Ta Quang Tung - 104222196

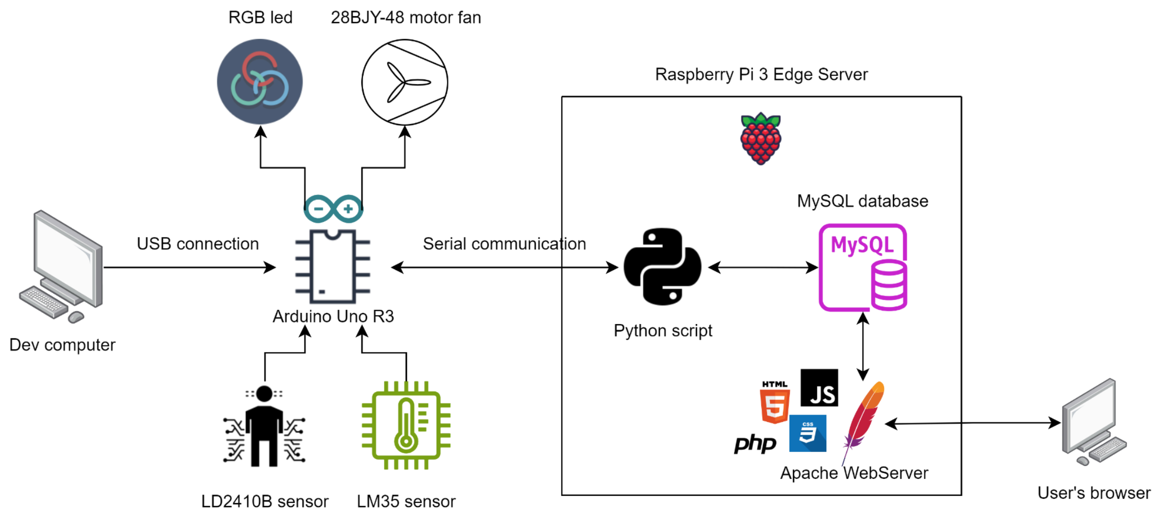
Individual Practical Assignment Report

Project summary

This report describes my Smart Home IoT project for the Individual Practical Assignment. This project is capable of sensing the presence of humans and measuring ambient temperature to turn devices on or off and adjust their settings. The temperature and presence sensor data is stored in a database and can be viewed through a web interface. The web interface also allows the user to directly interact with the devices.

The project is composed of three parts: (1) an IoT node consisting of a LM35 temperature sensor, a LD2410B human presence sensor, an RGB led, and a 28BYJ-48 stepper motor fan all connected to an Arduino Uno R3 board, (2) a Raspberry Pi 3 acting as an edge server connected to the IoT node through a serial cable, and (3) a web interface hosted on the edge server showing real-time sensor data and options to directly interact with the IoT devices.

Implementation



Operation

The Arduino receives data from the temperature and human presence sensor and sends it to the Raspberry Pi through a serial connection. A Python script on the Raspberry reads this data, saves it to the database, and checks if the sensor data is above a certain threshold. If it is, the script sends commands to the Arduino using the same serial connection, after which the Arduino can adjust the light and fan settings such as on/off status, speed, and color. To view the collected sensor data, the user accesses the web interface hosted on an Apache Web Server on the Raspberry Pi. The user can also interact directly with the actuators or update the threshold values through this web interface. To send the user's commands to the

Arduino, the desired settings of these commands are first saved to the MySQL database by the web server. These settings are then periodically read and sent to the Arduino by the Python script through the serial connection.

Hardware and software

1. Sensors:
    a. LM35 temperature sensor (analog): Measures ambient temperature in Celcius. Its data is read and saved to the database every second.
    b. LD2410B human presence sensor (digital): Detects whether someone is in the direction it is facing. Used to automatically turn devices on or off. Its data is read every second.
2. Actuators:
    a. RGB led: Can be turned on or off automatically or manually. Its color can change automatically based on the ambient temperature or manually through the web interface.
    b. 28BJY-48 stepper motor (in conjunction with a ULN2003 driver): Used as a fan. Can be turned on or off automatically or manually. Its speed can change automatically based on the ambient temperature or manually through the web interface.
3. Other hardware:
    a. Arduino Uno R3: Powers and connects the sensors and actuators together to form an IoT node. It also sends data to the edge server and controls the actuators based on the commands received from the edge server.
    b. Raspberry Pi 3: Acts as the edge server. It contains a Python script to read from and write to the Arduino through a serial connection, a database, and a web server.
    c. Various wires, batteries, resistors, and breadboards.
4. Software and libraries:
    a. Python with the following libraries:
        i. PySerial: Facilitates serial communication with the Arduino.
        ii. MySQLdb: Connects to the MySQL database and performs queries.
    b. MySQL: Stores the sensor and device settings data
    c. Apache Web Server: Hosts the web interface.
    d. HTML, CSS, JavaScript: Standard front-end web technologies used to build the interface.

Resources

- Interfacing LM35 Temperature Sensor with Arduino: https://lastminuteengineers.com/lm35-temperature-sensor-arduino-tutorial/
- LD2410B Human Presence Sensor Documentation: https://drive.google.com/file/d/1njofFdf22mKB-NkT6jt08wI2Ye6XqE3V/view
- Interfacing RGB led with Arduino: https://projecthub.arduino.cc/semsemharaz/interfacing-rgb-led-with-arduino-b59902
- Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/

- AccelStepper library for Arduino: https://www.airspayce.com/mikem/arduino/AccelStepper/
- PySerial API: https://pyserial.readthedocs.io/en/latest/pyserial_api.html
- PyMySQL documentation: https://pymysql.readthedocs.io/en/latest/

Appendix

Code

My code is hosted on the following GitHub repository: https://github.com/pine04/iot-practical-individual

assignment2.ino

```
#include <AccelStepper.h>

#define MotorInterfaceType 4

const int LM35_PIN = A0;
const int LD2410_PIN = 2;
const int RED_PIN = 7;
const int GREEN_PIN = 6;
const int BLUE_PIN = 5;
const int IN1_PIN = 11;
const int IN2_PIN = 10;
const int IN3_PIN = 9;
const int IN4_PIN = 8;
const int SENSOR_EMIT_INTERVAL = 1000;
const float FAN_LOW = 300.0;
const float FAN_MED = 450.0;
const float FAN_HIGH = 600.0;
AccelStepper stepper(MotorInterfaceType, IN1_PIN, IN3_PIN, IN2_PIN,
IN4_PIN);

unsigned long previousMillis = 0, currentMillis;
int temp_adc_val;
float temp_val;
int presence_val;

int red = 255;
int green = 255;
int blue = 255;

bool isLightOn = true;
bool isFanOn = true;

void setup() {
  Serial.begin(9600);

  pinMode(LM35_PIN, INPUT);
  pinMode(LD2410_PIN, INPUT);
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);

  stepper.setMaxSpeed(1000.0);
  stepper.setSpeed(FAN_MED);
}

void loop() {
  readCommandFromSerial();
  emitDataToSerial();

  if (isLightOn) {
    turnOnLight();
  } else {
    turnOffLight();
  }

  if (isFanOn) {
    runFan();
  }
}

void readCommandFromSerial() {
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');

    if (command == "ON_LIGHT") {
      isLightOn = true;
    }

    if (command == "OFF_LIGHT") {
      isLightOn = false;
    }

    if (command == "ON_FAN") {
      isFanOn = true;
    }

    if (command == "OFF_FAN") {
      isFanOn = false;
    }

    if (command.startsWith("LIGHT")) {
      String colors = command.substring(5);
      setLightColor(colors);
    }

    if (command.startsWith("FAN")) {
      String speed = command.substring(3);
      setFanSpeed(speed);
    }
  }
}

void emitDataToSerial() {
  currentMillis = millis();
  if (currentMillis - previousMillis >= SENSOR_EMIT_INTERVAL) {
    temp_adc_val = analogRead(LM35_PIN);
    temp_val = (temp_adc_val * 0.488);  // adc / 1023 * 5 (V) *
100(degC/V)
    Serial.print("TMP");
    Serial.println(temp_val);

    presence_val = digitalRead(LD2410_PIN);
    Serial.print("PRS");
    Serial.println(presence_val);

    previousMillis = currentMillis;
  }
}

void turnOnLight() {
  analogWrite(RED_PIN, red);
  analogWrite(GREEN_PIN, green);
  analogWrite(BLUE_PIN, blue);
}

void turnOffLight() {
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
}

void setLightColor(String colorString) {
  int searchIndex = 0;
  int commaIndex;
  String colorComponent;

  int colors[3] = { 255, 255, 255 };
  for (int i = 0; i < 3; i++) {
    commaIndex = colorString.indexOf(",", searchIndex);
    colorComponent = colorString.substring(searchIndex, commaIndex);
    colors[i] = colorComponent.toInt();
    searchIndex = commaIndex + 1;
  }

  red = colors[0];
  green = colors[1];
  blue = colors[2];
}

void runFan() {
  stepper.runSpeed();
}

void setFanSpeed(String speed) {
  if (speed == "LOW") {
    stepper.setSpeed(FAN_LOW);
  } else if (speed == "MED") {
    stepper.setSpeed(FAN_MED);
  } else if (speed == "HIGH") {
    stepper.setSpeed(FAN_HIGH);
  }
}
```

read_from_arduino.py

```
import serial
import MySQLdb
import time
from datetime import datetime
```

```
connection = MySQLdb.connect("localhost", "admin", "admin",
"assignment_2", autocommit=True) or die("Could not connect to
database.")
cursor = connection.cursor()
device = "/dev/ttyUSB0"
arduino = serial.Serial(device, 9600)

lastRefreshTime = 0
refreshInterval = 1000

lightManual = False
lightTmpThreshold = 30
lightOn = True
red = 255
green = 255
blue = 255
fanManual = False
mediumTmpThreshold = 20
highTmpThreshold = 25
fanOn = True
fanSpeed = "MED"

detectedPresenceSince = 0
detectingPresence = False

while True:
    try:
        currentTimeMs = round(time.time() * 1000)
        if (currentTimeMs - lastRefreshTime >= refreshInterval):
            cursor.execute("SELECT * FROM DeviceSettings WHERE
SettingID = 1")
            setting = cursor.fetchall()[0]
            print(setting)

            lightManual = setting[1]
            lightTmpThreshold = setting[2]
            lightOn = setting[3]
            red = setting[4]
            green = setting[5]
            blue = setting[6]
            fanManual = setting[7]
            mediumTmpThreshold = setting[8]
            highTmpThreshold = setting[9]
            fanOn = setting[10]
            fanSpeed = setting[11]

            if (lightManual):
                if (lightOn):
                    arduino.write(b"ON_LIGHT\n")
                else:
                    arduino.write(b"OFF_LIGHT\n")
                arduino.write(("LIGHT%d,%d,%d\n" % (red, green, 
blue)).encode("utf-8"))

            if (fanManual):
                if (fanOn):
                    arduino.write(b"ON_FAN\n")
                else:
                    arduino.write(b"OFF_FAN\n")
                arduino.write(("FAN%s\n" % (fanSpeed)).encode("utf-8"))

            lastRefreshTime = currentTimeMs

        now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        data = arduino.readline().decode("utf-8").strip()
        print(data)
        prefix = data[:3]

        if (prefix == "TMP"):
            # Handle temperature reading.
            temperature = data[3:]
            print(temperature)
            query = "INSERT INTO TemperatureReading (`Time`, `Value`) 
VALUES ('%s', %s)" % (now, temperature)
            cursor.execute(query)

            if (not lightManual):
                if (float(temperature) >= lightTmpThreshold):
                    arduino.write(b"LIGHT255,255,255\n")
                    cursor.execute("UPDATE DeviceSettings SET Red = 
255, Green = 255, Blue = 255 WHERE SettingID = 1")
                else:
                    arduino.write(b"LIGHT175,45,0\n")
                    cursor.execute("UPDATE DeviceSettings SET Red = 
175, Green = 45, Blue = 0 WHERE SettingID = 1")
```

```
            if (not fanManual):
                if (float(temperature) >= highTmpThreshold):
                    arduino.write(b"FANHIGH\n")
                    cursor.execute("UPDATE DeviceSettings SET FanSpeed
= 'HIGH' WHERE SettingID = 1")
                elif (float(temperature) >= mediumTmpThreshold):
                    arduino.write(b"FANMED\n")
                    cursor.execute("UPDATE DeviceSettings SET FanSpeed
= 'MED' WHERE SettingID = 1")
                else:
                    arduino.write(b"FANLOW\n")
                    cursor.execute("UPDATE DeviceSettings SET FanSpeed
= 'LOW' WHERE SettingID = 1")
        elif (prefix == "PRS"):
            # Handle human presence reading.
            presence = int(data[3:])
            print(presence)

            if (presence & (not detectingPresence)):
                detectedPresenceSince = datetime.now()
                detectingPresence = True

                if (not lightManual):
                    arduino.write(b"ON_LIGHT\n")
                    cursor.execute("UPDATE DeviceSettings SET LightOn =
1 WHERE SettingID = 1")
                if (not fanManual):
                    arduino.write(b"ON_FAN\n")
                    cursor.execute("UPDATE DeviceSettings SET FanOn = 1
WHERE SettingID = 1")

            if ((not presence) & detectingPresence):
                detectingPresence = False
                fromTime = detectedPresenceSince.strftime("%Y-%m-%d
%H:%M:%S")
                toTime = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                query = "INSERT INTO HumanPresenceReading (`FromTime`,
`ToTime`) VALUES ('%s', '%s')" % (fromTime, toTime)
                cursor.execute(query)

                if (not lightManual):
                    arduino.write(b"OFF_LIGHT\n")
                    cursor.execute("UPDATE DeviceSettings SET LightOn =
0 WHERE SettingID = 1")
                if (not fanManual):
                    arduino.write(b"OFF_FAN\n")
                    cursor.execute("UPDATE DeviceSettings SET FanOn = 0
WHERE SettingID = 1")

    except (KeyboardInterrupt, serial.serialutil.SerialException) as
error:
        if (type(error) is KeyboardInterrupt):
            print("Ctrl + C detected.")
        else:
            print("The serial port was disconnected.")

        if (detectingPresence):
            fromTime = detectedPresenceSince.strftime("%Y-%m-%d
%H:%M:%S")
            toTime = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            query = "INSERT INTO HumanPresenceReading (`FromTime`,
`ToTime`) VALUES ('%s', '%s')" % (fromTime, toTime)
            cursor.execute(query)

        print("Closing edge server...")
        connection.commit()
        cursor.close()
        break
```

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Smart Home Dashboard</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <link rel="stylesheet" href="index.css">

    <script src="//d3js.org/d3.v6.min.js"></script>
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.m
in.css" rel="stylesheet"

integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhj
Y6hW+ALEwIH" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bun
dle.min.js"

integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK
1eN7N6jIeHz"
        crossorigin="anonymous"></script>
</head>

<body>
    <h1 class="mx-5 my-4">Welcome to your Smart Home!</h1>

    <h2 class="mx-5 my-4">Realtime Temperature Data</h2>
    <div id="chart"></div>
    <p id="tempSummary" class="mx-5 my-4 fs-3">Min: 0; Max: 0; Mean: 0
(degrees Celcius)</p>

    <h2 class="mx-5 my-4">Recent Human Presence Data</h2>
    <ul id="presenceEventList" class="mx-5 my-4">
        <li>No data yet.</li>
    </ul>

    <h2 class="mx-5 my-4">Device settings and controls</h2>
    <div id="form" class="mx-5 my-4">
        <form @submit="handleSubmit">
            <button type="button" @click="getSettingsData" class="btn
btn-secondary">Refresh</button>

            <fieldset class="mx-5 my-4">
                <legend>Light</legend>
                <label class="d-block">
                    <input type="checkbox" name="lightManual"
v-model="lightManual"> Manually control light
                </label>
                <label class="d-block">
                    Temperature threshold to change color:
```

```
                    <input type="number" name="lightTmpThreshold"
v-model="lightTmpThreshold" step=".01">
                </label>
                <label class="d-block">
                    <input type="checkbox" name="lightOn"
v-model="lightOn"> On
                </label>
                <p class="fw-bold">Color:</p>
                <label class="d-block">
                    <input type="range" name="red" min="0" max="255"
v-model="red"> Red ({{red}})
                </label>
                <label class="d-block">
                    <input type="range" name="green" min="0" max="255"
v-model="green"> Green ({{green}})
                </label>
                <label class="d-block">
                    <input type="range" name="blue" min="0" max="255"
v-model="blue"> Blue ({{blue}})
                </label>
            </fieldset>

            <fieldset class="mx-5 my-4">
                <legend>Fan</legend>
                <label class="d-block">
                    <input type="checkbox" name="fanManual"
v-model="fanManual"> Manually control fan
                </label>
                <label class="d-block">
                    Temperature threshold for Medium speed: <input
type="number" name="mediumTmpThreshold"
                        v-model="mediumTmpThreshold" step=".01">
                </label>
                <label class="d-block">
                    Temperature threshold for High speed: <input
type="number" name="highTmpThreshold"
                        v-model="highTmpThreshold" step=".01">
                </label>
                <label class="d-block">
                    <input type="checkbox" name="fanOn"
v-model="fanOn"> On
                </label>
                <p class="fw-bold">Fan speed:</p>
                <label class="d-block">
                    <input type="radio" name="fanSpeed" value="LOW"
v-model="fanSpeed"> Low
                </label>
                <label class="d-block">
                    <input type="radio" name="fanSpeed" value="MED"
v-model="fanSpeed"> Medium
                </label>
                <label class="d-block">
                    <input type="radio" name="fanSpeed" value="HIGH"
v-model="fanSpeed"> High
                </label>
            </fieldset>

            <button type="submit" class="btn btn-primary">Update
settings</button>
        </form>
    </div>

    <script src="charts.js"></script>
    <script src="index.js"></script>

    <script>
        Vue.createApp({
            data() {
                return {
                    lightManual: false,
                    lightTmpThreshold: 30.0,
                    lightOn: false,
                    red: 255,
                    green: 255,
                    blue: 255,
                    fanManual: false,
                    mediumTmpThreshold: 20,
                    highTmpThreshold: 25,
                    fanOn: false,
                    fanSpeed: "LOW"
                }
            },
            methods: {
                async handleSubmit(e) {
                    e.preventDefault();
                    console.log(this.$data);

                    try {
                        const options = {
                            method: "PUT",
                            headers: {
                                "Content-Type": "application/json"
                            },
                            body: JSON.stringify({
                                lightManual: this.lightManual ? 1 : 0,
                                lightTmpThreshold:
this.lightTmpThreshold,
                                lightOn: this.lightOn ? 1 : 0,
                                red: this.red,
                                green: this.green,
                                blue: this.blue,
                                fanManual: this.fanManual ? 1 : 0,
                                mediumTmpThreshold:
this.mediumTmpThreshold,
                                highTmpThreshold:
this.highTmpThreshold,
                                fanOn: this.fanOn ? 1 : 0,
                                fanSpeed: this.fanSpeed
                            })
                        };

                        const res = await
fetch("/assignment_2/device_settings.php", options);
                        alert("Settings updated!");
                    } catch (error) {
                        console.log(error);
                    }
                },
                async getSettingsData() {
                    try {
                        const res = await
fetch("/assignment_2/device_settings.php");
                        const settings = await res.json();
                        console.log(settings);

                        this.lightManual = !!+settings["LightManual"];
                        this.lightTmpThreshold =
settings["LightTmpThreshold"];
```

```
                        this.lightOn = !!+settings["LightOn"];
                        this.red = settings["Red"];
                        this.green = settings["Green"];
                        this.blue = settings["Blue"];
                        this.fanManual = !!+settings["FanManual"];
                        this.mediumTmpThreshold =
settings["MediumTmpThreshold"];
                        this.highTmpThreshold =
settings["HighTmpThreshold"];
                        this.fanOn = !!+settings["FanOn"];
                        this.fanSpeed = settings["FanSpeed"];
                    } catch (error) {
                        console.log(error);
                    }
                }
            },
            mounted: function() {
                this.getSettingsData();

            }
        }).mount("#form");
    </script>
</body>

</html>
```

index.js

```
drawLineChart("#chart", []);
const temperatureSummary = document.querySelector("#tempSummary");
const presenceEventList = document.querySelector("#presenceEventList");

setInterval(async () => {
    try {
        const response = await fetch("/assignment_2/temperature.php");
        const rawData = await response.json();

        let chartData = rawData.map(record => {
            return {
                time: new Date(record["Time"]),
                value: record["Value"]
            };
        });
        const latestReadingTime = d3.max(chartData, d => d.time);

        chartData = chartData.filter(record =>
latestReadingTime.getTime() - record.time.getTime() <= 20000);

        drawLineChart("#chart", chartData);

        const min = d3.min(chartData, d => d.value);
        const max = d3.max(chartData, d => d.value);
        const mean = d3.mean(chartData, d => d.value);
        temperatureSummary.textContent = `Min: ${min}; Max: ${max};
Mean: ${mean.toFixed(2)} (degrees Celcius)`;
    } catch (error) {
        console.log(error);
    }
}, 1000);

setInterval(async () => {
    try {
        const response = await
fetch("/assignment_2/human_presence.php");
        const rawData = await response.json();

        presenceEventList.innerHTML = "";
        if (rawData.length === 0) {
            presenceEventList.innerHTML = "<li>No one has been in your
house recently.</li>";
        } else {
            rawData.forEach(record => {
                const li = document.createElement("li");
                const from = new Date(record["FromTime"]);
                const to = new Date(record["ToTime"]);
                const durationMs = to.getTime() - from.getTime();
                li.textContent = `${msToHMS(durationMs)} from
${from.toLocaleString()} to ${to.toLocaleString()}.`;
                presenceEventList.appendChild(li);
            });
        }
    } catch (error) {
        console.log(error);
    }
}, 1000);
```

```
function msToHMS(ms) {
    let seconds = Math.floor(ms / 1000);
    let hours = Math.floor(seconds / 3600);
    seconds = seconds % 3600;
    let minutes = Math.floor(seconds / 60);
    seconds = seconds % 60;

    return `${hours} hour(s), ${minutes} minute(s), ${seconds}
second(s)`;
}
```

temperature.php

```
<?php

$method = $_SERVER['REQUEST_METHOD'];
if ($method != "GET") {
    http_response_code(404);
    exit();
}

$connection= mysqli_connect('localhost', 'admin', 'admin',
'assignment_2');
$table = "TemperatureReading";

$sql = "SELECT * FROM `$table` ORDER BY `Time` DESC LIMIT 20";

$result = mysqli_query($connection, $sql);
if ($result) {
    header('Content-Type: application/json');
    echo '[';
    for ($i = 0; $i < mysqli_num_rows($result); $i++) {
        echo ($i > 0 ? ',' : '') .
json_encode(mysqli_fetch_object($result));
    }
    echo ']';
}

mysqli_close($connection);
```

human_presence.php

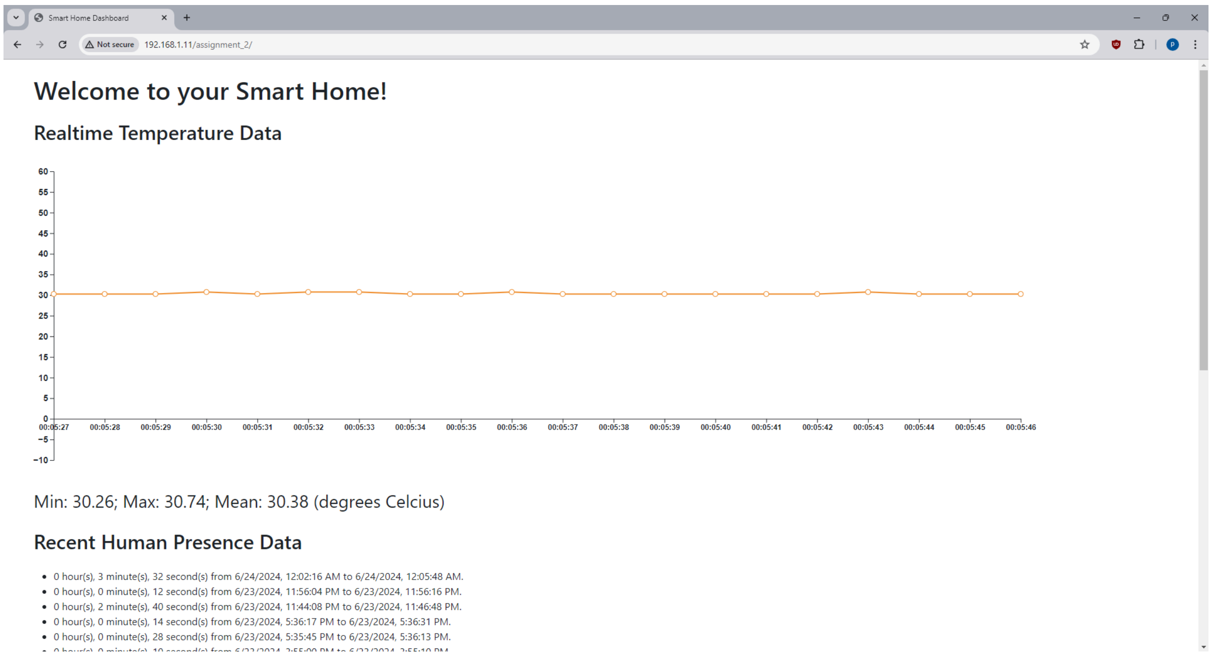
```php
<?php

$method = $_SERVER['REQUEST_METHOD'];
if ($method != "GET") {
    http_response_code(404);
    exit();
}

$connection= mysqli_connect('localhost', 'admin', 'admin',
'assignment_2');
$table = "HumanPresenceReading";

$sql = "SELECT * FROM `$table` ORDER BY `FromTime` DESC LIMIT 10";

$result = mysqli_query($connection, $sql);
if ($result) {
    header('Content-Type: application/json');
    echo '[';
    for ($i = 0; $i < mysqli_num_rows($result); $i++) {
        echo ($i > 0 ? ',' : '') .
json_encode(mysqli_fetch_object($result));
    }
    echo ']';
}

mysqli_close($connection);
```

device_settings.php

```php
<?php

$method = $_SERVER['REQUEST_METHOD'];
$request = explode('/', trim($_SERVER['PATH_INFO'], '/'));
$input = json_decode(file_get_contents('php://input'), true);

if ($method != "GET" && $method != "PUT") {
    http_response_code(404);
    exit();
}
```

```
$connection= mysqli_connect('localhost', 'admin', 'admin',
'assignment_2');
$table = "DeviceSettings";

if (isset($input)) {
    $lightManual = mysqli_escape_string($connection,
$input["lightManual"]);
    $lightTmpThreshold = mysqli_escape_string($connection,
$input["lightTmpThreshold"]);
    $lightOn = mysqli_escape_string($connection, $input["lightOn"]);
    $red = mysqli_escape_string($connection, $input["red"]);
    $green = mysqli_escape_string($connection, $input["green"]);
    $blue = mysqli_escape_string($connection, $input["blue"]);
    $fanManual = mysqli_escape_string($connection,
$input["fanManual"]);
    $mediumTmpThreshold = mysqli_escape_string($connection,
$input["mediumTmpThreshold"]);
    $highTmpThreshold = mysqli_escape_string($connection,
$input["highTmpThreshold"]);
    $fanOn = mysqli_escape_string($connection, $input["fanOn"]);
    $fanSpeed = mysqli_escape_string($connection, $input["fanSpeed"]);
}

if ($method == "GET") {
    $sql = "SELECT * FROM `$table` WHERE SettingID = 1";
} else {
    $sql = "UPDATE DeviceSettings SET LightManual = '$lightManual',
LightTmpThreshold = '$lightTmpThreshold', LightOn = '$lightOn', Red =
'$red', Green = '$green', Blue = '$blue', FanManual = '$fanManual',
MediumTmpThreshold = '$mediumTmpThreshold', HighTmpThreshold =
'$highTmpThreshold', FanOn = '$fanOn', FanSpeed = '$fanSpeed'";
}

$result = mysqli_query($connection, $sql);
if ($result) {
    header('Content-Type: application/json');
    if ($method == "GET") {
        for ($i = 0; $i < mysqli_num_rows($result); $i++) {
            echo ($i > 0 ? ',' : '') .
json_encode(mysqli_fetch_object($result));
        }
    } else {
        echo json_encode("Updated successfully!");
        }
}

mysqli_close($connection);
```

Screenshots



Welcome to your Smart Home!

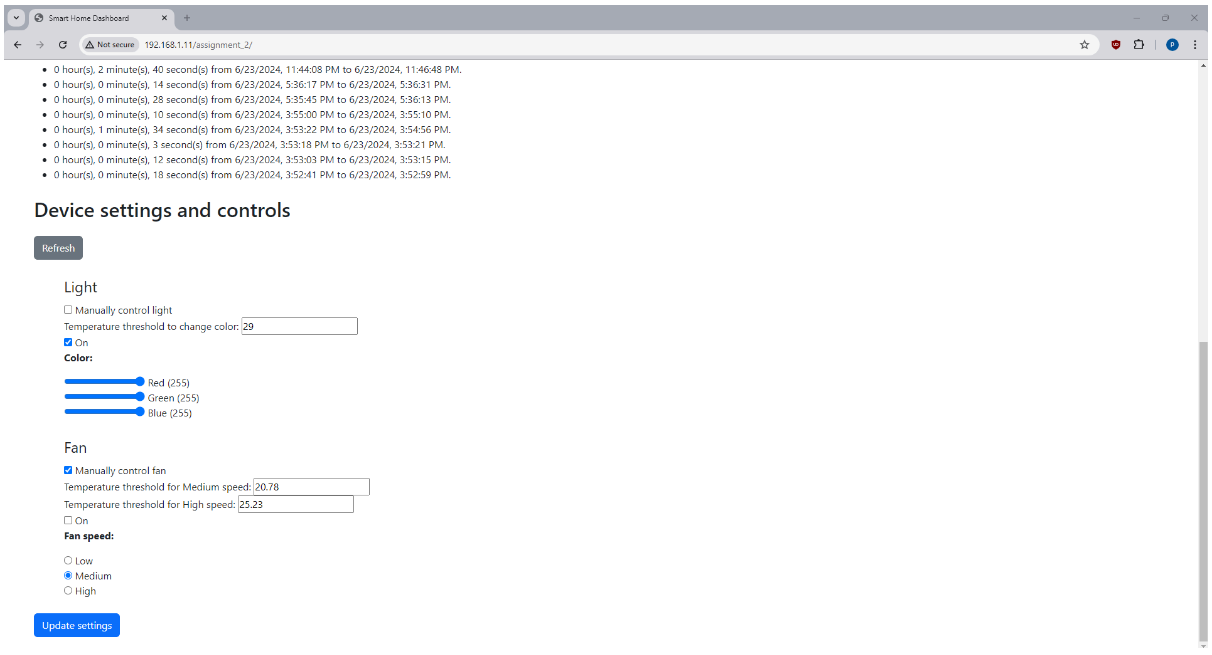
Realtime Temperature Data

Min: 30.26; Max: 30.74; Mean: 30.38 (degrees Celcius)

Recent Human Presence Data

- 0 hour(s), 3 minute(s), 32 second(s) from 6/24/2024, 12:02:16 AM to 6/24/2024, 12:05:48 AM.
- 0 hour(s), 0 minute(s), 12 second(s) from 6/23/2024, 11:56:04 PM to 6/23/2024, 11:56:16 PM.
- 0 hour(s), 2 minute(s), 40 second(s) from 6/23/2024, 11:44:08 PM to 6/23/2024, 11:46:48 PM.
- 0 hour(s), 0 minute(s), 14 second(s) from 6/23/2024, 5:36:17 PM to 6/23/2024, 5:36:31 PM.
- 0 hour(s), 0 minute(s), 28 second(s) from 6/23/2024, 5:35:45 PM to 6/23/2024, 5:36:13 PM.
- 0 hour(s), 0 minute(s), 10 second(s) from 6/23/2024, 3:55:00 PM to 6/23/2024, 3:55:10 PM.

Fig. 1: Web interface (1)



- 0 hour(s), 2 minute(s), 40 second(s) from 6/23/2024, 11:44:08 PM to 6/23/2024, 11:46:48 PM.
- 0 hour(s), 0 minute(s), 14 second(s) from 6/23/2024, 5:36:17 PM to 6/23/2024, 5:36:31 PM.
- 0 hour(s), 0 minute(s), 28 second(s) from 6/23/2024, 5:35:45 PM to 6/23/2024, 5:36:13 PM.
- 0 hour(s), 0 minute(s), 10 second(s) from 6/23/2024, 3:55:00 PM to 6/23/2024, 3:55:10 PM.
- 0 hour(s), 1 minute(s), 34 second(s) from 6/23/2024, 3:53:22 PM to 6/23/2024, 3:54:56 PM.
- 0 hour(s), 0 minute(s), 3 second(s) from 6/23/2024, 3:53:18 PM to 6/23/2024, 3:53:21 PM.
- 0 hour(s), 0 minute(s), 12 second(s) from 6/23/2024, 3:53:03 PM to 6/23/2024, 3:53:15 PM.
- 0 hour(s), 0 minute(s), 18 second(s) from 6/23/2024, 3:52:41 PM to 6/23/2024, 3:52:59 PM.

Device settings and controls

Light

☐ Manually control light
Temperature threshold to change color: 29
☑ On
Color:
Red (255)
Green (255)
Blue (255)

Fan

☑ Manually control fan
Temperature threshold for Medium speed: 20.78
Temperature threshold for High speed: 25.23
☐ On
Fan speed:
○ Low
● Medium
○ High

Fig. 2: Web interface (2)