Internet of Thing
Individual Assignment
(Practical)
Vo Dang Khoi Nguyen
105241532

Content:

1. Project summary

a) Topic background and purpose of the system

This is an IoT based room environment temperature monitoring system and it is designed to measure light intensity and temperature of the surrounding area. This system is built on Arduino and Raspberry Pi as well as sensors such as DHT22 temperature sensor and LDR light sensor to measure the indoor conditions. The Arduino will collect data from the light sensor and use the LED to indicate the current status of lighting in the room, Red LED is when the light in the room is too bright, Yellow LED for too dark and Green LED for ideal light levels. For the temperature sensor, the Arduino will also collect the data and based on that data if the room temperature

exceeds the threshold the system will activate the buzzer to alert about the heat level which warns of the need for changes. All of the sensor's data will be transmitted via serial to Raspberry Pi which acts as an edge device, it will store the collected data in the Mariadb database and run Flask which is a web interface that the users can use.

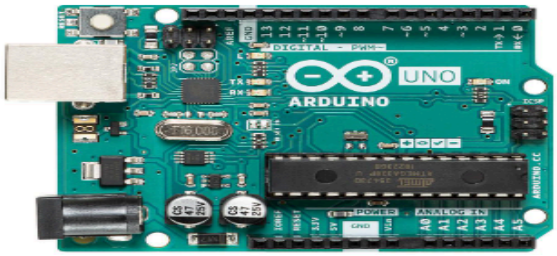
- View the latest temperature and light readings.
- Able to adjust, change the threshold of the light and temperature directly from the web.
- Able to see the max and min of the light and temperature throughout the entire database.

2. Conceptual design

a) Block diagram of IoT Architecture

DHT22
(Temperature Sensor)

SEN0043
(Light Sensor)

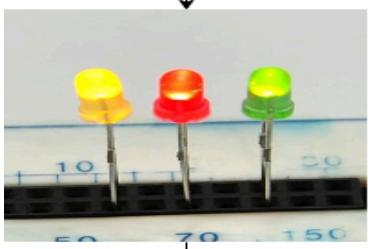


Arduino Uno

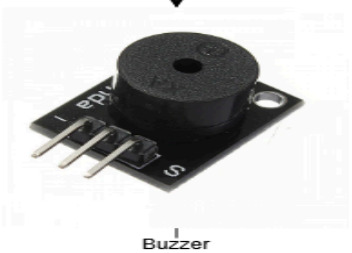
Raspberry Pi
(Edge device)

MariaDB

Mariadb Database



LED



Buzzer

Web Interface

The block diagram above shows an IoT based room temperature monitoring system which observes the condition of the room automatically and based on the current conditions there will be a response from both visual and audio actuator like LED lights and buzzer.

Sensors:

- **DHT22 (Temperature sensor)**
    - This sensor is used to measure the air temperature in real time. It collects data and sends it to the microcontroller, then it will check if the collected data exceeds the pre-defined value or not and it will trigger the buzzer to respond if necessary .
- **SEN0043 (Light sensor)**
    - The light sensors measure the intensity of the light of the room and Arudino also collects data from the sensors and decides which LED will be turned on or off to represent whether the room is too dark or too bright or just right.

Microcontroller:

- **Arduino Uno**
    - The arduino acts as a brain and collects the data from both light and temperature sensors and sends the collected data to the raspberry pi for further action and storage.

    - It sends data to the raspberry pi and also receives control signals from the raspberry pi such as updated threshold value from Raspberry Pi and allows dynamic configuration like turning on or off the LED, buzzer, changing the threshold.

Raspberry Pi:

- The raspberry pi acts as an edge device in the setup that performs multiple different tasks in the system like receiving real time data from the Arduino and storing those data in the Mariadb database for historical tracking and analysis.

- It also runs Flask to let us run a web interface that can interact with the system and also display values such as current temperature, current light level and also show the highest and lowest record of both the light and temperature in the entire database.

- The Raspberry Pi is also the one used to communicate with the web interface. Giving the user information about the system, is it working? What happened at what time?

Maria Database:

- The database is used to store sensor data like light level and temperature and it also keeps track of changes over time, it also lets the system report about minimum and maximum conditions observed.

- It also give the user about the history of the system of what happened for the past time and helping user to interactive with the system through the web interface

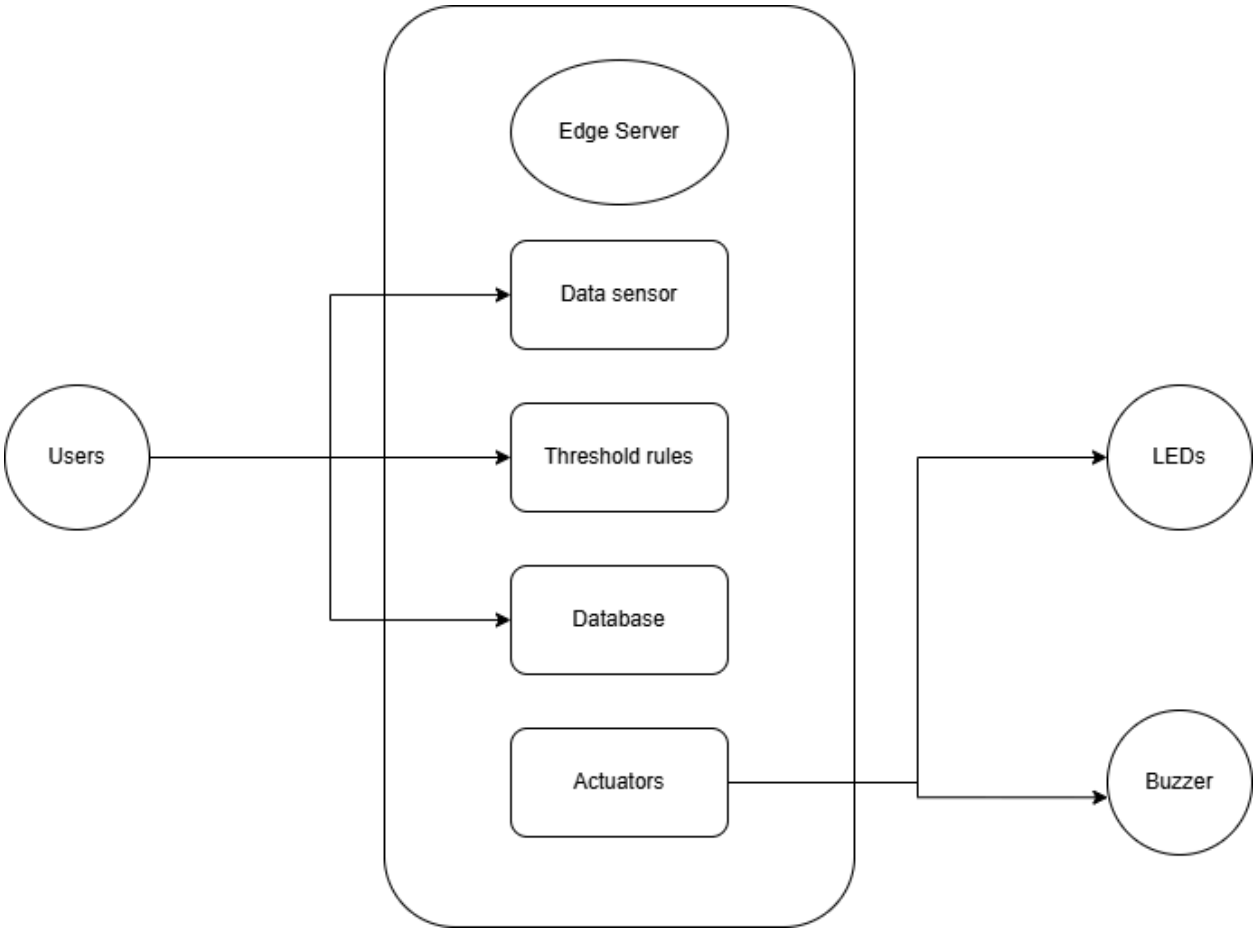
Actuators:

- **LEDs:**
    - The LED light is based on the light sensors. If there is no light or limited sunlight from a cloudy day it will turn on the yellow LED and if its too bright the red LED will power on and to alert about the condition and green light its when the light is optimal.
        - Yellow = too dark / dim
        - Red = too bright
        - Green = Optimal lighting

- **Buzzer:**
    - The buzzer is used to alert the user about the current temperature of the room when it's too hot based on the current threshold of the system. The buzzer helps to notify the user about the surrounding environment which may be uncomfortable or needing cooling.

- **Web Interface:**
    - The web interface is created by using Flask on Raspberry Pi and allow user to:
        - View live sensor readings
        - Allow user to adjust threshold of the light and temperature without directly interact with the sensors or Arduino
        - View max and min of recorded values

b) Case Diagram:



Users:

- These are the people who use the web interface to interact with the system
- They can:
    - Manually control the Actuators.
    - Change the threshold manually base on how the users want
    - View the sensor data in real time

- View the history of the system, for example what has happened for the last 2 hours.

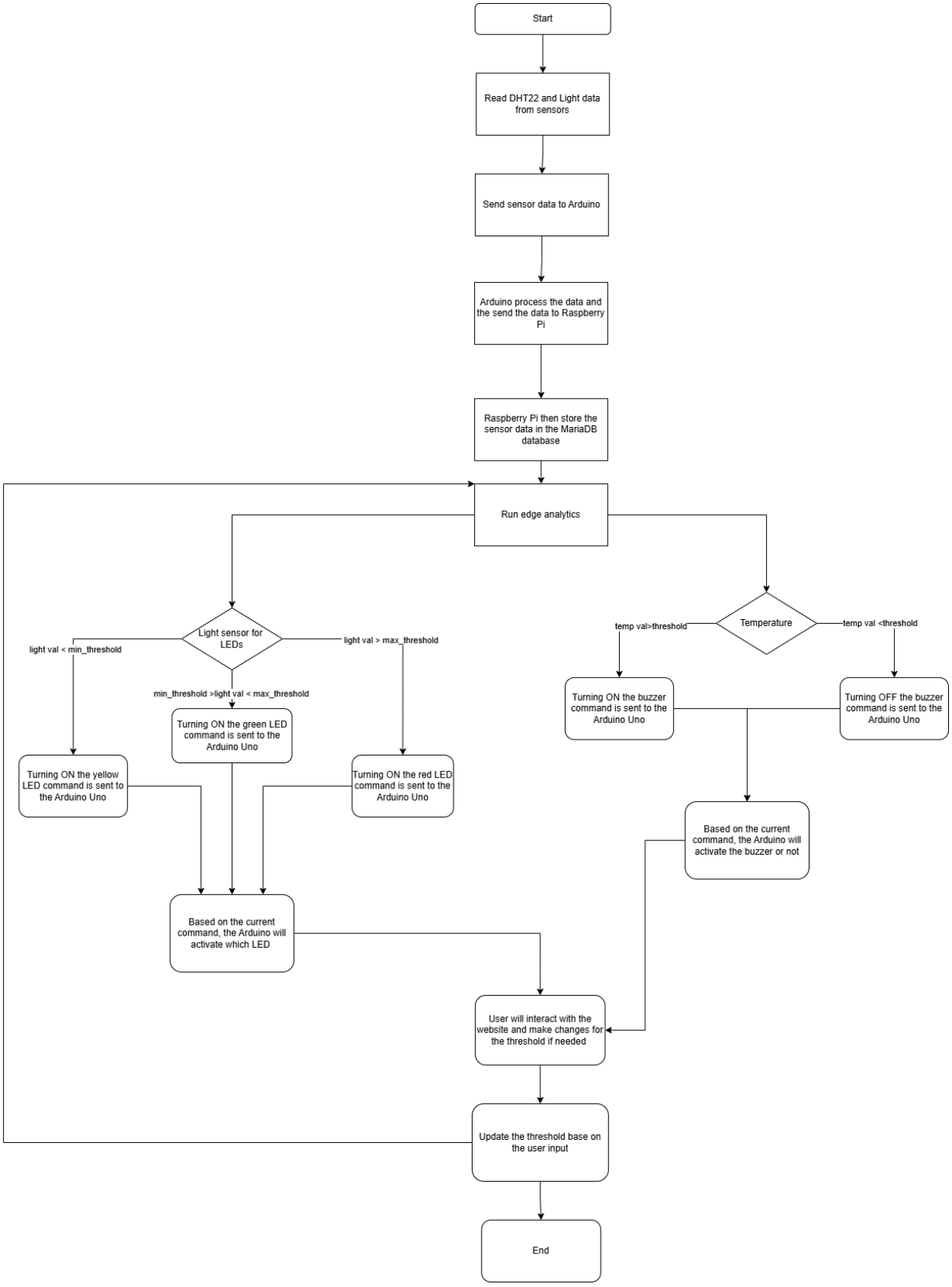
Edge Server:

- This is the main controller of the Raspberry Pi and it processes data in real time and makes decisions that are close to the devices for faster results.
- The edge server include 4 components:
  - Data Sensor
    - Arduino collect the data from the light and temperature sensors in real time
    - Data will then be send to the Raspberry Pi and allow users to read sensor data on the web interface
  - Threshold Conditions:
    - Allow the users to change the threshold conditions based on their input.
    - These rules can be either pre-designed or modified by the user.
    - The new threshold value will be sent to the Arduino from the Raspberry Pi to guide the physical output of the actuators.
  - Database:
    - The database uses to store the sensors reading
    - Keep user setting, including the threshold input which can be user for analysis or debugging
  - Actuators:
    - Based on the reading of the sensors and the threshold it will send a signal to activate which LEDs or to activate the buzzer or not.

Actuators:

- **LEDs:**
  - Indicate current light level, red means too bright, yellow is too dark / dim and green means perfect lighting condition.
- **Buzzer:**
  - It will be turned ON when the temperature crosses over the user defined threshold and audio alerts the users about overheating conditions currently in the room.

c) Flow chart


Start
Read DHT22 and Light data from sensors
Send sensor data to Arduino
Arduino process the data and the send the data to Raspberry Pi
Raspberry Pi then store the sensor data in the MariaDB database
Run edge analytics
Light sensor for LEDs
light val < min_threshold
min_threshold >light val < max_threshold
light val > max_threshold
Turning ON the yellow LED command is sent to the Arduino Uno
Turning ON the green LED command is sent to the Arduino Uno
Turning ON the red LED command is sent to the Arduino Uno
Based on the current command, the Arduino will activate which LED
Temperature
temp val>threshold
temp val <threshold
Turning ON the buzzer command is sent to the Arduino Uno
Turning OFF the buzzer command is sent to the Arduino Uno
Based on the current command, the Arduino will activate the buzzer or not
User will interact with the website and make changes for the threshold if needed
Update the threshold base on the user input
End

1) Read data from moisture sensor and light sensors:

- **Soil moisture sensor**:
    - Measure what is the current temperature of the room

- **Light sensor:**
    - It measures the light level to check if the current light around is too bright or too dark.

2) Arduino collect data from the sensors:

- The collected data from the sensors will be sent to the arduino to process.

3) Arduino send collected data to Raspberry Pi

- The processed data then gets transported to Raspberry Pi through a USB cable for further handling.

4) Raspberry Pi stores the sensor data in database

- This is where the Raspberry Pi saves the collected sensor data which is the MariaDB database, which keeps the record of the sensor over time for future analysis.

5) Edge analytic:

- The raspberry pi will run the data locally (on device) for faster run time, instead of sending it to the cloud and getting back which will cost more time and will not be efficient.
- It also check if any values cross the predefined value (threshold) such as the light levels, temperature level

6) Condition check:

- **<u>Light Sensor for LEDs</u>**
    - If the reading on the brightness is **higher** than the max_threshold
        - The system send a turn **ON** command for the red LED to the Arduino, indicating too bright lighting

    - If the reading on the brightness is **lower** than the min_threshold
        - The system send a turn **ON** command for the shading yellow LED to the Arduino, indicating low lighting

    - If the reading on the brightness is between the min_threshold and max_threshold then its mean the current light condition is good

        - The system send a turn **ON** command for green LED to the Arduino, indicating the perfect lighting range

- **<u>Temperature sensor for buzzer:</u>**
    - If the reading on the temperature is **lower** than the threshold
        - The system send a turn **OFF** command for the buzzer to the Arduino
    - If the reading on the temperature is **higher** than the threshold
        - The system send a turn **ON** command for the buzzer to the Arduino

7) <u>Activating actuator:</u>

- Arduino will receive the command from the Raspberry Pi and control the actuators.
    - It turn ON / OFF the LED light, buzzer on the command

8) <u>Interaction With Web Interface:</u>

- Manually override any system
- Can change the threshold value
- View real time reading of sensors

9) <u>Update threshold:</u>

- Through the web interface the user can update the threshold
- Update the system behaviour based on what the users change

3) Implementation

<u>3.1 Sensors integration:</u>

a) Light sensors (Analog sensor)

- What are you measuring?
    - The intensity of the light affects the surrounding environment.
- Why measure it?
    - Because based on the room light intensity we can see if the room it's too bright or it's too dark and the system can decide if it needs to turn ON or OFF which LEDs.
- How will I integrate?:

- The analog output pin of the light sensors will be connected to one of the Arduino input pins A0 and the Arduino Uno will read the value and send it to Raspberry Pi for analysis and storage.

b) Temperature Sensors:

- What are you measuring?
  - The temperature of the room to see what the current temperature is and alert if it's too hot.

- Why measure it?
  - Based on the temperature level of the room, the system decides if they need to turn on the buzzer to alert them about the condition of the room temperature.

- How will I integrate?
  - The digital output pin will be connected to an input digital input pin number 8 on the Arduino and the sensor's output will be HIGH or LOW based on the room condition. Finally the data will be sent to Raspberry Pi for storage and analysis.

3.2) Actuators

b) LEDs:

- What does it do?
  - It provides LED light for the user alert about the current light level and if it needs to be raised or not.

- How will I integrate it?
  - The LEDs are connected to the Arduino and based on the sensor reading it, the arduino will receive commands to turn on or off the LED.

c) Buzzer:

- What does it do?
  - It provides an audio alert when the room temperature exceeds the threshold alerting the user.

- How will I integrate it?
  - The buzzer will be connected to the digital pin number 8. The reading of the temperature will be processed by the Arduino and when the value is above the threshold the arduino will trigger the buzzer accordingly.

3.3) Software and Libraries

- Arduino (C++ libraries)
    - DHT.h
        - Allow the user to use the DHT22 temperature sensor and this library also allows the Arduino to read the temperature in Celsius. Finally it simplifies the communication of the sensor and handles timing better giving more accurate reading of the sensor.
    - Serial Communication
        - In order to send data from the Arduino to the Raspberry Pi and receive the updated value from the Raspberry Pi the Arduino has to use the built in Serial function. This is one of the most important aspects for real time interaction between the edge device and the microcontroller.

- Raspberry Pi
    - pyMySQL
        - A python library that I used to connect the database to the Flask web interface. It handles MySQL tasks such as inputting sensor data and getting the current value or giving the history.
    - Threading
        - Python threading is used to run the serial data reading and flask server, which help the data be inserted while the web page is still running and remain responsive.

- Flask
    - Flask is use to create a web page that can interact with the system directly and it allow user to:
        - View current temperature and current light level
        - Adjust threshold for both light and temperature
        - View the max and min reading for the light and temperature throughout the entire database

4) Resources

- Tutorial PDF (tutorial 6, 7, 3 and 4)
    - Step by step help me develop and install the database, connecting to Arduino and wiring physical components.

5) Appendix

- **Arduino code:**

```
#include "DHT.h"

const int Green_LED = 5;
const int Yellow_LED = 6;
const int Red_LED = 7;
const int Light_sensor = A0;
const int buzzer = 8;
const int temp_sensor = 3;

#define DHTTYPE DHT22
DHT dht(temp_sensor, DHTTYPE);

// Threshold variables
float tempThreshold = 24.0;
int lightMin = 10;
int lightMax = 200;

void setup() {
  pinMode(Green_LED, OUTPUT);
  pinMode(Yellow_LED, OUTPUT);
  pinMode(Red_LED, OUTPUT);
  pinMode(buzzer, OUTPUT);

  Serial.begin(9600);
  dht.begin();
}

void loop() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();  // Remove whitespace

    int firstComma = input.indexOf(',');
    int secondComma = input.lastIndexOf(',');

    if (firstComma > 0 && secondComma > firstComma) {
      float newTemp = input.substring(0, firstComma).toFloat();
      int newMin = input.substring(firstComma + 1, secondComma).toInt();
      int newMax = input.substring(secondComma + 1).toInt();
```

```
    // Update thresholds
    tempThreshold = newTemp;
    lightMin = newMin;
    lightMax = newMax;

    Serial.print("Updated thresholds: Temp=");
    Serial.print(tempThreshold);
    Serial.print(", LightMin=");
    Serial.print(lightMin);
    Serial.print(", LightMax=");
    Serial.println(lightMax);
  }
}


int light_val = analogRead(Light_sensor);
float temperatureC = dht.readTemperature();

digitalWrite(Green_LED, LOW);
digitalWrite(Yellow_LED, LOW);
digitalWrite(Red_LED, LOW);
digitalWrite(buzzer, LOW);

if (isnan(temperatureC)) {
  Serial.println("Error, failed to load temperature!");
  return;
}

Serial.print("Temp: ");
Serial.print(temperatureC);
Serial.print(" | Light: ");
Serial.println(light_val);


if (light_val > lightMax) {
  digitalWrite(Red_LED, HIGH);
} else if (light_val < lightMin) {
  digitalWrite(Yellow_LED, HIGH);
} else {
  digitalWrite(Green_LED, HIGH);
```

```
  }


  if (temperatureC > tempThreshold) {
    digitalWrite(buzzer, HIGH);
  }

  delay(500);
}
```

- **insert_data.py**

```
import serial
import time
import pymysql
import threading
import os

from flask import Flask, render_template, request, redirect

ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
time.sleep(2)

db = pymysql.connect(
    host="localhost",
    user="pi",
    password="",
    db="pot_db"
)

cursor = db.cursor()

app = Flask(__name__)

temperature_threshold = 24.0
lightMax_threshold = 200
lightMin_threshold = 10

print("Storing Arduino data in DB")
```

```
def read_Arduino():
    global temperature_threshold, lightMax_threshold, lightMin_threshold

    while True:
        try:
            line = ser.readline().decode('utf-8').strip()
            if line and "Temp:" in line:
                print("Received:", line)

                parts = line.replace("Temp:", "").replace("Light:", "").split("|")
                temperature = float(parts[0].strip())
                light = int(parts[1].strip())

                cursor.execute(
                    "INSERT INTO potlog (temperature, light) VALUES (%s, %s)",
                    (temperature, light)
                )
                db.commit()

                command = f"{temperature_threshold}, {lightMin_threshold},
{lightMax_threshold}\n"
                ser.write(command.encode())

        except Exception as e:
            print("Inserting error:", e)

@app.route('/', methods=['GET', 'POST'])
def index():
    global temperature_threshold, lightMax_threshold, lightMin_threshold

    if request.method == 'POST':
        temperature_threshold = float(request.form['temp'])
        lightMax_threshold = int(request.form['light_max'])
        lightMin_threshold = int(request.form['light_min'])

        print(f"Updated threshold: Temp= {temperature_threshold}, Light
Min={lightMin_threshold}, Light Max={lightMax_threshold}")
        return redirect('/')
```

```
    cursor.execute("SELECT temperature, light FROM potlog ORDER BY id DESC LIMIT
1")
    latest = cursor.fetchone()

    cursor.execute("SELECT MAX(temperature), MIN(temperature), MAX(light),
MIN(light) FROM potlog")
    summary = cursor.fetchone()

    return render_template("index.html",
                           latest=latest,
                           summary=summary,
                           temp=temperature_threshold,
                           light_min=lightMin_threshold,
                           light_max=lightMax_threshold)

if __name__ == '__main__':
    threading.Thread(target=read_Arduino, daemon=True).start()
    app.run(host='0.0.0.0', port=5000)
```

- **index.html**

```
<!DOCTYPE html>
<html>
<head>
  <title> IOT pratical assignment</title>
</head>
<body>
  <h1>
    Live Sensors Dashboard
  </h1>

  {% if latest %}
    <h3>Current readings of sensors</h3>
    <p>Temperature: <strong>{{ latest[0] }} c</strong></p>
    <p>Light: <strong>{{ latest[1] }}</strong></p>
  {% else %}
    <p>No data</p>
  {% endif %}

  <h3>Threshold for sensors</h3>
```

```
  <form method = "POST">
    <label>Temperature Threshold:</label>
    <input type="number" step="0.1" name="temp" value="{{ temp }}"><br><br>

    <label>Light Minimum Threshold (C):</label>
    <input type="number" name="light_min" value="{{ light_min }}"><br><br>

    <label>Light Maximum Threshold (C):</label>
    <input type="number" name="light_max" value="{{ light_max }}"><br><br>

    <button type="submit">Submit</button>
  </form>

  <h3>Database Summary:</h3>
  <table border ="1" cellpadding="5">
    <tr>
      <th>Max temp</th><th>Min temp</th><th>Max Light</th><th>Min Light</th>
    </tr>
    <tr>
      <td>{{summary[0]}}</td>
      <td>{{summary[1]}}</td>
      <td>{{summary[2]}}</td>
      <td>{{summary[3]}}</td>
    </tr>
  </table>
</body>
</html>
```