Code:

// C++ code

void setup()

{

pinMode(13, OUTPUT);

pinMode(11, OUTPUT);

pinMode(9, OUTPUT);

pinMode(7, OUTPUT);

pinMode(5, OUTPUT);

pinMode(3, OUTPUT);

pinMode(12, OUTPUT);

pinMode(10, OUTPUT);

pinMode(8, OUTPUT);

pinMode(6, OUTPUT);

pinMode(4, OUTPUT);

pinMode(2, OUTPUT);

}

void loop()

{

digitalWrite(13, HIGH);

digitalWrite(11, HIGH);

digitalWrite(9, HIGH);

digitalWrite(7, HIGH);

digitalWrite(5, HIGH);

digitalWrite(3, HIGH);

digitalWrite(12, LOW);

digitalWrite(10, LOW);

digitalWrite(8, LOW);

digitalWrite(6, LOW);

digitalWrite(4, LOW);

digitalWrite(2, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, HIGH);

digitalWrite(10, HIGH);

digitalWrite(8, HIGH);

digitalWrite(6, HIGH);

digitalWrite(4, HIGH);

digitalWrite(2, HIGH);

digitalWrite(13, LOW);

digitalWrite(11, LOW);

digitalWrite(9, LOW);

digitalWrite(7, LOW);

digitalWrite(5, LOW);

digitalWrite(3, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

**INFERENCE:**

From this experiment I understand how to control over the control action of LED using tinkercad.

**Hardware design and simulation of temperature sensor to measure the environmental temperature variation with LED glow, LCD display and buzzer for alarming signal.**

EXPERIMENT NO (HW/SW): 1

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

**DATE:**

**AIM:**

Aim of this experiment is to measure the temperature, LEDs to indicate temperature thresholds, an LCD display to show the measured temperature, and a buzzer to signal an alarm for specific temperature conditions.

**SOFTWARE USED:**

Tinkercad

**DESIGN AND THEORY:**

The temperature sensor senses the temperature of the environment. and the LCD display is to indicate the temperature of the environment. the LED will grow when it reaches a certain temperature level. The buzzer will alarm when it reaches the threshold value. In this experiment I set the threshold value as 40 degrees. The circuit diagram of the experiment is given in figure 2.1.

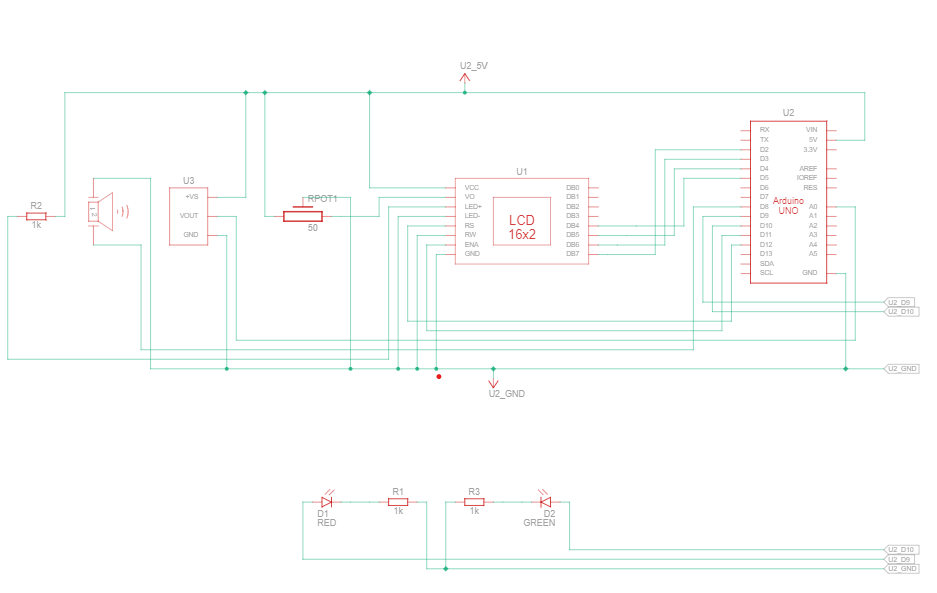


Figure 2.1

OBSERVATION:

When the temperature of the environment is above 40 degree the green LED stop glowing and red LED start glowing. The buzzer starts alarming. LCD display displays the temperature of the environment. Figure 2.2 shows the output of the simulation.

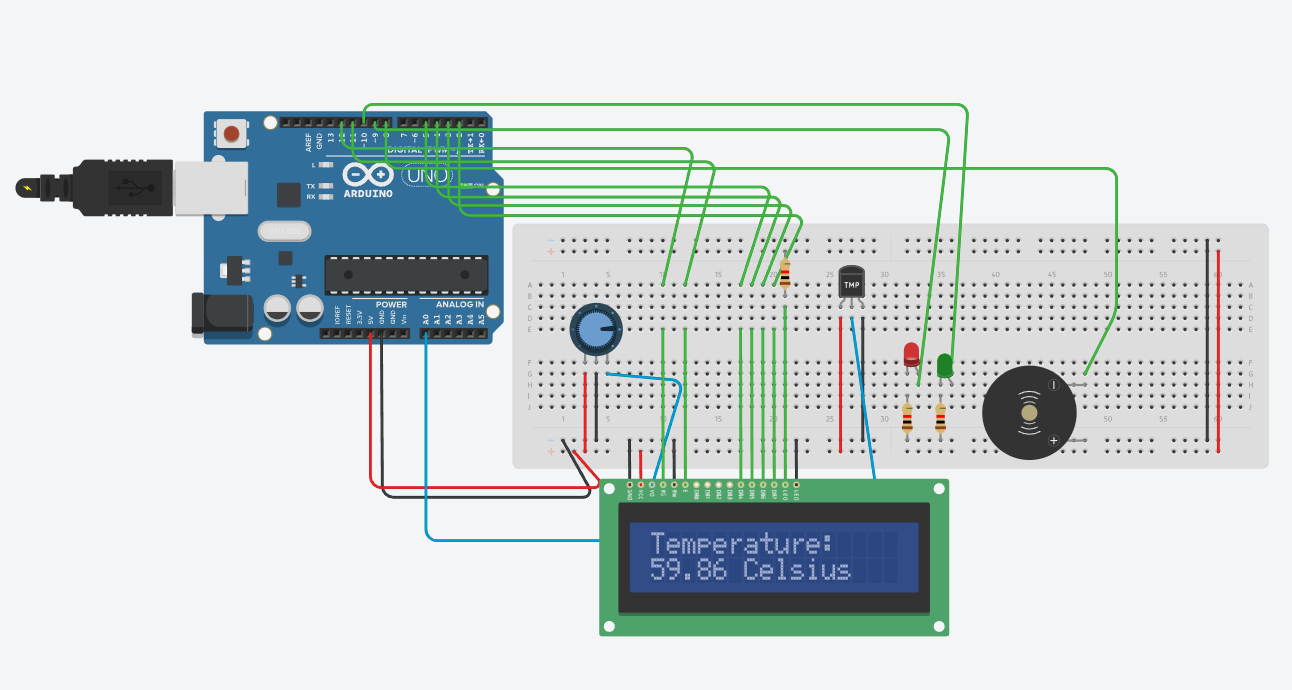


Figure 2.2

**Code:**

// C++ code

void setup()

{

pinMode(13, OUTPUT);

pinMode(12, OUTPUT);

pinMode(11, OUTPUT);

pinMode(10, OUTPUT);

pinMode(9, OUTPUT);

}

void loop()

{

digitalWrite(13, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(13, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(11, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(11, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(10, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(10, LOW);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(9, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(9, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

// C++ code

void setup()

{

pinMode(12, OUTPUT);

pinMode(11, OUTPUT);

pinMode(10, OUTPUT);

pinMode(9, OUTPUT);

}

void loop()

{

digitalWrite(12, HIGH);

digitalWrite(11, HIGH);

digitalWrite(10, HIGH);

digitalWrite(9, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(12, LOW);

digitalWrite(11, LOW);

digitalWrite(10, LOW);

digitalWrite(9, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

// Allows communication with alphanumerical liquid crystal displays (LCDs).

#include <LiquidCrystal.h

//Initialize the library with the numbers of the interface pins.

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

//This is the Arduino Pin that will read the sensor output.

int sensePin = A0;

//The variable we will use to store the sensor input.

int sensorInput;

//The variable we will use to store temperature in degrees.

double temp;

// Variable of the red led associated with the 9th pin

int redLed = 9 ;

// same, but for the green led with the 10th pin

int greenLed = 10 ;

void setup()

{

// pin of the red LED

pinMode(9, OUTPUT);

// pin of the green LED

pinMode(10, OUTPUT) ;

//Initialize the LCD's number of columns and rows.

lcd.begin(16, 2);

//Start the Serial Port at 9600 baud (default).

Serial.begin(9600);

}

void loop()

{

//Set the cursor to column 0, line 0

lcd.setCursor(0, 0);

//Read the analog sensor and store it.

sensorInput = analogRead(A0);

float volt = sensorInput \* 5;

//Multiply by 5V to get voltage.

volt /= 1024;

Serial.print(volt);

Serial.println(" volts ");

//Subtract the offset.

temp = volt - 0.5;

//Convert to degrees.

temp = temp \* 100;

// recovering the differents result of

// the circuit in the serial monitor

Serial.print("Temperature ") ;

Serial.print(temp);

Serial.println(" Celsius");

// printing on the lcd screen the word "temperature"

lcd.print("Temperature: ");

//Set the cursor to column 0, line 1

lcd.setCursor(0, 1);

// Printing the temperature on the lcd screen

lcd.print(temp);

// printing the name of the scale used for temperature

lcd.print(" Celsius");

// if the temperature (var associated: temp) is superior

// to 40, then we enter the if loop

if (temp >= 40)

{

// turning on the red LED, stated as 'high'

digitalWrite(redLed, HIGH);

// turning off the green led, stated as 'low'

digitalWrite(greenLed, LOW);

//INPUT - FREQUENCY - TIME THAT LASTS

tone(8, 440, 200);

}

// otherwise, if the temperature is below 40

else if (temp < 40)

{

// turning on the green led

digitalWrite(greenLed, HIGH);

// and turning off the red led

digitalWrite(redLed, LOW);

noTone(8);

}

delay(1000);

}

**INFERENCE:**

I learnt how to simulate a temperature sensor to measure the environmental temperature variation with LED glow, LCD display and buzzer for alarming signal.

**Hardware design and simulation of gas sensor for CO2 gas detection with LED glow, LCD display and buzzer for alarming signal.**

EXPERIMENT NO (HW/SW): 1

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

To construct and simulate a gas sensor for CO2 gas detection with LED GLOW, LCD display and buzzer for alarming.

**SOFTWARE USED:**

Tinkercad

Arduino ide (coding)

COMPONENTS REQUIRED FOR HARDWARE:

1. Gas sensor
2. Buzzer
3. LED
4. Arduino
5. Lighter/match box

**DESIGN AND THEORY:**

The gas sensor senses the concentration of carbon monoxide gas in the environment. And the LCD display is to indicate the concentration of CO2 of the environment. The both LED will grow when it reaches a certain threshold level. The buzzer will alarm when it reaches the threshold value. In this experiment I set the threshold value as 400. The circuit diagram of the experiment is given in figure 3.1.

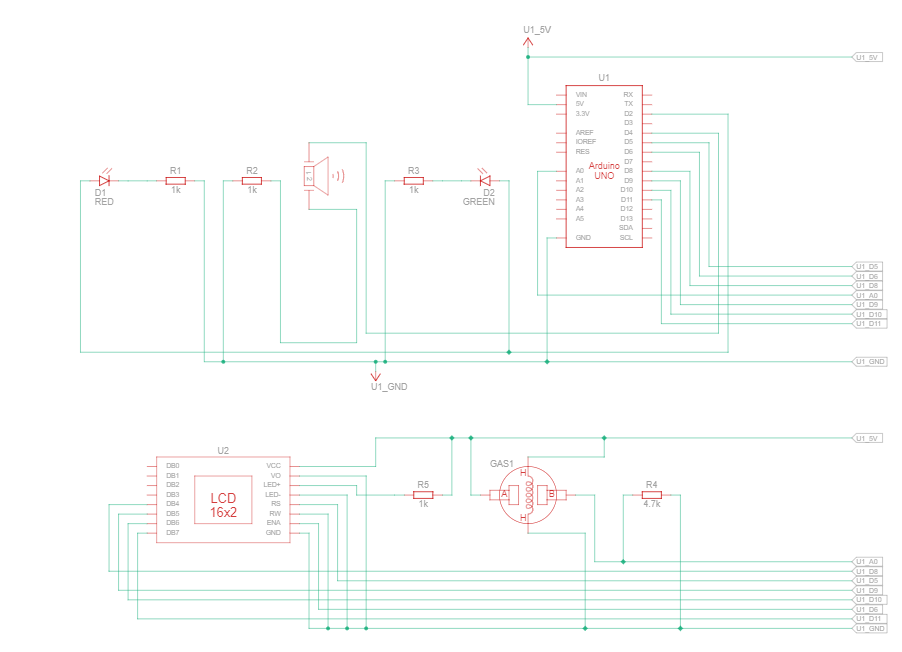


Figure 3.1

OBSERVATION OF SIMULATION:

When the gas value of the environment is above 400 the green LED and red LED start glowing. The buzzer starts alarming. LCD display displays that “ALERT”. From this we can prevent fire accident. Figure 3.2 shows the output of the simulation.

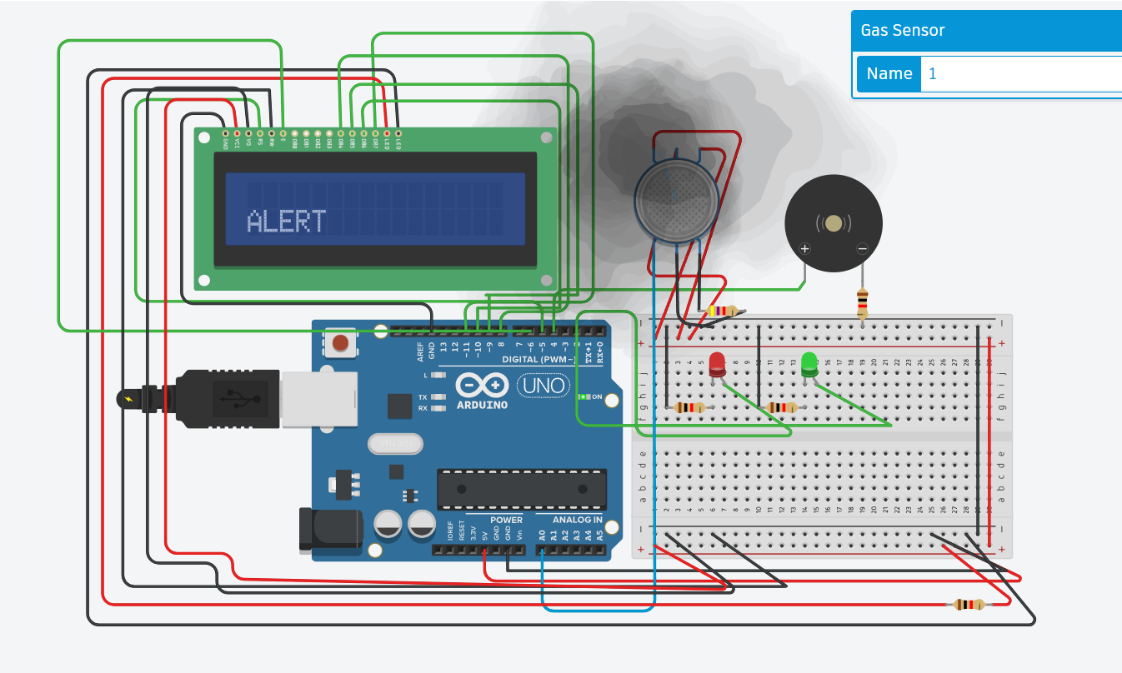


Figure 3.2

OBSERVATION OF HARDWARE:

When the gas value of the environment is above 400. the yellow LED stops glowing and red LED start glowing. The buzzer starts alarming. From this we can prevent fire accident. Figure 3.3 shows the output of the hardware.

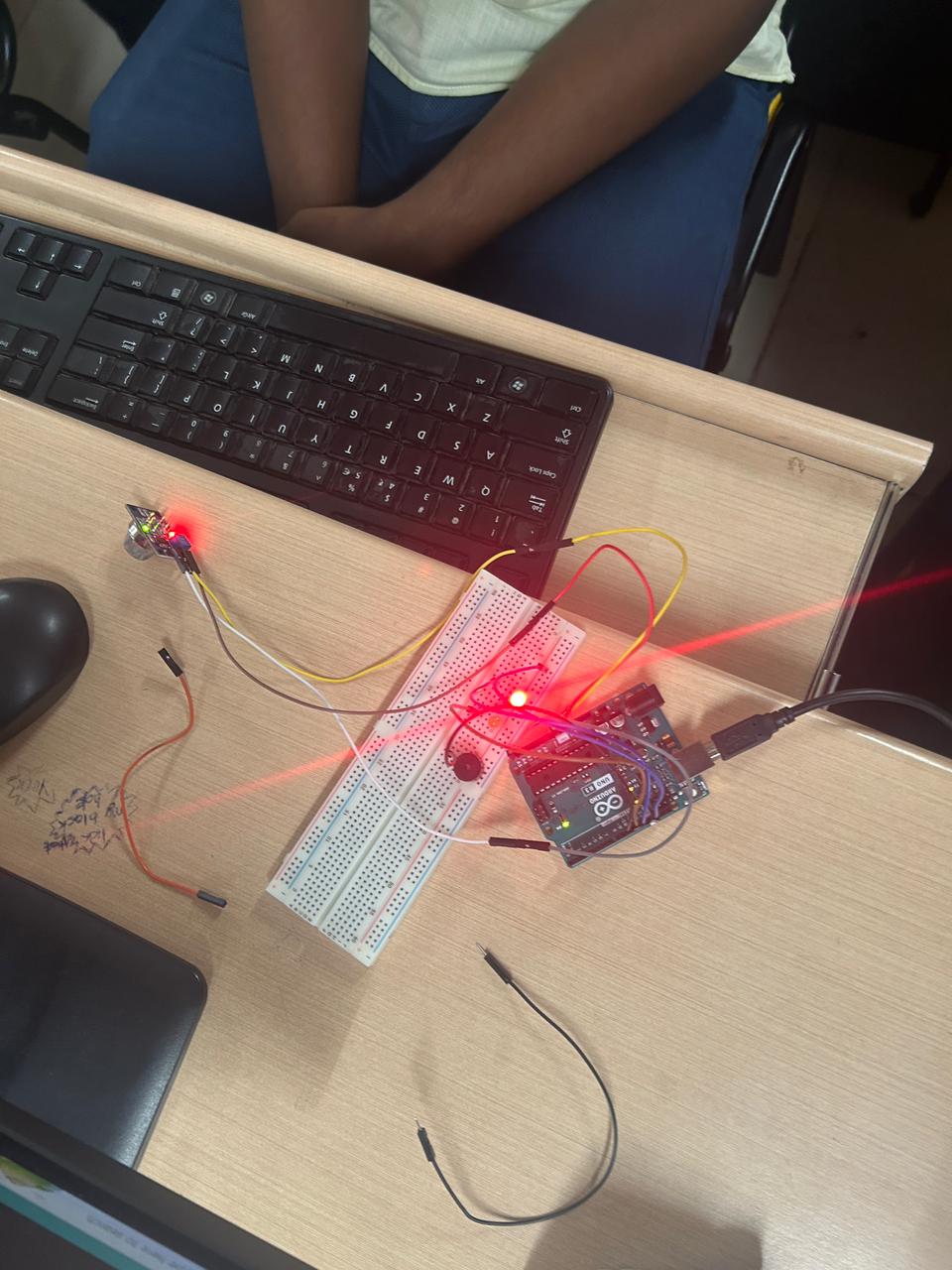


Figure 3.3

**Code:**

#include <LiquidCrystal.h>

LiquidCrystal lcd(5,6,8,9,10,11);

int redled = 2;

int greenled = 3;

int buzzer = 4;

int sensor = A0;

int sensorThresh = 400;

void setup()

{

pinMode(redled, OUTPUT);

pinMode(greenled,OUTPUT);

pinMode(buzzer,OUTPUT);

pinMode(sensor,INPUT);

Serial.begin(9600);

lcd.begin(16,2);

}

void loop()

{

int analogValue = analogRead(sensor);

Serial.print(analogValue);

if(analogValue>sensorThresh)

{

digitalWrite(redled,HIGH);

digitalWrite(greenled,LOW);

tone(buzzer,1000,10000);

lcd.clear();

lcd.setCursor(0,1);

lcd.print("ALERT");

delay(1000);

}

else

{

digitalWrite(greenled,HIGH);

digitalWrite(redled,LOW);

noTone(buzzer);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("SAFE");

delay(1000);

}

}

NOTE: In hardware we didn’t use LCD display. So we don’t want to import lcd display library.

**INFERENCE:**

I learnt how to assemble and simulate a gas sensor that alert the user by buzzer and LED.

**Hardware design and simulation of soil moisture sensor with LED glow, LCD display and buzzer for alarming signal.**

EXPERIMENT NO (HW/SW): 1

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

To construct and simulate a soil moisture sensor to detect the moisture content in the soil with the help of LED GLOW and buzzer for alarming.

**SOFTWARE USED:**

Tinkercad

Arduino ide (coding)

COMPONENTS REQUIRED FOR HARDWARE:

1. Moisture sensor
2. LED
3. Arduino
4. water

**DESIGN AND THEORY:**

The moisture sensor detects the moisture content in the system/environment. And there are four LED that will glow when it reaches a certain threshold level. this experiment when the value is less than300 red LED will glow. If the value is between 300<500 blue LED will glow. If the value is between 500<700 blue LED will glow. If the value is something else green LED will grow. The circuit diagram of the experiment is given in figure 4.1.

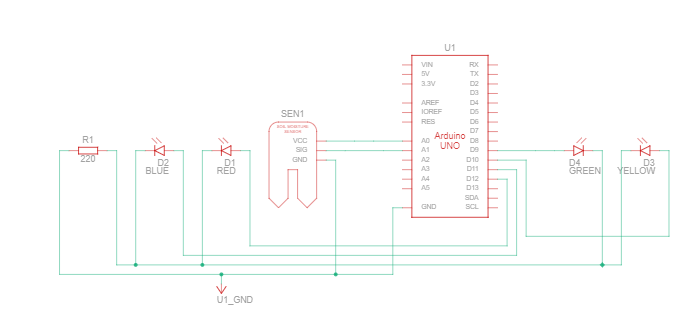


Figure 4.1

OBSERVATION OF SIMULATION:

* the LED start glowing when it reaches a certain value of moisture content.
* Code ran successfully
* The output is shown in figure 4.2

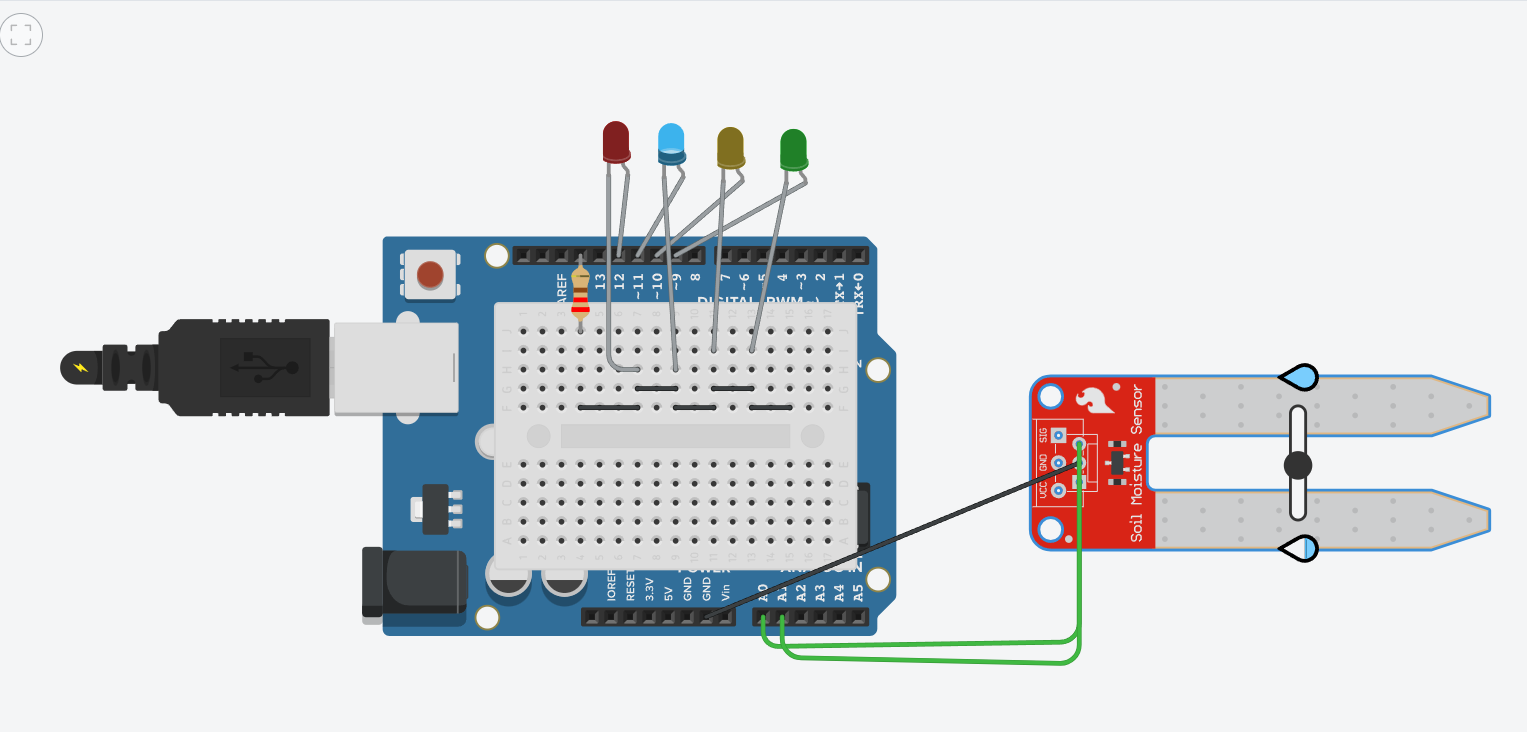
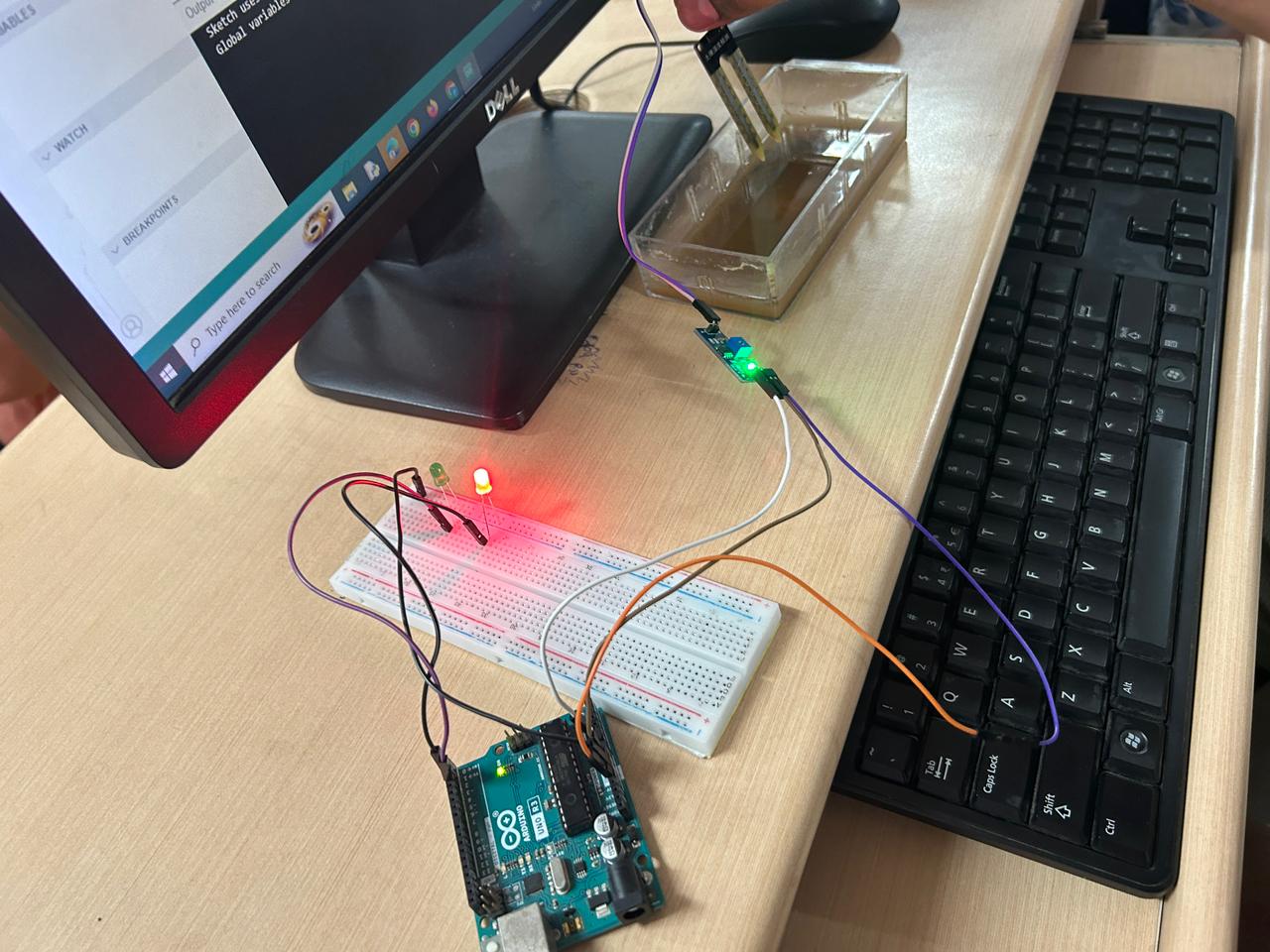


Figure 4.2

OBSERVATION OF HARDWARE:

* When the moisture content increase, the red LED will starts glowing.
* Code ran successfully



**Code:**

int moisture = 0;

void setup() {

pinMode(A0, OUTPUT);

pinMode(A1, INPUT);

Serial.begin(9600);

pinMode(9, OUTPUT);

pinMode(10, OUTPUT);

pinMode(11, OUTPUT);

pinMode(12, OUTPUT); }

void loop() {

digitalWrite(A0, HIGH);

delay(10);

moisture = analogRead(A1);

digitalWrite(A0, LOW);

Serial.println(moisture);

digitalWrite(9, LOW);

digitalWrite(10, LOW);

digitalWrite(11, LOW);

digitalWrite(12, LOW);

if (moisture < 200) {

digitalWrite(12, HIGH); }

else { if (moisture < 400) {

digitalWrite(11, HIGH); }

else { if (moisture < 600){

digitalWrite(10, HIGH);

} else {

if (moisture < 800) {

digitalWrite(9, HIGH); }

}

}

}

delay(100);

}

**INFERENCE:**

I learnt how to assemble and simulate a moisture sensor by using Arduino ide and tinkercad.

**IR Remote Control Home Automation using Arduino thinkercad simulation.**

EXPERIMENT NO (HW/SW): 1

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

**SIMULATION OF LDR SENSOR WITH LED GLOW, AND BUZZER FOR ALARMING SIGNAL.**

EXPERIMENT NO (HW/SW): 6

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

Aim of this experiment to Simulate LDR sensor with led glow, and buzzer for alarming signal.

**SOFTWARE USED:**

Tinkercad

**DESIGN AND THEORY:**

**Design Steps:**

1. **Connect the LDR in a Voltage Divider Configuration:**
   * Connect one leg of the LDR to the 5V on the Arduino.
   * Connect the other leg of the LDR to the A0 (analog input) on the Arduino.
   * Connect a resistor (10kΩ) from the A0 to GND on the Arduino.
2. **Connect the LED:**
   * Connect the positive leg of the LED to a digital pin (e.g., D2) on the Arduino.
   * Connect the negative leg of the LED to a current-limiting resistor and then to GND on the Arduino.
3. **Connect the Buzzer:**
   * Connect the positive terminal of the buzzer to a digital pin (e.g., D3) on the Arduino.
   * Connect the negative terminal of the buzzer to GND on the Arduino.
4. **Write Arduino Code:**
   * Use the Arduino IDE or Tinkercad Code Editor to write the code.
   * The threshold value (500 in this example) can be adjusted based on ambient light conditions. When the LDR value falls below the threshold, the LED turns on, and the buzzer emits a tone.
5. **Test the System:**
   * Upload the code to the Arduino in Tinkercad.
   * Use the simulation to test the system by adjusting the lighting conditions around the LDR.
6. **Adjust and Expand:**
   * Fine-tune the code and connections based on your specific components and requirements.
   * Expand the system by incorporating additional sensors or modifying the code for more complex behavior.

**THEORY:**

LDRs change their resistance based on the intensity of light falling on them. More light results in less resistance, and less light results in higher resistance.by this principle, when the light increase the buzzer starts alarming. The circuit diagram shown in figure 6.1

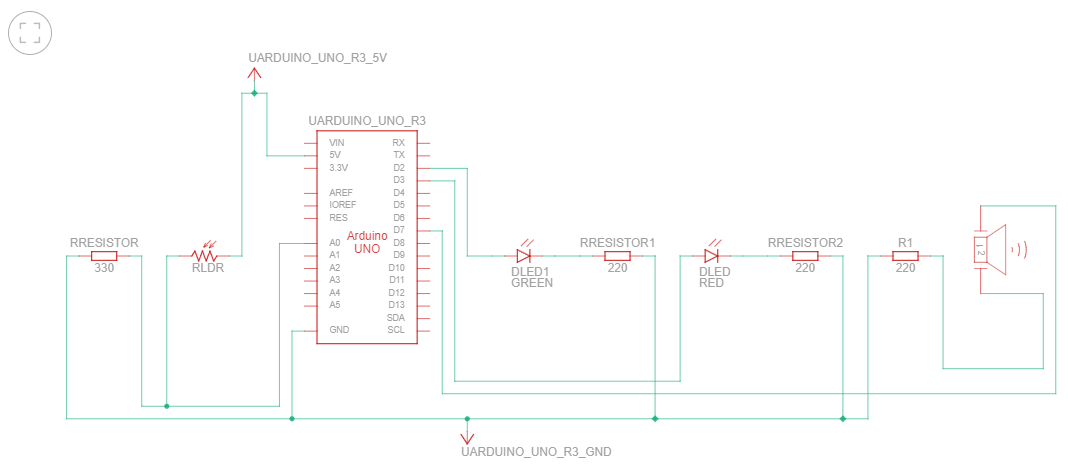
****

Figure 6.1

OBSERVATION OF SIMULATION:

* The red LED start glowing and green LED start glowing when the LDR gets light.
* Buzzer starts alarming when the LDR gets light
* Code ran successfully
* The output of the simulation is given in the figure 6.2

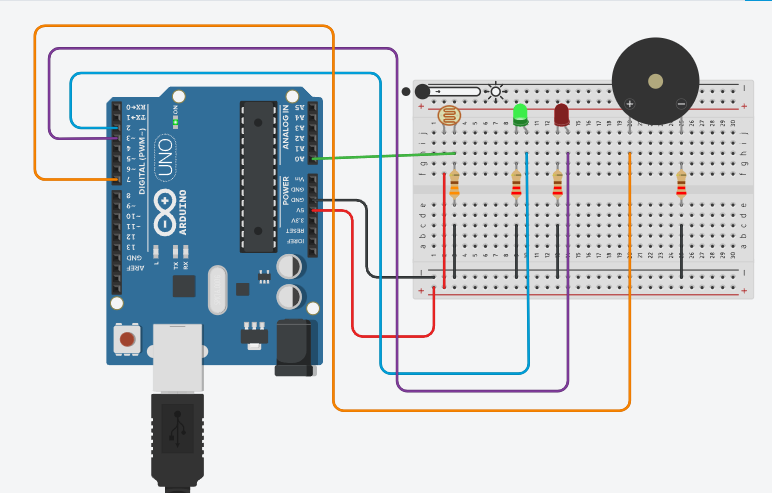


Figure 6.2

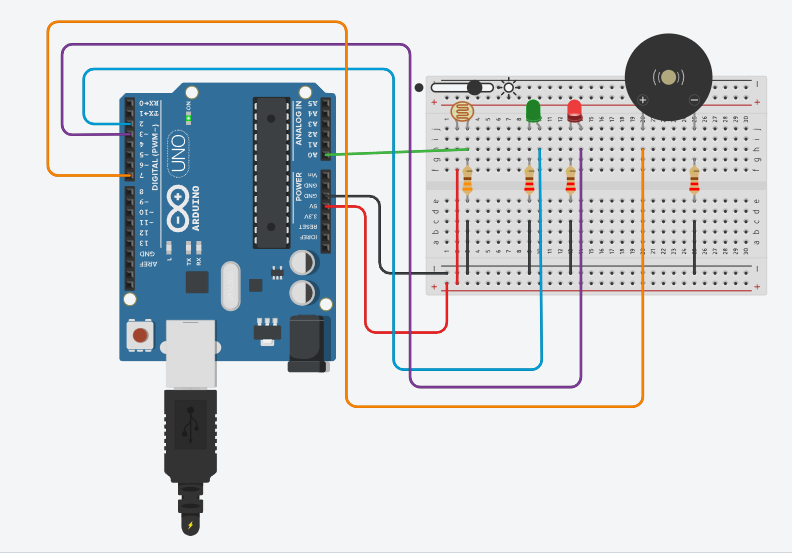


Figure 6.2

**Code:**

#define LDR A0

#define led\_green 2

#define led\_red 3

#define piezo 7

void setup()

{

Serial.begin (9600);

pinMode(LDR, INPUT);

pinMode (led\_green, OUTPUT);

pinMode (led\_red, OUTPUT);

pinMode (piezo, OUTPUT);

}

void loop()

{ int value = analogRead (LDR);

Serial.println (value);

delay (1000);

if (value < 250 )

{

digitalWrite (led\_green, HIGH);

digitalWrite (led\_red, LOW);

noTone (piezo);

delay (500); }

else {

digitalWrite (led\_green, LOW);

digitalWrite (led\_red, HIGH);

tone (piezo, 1000);

delay (500);

} }

**INFERENCE:**

I learnt how to simulate a LDR sensor by using Arduino in tinkercad.

**Hardware design and simulation of ultrasonic sensor for distance measurement with LED glow, LCD display and buzzer for alarming signal.**

EXPERIMENT NO (HW/SW): 7

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

Aim of this experiment to Simulate and assemble ultrasonic sensor with led’

**SOFTWARE USED:**

Tinkercad

**COMPONENTS USED:**

* Ultrasonic sensor
* Breadboard
* LED
* Arduino

**Design and Theory:**

**Design Steps:**

1. **Connect the Ultrasonic Sensor:**
   * Connect VCC and GND of the ultrasonic sensor to 5V and GND on the Arduino.
   * Connect the Trig pin of the sensor to a digital pin (e.g., D2).
   * Connect the Echo pin of the sensor to another digital pin (e.g., D3).
2. **Connect the LED:**
   * Connect the positive leg of the LED to a digital pin (e.g., D4) on the Arduino.
   * Connect the negative leg of the LED to a current-limiting resistor (220Ω) and then to GND on the Arduino.
3. **Connect the Buzzer:**
   * Connect the positive terminal of the buzzer to a digital pin (e.g., D5) on the Arduino.
   * Connect the negative terminal of the buzzer to GND on the Arduino.
4. **Connect the LCD Display:**
   * Connect the VSS and VDD pins of the LCD to GND and 5V on the Arduino.
   * Connect the V0 pin to the wiper of the potentiometer.
   * Connect the RS, RW, and E pins of the LCD to digital pins on the Arduino (e.g., D6, D7, D8).
   * Connect the D4 to D7 pins of the LCD to digital pins on the Arduino (e.g., D9, D10, D11, D12).
5. **Write Arduino Code:**
   * Use the Arduino IDE or Tinkercad Code Editor to write the code.
   * Include the "LiquidCrystal.h" library for interfacing with the LCD.

6. **Test the System:**

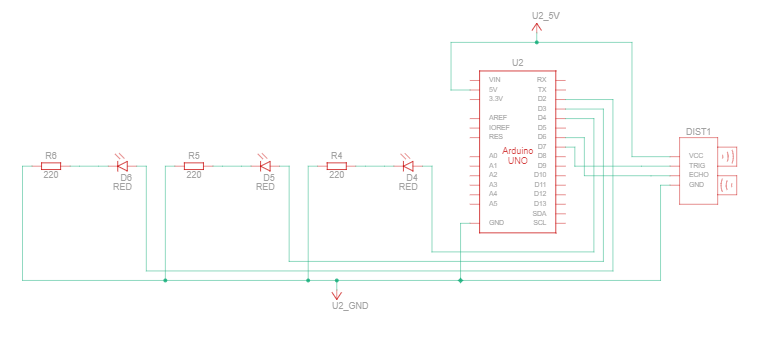
* + Upload the code to the Arduino in Tinkercad.
  + Use the simulation to test the system by moving objects in front of the ultrasonic sensor.

7. **Adjust and Expand:**

* + Fine-tune the code and connections based on your specific components and requirements.
  + Expand the system by incorporating additional features or sensors.

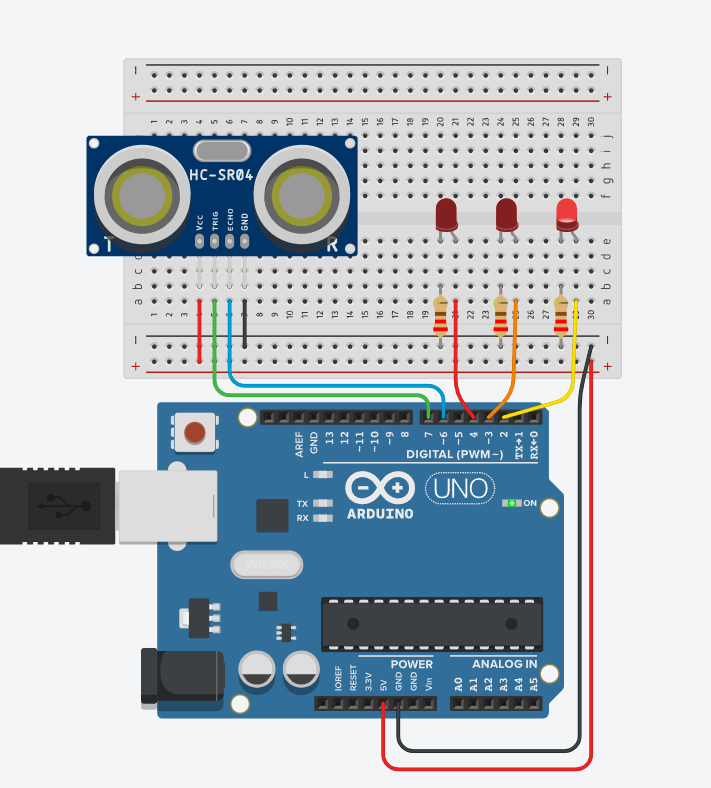
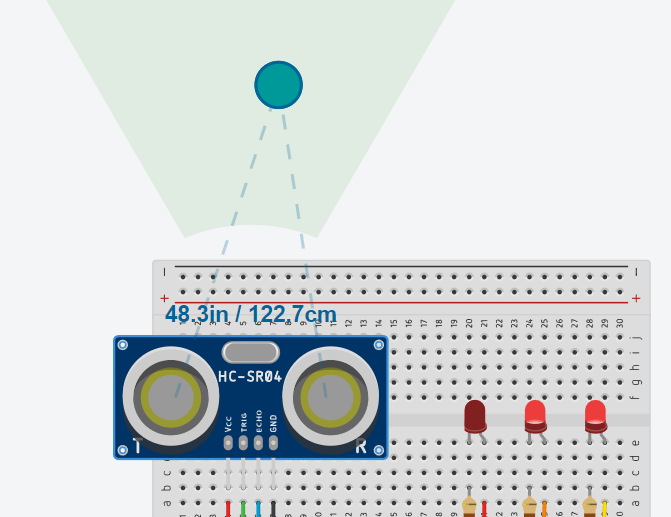
THEORY:

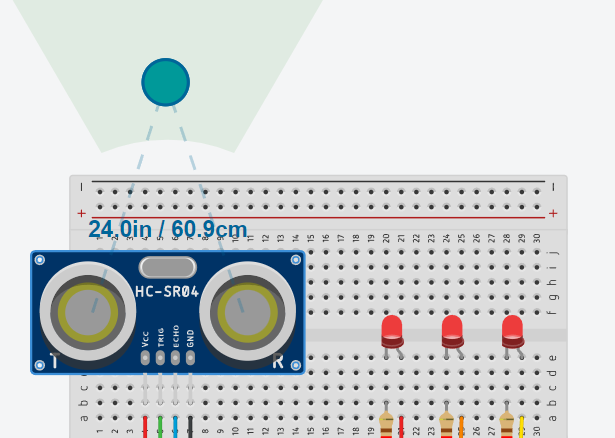
The ultrasonic sensor emits ultrasonic waves and measures the time it takes for the waves to bounce back after hitting an object.by using this principle when an object reaches a certain level LED should starts glowing.the circuit diagram is hown in figure 7.1



OBSERVATION OF SIMULATION:

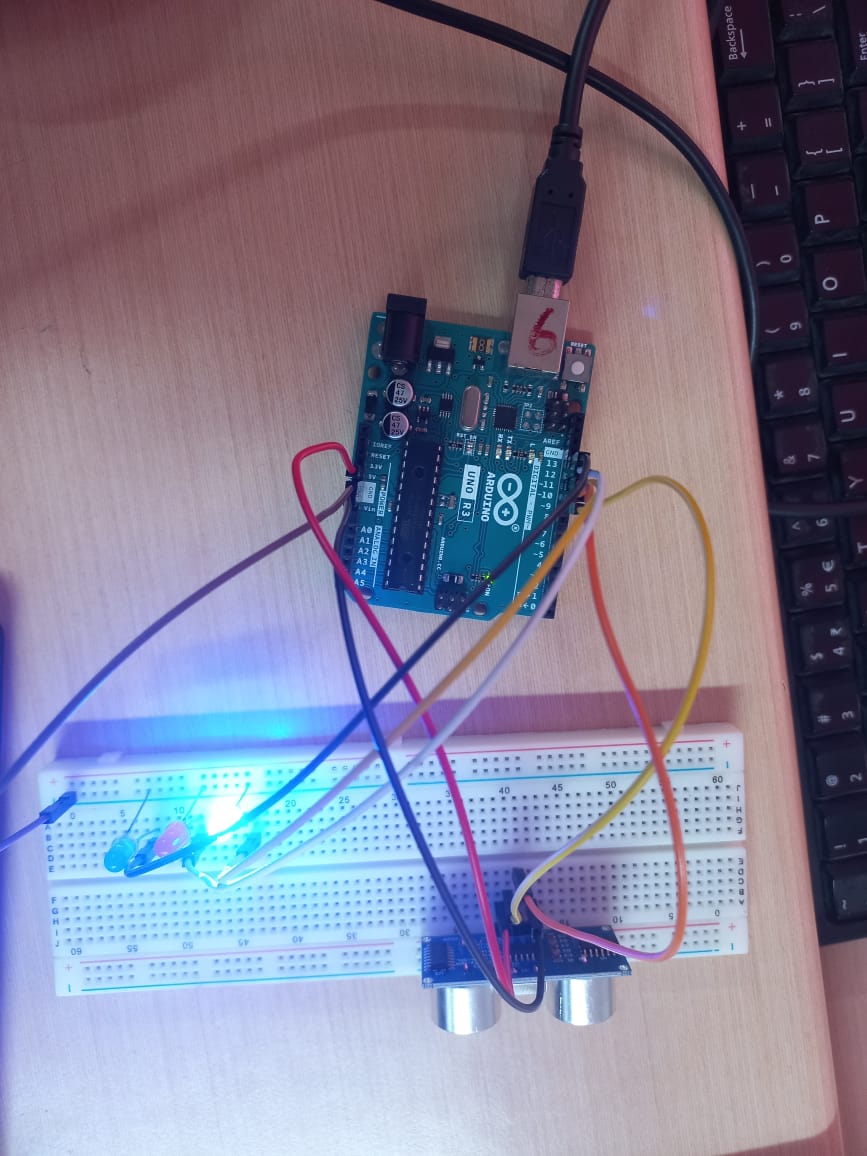
* All three LED start glowing when the distance is less than 100
* The two LED start glowing when the value of ultrasonic sensor is between 100-250
* Then ONE LED start glowing when the value of ultrasonic sensor is grater than 250
* It shown in below diagram.

****



OBSERVATION OF HARDWARE:

* All three LED start glowing when the distance is less than 100
* The two LED start glowing when the value of ultrasonic sensor is between 100-250
* Then ONE LED start glowing when the value of ultrasonic sensor is grater than 250
* It shown in below diagram.



**Code:**

// C++ code

//

int distanceThreshold = 0;

int cm = 0;

int inches = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT); // Clear the trigger

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds

return pulseIn(echoPin, HIGH);

}

void setup()

{

Serial.begin(9600);

pinMode(2, OUTPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

}

void loop()

{

// set threshold distance to activate LEDs

distanceThreshold = 350;

// measure the ping time in cm

cm = 0.01723 \* readUltrasonicDistance(7, 6);

// convert to inches by dividing by 2.54

inches = (cm / 2.54);

Serial.print(cm);

Serial.print("cm, ");

Serial.print(inches);

Serial.println("in");

if (cm > distanceThreshold) {

digitalWrite(2, LOW);

digitalWrite(3, LOW);

digitalWrite(4, LOW);

}

if (cm <= distanceThreshold && cm > distanceThreshold - 100) {

digitalWrite(2, HIGH);

digitalWrite(3, LOW);

digitalWrite(4, LOW);

}

if (cm <= distanceThreshold - 100 && cm > distanceThreshold - 250) {

digitalWrite(2, HIGH);

digitalWrite(3, HIGH);

digitalWrite(4, LOW);

}

if (cm <= distanceThreshold - 250 && cm > distanceThreshold - 350) {

digitalWrite(2, HIGH);

digitalWrite(3, HIGH);

digitalWrite(4, HIGH);

}

if (cm <= distanceThreshold - 350) {

digitalWrite(2, HIGH);

digitalWrite(3, HIGH);

digitalWrite(4, HIGH);

}

delay(100); // Wait for 100 millisecond(s)

}

**INFERENCE:**

I learnt how to assemble and simulate a ultrasonic sensor with Arduino using thinkercad and Arduino ide.

Simulation of Bluetooth module HC05 controlled device.

EXPERIMENT NO (HW/SW): 10

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

To simulate Bluetooth module HC05 controlled device

**SOFTWARE USED:**

Tinkercad

Arduino ide

**COMPONENTS USED:**

* Bluetooth HCO5
* Breadboard
* Arduino
* Mobile device

**Design and Theory:**

**Design Steps:**

1. **Connect the HC-05 Bluetooth Module:**
   * Connect the VCC and GND pins of the HC-05 to 5V and GND on the Arduino.
   * Connect the TX pin of the HC-05 to the RX pin (pin 0) on the Arduino.
   * Connect the RX pin of the HC-05 to the TX pin (pin 1) on the Arduino.
   * Connect the EN pin of the HC-05 to 5V (this sets the module to a data mode).
2. **Connect the LED:**
   * Connect the positive leg of the LED to a digital pin (e.g., D2) on the Arduino.
   * Connect the negative leg of the LED to a current-limiting resistor (220Ω) and then to GND on the Arduino.
3. **Write Arduino Code:**
   * Use the Arduino IDE or Tinkercad Code Editor to write the code.
   * Include the "SoftwareSerial.h" library for handling communication with the Bluetooth module.

4. **Test the System:**

* + Upload the code to the Arduino in Tinkercad.
  + Use the Tinkercad simulation to send commands (1 or 0) through the virtual serial monitor to control the LED.

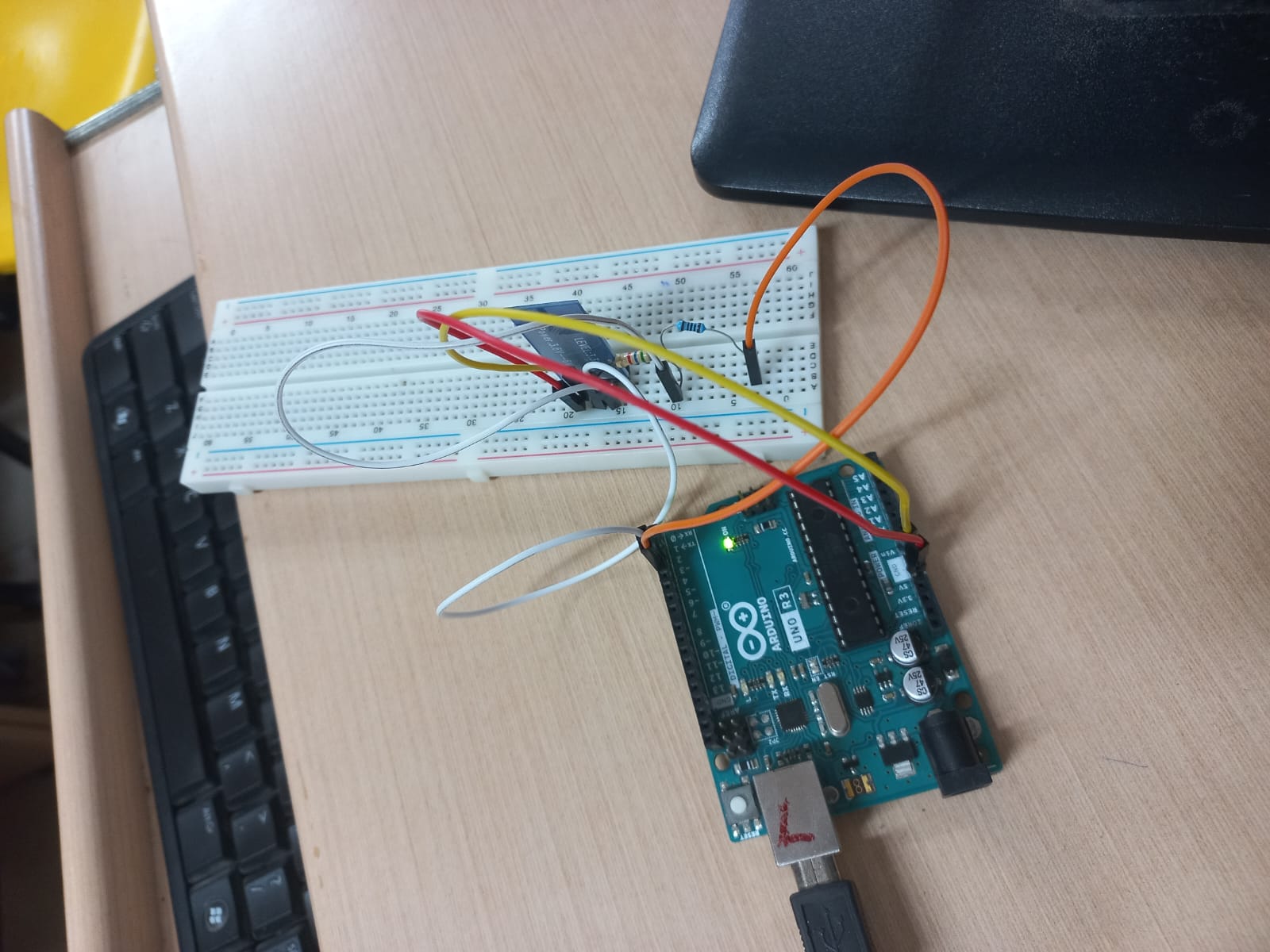
5. **Bluetooth Control in Tinkercad:**

* + In Tinkercad, open the Serial Monitor and send '1' to turn on the LED and '0' to turn it off.

**Theory:**

The HC-05 Bluetooth module enables wireless communication between devices.It uses the Serial Communication Protocol (UART) to transmit and receive data.The SoftwareSerial library allows the Arduino to create additional virtual serial ports.In this project, it's used to create a software-based serial port for communication with the HC-05.

**Experiment/Observation:**



**RINFERNCE:**

Simulation and hardware of Bluetooth module HC05 controlled device has been done successfully.

**Hardware design and simulation of servomotor based speed control device.**

EXPERIMENT NO (HW/SW): 9

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

Aim of this experiment to Simulate, assemble and code a servo motor.

**SOFTWARE USED:**

Tinkercad

Arduino ide

**COMPONENTS USED:**

* Servo motor
* Breadboard
* Arduino

**Design and Theory:**

**Design Steps:**

1. **Connect the Servo Motor:**
   * Connect the signal (control) wire of the servo motor to a PWM-enabled pin on the Arduino (e.g., D9).
   * Connect the power wire (usually red) of the servo motor to 5V on the Arduino.
   * Connect the ground wire (usually brown) of the servo motor to GND on the Arduino.
2. **Write Arduino Code:**
   * Use the Arduino IDE or Tinkercad Code Editor to write the code.

3. **Test the System:**

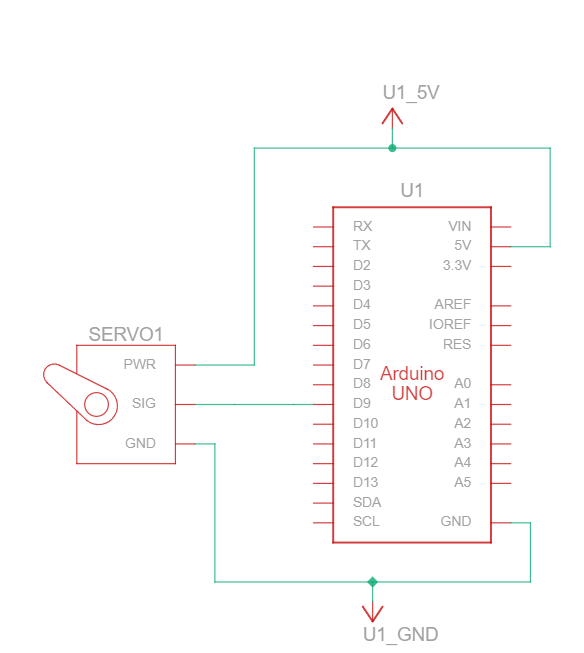
* + Upload the code to the Arduino in Tinkercad.
  + Use the simulation to observe how the servo motor responds to changes in the potentiometer position.

4. **Adjust and Expand:**

* + Fine-tune the code and connections based on your specific components and requirements.
  + Expand the system by incorporating additional sensors or modifying the code for more complex behavior.

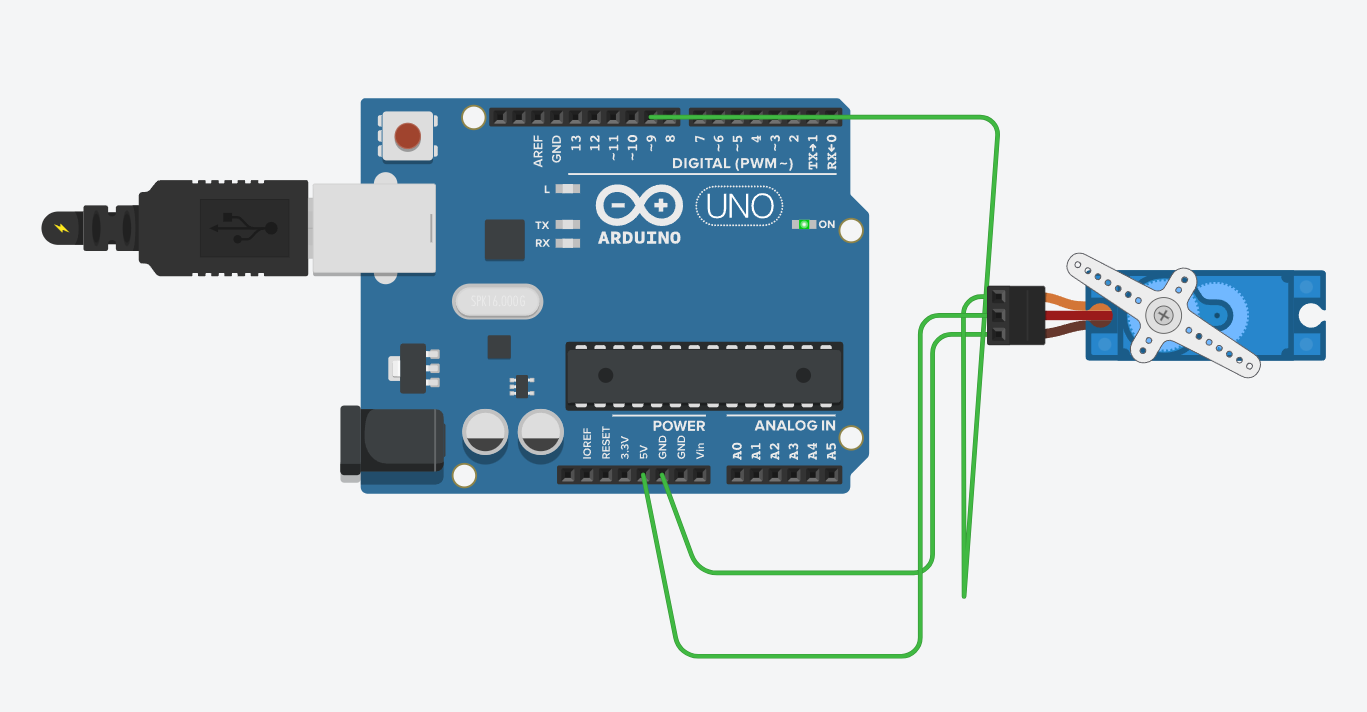
**Theory:**

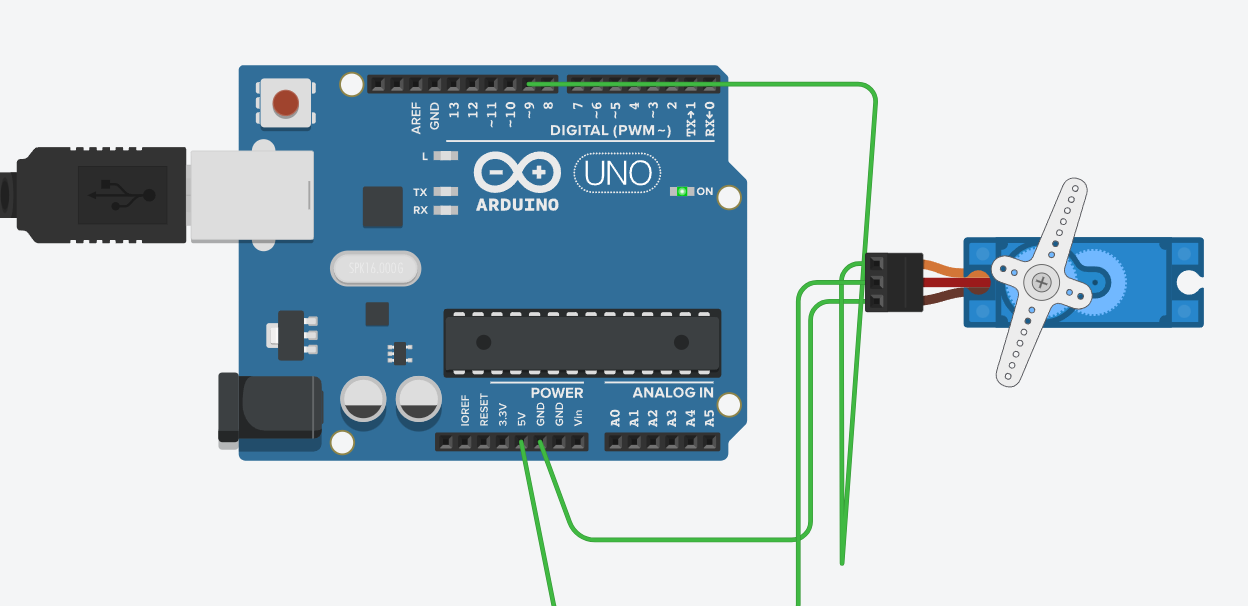
A servo motor is a rotary actuator that allows for precise control of angular position. We can determine the angle of rotation and speed of the rotation using the Arduino. The circuit diagram shown below.



OBSERVATION OF SIMULATION:

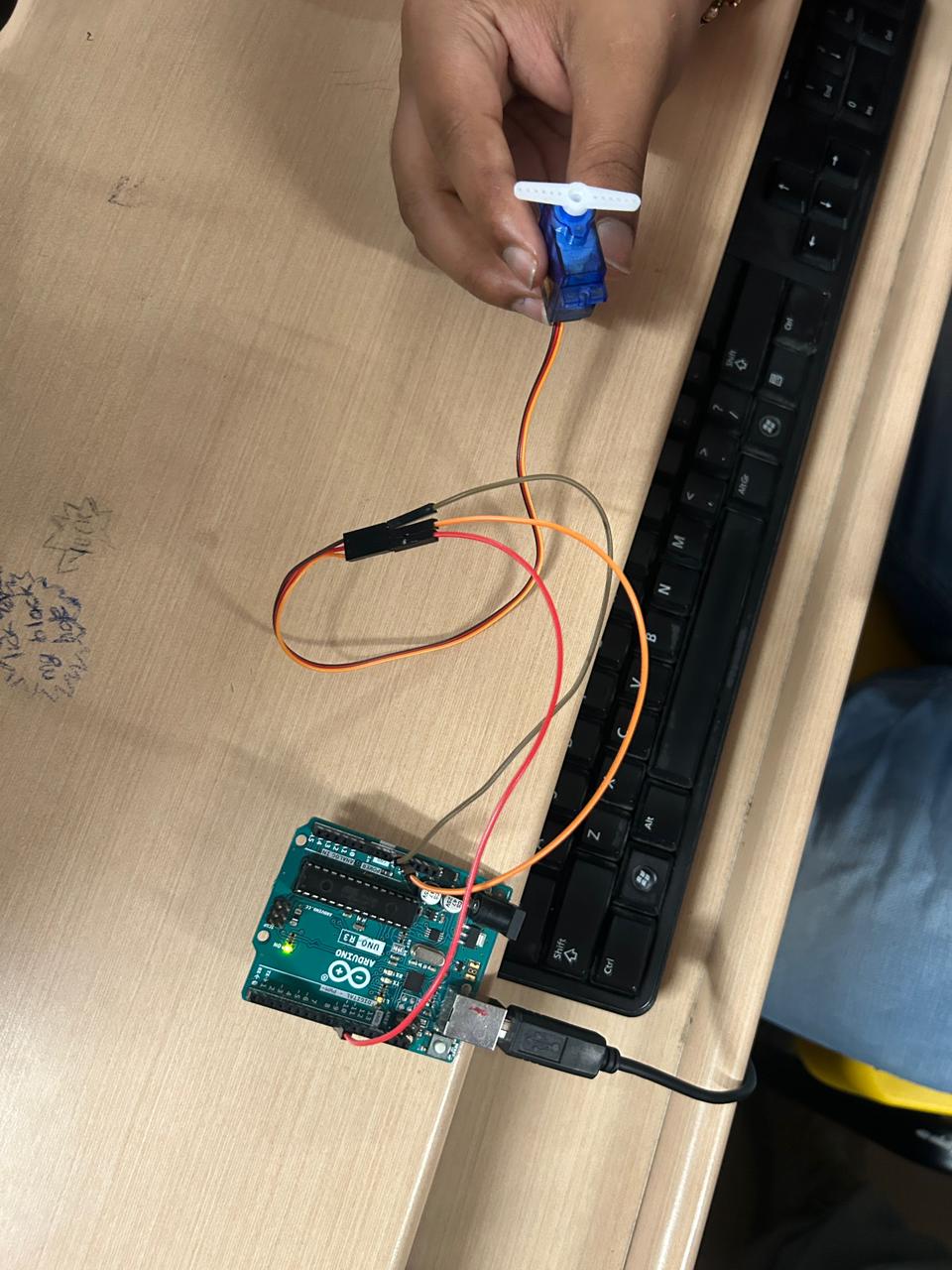
* Servo motor starts rotating
* Code ran successfully





OBSERVATION OF HARDWARE:

* Servo motor starts rotating
* Code ran successfully
* The output is shown in the diagram.



Code:

#include <Servo.h>

Servo myservo; // create servo object to control a servo

// twelve servo objects can be created on most boards

int pos = 0; // variable to store the servo position

void setup() {

myservo.attach(9); // attaches the servo on pin 9 to the servo object

}

void loop() {

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

**INFERENCE:**

I learnt how to assemble and simulate a servo motor using Arduino board in tinkercad and Arduino ide.

**Simulation of Bluetooth controlled door locking system.**

EXPERIMENT NO (HW/SW): 10

REGISTER NO: 22BEC1508

NAME: K.V.SREEJJESH

DATE:

**AIM:**

Aim of this experiment to Simulate a Bluetooth controlled door locking system.

**SOFTWARE USED:**

Tinkercad

**Design and Theory:**

**Design Steps:**

1. **Connect the HC-05 Bluetooth Module:**
   * Connect the VCC and GND pins of the HC-05 to 5V and GND on the Arduino.
   * Connect the TX pin of the HC-05 to the RX pin (pin 0) on the Arduino.
   * Connect the RX pin of the HC-05 to the TX pin (pin 1) on the Arduino.
   * Connect the EN pin of the HC-05 to 5V (this sets the module to data mode).
2. **Connect the Servo Motor:**
   * Connect the signal (control) wire of the servo motor to a PWM-enabled pin on the Arduino (e.g., D9).
   * Connect the power wire (usually red) of the servo motor to 5V on the Arduino.
   * Connect the ground wire (usually brown) of the servo motor to GND on the Arduino.
3. **Write Arduino Code:**
   * Use the Arduino IDE or Tinkercad Code Editor to write the code.
   * Include the "SoftwareSerial.h" library for handling communication with the Bluetooth module.

4. **Test the System:**

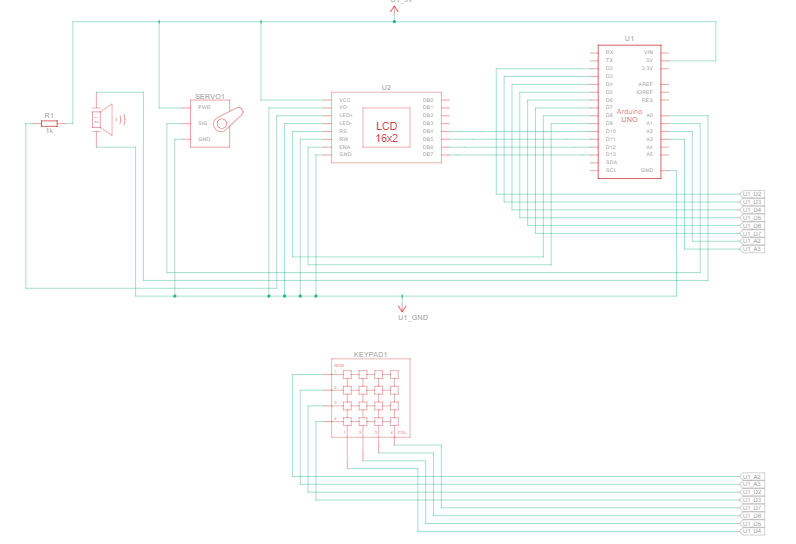
* + Upload the code to the Arduino in Tinkercad.
  + Use the Tinkercad simulation to send commands ('L' to lock, 'U' to unlock) through the virtual serial monitor to control the servo motor.

5. **Bluetooth Control in Tinkercad:**

* + In Tinkercad, open the Serial Monitor and send 'L' to simulate locking the door and 'U' to simulate unlocking the door.

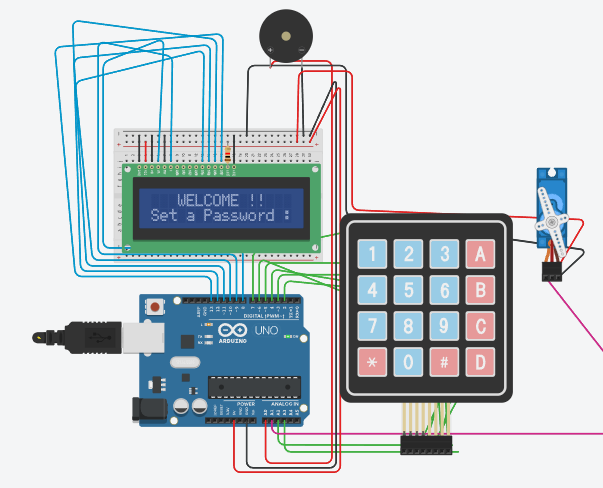
**Theory:**

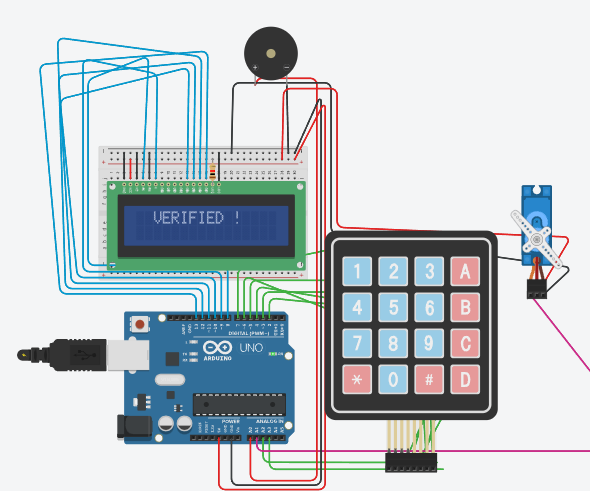
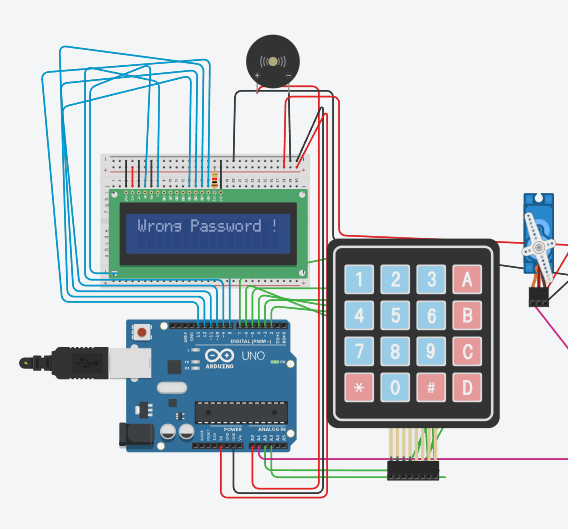
The servo motor is used to simulate the door locking mechanism. It moves to specific angles to represent the locked and unlocked states. Separate functions (**lockDoor** and **unlockDoor**) are created to control the servo motor's movement, simulating the door locking and unlocking processes. The circuit diagram is given below.



OBSERVATION OF SIMULATION:

* We can set the password
* Servo motor rotate when we entered the password correctly.
* Servo motor does not rotate when we entered the password wrongly and buzzer is alarming.
* The simulation outputs are given below.

****

**** 

**Code:**

#include <LiquidCrystal.h>

#include <Keypad.h>

#include <Servo.h>

LiquidCrystal LCD( 8, 9, 10, 11, 12, 13 ) ;

Servo servo ;

const int buzzerPin = 14 ;

const int servoPin = 15 ;

const byte ROWS = 4, COLS = 4 ;

char keys[ROWS][COLS] =

{

{'1','2','3','A'},

{'4','5','6','B'},

{'7','8','9','C'},

{'\*','0','#','D'}

} ;

byte rowPins[ROWS] = { 16, 17, 2, 3 } ;

byte colPins[COLS] = { 4, 5, 6, 7 } ;

Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ) ;

char password[16], string[16] ;

int flag\_h\_setpassword = 1, flag\_inputpassword = 0, flag\_inputstring = 0, flag\_opendoor = 1, flag\_state = 0, flag\_remoteopen = 0, flag\_lockdown = 0 ;

int count = 0, trial\_count=0, pos = 0, state = 0 ;

// flag\_setpassword = flag for setting the password,

// flag\_stringinput = flag for taking input the string,

void setup() {

for(int k=8 ; k<14 ; k++) {

pinMode(k,OUTPUT) ;

}

LCD.begin(16, 2) ;

pinMode(buzzerPin, OUTPUT) ;

pinMode(servoPin, OUTPUT) ;

servo.attach(servoPin) ;

// for Bluetooth-Module

Serial.begin(9600) ;

LCD.setCursor(0,0) ;

LCD.print(" WELCOME !!") ;

LCD.setCursor(0,1) ;

LCD.print("Set a Password :") ;

InitializePassword(), InitializeString() ;

CloseDoor() ;

}

void loop() {

// for Bluetooth-module

if( Serial.available() > 0 ) {

state = Serial.read() ;

flag\_state = 0 ;

}

// Bluetooth-module is used to remotely open the door lock when system is locked

if( state == '0' ) {

trial\_count = 4, CloseDoor() ;

if( flag\_state == 0 ) {

Serial.println("SYSTEM LOCKED") ;

flag\_state = 1 ;

}

flag\_lockdown = 1 ;

}

else if( state == '1' && flag\_lockdown == 1 ) {

trial\_count = 0 ;

LCD.clear() ;

LCD.setCursor(0,0) ;

LCD.print("LOCKDOWN LIFTED!") ;

LCD.setCursor(0,1) ;

LCD.print("Press \* ...") ;

if( flag\_state == 0 ) {

Serial.println("LOCKDOWN LIFTED") ;

flag\_state = 1 ;

}

flag\_lockdown = 0, state = 0 ;

}

else if( state == '2' && flag\_remoteopen == 0 && flag\_lockdown == 0 ) {

LCD.clear() ;

LCD.print("RemotelyVERIFIED") ;

Serial.println("UNLOCKED !!") ;

trial\_count = 0 ;

for( int i=0 ; i<3 ; ++i ) {

tone(buzzerPin, 500, 200) ;

delay(230) ;

tone(buzzerPin, 100, 200) ;

delay(300) ;

OpenDoor() ;

}

if( flag\_state == 0 ) {

Serial.println("Remotely Unlocked !!") ;

flag\_state = 1 ;

}

flag\_remoteopen = 1 ;

}

else if( state == '3' && flag\_lockdown == 0 ) {

trial\_count = 0, InitializeString(), H\_EnterPassword() ;

if( flag\_state == 0 ) {

Serial.println("Remotely Locked !!") ;

flag\_state = 1 ;

}

flag\_remoteopen = 0, state = 0 ;

}

//Keypad

if( trial\_count < 3 ) {

char key = kpd.getKey() ; //storing pressed key value in a char

if( key != NO\_KEY ) {

if( flag\_h\_setpassword == 1 ) {

H\_SetPassword() ;

}

if( key == '\*' ) {

if( flag\_inputpassword == 1 ) {

InitializePassword(), H\_SetPassword() ;

}

else if( flag\_inputstring = 1 ) {

InitializeString(), H\_EnterPassword() ;

}

}

else if( key == '#' ) {

if( flag\_inputpassword == 1 && count > 0 ) {

flag\_inputpassword = 0 ;

password[count] = '\0' ;

H\_EnterPassword() ;

}

else if( flag\_inputstring == 1 && count > 0 ) {

flag\_inputstring = 0 ;

string[count] = '\0' ;

if( Compare\_Password\_and\_String() == 1 ) {

LCD.clear() ;

LCD.print(" VERIFIED ! ") ;

Serial.println("UNLOCKED") ;

trial\_count = 0 ;

for( int i=0 ; i<3 ; ++i ) {

tone(buzzerPin, 500, 100) ;

delay(230) ;

tone(buzzerPin, 100, 100) ;

delay(230) ;

OpenDoor() ;

}

}

else {

LCD.clear() ;

LCD.print("Wrong Password !") ;

delay(1000) ;

Serial.println("Someone unsuccessfully attempted to open the lock !") ;

++trial\_count ;

tone(buzzerPin, 100, 1000) ;

delay(1000) ;

H\_EnterPassword() ;

}

}

}

else if( flag\_inputpassword == 1 || flag\_inputstring == 1 ) {

LCD.print(key) ;

delay(100) ;

LCD.setCursor(count,1) ;

LCD.print('\*') ;

if( flag\_inputpassword == 1 ) password[count] = key ;

else if( flag\_inputstring == 1 ) string[count] = key ;

++count ;

}

}

}

else {

LCD.clear() ;

LCD.setCursor(0,0) ;

LCD.print("SYSTEM LOCKDOWN!") ;

tone(buzzerPin, 1000, 1000) ;

delay(1500) ;

flag\_lockdown = 1 ;

}

}

void InitializePassword() {

for( int i=0 ; i<16 ; ++i )

password[i] = 0 ;

}

void InitializeString() {

for( int i=0 ; i<16 ; ++i )

string[i] = 1 ;

}

void H\_SetPassword() {

LCD.clear() ;

LCD.setCursor(0,0) ;

LCD.print("Set a Password :") ;

LCD.setCursor(0,1) ;

flag\_h\_setpassword = 0 ;

flag\_inputpassword = 1, count = 0 ;

}

void H\_EnterPassword() {

CloseDoor() ;

LCD.clear() ;

LCD.setCursor(0,0) ;

LCD.print("Enter Password :") ;

LCD.setCursor(0,1) ;

flag\_inputstring = 1, count = 0 ;

}

int Compare\_Password\_and\_String() {

int i ;

for( i=0 ; password[i]!='\0' && string[i]!='\0' ; ++i ) {

if( password[i] != string[i] )

return 0 ;}

if( password[i] == '\0' && string[i] == '\0' )

return 1 ;

else return 0 ;}

void OpenDoor() {

if( flag\_opendoor == 1 )

return;

for( pos=15 ; pos<=100 ; ++pos ) {

servo.write(pos) ;

delay(15) ;

}

flag\_opendoor = 1 ;

}

void CloseDoor() {

if( flag\_opendoor == 0 )

return;

for( pos=100 ; pos>=15; --pos ) {

servo.write(pos) ;

delay(15) ;

}

flag\_opendoor = 0 ;

}

**INFERENCE:**

I learnt how to simulate aBluetooth controlled door locking system using tinkercad.

**Automatic Plant Watering System using Arduino Tinkercad simulation.**

**Experiment No: 11**

**Register No: 22BEC1508**

**Name: k v sreejjesh**

**Date: 25-09-2023**

**Aim:**

To simulate an Automatic Plant Watering System using Tinkercad

**SOFTWARE USSED:**

Tinkercad

**Design and Theory:**

**Design Steps:**

1. **Connect the Soil Moisture Sensor:**
   * Connect the VCC and GND pins of the soil moisture sensor to 5V and GND on the Arduino.
   * Connect the analog output pin of the soil moisture sensor to an analog pin on the Arduino (e.g., A0).
2. **Connect the Water Pump:**
   * Connect the positive wire of the water pump to one of the relay module outputs (or the collector of the transistor).
   * Connect the negative wire of the water pump to the diode's cathode.
   * Connect the anode of the diode to GND on the Arduino.
   * Connect the emitter of the transistor (if using a transistor-based circuit) to GND on the Arduino.
3. **Connect the Relay Module (or Transistor-Based Circuit):**
   * If using a relay module, connect its control pin to a digital pin on the Arduino (e.g., D2).
   * If using a transistor-based circuit, connect the base of the transistor to a digital pin on the Arduino (e.g., D2).
4. **Connect Tubing and Water Reservoir:**
   * Connect tubing from the water pump outlet to the soil around the plant.
   * Place the water reservoir at an elevated position to create a gravity-fed system, ensuring water flows to the pump easily.
5. **Write Arduino Code:**
   * Use the Arduino IDE or Tinkercad Code Editor to write the code.

6. **Test the System:**

* + Upload the code to the Arduino in Tinkercad.
  + Use the simulation to observe how the water pump turns on when the soil moisture is below the threshold.

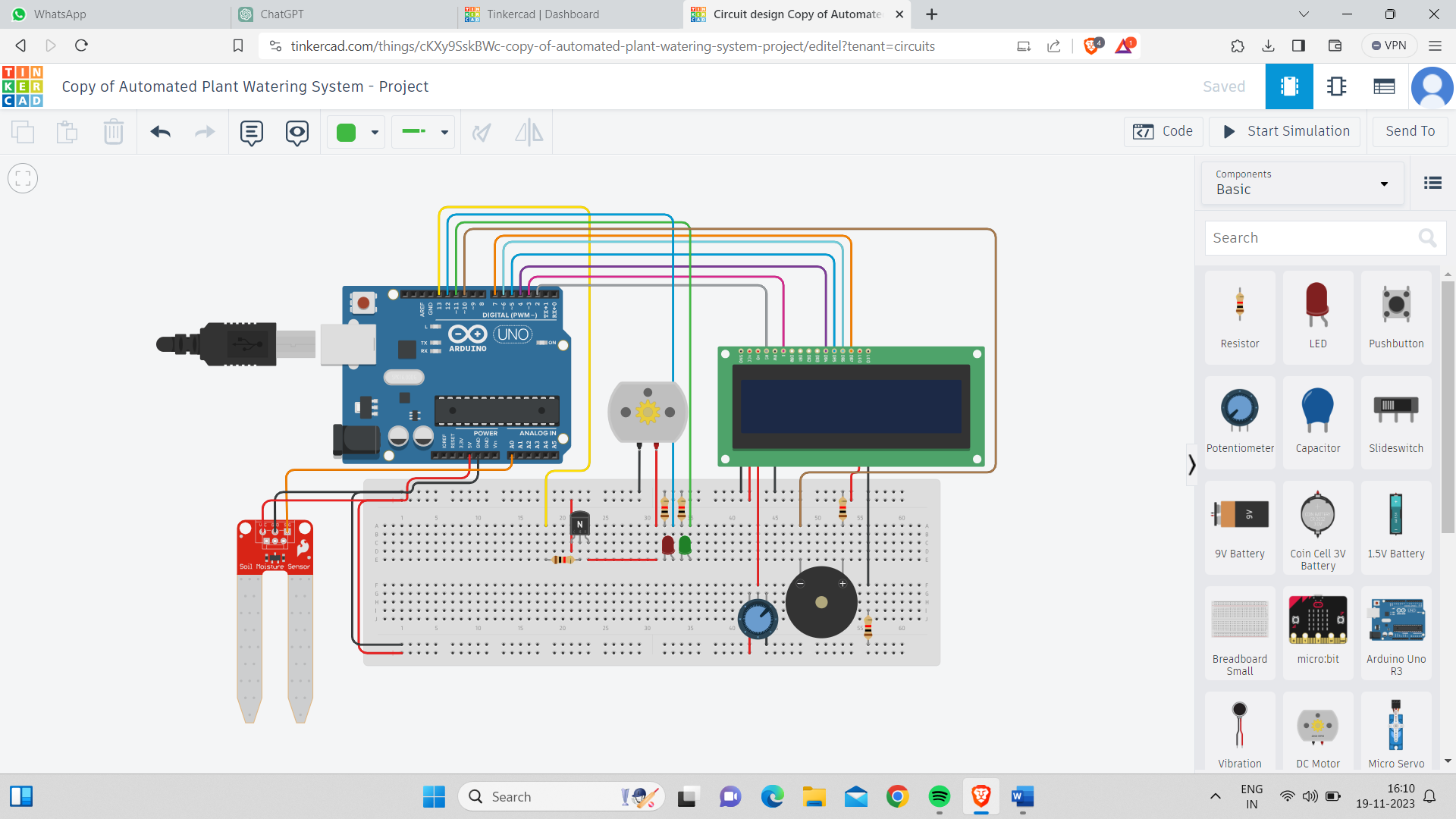
7. **Adjust and Expand:**

* + Fine-tune the code and connections based on your specific components and requirements.
  + Expand the system by adding more sensors, pumps, or incorporating additional features.

**THEORY:**

The soil moisture sensor measures the moisture content in the soil.In this project, a lower analog value indicates drier soil.The water pump is used to deliver water to the plant when the soil moisture falls below a certain threshold.The diode protects the circuit from voltage spikes generated by the water pump when it is turned off.Elevating the water reservoir ensures a gravity-fed system, allowing water to flow easily to the pump.

**Experiment/Observation:**



**CODE:**

#include <LiquidCrystal.h>

// Include the LiquidCrystal library, which provides functions to control the LCD display.

// Declaring variables

const int BuzzerPin = 10; // Pin connected to the piezo buzzer

const int LedRed = 12; // Pin connected to the red LED

const int LedGreen = 11; // Pin connected to the green LED

const int SoilMoistureSensor = A0; // Pin connected to the soil moisture sensor

const int WaterPump = 13; // Pin connected to the water pump relay

LiquidCrystal lcd(2, 3, 4, 5, 6, 7); // Create an instance of the LiquidCrystal class to control the LCD

void setup()

{

pinMode(WaterPump, OUTPUT); // Set the water pump pin as output

pinMode(LedRed, OUTPUT); // Set the red LED pin as output

pinMode(LedGreen, OUTPUT); // Set the green LED pin as output

pinMode(BuzzerPin, OUTPUT); // Set the piezo buzzer pin as output

Serial.begin(9600); // Initialize serial communication at 9600 bps

lcd.begin(16, 2); // Initialize the LCD with 16 columns and 2 rows

pinMode(BuzzerPin, OUTPUT); // Set the piezo pin as output

lcd.clear(); // Clear the LCD display

lcd.setCursor(0, 0); // Set the cursor to the first column and first row

String message1 = "Automated Plant";

String message2 = "Watering System";

// Display "Automated Plant" on the first row of the LCD with a delay of 100ms between characters

for (int i = 0; i < message1.length(); i++) {

lcd.print(message1.charAt(i));

delay(100);

}

lcd.setCursor(0, 1); // Set the cursor to the first column and second row

// Display "Watering System" on the second row of the LCD with a delay of 100ms between characters

for (int i = 0; i < message2.length(); i++) {

lcd.print(message2.charAt(i));

delay(100);

}

delay(2500); // Delay for 2.5 seconds to display the messages

lcd.clear(); // Clear the LCD display again

lcd.setCursor(0, 0); // Set the cursor to the first column and first row

lcd.print("Moisture ="); // Display "Moisture =" on the first row

lcd.setCursor(0, 1); // Set the cursor to the first column and second row

lcd.print("Water Pump ="); // Display "Water Pump =" on the second row

}

void loop()

{

int Sensor = analogRead(SoilMoistureSensor); // Read the value from the soil moisture sensor

int mappedValue = map(Sensor, 0, 876, 0, 99); // Map the sensor value to a range from 0 to 99

lcd.setCursor(11, 0); // Set the cursor to the 12th column and first row

lcd.print(mappedValue); // Display the mapped value (moisture percentage) on the first row

lcd.setCursor(14, 0); // Set the cursor to the 15th column and first row

lcd.print("%"); // Display the percentage symbol on the first row

lcd.setCursor(13, 1); // Set the cursor to the 14th column and second row

// Control the water pump and LEDs based on the moisture percentage

if (mappedValue < 50) {

digitalWrite(WaterPump, HIGH); // Turn on the water pump

digitalWrite(LedGreen, HIGH); // Turn on the green LED

digitalWrite(LedRed, LOW); // Turn off the red LED

lcd.print("ON "); // Display "ON" on the second row

playSound(); // Play a sound using the piezo buzzer

}

else {

digitalWrite(WaterPump, LOW); // Turn off the water pump

digitalWrite(LedGreen, LOW); // Turn off the green LED

digitalWrite(LedRed, HIGH); // Turn on the red LED

lcd.print("OFF"); // Display "OFF" on the second row

}

}

void playSound() {

tone(BuzzerPin, 87, 100); // Play a tone of 87 Hz for 100 milliseconds on the piezo buzzer

delay(1000); // Delay for 1 second

}

**INFERNCE:**

I learnt the Simulation of Automatic Plant Watering System using tinkercad.