

SWE30011 – IoT Programming

IoT Assignment 3 – Group Assignment (Practical)

Proposed System: Smart Greenhouse

Team Member Names & IDs:

Andrea Espitia 103165504

Thiem Quyen 102436746

Autumn Tan 103163663

Tutor Date & Time: Wed 10:30 ATC325

Tutor Name: Anika Kanwal

Video Link: <https://www.youtube.com/watch?v=5BMPULlulEs>

Table of Contents

Introduction	3
Background	3
Proposed System	3
Conceptual Design.....	4
Block Diagrams	4
Task Breakdown	6
Project Implementation	8
Hardware Schematic	8
Software Schematic	9
Temperature and Humidity IoT Node	9
Soil Moisture IoT Node.....	10
Light Level IoT Node.....	11
Pin Connections on Each Node.....	11
Temperature and Humidity IoT Node:	12
Soil Moisture IoT Node:.....	12
IoT Nodes.....	13
Edge Servers.....	13
Communication Protocols.....	15
Website	16
Cloud Computing.....	18
User Manual.....	20
Limitations.....	21
Resources	22
Appendix	24
A - Source Code for Temperature Sensor Node.....	24
B - Source Code for Soil Moisture Sensor Node.....	29
C - Source Code for Light Level Sensor Node.....	30

Introduction

As favorable climate and soil conditions support the production of crops, a greenhouse would provide a controlled environment customized to the vegetation cultivated inside of it [1]. This paper describes an Internet of Things (IoT) smart greenhouse project that adapts crop environment conditions such as temperature levels, soil moisture, and light levels to allow reactive solutions to outside influences that may impact the plant being cultivated inside of the greenhouse.

Background

Nowadays, climate change patterns and the increasing consumption of resources are impacting the farming industry significantly. Due to unpredictable environmental conditions, advanced technologies need to be implemented to improve production efficiency and crop resilience. IoT technology offers an efficient way to monitor and improve environmental conditions during the farming process by using sensing devices for real-time monitoring and data collection, as well as actuators to provide enough warmth, light and water for crops, thus supporting an effective and scalable automatic decision-making process to control the conditions of the crops' environment.

Proposed System

Therefore, we would like to propose an IoT-based smart greenhouse system that makes use of sensors for real-time monitoring, a cloud for data collection by considering the current temperature, soil moisture and light level within the greenhouse and then making data driven decisions via edge server analytics to activate various actuators that will change the crops' environmental conditions for optimal plant growth. We will also be developing a website to control the actuators remotely and making use of a cloud computing platform to visualise the data collected.

Conceptual Design

Block Diagrams

The following block diagrams show the hardware/software used in our project:

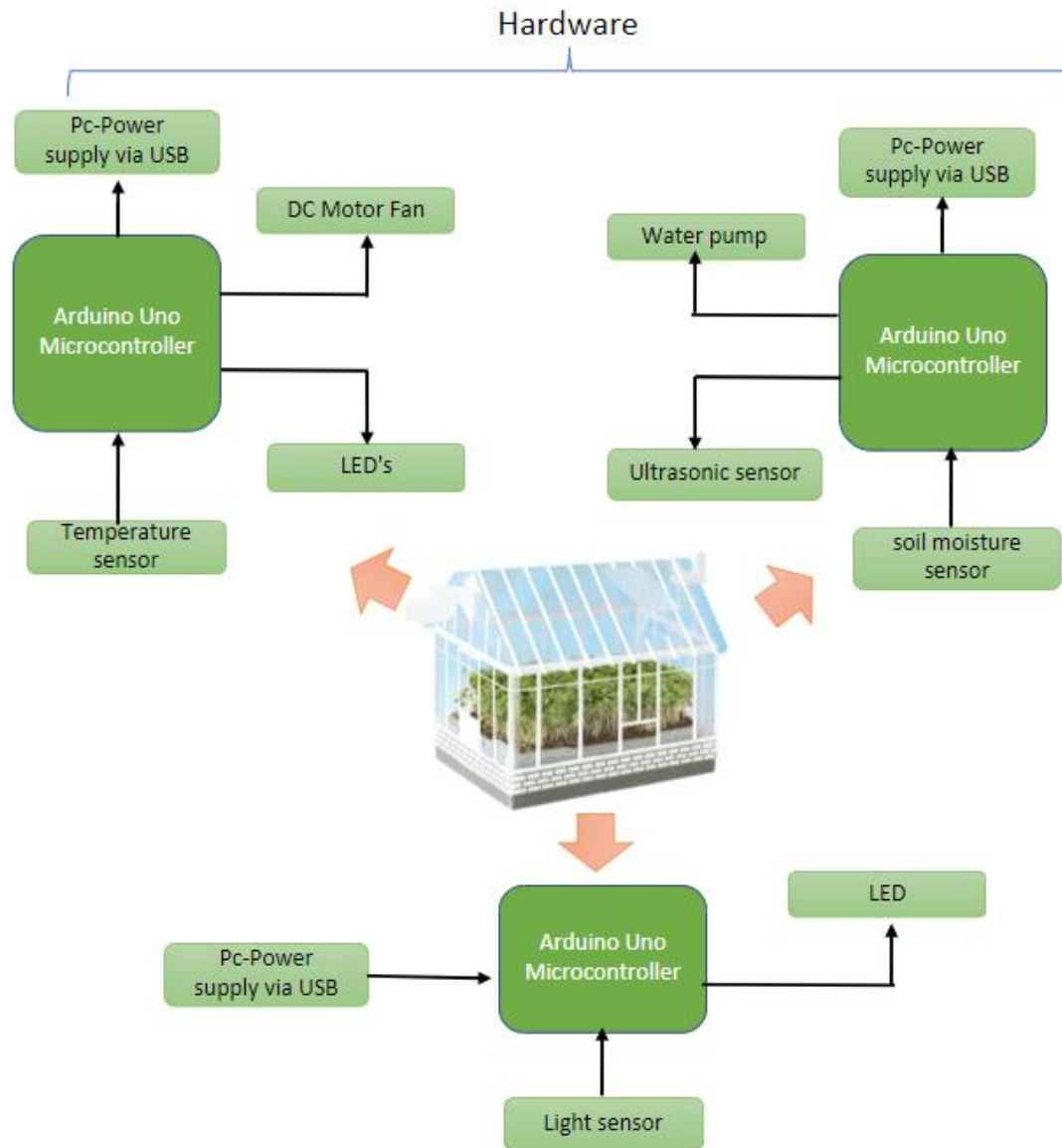


Figure 4: Block Diagram of IoT smart greenhouse hardware system.

As shown in Figure 4, the hardware consists of three IoT nodes, each connected to an Arduino Uno microcontroller, which monitor and sense the various environmental conditions of the greenhouse. The IoT nodes have sensors and actuators to identify changes in the temperature, humidity, soil moisture and light level and act accordingly.

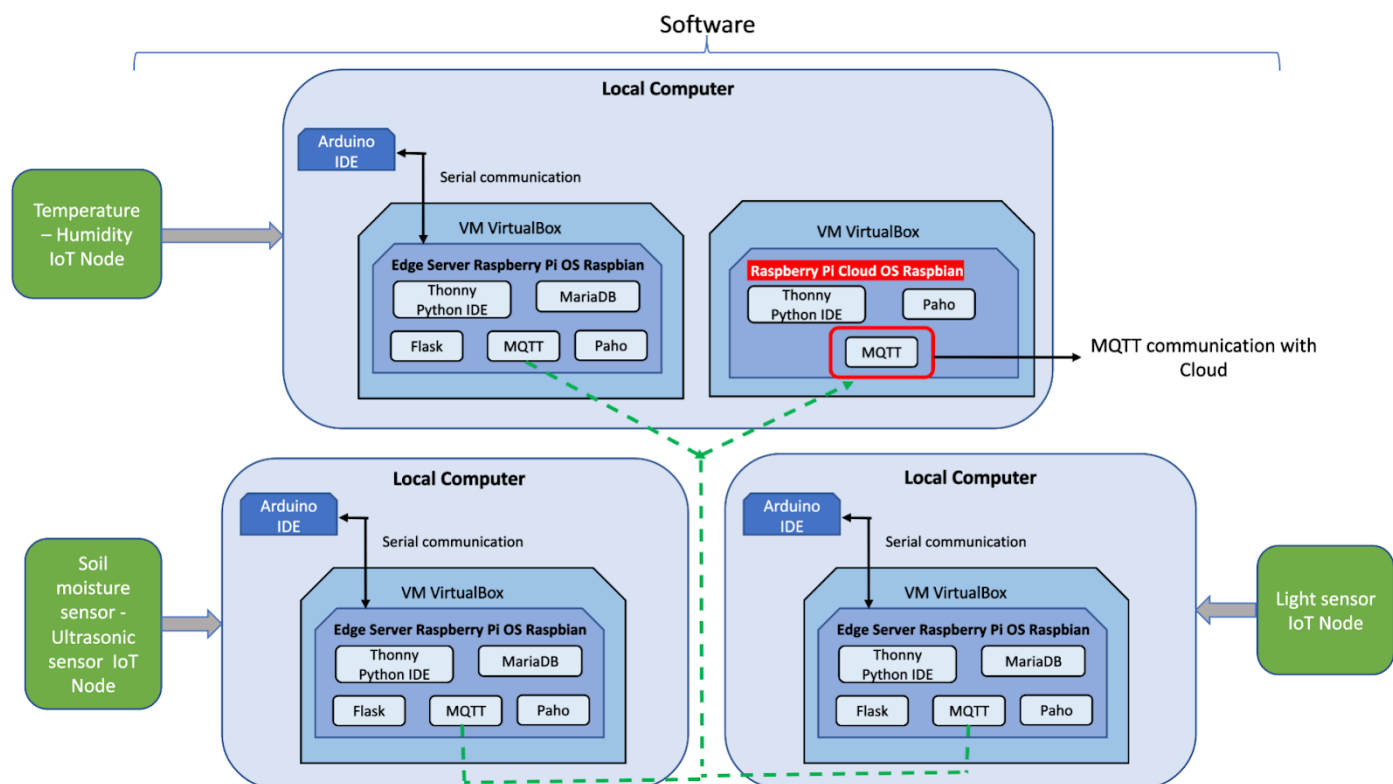


Figure 5: Block Diagram of IoT smart greenhouse software system.

The IoT nodes are controlled by three local computers connected via USB, which send their sensor data to the Edge server to be analysed as shown in Figure 5. Furthermore, to set communication between the three nodes another Raspberry Pi that works as a cloud server is installed in one of the local computers and uses MQTT protocol to receive data and store it into one database.

Task Breakdown

The table below shows the breakdown of tasks developed in the project by each team member:

IoT Node	Tasks	Group Member
Temperature & Humidity Sensor	T1: Connect sensors and actuators to Arduino Uno microcontroller	Andrea
	T2: Code sketch in Arduino IDE to allow microcontroller to read data from the temperature and humidity sensor	
	T3: Program the Edge server to receive the data from the microcontroller via serial communication	
	T4: Program the Edge Server to store the received data in a database table	
	T5: Send temperature readings to Cloud server using MQTT protocol	
	T6: Check that data from the edge server and other IoT nodes have been received	
	T7: Program a website to display temperature and humidity data and to control the actuators on the IoT node (e.g., fan speed)	
	T8: Program the cloud computing platform (ThingsBoard) to receive data readings from the edge server and display the status of each sensor in the system	
Soil Moisture Sensor	T1: Connect sensors and actuators to Arduino Uno microcontroller	Thiem
	T2: Code sketch in Arduino IDE to allow microcontroller to read data from the soil moisture sensor	
	T3: Program the Edge Server to receive the soil moisture sensor's readings from the microcontroller via serial communication	

	T4: Send Moisture, Water Level and Pump state reading to cloud using MQTT protocol	
	T6: Add rules on Thingsboard and python script on Edge device to control actuators,	
Light Level Sensor	T1: Connect sensors and actuators to Arduino Uno microcontroller	Autumn
	T2: Code sketch in Arduino IDE to allow microcontroller to read data from the phototransistor and to turn the LED turn on or off based on light sensor reading	
	T3: Program the Edge Server to receive the light sensor's readings from the microcontroller via serial communication	
	T4: Program the Edge Server to store the received data in a database table	
	T5: Send light sensor readings to the Cloud from the microcontroller and Edge Server via MQTT protocol	
	T6: Program the website to display the latest light sensor reading and allow the user to turn on or off the LED remotely	

Project Implementation

Our smart greenhouse project monitors the temperature, humidity, light levels and soil moisture of the plant.

Our first IoT node regulates the temperature inside the greenhouse by switching on a fan automatically when the temperature and humidity levels are high. Meanwhile, our second IoT node measures moisture levels of the soil so when low moisture levels are detected, it waters the plant automatically. And lastly, our third IoT node switches on an LED when the phototransistor detects that the environment is too dark.

Our system helps to maintain an ideal and consistent environmental conditions for the plant in the greenhouse

Hardware Schematic

This section shows the hardware that we used to develop the smart greenhouse system as shown in the table below:

Item Name	Quantity
Arduino UNO Microcontroller	3
Breadboard	3
DHT11 Temperature & Humidity Sensor	1
DC Motor Fan	1
Soil Moisture Sensor	1
Ultrasonic Sensor	1
Water Pump (5V DC Motor)	1
Silicone Tubes	2
LED (Red)	2
LED (Green)	2
LED (Multicolor)	1
Light-dependent Resistor (NPN Phototransistor)	1
560 Ω Resistor	2
220 Ω Resistor	4
Jumper Wires	20
USB 2.0 Cable	3

Table 4: Hardware requirements for smart greenhouse

Software Schematic

The various software we used for this project are:

- Arduino IDE: To write Arduino code (C/C++ language) to control the microcontrollers.
- Oracle VM VirtualBox: For both the Edge server and Cloud which make use of the Raspberry Pi OS
- Thonny Python IDE: To program the edge server to receive data from Arduino, store data to the database and program the websites (using Flask) that display our collected data and control the actuators
- MariaDB (PymySQL): To store data received from sensors.
- Libraries (Flask): Server used to render data on the website including sensors data.
- MQTT protocol: To set the communication between the Edge server and the cloud to send data from the sensors and save it into the database.
- Paho MQTT Python: A Python Library used to subscribe and publish data from the Edge Server to the Cloud
- ThingsBoard Cloud Computing Platform: To display sensors' data as graphs and alarms.

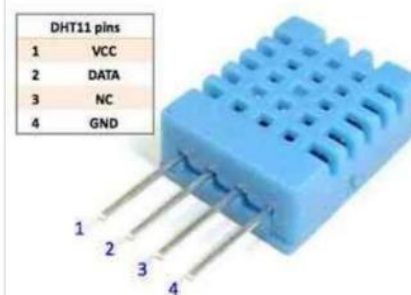
Implementation of IoT Nodes

Temperature & Humidity IoT Node

The Temperature & Humidity IoT Node includes the following sensor, actuators and libraries which support the system operation to automate the use of the fan to regulate the temperature inside of the greenhouse.

1) Sensor

A DHT11 temperature and humidity sensor is connected to an analog pin in the Arduino Uno microcontroller, which interfaces with the PC via USB, to read the temperatures and humidity levels in the greenhouse.



2) Pin connections on the node

Pin Type	Pin Number	Connected to
Analog	A0	DHT11 sensor
Digital	2	Green LED
Digital	4	Red LED
Digital	9	DC Motor Fan

3) Temperature thresholds are programmed in the Arduino sketch (See appendix a) to switch on the fan automatically when the greenhouse environment gets too hot.

Table 1: Temperature controlled fan system

Temperature (C)	Fan Status
≤ 20	OFF
≥ 26 & ≤ 40	ON
> 40	ON

4) LED's actuators allow the IoT system to identify levels of temperature according to pre-determined threshold.

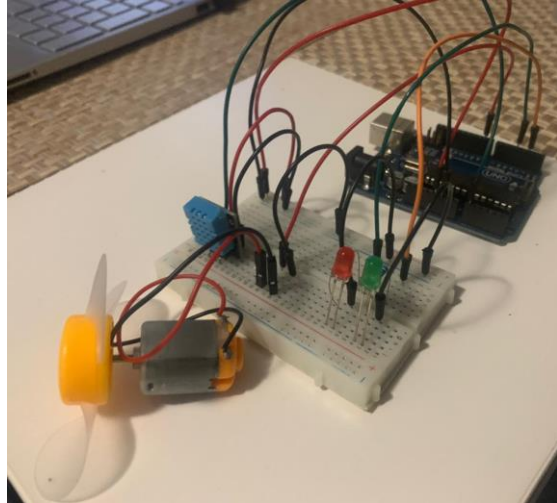
Table 1: Lighting Control System

Temperature (C)	Light Status
≤ 20	OFF
≥ 26 & ≤ 40	ON
> 40	ON

5) The motor fan is automatically on when any of the pre-set thresholds are reached. The user can also adjust the speed of the fan from the website.



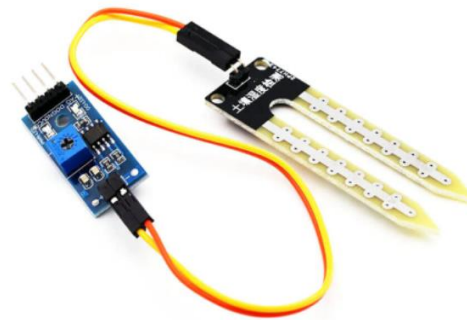
6) Photo of the node



Soil Moisture IoT Node

The Soil Moisture IoT Node includes the following sensor, actuators and libraries which support the system operation to automate the use of a water pump to automatically water the crops inside of the greenhouse:

- 1) Soil moisture sensors measure the water content in the soil estimating the amount of stored water in the soil horizon. Soil moisture sensor measures changes in the soil property that is related to water content.



- 2) Ultrasonic sensor measures the distance of the water levels by emitting ultrasonic sound waves and converts the reflected sound into an electrical signal. In the IoT system it measures the distance to the water level in the water resource making sure there is water reserve.



3) Pin connections on Soil Moisture IoT node

Pin Type	Pin Number	Connected to
Analog	A0	Analog signal leg of Soil moisture sensor
Digital	2	Echo Pin of Ultrasonic Sensor
Digital	3	Pump
Digital	4	Trig pin of Ultrasonic Sensor
Digital	9	LED Green
Digital	10	LED Blue
Digital	11	LED Red

4) Threshold of soil moisture and water pump level

Soil Moisture Threshold	Water Pump
<200	On
>=200	Off

5) Water Threshold and LED status

Water Threshold	LED status
>10	Red
<= 10 && >5	Yellow
<=5	Green

Light Level IoT Node

The Light Level IoT Node includes the following sensor, actuators and libraries which support the system operation to automate the toggling of an LED according to how dark it is in the greenhouse:

1) The light level is read by the NPN phototransistor where its analog signal is sent to the Arduino Uno microcontroller.



2) Pin connections on microcontroller

Pin Type	Pin Number	Connected to
Analog	A0	NPN Phototransistor
Digital	2	Red LED

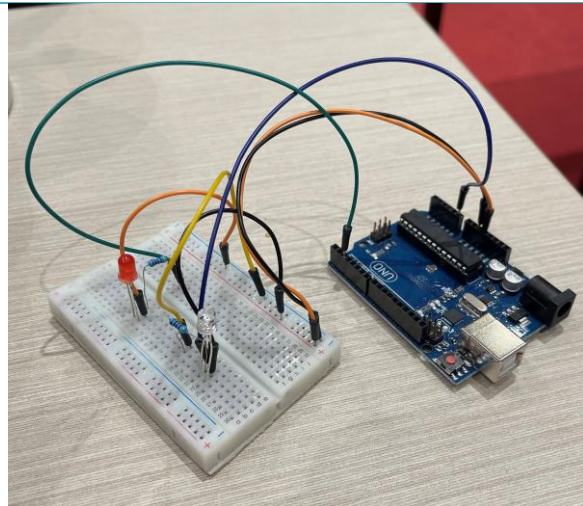
- 3) The light level is compared against a threshold and will cause the LED to turn on or off automatically

Detected Light Level	LED
< 6	On
>= 6	Off

- 4) The LED will turn on or off depending on how dark the greenhouse is



- 5) Photo of the node



Edge Server(s)

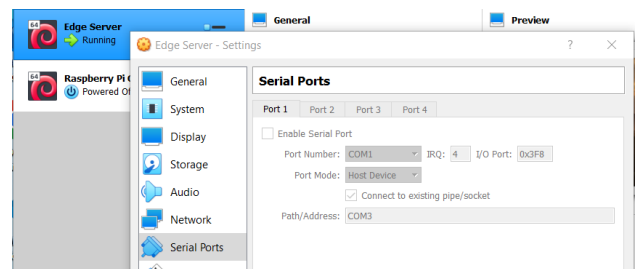
Each IoT node connected to one local computer is communicated with one Edge server Raspberry Pi installed in a VM with Raspbian OS to store and process the data read from the sensors. The following implementations were included in the project

- 1) Configure each Edge server serial port to communicate with Arduino uno microcontroller:

Settings -> Serial Ports ->

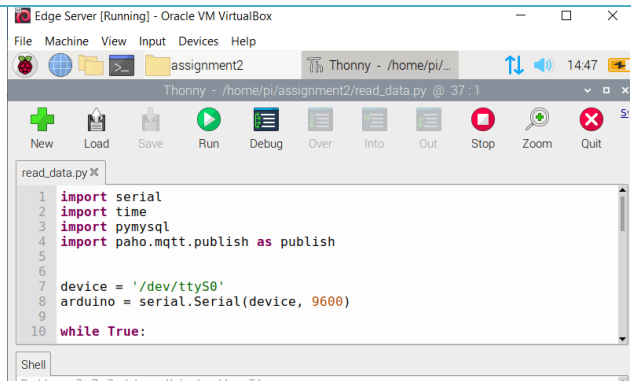
Port 1: Enable Serial Port

Port Number: COM1 (in Raspberry Pi OS /dev/ttyS0)



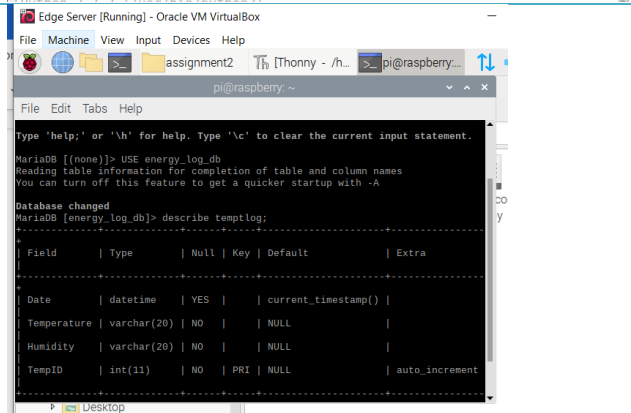
Port Mode: Host Device
Path/Address: COM3

Use python Thonny IDE to read the data of sensor from Arduino uno. To set the communication both systems must set Serial baud rate to 9600. See Appendix b.



```
1 import serial
2 import time
3 import pymysql
4 import paho.mqtt.publish as publish
5
6
7 device = '/dev/ttyS0'
8 arduino = serial.Serial(device, 9600)
9
10 while True:
```

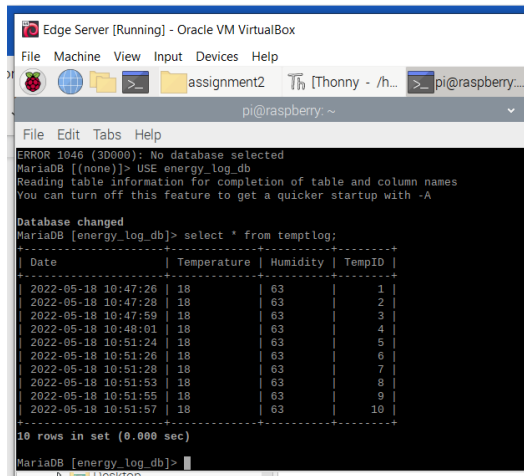
Create database in MySQL and table to store data. See appendix C



```
MariaDB [(none)]> USE energy_log_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [energy_log_db]> describe temptlog;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Date  | datetime | YES | | current_timestamp() | |
| Temperature | varchar(20) | NO | | NULL | |
| Humidity | varchar(20) | NO | | NULL | |
| TempID | int(11) | NO | PRI | NULL | auto_increment |
+-----+-----+-----+-----+-----+-----+
```

Python file is executed so the data is sent to each edge server and saved it in each respective database



```
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> USE energy_log_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [energy_log_db]> select * from temptlog;
+-----+-----+-----+-----+
| Date                | Temperature | Humidity | TempID |
+-----+-----+-----+-----+
| 2022-05-18 10:47:26 | 18          | 63       | 1      |
| 2022-05-18 10:47:28 | 18          | 63       | 2      |
| 2022-05-18 10:47:59 | 18          | 63       | 3      |
| 2022-05-18 10:48:01 | 18          | 63       | 4      |
| 2022-05-18 10:51:24 | 18          | 63       | 5      |
| 2022-05-18 10:51:26 | 18          | 63       | 6      |
| 2022-05-18 10:51:28 | 18          | 63       | 7      |
| 2022-05-18 10:51:53 | 18          | 63       | 8      |
| 2022-05-18 10:51:55 | 18          | 63       | 9      |
| 2022-05-18 10:51:57 | 18          | 63       | 10     |
+-----+-----+-----+-----+
10 rows in set (0.000 sec)

MariaDB [energy_log_db]>
```

*See appendix for full code scripts of each IoT node

Communication Protocols

Now that each Edge server Raspberry Pi set serial communication with Arduino microcontroller, another Raspberry pi in the VM is installed to work as a Cloud server. MQTT is a lightweight, publish-subscribe, machine to machine network protocol designed to send data from each edge server of the greenhouse IoT systems and store the data in the cloud database.

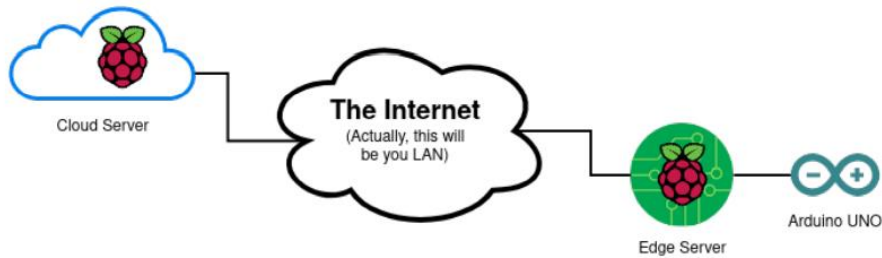


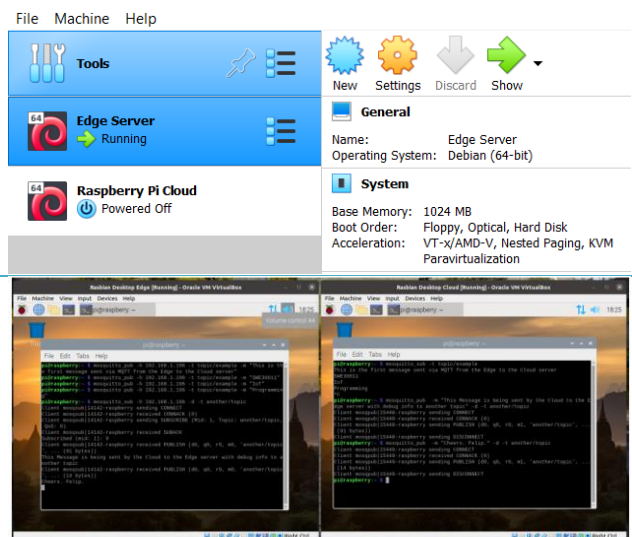
Figure 1. Communication with Raspberry Pi Cloud

Communication protocol MQTT - Implementation

- 1) Full clone of the existing virtual machine with the name Raspberry Pi Cloud.

Installation of MQTT broker in the Cloud and the MQTT client in the Edge server to subscribe and publish respectively messages.

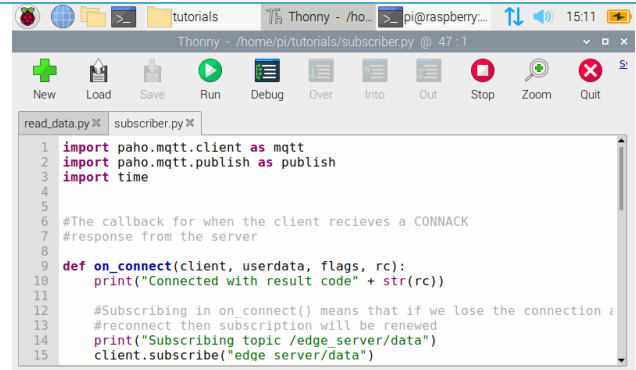
```
sudo apt-get install mosquitto  
mosquitto-clients  
sudo apt-get install mosquitto-clients
```



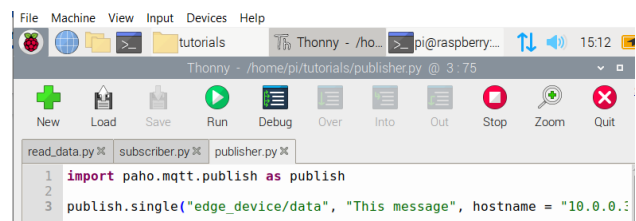
To send the sensor readings from Arduino microcontroller, Eclipse Paho MQTT Python client library is installed in the Edge server and Cloud.

The paho python program in the Cloud subscribes to the topic `edge_device/data` to receive the messages from the Edge Server

The Edge server paho program reads the data by serial communication from the Arduino and publish the message to the Cloud. (Each Edge server connects with the IP address of one Raspberry Pi Cloud to publish the sensor readings)

A screenshot of the Thonny IDE interface on a Raspberry Pi. The window title is 'Thonny - /home/pi/tutorials/subscriber.py @ 47:1'. The code editor shows the following Python code:

```
1 import paho.mqtt.client as mqtt
2 import paho.mqtt.publish as publish
3 import time
4
5
6 #The callback for when the client receives a CONNACK
7 #response from the server
8
9 def on_connect(client, userdata, flags, rc):
10     print("Connected with result code" + str(rc))
11
12     #Subscribing in on_connect() means that if we lose the connection
13     #reconnect then subscription will be renewed
14     print("Subscribing topic /edge_server/data")
15     client.subscribe("edge_server/data")
```

A screenshot of the Thonny IDE interface on a Raspberry Pi. The window title is 'Thonny - /home/pi/tutorials/publisher.py @ 3:75'. The code editor shows the following Python code:

```
1 import paho.mqtt.publish as publish
2
3 publish.single("edge_device/data", "This message", hostname = "10.0.0.1")
```

The MQTT broker installed in the Cloud receives the data from each Edge Server and stores it in the database created previously.

*See appendices A, B, and C for full code scripts of each IoT node

Website

After sending the sensors' data and storing them in the Cloud, we implemented a website to display the latest readings obtained. Some features of the website allow the user to interact with the actuators of the various IoT nodes. The following implementation was made:

Website - Implementation

- 1) Installation of Flask python library to program and run the server which hosts the website

```
# Load the Flask module into your Python script
from flask import Flask

# Create a Flask object called app
app = Flask(__name__)
```



```
sudo apt-get install python-flask
```

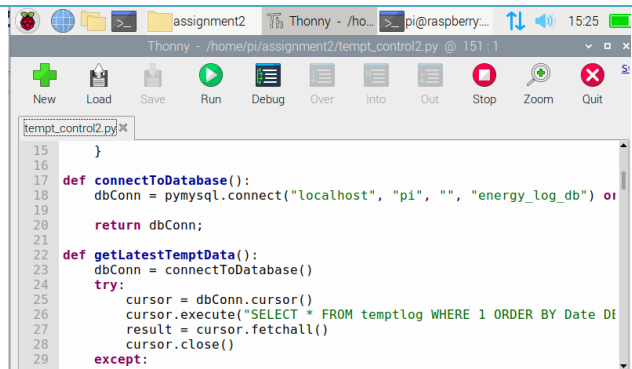
The Python Flask program is developed in the Cloud with data recorded from the Edge Server communication. The data is obtained by querying the database with the last 5 readings recorded.

```
SELECT * FROM greenhouse_log  
WHERE 1 ORDER BY date DESC LIMIT  
5;
```

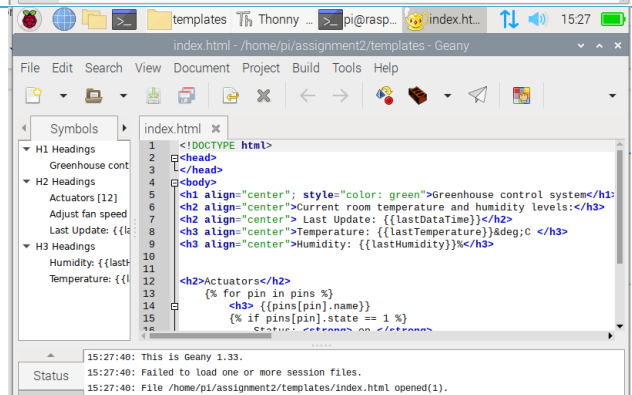
A folder is created "templates" to include the index.html file which structures and displays the information on the website.

When the python flask program is executed, the website displays the current sensors data in the greenhouse.

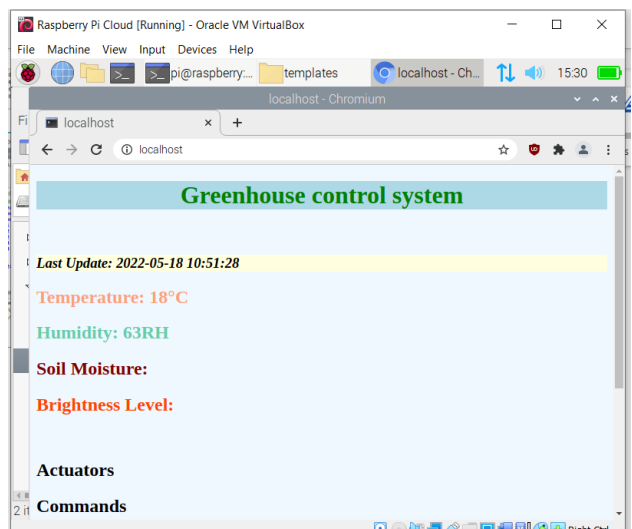
Additional features in the website allow the user to manipulate the system from the website such as Turn on – off a LED or control the fan speed.



```
15 }  
16  
17  
18 def connectToDatabase():  
19     dbConn = pymysql.connect("localhost", "pi", "", "energy_log_db")  
20     return dbConn;  
21  
22 def getLatestTemptData():  
23     dbConn = connectToDatabase()  
24     try:  
25         cursor = dbConn.cursor()  
26         cursor.execute("SELECT * FROM temptlog WHERE 1 ORDER BY Date DESC LIMIT 5")  
27         result = cursor.fetchall()  
28         cursor.close()  
29     except:
```



```
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4 <title>Greenhouse control system</title>  
5 <h1 align="center" style="color: green">Greenhouse control system</h1>  
6 <h2 align="center">Current room temperature and humidity levels:</h2>  
7 <h2 align="center">Last Update: {{lastDataTime}}</h2>  
8 <h3 align="center">Temperature: {{lastTemperature}}deg:C </h3>  
9 <h3 align="center">Humidity: {{lastHumidity}}%</h3>  
10  
11 <h2>Actuators</h2>  
12 {%- for pin in pins %}  
13 <div>  
14 <h3>{{pin.name}}  
15 {%- if pins[pin].state == 1 %}  
16     Status: ON </div>  
17 {%- else %}  
18     Status: OFF </div>  
19 {%- endfor %}
```



*See appendices A, B and C for full code scripts of each IoT node respectively

Cloud Computing

The greenhouse IoT system implements ThingsBoard cloud computing platform for data collection, processing and visualization of the whole system. ThingsBoard is used in a local computer communicated with each Edge server via MQTT protocol. Figure # shows communication architecture with the cloud platform.

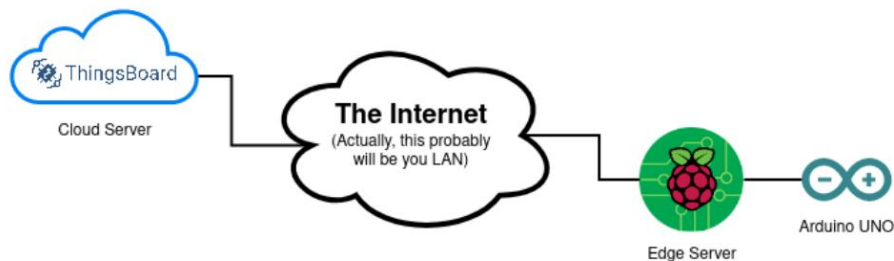


Figure 1. Communication with ThingsBoard Cloud platform

Cloud Computing - Implementation

1) Installation of ThingsBoard in local computer includes:
OpenJDK Java
PostgreSQL

ThingsBoard on premise installation options

Navigation: LIVE DEMO | ON PREMISE | CLOUD

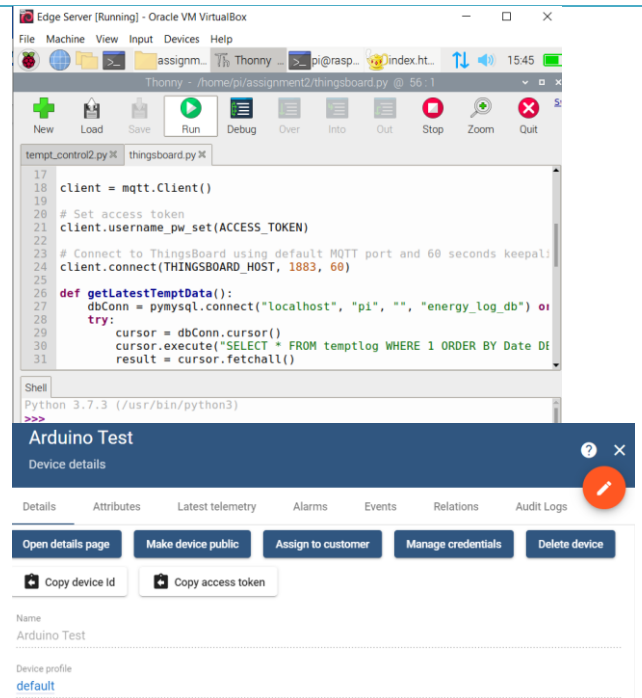
Installation Options:

- ubuntu
- CentOS
- redhat
- Windows
- Raspberry Pi
- docker + Windows
- docker + Apple
- maven
- Building from sources
- Cluster setup

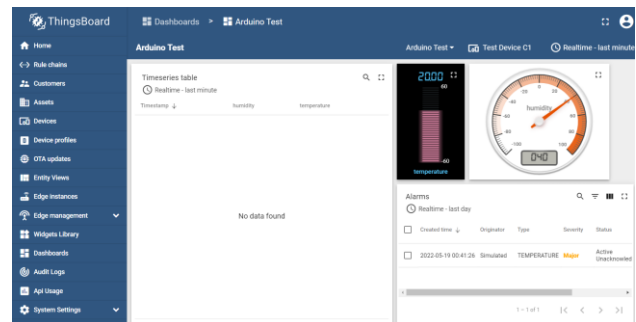
ThingsBoard Dashboard allows to add new devices. Greenhouse device was created

Created time	Name	Device profile	Label	Customer	Public	In gateway	
2022-05-11 18:43:06	Arduino Test	default			<input type="checkbox"/>	<input type="checkbox"/>	Edit Delete
2022-05-11 18:42:11	Charging Port 2	Charging port		Demo Customer	<input type="checkbox"/>	<input type="checkbox"/>	Edit Delete
2022-05-11 18:42:11	Charging Port 1	Charging port		Demo Customer	<input type="checkbox"/>	<input type="checkbox"/>	Edit Delete
2022-05-11 18:42:10	Air Quality Sensor T1	Air Quality Sensor			<input type="checkbox"/>	<input type="checkbox"/>	Edit Delete
2022-05-11 18:42:10	Air Quality Sensor C1	Air Quality Sensor		Demo Customer	<input type="checkbox"/>	<input type="checkbox"/>	Edit Delete

On each Edge server, develop a python program using paho and json library which includes the latest sensor readings data which is published using MQTT protocol to the device created in ThingsBoard using ACCESS TOKEN.



The data is received in ThingsBoard device telemetry in real time and is included in the Dashboard of the device for better visualization



*See appendix for full code scripts of each IoT node

User Manual

The following manual describes the main operations of our smart greenhouse system.

Section	Description
How does the system work?	The smart garden monitors the temperature, humidity, light levels and soil moisture of the plant. It has an automated system that waters the plant when the soil is too dry, regulates the temperature by switching the fan automatically when the temperature and humidity levels are high and switches on the light when the phototransistor detects the environment it too dark
Environment	The greenhouse is a small plastic building which allows regulated climatic conditions to grown crops/plants.
Sensors	Temperature and humidity, soil moisture, ultrasonic and phototransistor are placed inside the greenhouse to determine environmental conditions and regulate them to facilitate the growth of plants.
How to use the smart green house?	Connect the microcontrollers to a PC where Arduino sketch is installed. Upload the program on each IoT node and see current environment status
Run Edge server	Run Edge server from VM Virtual box connected via serial communication with Arduino microcontroller via port COM3
Run python scripts	Execute tempt_control.py file located in Edge server to start the server and the website on localhost webbrowser
Temperature & Humidity	As the greenhouse is a closed building, if the temperature of the environment gets too hot the fan will automatically turn on. Users can monitor current temperatures and adapt fan speed from the website.
Soil Moisture	The sensors are placed in the soil of the greenhouse which determine if the moisture of the soil is too dry, the watering system will water the plants automatically. Users can monito soil moisture levels from the website
Lighting	The phototransistor is placed inside of the greenhouse to detect light levels. In case the environment gets too dark the LED will automatically turn on. Users can monitor and turn on LED from the website
Uses	This IoT system is a small prototype of how IoT devices connected to each other can help automatization in greenhouse environments to facilitate crops growth.

Limitations

The following limitations were found during the development of the IoT system:

Integration of analytics from each IoT node to one website requires proper bidirectional communication between one computer to all IoT nodes.

Current issues presented on the website trigger automatic actions on the system.

Communication with the cloud via MQTT protocol requires an adequate Timestamp between all three IoT devices to avoid saturation of the MQTT broker.

Resources

[1] Mok, X. and Arduino, S., 2022. Smart Garden - Raspberry Pi & Arduino. [online] Hackster.io. Available at: <<https://www.hackster.io/mokxf16/smart-garden-raspberry-pi-arduino-65c7b7>> [Accessed 11 May 2022].

[2] <https://duino4projects.com/automated-greenhouse/>. 2022. For Project. [online] Available at: <<https://duino4projects.com/automated-greenhouse/>> [Accessed 12 May 2022].

[3] Arduino Project Hub. 2022. Automated greenhouse. [online] Available at: <<https://create.arduino.cc/projecthub/kethmal/automated-greenhouse-fccfbf>> [Accessed 13 May 2022].

[4] Wickey, A., 2022. IOT Greenhouse (Embedded Project of the greenhouse concept). [online] Youtube.com. Available at: <<https://www.youtube.com/watch?v=ow1lkOC8fyY>> [Accessed 13 May 2022].

[5] BehrTech. 2022. 4 Benefits of Smart Greenhouses and How to Get Started. [online] Available at: <<https://behrtech.com/blog/4-benefits-of-smart-greenhouses-and-how-to-get-started/>> [Accessed 14 May 2022].

[6] Autogrow. 2022. Greenhouse Automation Systems, Fully Automated Greenhouse Control — Autogrow. [online] Available at: <<https://autogrow.com/your-growing-environment/automated-greenhouse#:~:text=Crops%20can%20be%20fragile%20things,the%20yield%20of%20your%20crops.>> [Accessed 14 May 2022].

[7] Iyer, S., 2022. Why you should adopt a commercial greenhouse automation system. [online] soft WebSolutions. Available at: <<https://www.softwebsolutions.com/resources/commercial-greenhouse-automation-control-system.html>> [Accessed 15 May 2022].

[8] GitHub 2022.

Mosquitto_Subscriber_MySQL_Publisher/Mosquitto_Subscriber_MySQL_Publisher.py at master · ncd-io/Mosquitto_Subscriber_MySQL_Publisher. [online] Available at:

<[https://github.com/ncd-](https://github.com/ncd-io/Mosquitto_Subscriber_MySQL_Publisher/blob/master/Mosquitto_Subscriber_MySQL_Publisher.py#L86)

[io/Mosquitto_Subscriber_MySQL_Publisher/blob/master/Mosquitto_Subscriber_MySQL_Publisher.py#L86](https://github.com/ncd-io/Mosquitto_Subscriber_MySQL_Publisher/blob/master/Mosquitto_Subscriber_MySQL_Publisher.py#L86)<https://ncd.io/log-mqtt-gateway-data-to-mysql-database/>> [Accessed 15 May 2022].

Appendix

A - Source Code for Temperature Sensor Node

Arduino Sketch

```
temp-hum.ino

#include <dht.h>

dht DHT;

//Define analog and digital pins
#define DHT11_PIN A0
int greenLed = 2;
int redLed = 4;
int fan = 8;
String command;

//Set methods to control pin status.
void greenLightOn(){
    digitalWrite(greenLed, HIGH);
}
void greenLightOff(){
    digitalWrite(greenLed, LOW);
}
void redLightOn(){
    digitalWrite(redLed, HIGH);
}
void redLightOff(){
    digitalWrite(redLed, LOW);
}
```



```

void fanOn(){
    digitalWrite(fan, HIGH);
}
void fanOff(){
    digitalWrite(fan, LOW);
}

void setup(){
    //Set baud rate to at 9600 to set serial communication
    Serial.begin(9600);
    //Initialize pins mode
    pinMode(redLed, OUTPUT);
    pinMode(greenLed, OUTPUT);
    pinMode(fan, OUTPUT);
    pinMode(DHT11_PIN, INPUT);
}

void loop(){
    //Read temperature and Humidity values from DHT library
    int chk = DHT.read11(DHT11_PIN);
    int temperature = DHT.temperature;
    int humidity = DHT.humidity;

    //print temperature and humidity to the terminal
    Serial.print("Temperature ");
    Serial.print(temperature);
    //tab key to separate data printed
    Serial.print("\t");
    //Serial.print("Humidity ");
    Serial.print(humidity);
    Serial.println("\t");
    delay(1000);

    //set 1st threshold between 20C - 22C green LED on
    if(temperature >= 20 && temperature <= 22 ){
        greenLightOn();
        redLightOff();
        fanOff();
    }
    //set 2st threshold between 22C - 24C green LED off - Red LED on
    else if(temperature > 22 && temperature <= 24){
        greenLightOff();
        redLightOn();
        fanOff();
    }
}

```

```

//set 3st threshold greater than 24C green LED off - Red LED on - Fan on
else if(temperature >= 24){
    redLightOn();
    fanOn();
    greenLightOff();

} else {

//Temperature levels normal

    greenLightOff();
    redLightOff();
    fanOff();
}

```

MQTT communication - Database

Subscriber

```

1  import paho.mqtt.client as mqtt
2  import serial
3  import time
4  import pymysql
5  import sys
6  import json
7
8
9  def on_connect(client, userdata, flags, rc):
10     print ("Connected with result code" +str(rc))
11
12     client.subscribe("edge_device/data")
13     try:
14         dbConn = pymysql.connect("localhost", "pi", "", "energy_log_db")
15     except:
16         dbConn.close()
17
18
19  def log_data (dbConn,payload):
20     cursor = dbConn.cursor()
21     #cursor.execute("INSERT INTO tempt_log (temperature) VALUES (%s)" %
22     #cursor.execute("INSERT INTO humidity_log (humidity) VALUES (%s)" %
23     #cursor.execute("INSERT INTO light_log (light_log) VALUES (%s)" %
24     cursor.execute("INSERT INTO soil_log (soil_moisture) VALUES (%s)" %
25     dbConn.commit()
26
27
28
29  def on_message(client, userdata, msg):
30     print(msg.topic+" "+str(msg.payload))
31     payload = str(msg.payload)
32     dbConn = pymysql.connect("localhost", "pi", "", "energy_log_db")
33     log_data(dbConn,payload)
34     dbConn.close()

```

```

35
36
37 client = mqtt.Client()
38 client.on_connect = on_connect
39 client.on_message = on_message
40
41 client.connect("172.20.10.8", 1883, 60)
42
43 client.loop_forever()

```

Website

subscriber-new.py % tempt_control2.py %

```

1  import serial
2  import time
3  import pymysql
4  from flask import Flask, render_template, request
5  import Adafruit_DHT as dht
6
7
8  app = Flask(__name__)
9
10 #Dictionary of digital pins
11 pins = {
12     2 : {'name' : 'Green LED', 'state' : 0},
13     4 : {'name' : 'Red LED', 'state' : 0},
14     7 : {'name' : 'FAN', 'state' : 0}
15 }
16
17 def connectToDatabase():
18     dbConn = pymysql.connect("localhost", "pi", "", "energy_log_db") or
19
20     return dbConn;
21
22 def getLatestTempData():
23     dbConn = connectToDatabase()
24     try:
25         cursor = dbConn.cursor()
26         cursor.execute("SELECT * FROM tempt_log WHERE 1 ORDER BY Date I
27         result = cursor.fetchall()
28         cursor.close()
29     except:
30         cursor.close()
31         result = False;
32
33     return result
34
35 def getLatestHumiData():
36     dbConn = connectToDatabase()

```

```

37         try:
38             cursor = dbConn.cursor()
39             cursor.execute("SELECT * FROM humidity_log WHERE 1 ORDER BY Date")
40             result = cursor.fetchall()
41             cursor.close()
42         except:
43             cursor.close()
44             result = False;
45
46         return result
47
48     def getLatestLightData():
49         dbConn = connectToDatabase()
50         try:
51             cursor = dbConn.cursor()
52             cursor.execute("SELECT * FROM light_log WHERE 1 ORDER BY date")
53             result = cursor.fetchall()
54             cursor.close()
55         except:
56             cursor.close()
57             result = False;
58
59         return result
60
61     def getLatestSoilData():
62         dbConn = connectToDatabase()
63         try:
64             cursor = dbConn.cursor()
65             cursor.execute("SELECT * FROM soil_log WHERE 1 ORDER BY date")
66             result = cursor.fetchall()
67             cursor.close()
68         except:
69             cursor.close()
70             result = False;
71
72         return result
73
74     @app.route("/")
75     def data ():
76         #data send to index.html
77         latestData = getLatestTempData()
78         lastDataTime = latestData[0][0]
79         lastTemperature = latestData[0][1]
80         latestData = getLatestHumiData()
81         lastHumidity = latestData[0][1]
82         latestData = getLatestLightData()
83         latestLightData = latestData[0][1]
84         latestData = getLatestSoilData()
85         latestSoilData = latestData[0][1]
86
87
88         return render_template ('index.html', lastDataTime = lastDataTime,
89
90     @app.route("/pins")
91

```

```

92 def pinsData ():
93     templateData = {'pins' : pins}
94     return render_template ('index.html', **templateData)
95
96
97
98
99
100 @app.route('/<request>/', methods=['GET', 'POST'])
101
102 def request():
103     if request.method == "POST":
104         speed = request.form.get("speed")
105         ser.write(str(val).encode('ascii'))
106
107     templateData = {'pins' : pins}
108     return render template ('index.html')
109
110
111
112 @app.route("/<pin>/<state>")
113
114 def state(pin, state):
115
116     temperature = ''
117     humidity = ''
118
119     if pin == "dhtpin" and state == "get":
120         tempt, humdt = dht.read_retry(dht.DHT11, 8)
121         tempt = '{0:0.1f}'.format(tempt)
122         humdt = '{0:0.1f}'.format(humdt)
123         temperature = 'Temperature: ' + tempt
124         humidity = 'Humidity: ' + humdt
125
126     temptData = {
127         'temperature' : temperature,
128         'humidity' : humidity
129     }
130
131
132
133
134
135
136
137
138 if __name__ == '__main__':
139     #ser = serial.Serial('/dev/ttyS0', 9600, timeout = 1)
140     #ser.flush()
141     app.run(debug = True, host = '0.0.0.0' , port = 8080)
142
143

```

thingsboard.py ✕

```
1 import os
2 import time
3 import sys
4 import pymysql
5 import paho.mqtt.client as mqtt
6 import json
7
8 THINGSBOARD_HOST = 'demo.thingsboard.io'
9 ACCESS_TOKEN = 'Pnki0dkz25Lh3utRpbbD'
10
11 # Data capture and upload interval in seconds.
12 INTERVAL = 2
13
14 sensor_data = {'temperature': 0, 'humidity': 0}
15
```

thingsboard.py ✕

```
16 next_reading = time.time()
17
18 client = mqtt.Client()
19
20 # Set access token
21 client.username_pw_set(ACCESS_TOKEN)
22
23 # Connect to ThingsBoard using default MQTT port and 60 seconds keepalive
24 client.connect(THINGSBOARD_HOST, 1883, 60)
25
26 def getLatestTempData():
27     dbConn = pymysql.connect("localhost", "pi", "", "energy_log_db") or
28     try:
29         cursor = dbConn.cursor()
30         cursor.execute("SELECT * FROM temptlog WHERE 1 ORDER BY Date DESC")
```

thingsboard.py ✕

```
31         result = cursor.fetchall()
32         cursor.close()
33     except:
34         cursor.close()
35         result = False;
36
37     return result
38
39 client.loop_start()
40
41 try:
42     while True:
43
44         latestData = getLatestTempData()
45         temperature = latestData[0][1]
```

thingsboard.py

```
46     humidity = latestData[0][2]
47
48     sensor_data['temperature'] = temperature
49     sensor_data['humidity'] = humidity
50
51     # Sending humidity and temperature data to ThingsBoard
52     client.publish('v1/devices/me/telemetry', json.dumps(sensor_data))
53
54     next_reading += INTERVAL
55     sleep_time = next_reading - time.time()
56     if sleep_time > 0:
57         time.sleep(sleep_time)
58 except KeyboardInterrupt:
59     pass
60
61 client.loop_stop()
62 client.disconnect()
```

