JCEI's

# JAIHIND COLLEGE OF ENGINEERING, KURAN

Department Of

Artificial Intelligence and Data Science

LAB MANUAL

Industrial Iternet Of Things(IIOT), BE

Semester I

Prepared by:

Prof.Auti M.A.

# Computer Laboratory – II

| Course Code | Course Name | Teaching Scheme(Hrs./Week) | Credits |
|---|---|---|---|
| 417526 | Computer Laboratory II – Industrial Internet Of Things[IIOT] | 4 | 2 |

**Course Objectives:**

- To explore the needs and fundamental concepts of IIoT .
- To elucidate the roles of sensors and protocols in IIoT.
- To design and assemble IIOT system for various applications.

**Course Outcomes:**

On completion of the course, learners will be able to–

**CO1:** Understand IIoT technologies, architectures, standards, and regulation.

**CO2:** Build IIOT systems that include hardware and software and be exposed to modern and exciting hardware prototyping platforms.

**CO3:** Develop real applications and improve them through smart technologies.

**Instructions:**

1. Practical work can be performed on a suitable development platform (Arduino/ Raspberrypi).

2. Perform total 5 experiments.

# Table Of Contents

| | |
|---|---|
| **Lab Assignment No.** | 01 |
| **Title** | Write a program for sending alert messages to the user for controlling and interacting with your environment. |
| **Roll No.** | |
| **Class** | BE AI & DS |
| **Date Of Completion** | |
| **Subject** | Computer Laboratory II[417526] |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# Assignment No. 01

**Title -** Write a program for sending alert messages to the user for controlling and interacting with your environment.

**Problem Statement -** Design a program that simulates sending alert messages to the user based on environmental conditions in Tinkercad.

**Prerequisite –** C/C++ programming, basic understanding of conditionals, loops, and Tinkercad environment.

**Software Requirements -** Tinkercad simulation platform, internet connection, computer or compatible device.

**Hardware Requirements –** Arduino Uno R3, Teamperature Sensor, Jumpre wires, Breadboard Small, Piezo.

**Learning Objectives –** Learn to Environmental monitoring, conditional logic, event handling, simulated sensor data, alert notifications.

**Outcomes –** After Completion of this assignment students are able to understand how to create Effective alert system, understanding of sensor data simulation, and programming proficiency.

**Theory** - Designing a program to simulate sending temperature alert messages to the user based on environmental conditions in Tinkercad involves integrating several theoretical concepts related to sensor interfacing, data acquisition, condition monitoring, and user interaction. Let's delve into each of these aspects in detail:

**1. Sensor Interfacing:**

In Tinkercad, sensor interfacing typically involves connecting a virtual sensor to a microcontroller (like Arduino Uno). The sensor used for temperature measurement could be simulated using an analog input pin on the Arduino.

**-TMP Sensor Simulation:** The TMP sensor (e.g., TMP36) simulates temperature measurement by outputting a voltage proportional to the temperature. For instance, it might output 0.5V at 0°C and 2.5V at 100°C when supplied with a 5V reference.

**-Analog-to-Digital Conversion (ADC):** The Arduino Uno reads analog voltages using its built-in ADC. The analogRead() function converts the analog voltage from the sensor into a digital value (0-1023 for a 10-bit ADC).

**2. Data Acquisition:**

**-Virtual Sensor Data Simulation:** In Tinkercad, sensor data is simulated rather than real. The sensor (TMP36) might output voltages that correspond to various temperature values within its simulated range.

**-Data Representation:** Temperature data obtained from the sensor is represented as a floating-point value in Celsius. This representation is crucial for further processing and comparison with predefined thresholds.

**3. Condition Monitoring:**

**-Thresholds:** Threshold values are predefined limits that trigger alerts when exceeded. For temperature monitoring, a threshold might be set to 23°C. When the measured temperature exceeds this threshold, an alert message is triggered.

**-Conditional Statements:** In the Arduino program, conditional statements (if statements) are used to compare the current temperature reading with the predefined threshold. If the temperature exceeds the threshold, an alert is generated.

**4. User Interaction:**

**-Alert Notification**: When the temperature exceeds the predefined threshold, the program simulates alerting the user. In Tinkercad, this might involve displaying a message in the serial monitor or activating a virtual LED to indicate an alert condition.

**-Serial Communication:** The Arduino communicates with the Tinkercad interface through serial communication. This allows for real-time monitoring of sensor readings and alerts in the simulated environment.

**Components Overview:**

**1.Arduino Uno:** The Arduino Uno is a popular microcontroller board based on the ATmega328P chip. It provides an easy-to-use platform for interfacing with sensors and controlling external devices.

**2.TMP Sensor:** The TMP sensor (TMP36, TMP102, etc.) is a temperature sensor that can accurately measure temperature in degrees Celsius. It typically outputs an analog voltage proportional to the temperature.

**3.Jump Wires:** These wires are used to establish electrical connections between components on the breadboard and Arduino Uno.

**4.Breadboard:** A breadboard is a solderless prototyping board used for building and testing electronic circuits. It allows components to be easily connected and rearranged.

**5.Piezo Buzzer:** A piezo buzzer is an electronic device that generates sound when an alternating voltage is applied. It will be used to notify the user when the temperature exceeds a certain threshold.

**Application:**

**1.Prototype Development:** Tinkercad allows developers to prototype and test circuits virtually before physical implementation. This reduces costs and time associated with hardware iterations.

**2.Sensor Calibration:** Understanding sensor characteristics (such as voltage-temperature relationship for TMP sensors) is crucial for accurate data acquisition and threshold setting.

**3.Event-Driven Programming:** The program's logic is event-driven; it continuously monitors sensor data and responds to predefined conditions (temperature exceeding threshold) by activating the buzzer.

**4.Serial Communication:** Serial communication (Serial.begin() and Serial.print()) facilitates debugging and monitoring sensor readings in real-time through the Arduino IDE's serial monitor.

**5.Educational Use:** This project is ideal for educational purposes, teaching students about sensor interfacing, analog-to-digital conversion, conditional statements, and using external devices (like buzzers) for notifications.

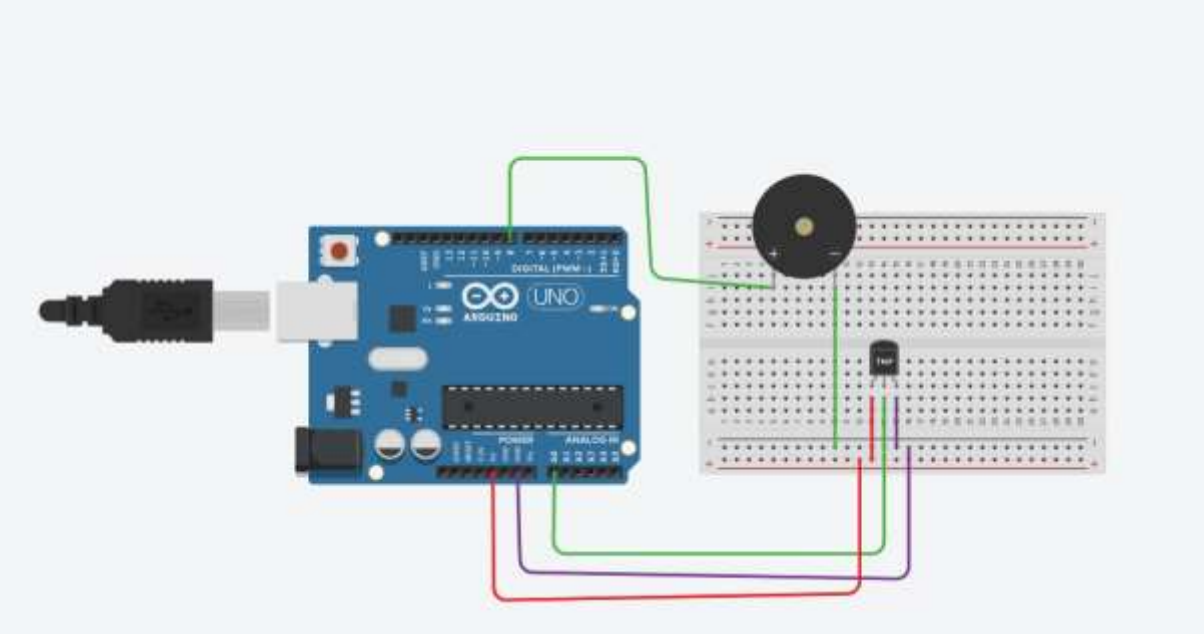
**Source Code –**

```
#define TEMP_PIN A0         // Pin where the TMP36 sensor is connected

#define BUZZER_PIN 8        // Buzzer pin
```

```
const float TEMPERATURE_THRESHOLD = 23.0; // Temperature threshold in Celsius
void setup() {
  // Initialize the buzzer pin as an output
  pinMode(BUZZER_PIN, OUTPUT);
  // Start the Serial Monitor for debugging
  Serial.begin(9600);
}
void loop() {
  // Read the temperature from the TMP36 sensor
  int tempReading = analogRead(TEMP_PIN);
  float voltage = tempReading * (5.0 / 1023.0);
  float temperatureC = (voltage - 0.5) * 100.0;
  // Print the temperature to the Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" C");
   // Check if the temperature exceeds the threshold
  if (temperatureC > TEMPERATURE_THRESHOLD) {
    // Turn on the buzzer
    digitalWrite(BUZZER_PIN, HIGH);
     // Print an alert message to the Serial Monitor
    Serial.println("ALERT: Temperature is too high!");
  } else {
    // Turn off the buzzer
```

```
    digitalWrite(BUZZER_PIN, LOW);
  }
  // Wait for a short period before the next loop
  delay(500);
}
```

**Circuit Diagram –**



**Conclusion –**

Designing a program in Tinkercad to simulate temperature alerts fosters understanding of sensor integration, data analysis, and user notification systems. This project enhances skills in IoT development, preparing users to create responsive and effective environmental monitoring solutions applicable across various industries and scenarios.

| | |
|---|---|
| **Lab Assignment No.** | 02 |
| **Title** | Write an Arduino program for interfacing with PIR sensor Experiment . |
| **Roll No.** | |
| **Class** | BE AI & DS |
| **Date Of Completion** | |
| **Subject** | Computer Laboratory II[417526] |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# Assignment No.2

**Title -** Write an Arduino/ Raspberry pi program for interfacing with PIR sensor Experiment.

**Problem Statement -** Design an Arduino program to interface with a PIR (Passive Infrared) sensor on Tinkercad. Simulate the detection of motion using the PIR sensor and visualize the results through serial communication or a virtual LED.

**Prerequisite –** C/C++ programming, basic understanding of conditionals, loops, and Tinkercad environment.

**Software Requirements -** Tinkercad simulation platform, internet connection, computer or compatible device.

**Hardware Requirements –** Arduino UNO R3, Breadboard Small, Jumpre Wires, Resistor, PIR Sensor.

**Learning Objectives –** Learn to Sensor interfacing, event detection, visualization, Arduino programming, Tinkercad simulation.

**Outcomes -** After Completion of this assignment students are able to Motion detection simulation, data visualization, Arduino proficiency, IoT understanding.

**Theory** - The Passive Infrared (PIR) sensor is a fundamental component used for motion detection by measuring infrared radiation changes. It is widely utilized in applications ranging from security systems to automated lighting. In this experiment, we will interface a PIR sensor with an Arduino microcontroller to detect motion and provide a visual indication using an LED. The setup will also involve a resistor to ensure proper current flow through the LED. This experiment demonstrates the practical integration of these components and the basics of digital input-output operations with the Arduino.

**Working Principle of PIR Sensor**

The PIR sensor operates based on the principle of detecting infrared radiation. All objects emit infrared radiation as a function of their temperature. The PIR sensor includes a pyroelectric material that generates a voltage when exposed to infrared radiation. This voltage change is processed internally and converted into a digital signal.

**-Detection Range and Angle:** PIR sensors have a specified field of view and detection range, usually determined by the lens and design of the sensor. The typical range is between 3 to 12 meters, and the field of view is approximately 120 degrees.

**-Signal Output:** The sensor outputs a HIGH signal (1) when motion is detected and a LOW signal (0) when no motion is detected.

**Components Overview**

**1.PIR Sensor:** A PIR sensor detects motion by sensing changes in infrared radiation. It typically has three pins:

-VCC: Power supply (5V).

-GND: Ground connection.

-OUT: Digital output indicating motion detection (HIGH or LOW).

**2.Arduino Microcontroller:** An Arduino board will read the digital signal from the PIR sensor and control the LED based on the sensor's output.

**3.LED:** A Light Emitting Diode (LED) will visually indicate motion detection.

**4.Resistor:** A resistor is used to limit the current flowing through the LED to prevent damage. For a standard LED, a 220-ohm resistor is typically used.

**5.Breadboard:** A breadboard is used to build the circuit by connecting the components without soldering.

**Circuit Design and Wiring**

**1.Wiring the PIR Sensor:**

Connect the VCC pin of the PIR sensor to the 5V pin on the Arduino.

Connect the GND pin of the PIR sensor to the GND pin on the Arduino.

Connect the OUT pin of the PIR sensor to a digital input pin on the Arduino (e.g., pin 2).

**2.Connecting the LED:**

Connect the anode (longer leg) of the LED to a digital output pin on the Arduino (e.g., pin 13).

Connect the cathode (shorter leg) of the LED to one end of a 220-ohm resistor.

Connect the other end of the resistor to GND on the Arduino.

**3.Breadboard Layout:**

Use the breadboard to make connections more organized and ensure stable connections.

Insert the PIR sensor, LED, and resistor into the breadboard and make connections according to the wiring instructions above.

**Testing and Observations**

**1.Build the Circuit:** Assemble the circuit on a breadboard following the wiring instructions.

**2.Upload and Run the Code:** Upload the provided code to the Arduino using the Arduino IDE and start the program.

**3.Observe the LED and Serial Monitor:**

When motion is detected within the PIR sensor's field of view, the LED should light up, and the serial monitor should display "Motion detected!"

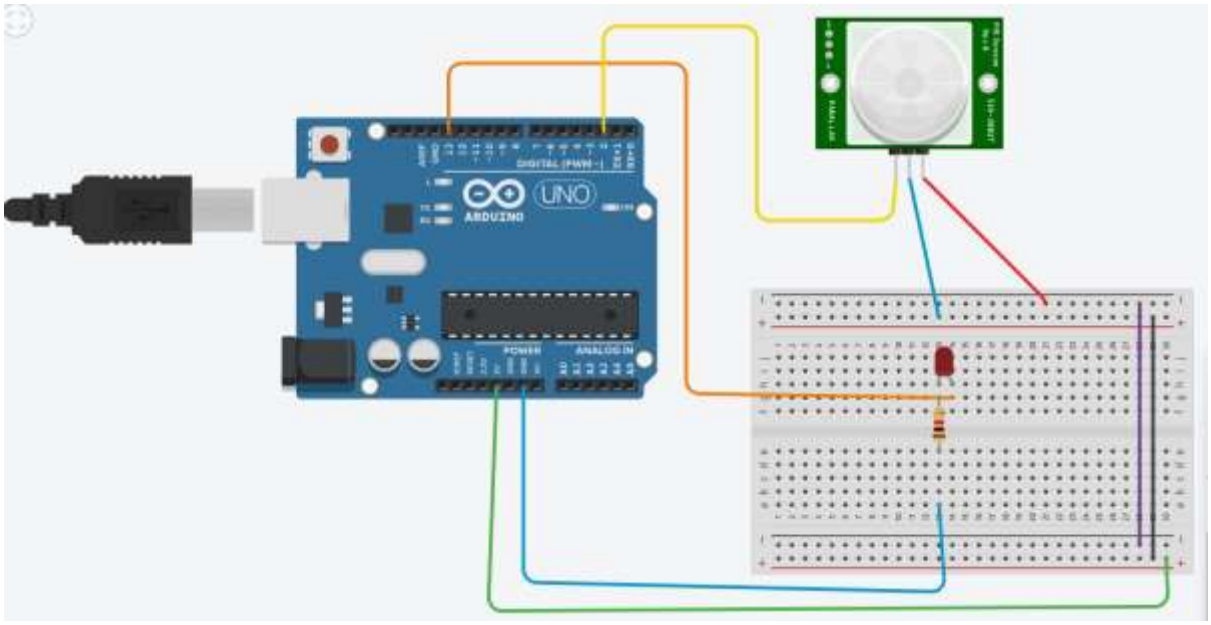
When no motion is detected, the LED will turn off, and the serial monitor will display "No motion detected."

**Source Code –**

```
// Define pin numbers
const int pirPin = 2;    // PIR sensor input pin
const int ledPin = 13;   // LED output pin (built-in on many Arduino boards)
void setup() {
  pinMode(pirPin, INPUT);    // Set PIR pin as input
  pinMode(ledPin, OUTPUT);   // Set LED pin as output
Serial.begin(9600);        // Initialize serial communication for debugging
}
void loop() {
  int pirState = digitalRead(pirPin);  // Read PIR sensor state
```

```
 if (pirState == HIGH) {            // If motion is detected
   digitalWrite(ledPin, HIGH);      // Turn on LED
   Serial.println("Motion detected!");
 } else {
   digitalWrite(ledPin, LOW);       // Turn off LED
   Serial.println("No motion");
 }
delay(1000);  // Wait for a second before rechecking
}
```

**Circuit Diagram –**




**Conclusion –**

This experiment demonstrates the integration of a PIR sensor with an Arduino to create a basic motion detection system. The use of a PIR sensor, LED, resistor, and Arduino provides a comprehensive understanding of digital input-output operations and practical circuit design.

| | |
|---|---|
| **Lab Assignment No.** | 03 |
| **Title** | Write a program for developing an IIoT application for energy monitoring and optimization. |
| **Roll No.** | |
| **Class** | BE AI & DS |
| **Date Of Completion** | |
| **Subject** | Computer Laboratory II[417526] |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# Assignment No. 03

**Title -** Write a program for developing an IIoT application for energy monitoring and optimization.

**Problem Statement –** Design and Development of an IIoT Application for Smart Home Energy Monitoring and Optimization.

**Prerequisite –** C/C++ programming, basic understanding of conditionals, loops, and Tinkercad environment.

**Software Requirements -** Tinkercad simulation platform, internet connection, computer or compatible device.

**Hardware Requirements –** Arduino Uno R3, Breadboard small, Light Bulb, TMP Sensor, Jumpre Wires, Resistor, Dc Motor, Power Supply, and Relay.

**Learning Objectives –** 1.Understand IoT fundamentals and device integration.

2.Implement real-time data acquisition and visualization techniques.

3. Design user interfaces for effective interaction and feedback.

**Outcomes -** After Completion of this assignment students are able to Real-time energy monitoring, User-friendly interface, Smart device integration, and Secure data handling should be done.

**Theory -** The rapid advancement of technology has significantly impacted how we manage and optimize energy consumption in modern homes. The Industrial Internet of Things (IIoT) leverages interconnected devices and sensors to monitor, control, and optimize energy usage in real-time. This experiment focuses on developing a prototype IIoT application using Arduino, a light bulb, a power supply, a relay, a temperature sensor (TMP), a DC motor, and resistors. The goal is to create a system that monitors energy consumption, controls electrical devices, and optimizes their operation to improve efficiency.

**Components and Their Functions**

**1.Arduino Microcontroller:** The Arduino microcontroller is the central unit of the system. It processes input from sensors and controls outputs to devices like relays and motors. Arduino is chosen for its simplicity, versatility, and ease of integration with various sensors and actuators.

**2.Light Bulb:** The light bulb represents a typical home appliance. Its power consumption will be monitored and controlled to demonstrate energy management and optimization.

**3.Power Supply:** The power supply provides the necessary electrical power for the light bulb, Arduino, and other components. It ensures that the system operates within safe voltage and current limits.

**3.Relay:** The relay acts as a switch that controls the light bulb. It allows the Arduino to turn the light bulb on or off based on sensor readings and optimization algorithms.

**4.Temperature Sensor (TMP):** The TMP sensor measures the ambient temperature. Temperature data is crucial for understanding the thermal impact on energy consumption and for implementing optimization strategies.

**5.DC Motor:** The DC motor can simulate additional load or appliance. It represents another common household device whose power usage can be monitored and optimized.

**6.Resistor:** Resistors are used to control current flow and ensure safe operation of the components. They help in setting appropriate voltage levels for sensors and the Arduino.

**System Design and Operation**

The system design includes several key components and their interactions. Below is an overview of the design, operation, and implementation.

**1.Data Acquisition:**

The Arduino collects data from the temperature sensor to monitor the ambient temperature. It also measures power consumption indirectly through the relay and light bulb setup. By integrating a current sensor or power meter (which could be simulated by a resistor in this experiment), the Arduino can gauge the energy usage of the light bulb and DC motor.

**2.Control Mechanism:**

The Arduino controls the light bulb and DC motor via the relay. The relay acts as an intermediary switch that the Arduino can activate to turn the devices on or off. The relay's state is managed based on the data received from the temperature sensor and any optimization algorithms applied.

**3.Optimization Algorithms:**

The optimization algorithm can be designed to adjust the operation of the light bulb and DC motor based on real-time data. For instance, if the temperature is high, the algorithm might turn off the light bulb to reduce heat generation, thereby optimizing energy use. Alternatively, it could adjust the DC motor's operation to maintain efficient energy consumption.

**4.User Interface:**

Although the prototype might not include a graphical user interface, the Arduino can be programmed to output data to a serial monitor for real-time observation. For a complete system, a user interface could be developed to visualize data and control devices remotely.

**5.Energy Monitoring and Reporting:**

Energy monitoring involves tracking the power consumption of connected devices. The Arduino processes sensor data and relay states to compute energy usage. This information can be reported to users or logged for further analysis. In this setup, energy usage can be approximated based on the operation time and power rating of the light bulb and DC motor.

**6.Safety Considerations:**

Safety is crucial when working with electrical components. The power supply should be correctly rated for the components to prevent overloading. Relays should be selected based on their voltage and current ratings to handle the load of the light bulb and DC motor safely. Resistors must be used to limit current and protect sensitive components like sensors and the Arduino.

**Implementation Steps**

**1.Component Setup:**

Connect the power supply to the light bulb, relay, and DC motor, ensuring all connections are secure and correctly rated.

Attach the temperature sensor to the Arduino using appropriate pins and connections.

Integrate the relay with the Arduino to control the light bulb and motor.

Use resistors where necessary to protect components and ensure correct voltage levels.

**2.Programming the Arduino:**

Write the Arduino code to read temperature data from the TMP sensor.

Develop logic to control the relay based on temperature readings and optimization criteria.

Implement algorithms to monitor energy usage and adjust device operation accordingly.

Use serial communication to output data for real-time monitoring and analysis.

**3.Testing and Calibration:**

Test each component individually to ensure proper operation.

Calibrate the temperature sensor and verify that the relay controls the light bulb and motor accurately.

Validate the energy monitoring functionality and optimization algorithms.

**4.Analysis and Optimization:**

Analyze the data collected during tests to evaluate the performance of the optimization algorithms.

Adjust the algorithms as needed to improve energy efficiency and system response.

**5.Documentation:**

Document the system design, programming details, and test results.

Prepare a report on the energy monitoring and optimization performance, including any observed improvements in efficiency.

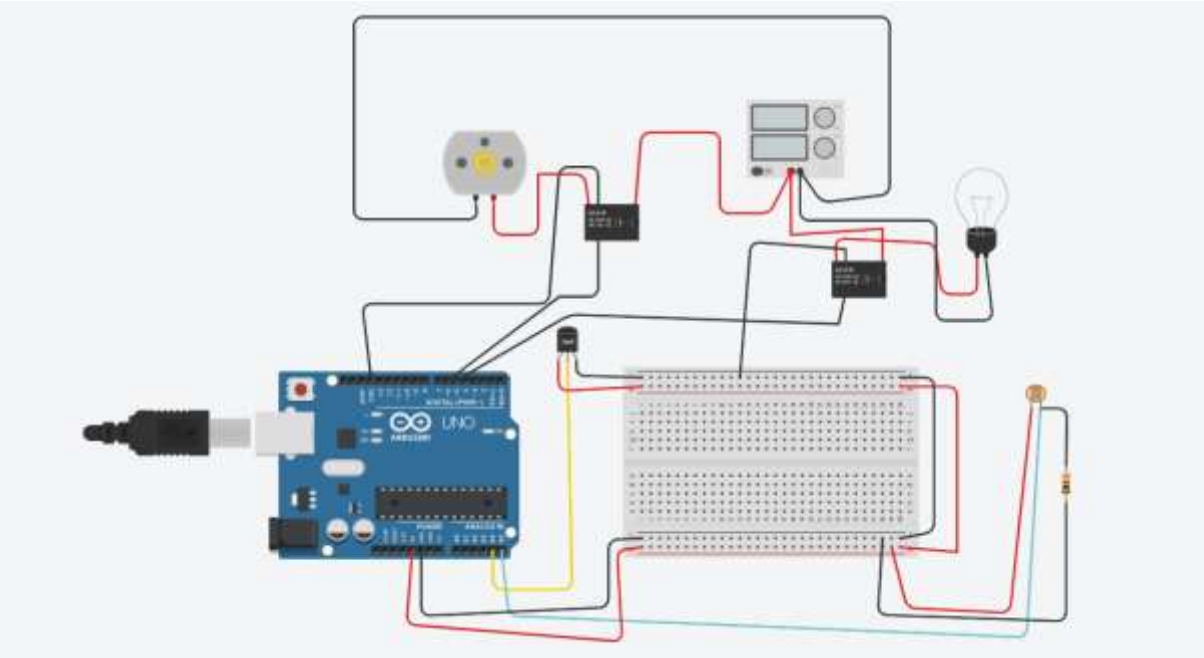
**Source Code –**

```
float x,y,z,temp;

void setup()

{

 // pinMode(8, INPUT);

  pinMode(5, OUTPUT);

  pinMode(6, OUTPUT);
```

```
  pinMode(A5, INPUT);
  pinMode(A4, INPUT);
Serial.begin(9600);
}
void loop()
{
 // x= digitalRead(8);
  y= analogRead(A5);
  z= analogRead(A4);
  Serial.println(x);
  Serial.println(y);
  Serial.println(z);
  temp = (double)z / 1024;
  temp = temp * 5;
  temp = temp - 0.5;
  temp = temp * 100;
  //if ( (x>0) )
  //{
    if ((y<550)&&(temp>30))
    {
      digitalWrite(5, HIGH);
      digitalWrite(6, HIGH);
    }
    else if((y<550)&&(temp<30))
```

```
    {
      digitalWrite(5, HIGH);
      digitalWrite(6, LOW);
    }
    else if((y>550)&&(temp>30))
    {
      digitalWrite(5, LOW);
      digitalWrite(6, HIGH);
    }
    else if((y>550)&&(temp<30))
    {
      digitalWrite(5, LOW);
      digitalWrite(6, LOW);
    }
  /*}
  else
  {
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
  }*/
}
```

**Circuit Diagram -**



**Conclusion -** This experiment provides a practical demonstration of IIoT applications for smart home energy management. By using Arduino and various components, the system enables real-time monitoring and optimization of energy consumption.

| | |
|---|---|
| **Lab Assignment No.** | 04 |
| **Title** | Write a program for implementing security measures in an IIoT system. |
| **Roll No.** | |
| **Class** | BE AI & DS |
| **Date Of Completion** | |
| **Subject** | Computer Laboratory II[417526] |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# Assignment No. 04

**Title -** Write a program for implementing security measures in an IIoT system.

**Problem Statement -** Design and develop security measures for an IIoT system on Tinkercad, including authentication, authorization, encryption, data integrity, and monitoring.

**Prerequisite –** C/C++ programming, basic understanding of conditionals, loops, and Tinkercad environment.

**Software Requirements -** Tinkercad simulation platform, internet connection, computer or compatible device.

**Hardware Requirements -** Arduino Uno R3, Resistor, LED, Breadboard Small, Gas sensor, Potentiometer, Piezo, TMP Sensor, Jumper Wires, and LCD 16/2.

**Learning Objectives -** Implement authentication and authorization protocols.

Develop data encryption techniques.

Design data integrity verification methods.

Create audit and monitoring systems.

**Outcomes -** After Completion of this assignment students are able to Secure authentication Role-based authorization, Data encryption, Integrity verification, and Activity monitoring.

**Theory -** The Industrial Internet of Things (IIoT) represents a significant advancement in connecting and managing a wide range of devices and sensors to enhance operational efficiency and data collection. However, with these advancements come substantial security concerns, as interconnected systems are vulnerable to unauthorized access, data breaches, and other cyber threats. Implementing robust security measures is essential for protecting these systems. This experiment demonstrates how to design and implement security measures in an IIoT system using Arduino, breadboard, various sensors (gas and temperature), resistors, LEDs, LCDs, a potentiometer, a piezo buzzer, and jumper wires, all within the Tinkercad simulation environment.

**Components and Their Functions**

**1.Arduino Microcontroller:**

**Function:** Acts as the central processing unit of the IIoT system. It handles data from sensors, processes security measures, and controls outputs based on programmed security protocols.

**2.Breadboard:**

**Function:** Provides a platform for prototyping the circuit without soldering. It allows for easy connections and modifications of the components.

**3.Gas Sensor:**

**Function:** Monitors the presence of gases in the environment. This sensor can be used to detect potential hazards or unauthorized presence based on gas levels, providing an additional layer of security.

**4.Temperature Sensor (TMP):**

**Function:** Measures the ambient temperature. In security contexts, it can detect environmental changes that might indicate tampering or system failure.

**5.Resistors:**

**Function:** Control the flow of current through various components, protecting sensitive parts from damage and ensuring proper operation.

**6.LEDs:**

**Function:** Provide visual feedback for various system statuses, such as authentication success, alert signals, or system errors.

**7.LCD:**

**Function:** Displays real-time information, such as system status, alerts, or authentication messages. It helps in monitoring and interacting with the system.

**8.Potentiometer:**

**Function:** Acts as a variable resistor that can be adjusted to simulate different conditions or settings, such as sensitivity levels for sensors.

**9.Piezo Buzzer:**

**Function:** Generates sound alerts for notifications or warnings, such as unauthorized access attempts or system errors.

**10.Jumper Wires:**

**Function:** Connect various components on the breadboard and to the Arduino, ensuring proper circuit connectivity.

**System Design and Security Measures**

**1.Authentication and Authorization:**

**Implementation:** Authentication can be implemented using a simple password mechanism or code. The Arduino can be programmed to check a predefined password input against a stored value.

**Components Used:** The LCD displays prompts and status messages, while LEDs indicate authentication success or failure.

**2.Data Encryption:**

**Implementation:** Although full encryption may be complex, a simplified encryption scheme can be used to demonstrate basic principles. For example, XOR-based encryption can be applied to sensor data before transmission.

**Components Used:** Arduino handles the encryption/decryption process, ensuring that sensor data remains secure.

**3.Data Integrity Verification:**

**Implementation:** Use checksums or simple hash functions to verify the integrity of transmitted data. This can ensure that data has not been altered during transmission.

**Components Used:** The Arduino processes and verifies data from sensors, comparing checksums before taking action.

**4.Audit and Monitoring:**

**Implementation:** Logs system activity, such as sensor readings and authentication attempts, to provide a record of events. This helps in identifying and investigating potential security issues.

**Components Used:** The LCD and LEDs display real-time monitoring data, while the piezo buzzer alerts users to critical events.

**5.Alert Mechanisms:**

**Implementation:** Set up visual and auditory alerts to notify users of security breaches, system failures, or unauthorized access attempts.

**Components Used:** LEDs and the piezo buzzer provide immediate feedback and alerts.

**Circuit Design and Implementation**

**1.Building the Circuit:**

**Breadboard Setup:** Arrange the Arduino, sensors, and other components on the breadboard. Connect the gas sensor, temperature sensor, and potentiometer to the Arduino using jumper wires.

**Wiring:** Connect the sensors' output pins to Arduino analog or digital input pins. Connect the LCD, LEDs, and piezo buzzer to appropriate output pins on the Arduino.

**Power Supply:** Ensure that the power supply to the Arduino and components is stable and within the required voltage range.

**2.Programming the Arduino:**

**Authentication Code:** Write code to handle user authentication, checking input values against predefined credentials and updating the LCD and LED status accordingly.

**Encryption and Decryption:** Implement a simple encryption algorithm for sensor data, ensuring that data integrity is maintained.

**Data Integrity:** Write functions to calculate and verify checksums or hashes for transmitted data.

**Logging and Alerts:** Program the Arduino to log system activities and provide alerts via the LCD and piezo buzzer.

**3.Testing and Calibration:**

**Component Testing:** Test each component individually to ensure proper operation. Check sensor readings, authentication processes, and alert mechanisms.

**Calibration:** Adjust the potentiometer to calibrate sensor sensitivity levels. Verify that data encryption and integrity verification are functioning correctly.

**4.Security Analysis:**

**Evaluate Security Measures:** Assess the effectiveness of implemented security features. Identify any potential vulnerabilities or areas for improvement.

**Document Findings:** Prepare a report detailing the security measures, implementation process, test results, and any observed issues.

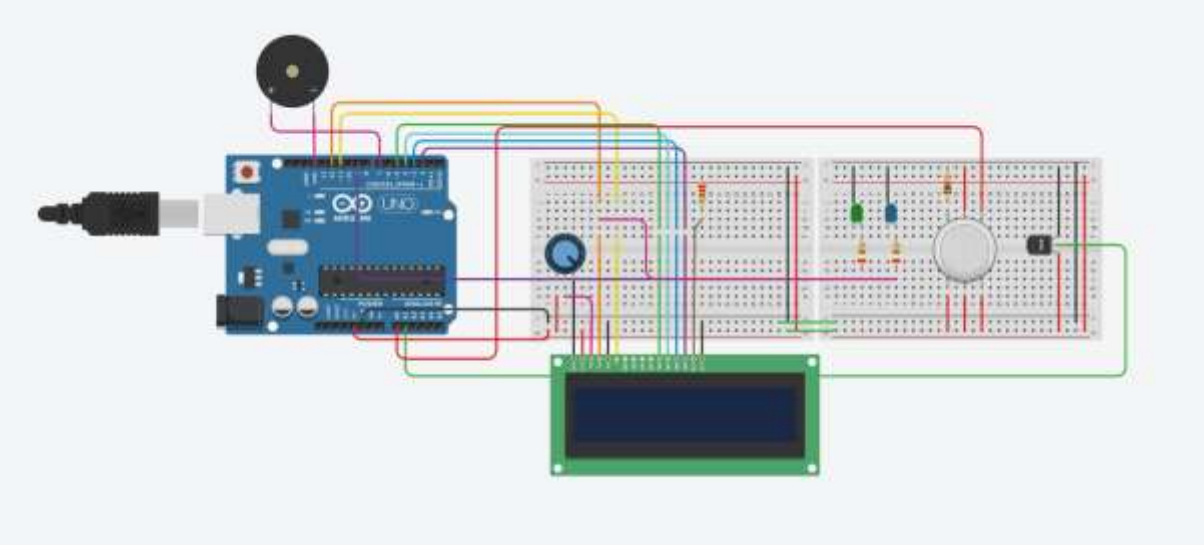
**Source Code –**

```
#include <LiquidCrystal.h>

// Initialize the LCD with the pin numbers

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

int V_GasSen = 0;

int V_TempSens = 0;

void setup() {

  pinMode(A0, INPUT); // Gas sensor pin

  pinMode(A1, INPUT); // Temperature sensor pin

  pinMode(7, OUTPUT); // Buzzer pin

  pinMode(9, OUTPUT); // LED for gas detection

  pinMode(12, OUTPUT); // LED for temperature warning

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

}
```

```
void loop() {
  // Read gas sensor value
  V_GasSen = analogRead(A0);
// Read temperature sensor value and calculate temperature
  V_TempSens = -40 + 0.488155 * (analogRead(A1) - 20);
// Display temperature and gas status on the LCD
  lcd.clear(); // Clear the LCD
  lcd.setCursor(0, 0); // Set cursor to the first row
  lcd.print("Temperature: "); // Print temperature label
  lcd.print(V_TempSens); // Print temperature value
  lcd.print(" C"); // Print temperature unit
 lcd.setCursor(0, 1); // Set cursor to the second row
  lcd.print("Gas: "); // Print gas label
  lcd.print(V_GasSen); // Print gas sensor value
 // Check for alerts
  if (V_GasSen >= 250) {
    tone(7, 523, 1000); // Play tone if gas is detected
    digitalWrite(9, HIGH); // Turn on the gas detection LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ALERT: Gas Detected");
  } else {
    digitalWrite(9, LOW); // Turn off the gas detection LED
  }
```

```
  if (V_TempSens >= 70) {
    tone(7, 523, 1000); // Play tone if temperature exceeds the threshold
    digitalWrite(12, HIGH); // Turn on the temperature warning LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ALERT: Temp High");
  } else {
    digitalWrite(12, LOW); // Turn off the temperature warning LED
  }
delay(1000); // Delay for one second
}
```

**Circuit Diagram –**



**Conclusion -** This experiment demonstrates the implementation of security measures in an IIoT system using Arduino and various components within Tinkercad. By integrating authentication, authorization, encryption, data integrity verification, and monitoring, the system provides a comprehensive approach to securing IIoT operations.

| | |
|---|---|
| **Lab Assignment No.** | 05 |
| **Title** | Write a program for performing industrial data analysis using relevant tools and techniques. |
| **Roll No.** | |
| **Class** | BE AI & DS |
| **Date Of Completion** | |
| **Subject** | Computer Laboratory II[417526] |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# Assignment No.05

**Title -** Write a program for performing industrial data analysis using relevant tools and techniques.

**Problem Statement -** Design a basic industrial data monitoring system using Arduino in Tinkercad.

**Prerequisite –** C/C++ programming, basic understanding of conditionals, loops, and Tinkercad environment.

**Software Requirements -** Tinkercad simulation platform, internet connection, computer or compatible device.

**Hardware Requirements** – Arduino Uno R3, TMP Sensor, Resistor, LCD Display 16 *2, and Jumper Wires.

**Learning Objectives** - Learn how to connect and interface temperature sensors with an Arduino. Gain experience in using an LCD to display real-time sensor data. and Develop skills in writing and debugging Arduino code for data collection and display.

**Outcomes -** After Completion of this assignment students are able to Integrate and read data from temperature sensors, show real-time data on an LCD and develop and debug Arduino code for data collection and display easily.

**Theory -** In industrial environments, monitoring environmental conditions like temperature is crucial for ensuring operational efficiency and safety. A basic industrial data monitoring system using Arduino can provide real-time temperature data, which is essential for various applications, such as preventing overheating of machinery or ensuring optimal storage conditions.

**Components Overview:**

**Arduino Uno:** The microcontroller used to process sensor data and control the display.

**TMP36 Temperature Sensor:** An analog sensor that provides a voltage proportional to the ambient temperature.

**Resistor:** Used to stabilize the sensor readings and protect the circuit.

**LCD Display (16x2):** A screen to display the temperature readings in a readable format.

**Jumper Wires:** To connect the components on the breadboard and Arduino.

**Data Acquisition and Processing:**

**Temperature Measurement:**

The TMP36 sensor measures the temperature by outputting a voltage that varies linearly with temperature. The sensor's output voltage is calibrated to provide a temperature reading where:

At 0°C, the output voltage is 500 mV.

For every degree Celsius increase, the output voltage increases by 10 mV.

The Arduino reads this analog voltage via an analog input pin and converts it to a digital value. This digital value is then processed to calculate the temperature in Celsius using the formula:

Temperature (°C)= (1024.0Analog Value × 5.0 × 100.0) / 1024.0 – 50

where the analog value is the raw data read from the sensor, and 1024.0 represents the ADC resolution of the Arduino (10-bit ADC).

**Data Display:**

The LCD display is connected to the Arduino via I2C communication. This interface requires only two data lines (SDA and SCL) for communication, making it convenient for displaying data with minimal wiring. The Arduino code controls the LCD to show the current temperature reading, updating the display periodically to provide real-time feedback.

**Practical Implications:**

Implementing this basic monitoring system allows for the real-time tracking of temperature, which is crucial in industrial settings for:

**Preventing Overheating:** Monitoring temperature can prevent equipment from reaching unsafe temperatures.

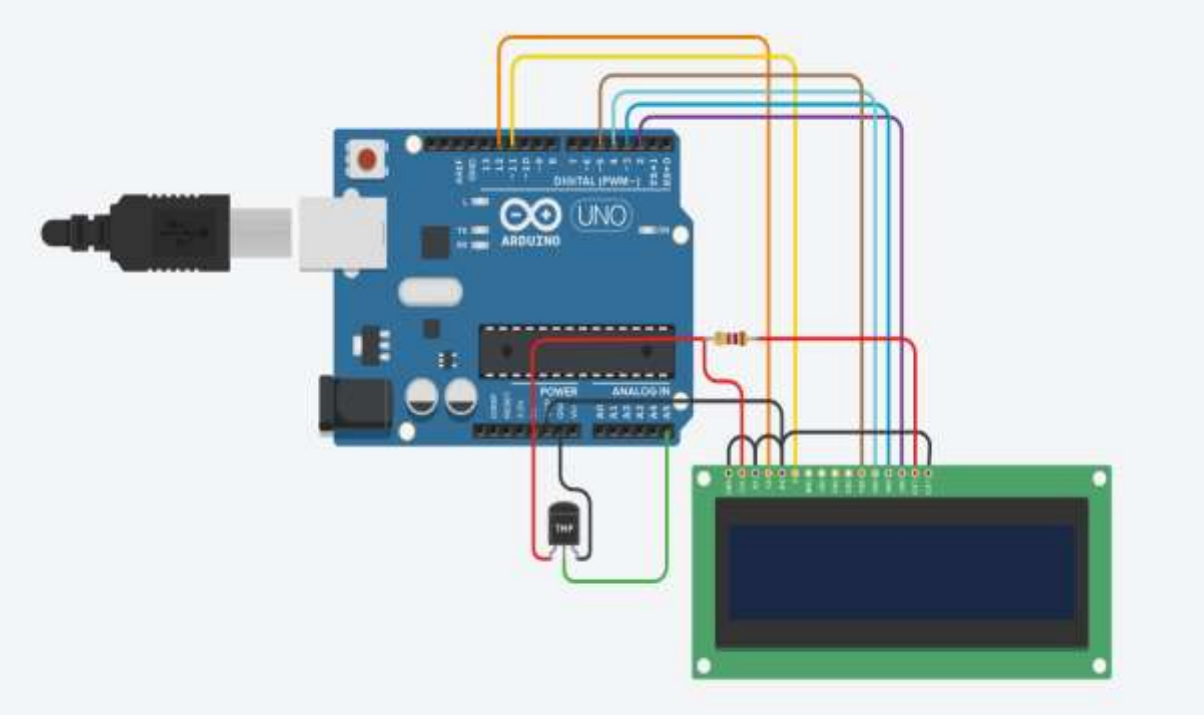
**Maintaining Optimal Conditions:** Ensuring machinery operates within specified temperature ranges to maintain efficiency and extend service life.

**Safety Compliance:** Meeting safety regulations and standards by continuously monitoring environmental conditions.

**Source Code –**

```
#include <LiquidCrystal.h>  // Include the LCD library

// Initialize the LCD with the number of columns and rows

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

const int tempPin = A0;  // Analog pin connected to the LM35

void setup() {

  lcd.begin(16, 2); // Set up the LCD's number of columns and rows

  lcd.print("Temp: "); // Print a message to the LCD

}

void loop() {

  int analogValue = analogRead(tempPin); // Read the analog value from LM35

  float voltage = analogValue * (5.0 / 1023.0); // Convert the analog value to voltage

  float temperatureC = voltage * 100; // Convert voltage to temperature in Celsius

 lcd.setCursor(0, 1); // Set cursor to the beginning of the second line

  lcd.print("Temp: ");

  lcd.print(temperatureC); // Print the temperature to the LCD

  lcd.print(" C");

delay(1000); // Wait for a second before updating the display

}
```

**Circuit Diagram –**




**Conclusion -** This basic industrial data monitoring system demonstrates fundamental concepts of data acquisition and display using Arduino technology. It integrates a temperature sensor with a microcontroller and an LCD to provide real-time environmental data and highlights the importance of effective data monitoring in maintaining operational efficiency and safety in industrial environments.