Smart Light Design Report

SWINBURNE UNIVERSITY OF TECHNOLOGY

Bachelor of Computer Science

IoT Programming (COS30011)

May 15, 2022

GROUP MEMBERS

Name	Student ID	Student Email
Phu Danh Pham	102619398	102619398@student.swin.edu.au
Minh Le Anh Nguyen	103137309	103137309@student.swin.edu.au
Zachary Stanford	102496096	102496096@student.swin.edu.au

SWE30011, IoT PROGRAMING, SEMESTER 1 2022

Table of Contents

I. Project Summary

1.1. Topic background

One of the most popular themes in modern technology is automation. People utilise automated gadgets in their daily activities for a variety of reasons, ranging from safety to convenience. Automation systems were once solely available to businesses due to their high cost; but, as technology advances, automation has become increasingly accessible to everyone. Home automation systems are becoming increasingly popular across the world. The automated electric system will also save energy by shutting off equipment when they are not in use. Furthermore, automated devices may be readily operated from afar using simple software such as web-based or mobile apps. As a result, advancements in automation not only improve everyone's quality of life but also provide people more control over their own property at home.

1.2. Proposed System

In this report, an automatic fan will be developed using Arduino Uno boards along with a light sensor and motion detection sensor. Raspberry Pi will be used for the edge server with Raspbian (a Debian-based operating system) running on personal laptops via Oracle VM VirtualBox (a cross-platform virtualization software). Besides, a web-based application written in Python Flask would also be developed to remotely control the device.

The device includes 3 separate nodes: motion sensor, light sensor, and the actuator - RGB LED. Light sensors and motion sensors will constantly send sensor data using MQTT protocol to the RGB LED. If any motion is detected, which means there is someone in the room, the light level value will be used to determine the R, G, B values of the LED and turn the LED on with the respective color. If the light level is high enough, the LED will turn off; otherwise, it will turn on.

After the color is determined, all data including light level, motion detected (yes or no), and RGB value will be pushed as one record with the current timestamp to the PostgreSQL database on cloud (AWS).

The web user interface fetches data from the database in the cloud server and displays them in the form of a table. The interface allows users to turn on/off the LED manually. Moreover, users can choose a wanted color for the LED by entering the respective R, G, B values in text boxes to change the LED color. By controlling the RGB LED on this website, the automation of the device will be temporarily turned off.

a. Sensors:

- Motion Sensor: senses if there is any movement in the room (if there is someone in the room). If yes, the current light level will be used to determine the color for the RGB LED, and turn it on. Otherwise, the LED will turn off. Motion is checked each 5 minutes.

- Light Sensor: detects the current light level in the room. The light level will be sent to the edge server of the LED and used to determine the color of the LED. If it is too high, the LED will also be turned off even if there is still someone in the room (motion is detected).

b. Actuators:

- RGB LED: will be turned on/off depending on the current light level and motion detected. The current light level will decide the color and it can be changed while the LED is still on. If users control the RGB LED via web interface. The automation will turn off.

- LED light: installed on the motion sensor node. The LED will turn on when any motion is detected and turn off otherwise.

c. Servers and User Interface:

- Edge Server: contains conditional for each IoT node: light sensor, motion sensor, and RGB LED. The servers communicate with each other using MQTT protocol. Data containing RGB value, light level, and motion status is pushed to the cloud database after sensor data from sensors being sent to the RGB LED.

- Web Server and Interface: Web Server works as the broker for MQTT communication and hosts the user interface. The interface displays data fetched from the cloud database and provides users buttons to control the RGB LED. Upon button pressed, the respective command will be sent to the edge server of the LED and be processed to control it.

- Cloud Server: Amazon Web Services (AWS) is facilitated to store the device data for retrieval and further processing. The cloud server communicates with other servers over the internet.

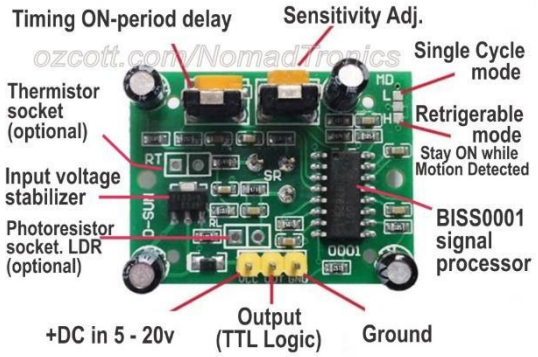
II. Conceptual Design and System Implementation

1. Hardware Components

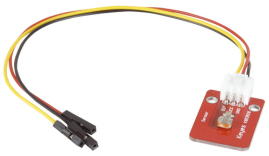
1.1. IoT Node:

The automatic fan device consists of these hardware components and their specifications:

- 3 x Duinotech UNO Arduino-Compatible Board
- 3 x Breadboard
- PIR Motion Detector Module:
    - Sensor Output: Digital
    - Required Supply Voltage: 5 – 20V



- Arduino Compatible Photosensitive LDR Sensor Module
    - Sensor Output: Analog
    - Required Supply Voltage: 3V



- 1 xTricolour RGB 5mm LED 600-1000mcd Round Diffused
    - Colour: Red, Green, and Blue with common cathode
    - • Size: 5mm
    - • Lens: Diffused
    - • Viewing Angle: 45 degrees
    - • Wavelength: Red 625nm, Green 520nm, Blue 470nm
    - • Typical Forward Current: 20mA
    - • Typical Forward Voltage: Red 2.0V, Green/Blue 3.5V
    - • Luminous Intensity: Red 600mcd, Green 1000mcd, Blue 200mcd
    - • Continuous Maximum Current: 30mA
- 4 x Resistors

2. Software Components

2.1. Programming Languages and Frameworks:

- Python and Flask: Python is used to facilitate the automation of the device and Flask web framework is used to build the web-based application for controlling the device remotely and manually.

- HTML and CSS: used to build and style the web app.

- Arduino: used to write working sketches for IoT nodes

- AWS: Used to host a cloud PostgreSQL database for data storage and retrieval

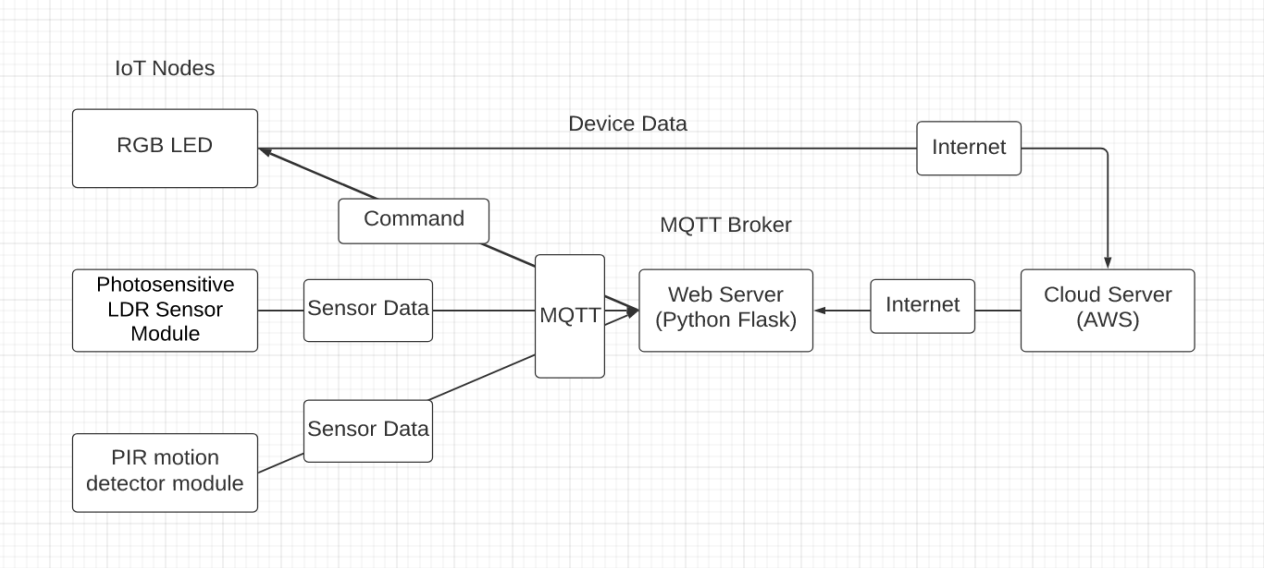
2.2. Hypervisor

- Oracle VirtualBox VM Manager: used to create and run the Edge Server on Linux version Debian-64 bit.

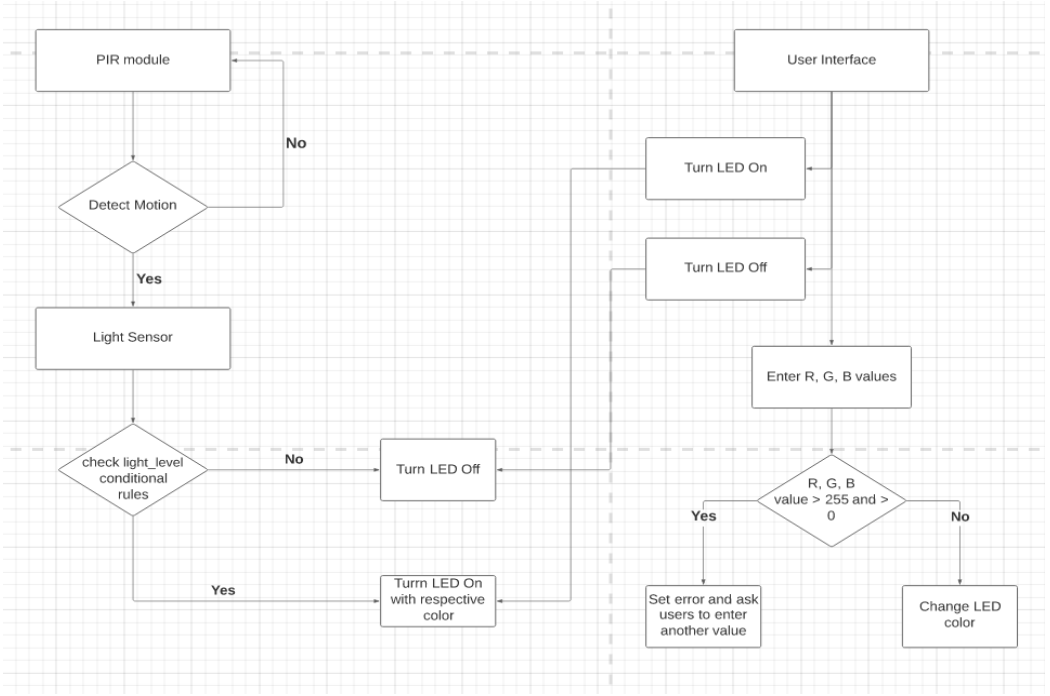
2.3. Communication Protocols

- MQTT: Used to send commands from the Web Interface to the Edge Device
- PostgreSQL: Used to store the data collected from the sensors, the data was then retrieved and displayed by the Web Interface
- Serial Communication: Used to transfer data from the Edge Device to the Arduino

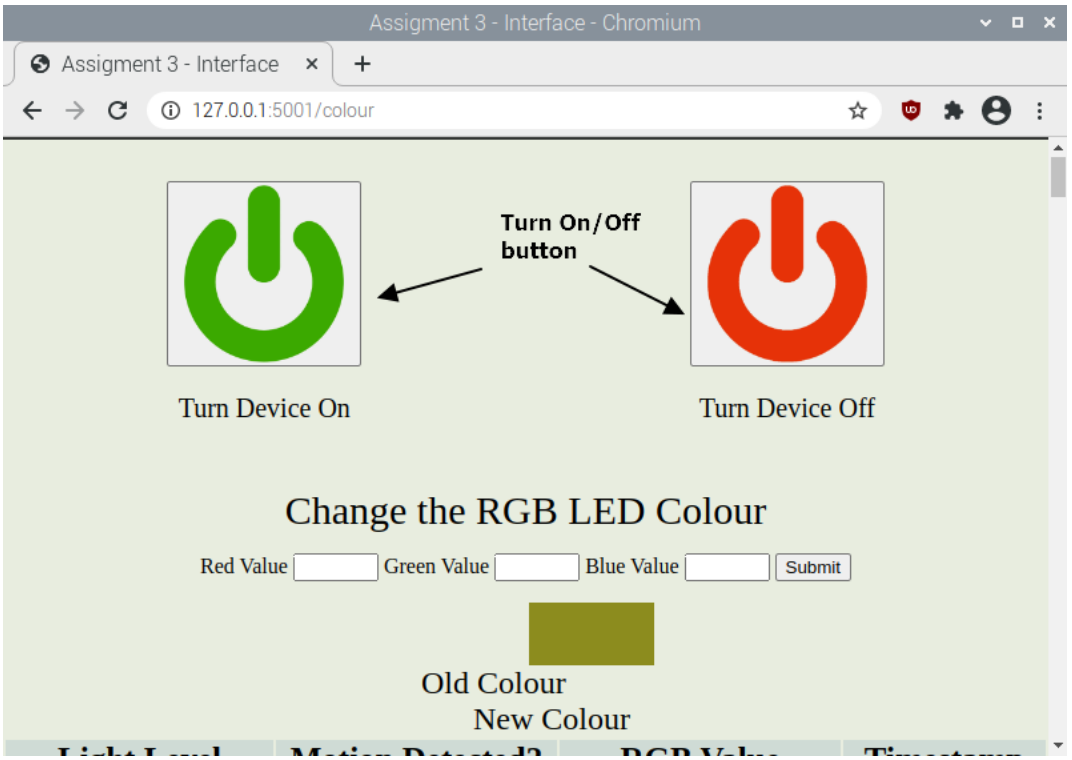
3. Block Diagram



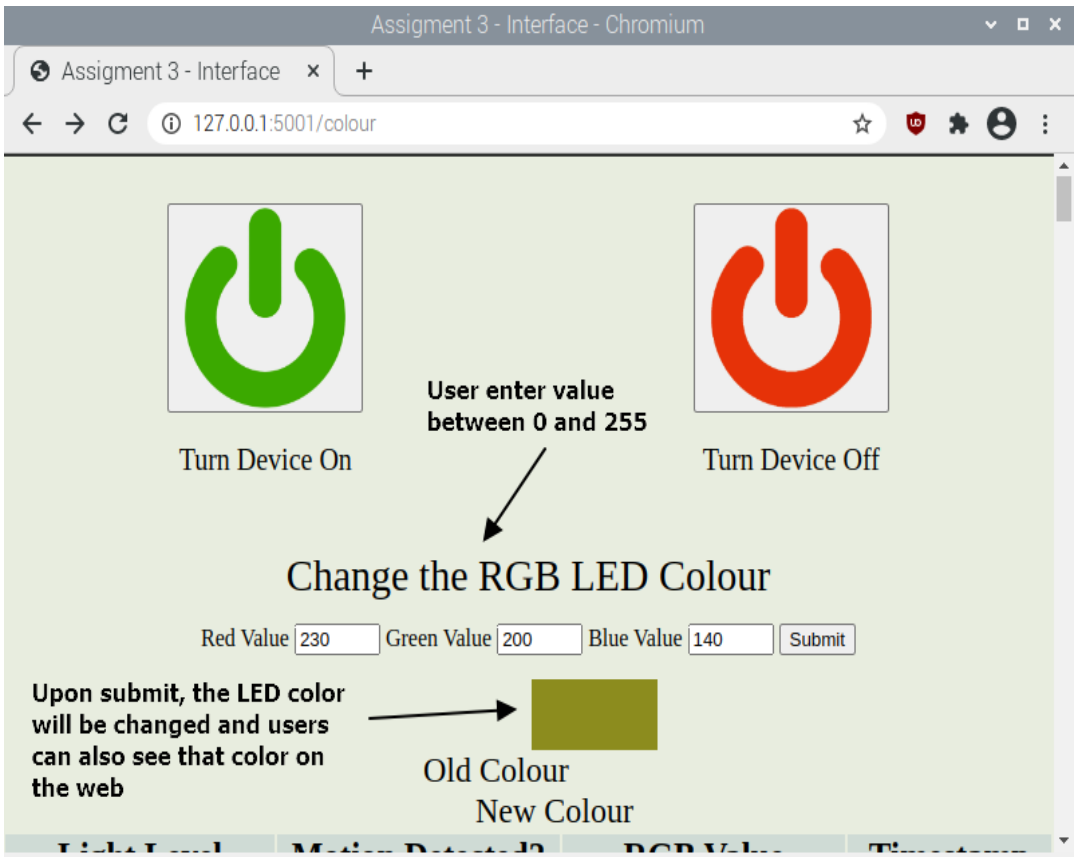
4. UML Diagram:



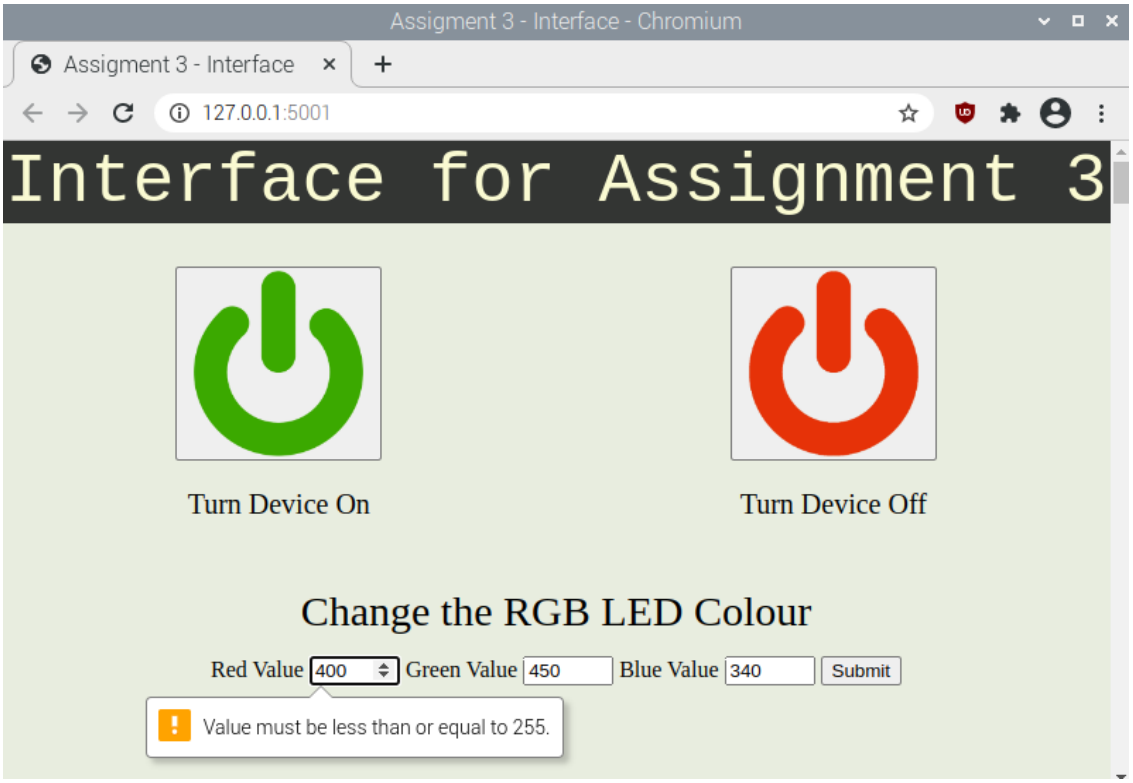
5. User Manual



Turn On/Off buttons



Users change RGB values by entering values from 0 to 255 to text boxes



Error shows up when entering value not in range

6. Implementing

Light-Sensor Code:

Arduino Sketch:

Light_sensor

```
#define light_sensor 10
void setup() {
  // put your setup code here, to run once:
  pinMode(light_sensor,INPUT);
  Serial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:
  int light_value = analogRead(light_sensor);
  Serial.println(light_value);
  delay(1000);
}
```

Arduino code for analog reading from light sensor

Python Script:

```
import serial
import pymysql
import time
import paho.mqtt.client as mqtt
import json
import requests

device ="/dev/ttyS0"
arduino = serial.Serial(device,9600)

port = 1883
sensor_data = {'light_level' : 0}
client = mqtt.Client()
client.connect("172.20.10.5")
client.loop_start()
try:
    while True:
        data = arduino.readline()
        sensor_data['light_level'] = data
        client.publish('/device/light', json.dumps(sensor_data), 1)
        time.sleep(3)
except KeyboardInterrupt
    pass
client.loop_stop()
client.disconnect()
```

Python code for receiving data via serial communication then publishing the topic to the cloud

Motion-detect sensor Code:

Arduino Sketch:

Verify

Sensor_motion

```
#define red_LED 2
#define motion_sensor 6


void setup() {
  // put your setup code here, to run once:
  pinMode(motion_sensor,INPUT);
  pinMode(red_LED,OUTPUT);
  Serial.begin(9600);
  cli(); // Stop interrupts for till we make the settings
  // reset the control register to make sure starting with everything disabled
  TCCR1A = 0;
  TCCR1B = 0;
  // set prescalar
  TCCR1B |= B00000100;
  // enable compare match mode on register A
  TIMSK1 |= B00000010;
  // set the value of register A to 31250
  OCR1A = 65535;
  // enable back the interrupts
  sei();
}
// this IRS triggers each 1000ms
ISR(TIMER1_COMPA_vect){
  TCNT1 = 0;
  int motion_value = digitalRead(6);
  Serial.println(motion_value);
//  Serial.print(" ");
//  Serial.println(light_value);
  }

void loop() {
  // put your main code here, to run repeatedly:
  int motion_value = digitalRead(motion_sensor);
    if(motion_value==HIGH){
        digitalWrite(red_LED,HIGH);
      }
    else{
        digitalWrite(red_LED,LOW);
      }
}
```

Arduino code for digital reading from motion-detect sensor and turning on/off LED based on conditions

Python script:

```
import serial
import pymysql
import time
import paho.mqtt.client as mqtt
import json
import requests

device ="/dev/ttyS0"
arduino = serial.Serial(device,9600)

port = 1883
sensor_data = {'detect_motion' : 0}
client = mqtt.Client()
client.connect("172.20.10.5")
client.loop_start()
try:
    while True:
        data = arduino.readline()
        sensor_data['detect_motion'] = data
        client.publish('/device/motion', json.dumps(sensor_data), 1)
        time.sleep(3)
except KeyboardInterrupt
    pass
client.loop_stop()
client.disconnect()
```

Python code for receiving data via serial communication then publishing the topic to the cloud

RGB LED:

Arduino Sketch:

```
int val;
int redPin = 9;
int greenPin = 10;
int bluePin = 11;
int redIntensity = 200;
int greenIntensity = 200;
int blueIntensity = 200;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  if (Serial.available()) {
    String value = Serial.readStringUntil('\n');
    Serial.println(value);

    if (value.indexOf("custom") == -1) {
        if (value.equals("0")) {
          redIntensity = 0;
          blueIntensity = 0;
          greenIntensity = 0;
        }

        if (value.equals("1")) {
          redIntensity = 255;
          blueIntensity = 0;
          greenIntensity = 0;
        }
```

```
//Values from 2 to 7 is for automation only
        if (value.equals("2")) {
          redIntensity = 0;
          blueIntensity = 255;
          greenIntensity = 0;
        }

        if (value.equals("3")) {
          redIntensity = 0;
          blueIntensity = 0;
          greenIntensity = 255;
        }

        if (value.equals("4")) {
          redIntensity = 255;
          blueIntensity = 0;
          greenIntensity = 255;
        }

        if (value.equals("5")) {
          redIntensity = 255;
          blueIntensity = 255;
          greenIntensity = 0;
        }

        if (value.equals("6")) {
          redIntensity = 0;
          blueIntensity = 255;
          greenIntensity = 255;
        }

        if (value.equals("7")) {
          redIntensity = 255;
          blueIntensity = 255;
```

```
      if (value.equals("7")) {
        redIntensity = 255;
        blueIntensity = 255;
        greenIntensity = 255;
      }
    } else {
      //user change intensity of red, green, blue
      //serial order in form of: "custom,red: {int},green: {int},blue: {int}"
      //example: "custom,red: 100,green: 140,blue: 50"
      String str1 = value.substring(value.indexOf(',')+1, value.length()); //red: 100,green: 140,blue: 50
      String redOrder = str1.substring(0, str1.indexOf(',')); //red: 100
      redIntensity = redOrder.substring(redOrder.indexOf(' '), redOrder.length()).toInt(); // redIntensity = 100
      //Serial.println(redIntensity);

      String str2 = str1.substring(str1.indexOf(',')+1, str1.length());
      String greenOrder = str2.substring(0, str2.indexOf(',')); //green: 140
      greenIntensity = greenOrder.substring(greenOrder.indexOf(' '), greenOrder.length()).toInt();
      //Serial.println(greenIntensity);

      String str3 = str2.substring(str2.indexOf(',')+1, str2.length());
      String blueOrder = str3.substring(0, str3.indexOf(',')); //blue: 50
      blueIntensity = blueOrder.substring(blueOrder.indexOf(' '), blueOrder.length()).toInt();

      //Serial.println(blueIntensity);
    }

    analogWrite(redPin, redIntensity);
    analogWrite(bluePin, blueIntensity);
    analogWrite(greenPin, greenIntensity);
  }

}
```

Python Script:

```
import paho.mqtt.publish as publish
import paho.mqtt.client as client
import serial
import json
import os
import time
from datetime import datetime
import psycopg2
import random

ON = True
device = '/dev/ttyS0'

database = "arduino-1.c9dhukv1umdc.us-west-1.rds.amazonaws.com"
port = 1883
broker = "172.20.10.5"
arduino = serial.Serial(device, 9600)
data = {
    "light_level": '0',
    "motion": '0'
}

command = {
    "command": ''
}

#for communication with sensors
def on_connect_mqtt(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe("/device/light")
    client.subscribe("/device/motion")
    client.subscribe("/actuator/command")
```

```
def on_message_mqtt(client, userdata, msg):
    print(msg.topic + " "+ str(msg.payload))
    if str(msg.topic) == "/device/light":
        data["light_level"] = json.loads(msg.payload)["light_level"]
#         print(json.loads(msg.payload)["light_level"])
    if str(msg.topic) == "/device/motion":
        data["motion"] = json.loads(msg.payload)["detect_motion"]
#         print(json.loads(msg.payload)["detect_motion"])
    if str(msg.topic) == "/actuator/command":
        command["command"] = str(msg.payload)

client = client.Client()
client.on_connect = on_connect_mqtt
client.on_message = on_message_mqtt


def get_db_connection():
    conn = psycopg2.connect(
        host = database,
        database = "postgres",
        user = "postgres",
        password = "postgres")
    return conn

def push_data(light_level, motion, rgb_value, timestamp):
    conn = get_db_connection()
    cur = conn.cursor()
    cur.execute("INSERT INTO arduino_data(light_level, motion, rgb_value, timestamp) VALUES (%s, %s, '%s', '%s')" % (light_level, motion, rgb_value, time
    conn.commit()
    cur.close()

#led_control function used for automation only
def led_control(light, motion):
    led_state = 0
    print('led control ' + str(light) + str(motion))
    if motion == "0\r\n":
        arduino.write('0\n'.encode('utf-8'))
```

```
#led_control function used for automation only
def led_control(light, motion):
    led_state = 0
    print('led control ' + str(light) + str(motion))
    if motion == "0\r\n":
        arduino.write('0\n'.encode('utf-8'))

    if motion == "1\r\n":
        if light > 640:
            arduino.write('0\n'.encode('utf-8'))
            led_state = "0-0-0"
        if light >= 625 and light < 640:
            arduino.write('1\n'.encode('utf-8'))
            led_state = "255-0-0"
        if light >= 550 and light < 600:
            arduino.write('2\n'.encode('utf-8'))
            led_state = "0-255-0"
        if light >= 500 and light < 550:
            arduino.write('3\n'.encode('utf-8'))
            led_state = "0-0-255"
        if light >= 450 and light < 500:
            arduino.write('4\n'.encode('utf-8'))
            led_state = "255-0-255"
        if light >= 400 and light < 450:
            arduino.write('5\n'.encode('utf-8'))
            led_state = "255-255-0"
        if light >= 350 and light < 400:
            arduino.write('6\n'.encode('utf-8'))
            led_state = "0-255-255"
        if light >= 200:
            arduino.write('2\n'.encode('utf-8'))
            led_state = "255-255-255"
    return led_state
```

```
def led_control_by_command(command):
    rgb_value = '0'
    arduino.write(command.encode('utf-8'))
    str = command.encode('utf-8')
    if len(str) > 1:
        numbers = str.split(" ")
        str1 = numbers[1].split(',')
        red = str1[0]
        str2 = numbers[2].split(',')
        green = str2[0]
        blue = numbers[3]
        rgb_value = "%s-%s-%s" % (red, green, blue)
        print(rgb_value)
        print("")
        print(data["light_level"])
        print(data["motion"])
    return rgb_value

while True:
# Random for testing purpose
#     order = random.randrange(350, 650)
#     print(order)
#     led_control(order, 1)
#     push_data(460, 1, "255-255-0", datetime.now())
    client.loop_start()
    client.connect(broker, port, 60)
# For automation only
    rgb_value = led_control(data["light_level"], data["motion"])
# For control from web interface
#     rgb_value = led_control_by_command(command["command"])
    push_data(data["light_level"], data["motion"], rgb_value, datetime.now())
    client.loop_stop()
```

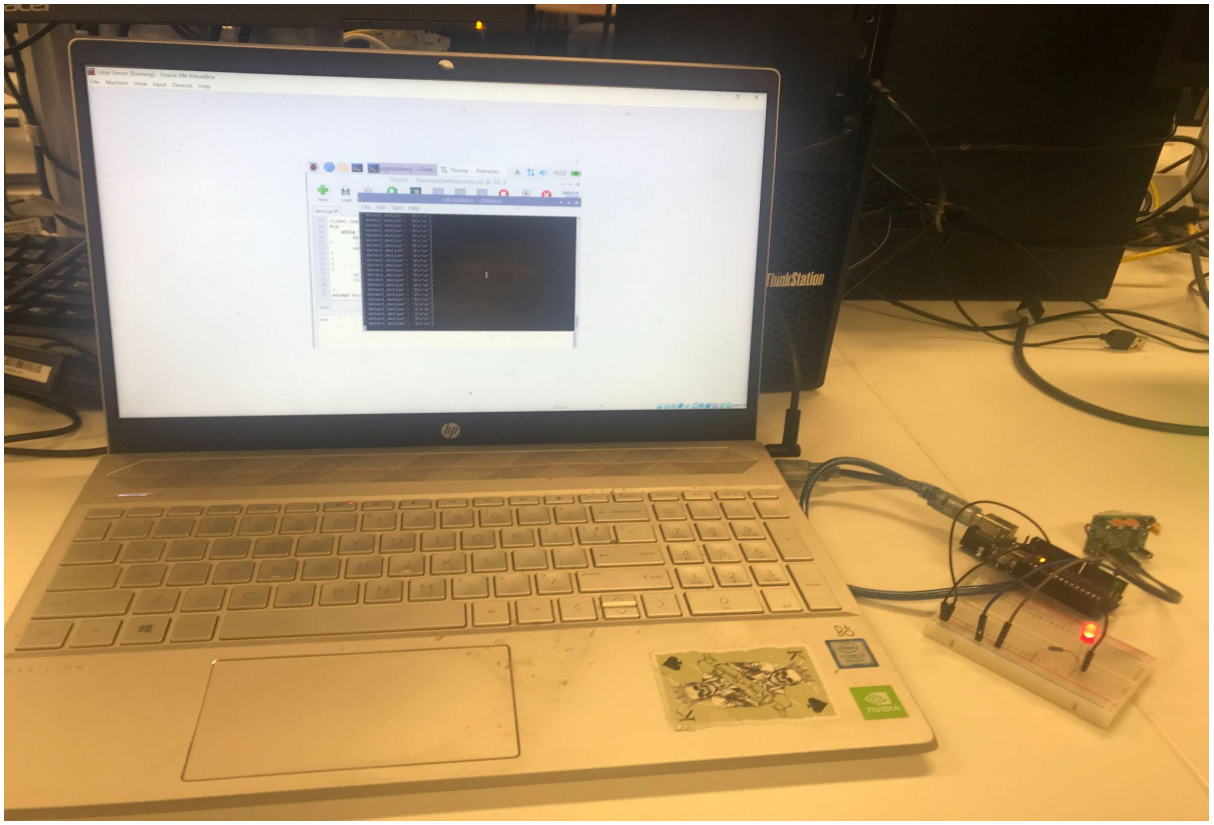
7. Data collection

The project used a cloud-based PostgreSQL database that was hosted on Amazon Web Services. We chose PostgreSQL as we were more familiar with the structure due to it being taught during this unit. pgAdmin4 was also used to monitor connections and view live transactions.

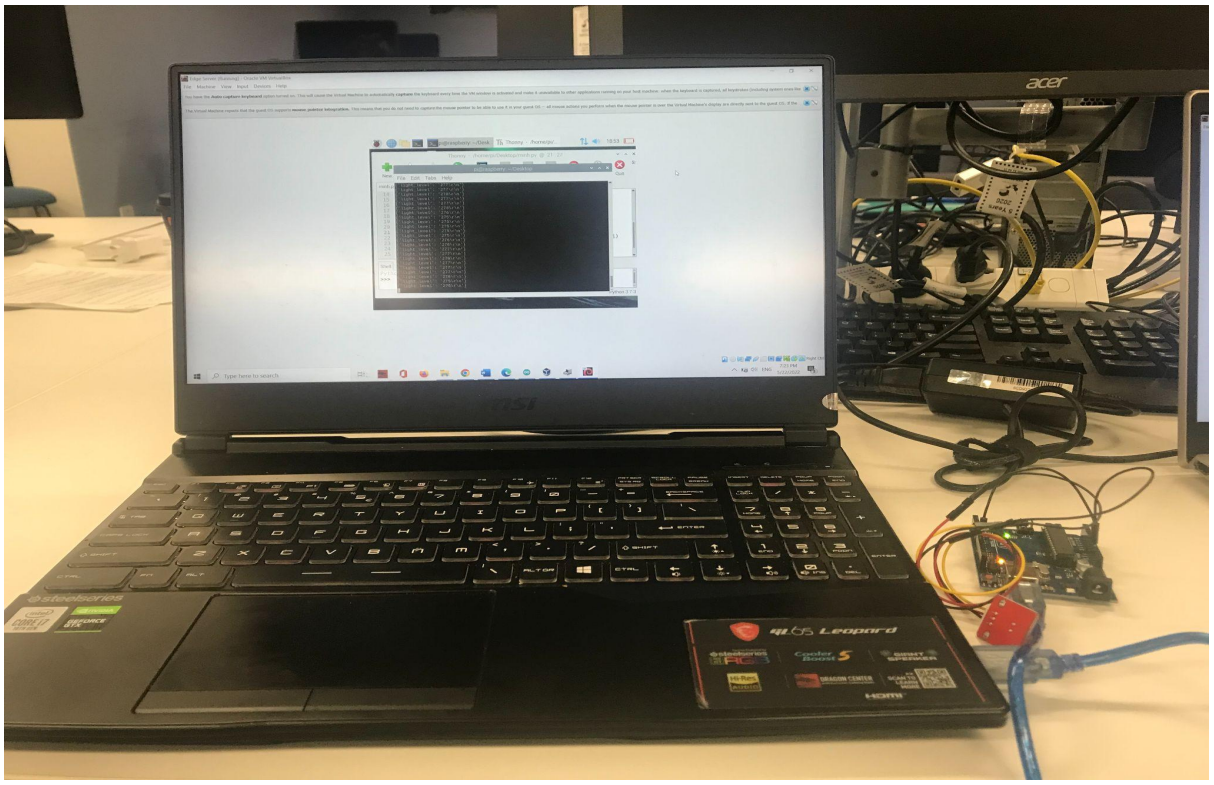
The database contained a single table that stored data for the current light level, whether motion was detected, and the RGB value of the LED. Each row also contained an entry id and a timestamp that were automatically generated whenever data was pushed to the table.

By using a cloud-based database, pushing and pulling data became a lot simpler as anyone was able to access the database provided that they had the correct hostname and password. Security measures can be tightened, however the group felt that it was unnecessary for a project of this scale.

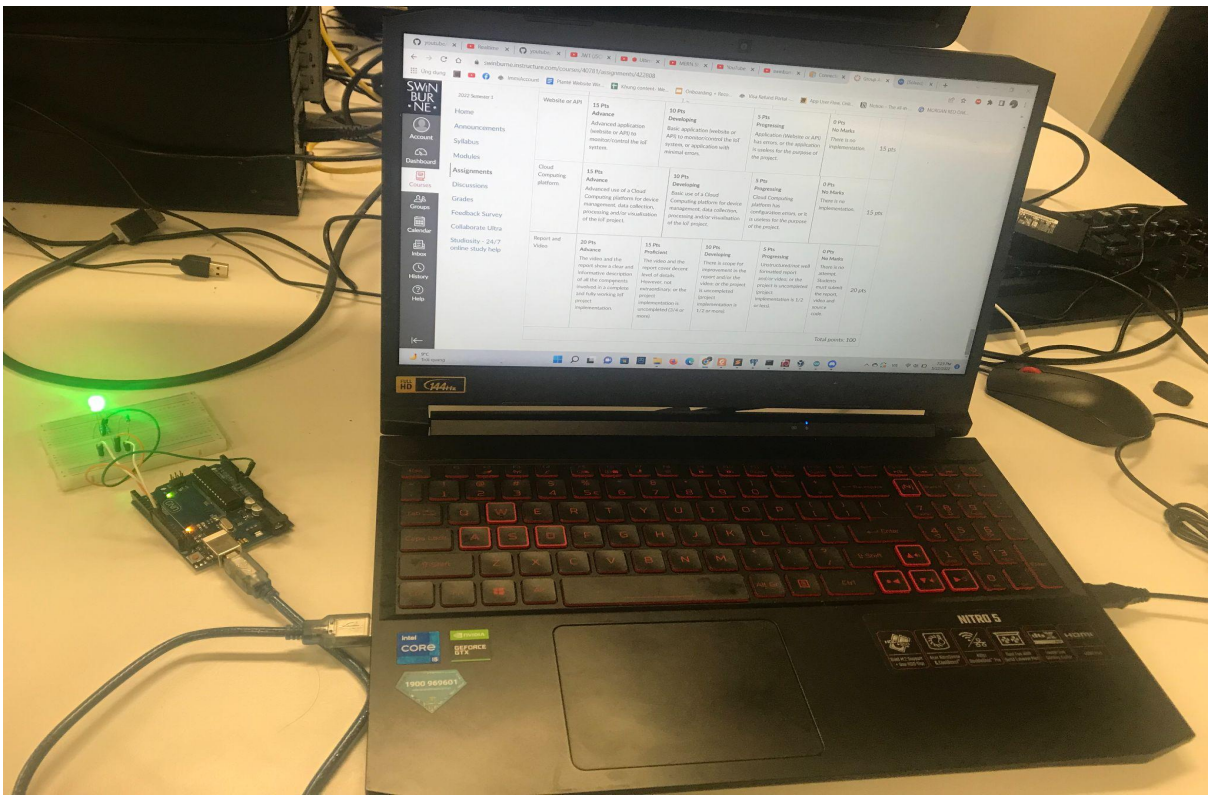
8. Edge Devices



Motion detection device



Light sensor device



RGB light device

III. Tasks Breakdown

Name	Student ID	Tasks
Phu Danh Pham	102619398	Arduino sketches and Python scripts for light sensor and motion detection sensor Edit demonstration video
Minh Le Anh Nguyen	103137309	Arduino sketches and Python scripts for RGB LED Setting up MQTT communication between IoT nodes Create conceptual and block diagrams
Zachary Stanford	102496096	Setting up cloud database and server Python scripts for web user interface