Task 1 Report:

Smart Home Energy Management System Simulation

1. Introduction

This project demonstrates the implementation of a “Smart Home Energy Management System” using an ‘Arduino Uno’ and multiple sensors in a simulated environment on Wokwi. The system is designed to monitor a) temperature, b) motion, and c) distance in a virtual smart apartment. Sensor readings are output to the Serial Monitor (or an LCD, if desired), and individual LEDs indicate when certain thresholds are met.

2. Objectives

- **Simulation Setup:** Build a circuit in Wokwi incorporating a “Temperature sensor”, a “PIR motion sensor”, an “HC-SR04 Ultrasonic distance sensor”, an “LCD display” (optional), and three LEDs (one for each sensor).
- **Functionality:**
  - **Temperature Monitoring:** Read temperature values (adjustable via a slider) and light an LED when a specified threshold is exceeded.
  - **Motion Detection:** Detect any motion using PIR sensor. The sensor should output a signal for 5 seconds when triggered and then remain inactive for an inhibit period (approximately 1.2 seconds) before detecting any motion again, with a corresponding LED indicator.
  - **Distance Measurement:** Use HC-SR04 sensor to measure distance (via an adjustable slider) and light an LED when a designated distance threshold is met.
- **Output:** Display sensor readings and system status on the Serial Monitor (or an LCD) and status through the LEDs.

3. List of components Used to design circuit

- **Arduino Uno** (Microcontroller board)
- **Temperature Sensor**
  (Simulated with a slider control on Wokwi to adjust the reading)
- **PIR Motion Sensor**
  (Includes a toggle button for simulating motion)
- **HC-SR04 Ultrasonic Distance Sensor**
  (Reading adjustable via a slider on the simulator)
- **LCD Display** (Optional)
  (Can be used to display sensor readings instead of or in addition to the Serial Monitor.)
- **3 LEDs**
  (One LED per sensor, lighting up when specific conditions are met.)

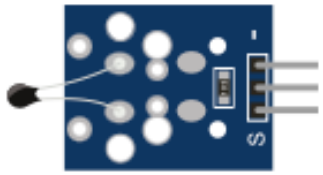
4. Circuit Design and Connections

In the Wokwi simulation, the components are connected as follows:

- **Temperature Sensor: (NTC Thermistor-Based Temperature Monitoring System)**
  Connected to Arduino's analog input port pin (#A5). The simulated slider allows for dynamic adjustment of the temperature reading in the run-time. An LED is connected to a digital output (GPIO) pin (#3) to glow and indicate when temperature exceeds greater than 24 (deg. C) & goes lesser than 19 (deg. C.)

```
- Controls an LED indicator to simulate a smart cooling and heating system:
    - Temperature < 19°C → Heating LED ON
    - Temperature > 24°C → Cooling LED ON
    - 19°C ≤ Temperature ≤ 24°C → System remains stable, LED OFF
```

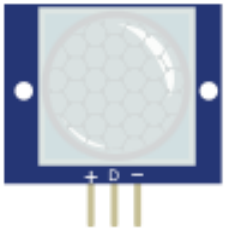
```
Expected Behavior:
- Displays temperature in real-time.
- Activates respective LEDs based on temperature thresholds.
```



Please refer schematic diagram to get more details about pin configurations performed.

- **PIR Motion Sensor: (Passive Infrared Motion Sensor System)**
  Connected to Arduino's digital input port (GPIO) pin (#5). When motion is simulated using the toggle button in the run-time, sensor outputs a high signal for "5 seconds", and then goes low, with an inhibit period of "1.2 seconds". An associated LED connected to another digital (GPIO) port pin (#6) lights up whenever motion is detected.

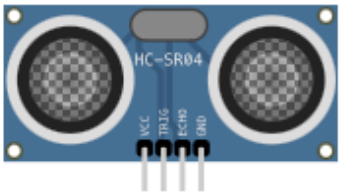
```
Expected Behavior:
- Motion detected → LED ON, LCD updates, Serial prints "Motion detected!"
- After 5 seconds → LED OFF, LCD updates, Serial prints "Motion ended!"
- 1.2 seconds inhibit time before detecting new motion.
```



Please refer schematic diagram to get more details about pin configurations performed.

- **HC-SR04 Ultrasonic Sensor: (Ultrasonic Distance Measurement System)**
  Uses two digital pins (Trigger on pin #9 and Echo on pin #2) and connected to Arduino's digital GPIO port. A simulated slider allows for adjusting the measured distance in the run-time. A corresponding LED (connected to digital GPIO port pin #10) lights up when the measured distance meets the below defined threshold levels.

```
Expected Behavior:
- Displays real-time distance on the LCD.
- If an object is closer than 15 cm → LED ON.
- If an object is farther than 15 cm → LED OFF.
- Logs measurements to the Serial Monitor.
```

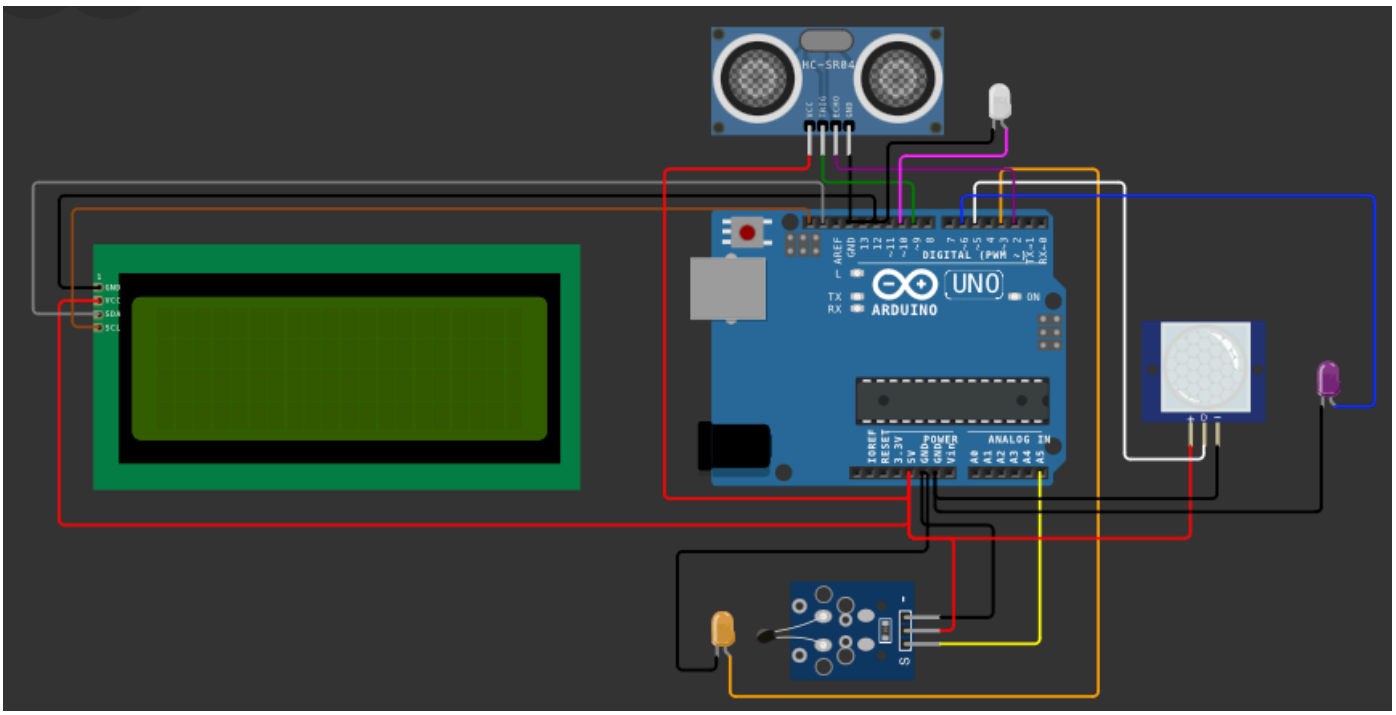


  Please refer schematic diagram to get more details about pin configurations performed.

- **LCD Display (Optional): (Liquid Crystal Display Monitoring System)**
  LCD (4 lines x 20 characters) is interfaced with Arduino's I2C port. It displays the sensor readings for easy monitoring and to understand the changes happening.

The complete wiring diagram is created and simulated within the Wokwi environment.

Sharing below the schematic diagram design of sensors and components interfaced with Arduino Micro-controller.



Algorithm preparation:

Temperature Monitoring System:

- Read Temperature sensor input at Analog input port of Arduino Micro-controller
- Convert Sensor value to temp in Celsius, and display the real-time temperature on LCD screen
- Display heating or cooling status based on temperature range
- Turn on LED when heater or cooler is switched ON, and turn-off when temperature is normal

PIR Motion Sensor System:

- Read Motion sensor value at GPIO port pin (turns high while any motion is detected)
- Give update at display (Motion Detected), and turn-on LED
- Start timer, and turn-off after 5 seconds (Motion Ended)
- Reset motion status after 5 seconds
- Inhibit motion detection for 1.2 sec. to detect the next motion
- Motion status display on LCD and logged in the Serial Monitor.

Ultrasonic Distance Measurement System:

- Send short pulse to trigger the sensor as first step
- Then capture the time consumed for the Echo pulse to return
- Convert the duration into distance in “centimeters” format
- Simulation of real-world noise scenario to deviate 5 to 12 cm randomly
- Turn-on LED when distance is less than 15 cm, otherwise keep LED turned-off
- Display the value at both LCD, and Serial Monitor for debugging purpose.

5. Code Overview and Implementation

The code for Task 1 includes the following key sections:

a. Setup

- **Initialization:**
  In the `setup()` function, the code initializes the Serial Monitor (for debugging and monitoring), sets the pin modes for the sensors and LEDs, and initializes the LCD.

```
void setup() {
  // Initialize LCD screen
  lcd.init();
  lcd.backlight();

  // Start serial communication
  Serial.begin(9600);

  // Display initial message on display
  lcd.setCursor(2, 2);
  lcd.print("Welcome to Rita's Smart Home");

  // Set pin modes
  pinMode(TEMP_SENSOR, INPUT);
  pinMode(TEMP_LED, OUTPUT);
  pinMode(MOTION_SENSOR, INPUT);
  pinMode(MOTION_LED, OUTPUT);
  pinMode(ULTRASONIC_SENSOR_TRIG, OUTPUT);
  pinMode(ULTRASONIC_SENSOR_ECHO, INPUT);
  pinMode(ULTRASONIC_LED, OUTPUT);
}
```

b. Main Loop

Which calls the following individual sub-functions,

- **Temperature Monitoring:**
  The code reads the temperature sensor value (which is adjustable via the slider), converts it to a temperature value, and prints it to the Serial Monitor (or displays it on the LCD). LED is turned ON/OFF according to design considerations performed,

```
void senseTemperature() {
  // Read analog value from temperature sensor
  int tempSensorValue = analogRead(TEMP_SENSOR);

  // Convert sensor value to temperature in Celsius
  float temperature = 1 / (log(1 / (1023.0 / tempSensorValue - 1)) / 3950 + 1.0 /
298.15) - 273.15;

```

```
    // Convert temperature to a string
    char tempStr[6];
    dtostrf(temperature, 4, 1, tempStr);

    // Display temperature on display
    lcd.setCursor(0, 0);
    lcd.print("Temp: ");
    lcd.print(tempStr);
    lcd.print(" *C  ");

    // Display heating or cooling status based on temperature range
    lcd.setCursor(0, 1);
    if (temperature < 19) {
      digitalWrite(TEMP_LED, HIGH); // Turn on heating LED
      lcd.print("Heating: ON    ");
    } else if (temperature > 24) {
      digitalWrite(TEMP_LED, HIGH); // Turn on cooling LED
      lcd.print("Cooling: ON    ");
    } else {
      digitalWrite(TEMP_LED, LOW); // Turn off LED when temperature is stable
      lcd.print("Stable Temp    ");
    }
  }
```

- **Motion Detection:**
  The PIR sensor state is read. When motion is detected, the code prints a message and lights the corresponding LED for 5 seconds, then respects the inhibit period (1.2 seconds) before checking for motion again.

```
  void senseMotion() {
    // Read motion sensor value (HIGH if motion detected)
    motionStatus = digitalRead(MOTION_SENSOR);

    // If motion is detected for the first time
    if (motionStatus == HIGH && !motionDetected) {
      digitalWrite(MOTION_LED, HIGH); // Turn on motion LED
      motionDetected = true;
      motionTimer = millis(); // Save the current time

      // Display motion status on dislay
      lcd.setCursor(0, 2);
      lcd.print("Motion: Detected ");
      Serial.println("Motion detected!");
    }

    // If motion was detected but 5 seconds have passed, reset motion status
    if (motionDetected && millis() - motionTimer > 5000) {
      motionDetected = false;
      digitalWrite(MOTION_LED, LOW); // Turn off motion LED

```

```
      // Display no motion status on LCD
      lcd.setCursor(0, 2);
      lcd.print("Motion: None     ");
      Serial.println("Motion ended!");
    }

    // Add 1.2s inhibit time before detecting motion again
    delay(1200);
  }
```

- **Distance Measurement:**
  The HC-SR04 sensor is used to measure the distance. The code sends a trigger pulse, waits for the echo, calculates the distance, and prints the value. If the distance falls below (or above) a set threshold, the corresponding LED is activated.

```
void senseUltrasonicDistance() {
  // Send a short pulse to trigger the ultrasonic sensor
  digitalWrite(ULTRASONIC_SENSOR_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC_SENSOR_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_SENSOR_TRIG, LOW);

  // Measure the time taken for the echo pulse to return
  long duration = pulseIn(ULTRASONIC_SENSOR_ECHO, HIGH);

  // Convert time to distance (cm)
  int distance = duration * 0.034 / 2;

  // Apply a random deviation of 5-12 cm to simulate real-world noise
  int adjustedDistance = distance + random(5, 13);

  // Display distance on LCD
  lcd.setCursor(0, 3);
  lcd.print("Dist: ");
  lcd.print(adjustedDistance);
  lcd.print(" cm ");

  // If distance is less than 15 cm, turn on LED
  if (adjustedDistance < 15) {
    digitalWrite(ULTRASONIC_LED, HIGH);
  } else {
    digitalWrite(ULTRASONIC_LED, LOW);
  }

  // Print distance values to Serial Monitor for debugging
  Serial.print("Measured Distance: ");
  Serial.print(distance);
  Serial.print(" cm | Adjusted Distance: ");
```

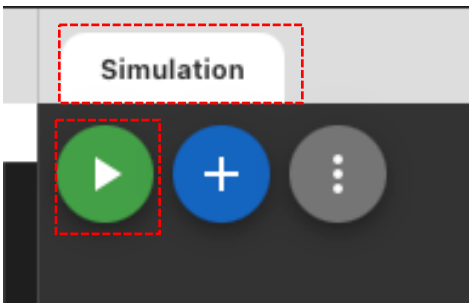
```
  Serial.print(adjustedDistance);
  Serial.println(" cm");
}
```

c. Additional Details

- **Function Definitions:**
  Followed according to Wokwi suggested guidelines to simulate the system.
- **Interactivity:**
  The Wokwi simulator's interface allows users to interact with the simulation by adjusting the sensor values (via sliders and toggle buttons).

6. Simulation Environment

- **Platform:**
  The entire circuit is built and simulated on Wokwi. This online simulator enables interactive testing of the Arduino circuit and code without the need for physical components.
- **Interactivity:**
  - **Temperature Sensor:** Adjust the slider to simulate changes in temperature.
  - **PIR Sensor:** Use the toggle button to simulate motion.
  - **Ultrasonic Sensor:** Adjust the distance reading slider to simulate varying distances.

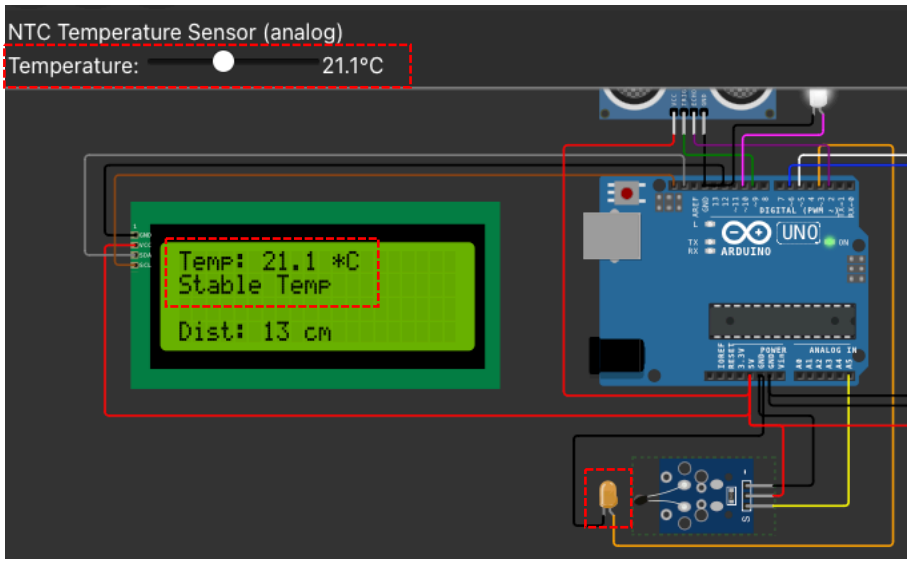


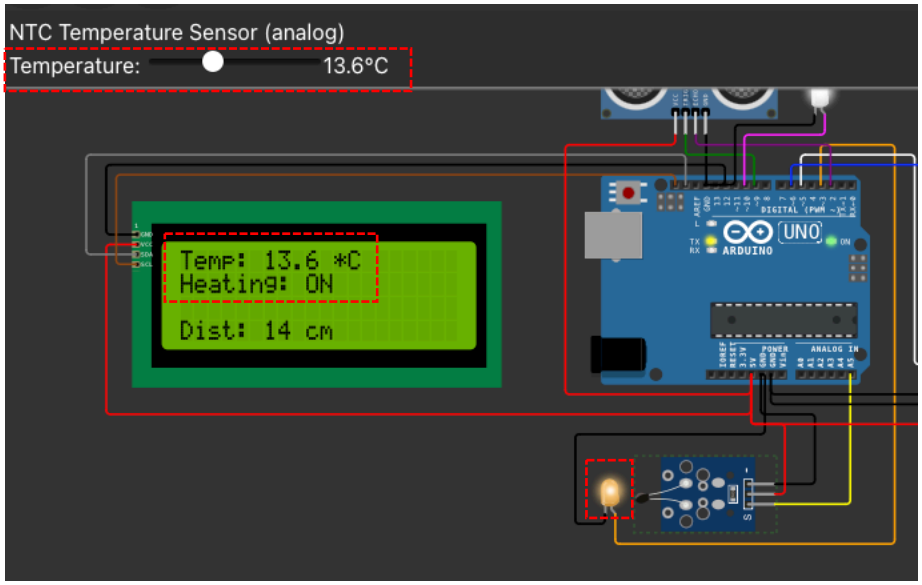
Above Simulation pane shows the Arrow button to start to build and execute the environment (and second image helps to understand the run-time duration)

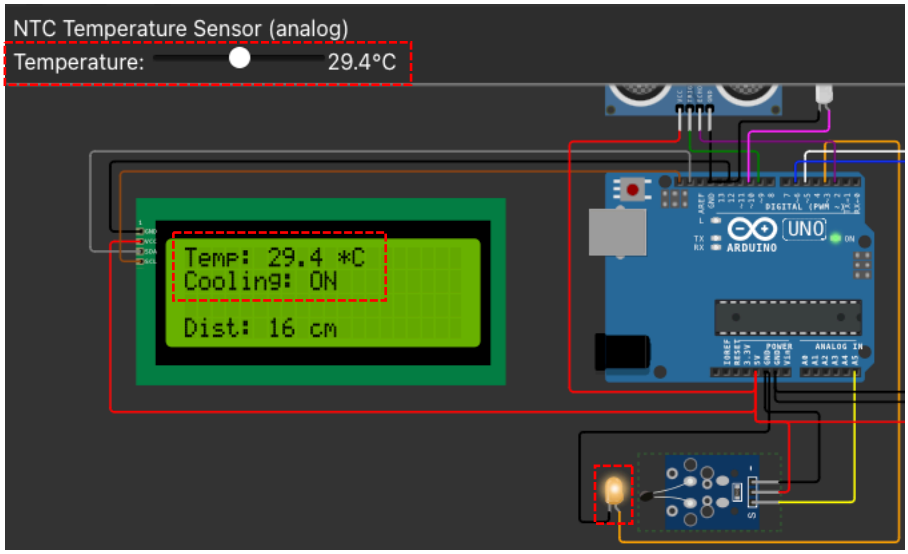
- **Outputs:**
  The Serial Monitor displays real-time sensor readings. Additionally, the respective LEDs light up based on sensor thresholds, providing immediate visual feedback.

7. Results and Observations

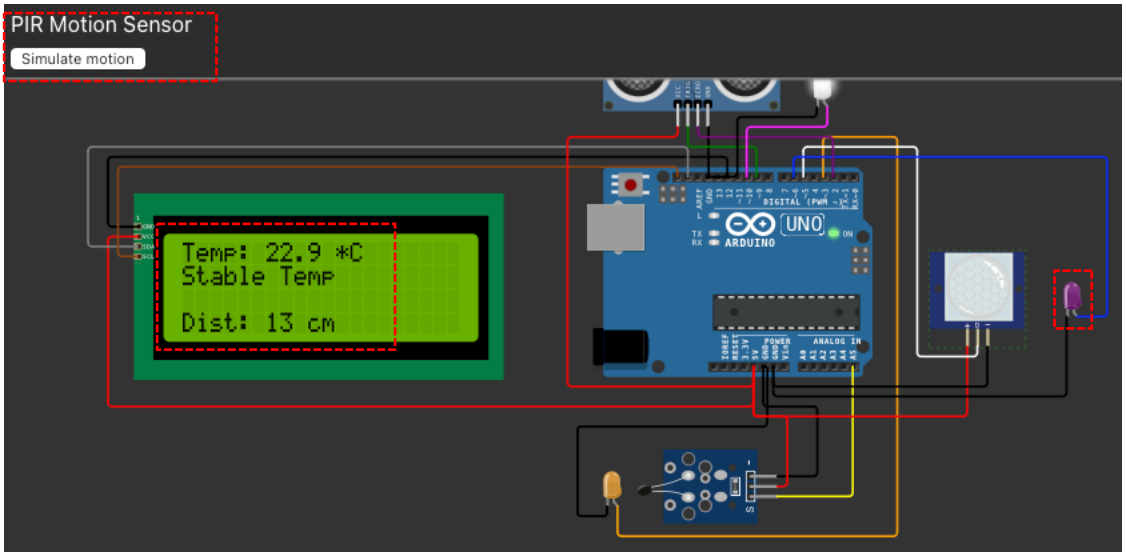
- **Temperature Monitoring:**
  When the temperature reading (simulated via the slider) exceeds the defined threshold, the temperature LED lights up and the Serial Monitor (or LCD) displays the current temperature.

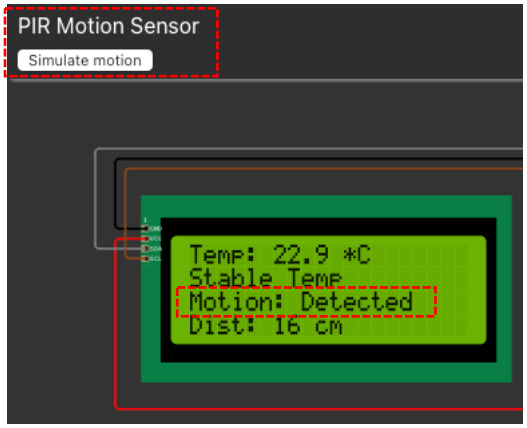

NTC Temperature Sensor (analog)
Temperature:
21.1°C
Temp: 21.1 *C
Stable Temp
Dist: 13 cm
UNO
ARDUINO

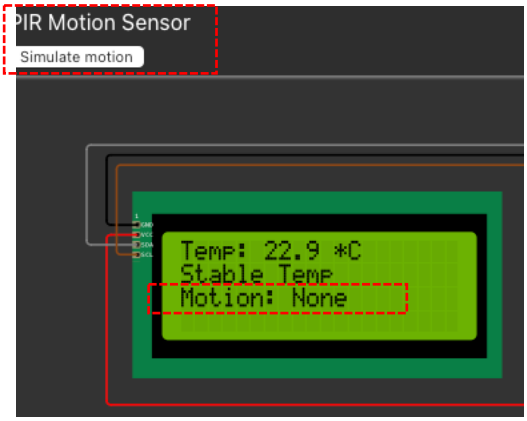

NTC Temperature Sensor (analog)
Temperature:
13.6°C
Temp: 13.6 *C
Heating: ON
Dist: 14 cm
UNO
ARDUINO

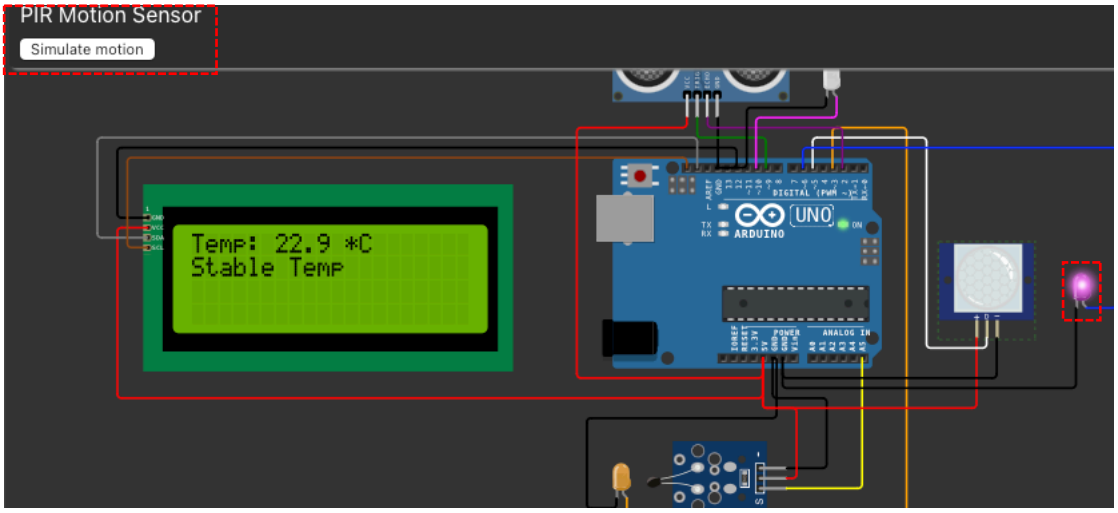

NTC Temperature Sensor (analog)
Temperature:
29.4°C
Temp: 29.4 *C
Cooling: ON
Dist: 16 cm
UNO
ARDUINO

- **Motion Detection:**
  When motion is simulated, a message is printed, and the motion LED remains on for 5 seconds before automatically turning off and entering a 1.2-second inhibit period.

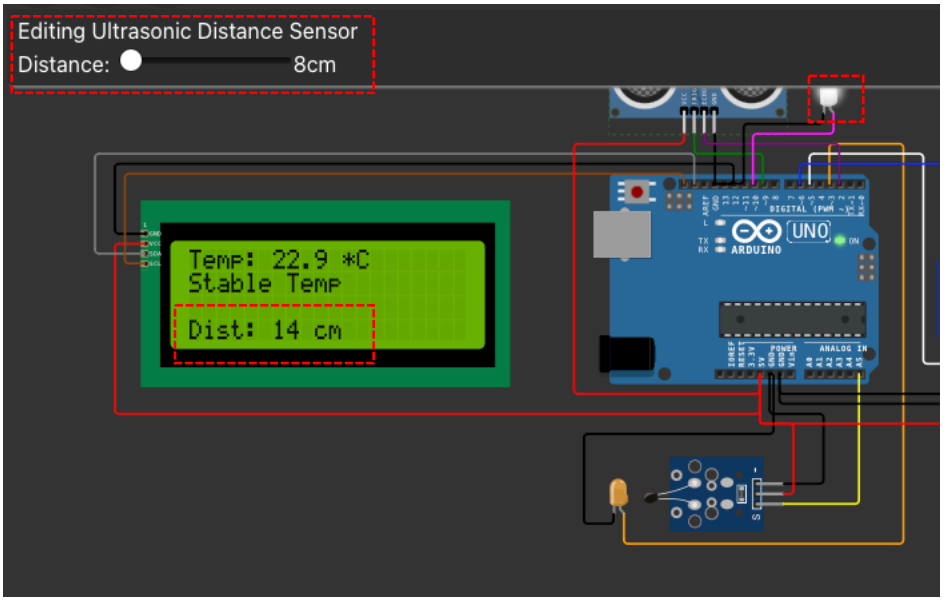


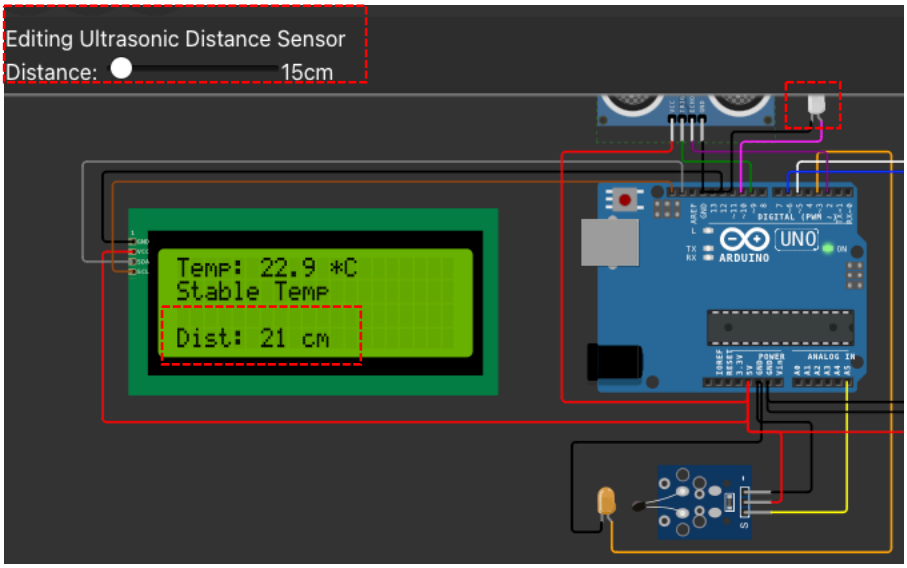






- **Distance Measurement:**
  Adjusting the ultrasonic sensor's slider changes the measured distance, and when the distance meets the specified criteria, the corresponding LED is activated.





Overall, the simulation successfully mimics a smart monitoring system by integrating sensor inputs with corresponding outputs.

8. Conclusion

The Task 1 solution for the Smart Home Energy Management System has been successfully implemented on Wokwi. The project integrates a temperature sensor, PIR motion sensor, and HC-SR04 ultrasonic sensor with an Arduino Uno. The sensor data is processed in real-time, displayed via the Serial Monitor (or an LCD), and indicated with dedicated LEDs when specific thresholds are met. This simulation provides a solid foundation for further experimentation and potential integration with predictive models.

Project Link: https://wokwi.com/projects/423415297924133889