# Vivekanand Education Society's

## Institute of Technology

(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

## Department of Information Technology


# IOE Lab

## Assignment - 1

**Aim :** Explore IoT simulation tools.

| | |
|---|---|
| Name | Anjali punsi |
| Roll No. | 57 |
| Class | D20B |
| Subject | Internet of Everything |
| Grade: | |

**Aim: To explore IoT simulation tools**

**Theory:**

In this lab assignment, we will explore two popular IoT simulation tools: Cisco Packet Tracer and IoTIFY. We will provide a description of each tool, its advantages and disadvantages, and include screenshots for a sample case study as well as the results of the simulations.

## Tool 1: Cisco Packet Tracer

Cisco Packet Tracer is a versatile network simulation tool primarily designed for networking education and training. It offers several unique features that make it a valuable tool in the field of network and IoT education:

**Unique features:-**

**1. Educational Focus:**

Packet Tracer is specifically designed for educational purposes. It provides a safe and controlled environment for students to learn about networking and IoT concepts, experiment with different network topologies, and practice configuring network devices without the need for physical hardware.

**2. User-Friendly Interface:**

One of its standout features is its user-friendly drag-and-drop interface. This makes it accessible even to beginners who may not have extensive networking experience. Users can easily add, configure, and connect network devices.

**3.Multifunctional Device Support:**

Packet Tracer offers a wide range of network devices that can be used in simulations, including routers, switches, hubs, IoT devices, and more. This diversity allows users to create complex network topologies and IoT scenarios.

**4. Realistic Simulation:**

While Packet Tracer may not offer the highest level of realism compared to some commercial network simulation tools, it provides a reasonable level of realism for educational purposes. Users can simulate the behavior of devices, including IoT sensors and actuators, and observe how they interact within a network.

**5. Integrated Learning Materials:**

Cisco provides a wealth of educational resources and tutorials that are integrated into Packet Tracer. These materials include labs, activities, and guided exercises to help users learn networking and IoT concepts effectively.

**6. Multi-Platform Support:**

Packet Tracer is available for both Windows and Linux operating systems, ensuring that users on various platforms can access and use the tool.

**7.Packet Capture and Analysis**
:It allows users to capture and analyze network packets, helping them understand the flow of data and troubleshoot network issues. This feature is valuable for teaching and learning network analysis and troubleshooting.

**8.Protocols and IoT Support:**
While Packet Tracer may not cover all IoT protocols and standards, it does support a range of networking protocols, including IPv6, SNMP, DHCP, and more. This makes it suitable for basic IoT simulations and networking experiments.

**9.Community and Support**
Cisco Packet Tracer has a large user community and support forums where users can ask questions, share knowledge, and seek assistance with their networking and IoT projects. This community aspect enhances the learning experience.

**10.Scalability**: Users can create both small-scale and large-scale network simulations in Packet Tracer, allowing them to explore various scenarios and network designs.

**Advantages :**
- Educational Focus: Packet Tracer is an excellent tool for learning and teaching IoT concepts and network protocols. It is widely used in networking courses and has a large user community.
- User-Friendly: It offers an intuitive drag-and-drop interface, making it accessible even to beginners.
- Multi-Platform: Cisco Packet Tracer is available for Windows and Linux, allowing users on various platforms to access and use it.
- Extensive Device Library: It includes a vast library of IoT devices, routers, switches, and other network components for creating realistic simulations.

**Disadvantages:**
- Limited Realism: While Packet Tracer is great for educational purposes, it may not provide the level of realism required for advanced IoT projects.
- Limited IoT Protocols: It supports a limited set of IoT protocols, which might not cover the full range of IoT technologies and standards.

## Simulation using cisco Packet Tracer :

Title: Forest Fire Detection

Description :
Forest fire detection is the proactive and vital process of identifying and monitoring wildfires in natural environments, employing a combination of sensor networks, remote sensing technologies, data analytics, and human observation. These integrated systems, often in collaboration with local communities and multiple agencies, enable the early detection of wildfires, allowing for swift response, evacuation, and mitigation efforts to protect human lives, ecosystems, and natural resources, while also

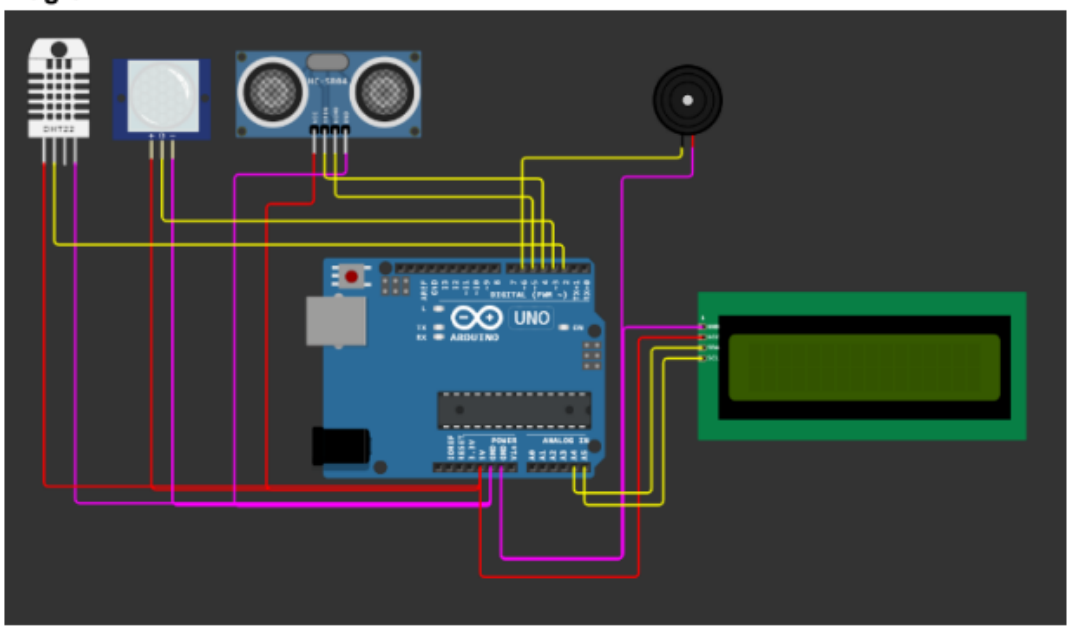
emphasizing prevention measures and environmental monitoring to reduce the risk and impact of devastating forest fires.

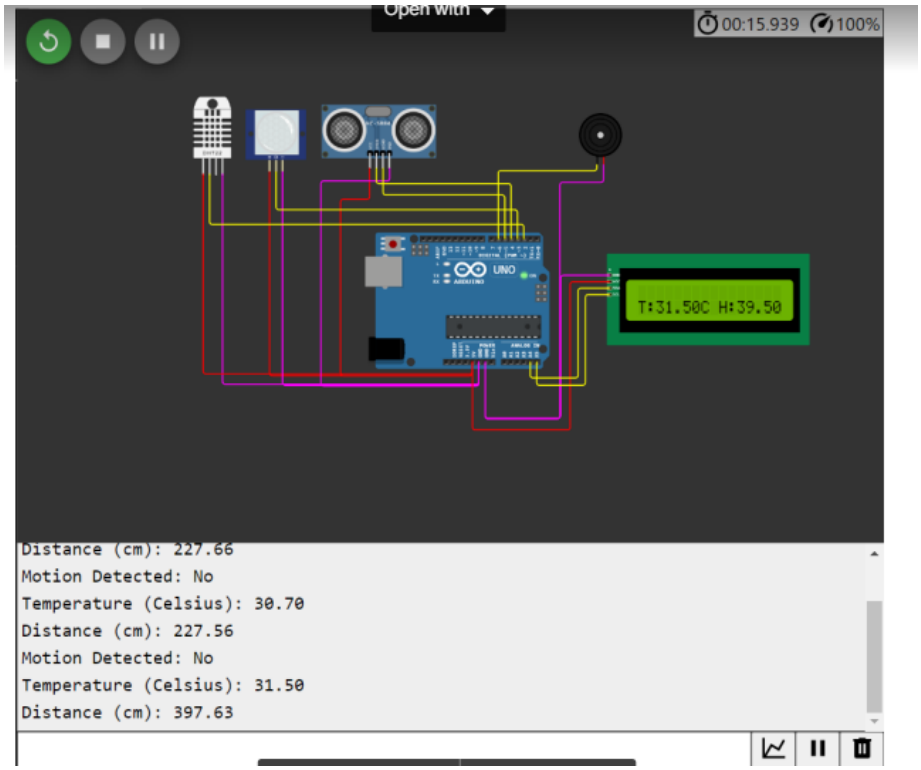
Code-

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTPIN 2 // Pin for the DHT22 sensor
#define DHTTYPE DHT22 // DHT sensor type
#define PIR_PIN 3 // Pin for the PIR sensor
#define TRIGGER_PIN 4 // Pin for the ultrasonic sensor trigger
#define ECHO_PIN 5 // Pin for the ultrasonic sensor echo
#define BUZZER_PIN 6 // Pin for the buzzer
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD I2C address (use the correct address for your module)
void setup() {
Serial.begin(9600);
dht.begin();
pinMode(PIR_PIN, INPUT);
pinMode(TRIGGER_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
pinMode(BUZZER_PIN, OUTPUT);
lcd.init(); // Initialize the LCD
lcd.backlight(); // Turn on the backlight
lcd.clear(); // Clear the LCD screen
lcd.setCursor(0, 0); // Set the cursor to the top-left corner
lcd.print("Monitoring System");
}
void loop() {
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();
int motionDetected = digitalRead(PIR_PIN);
float distance = measureDistance();
lcd.setCursor(0, 1); // Set cursor to the second row
lcd.print("T:");
lcd.print(temperature);

lcd.print("C H:");
lcd.print(humidity);
lcd.print("% ");
if (motionDetected || temperature > 35.0) { // Adjust the temperature threshold as needed
```

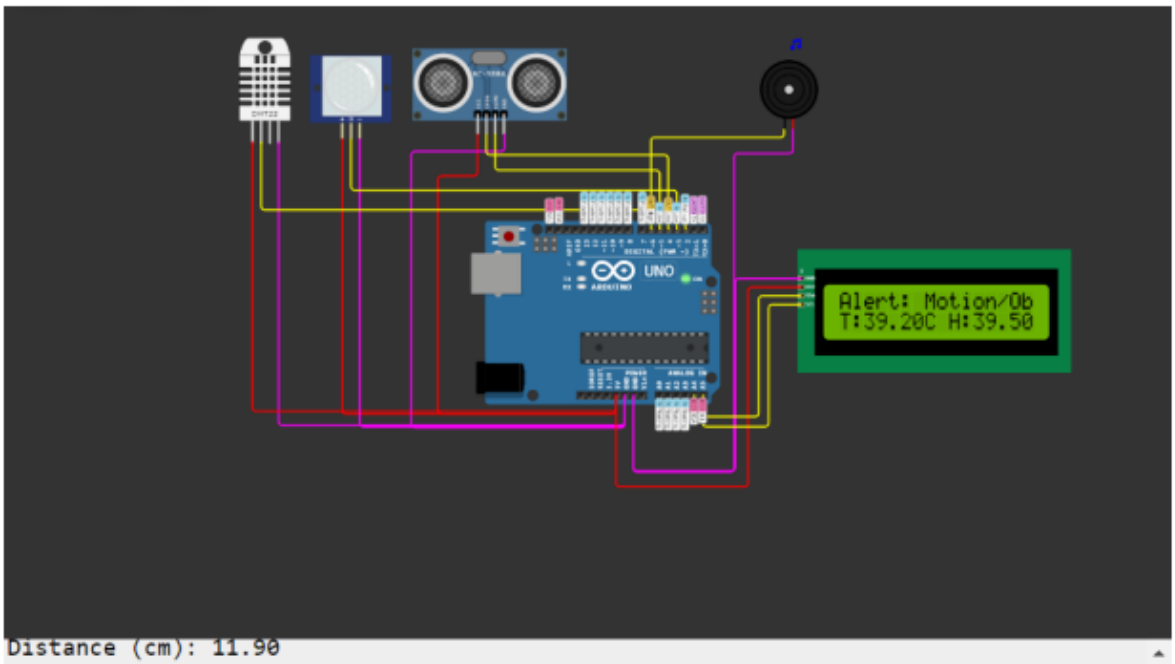
```
lcd.setCursor(0, 0); // Set cursor to the first row
lcd.print("Alert: Motion/Obs!");
tone(BUZZER_PIN, 1000); // Turn on the buzzer at 1 kHz
delay(500); // Buzz for half a second
noTone(BUZZER_PIN); // Turn off the buzzer
delay(500); // Delay to prevent rapid alerts
} else {
lcd.setCursor(0, 0); // Set cursor to the first row
lcd.print(" "); // Clear the previous message
}
Serial.print("Motion Detected: ");
Serial.println(motionDetected ? "Yes" : "No");
Serial.print("Temperature (Celsius): ");
Serial.println(temperature);
Serial.print("Distance (cm): ");
Serial.println(distance);
delay(5000); // Read data every 1 second
}
float measureDistance() {
digitalWrite(TRIGGER_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIGGER_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN, LOW);
long duration = pulseIn(ECHO_PIN, HIGH);
// Calculate distance in centimeters
float distance = duration * 0.034 / 2;
return distance;
}
```

## Diagram


UNO
ARDUINO

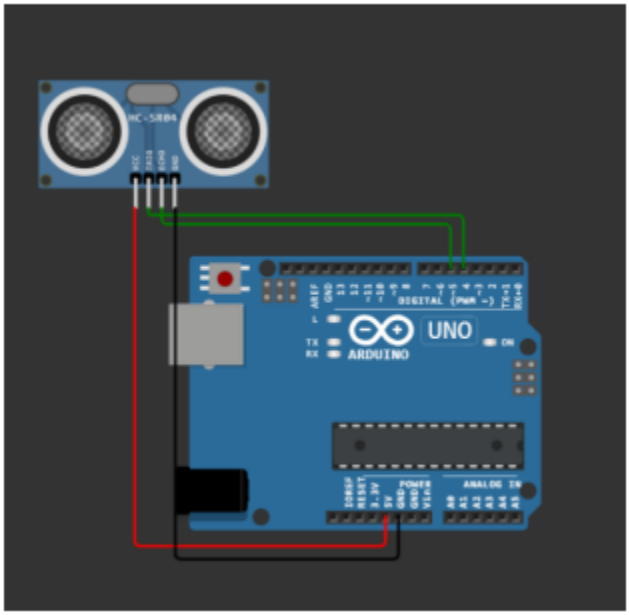

Open with
00:15.939
100%
UNO
T:31.50C H:39.50
Distance (cm): 227.66
Motion Detected: No
Temperature (Celsius): 30.70
Distance (cm): 227.56
Motion Detected: No
Temperature (Celsius): 31.50
Distance (cm): 397.63

**Active Condition:**




```
Distance (cm): 11.90
Motion Detected: Yes
Temperature (Celsius): 39.20
Distance (cm): 12.00
Motion Detected: No
Temperature (Celsius): 39.20
Distance (cm): 11.90
```

**Connecting Sensor:**



**Tool no - 2- Tinkercad**
Simulation using Tinkercad:

**Title: Forest Fire Detection**

**Description :**
Forest fire detection is the proactive and vital process of identifying and monitoring wildfires in natural environments, employing a combination of sensor networks, remote sensing technologies, data analytics, and human observation. These integrated systems, often in collaboration with local communities and multiple agencies, enable the early detection of wildfires, allowing for swift response, evacuation, and mitigation efforts to protect human lives, ecosystems, and natural resources, while also emphasizing prevention measures and environmental monitoring to reduce the risk and impact of devastating forest fires.

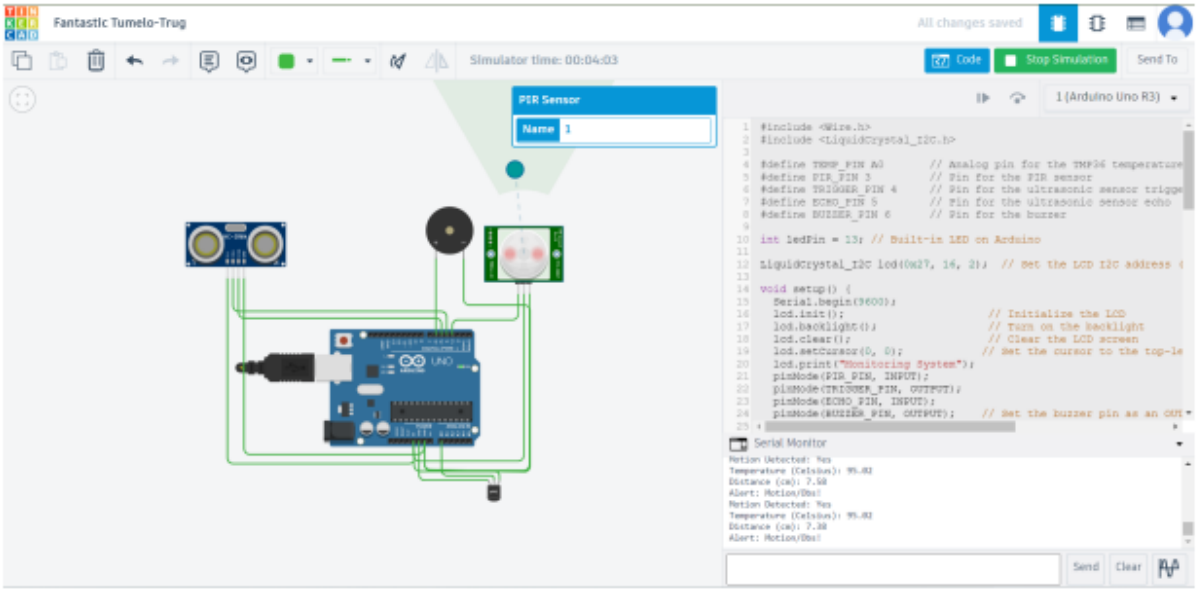
**Code**

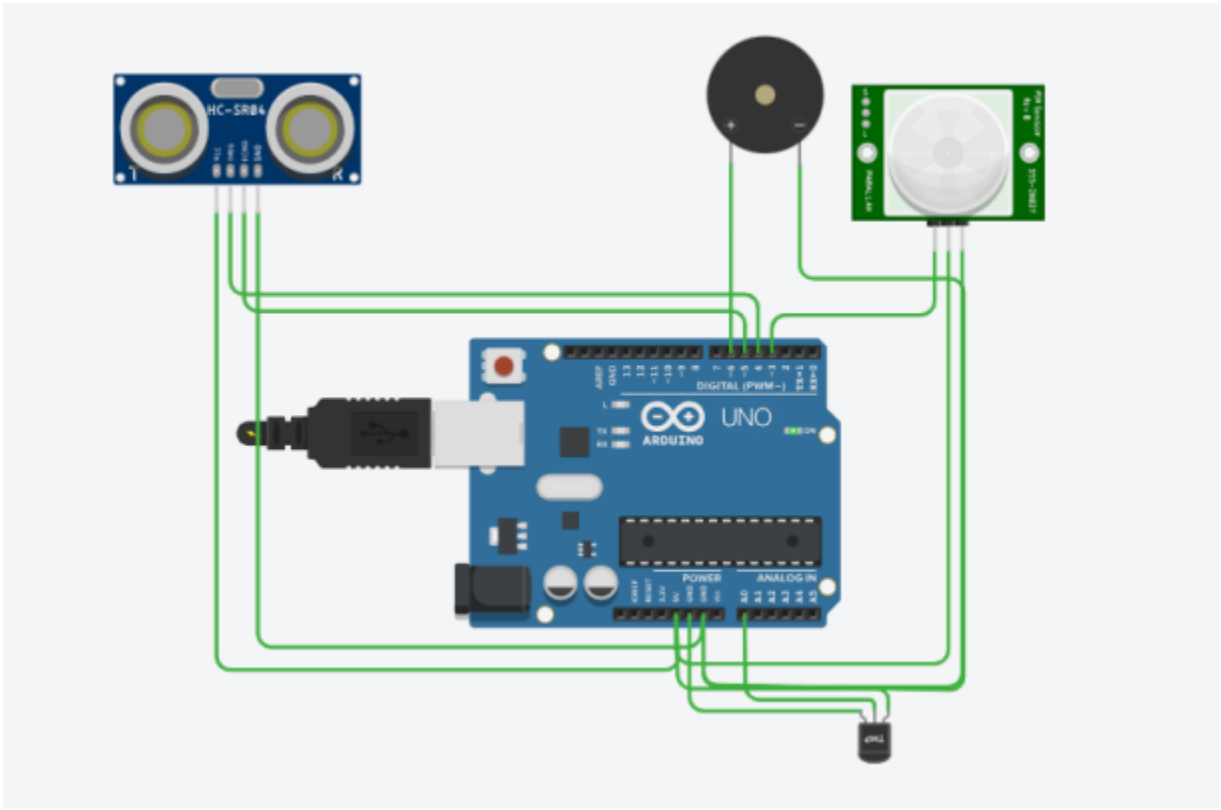
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define TEMP_PIN A0 // Analog pin for the TMP36 temperature sensor
#define PIR_PIN 3 // Pin for the PIR sensor
#define TRIGGER_PIN 4 // Pin for the ultrasonic sensor trigger
#define ECHO_PIN 5 // Pin for the ultrasonic sensor echo
#define BUZZER_PIN 6 // Pin for the buzzer
int ledPin = 13; // Built-in LED on Arduino
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD I2C address (use the correct address for your module)
void setup() {
Serial.begin(9600);
lcd.init(); // Initialize the LCD
lcd.backlight(); // Turn on the backlight
lcd.clear(); // Clear the LCD screen
lcd.setCursor(0, 0); // Set the cursor to the top-left corner
lcd.print("Monitoring System");

pinMode(PIR_PIN, INPUT);
pinMode(TRIGGER_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer pin as an OUTPUT
pinMode(ledPin, OUTPUT);
}
void loop() {
int motionDetected = digitalRead(PIR_PIN);
float temperatureC = readTemperature(); // Read temperature from TMP36 sensor
float distance = measureDistance();
```

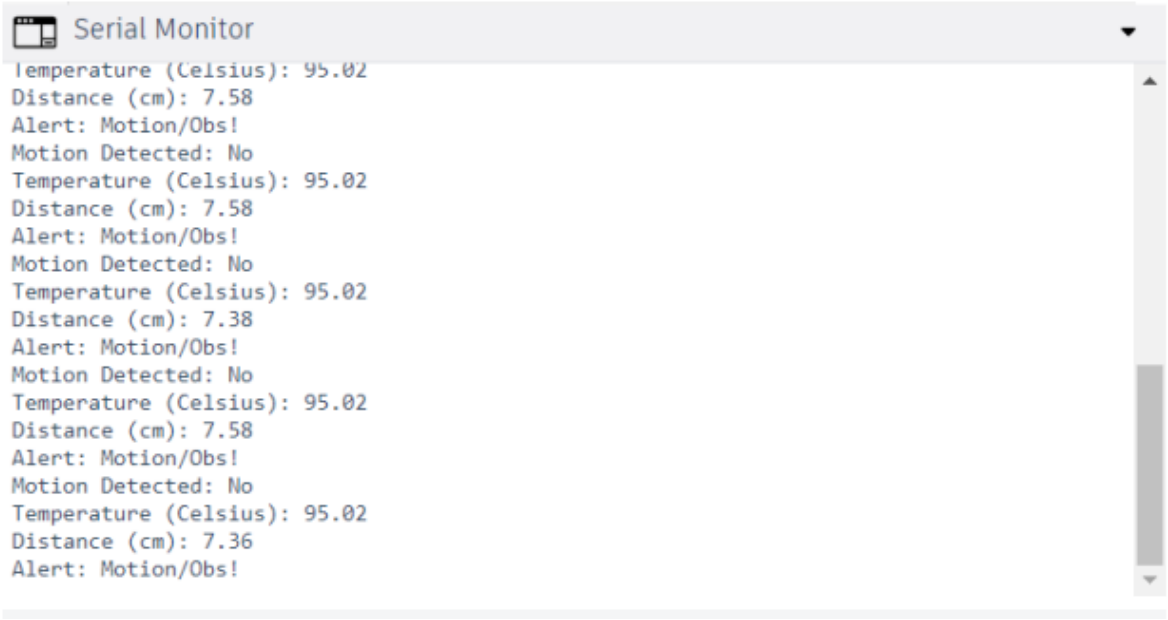
```
Serial.print("Motion Detected: ");
Serial.println(motionDetected ? "Yes" : "No");
Serial.print("Temperature (Celsius): ");
Serial.println(temperatureC);
Serial.print("Distance (cm): ");
Serial.println(distance);
lcd.setCursor(0, 1); // Set cursor to the second row
lcd.print("Motion: ");
lcd.print(motionDetected ? "Yes" : "No");
lcd.print(" Temp: ");
lcd.print(temperatureC);
lcd.print("C");
if (motionDetected || distance < 30) { // Adjust the distance threshold as needed
Serial.println("Alert: Motion/Obs!");
lcd.setCursor(0, 0); // Set cursor to the first row
lcd.print("Alert: Motion/Obs!");
digitalWrite(BUZZER_PIN, HIGH); // Turn on the buzzer
digitalWrite(ledPin, HIGH); // Turn on the built-in LED
delay(500); // Buzz and light for half a second
digitalWrite(BUZZER_PIN, LOW); // Turn off the buzzer
digitalWrite(ledPin, LOW); // Turn off the built-in LED
delay(500); // Delay to prevent rapid alerts
} else {
lcd.setCursor(0, 0); // Set cursor to the first row
lcd.print(" "); // Clear the previous message
}

delay(5000); // Read data every 1 second
}
float measureDistance() {
digitalWrite(TRIGGER_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIGGER_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN, LOW);
long duration = pulseIn(ECHO_PIN, HIGH);
// Calculate distance in centimeters
float distance = duration * 0.034 / 2;
return distance;
}
float readTemperature() {
int sensorValue = analogRead(TEMP_PIN);
float voltage = (sensorValue / 1024.0) * 5.0; // Convert to voltage (assuming 5V
Arduino)
```

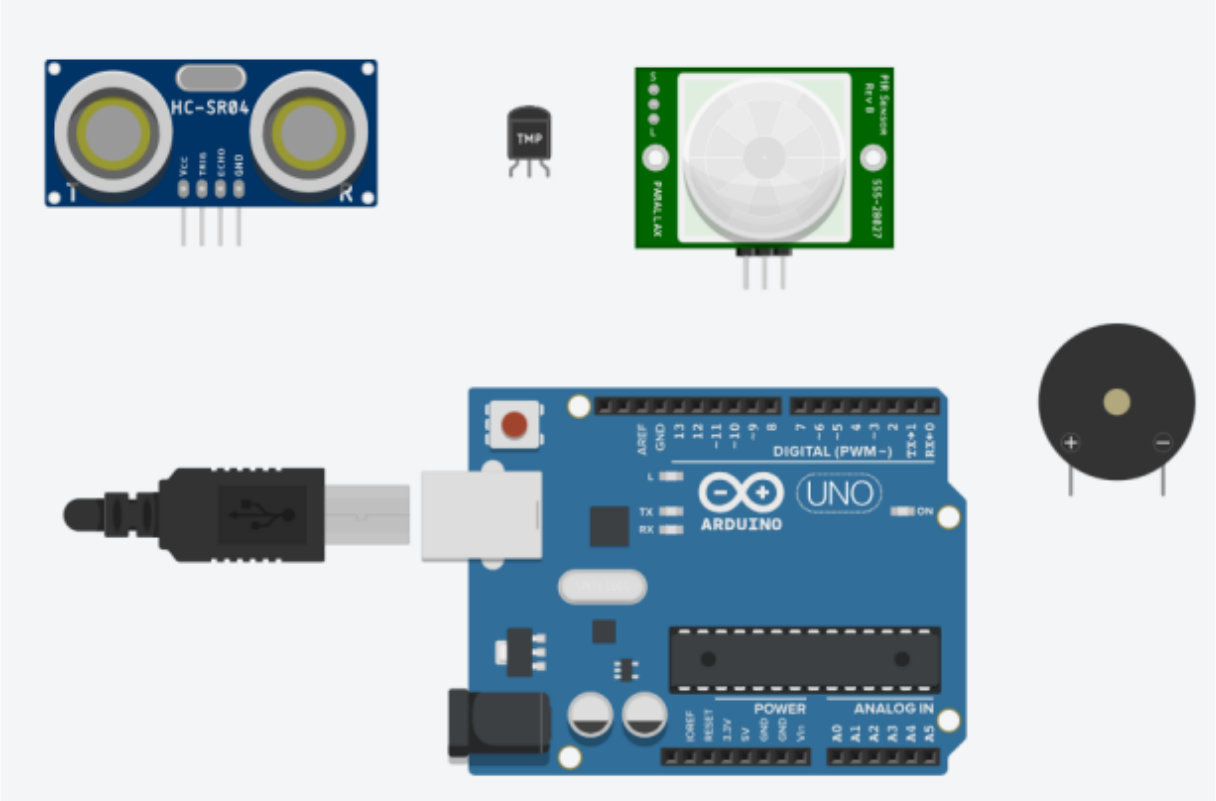
```
float temperatureC = (voltage - 0.5) * 100.0; // TMP36 scale factor
return temperatureC;
}
```

**Results :-**






**Conclusion:**

We explored the IoT simulation tools, namely Wowki and TinkerCAD, and revealed their unique advantages and disadvantages. Both tools have their place in IoT development, depending on thespecific project requirements and budget constraints.