**Experiment no : 01**

**Date : 22.07.2024**

**BLINKING OF RGB LED USING ARDUINOBOARD**

**AIM**

To design and implement a circuit to blink a single LED and an RGB LED using an Arduino

**PROCEDURE**

**Step-1:** Place the RGB on the breadboard. Connect each of its red, green, and blue pins to separate resistors and connect the common cathode (or anode) pin to GND (or 5V) based on LED type.

**Step-2**: Connect each color pin (red, green, blue) to different digital pins on the Arduino, such as pins 9, 10, and 11.

**Step-3**: In the Arduino code, set the pins connected to the red, green, and blue terminals as outputs.

**Step-4:** Use a loop in the code to turn on one color at a time with a delay of 1000milliseconds, then turn it off before moving to the next color.

**Step-5:** Upload the code to the Arduino and observe each color (red, green, blue) blinking sequentially with a 1-second delay

**PROGRAM**

int led\_red = 11;

int led\_yellow = 12;

int led\_green =13;

void setup() {

pinMode(led\_red, OUTPUT);

pinMode(led\_yellow, OUTPUT);

pinMode(led\_green, OUTPUT);

}

void loop() {

digitalWrite(led\_red, LOW);

digitalWrite(led\_yellow, LOW);

digitalWrite(led\_green, HIGH);

delay(2000);

digitalWrite(led\_red, LOW);

digitalWrite(led\_yellow, HIGH);

digitalWrite(led\_green, LOW);

delay(1000);

digitalWrite(led\_red, HIGH);

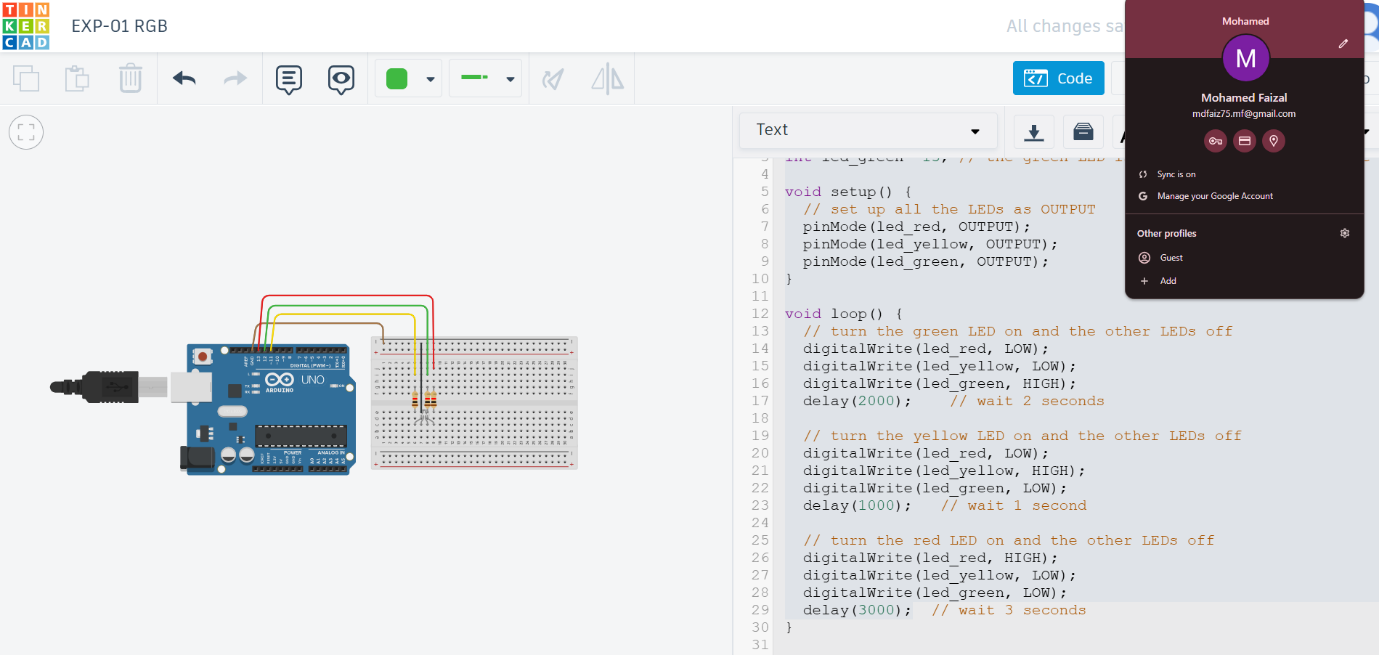
digitalWrite(led\_yellow, LOW);

digitalWrite(led\_green, LOW);

delay(3000);

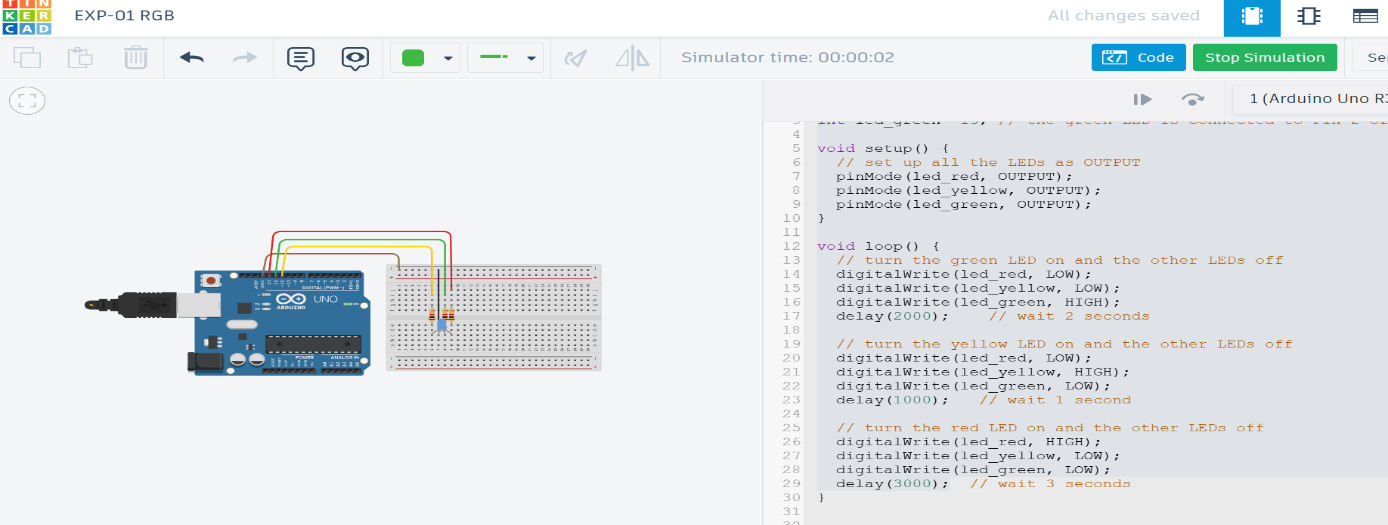
**OUTPUT**

**BEFORE SIMULATION**

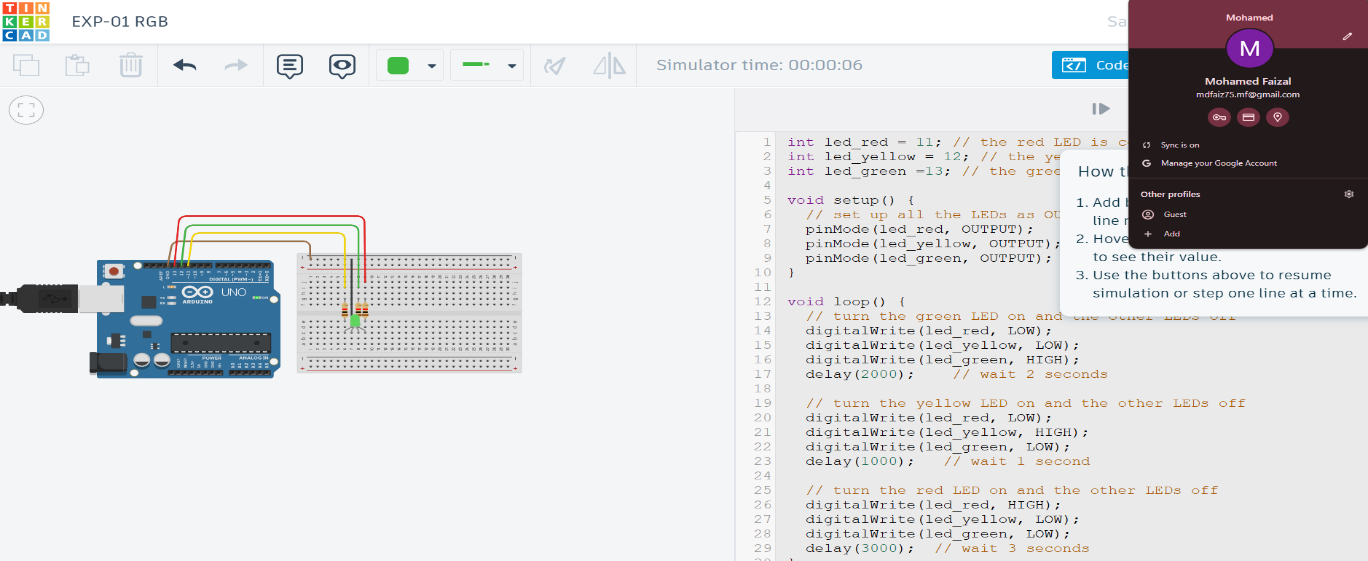
****

**AFTER SIMULATION**

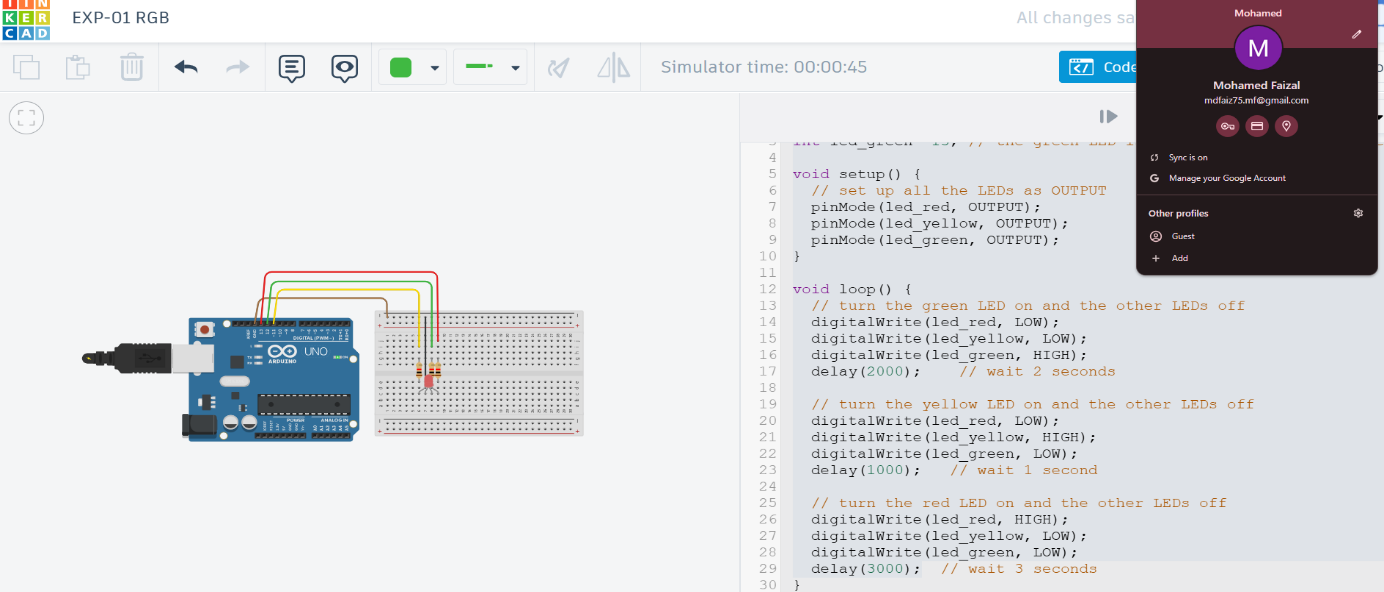
**Blue**

****

**Green**

****

**Red**

****

**RESULT**

To design and implement a circuit to blink a RGB LED using an Arduino was executed successfully

**Experiment no : 02**

**Dtae : 29.07.2024**

**TRAFFIC LIGHT CONTROL USING ARDUINO**

**AIM**

To create a traffic light control system using an Arduino Uno and LEDs, simulating traffic signals for multiple directions at an intersection.

**PROCEDURE**

**Step-1:** Connect LEDs to the Arduino Uno through a breadboard. Use resistors to protect the LEDs from excess current. Each direction has three LEDs(red, yellow,green) representing stop, wait, and go signals.

**Step-2:** Assign digital pins on the Arduino for each LED, mapping them to variables in the code (e.g., `int red\_1 = 13;`). Ensure each LED has a unique pin number and corresponds to a specific direction.

**Step-3:** Program functions for each traffic light sequence (e.g., `direction\_1\_green()`, ` direction\_2\_yellow()`). Each function turns specific LEDs on or off to simulate theflow of traffic in one direction at a time.

**Step-4:** In `setup()`, set each pin as an output. In `loop()`, call each drectional function with delays to mimic real traffic light timing. Alternate between green, yellow, and red states with appropriate delays for each signal phase.

**Step-5:** Simulate the program in Tinker cad. Observe the LEDs to ensure they mimica real traffic control system, with each direction allowing traffic to proceed while others are stopped. Adjust timing and connections as needed for smooth operation.

**PROGRAM**

int \_red=2;

int \_yellow=3;

int \_green=4;

int led\_red=5;

int led\_yellow=6;

int led\_green=7;

void setup()

{

pinMode(\_red,OUTPUT);

pinMode(\_yellow,OUTPUT);

pinMode(\_green,OUTPUT);

pinMode(led\_red,OUTPUT);

pinMode(led\_yellow,OUTPUT);

pinMode(led\_green,OUTPUT);

}

void loop()

{

digitalWrite(\_red,HIGH);

digitalWrite(led\_green,HIGH);

digitalWrite(\_green,LOW);

digitalWrite(\_yellow,LOW);

digitalWrite(led\_yellow,LOW);

digitalWrite(led\_red,LOW);

delay(5000);

digitalWrite(\_yellow,HIGH);

digitalWrite(\_red,LOW);

digitalWrite(\_green,LOW);

digitalWrite(led\_yellow,HIGH);

digitalWrite(led\_green,LOW);

digitalWrite(led\_red,LOW);

delay(5000);

digitalWrite(\_green,HIGH);

digitalWrite(led\_red,HIGH);

digitalWrite(\_red,LOW);

digitalWrite(\_yellow,LOW);

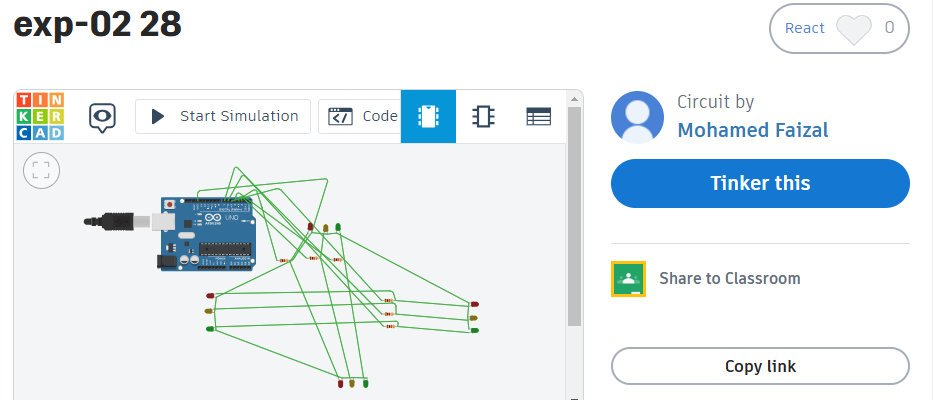
digitalWrite(led\_yellow,LOW);

digitalWrite(led\_green,LOW);

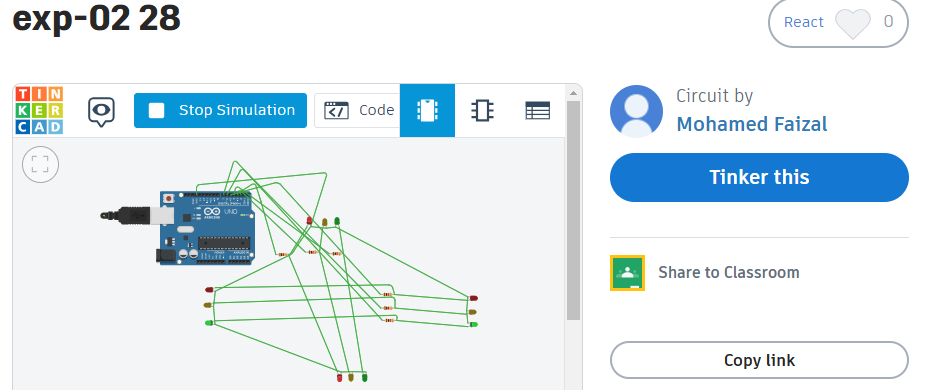
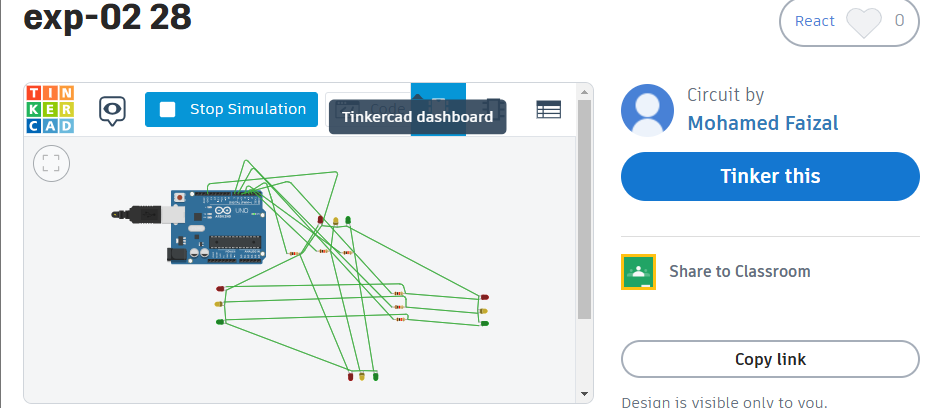
delay(5000);}

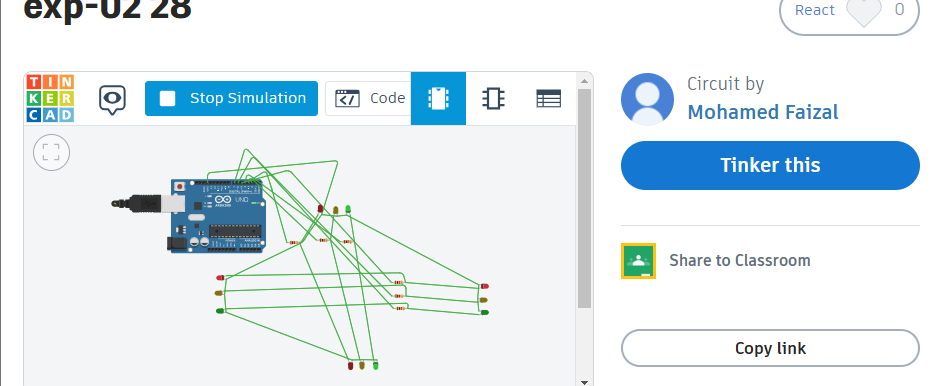
**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

****

**RESULT:**

To create a traffic light control system using an Arduino Uno and LEDs, simulating traffic signals for multiple directions at an intersection was executed successfully

**Experiment no : 03 (a)**

**Date: 05.08.2024**

**DISTANCE MEASUREMENTS USING UV SENSORWITHARDUINO**

**AIM**

To create a simple Arduino circuit that uses an ultrasonic sensor to measuredistance

**PROCEDURE**

**Step-1**: Attach the VCC and GND pins of the HC-SR04 sensor to the 5VandGNDpins of the Arduino. Connect the Trig pin to digital pin 9 and the Echo pin to digital pin 10.

**Step-2**: In the Arduino code, set the Trig pin as an output and the Echo pin as an input.

**Step-3:** Send a 10-microsecond pulse to the Trig pin to initiate distance measurement.

**Step-4:** Use `pulse In()` on the Echo pin to capture the duration of the echo pulse.

**Step-5:** Convert the echo time to distance and print the result on the Serial Monitor to observe the measured values.

**PROGRAM**

int inches = 0;

int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT);

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

return pulseIn(echoPin, HIGH);

}

void setup()

{

Serial.begin(9600);

}

void loop()

{

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

inches = (cm / 2.54);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

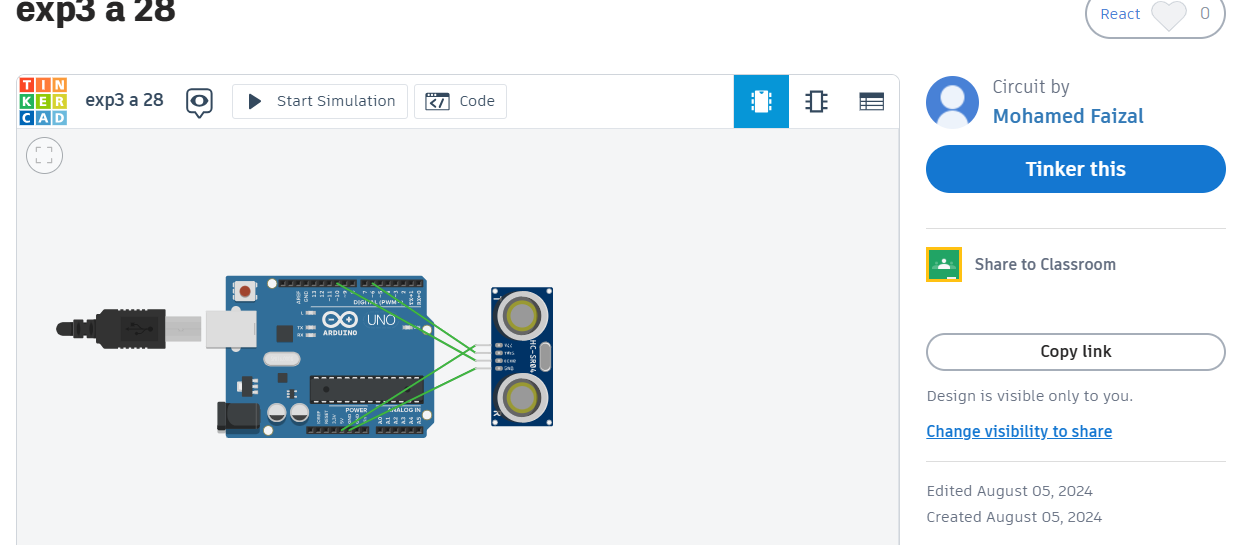
Serial.println("cm");

delay(100);

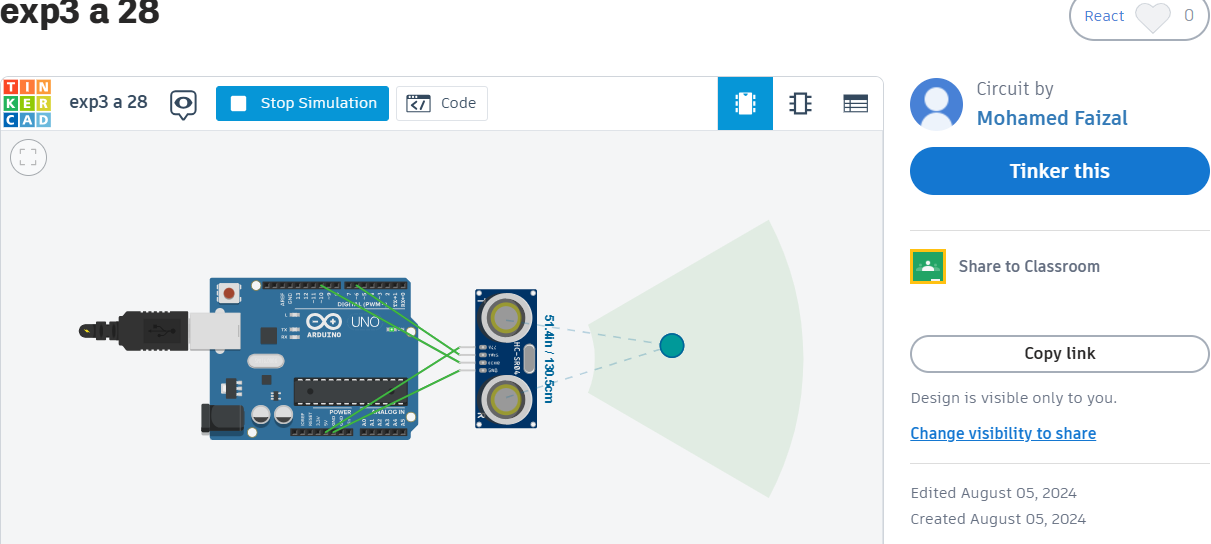
}

**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

**RESULT**

To create a simple Arduino circuit that uses an ultrasonic sensor to measure distance was executed successfully.

**Experiment no : 03 (b)**

**Date: 05.08.2024**

**DISTANCE MEASUREMENTS USING UV SENSOR WITH ARDUINO**

**AIM**

To design a distance measurement system using an ultrasonic sensor with Arduino, where the distance to an object is displayed on an LCD, and a buzzer alerts the user if the object is too close or too far

**PROCEDURE**

**Step-1:** Connect the ultrasonic sensor (HC-SR04) to the Arduino. Attach the LCD to display the distance reading, and set up a buzzer to alert if the distance is outside the specified range.

**Step-2**: Write code to send and receive ultrasonic signals from the sensor, calculate the distance based on the time it takes for the signal to return, and convert it to centimeters.

**Step-3**: Initialize the LCD and set up the cursor positions to display the text "Distance is:" along with the calculated distance in real time.

**Step-4:** Implement a conditional check where the buzzer is activated if the distance falls below 90 cm or goes above 300 cm.

**Step-5:** Simulate and test the setup in Tinker cad to ensure accurate distance measurements, LCD display, and buzzer alert functionality

**PROGRAM**

#include <LiquidCrystal.h> // includes the LiquidCrystal Library

LiquidCrystal lcd(1, 2, 4, 5, 6, 7);

const int trigPin = 9;

const int echoPin = 10;

long duration;

int distanceCm, distanceInch;

void setup() {

lcd.begin(16,2

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distanceCm= duration\*0.034/2;

distanceInch = duration\*0.0133/2;

lcd.setCursor(0,0);

lcd.print("Distance: ");

lcd.print(distanceCm)

lcd.print(" cm");

delay(10);

lcd.setCursor(0,1);

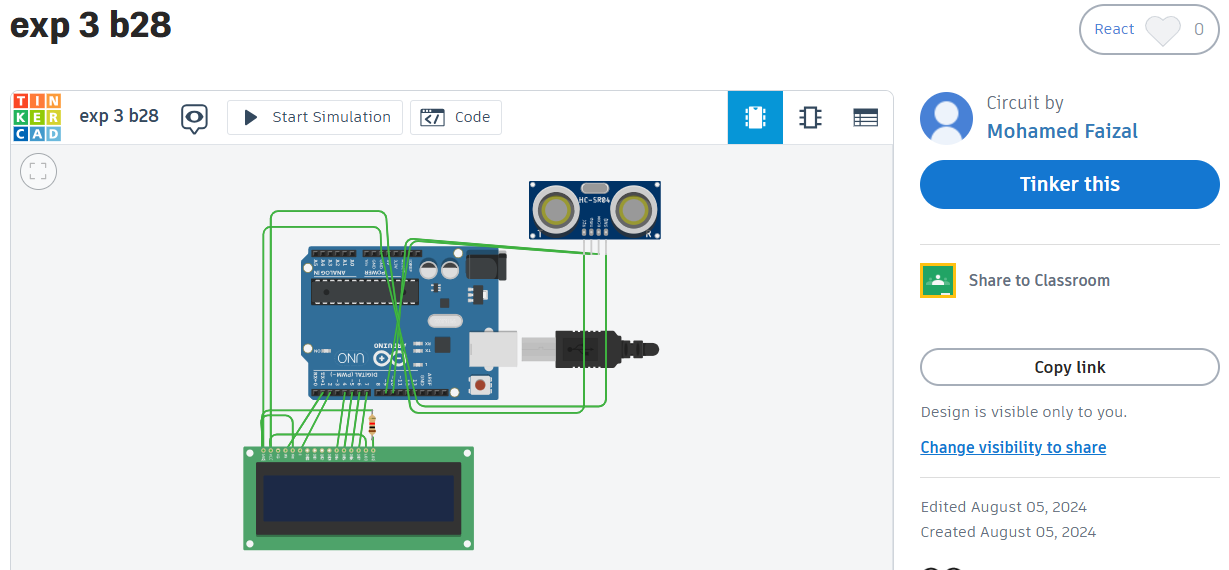
lcd.print("Distance: ");

lcd.print(distanceInch);

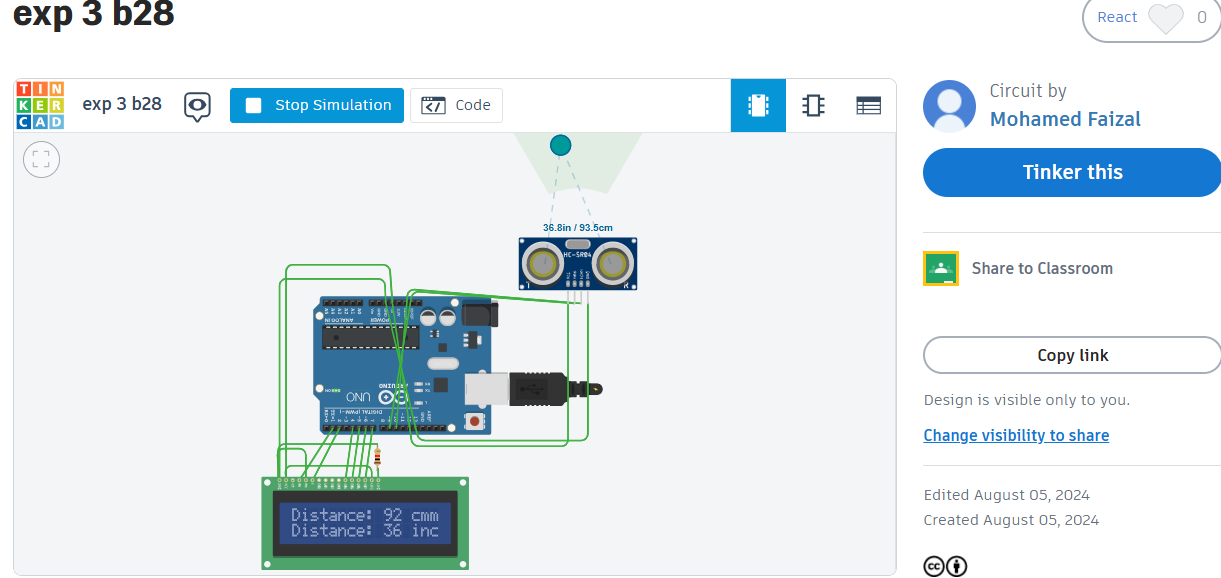
lcd.print(" inch");

delay(10);}

**OUTPUT**

**BEFORE SIMULATION**

**AFTER SIMULATION**

****

**RESULT**

To design a distance measurement system using an ultrasonic sensor with Arduino, where the distance to an object is displayed on an LCD, and a buzzer alerts the user if the object is too close or too far was executed successfully

**Experiment no : 04**

**Date : 19.08.2024**

**SMOKE DETECTION USING ARDUINO**

**AIM**

To design and simulate a gas detection system using an Arduino Uno, a gas sensor, an LED indicator, and a buzzer to detect the presence of gas and alert users accordingly.

**PROCEDURE**:

**Step-1**: Connect the gas sensor to analog input A0, red LED to digital pin2, green LED to digital pin 1, and buzzer to digital pin 0 on the Arduino Uno.

**Step-2**: In the Arduino code, define the gas sensor as an input and the LED sand buzzer as outputs.

**Step-3:** Define a threshold value (e.g., 600) for gas concentration to differentiate between normal and high levels.

**Step-4:** Continuously read the analog value from the gas sensor in the `loop()` function and compare it with the threshold.

**Step-5:** If the gas level is above the threshold, activate the buzzer and red LED(alert state). Otherwise, turn on the green LED (normal state).

**PROGRAM**

void setup() {

pinMode(gasSensor, INPUT);

pinMode(buzzer, OUTPUT);

pinMode(ledGreen, OUTPUT);

pinMode(ledRed, OUTPUT);

}

void loop() {

int gas\_value = analogRead(gasSensor);

if(gas\_value > HIGH)

{

tone(buzzer,1000,500);

digitalWrite(ledRed, HIGH);

digitalWrite(ledGreen,LOW);

}

else

{

noTone(buzzer);

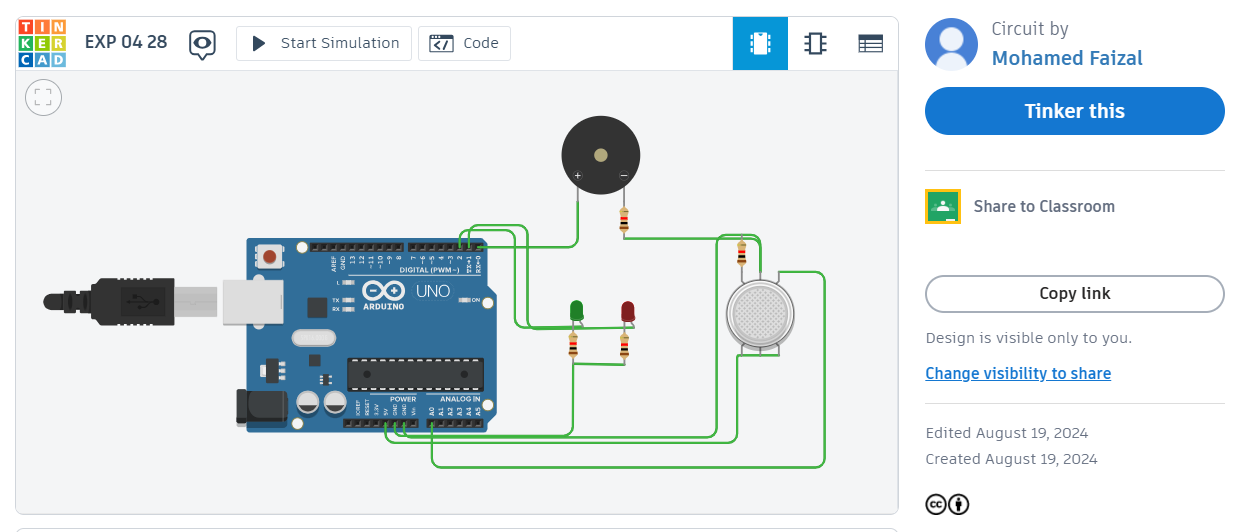
digitalWrite(ledGreen,HIGH);

digitalWrite(ledRed, LOW);

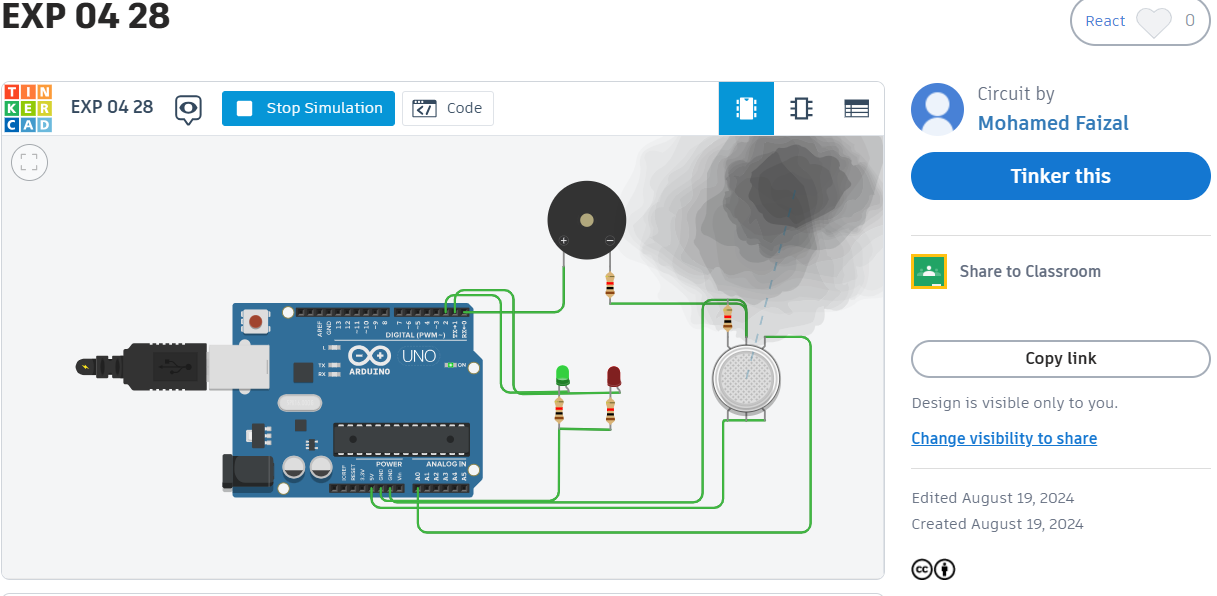
}

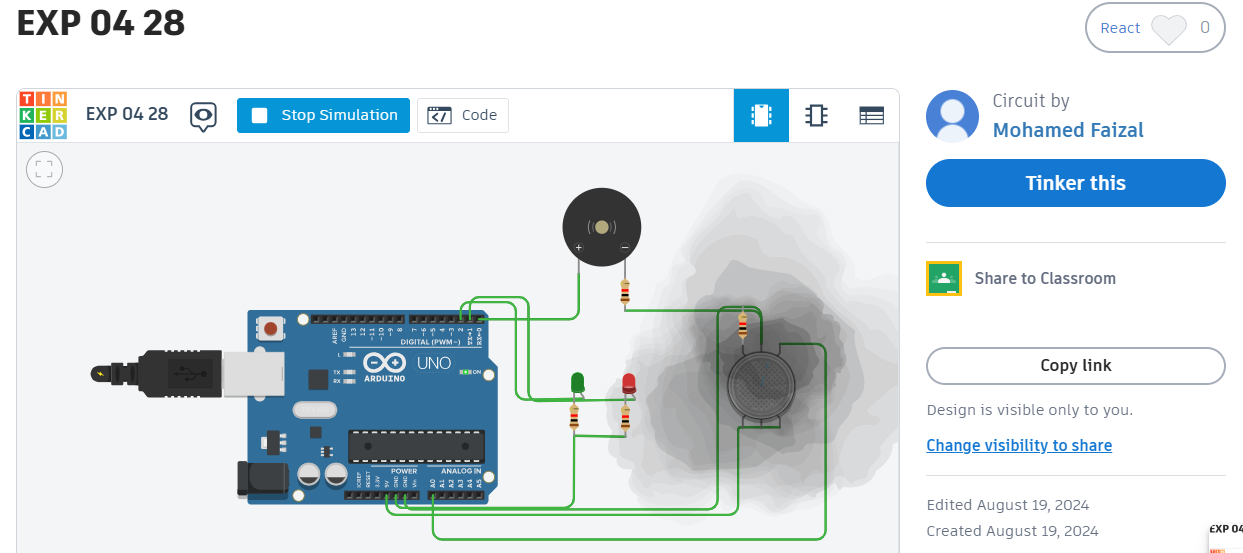
delay(200); }

**OUTPUT**

**BEFORE SIMULATION**

**AFTER SIMULATION**

****

****

**RESULT**

To design and simulate a gas detection system using an Arduino Uno, a gas sensor, an LED indicator, and a buzzer to detect the presence of gas and alert users

**Experiment no : 05**

**Date : 02.09.2024**

**SMART HOME AUTOMATION**

**AIM**

To design and implement a home automation system using an Arduino microcontroller, sensors (gas, PIR, ultrasonic, temperature), an LCD display, RGBLED, and a buzzer to monitor environmental conditions and trigger alerts when certain thresholds are met.

**PROCEDURE**

**Step-1**: Design a home automation system using Arduino to monitor gas, motion, temperature, and distance, and trigger alerts when specified conditions are met.

**Step-2:** Set up the gas sensor, PIR motion sensor, ultrasonic sensor, temperature sensor, RGB LED, buzzer, and LCD with the Arduino based on the schematic provided.

**Step-3:** Program the Arduino to detect specific conditions like gas leakage, motion, close objects, and high temperature, with predefined threshold values.

**Step-4:** Configure the system to activate the buzzer, display alerts on the LCD, and change LED colors to red for alerts and green for normal status.

**Step-5**: Test the system under different scenarios, and fine-tune sensor there holds for reliable and accurate performance.

**PROGRAM**

int output1Value = 0;

int sen1Value = 0;

int sen2Value = 0;

int const gas\_sensor = A1;

int const LDR = A0;

int limit = 400;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

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

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

return pulseIn(echoPin, HIGH);

}

Servo servo\_7;

void setup()

{

Serial.begin(9600);

pinMode(A0, INPUT);

pinMode(A1,INPUT

pinMode(12, OUTPUT);

servo\_7.attach(7, 500, 2500

pinMode(8,OUTPUT);

pinMode(9, INPUT);

pinMode(4, OUTPUT);

pinMode(3, OUTPUT);

}

void loop()

{

int val1 = analogRead(LDR);

if (val1 > 500)

{

digitalWrite(12, LOW);

Serial.print("Bulb ON = ");

Serial.print(val1);

}

else

{

digitalWrite(12, HIGH);

Serial.print("Bulb OFF = ");

Serial.print(val1);

}

sen2Value = digitalRead(9);

if (sen2Value == 0)

{

digitalWrite(10, LOW);

digitalWrite(4, HIGH);

digitalWrite(3, LOW

Serial.print(" || NO Motion Detected " );

}

if (sen2Value == 1)

{

digitalWrite(10, HIGH

delay(5000);

digitalWrite(4, LOW);

digitalWrite(3, HIGH)

Serial.print(" || Motion Detected! " );

}

int val = analogRead(gas\_sensor);

Serial.print("|| Gas Sensor Value = ");

Serial.print(val);

//val = map(val, 300, 750, 0, 100);

if (val > limit)

{

tone(8, 650);

}

delay(300);

noTone(8);

sen1Value = 0.01723 \* readUltrasonicDistance(6, 6);

if (sen1Value < 100)

{

servo\_7.write(90);

Serial.print(" || Door Open! ; Distance = ");

Serial.print(sen1Value);

Serial.print("\n");

}

else

{

servo\_7.write(0);

Serial.print(" || Door Closed! ; Distance = ");

Serial.print(sen1Value);

Serial.print("\n");

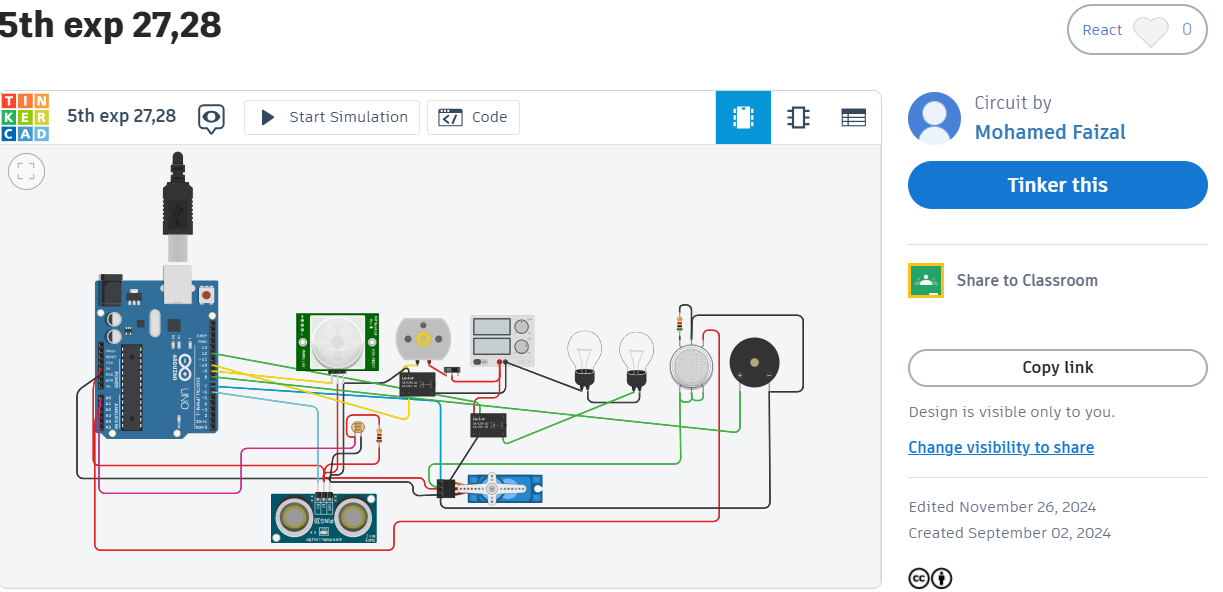
}

delay(10);

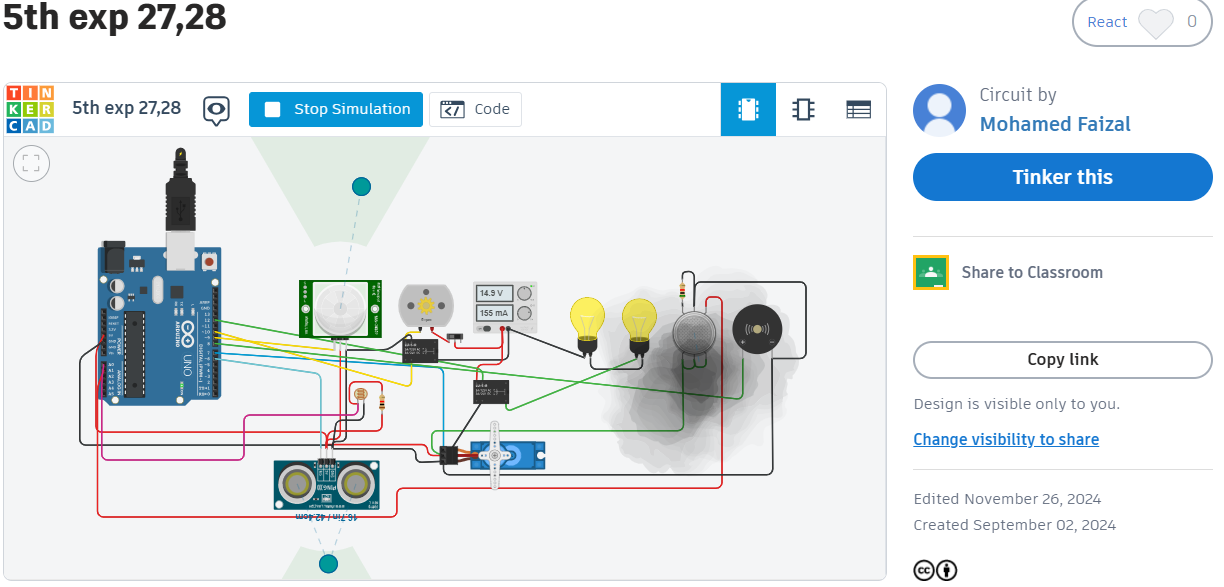
}

**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

**RESULT**

To design and implement a home automation system using an Arduino microcontroller, sensors (gas, PIR, ultrasonic, temperature), an LCD display, RGBLED, and a buzzer to monitor environmental conditions and trigger alerts when certain thresholds are met was executed successfully

**Experiment no : 06**

**Date : 23.09.2024**

**SEVEN SEGMENT**

**AIM**

To create a seven-segment display counter using an Arduino Uno that displays numbers from 0 to 9 sequentially.

**PROCEDURE**

**Step-1:** Connect each segment of the seven-segment display (labelled A to G) to the respective pins on the Arduino (Pins 1 to 7).

**Step-2**: Use constants in the code to define each pin connected to the segments. Set each pin as an output in the setup() function.

**Step-3:** Implement logic in the loop() function to turn on/off the appropriate segments for each digit (0-9).

**Step-4:** Use digital Write to control each segment, setting it to HIGH or LOW to form the required digit on the display.

**Step-5:** Use Tinker cad's simulator to test and debug the code, ensuring each number appears correctly on the seven-segment display.

**PROGRAM**

unsigned const int A=1;

unsigned const int B=2;

unsigned const int C=3;

unsigned const int D=4;

unsigned const int E=5;

unsigned const int F=7;

unsigned const int G=6;

void setup (void)

{

pinMode(A,OUTPUT);

pinMode(B,OUTPUT);

pinMode(C,OUTPUT);

pinMode(D,OUTPUT);

pinMode(E,OUTPUT);

pinMode(F,OUTPUT);

pinMode(G,OUTPUT);

}

void loop(void)

{

digitalWrite(A,HIGH);

digitalWrite(B,HIGH);

digitalWrite(C,HIGH);

digitalWrite(D,HIGH);

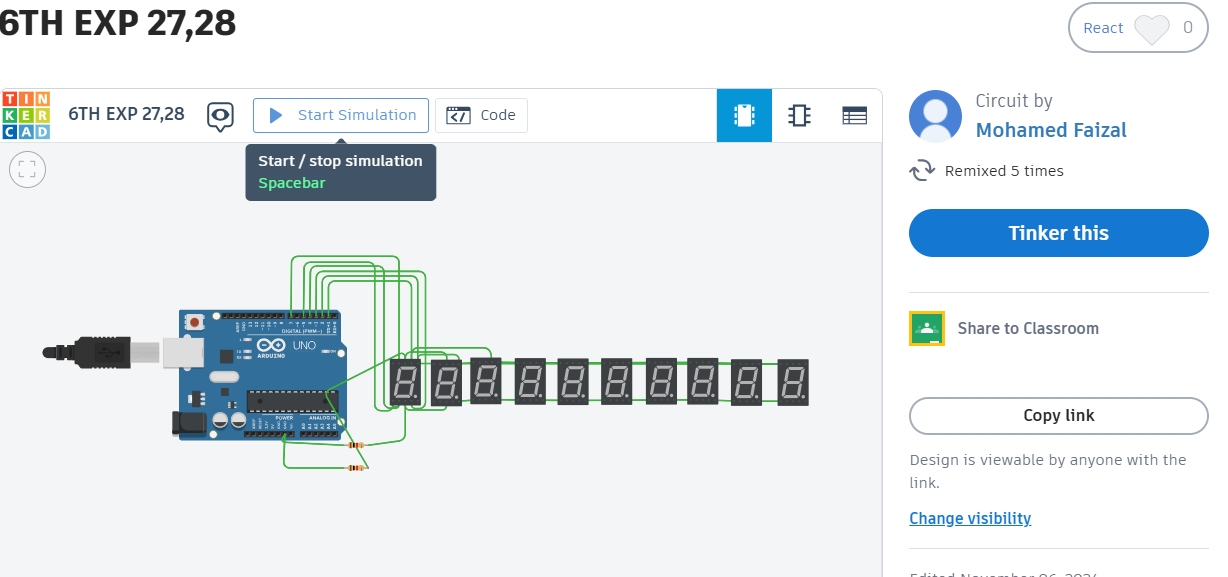
digitalWrite(E,HIGH);

digitalWrite(F,HIGH);

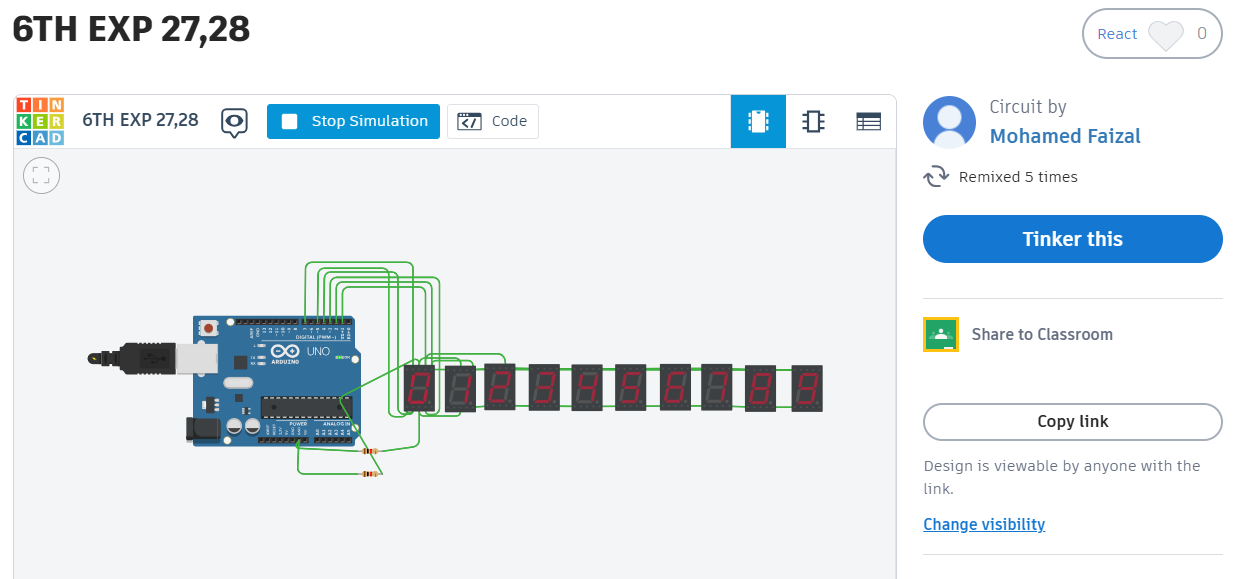
digitalWrite(G,HIGH); }

**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

**RESULT**

To create a seven-segment display counter using an Arduino Uno that displays numbers from 0 to 9 sequentially was executed successfully.

**Experiment no : 07**

**Date : 21.10.2024**

**SCROLLING DISPLAY**

**AIM**:

To create a temperature display system using an Arduino, TMP36 temperature sensor, and LCD screen, which continuously reads the temperature in Celsius and displays it as a scrolling message on the LCD.

**PROCEDURE**

**Step-1:** Attach the ultrasonic sensor to the Arduino and connect the LCDwithI2Ctothe respective pins.

**Step-2:** In the setup function, initialize the LCD and configure the ultrasonic sensor pins (Trig and Echo).

**Step-3**: Use pulse In() to measure the echo duration and calculate the distance in centimeter.

**Step-4:** Display the distance on the LCD and scroll it across the screen.

**Step-5:** Check the distance and trigger the buzzer if the distance is out of the desired range (less than 90 cm or more than 300 cm).

**Step-6:** Continuously update the distance on the LCD and refresh the display for real-time measurements.

**PROGRAM**

#include <LiquidCrystal.h>

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

#define temp A5

#define led 13

void setup()

{

lcd.begin(16, 2); // 16 columns and 2 rows

pinMode(led, OUTPUT);

pinMode(temp, INPUT);

Serial.begin(9600);

lcd.clear();

}

float pre\_temp = 0;

void loop() {

float temperature = 0;

temperature = (analogRead(temp) \* 0.48828125) - 27.85;

if(pre\_temp != temperature)

{

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

lcd.setCursor(0, 0); // Set cursor to the first line

lcd.print("Temp: "); // Print the label "Temp: "

lcd.print(temperature); // Print the temperature

lcd.print(" C"); // Print the unit "C"

pre\_temp = temperature;

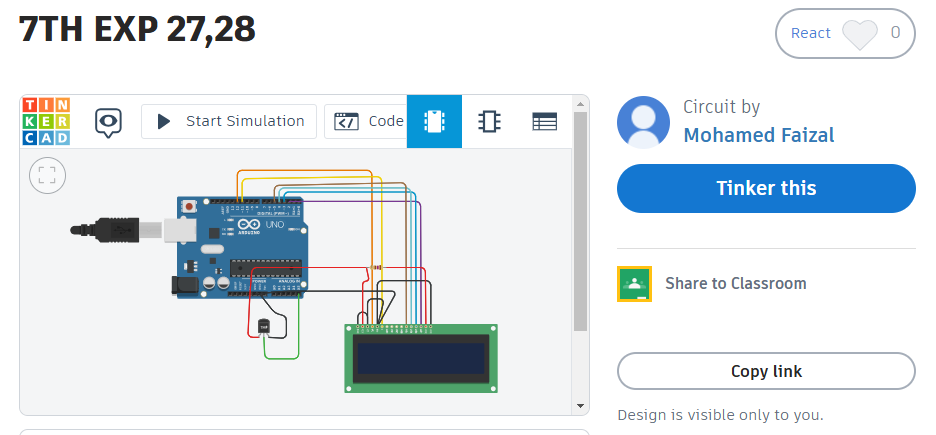
}

lcd.scrollDisplayLeft();

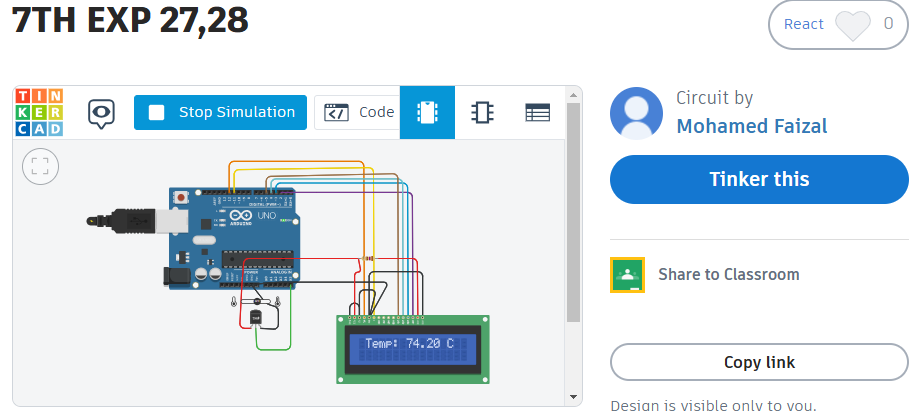
delay(100);}

**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

**RESULT**

To create a temperature display system using an Arduino, TMP36 temperature sensor, and LCD screen, which continuously reads the temperature in Celsius and displays it as a scrolling message on the LCD was executed successfully**.**

**Experiment no : 08**

**Date : 28.10.2024**

**LDR SENSOR**

**AIM**

The aim of the LDR sensor project is to detect ambient light levels and automate responses, such as turning on lights or controlling curtains, based on the surrounding light intensity.

**PROCEDURE**

**Step-1:** Connect the LDR with a resistor in a voltage divider to an analog pin (e.g., A0) on the Arduino.

**Step-2**: Attach two bulbs with resistors to digital pins (e.g., pins 8 and 9) for output control.

**Step-3:** Connect the servo motor to a PWM pin (e.g., pin 10) and ensure proper power and ground connections.

**Step-4**: Program the Arduino to read LDR values, turn bulbs on/off, and adjust the servo motor based on light intensity.

**Step-5:** Test the setup in various lighting conditions and fine-tune the code for accurate automation.

**PROGRAM**

int LDR\_Pin = A0;

int light\_threshold = 300;

int ledPins = 3;

int motorPin = 9;

void setup() {

pinMode(ledPins, OUTPUT);

pinMode(motorPin, OUTPUT);

Serial.begin(9600);

}

void loop() {

int lightValue = analogRead(LDR\_Pin);

Serial.println(lightValue);

if (lightValue < light\_threshold) {

digitalWrite(ledPins, HIGH);

analogWrite(motorPin, 300);

} else {

digitalWrite(ledPins, LOW);

analogWrite(motorPin, 180);

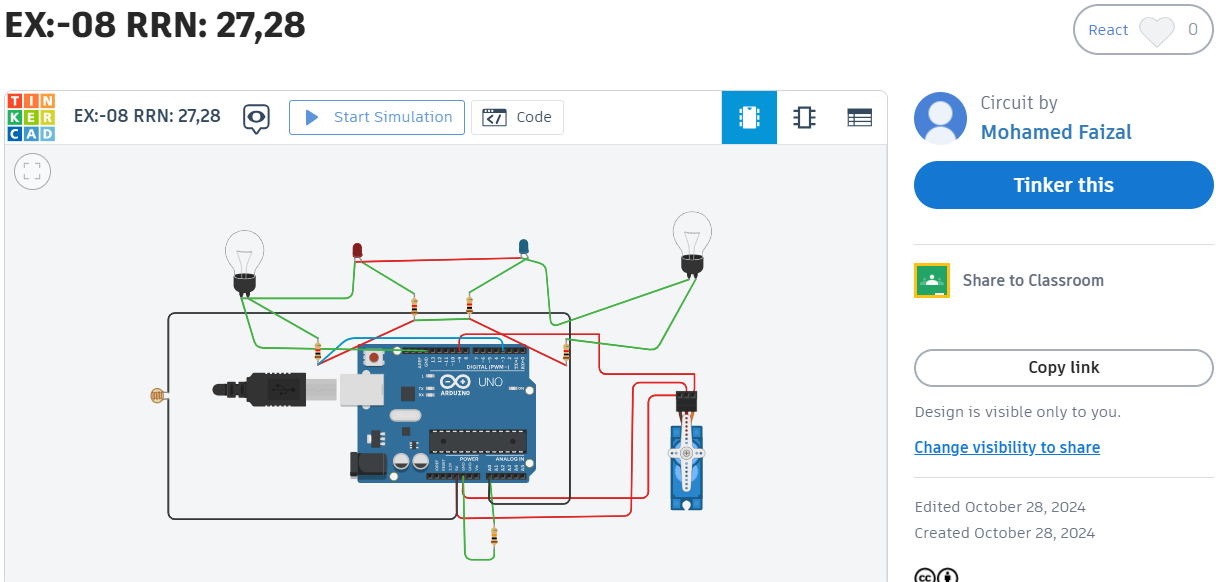
}

delay(1000);

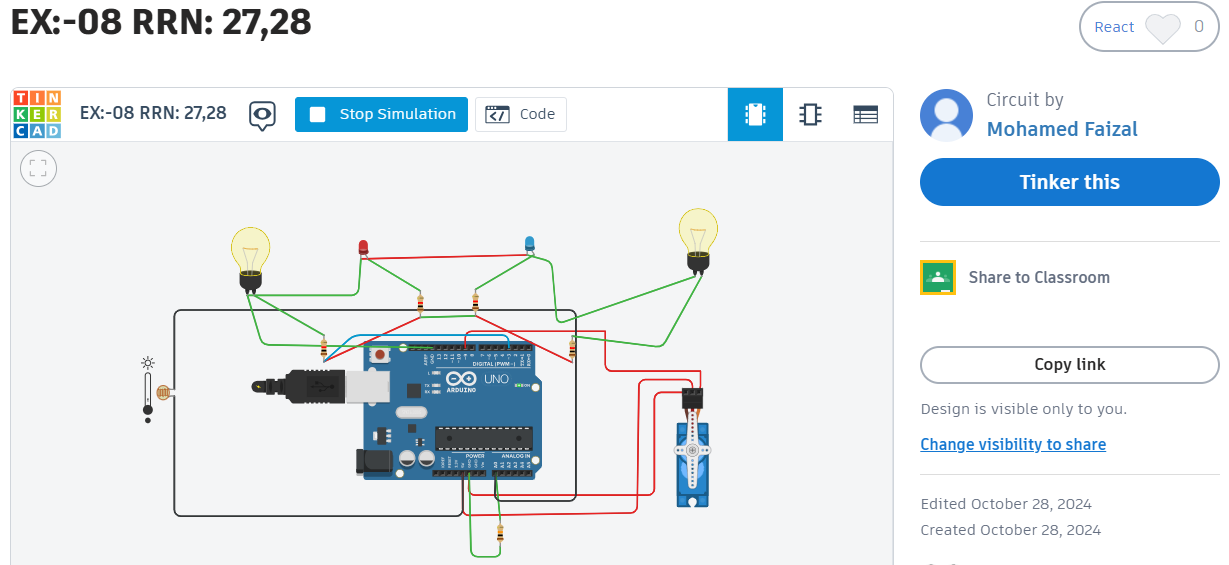
}

**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

**RESULT**

The LDR sensor project is to detect ambient light levels and automate responses, such as turning on lights or controlling curtains, based on the surrounding light intensity was executed successfully

**Experiment no : 09**

**Date : 04.11.2024**

**HOME SECURITY SYSTEM**

**AIM**

The aim is to design and implement an Home Security System that enhance ssafety by allowing secure access control using a keypad and displays system status on an LCD.

**PROCEDURE**

**Step 1:** Connect the Arduino Uno, keypad, LCD, servo motor, and buzzer to the appropriate pins.

**Step 2:** Write the code for handling keypad input, displaying messages on the LCD, and controlling the servo motor.

**Step 3:** Upload the code and test the system with both correct and incorrect passwords to ensure it works properly.

**Step 4:** Place the system in the desired location and ensure it has a stable power supply.

**Step 5:** Add extra features like IoT integration or an RFID system for advanced functionality.

**PROGRAM**

#include <Keypad.h>

#include <LiquidCrystal.h>

#include <Servo.h>

#define Password\_Length 5

Servo myservo;

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);

int pos = 0;

char Data[Password\_Length];

char Master[Password\_Length] = "1234";

byte data\_count = 0, master\_count = 0;

bool Pass\_is\_good;

bool door = false;

char customKey;

const byte ROWS = 4;

const byte COLS = 4;

char keys[ROWS][COLS] = {

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

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

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

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

};

byte rowPins[ROWS] = {0, 1, 2, 3};

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

Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup()

{

myservo.attach(9, 2000, 2400);

ServoClose();

lcd.begin(16, 2);

lcd.print("Protected Door");

loading("Loading");

lcd.clear();

}

void loop()

{

if (door == true)

{

customKey = customKeypad.getKey();

if (customKey == '#')

{

lcd.clear();

ServoClose();

lcd.print("Door is closed");

delay(3000);

door = false;

}

}

else

Open();

}

void loading (char msg[]) {

lcd.setCursor(0, 1);

lcd.print(msg);

for (int i = 0; i < 9; i++) {

delay(1000);

lcd.print(".");

}

}

void clearData()

{

while (data\_count != 0)

{

Data[data\_count--] = 0;

}

return;

}

void ServoClose()

{

for (pos = 90; pos >= 0; pos -= 10) {

myservo.write(pos);

}

}

void ServoOpen()

{

for (pos = 0; pos <= 90; pos += 10) {

myservo.write(pos);

}

}

void Open()

{

lcd.setCursor(0, 0);

lcd.print("Enter Password");

customKey = customKeypad.getKey();

if (customKey)

{

Data[data\_count] = customKey;

lcd.setCursor(data\_count, 1);

lcd.print(Data[data\_count]);

data\_count++;

}

if (data\_count == Password\_Length - 1)

{

if (!strcmp(Data, Master))

{

lcd.clear();

ServoOpen();

lcd.print(" Door is Open ");

door = true;

delay(5000);

loading("Waiting");

lcd.clear();

lcd.print(" Time is up! ");

delay(1000);

ServoClose();

door = false;

}

else

{

lcd.clear();

lcd.print(" Wrong Password ");

door = false;

}

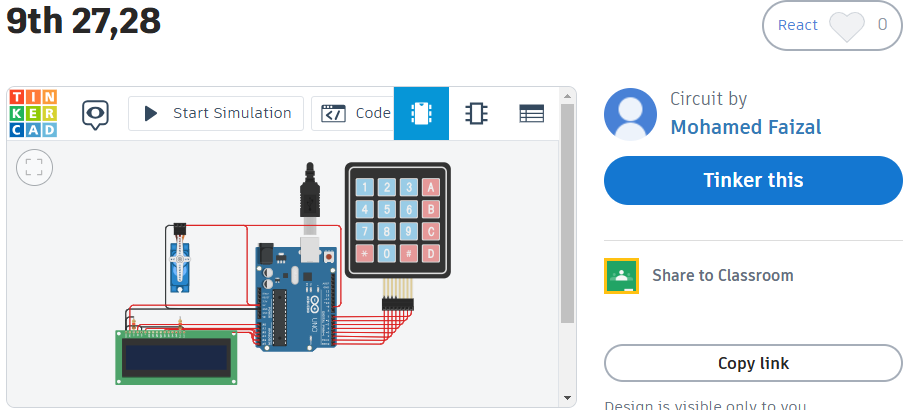
delay(1000);

lcd.clear();

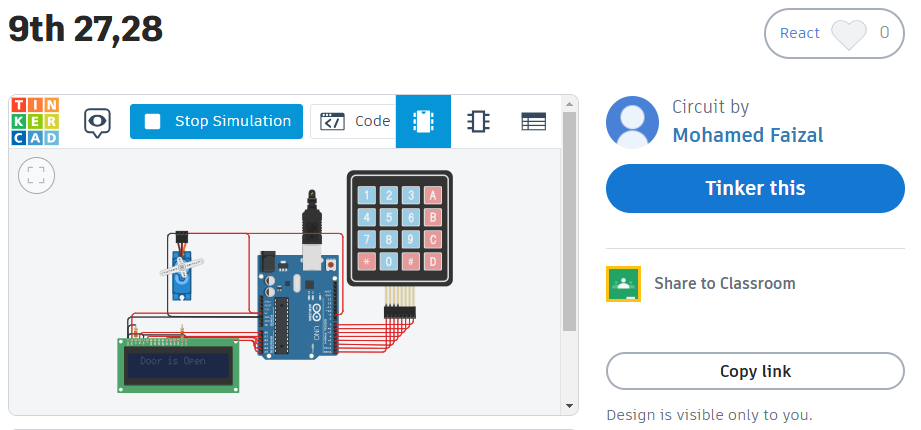
clearData(); }}

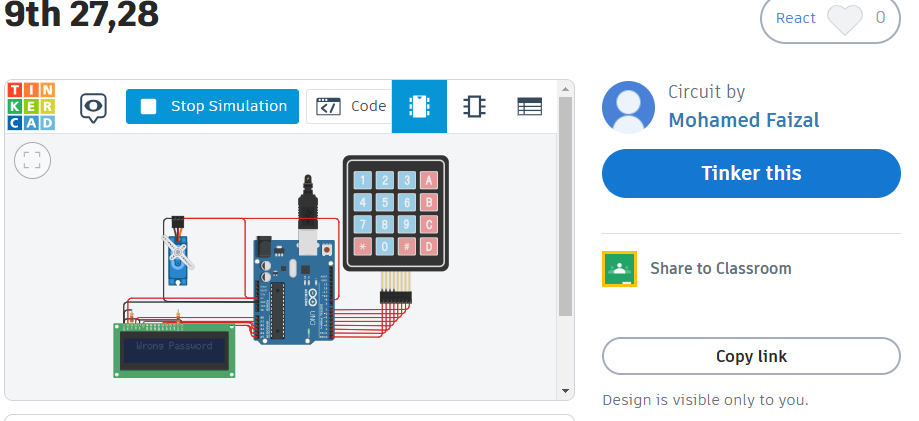
**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER** **SIMULATION**

****



**RESULT:**

The aim is to design and implement an Home Security System that enhances safety by allowing secure access control using a keypad and displays system status on an LCD.

**Experiment no : 10**

**Date : 25.11.2024**

**HOME AUTOMATION SYSTEM USING BLYNK**

**AIM**

To design and implement a home automation IoT system for monitoring gas levels, temperature, humidity, and water levels, and to enable real-time control and data visualization using the Blynk app for enhanced safety, convenience, and efficiency in home management.

**PROCEDURE**

**Step 1:** Hardware SetupConnect the sensors (gas, temperature, humidity, and water level) to a compatible microcontroller ESP8266 Ensure proper power connections and test each sensor to verify readings.

**Step2:** Firmware DevelopmentWrite the firmware using a programming environment like Arduino IDE. Include the Blynk library in the code to establish communication with the Blynk app. Configure the sensors to send data to the microcontroller and integrate them into the code logic.

**Step 3**: Blynk App ConfigurationCreate a new project in the Blynk app and generate an authentication token. Add widgets (e.g., gauge, label, slider) to display sensor data and control outputs. Configure the widgets to interact with the respective virtual pins used in the firmware.

**Step 4:** Testing and Calibration Upload the firmware to the microcontroller and connect it to the Blynk app using Wi-Fi. Monitor the real-time data in the app and adjust sensor calibration if needed. Test control functionality through the app to ensure commands are correctly executed.

**Step 5 :** Deployment and Monitoring Place the sensors in appropriate locations for gas detection, temperature, humidity monitoring, and water level sensing. Use the Blynk app to monitor environmental conditions and control connected devices. Ensure stable internet connectivity for continuous operation.

**PROGRAM**

#define BLYNK\_TEMPLATE\_ID "TMPL3KpBGpJg"

#define BLYNK\_DEVICE\_NAME "home automation""

#define BLYNK\_AUTH\_TOKEN "•••• - NyOW"

#include <BlynkSimpleEsp8266.h> // For ESP8266

#include <DHT.h>

// Wi-Fi credentials

char ssid[] = "Your\_SSID";

char pass[] = "Your\_Password";

// Blynk Auth Token

char auth[] = BLYNK\_AUTH\_TOKEN;

// Pin assignments

#define DHTPIN D2

#define DHTTYPE DHT11

#define GAS\_SENSOR\_PIN A0

#define WATER\_LEVEL\_PIN A1

DHT dht(DHTPIN, DHTTYPE);

#define V0\_GAS\_LEVEL V0

#define V1\_TEMP V1

#define V2\_HUMIDITY V2

#define V3\_WATER\_LEVEL V3

void setup() {

Serial.begin(9600);

dht.begin();

Blynk.begin(auth, ssid, pass);

pinMode(GAS\_SENSOR\_PIN, INPUT);

pinMode(WATER\_LEVEL\_PIN, INPUT);

Serial.println("System started!");

}

void loop() {

Blynk.run();

int gasValue = analogRead(GAS\_SENSOR\_PIN);

Blynk.virtualWrite(V0\_GAS\_LEVEL, gasValue);

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

if (!isnan(temperature) && !isnan(humidity)) {

Blynk.virtualWrite(V1\_TEMP, temperature);

Blynk.virtualWrite(V2\_HUMIDITY, humidity);

} else {

Serial.println("Failed to read from DHT sensor!");

}

int waterLevel = analogRead(WATER\_LEVEL\_PIN);

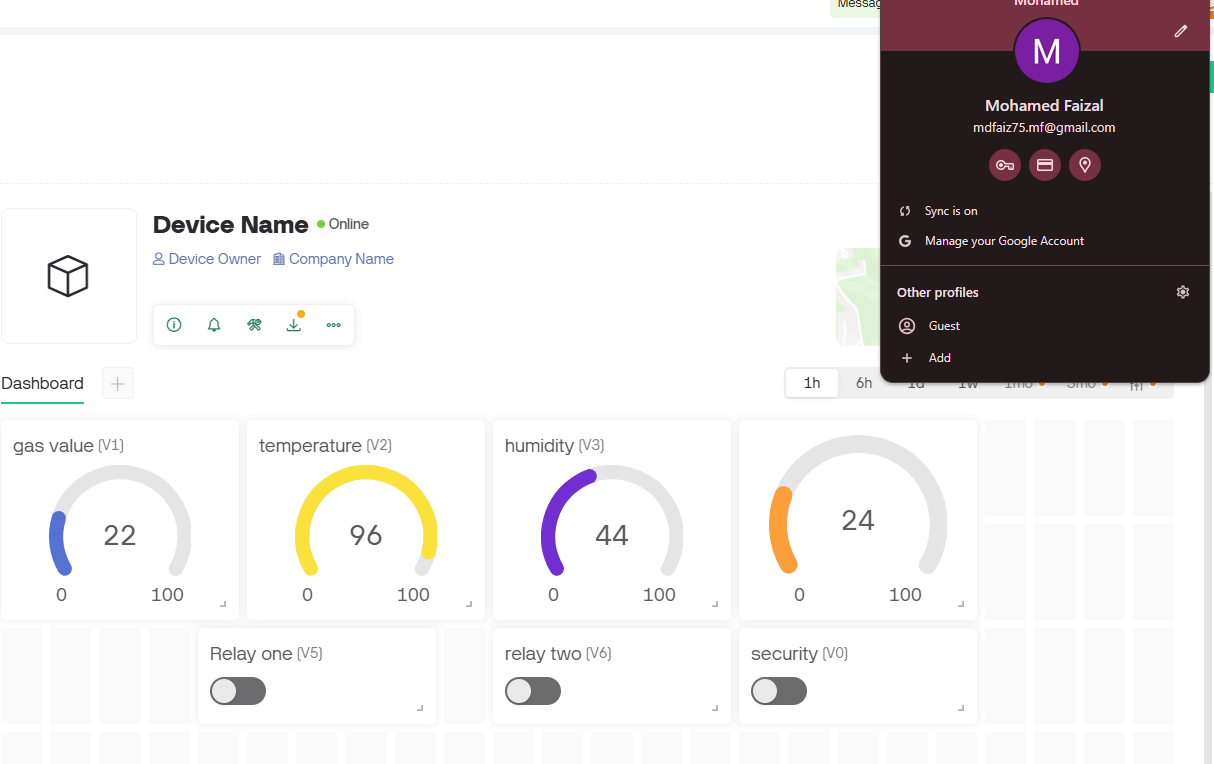
Blynk.virtualWrite(V3\_WATER\_LEVEL, waterLevel);

delay(2000); // Delay for stable readings

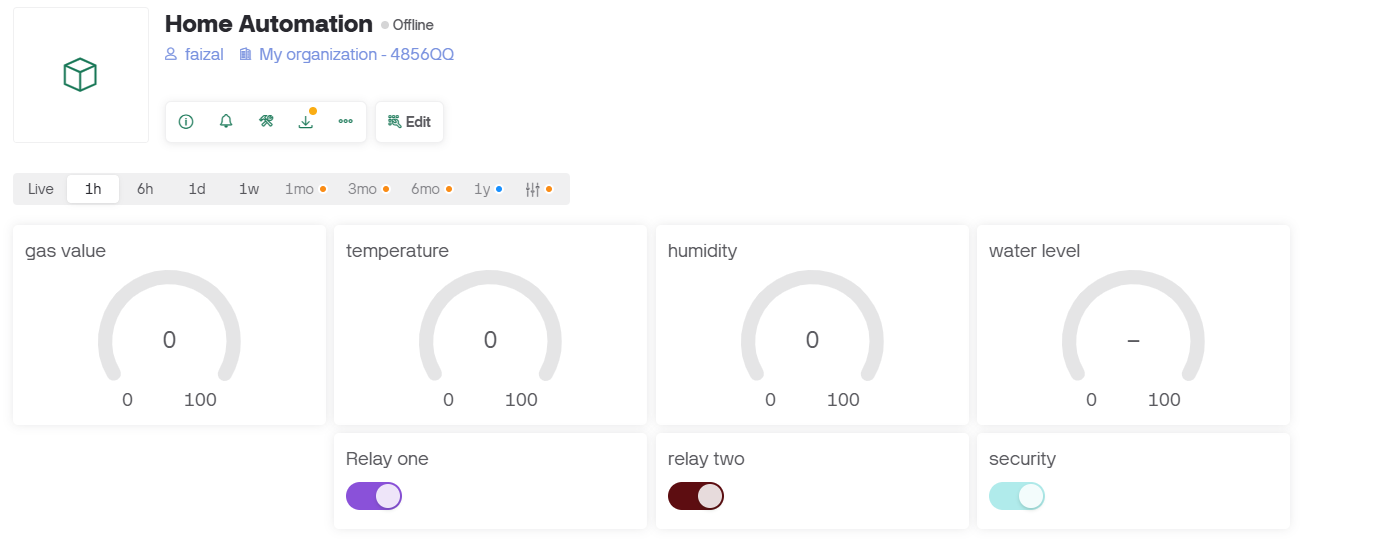
}

**OUTPUT**

**BEFORE SIMULATION**

****

**AFTER SIMULATION**

****

**RESULT**

The project successfully monitors gas levels, temperature, humidity, and water levels, and provides real-time control through the Blynk app, ensuring improved safety, efficiency, and convenience in home automation.