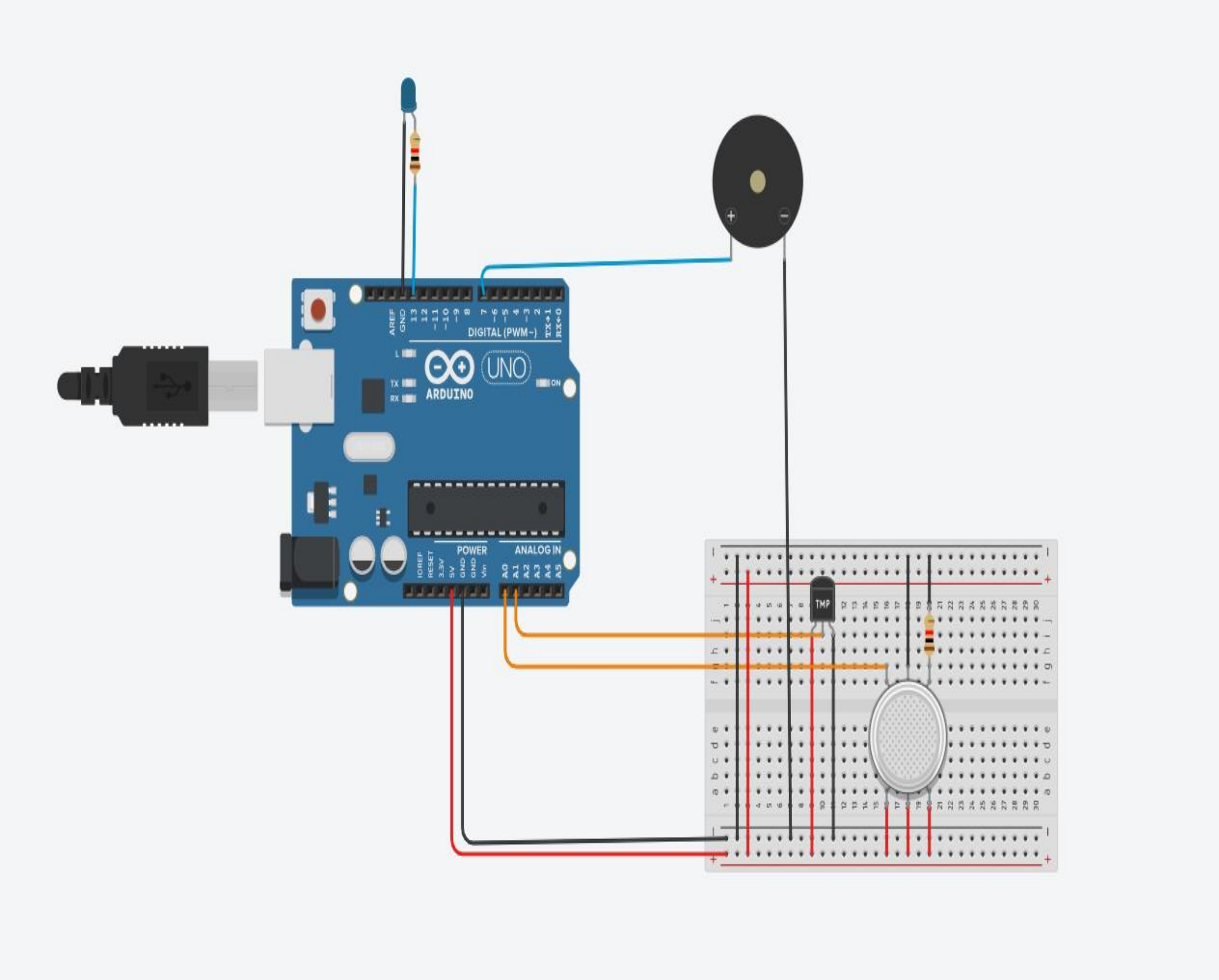
Sensor-Based Simulation

➢ Fire Alarm System Project by Interfacing Arduino with Temperature & Gas Sensor



Circuit diagram

Code:-

```
float temp;
float vout;
float vout1;
int LED = 13;
int gasSensor;
int piezo = 7;
void setup()
{
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(piezo, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  vout = analogRead(A1);
  vout1 = (vout / 1023) * 5000;
  temp = (vout1 - 500) / 10;
  gasSensor = analogRead(A0);
  if (temp >= 80)
  {
    digitalWrite(LED, HIGH);
  }
  else
  {
    digitalWrite(LED, LOW);
  }
  if (gasSensor >= 100)
  {
    digitalWrite(piezo, HIGH);
  }
  else
  {
    digitalWrite(piezo, LOW);
  }
  Serial.print("in DegreeC= ");
  Serial.print(" ");
  Serial.print(temp);
  Serial.print("\t");
  Serial.print("GasSensor= ");
  Serial.print(" ");
  Serial.print(gasSensor);
  Serial.println();
  delay(1000);
}
```

➢ Working of Code :-

This code reads two sensors:- 1) Temperature sensor on pin A1

2) Gas sensor on pin A0.

The temperature reading is converted from analog value into actual Celsius temperature. If the temperature goes above 80°C, the LED on pin 13 turns ON as a heat warning. The gas sensor gives an analog value based on the gas concentration. If this value goes above 100, the buzzer on pin 7 turns ON to warn about gas leakage. The program continuously prints both temperature and gas values on the Serial Monitor and updates readings every second. The main purpose of the code is to monitor safety conditions (heat and gas) and give alerts using an LED and a buzzer.

➢ Code explanation:-

- temp → final temperature in °C (float = decimal number).
- vout → raw analog value from A1 (0 to 1023).
- vout1 → converted voltage in millivolts.
- LED → LED is connected to pin 13.
- gasSensor → stores gas sensor reading (0 to 1023).
- piezo → buzzer is connected to pin 7.
- pinMode(A0, INPUT) → A0 reads gas sensor values.
- pinMode(A1, INPUT) → A1 reads temperature sensor values.
- pinMode(LED, OUTPUT) → Arduino will turn the LED ON/OFF.
- pinMode(piezo, OUTPUT) → Arduino will activate the buzzer.
- Serial.begin(9600) → Starts Serial Monitor at 9600 baud rate.
- vout = analogRead(A1); →Arduino reads value from A1.
- temp = (vout1 - 500) / 10; →This formula is used for TMP36 sensor**.**
- gasSensor = analogRead(A0); →Reads analog value from A0.
- If temperature >= **80°C**, LED lights up.
  Meaning: High temperature alert.
  Otherwise LED stays OFF.
- If gas value >= **100**, turn ON buzzer.

This means gas leakage alert.

If gas is safe → buzzer OFF.

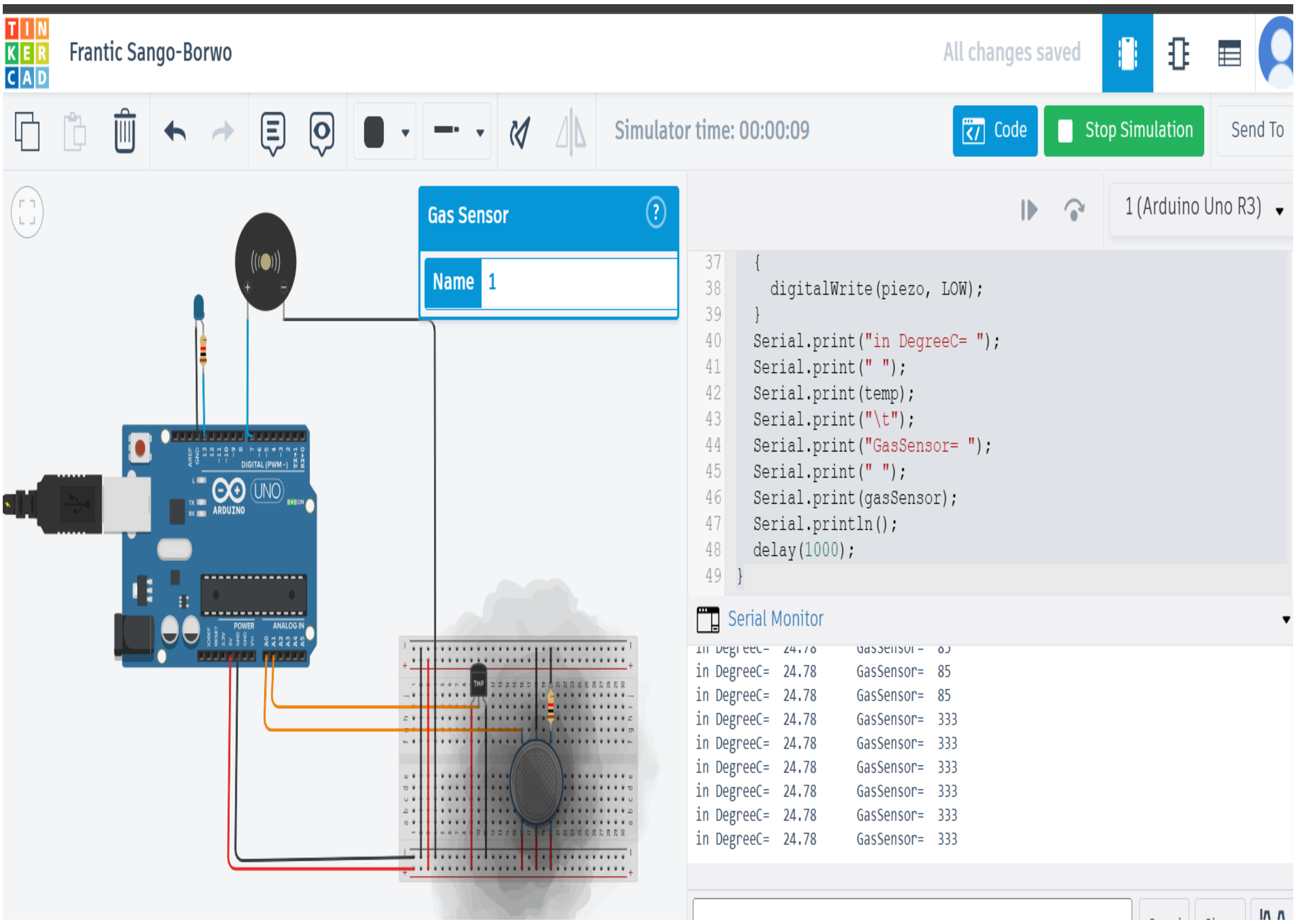
- Arduino waits 1 second before next reading.

➢ Components Used

- Arduino UNO
- TMP36 / LM35 Temperature Sensor
- Gas Sensor (MQ-type / Analog gas sensor)
- LED + Resistor
- Buzzer (Piezo)
- Breadboard
- Jumper wires

# Stimulation:-

- **Gas Sensor Simulation**



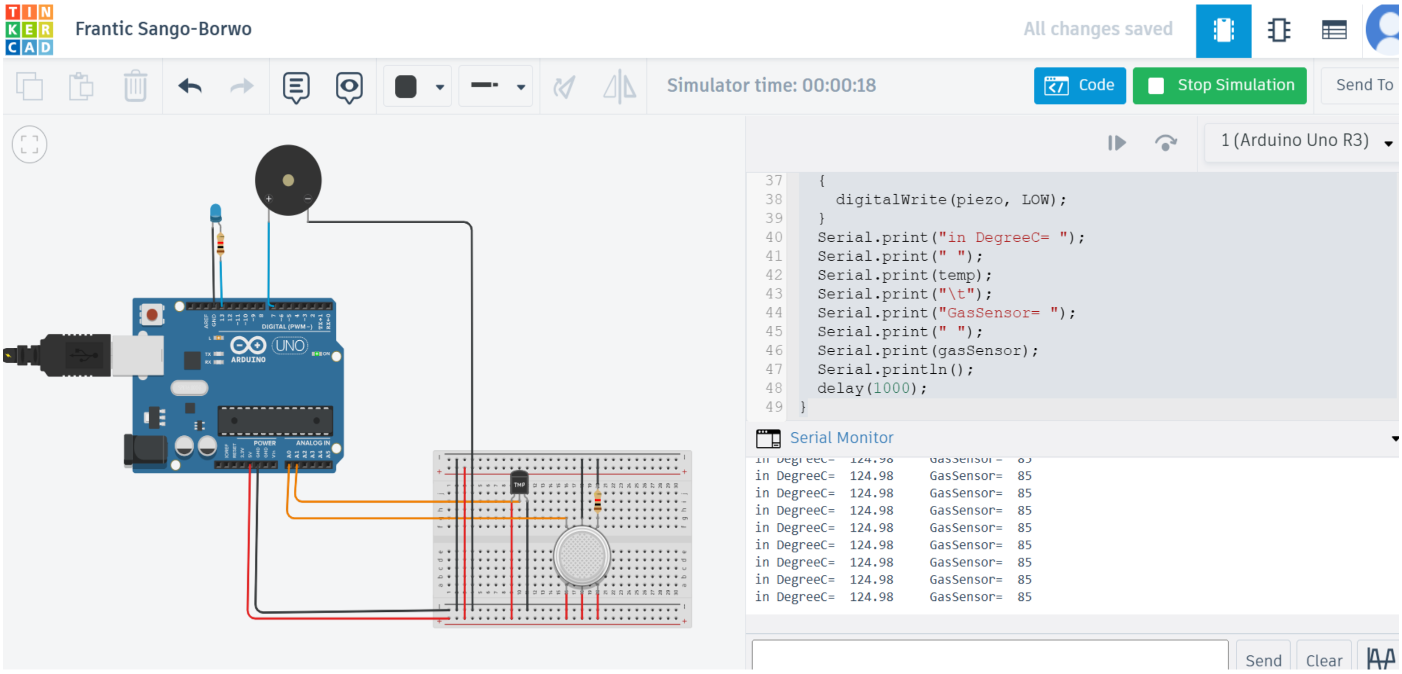
```
  {
    digitalWrite(piezo, LOW);
  }
  Serial.print("in DegreeC= ");
  Serial.print(" ");
  Serial.print(temp);
  Serial.print("\t");
  Serial.print("GasSensor= ");
  Serial.print(" ");
  Serial.print(gasSensor);
  Serial.println();
  delay(1000);
}
```

```
in DegreeC=  24.78      GasSensor=  85
in DegreeC=  24.78      GasSensor=  85
in DegreeC=  24.78      GasSensor=  85
in DegreeC=  24.78      GasSensor=  333
in DegreeC=  24.78      GasSensor=  333
in DegreeC=  24.78      GasSensor=  333
in DegreeC=  24.78      GasSensor=  333
in DegreeC=  24.78      GasSensor=  333
in DegreeC=  24.78      GasSensor=  333
```

In this simulation, the gas sensor is used to detect the presence of gas or smoke near the sensor. The Arduino continuously reads the sensor's output through the analog pin A0. When there is no smoke, the gas sensor gives a normal reading, and the buzzer stays OFF.

But when we simulate smoke in Tinkercad (by increasing the gas value manually), the gas sensor value becomes high. As soon as the reading crosses the danger limit (in the code, the limit is 100), the buzzer starts ringing to warn us about the gas leak or smoke.

- **Temperature Sensor Simulation**



```
  {
    digitalWrite(piezo, LOW);
  }
  Serial.print("in DegreeC= ");
  Serial.print(" ");
  Serial.print(temp);
  Serial.print("\t");
  Serial.print("GasSensor= ");
  Serial.print(" ");
  Serial.print(gasSensor);
  Serial.println();
  delay(1000);
}
```

```
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
in DegreeC=  124.98     GasSensor=  85
```

In this simulation, the temperature sensor (TMP36/LM35) sends a voltage to the Arduino based on the current temperature. This voltage is converted into a temperature value in °C and printed on the Serial Monitor.

The Arduino constantly checks the temperature value. If the temperature becomes higher than the danger level (in this code the limit is 80°C), the LED connected to pin 13 turns ON to give a visual warning.

➢ Conclusion:-

In this project, a simple IoT-based safety monitoring system was successfully simulated using Arduino, a temperature sensor, a gas sensor, an LED, and a buzzer. The simulation demonstrated how real-time environmental data can be monitored and used to trigger safety alerts. The temperature sensor measured the surrounding heat level, and when the temperature crossed the danger limit, the LED automatically glowed to indicate overheating. Similarly, the gas sensor detected changes in gas concentration, and when the gas level became unsafe, the buzzer activated to warn about possible gas leakage or smoke.

The Serial Monitor clearly displayed both temperature and gas values, allowing us to observe how the system responds to different conditions.

This project proves how simple low-cost sensors and microcontrollers can be combined to create an efficient early-warning safety system for homes, kitchens, laboratories, and industries. It also reflects how IoT technologies help in building smarter and safer environments. Overall, the simulation worked successfully and demonstrated the practical use of sensor-based automation in real-life safety applications.