Student 1: Nguyen Hoang Nguyen

Student ID: 104175342

Student 2: Cao Xuan Bach

Student ID: 103525470

Group project

GREEN FARM

Course:

SWE30011 – IoT Programming

Instructor's Name: Pham Van Dai

Submission Date: 7 April 2023

Table of contents

I. Introduction

1.1. Background

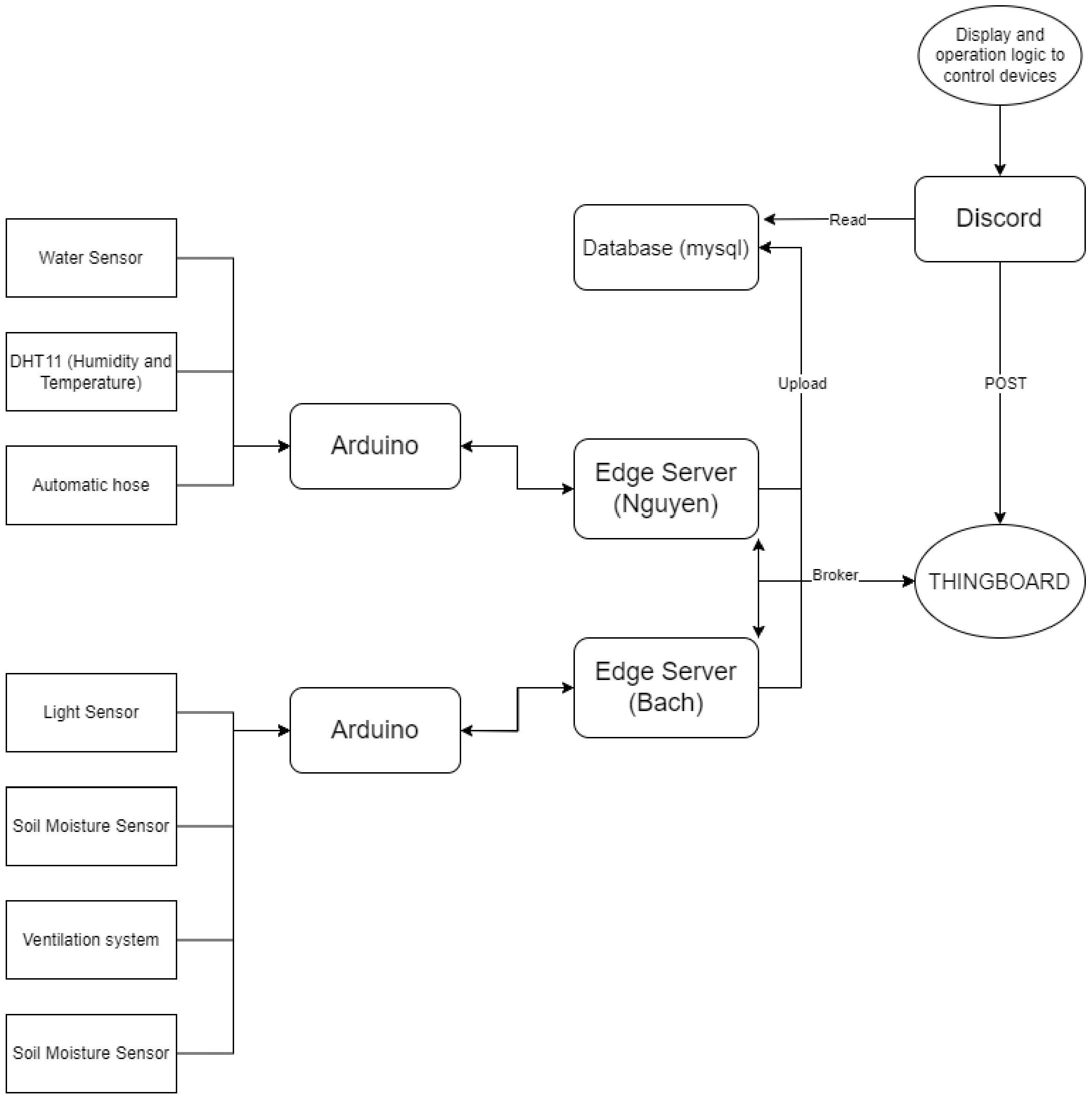
In today’s world, the Internet of Things (IoT) is proving to be noticeably impactful and incredibly useful for our daily lives as it connects almost everything around us to the internet, enabling the ability to view data collected, remote control, and automation. This interconnected network of devices has transformed numerous industries, from healthcare to agriculture, by giving a lot of new insights and efficiency increases. This project aims to improve farming performance by developing an automated watering system based on the soil moisture levels and the weather conditions in the farm area. This design not only optimizes water consumption but also enhances crop yields and reduces manual involvement.

1.2. Proposed system

The proposed system involves the usage of two Arduino Uno, 2 Edge devices, and four sensors: DHT11 to measure temperature and humidity, a soil moisture sensor to measure the current soil’s condition, a rain sensor to check whether it is raining or not, and a light sensor to detect the intensity of the light. All of the data collected will be sent through the edge device and onto the database, that database, simultaneously, will connect with an IoT Cloud system – ThingsBoard to display the data collected as charts and tables, also allow us to control the system remotely from the dashboard itself. Automation is also included, as the system will automatically water the plants by integrating with a solenoid valve and water pump (for simplicity, an LED will be used for demonstration). The design has an alarm system that notifies the user through Discord Bot if certain conditions are met.

II. Conceptual Design

- UML



III. Tasks Breakdown

Task Descriptions	Responsible Person
Code Arduino system with Water sensor and DHT11 with necessary display systems (if have any)	Nguyen
Code Arduino system with Light sensor and Soil Moisture sensor with necessary display systems (if have any)	Bach
Connect Arduino to Edge device and upload received data to the database and ThingsBoard (Water sensor and DHT11)	Nguyen
Connect Arduino to Edge device and upload received data to the database and ThingsBoard (Light sensor and Soil Moisture sensor)	Bach
Code a discord bot with display and operation logic to control devices	Nguyen, Bach
Create dashboard on ThingsBoard with necessary display	Nguyen
Report	Bach

IV. Implementation

1. Sensor:

1.1 DHT11 Temperature and Humidity Sensor (Digital):

- Measures temperature and humidity.
- Provide insights into the current environmental conditions.
- Integrated with Arduino for data collection. Arduino reads the digital output from the sensor, which provides temperature and humidity values. Then virtual Raspberry Pi reads the serial output from Arduino and saves the data to the database and ThingsBoard and displays those data on the ThingsBoard's dashboard. Discord bot also uses the same data for operation logic.

1.2. Water sensor (Analog):

- Detect the presence of water and measure how wet the area is.

- Provide insights into the current environmental conditions.

- Integrated with Arduino for data collection. Arduino reads the analog output from the sensor, which tells whether there is water in the area or not and how much water there is (sort of). Then virtual Raspberry Pi reads the serial output from Arduino and saves the data to the database and ThingsBoard and displays those data on the ThingsBoard's dashboard. Discord bot also uses the same data for operation logic.

1.3. Light sensor (Analog):

- Detect the presence of light and measure the intensity of light.

- Provide insights into the current environmental conditions.

- Integrated with Arduino for data collection. Arduino reads the analog output from the sensor, which tells whether there is light in the area and how strong the light is. Then virtual Raspberry Pi reads the serial output from Arduino and saves the data to the database and ThingsBoard and displays those data on the ThingsBoard's dashboard. Discord bot also uses the same data for operation logic.

1.4. Soil moisture sensor (Analog):

- Measure how wet the soil is.

- Provide insights into the current environmental conditions.

- Integrated with Arduino for data collection. Arduino reads the analog output from the sensor, which tells how much moisture there is in the soil. Then virtual Raspberry Pi reads the serial output from Arduino and saves the data to the database and ThingsBoard and displays those data on the ThingsBoard's dashboard. Discord bot also uses the same data for operation logic.

-2. Actuator:

+ LED(s):

- Act as indicators.

- Controlled remotely through the discord and ThingsBoard's dashboard.

- LEDs indicators for system status and act as simple replacements for the ventilation system and automatic hose.

- Connected to Arduino for control. LED can be controlled from Arduino Serial. Raspberry Pi connects through virtual box and uses Python code to remotely control and display status through the discord bot and ThingsBoard's dashboard. LED also has auto control based on the data collected in the database.

3. Communication protocol:

+ MQTT:

- Facilitates real-time communication between sensor nodes, actuators, and central processing unit.
- MQTT brokers manage message distribution.
- Sensor nodes publish data to MQTT topics.
- Actuators subscribe to control commands.
- Enables interaction with ThingsBoard dashboard and Discord bot.

4. API or website detail:

+ Discord Bot:

- Enables remote monitoring and control of the IoT system via Discord.
- Provides updates on environmental conditions. Allows remote control of actuators like LEDs. Sends alerts based on thresholds or events.
- Developed using Python for Discord's API. Subscribes to MQTT topics for data and control.
- Fetches data from ThingsBoard for user access.

5. Cloud Computing

+ Thingsboard (demo.thingsboard):

- Acts as the central platform for data visualization, analytics, and device management.
- Hosted on Thingsboard's cloud platform (demo.thingsboard.io). Receives data from sensors via MQTT for visualization and storage. Allows for the creation of automation rules based on incoming data.
- Can send commands back to the IoT system for actuator control or automation triggers.

6. Software/Libraries:

+ Arduino IDE:

- Used for programming the Arduino board, enabling the implementation of sensor readings and actuator control logic.

+ DHT Library:

- Utilized for interfacing with the DHT11 sensor, simplifying the process of reading temperature and humidity values.

+ MySQL Database:

- Data from sensors are stored in a database for further analysis or historical tracking.

+ Python:

- Used for scripting Discord bot commands, MQTT communication, database interaction, and data exchange with Thingsboard.

- Integrates with Discord API, MQTT, MySQL database, and Thingsboard for system control and data management.

- Manages data exchange between sensors, database, and Thingsboard, executes automation tasks, and facilitates user interaction via Discord.

V. User Manual

Initially, users will be provided with a link facilitating the addition of the Discord bot to their server. Subsequently, they are instructed to utilize the command "!set_channel_warning" followed by the respective channel ID to designate the channel responsible for issuing warnings. Similarly, the command "!set_channel_lastest_data" followed by the corresponding channel ID is employed to assign the channel responsible for disseminating the latest data.

For a more granular retrieval of information, users may utilize the commands "!query_data" or "!query_saved_data" appended with the desired data type. This action facilitates the retrieval of either the most recent data or the historical data spanning the preceding 30 days. As an illustrative example, the commands "!query_data lights" and "!query_saved_data lights" would respectively procure the latest information pertaining to lights or retrieve the historical data concerning lights within the last 30 days.

VI. Limitations

Our group met some difficulties during the development of the system. First of all, we originally wanted to do two Arduino system on two laptops and use them as edge devices, but we had a lot of small errors during the session and changed to using two virtual box machines as edge devices for simplicity. Additionally, coding for discord bot and ThingsBoard took too much time and had too many errors for the original and more complex features, therefore we had to simplify the features so we can finish the project in time for deadline. Lastly, we did not the necessary tools for the full functional system, as we were missing the ventilation system and automatic hose. They were decently expensive and difficult to buy and our budget could not make up for it.

VII. Resources

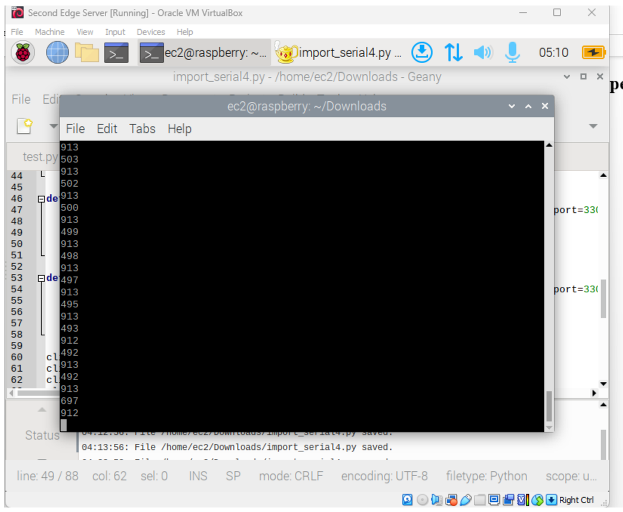
Tutorial week 6: login.microsoftonline.com. (n.d.). *Redirecting.* [online] Available at: https://swinburne.instructure.com/courses/56603/files/30144845?module_item_id=4120845

Tutorial week 7: login.microsoftonline.com. (n.d.). *Redirecting.* [online] Available at: https://swinburne.instructure.com/courses/56603/files/27898160?module_item_id=3810616

Tutorial week 8: login.microsoftonline.com. (n.d.). *Redirecting.* [online] Available at: https://swinburne.instructure.com/courses/56603/files/27898165?module_item_id=3810630

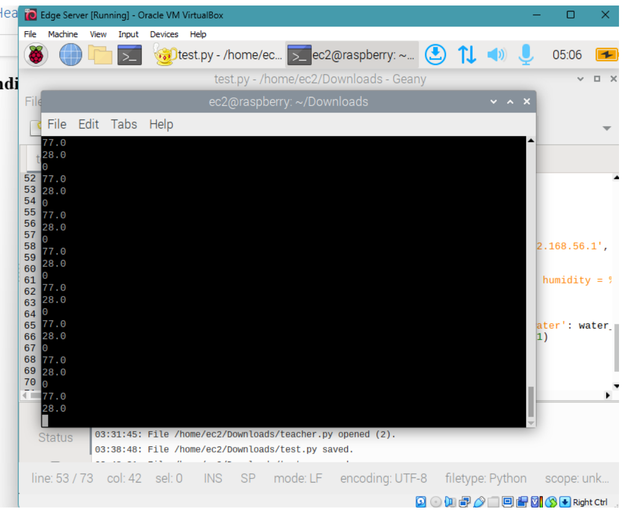
Tutorial week 9: login.microsoftonline.com. (n.d.). *Redirecting.* [online] Available at: https://swinburne.instructure.com/courses/56603/files/27898134?module_item_id=3810639

VIII. Appendix



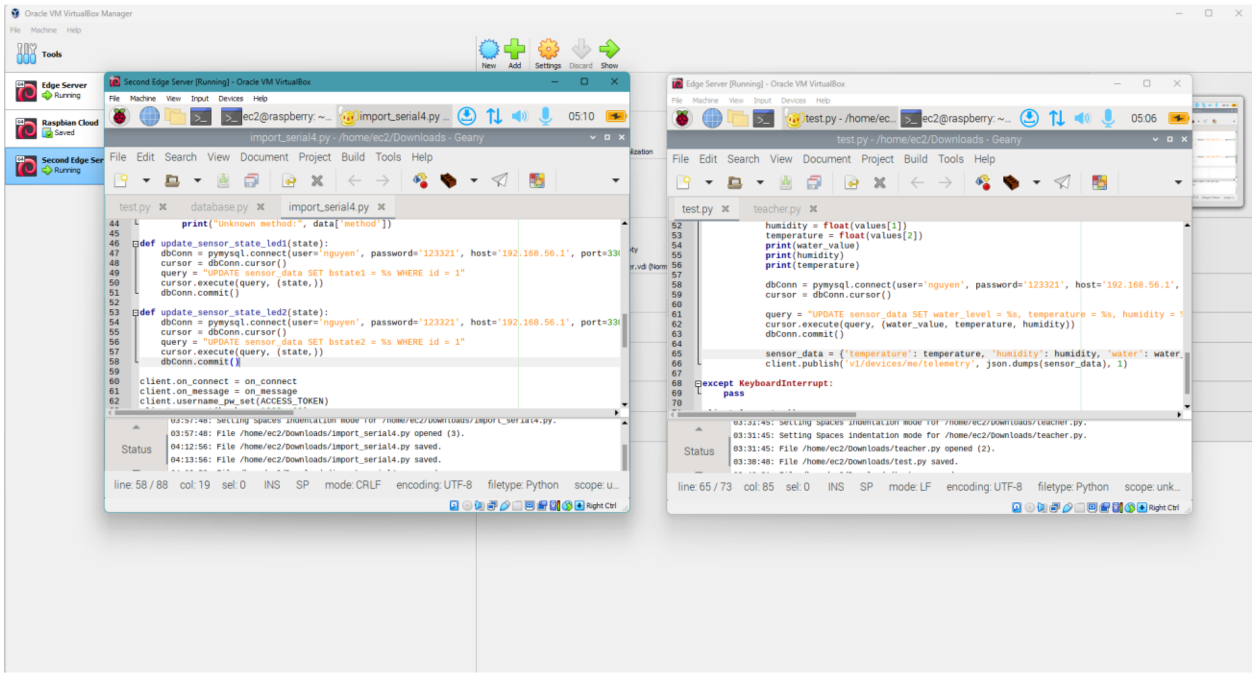
```
913
503
913
502
913
500
913
499
913
498
913
497
913
495
913
493
912
492
913
492
913
697
912
```

```
04:13:56: File /home/ec2/Downloads/import_serial4.py saved.
```



```
77.0
28.0
0
77.0
28.0
0
77.0
28.0
0
77.0
28.0
0
77.0
28.0
0
77.0
28.0
0
77.0
28.0
0
77.0
28.0
```

```
03:31:45: File /home/ec2/Downloads/teacher.py opened (2).
03:38:48: File /home/ec2/Downloads/test.py saved.
```



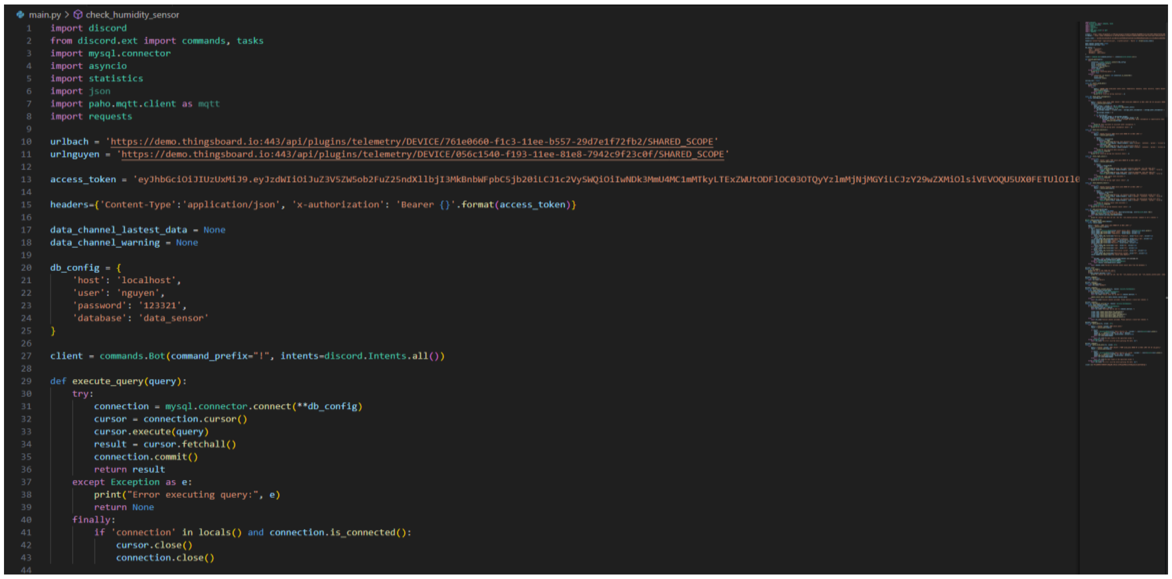
```
        print("Unknown method:", data['method'])

def update_sensor_state_led1(state):
    dbConn = pymysql.connect(user='nguyen', password='123321', host='192.168.56.1', port=33
    cursor = dbConn.cursor()
    query = "UPDATE sensor_data SET bstate1 = %s WHERE id = 1"
    cursor.execute(query, (state,))
    dbConn.commit()

def update_sensor_state_led2(state):
    dbConn = pymysql.connect(user='nguyen', password='123321', host='192.168.56.1', port=33
    cursor = dbConn.cursor()
    query = "UPDATE sensor_data SET bstate2 = %s WHERE id = 1"
    cursor.execute(query, (state,))
    dbConn.commit()

client.on_connect = on_connect
client.on_message = on_message
client.username_pw_set(ACCESS_TOKEN)
```

```
03:57:48: File /home/ec2/Downloads/import_serial4.py opened (3).
04:12:56: File /home/ec2/Downloads/import_serial4.py saved.
04:13:56: File /home/ec2/Downloads/import_serial4.py saved.
```

```
            humidity = float(values[1])
            temperature = float(values[2])
            print(water_value)
            print(humidity)
            print(temperature)

            dbConn = pymysql.connect(user='nguyen', password='123321', host='192.168.56.1',
            cursor = dbConn.cursor()

            query = "UPDATE sensor_data SET water_level = %s, temperature = %s, humidity = 
            cursor.execute(query, (water_value, temperature, humidity))
            dbConn.commit()

            sensor_data = {'temperature': temperature, 'humidity': humidity, 'water': water_
            client.publish('v1/devices/me/telemetry', json.dumps(sensor_data), 1)

except KeyboardInterrupt:
    pass
```

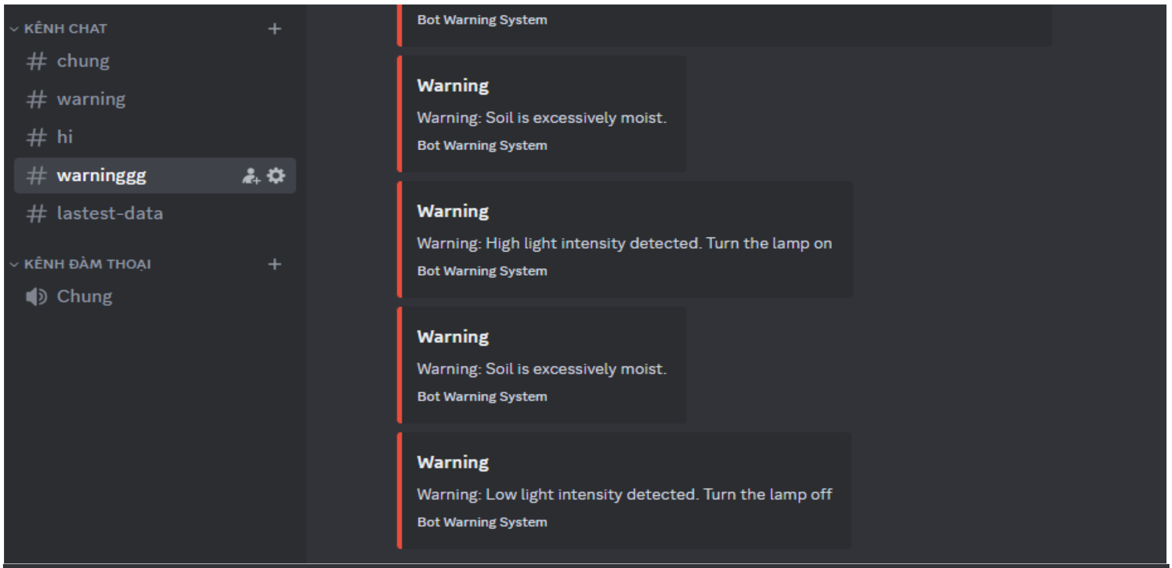
```
03:31:45: Setting Spaces indentation mode for /home/ec2/Downloads/teacher.py.
03:31:45: File /home/ec2/Downloads/teacher.py opened (2).
03:38:48: File /home/ec2/Downloads/test.py saved.
```

2 Edge Devices with their code and result after run



```python
import discord
from discord.ext import commands, tasks
import mysql.connector
import asyncio
import statistics
import json
import paho.mqtt.client as mqtt
import requests

urlbach = 'https://demo.thingsboard.io:443/api/plugins/telemetry/DEVICE/761e0660-f1c3-11ee-b557-29d7e1f72fb2/SHARED_SCOPE'
urlnguyen = 'https://demo.thingsboard.io:443/api/plugins/telemetry/DEVICE/056c1540-f193-11ee-81e8-7942c9f23c0f/SHARED_SCOPE'

access_token = 'eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJuZ3V5ZW5ob2FuZ25ndXllbjI3MkBnbWFpbC5jb20iLCJ1c2VySWQiOiIwNDk3MmU4MC1mMTkyLTExZWUtODFlOC03OTQyYzlmMjNjMGYiLCJzY29wZXMiOlsiVEVOQU5UX0FETUlOIl0

headers={'Content-Type':'application/json', 'x-authorization': 'Bearer {}'.format(access_token)}

data_channel_lastest_data = None
data_channel_warning = None

db_config = {
    'host': 'localhost',
    'user': 'nguyen',
    'password': '123321',
    'database': 'data_sensor'
}

client = commands.Bot(command_prefix="!", intents=discord.Intents.all())

def execute_query(query):
    try:
        connection = mysql.connector.connect(**db_config)
        cursor = connection.cursor()
        cursor.execute(query)
        result = cursor.fetchall()
        connection.commit()
        return result
    except Exception as e:
        print("Error executing query:", e)
        return None
    finally:
        if 'connection' in locals() and connection.is_connected():
            cursor.close()
            connection.close()
```

Discord Code



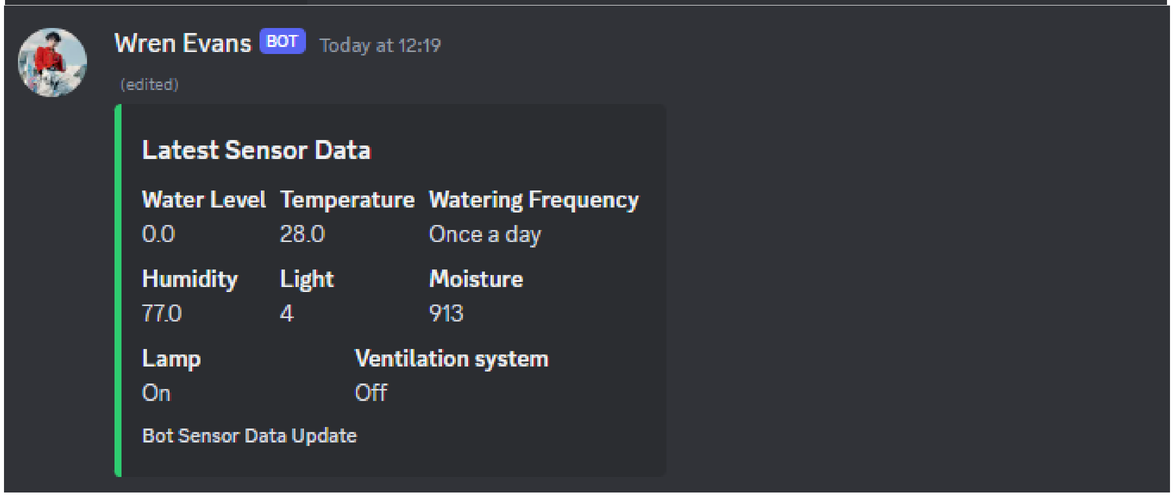
Bot Warning System

Warning
Warning: Soil is excessively moist.
Bot Warning System

Warning
Warning: High light intensity detected. Turn the lamp on
Bot Warning System

Warning
Warning: Soil is excessively moist.
Bot Warning System

Warning
Warning: Low light intensity detected. Turn the lamp off
Bot Warning System

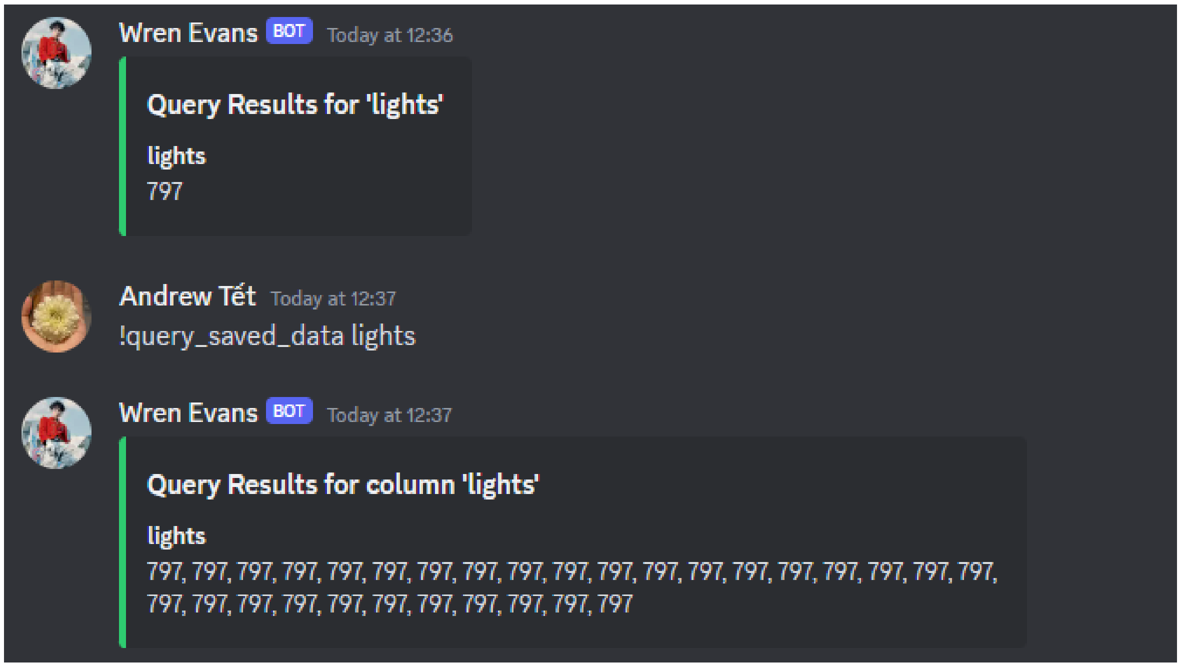


Wren Evans BOT Today at 12:19
(edited)

Latest Sensor Data

Water Level	Temperature	Watering Frequency
0.0	28.0	Once a day
Humidity	**Light**	**Moisture**
77.0	4	913
Lamp	**Ventilation system**	
On	Off	

Bot Sensor Data Update



Wren Evans BOT Today at 12:36

Query Results for 'lights'

lights
797

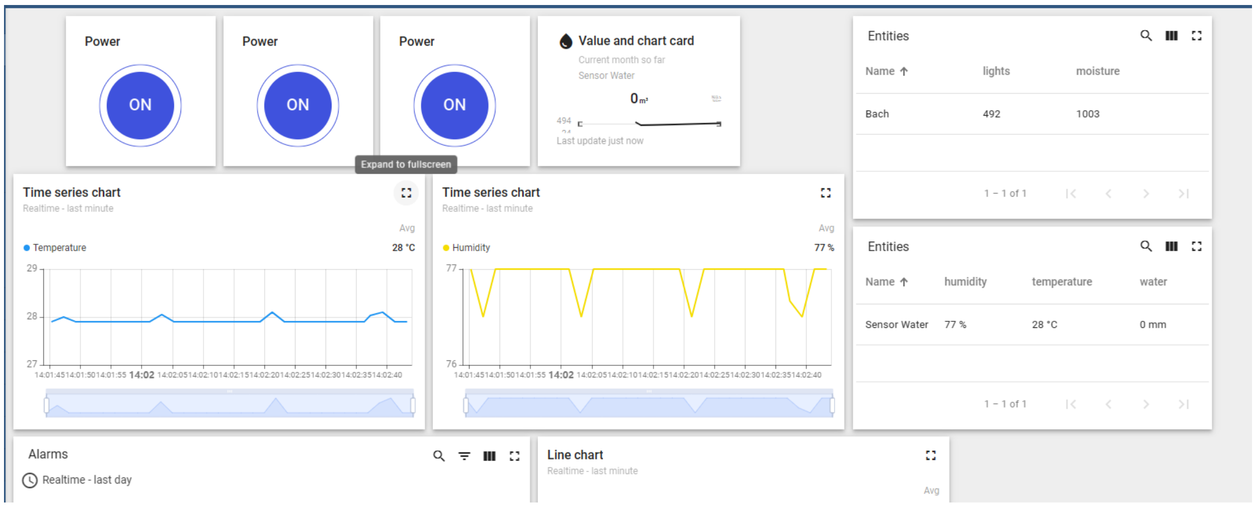
Andrew Tết Today at 12:37
!query_saved_data lights

Wren Evans BOT Today at 12:37

Query Results for column 'lights'

lights
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797

Discord Result



Entities

Name ↑	lights	moisture
Bach	492	1003

1 – 1 of 1

Entities

Name ↑	humidity	temperature	water
Sensor Water	77 %	28 °C	0 mm

1 – 1 of 1

Thing board