Link to Video Demonstration for proposed IoT system

https://www.youtube.com/watch?v=6iy1MO1Hi74

Summary

Topic background

The topic background that I have decided on is related to a smart home system. A smart home system can be defined as appliances in a home inter-connected to each other through the internet and this enables the house owners to remotely monitor and manage these appliances like controlling the temperature of the house or the lighting of the house. With the introduction of a smart home system, our daily lives can be improved where we are able to control the level of comfort, increase the level of conveniences by not having to manually switch off a light or help to increase the energy efficiency in a house which can help to cut cost in terms of paying less electricity bills. All of this can be achieved by pressing a few buttons on a smart home app where all the appliances in the house are connected to it through the internet.

The general idea on how these smart home appliances work is that a sensor in a smart home appliance is used to detect the readings of the environment and the data collected by the sensors will then be sent to the actuators in the smart home appliance and based on the readings obtained the smart home appliances will then change the environment of the house like turning on the aircon to increase the comfort level of the house owners.

There are different types of smart home technologies like smart locks where users can grant access to visitors without having to be near the door or help to unlock the door when the house owner is near. Another example would be a smart pet feeder where the pet feeder will automatically fill the food bowl of pets based on the timer set by the house owners. One more example would be the smart thermostat like Nest created by Nest Labs Inc. This smart thermostat allows the users to remotely monitor and control the home temperature. It also can learn the house owner's behaviors and based on these learnings it will automatically modify the settings of the thermostat to provide the best level of comfort available to the house owners.

Proposed System

The system that I am proposing is a system that detects the reading of light density and temperature in a room. Based on the readings obtained, it will trigger the actuators to signal the user to turn on the aircon or to turn off the aircon. All of these will be done using an Arduino Uno and the sensors used which is a LM35 temperature sensor and a Light Dependent Resistor (LDR) and actuators used which are Light-Emitting diodes (LEDs) will be connected to the Arduino Uno via a breadboard and jump wires. The data obtained from these sensors will then be sent to an edge device in this case a Raspberry PI via serial communication and will be stored in a MYSQL database in the edge device where the data will be used for analytics purposes and will be shown on a simple website hosted on the edge device for the users to see.

The problem that I have faced during the implementation of this proposed system is the LM35 temperature sensor that I have is not working so I had to improvise as suggested by Dr. Mark where I replaced the LM35 temperature sensor with a potentiometer. I used the potentiometer to simulate the temperature reading based on the ranges of the potentiometer reading.

Conceptual Design

Wiring diagram between Arduino Uno and Circuit using TinkerCad

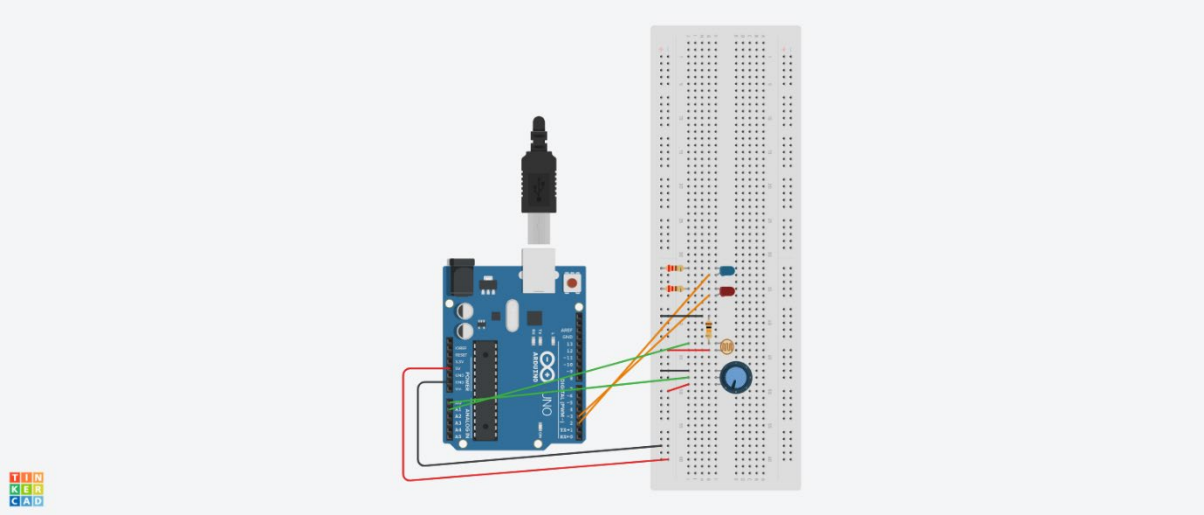


Figure (1) Wiring between circuit and Arduino Uno

Block Diagram for physical system

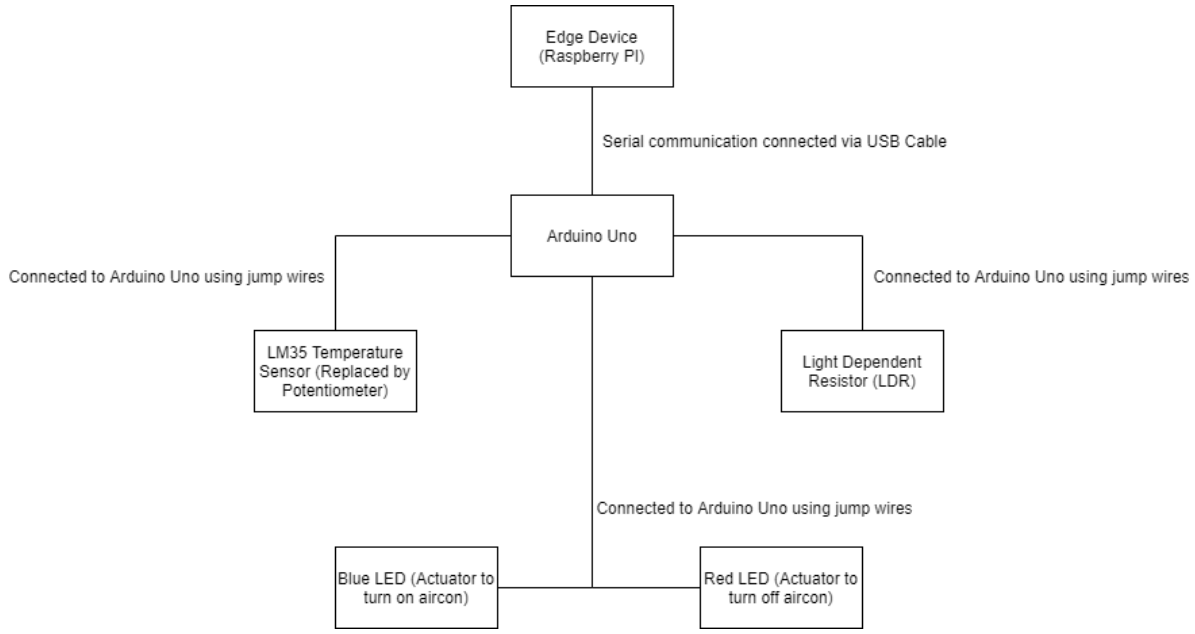


Figure (2) Block diagram for physical system

Block diagram for Software system

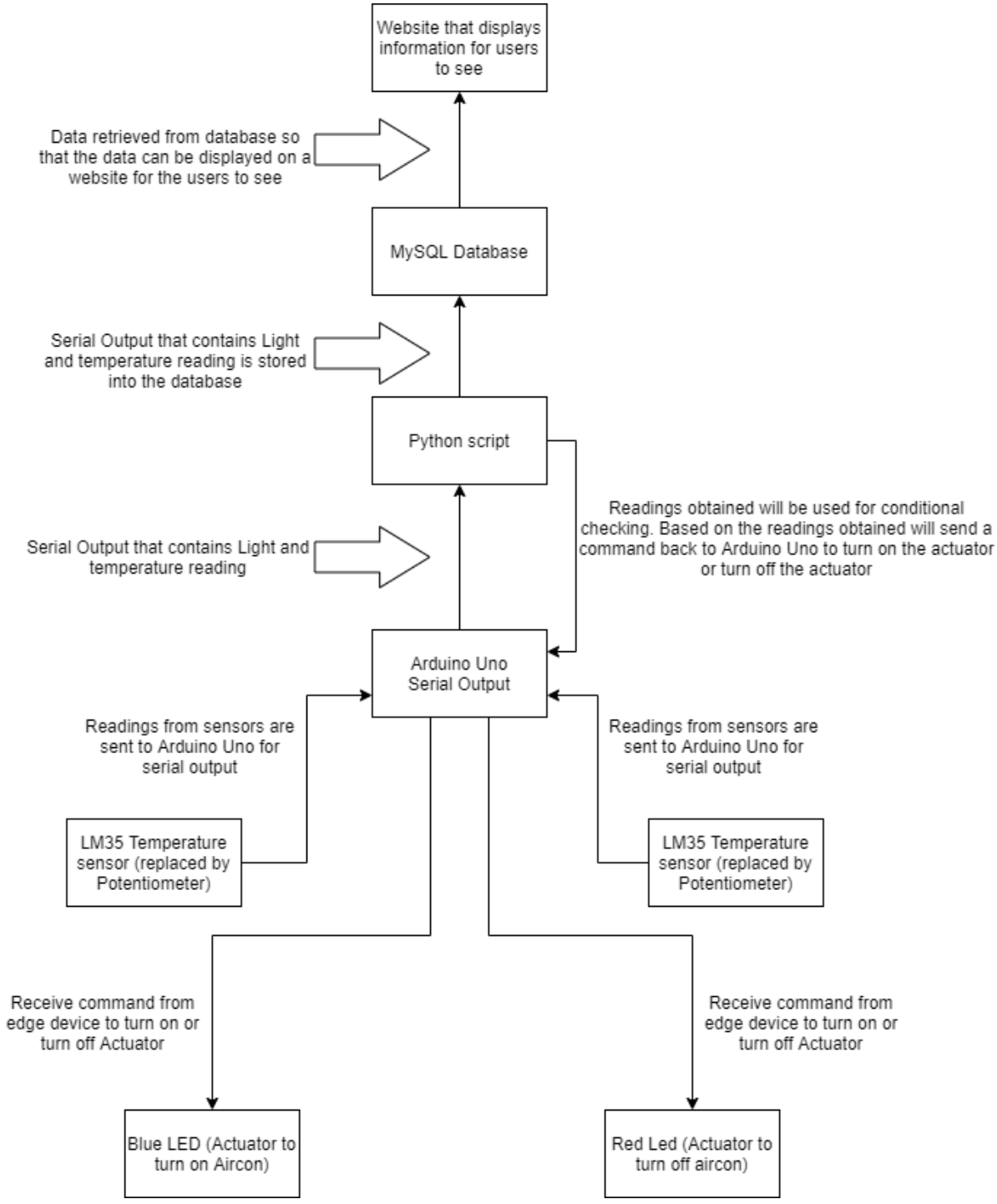
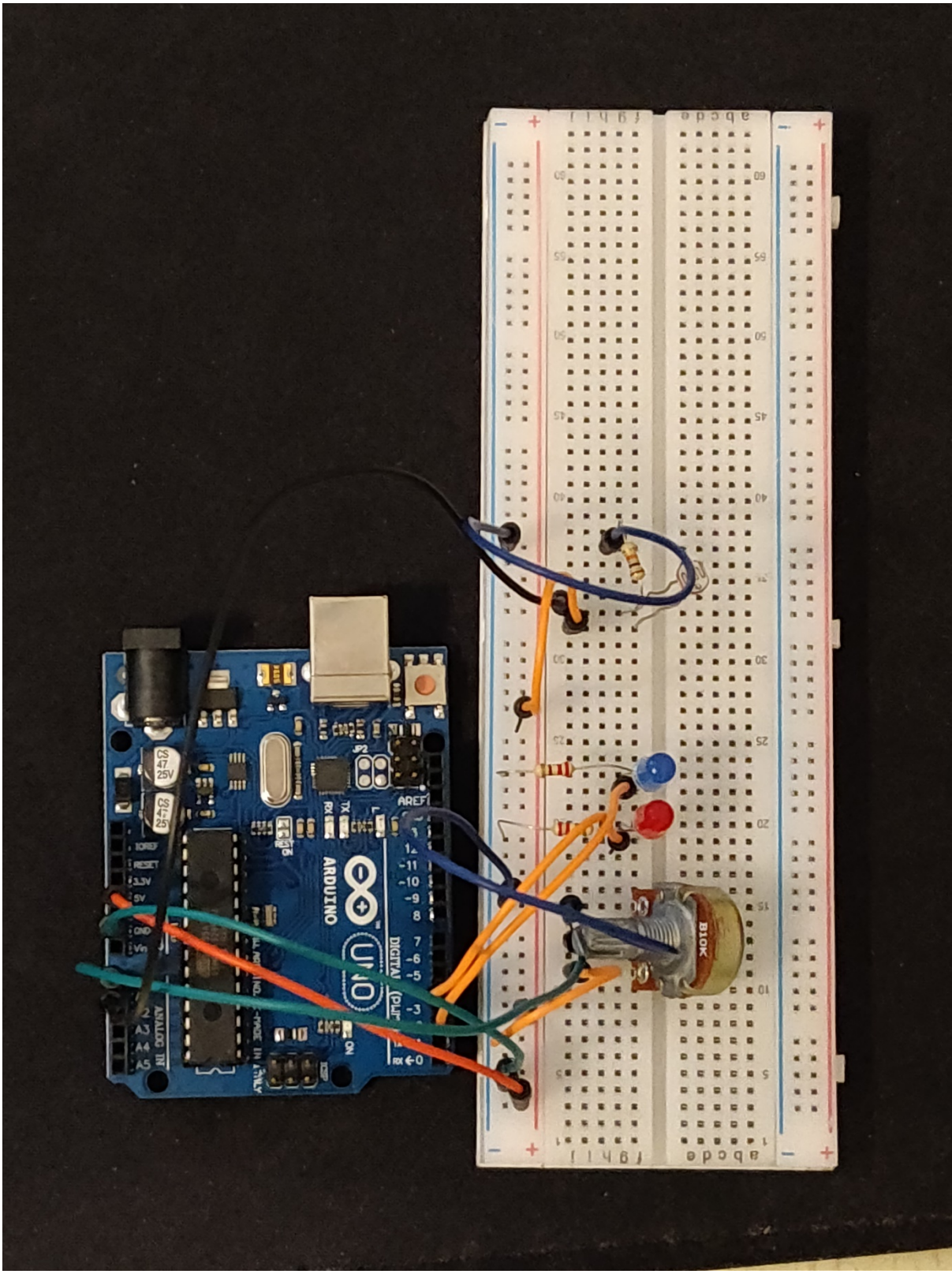


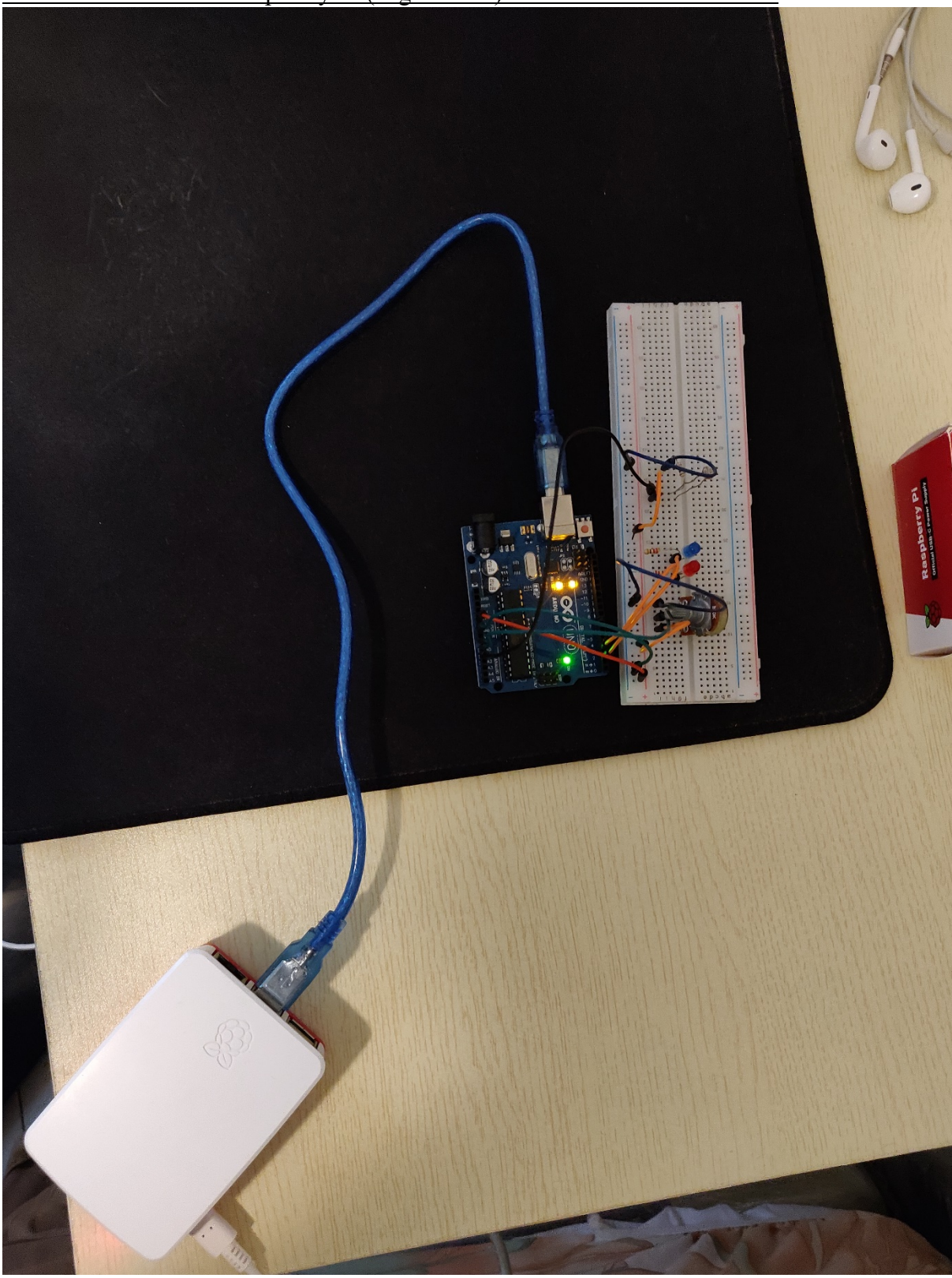
Figure (3) Block diagram for software system

Implementation

1. Arduino Set-up
    I. Breadboard with sensors connected to the Arduino

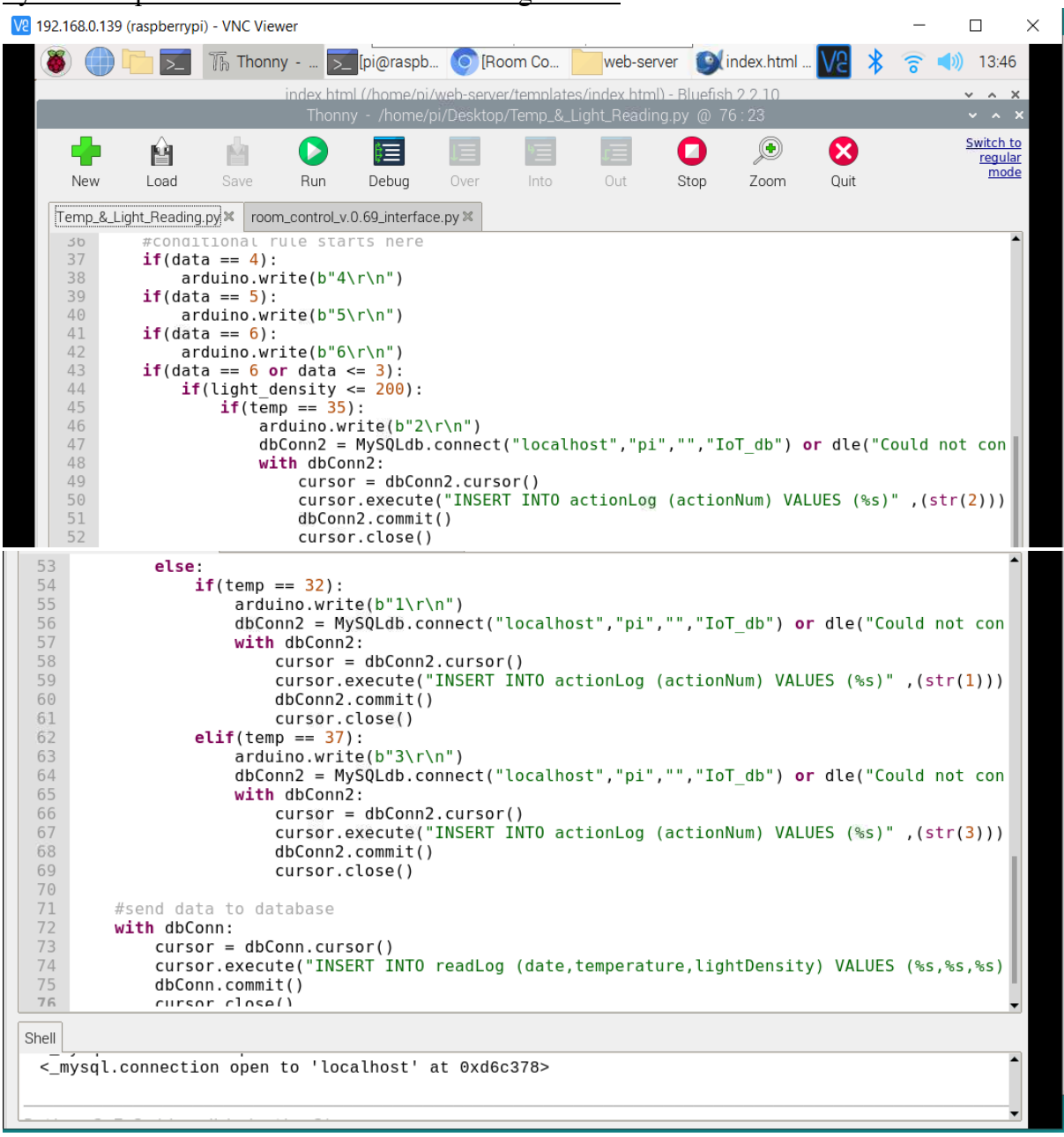


II. Arduino connected to Raspberry PI (edge device) via Serial USB connection

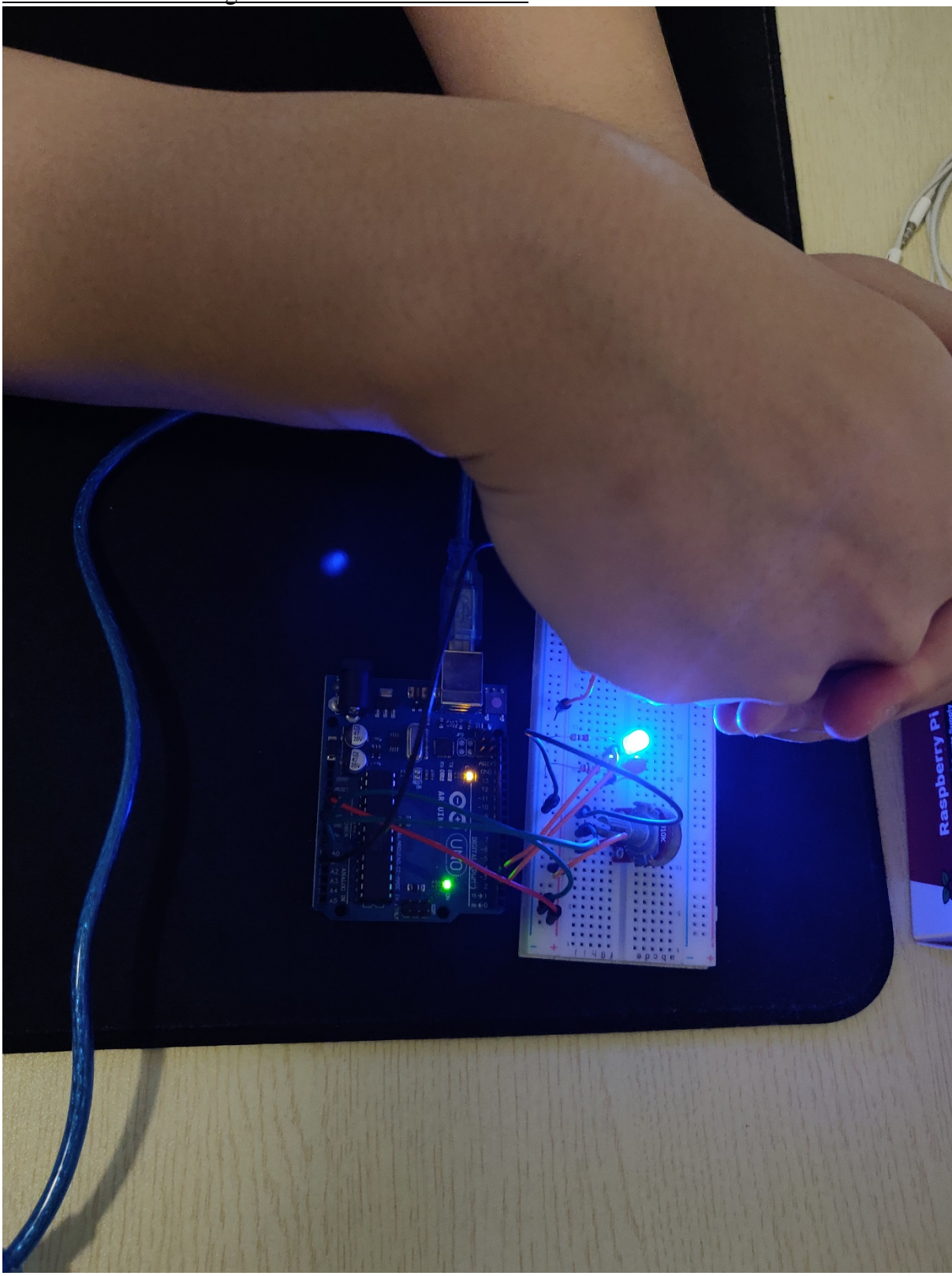


2. Demonstration on how the conditional checking works
(Red LED means to turn off Aircon and Blue LED means to turn on aircon both LED turning on means emergency to turn on aircon)

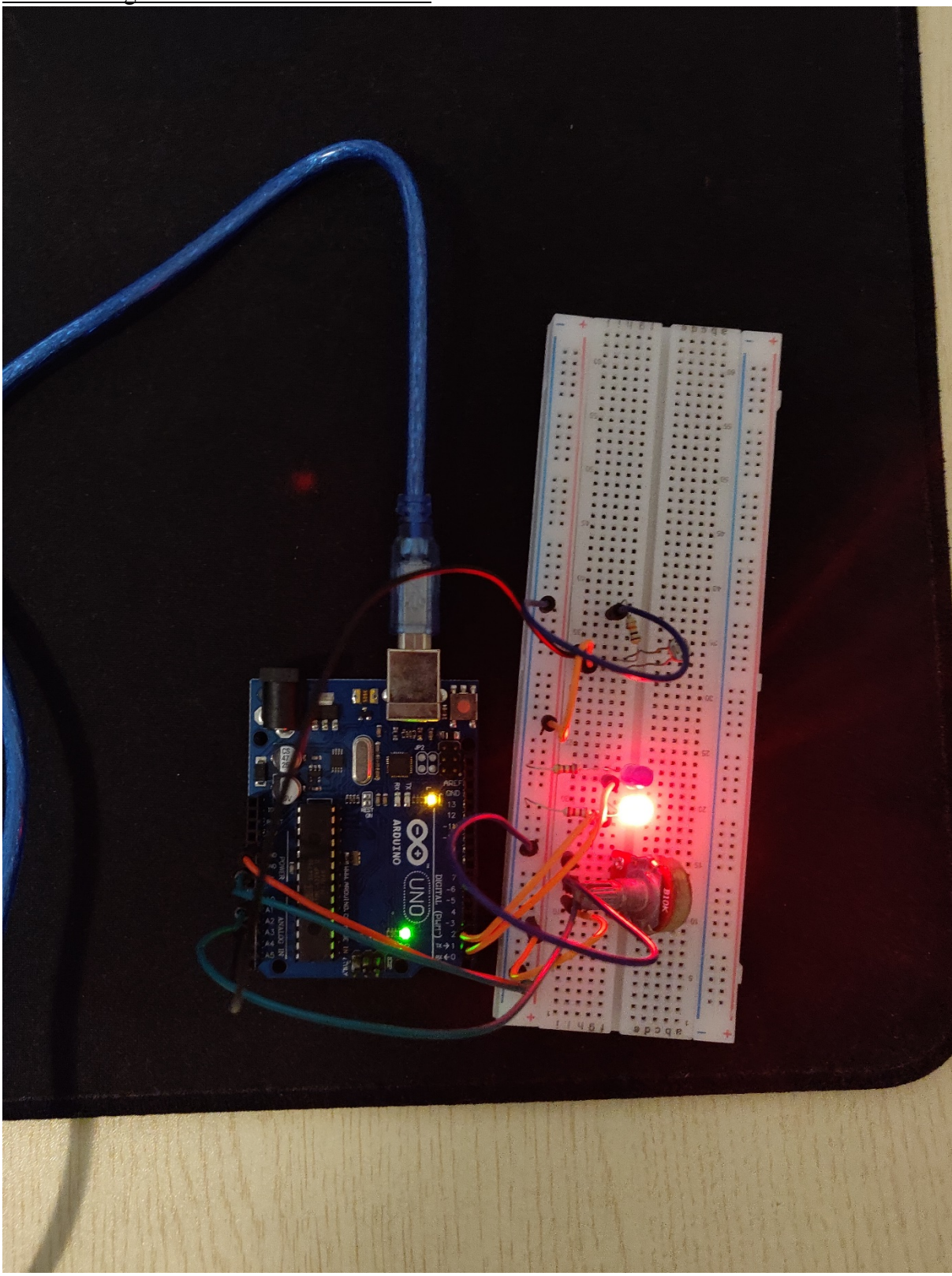
I. Python Script on how the conditional checking is done



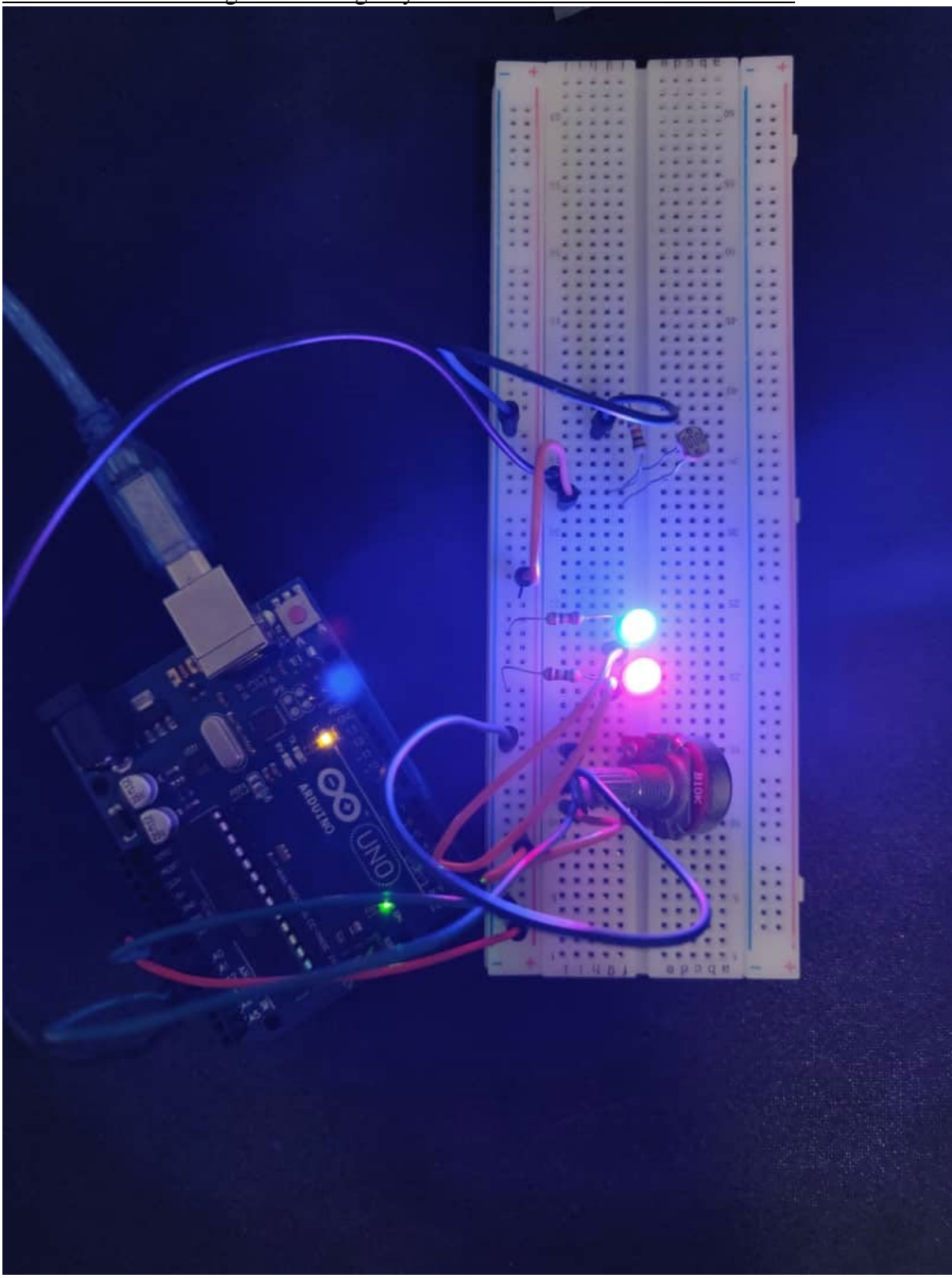
II. If temperature is 35 and light density reading is less than or equal to 200 blue LED will turn on to signal user to turn on the aircon



III. If temperature is 32 and light density reading is bigger than 200 red LED will turn on to signal user to turn off the aircon

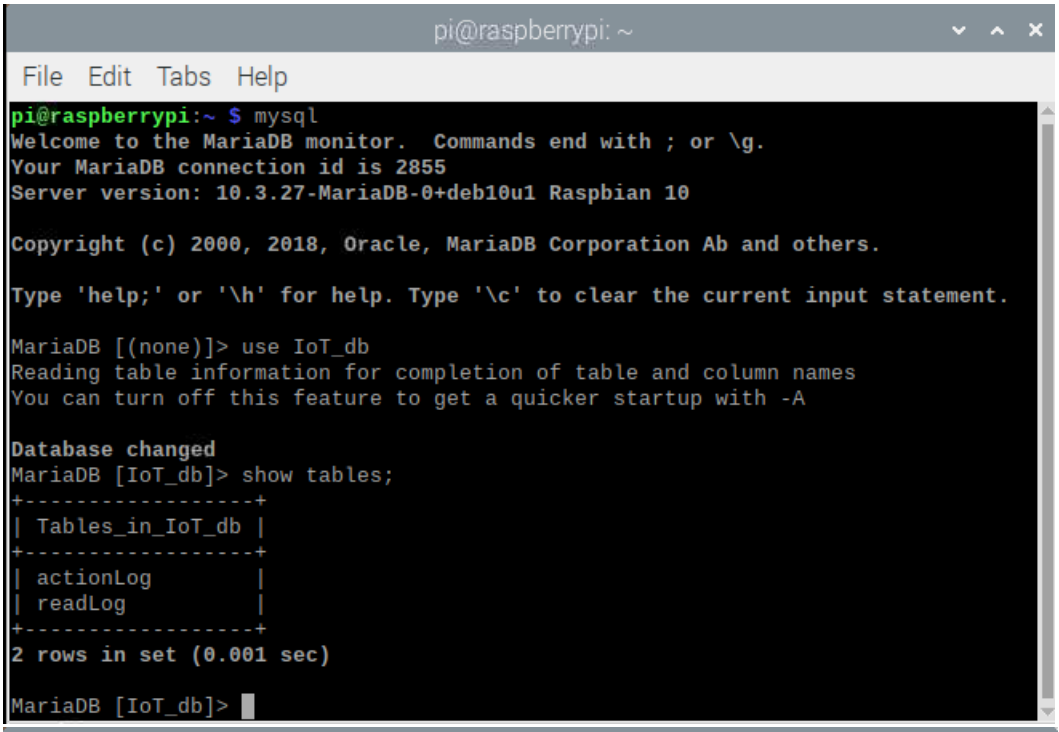


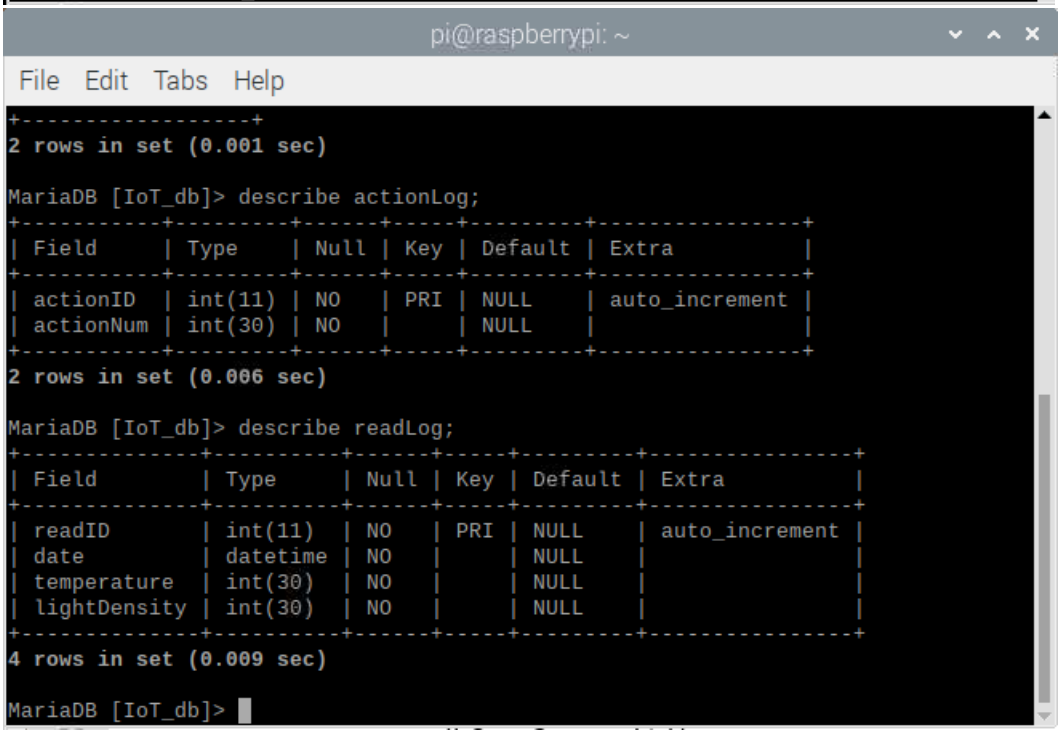
IV. If temperature is 37 and light density reading is bigger than 200 red and blue LED will turn on to signal an emergency situation to user to turn on the aircon



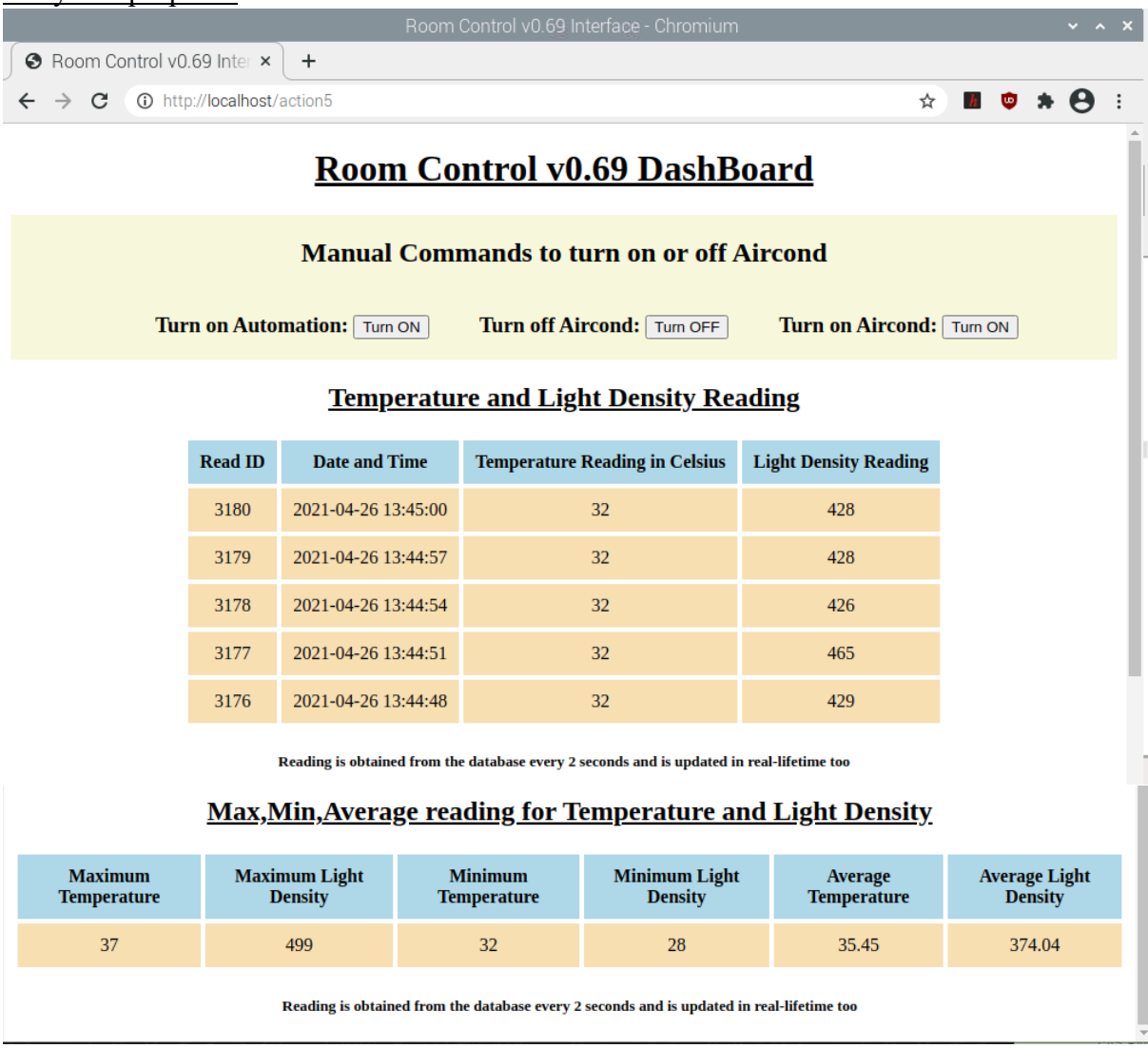
3. Database to store the data locally in Raspberry PI (edge device)

I. Database created and used to store the readings of IoT sensors and also action numbers

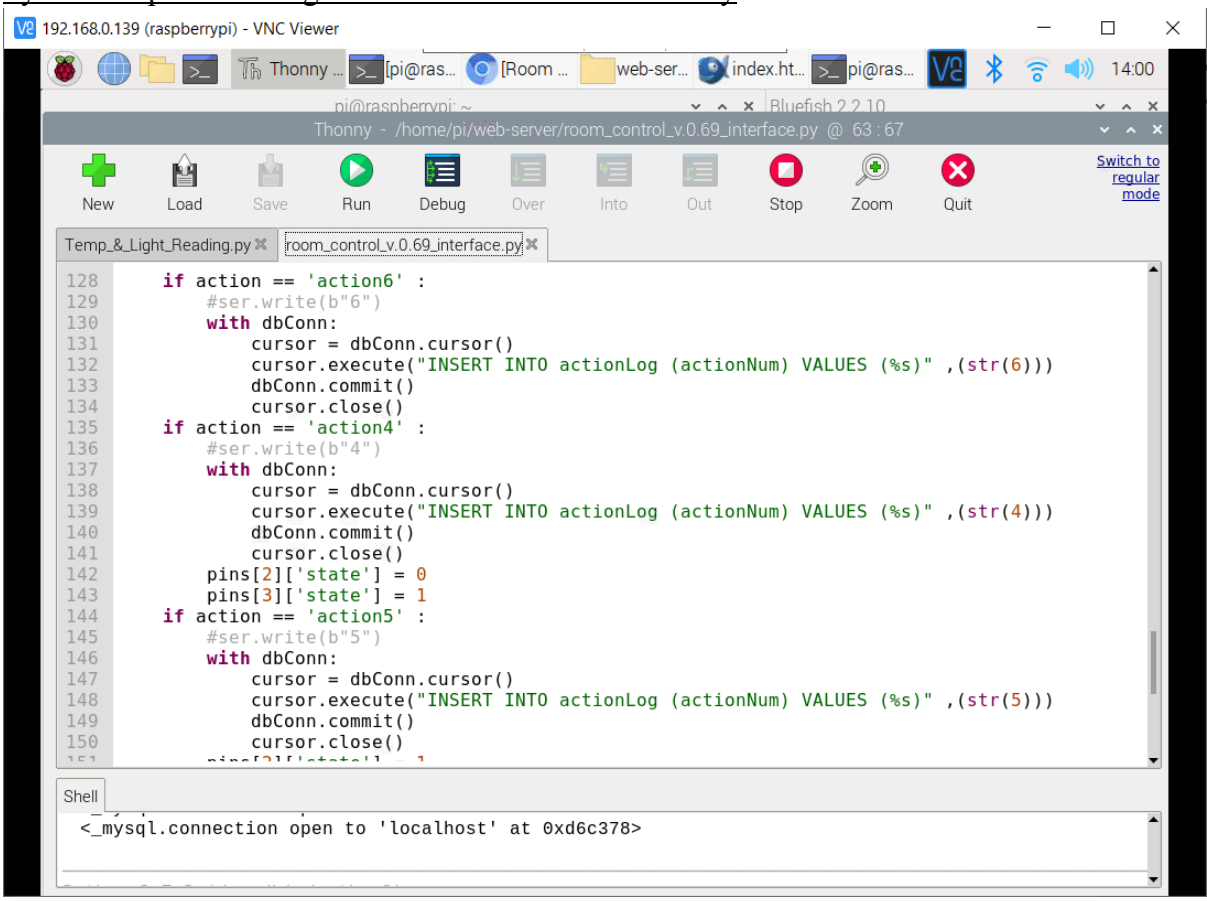




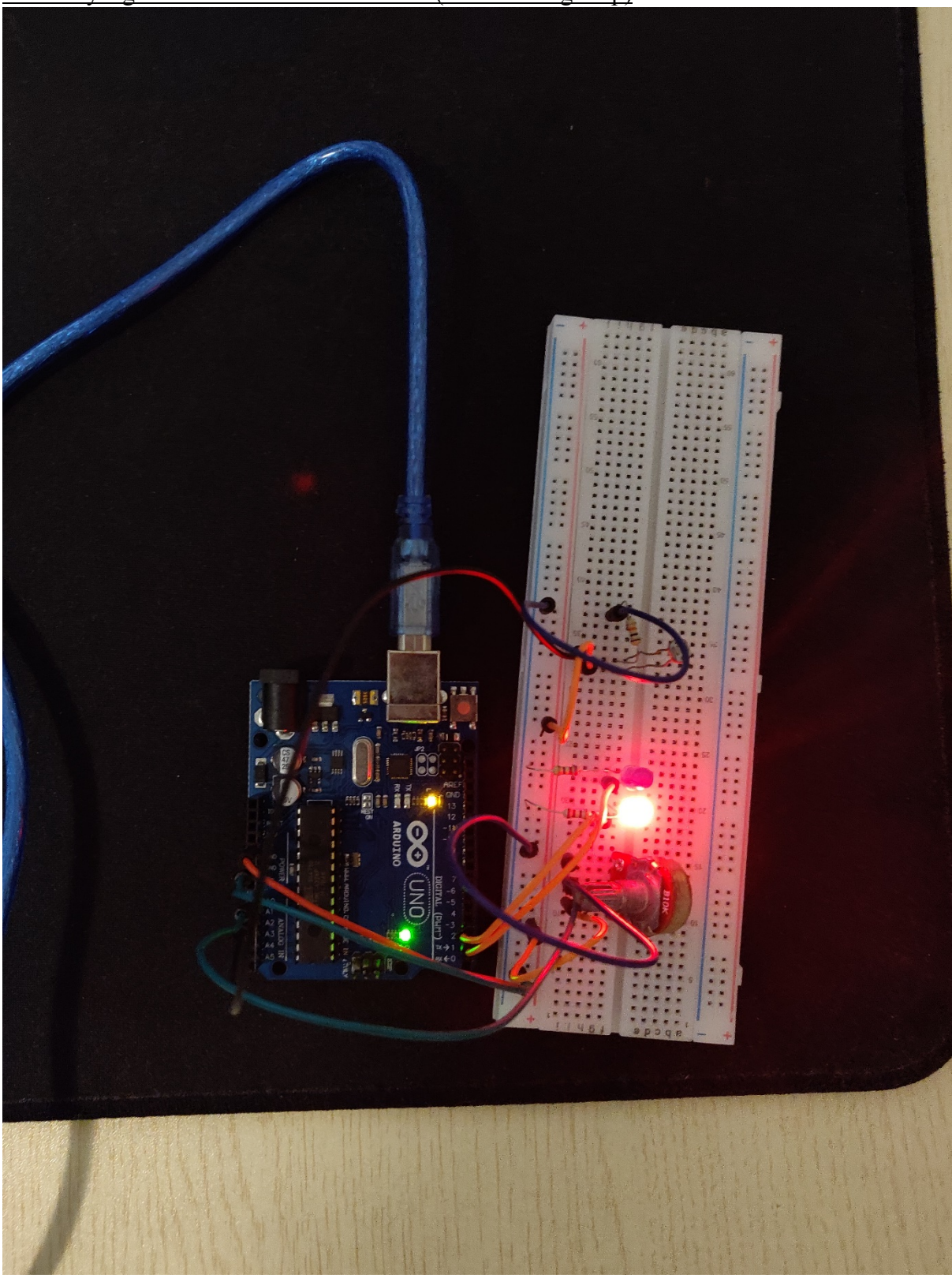
4. Website to show user data collected and show data for analytical purposes
    I. Website to show users the data collected from IoT sensors and also show the minimum, maximum and average readings of temperature and light density for analytical purposes



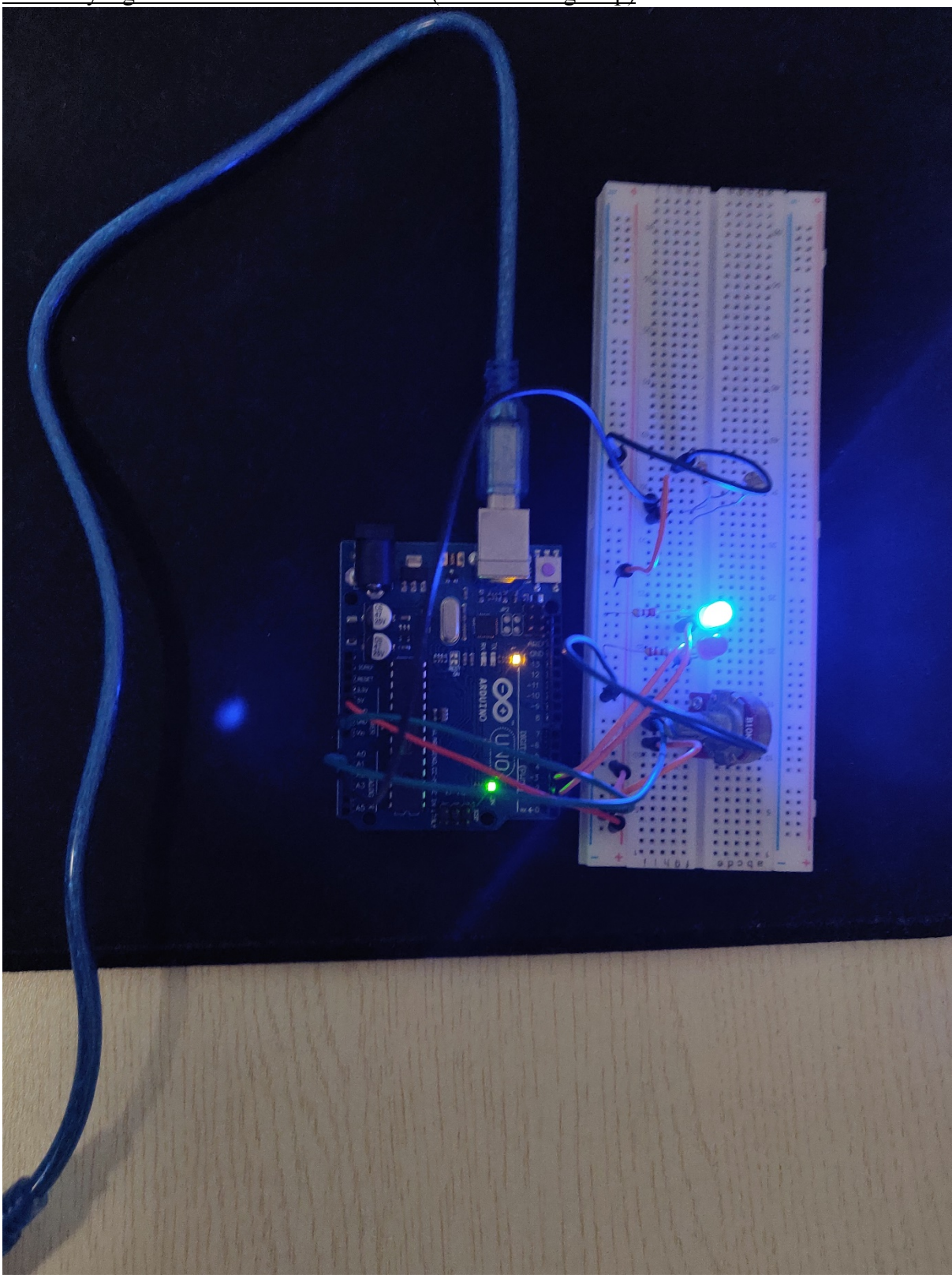
II. Python script for turning the actuators on or off manually



III. Manually signal user to turn off the aircon (Red LED light up)



IV. Manually signal user to turn on the aircon (Blue LED light up)



Resources used:

Software:

- **Arduino Sketch IDE**
- **Thonny Python IDE**
- **Bluefish IDE**
- **MySQL**

Online tutorials:

- **Raspberry Pi Arduino Serial Communication – Everything You Need To Know**
- **Temperature Sensor With Arduino UNO**
- **LDR (Light Dependent Resistor) Based Light Sensor using Arduino**

Appendix:

Arduino code:

Pen_LDR_Code.ino

[code]

```
#include <stdio.h>

//sensors and actuators here
int penmeter = A0;
int light_sensor = A1;
int blueLED = 2;
int redLED = 3;
bool manualInput;

//variables used to store values here
int tempSensorValue;
int lightSensorValue;
int temp;
unsigned int roomStatus = 0;

//variable to store readings and send to arduino
char buffer[30];

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(blueLED, OUTPUT);
  pinMode(redLED, OUTPUT);

}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  tempSensorValue = analogRead(penmeter);
  lightSensorValue = analogRead(light_sensor);

  // convert potentiometer reading to sensor based on ranges
  if(tempSensorValue >= 0 && tempSensorValue <342)
  {
    temp = 32;
  }
  else if (tempSensorValue > 341 && tempSensorValue <= 682)
  {
    temp = 35;
  }
  else
  {
    temp = 37;
  }

  //receive data from Edge_Device and turn on LED based on
  //commands received
if (Serial.available()>0)
{
  roomStatus = Serial.parseInt();
  if (roomStatus == 6)
  {
    manualInput = false;
  }
  else if (roomStatus == 1 && manualInput == false)
  {
```

```
    digitalWrite(redLED, HIGH);
    digitalWrite(blueLED, LOW);
  }
  else if (roomStatus == 2 && manualInput == false)
  {
    digitalWrite(redLED, LOW);
    digitalWrite(blueLED, HIGH);
  }
  else if (roomStatus == 3 && manualInput == false)
  {
    digitalWrite(redLED, HIGH);
    digitalWrite(blueLED, HIGH);
  }
  else if (roomStatus == 4)
  {
    digitalWrite(redLED, HIGH);
    digitalWrite(blueLED, LOW);
    manualInput = true;
  }
  else if (roomStatus == 5)
  {
    digitalWrite(redLED, LOW);
    digitalWrite(blueLED, HIGH);
    manualInput = true;
  }
}



  sprintf(buffer,"IOT Reading: %02d,%03d", temp, lightSensorValue);
```

```
  Serial.println(buffer);
  delay (2000);



}

   /*switch (roomStatus)
    {
      case 1:
        //TURN OFF AIRCOND ALMOST MORNING OR SOMETHING
        digitalWrite(redLED, HIGH);
        digitalWrite(blueLED, LOW);
        break;

      case 2:
        //TURN ON AIRCOND NIGHT TIME
        digitalWrite(redLED, LOW);
        digitalWrite(blueLED, HIGH);
        break;
      case 3:
        //EMERGENCY TURN ON AIRCOND TOO HOT AND IT IS DAYTIME
        digitalWrite(redLED, HIGH);
        digitalWrite(blueLED, HIGH);
        break;
      case 4:
        //TURN OFF AIRCOND MANUALLY
        digitalWrite(redLED, HIGH);
        digitalWrite(blueLED, LOW);
        break;
      case 5:
```

```
      //TURN ON AIRCOND MANUALLY
      digitalWrite(redLED, LOW);
      digitalWrite(blueLED, HIGH);
      break;
    default:
      break;
  }*/
```
[/code]

Python script for retrieving serial output from Arduino, Conditional checking and storing data to database

Temp_&_Light_Reading.py

```
import serial
import datetime
import time
import MySQLdb


device = '/dev/ttyACM0'
arduino = serial.Serial(device, 9600)


while 1:


    dbConn = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not connect to database")
    dbConn2 = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not connect to database")
    print(dbConn)
    print(dbConn2)



    while(arduino.in_waiting == 0):
        pass


    line = arduino.readline()
    temp = int(line[13:15])
    light_density = int(line[16:])
    currentDate = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S') #format to remove micro seconds
    print(temp)
    print(light_density)
    print(currentDate)
```

```
    with dbConn2:
        cursor = dbConn2.cursor()
        cursor.execute("SELECT actionNum FROM actionLog ORDER BY actionID DESC
LIMIT 1")
        data = cursor.fetchall()
        data = int(data[0][0])
        cursor.close()
    print(data)


    #conditional rule starts here
    if(data == 4):
        arduino.write(b"4\r\n")
    if(data == 5):
        arduino.write(b"5\r\n")
    if(data == 6):
        arduino.write(b"6\r\n")
    if(data == 6 or data <= 3):
        if(light_density <= 200):
            if(temp == 35):
                arduino.write(b"2\r\n")
                dbConn2 = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not
connect to database")
                with dbConn2:
                    cursor = dbConn2.cursor()
                    cursor.execute("INSERT INTO actionLog (actionNum) VALUES
(%s)" ,(str(2)))
                    dbConn2.commit()
                    cursor.close()
        else:
            if(temp == 32):
```

```
            arduino.write(b"1\r\n")
            dbConn2 = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not connect to database")
            with dbConn2:
                cursor = dbConn2.cursor()
                cursor.execute("INSERT INTO actionLog (actionNum) VALUES (%s)" ,(str(1)))
                dbConn2.commit()
                cursor.close()
        elif(temp == 37):
            arduino.write(b"3\r\n")
            dbConn2 = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not connect to database")
            with dbConn2:
                cursor = dbConn2.cursor()
                cursor.execute("INSERT INTO actionLog (actionNum) VALUES (%s)" ,(str(3)))
                dbConn2.commit()
                cursor.close()

    #send data to database
    with dbConn:
        cursor = dbConn.cursor()
        cursor.execute("INSERT INTO readLog (date,temperature,lightDensity) VALUES (%s,%s,%s)" ,(currentDate,temp,light_density))
        dbConn.commit()
        cursor.close()
```

Python script for retrieving data from database and show the information on the website and also python script to manually turn off and turn on the actuators

room_control_v0.69_interface.py

```
import serial
import datetime
import MySQLdb
from flask import Flask, render_template


#create flask object called app
app = Flask(__name__)


# Dictionary of pins with name of pin and state ON/OFF
pins = {
    2: {'name' : 'Blue LED', 'state' : 0},
    3: {'name' : 'Red LED', 'state' : 0}
    }


#return index function when someone accsses the root URL ('/') of the server
@app.route("/")
def index():
   dbConn = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not connect to database")


   print(dbConn)


   with dbConn:
      cursor = dbConn.cursor()
      cursor.execute("SELECT * FROM readLog ORDER BY date DESC LIMIT 5")
      datafetch = cursor.fetchall()
      cursor.close()
```

```
with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MAX(temperature) FROM readLog")
    maxtemp = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MAX(lightDensity) FROM readLog")
    maxlight = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MIN(temperature) FROM readLog")
    mintemp = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MIN(lightDensity) FROM readLog")
    minlight = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT ROUND(AVG(temperature),2) FROM readLog")
    avgtemp = cursor.fetchall()
    cursor.close()
```

```
    with dbConn:
        cursor = dbConn.cursor()
        cursor.execute("SELECT ROUND(AVG(lightDensity),2) FROM readLog")
        avglight = cursor.fetchall()
        cursor.close()

    # TODO: Read the status of the pins ON/OFF and update dictionary
    # This data will be sent to index.html (pin dictionary)
    templateData = {
        'pins' : pins,
        'datafetch' : datafetch,
        'maxtemp' : maxtemp,
        'mintemp' : mintemp,
        'maxlight': maxlight,
        'minlight': minlight,
        'avgtemp' : avgtemp,
        'avglight': avglight
    }
    # Pass the template data into the template index.html and return it
    return render_template('index.html', **templateData)


# Function to send simple commands
@app.route("/<action>")
def action(action):
    dbConn = MySQLdb.connect("localhost","pi","","IoT_db") or dle("Could not connect to
database")


    print(dbConn)


    with dbConn:
```

```
    cursor = dbConn.cursor()
    cursor.execute("SELECT * FROM readLog ORDER BY date DESC LIMIT 5")
    datafetch = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MAX(temperature) FROM readLog")
    maxtemp = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MAX(lightDensity) FROM readLog")
    maxlight = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MIN(temperature) FROM readLog")
    mintemp = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT MIN(lightDensity) FROM readLog")
    minlight = cursor.fetchall()
    cursor.close()

with dbConn:
```

```
    cursor = dbConn.cursor()
    cursor.execute("SELECT ROUND(AVG(temperature),2) FROM readLog")
    avgtemp = cursor.fetchall()
    cursor.close()

with dbConn:
    cursor = dbConn.cursor()
    cursor.execute("SELECT ROUND(AVG(lightDensity),2) FROM readLog")
    avglight = cursor.fetchall()
    cursor.close()

if action == 'action6' :
    #ser.write(b"6")
    with dbConn:
        cursor = dbConn.cursor()
        cursor.execute("INSERT INTO actionLog (actionNum) VALUES (%s)" ,(str(6)))
        dbConn.commit()
        cursor.close()
if action == 'action4' :
    #ser.write(b"4")
    with dbConn:
        cursor = dbConn.cursor()
        cursor.execute("INSERT INTO actionLog (actionNum) VALUES (%s)" ,(str(4)))
        dbConn.commit()
        cursor.close()
    pins[2]['state'] = 0
    pins[3]['state'] = 1
if action == 'action5' :
    #ser.write(b"5")
    with dbConn:
```

```
            cursor = dbConn.cursor()
            cursor.execute("INSERT INTO actionLog (actionNum) VALUES (%s)" ,(str(5)))
            dbConn.commit()
            cursor.close()
        pins[2]['state'] = 1
        pins[3]['state'] = 0

    # This data will be sent to index.html (pins dictionary)
    templateData = {
        'pins' : pins,
        'datafetch' : datafetch,
        'maxtemp' : maxtemp,
        'mintemp' : mintemp,
        'maxlight': maxlight,
        'minlight' : minlight,
        'avgtemp' : avgtemp,
        'avglight': avglight
    }
    # Pass the template data into the template index.html and return it
    return render_template('index.html', **templateData)

if __name__ == '__main__':
    #ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    #ser.flush()
    app.run(host='0.0.0.0', port=80, debug=True)
```

HTML and CSS script for website

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Room Control v0.69 Interface</title>
        <link rel="stylesheet" href="/static/styles.css">
    </head>

    <body>
        <h1 id="myHeader"> Room Control v0.69 DashBoard</h1>


                    <div id="buttonContainer">
                        <h2 id="manualCommand"> Manual Commands to turn on or off Aircond </h2>
                            <h3 id="button1"> Turn on Automation: <a href="/action6"><button> Turn ON</button> </a> </h3>
                            <h3 id="button2"> Turn off Aircond: <a href="/action4"><button>Turn OFF</button> </a> </h3>
                            <h3 id="button3"> Turn on Aircond: <a href="/action5"><button>Turn ON</button> </a> </h3>
                    </div>


        <div id="tableContainer">
            <h2 id="tableHeader"> Temperature and Light Density Reading </h2>
                <table id="myTable">
                    <thead>
                    <tr>
                        <th> Read ID </th>
                        <th> Date and Time </th>
                        <th> Temperature Reading in Celsius </th>
```

```
                    <th> Light Density Reading </th>
                </tr>
                </thead>
                <tbody>
                    {%for row in datafetch %}
                        <tr>
                            <td>{{row[0]}}</td>
                            <td>{{row[1]}}</td>
                            <td>{{row[2]}}</td>
                            <td>{{row[3]}}</td>

                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        <h5 id="tableFooter">Reading is obtained from the database every 2 seconds and is updated in real-lifetime too</h5>
        </div>

        <div id="table2Container">
            <h2 id="tableHeader"> Max,Min,Average reading for Temperature and Light Density </h2>
            <table id="myTable2">
                <thead>
                <tr>
                    <th> Maximum Temperature </th>
                    <th> Maximum Light Density </th>
                    <th> Minimum Temperature </th>
```

```
                <th> Minimum Light Density
</th>
                <th> Average Temperature </th>
                <th> Average Light Density
</th>
            </tr>
            </thead>
            <tbody>
                <tr>
                    {%for i in
maxtemp%}
        <td>{{i[0]}}</td>
                    {%endfor%}
                    {%for j in
maxlight%}
        <td>{{j[0]}}</td>
                    {%endfor%}
                    {%for k in
mintemp%}
        <td>{{k[0]}}</td>
                    {%endfor%}
                    {%for l in
minlight%}
        <td>{{l[0]}}</td>
                    {%endfor%}
                    {%for m in
avgtemp%}
        <td>{{m[0]}}</td>
                    {%endfor%}
```

{%for n in avglight%}

<td>{{n[0]}}</td>

{%endfor%}

</tr>

</tbody>

</table>

<h5 id="table2Footer">Reading is obtained from the database every 2 seconds and is updated in real-lifetime too</h5>

</body>

</html>

styles.css

```
/*header and manual commands starts here*/
#myHeader{

    text-align: center;
    text-decoration: underline;
}

#manualCommand{
    text-align: center;
    background-color: beige;
    padding-top: 20px;
}

body{
    background-color: F0EAE3;
}

#buttonContainer{
    text-align: center;
    background-color: beige;

}


#button1, #button2, #button3{
    display: inline-block;
    margin-left: 40px;
}
```

```
/* table starts here */
#tableHeader{
	text-align: center;
	text-decoration: underline;
}

#tableFooter{
	text-align: center;

}

#tableContainer{

}

#myTable{
	margin-left: auto;
	margin-right: auto;
}

td{
	text-align: center;
	border: 1px solid white;
	border-collapse:collpase;
	background-color: wheat;
	padding: 10px;


}
```

```
table{
	text-align: center;
	border: 1px solid white;
	border-collapse:collpase;

}

th{
	text-align: center;
	border: 1px solid white;
	border-collapse:collpase;
	background-color: lightblue;
	padding: 10px;

}

/*table 2 starts here */

#table2Container{

}

#myTable2{
	margin-left: auto;
	margin-right: auto;
}

#table2Footer{
	text-align: center;
```

}

References:

Shea, S 2020, *smart home or building (home automation or domotics)*, IoT Agenda, Viewed 25 April 2021, < https://internetofthingsagenda.techtarget.com/definition/smart-home-or-building >