SWE30011 - IoT Programming

Smart home monitor and security system

Student 1 name: Trac Duc Anh Luong
Student 1 ID: 103488117

Student 2 name: Dao Dung Tri Luu
Student 2 ID: 103488007

Link to video:
https://drive.google.com/file/d/1q9S8ikxFQl3uK8WLUEAe0W2KqSiNgu3I/view?usp=sharing

Introduction

As technological advances are becoming ubiquitous, the concept of smart home has become more popular and adaptable for households all around the world. The components that made homes smart are Internet of Things (IoT) devices installed to automate tasks and control various systems around the house, from light and temperature control to security protocols. For this project, our goal is to develop a smart home system for sensing and security by using IoT nodes and edge devices. By using sensors, a webcam, cloud servers, and a user interface, the homeowner can not only monitor in-door environment statistics (temperature, humidity, gas level) but also detect and alert real-time security threats to increase safety in their home.

Conceptual Design

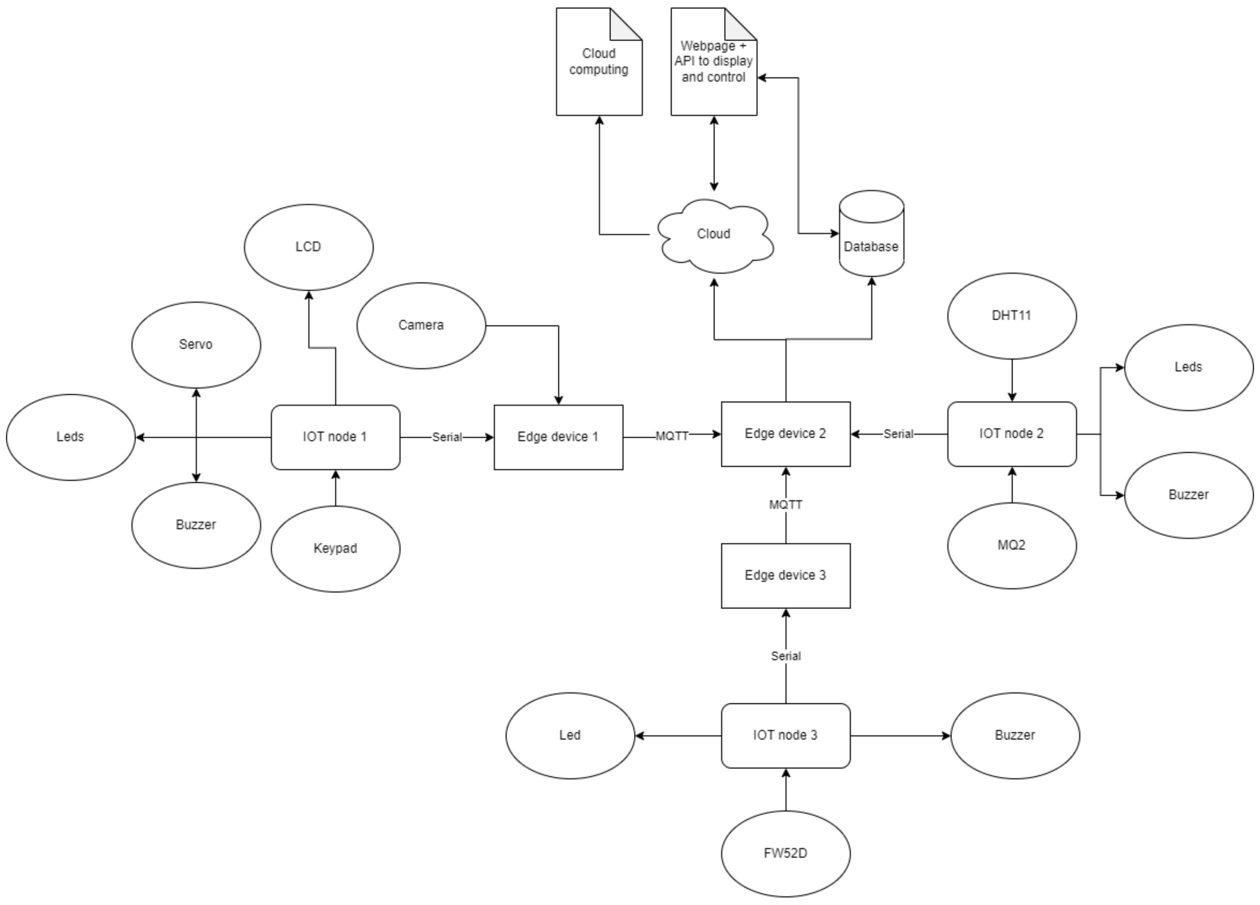


Figure 1: The overall system

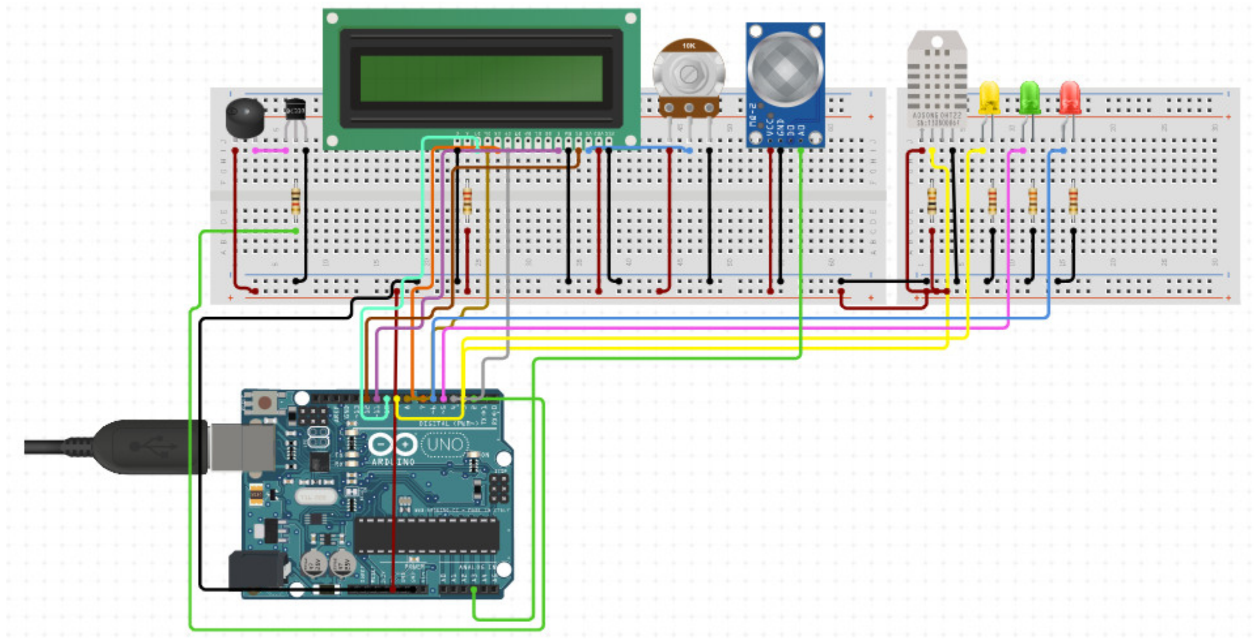


Figure 2: The weather monitor node

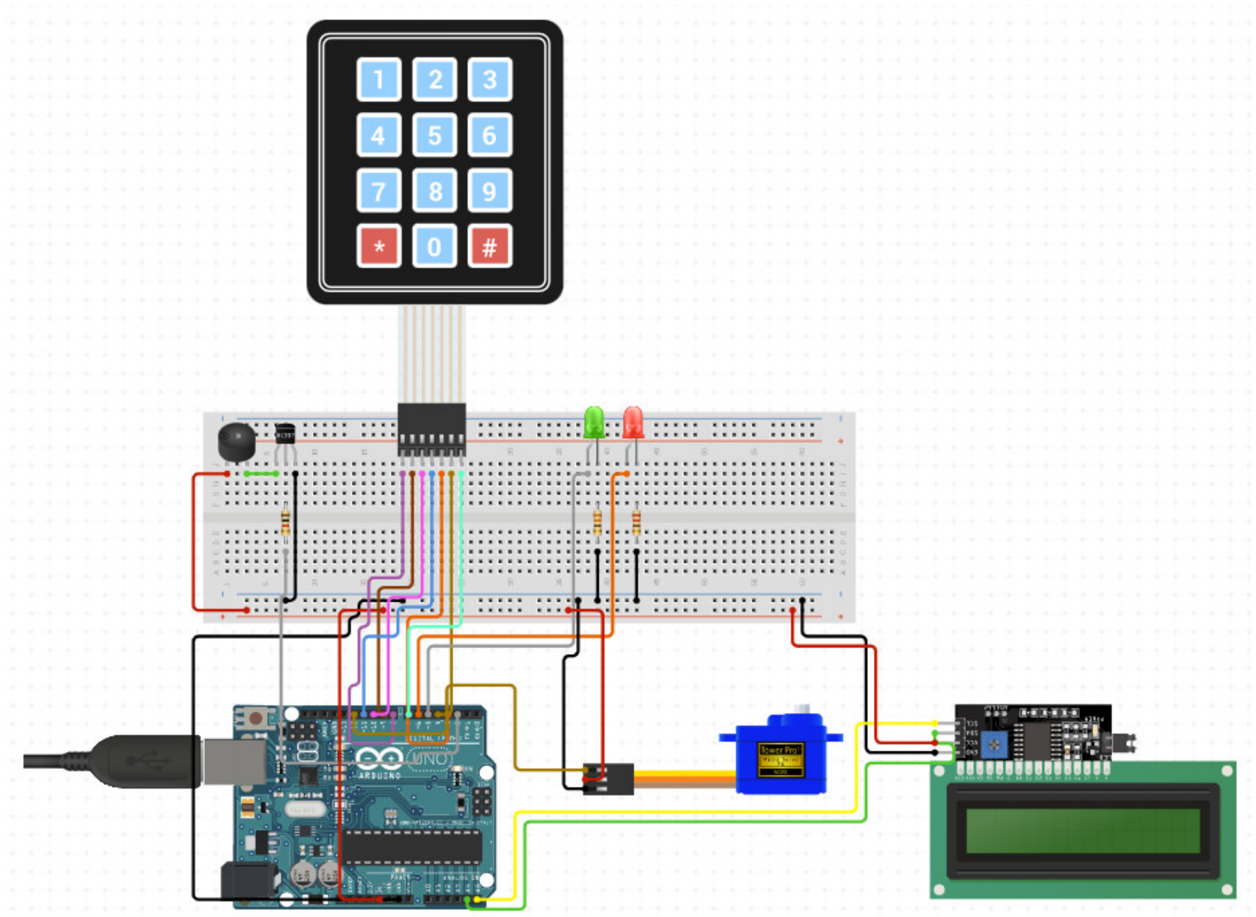


Figure 3: The gate(door) security node

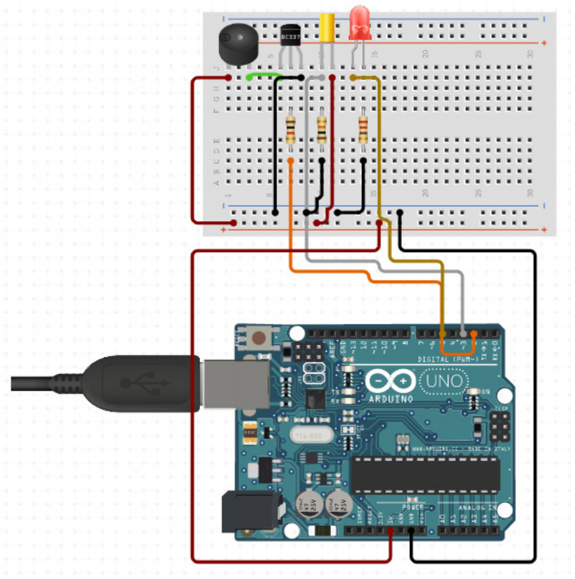


Figure 4: The window security node

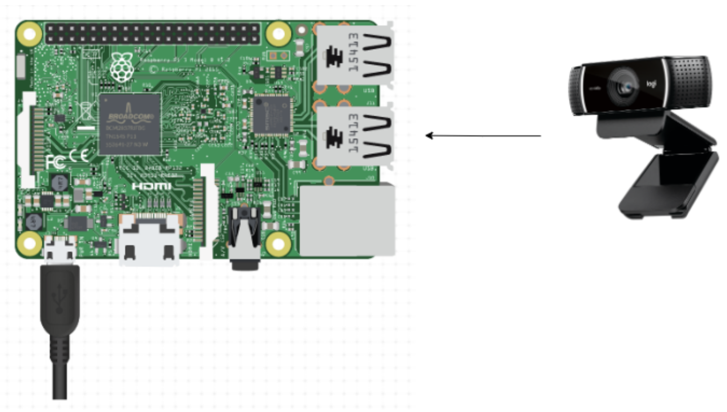


Figure 5: Camera attach to Raspberry Pi in door/gate security node

Task breakdown

Trac Duc Anh Luong	Dao Dung Tri Luu
Set up hardware and sensing system for the Gate Security node	Extend the weather node which originated from Duc Anh individual assignment
Set up an edge server for the Gate Security Node	Set up hardware(arduino + sensor + actuators + edge devices) for Window Security Node
Create a web server to visualize the data collected and interact with the IoT nodes	Establish and program MQTT communication among nodes
Integrate the Discord API to the system for	Add some Discord bot commands to control

pulling data, interacting with the IoT nodes, and getting alerts from events	Window Security Node and alert users when window security problems happen
Connect the Weather node to Thingspeak to fetch data and get visualizations	Design database to store security data collected from Gate and Window Security Node
Connect the Gate Security node to Google Cloud Platform to store images from the security webcam	Contribute to the report and video presentation
Contribute to the report and video presentation	

Implementation

1. Sensing system

a) Weather Node
Our design for the weather node is shown in Figure 2. This weather sensing system is an extension of Duc Anh's previous weather sensing system. Our weather sensing system includes 2 sensors, which are DHT11 (for measuring the temperature and humidity) and MQ-2 (for measuring the various kinds of gas related to fire such as LPG, methane, or carbon monoxide). The actuators for displaying the status of the sensing system are a 16 x 2 LCD module which displays the temperature and humidity value, 3 LEDs and a buzzer. Each led is associated with a unit of measuring e.g the green led is associated with humidity value, the yellow led is associated with temperature value and the red led is associated with gas value; and these led will be turned on if the collected values are higher than certain thresholds. The thresholds can be changed using the Discord app or the website which will be shown in detail in the next sections. The buzzer is associated with the gas value, and when the gas value is higher than the threshold, the buzzer will buzz, as well as the led red will be turned on indicating the house is in fire danger. An Arduino Uno is in charge of controlling the sensor and actuators and this Arduino Uno has serial communication with a Raspberry Pi as an edge device.

b) Gate Node
The Gate Security Node implements both password checking and face recognition for access permission. On the node, the Arduino is attached with a servo, which serves as our gate lock, a keypad for the user to enter their password, a 16x2 LCD I2C display to guide the user during the authentication protocol, 2 LEDs to indicate whether the authentication process was successful or not (red led indicates wrong password while green led indicates correct password), and a buzzer to alert if someone tries to break in as well as providing a countdown before locking the door. The servo will rotate 180 degrees to open the door and automatically rotate back to its original position after around 18 seconds for the user to come in. The camera for face recognition is attached to the Raspberry Pi. When the camera detects a face, it will capture the image, remove the surroundings so that there is only the face left, convert the image color to

grayscale and break the image down to vector matrices. The program then runs the matrices through our trained model to determine if the person in front of the camera is identical to the person registered in our database. If the confidence meets the threshold, the Raspberry Pi will send a serial signal to Arduino. After receiving the message, the Arduino will open the door with a countdown to auto-lock similar to the entering password protocol.

c) Window Node

The Window Security Node also has an Arduino Uno to control the operation of sensors and actuators. We used an SW-520D tilt sensor to define whether the window is closed or open. The operation principle of SW-520D is simple. The sensor includes two balls and the balls close or opens the two pins of the sensor based on the angle of the sensor. This principle can be summed up through the figure below.

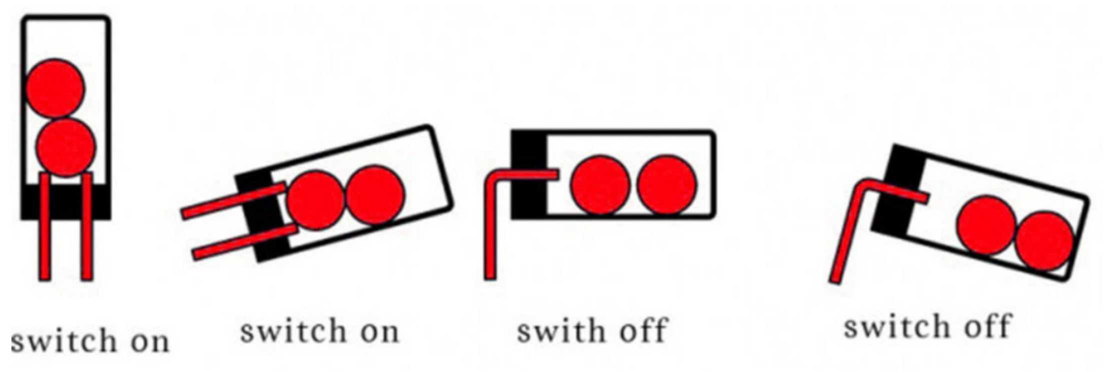


Figure 6: Operation principle of SW-520D sensor

Using the SW-520D sensor, we define the state of our window open or close. The Window Security Node can be disabled or enabled through the Discord app. When the node is enabled, if the window is in an open state, the attached led will and buzzer will flash and buzz to a beat showing that someone is trying to open the window. The data about this event will be transmitted to the Raspberry Pi (edge device) of this node through serial communication, and this Raspberry Pi transmitted this data to the Raspberry Pi of the Weather Node (considered as the Master edge device) by MQTT protocol. The Master Raspberry Pi will store the data related to this event in the database, and also an alert will be sent to the user through Discord. The states defined in the Window Security Node are demonstrated in the figures below.

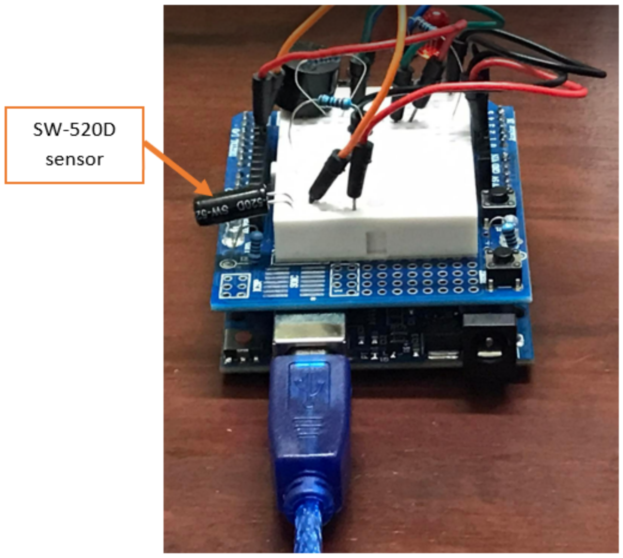


Figure 7: Window close state

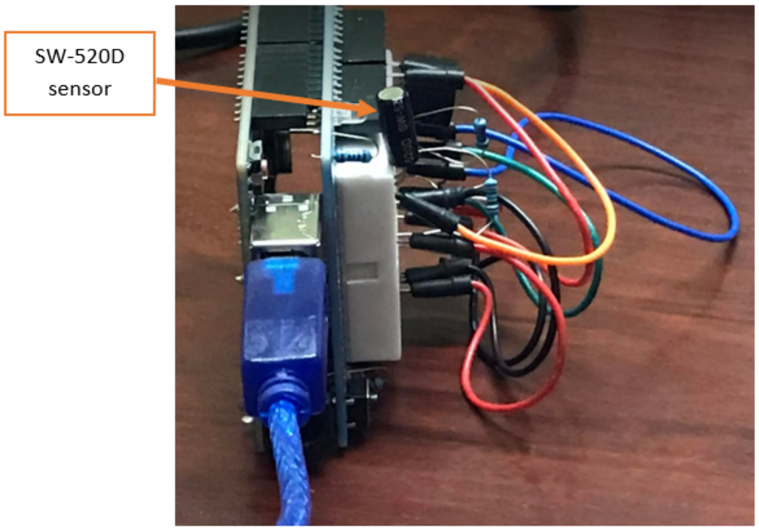


Figure 8: Window open state

2. Edge servers

For each node in the system we used one Raspberry Pi as an edge device. At each node, the edge device communicates with the Arduino using bidirectional serial communication. Edge devices communicate with each other using the MQTT protocol, which will be covered in more detail in the Communication Protocols section.

a) Weather Node

As we mentioned above, the Raspberry Pi of this node serves as a Master of the edge devices. It not only collects data from the Arduino of the Weather Node but also collects and stores into the database data coming from Window Node and Door Node. The main functions of the Weather Node's edge device are:

- Collect and store data from Weather Node's Arduino into the database and control the operation of this Arduino through serial communication.
- Collect and store data from Door Node and Window Node into the database through MQTT communication protocols.
- Host a website that allows users to view content such as weather conditions, the gas level of the house or security logs. The website also lets users control the system by setting the threshold to alert high temperature, high humidity and high gas level.
- Run mosquitto broker for MQTT.
- Run two Python scripts that serve as MQTT subscribers to receive data from other nodes.
- Run a Python script allowing users to interact with the Weather Node by Discord.

b) Door Node
The Raspberry Pi integrates bi-directional serial communication with Arduino on the Gate Security Node and is also a Slave to the Weather Node Master. The security data in terms of authenticated and unauthenticated will be sent to the Master. The main functions of the Gate Security Node's edge device are:

- Use the OpenCV module and haarcascade_frontalface_default.xml in the face recognition package in Python.
- Register the user's credentials (ID, Name) into the database, and get image data from the webcam.
- Train the dataset of images that we collected to build the model which will be used for our face recognition authentication.
- Run the Recognition program continuously and detect if there is any face within the camera frame. If there is one, run the image matrices through our model to determine if that person is in our database or not. If the person is authorized, send a signal to the Arduino to open the gate. Otherwise, send an alert signal that triggers our red led and buzzer.
- Publish the data of entries (authorized and unauthorized) to the Master.
- Run a Python script that allows users to open the door using the Discord API with a secure password.
- Run a Python script that automatically uploads the images that the webcam captures to the Bucket registered for this project on GCP - Google Cloud Platform.
- Send messages to Discord to alert the user if someone entered an invalid password or does not have their face registered in our database and is trying to break in.

c) Window Node
Beside having serial communication with the Arduino, the Window Node's edge device serves as a Slave to the edge device of the Weather Node and it will transfer the data collected from the Arduino to the edge device of the Weather Node. The main functions of the edge device of the Window Node are:

- Collect data from Window Node's Arduino to detect whether the window is opened or closed.
- Publish data of the window state to the Master.

- Run a Python script allowing users to disable/enable Window Node operation and alert users when a window intrusion intent is happening.

3. Communication protocols

Our proposed system uses two communication protocols:

- Serial communication:
  We use this type of communication for transferring data between Arduino and Raspberry Pi in each node of our system. The way we implement this type of communication is the same as how each member did in an individual project, so we will not go into further details with this protocol.
- MQTT protocols:
  MQTT is a lightweight communication protocol that is used for IoT projects. To implement this protocol, we need an MQTT broker and in our project, we used Mosquitto MQTT broker, which is installed and configured on the Weather Node edge device. To make the Mosquitto broker auto start when the Raspberry Pi boots, after installing, we run the command: '**sudo systemctl enable mosquitto.service**'

  For programming the MQTT subscribers and publisher, we use the Python library paho-mqtt. This library allows us to create publishers and subscribers, allows publishers and subscribers to connect to the MQTT broker through the broker's IP address, and also lets programmers define the behavior of publishers and subscribers. It is worth noting that the broker, publishers and subscribers must be connected to the same local area network. In our system, we have two publishers, one is on Door Node's edge device and the other is on Window Node's edge device, and the two corresponding subscribers are on Weather Node's edge device to receive data and insert it into the database on Weather Node's edge device.

4. API and web design

For this project, we have built a web server dedicated to displaying the collected data from our Weather Node and both of our Security Nodes. The site was built with Chart.js, PHP, and Bootstrap to furnish the UI. The website displays the temperature and humidity data in 2 line graphs and displays the gas level as "Safe" or "Dangerous". To interact with the Weather node, the user can set the thresholds for temperature, humidity, and gas level individually. After the threshold is set, if the data fields recorded exceed that threshold, an alert for each field will be triggered, as well as a notification sent to our Discord server.

For the API, this project implemented the Discord API to create a bot that monitors and logs messages to our Discord server, as well as receives commands from the user. Each IOT node hosts a different command that contains queries, inserts statements to the MySQL server to manipulate data, and serial messages to control the function of the Arduino. We also take advantage of the requests module in Python to automatically send command messages and receive notifications indicating the states of the alert, check if any breach has been caught and give the user ease of mind. In addition, we also used the Thingspeak API write key to insert our collected data into a dedicated Weather Channel.

5. Cloud computing

The two cloud computing platforms that our project has implemented are Thingspeak and GCP - Google Cloud Platform. As we have already mentioned previously, Thingspeak is used to analyze and record data in real time for easy access and chart visualizations.

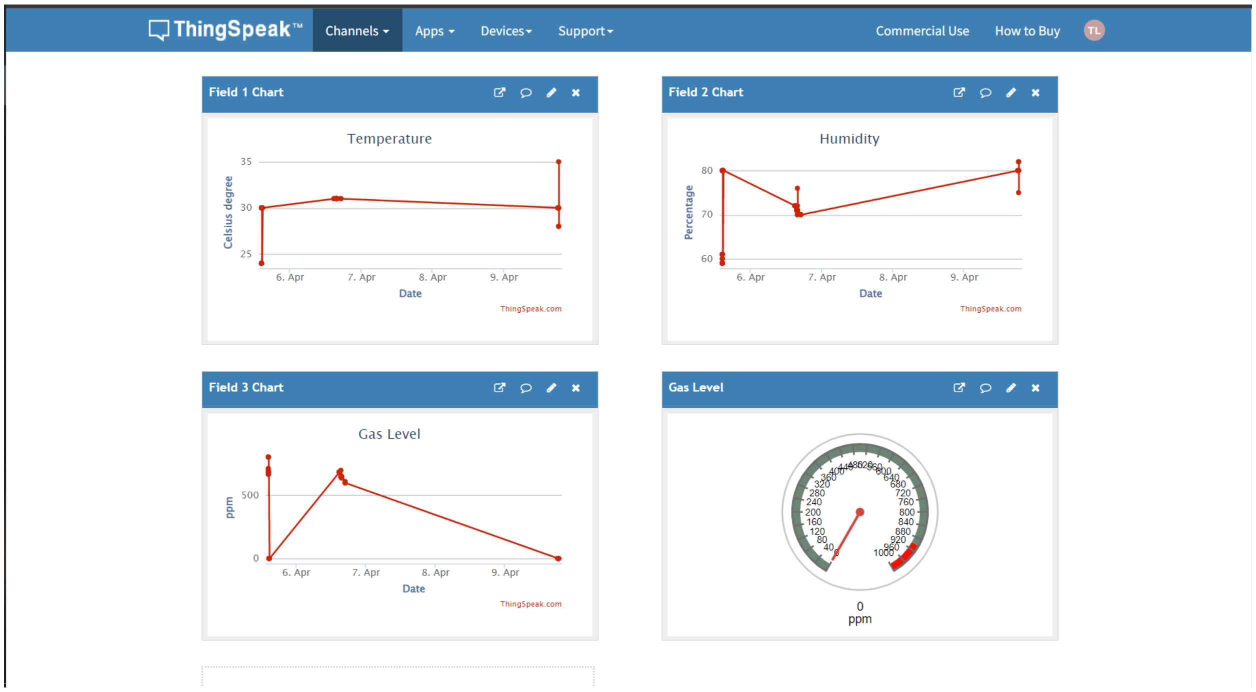
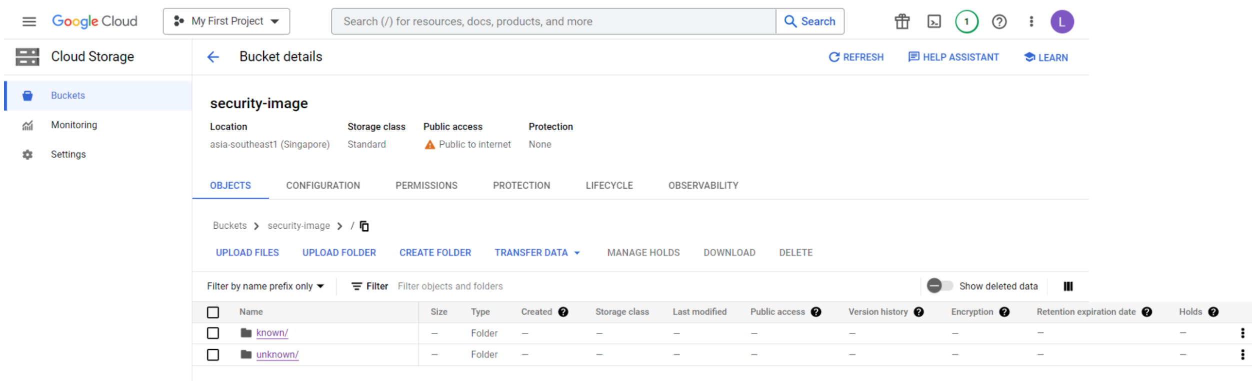


Figure 9: Thingspeak data visualization

For the Google Cloud platform, our main usage is to store image data that is collected by the webcam. Therefore the user could easily be aware of if the security of the house has been breached, and the identity of the breacher through our captured photo. The photos are captured, stored locally in the Gate Security Node, and sent to the Google Cloud Platform using a credential key. After that, the images are stored in 2 separate folders in a Bucket named security-image, with each provided an authenticated link that is later embedded into our web server and can easily be downloaded by our user.



security-image

Location	Storage class	Public access	Protection
asia-southeast1 (Singapore)	Standard	Public to internet	None

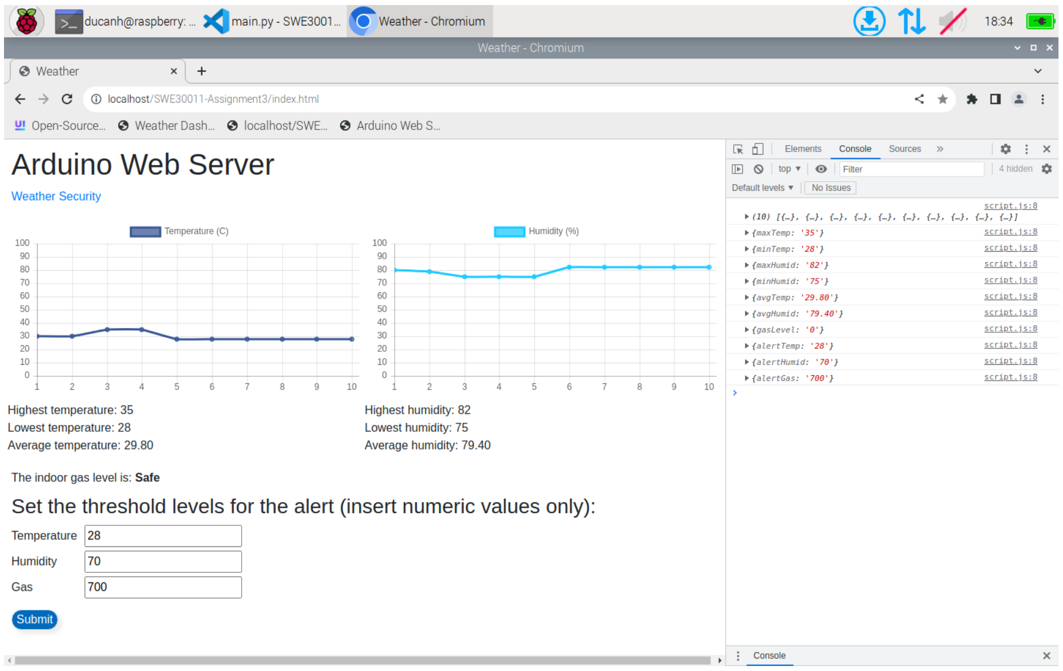
Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date	Holds
known/	–	Folder	–	–	–	–	–	–	–	–
unknown/	–	Folder	–	–	–	–	–	–	–	–

Figure 10: Google Cloud Platform security-image Bucket

User Manual

We provide two ways that users can interact with our system, which are the website and Discord.

- Website:
  The website provides users with a way to view the data collected from IoT nodes and interact with the system. On the website, there are 2 hyperlinks on the navigation bar, one is Weather and the other is Security as shown in the figure below. The Weather page allows users to view some statistics and data visualization of the temperature and humidity such as charts of temperature and humidity, average humidity and temperature, etc. It also shows the state of the gas level in the house. At the end of the Weather page, users can change the threshold value to show alerts on the Weather node of the system. On the Security page, the user will see two security logs, which include the data of the Door Node and Window Node. The data of the Door Node include the number of authorized and unauthorized access attempts and a table of attempts history with detailed time. In addition, users can download the images of people who are trying to open the door (both authorized and unauthorized) by clicking on the corresponding buttons. Whereas, the data of the Window Security Node include the number of window breach attempts and a table of breach attempts history with time and duration.



Arduino Web Server

Weather Security

Highest temperature: 35
Lowest temperature: 28
Average temperature: 29.80

Highest humidity: 82
Lowest humidity: 75
Average humidity: 79.40

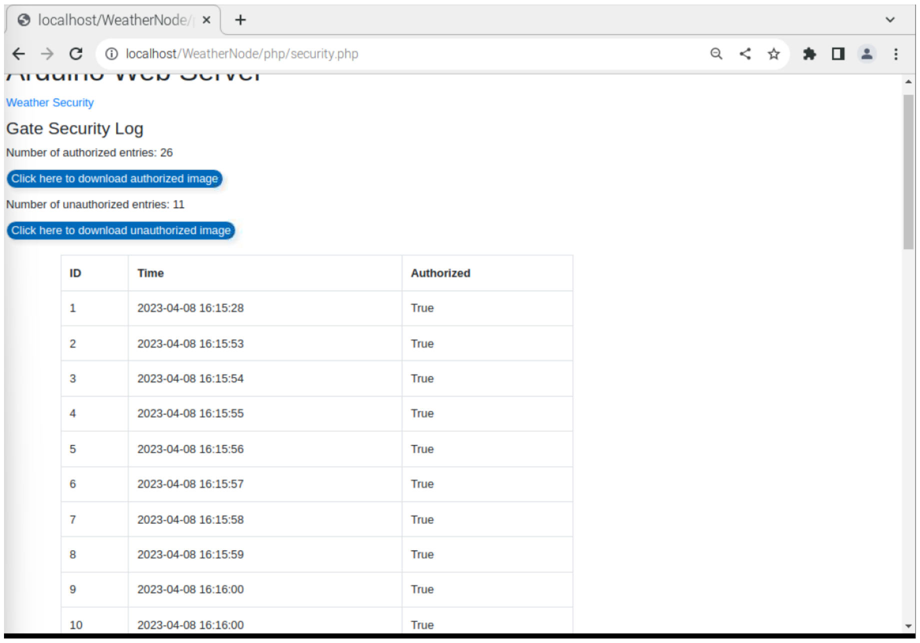
The indoor gas level is: **Safe**

Set the threshold levels for the alert (insert numeric values only):

Temperature	28
Humidity	70
Gas	700

```
                                                     script.js:8
▸ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
▸ {maxTemp: '35'}                                    script.js:8
▸ {minTemp: '28'}                                    script.js:8
▸ {maxHumid: '82'}                                   script.js:8
▸ {minHumid: '75'}                                   script.js:8
▸ {avgTemp: '29.80'}                                 script.js:8
▸ {avgHumid: '79.40'}                                script.js:8
▸ {gasLevel: '0'}                                    script.js:8
▸ {alertTemp: '28'}                                  script.js:8
▸ {alertHumid: '70'}                                 script.js:8
▸ {alertGas: '700'}                                  script.js:8
>
```

Figure 11: Weather page of the website



Gate Security Log

Number of authorized entries: 26

Click here to download authorized image

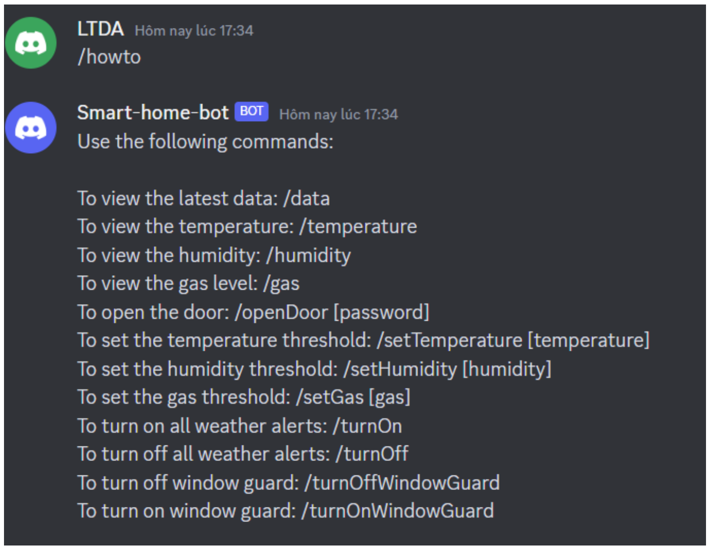
Number of unauthorized entries: 11

Click here to download unauthorized image

ID	Time	Authorized
1	2023-04-08 16:15:28	True
2	2023-04-08 16:15:53	True
3	2023-04-08 16:15:54	True
4	2023-04-08 16:15:55	True
5	2023-04-08 16:15:56	True
6	2023-04-08 16:15:57	True
7	2023-04-08 16:15:58	True
8	2023-04-08 16:15:59	True
9	2023-04-08 16:16:00	True
10	2023-04-08 16:16:00	True

Figure 12: Security page of the website

- On Discord:
  The operation of our system can be monitored and controlled through a Discord server. On this server, there are two channels: #general and #notification. The #general channel is where users write commands to view the data or change some settings of the system and the #notification channel is where users will receive alerts from the system such as when someone is trying to break in using the main door, or someone is trying to break in through the window. By typing the command /howto, users will see a list of available commands to interact with the system. The list of commands and their purpose are summed up in the figure below.



LTDA Hôm nay lúc 17:34
/howto

Smart-home-bot BOT Hôm nay lúc 17:34
Use the following commands:

To view the latest data: /data
To view the temperature: /temperature
To view the humidity: /humidity
To view the gas level: /gas
To open the door: /openDoor [password]
To set the temperature threshold: /setTemperature [temperature]
To set the humidity threshold: /setHumidity [humidity]
To set the gas threshold: /setGas [gas]
To turn on all weather alerts: /turnOn
To turn off all weather alerts: /turnOff
To turn off window guard: /turnOffWindowGuard
To turn on window guard: /turnOnWindowGuard

Figure 13: List of commands syntax and their purposes

Limitations

Overall, our system is functional and bug-free, but some problems should be solved in future development.

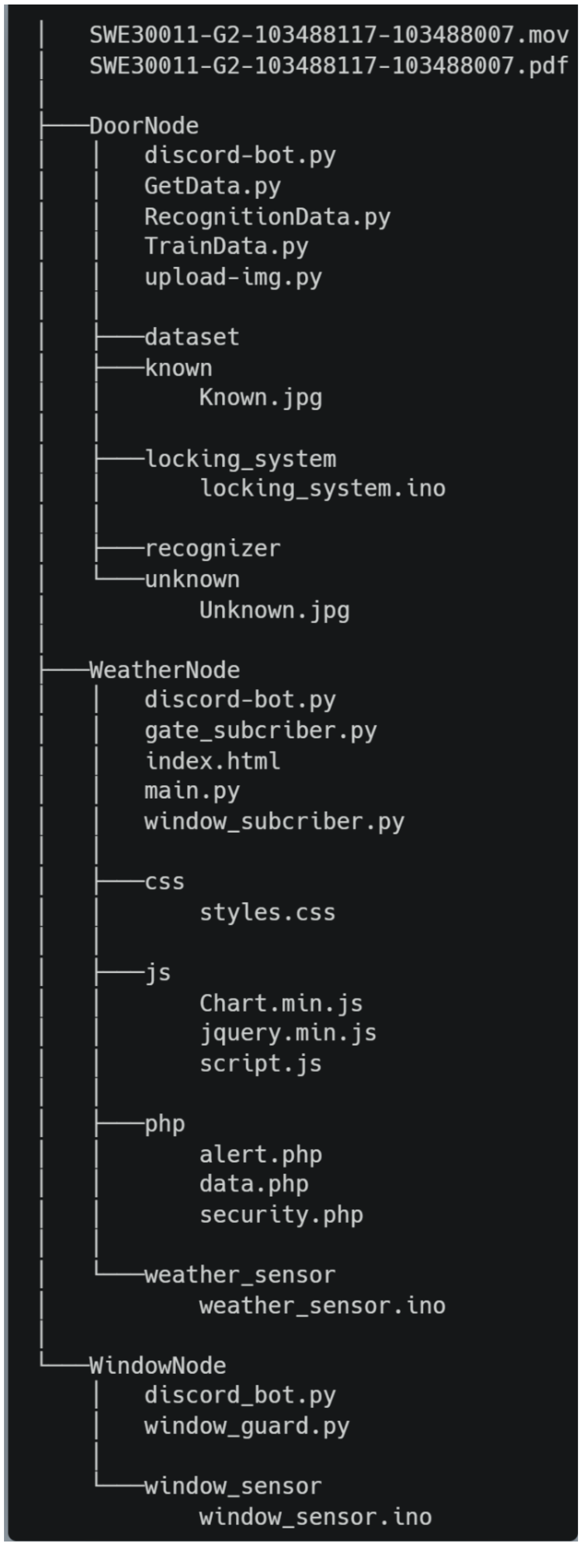
- The website is quite simple and does not have an eye-catching design as it is held together using the default styles from Bootstrap. The function of the web is mostly read, with limited options to let users control the system. Some new features that we can add in the future are door password management, unauthorized door access image gallery, or disabling/enabling Window Node through the web.
- Users may want an app with a user interface rather than typing discord bot to control the system.
- Performance of the system is another problem, as we collect much more data, the performance of the system may reduce. This can be improved with faster SQL insertion and query statements.
- The Cloud Platform still has potential for development, as we can use MQTT and HTTP to send log messages to GCP, which will monitor the state of the Raspberry Pi machines and open the space for upgrades and maintenance. On a side note, ThingSpeak also allows us to communicate our devices using MQTT through channels and topics.

Resources

- Lecture slides, modules and tutorial exercises on Canvas
- Install Mosquitto MQTT Broker on Raspberry Pi (source: https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/)
- SW520D SW-520D Tilt Sensor (source: https://www.sunrom.com/p/sw520d-sw-520d-tilt-sensor)
- Display Data in an HTML Table Using PHP & MySQL (source: https://codingstatus.com/display-data-in-html-table-using-php-mysql/)
- Get started with Bootstrap (source: https://getbootstrap.com/docs/5.3/getting-started/introduction/)
- Working with USB webcams on your Raspberry Pi (source: https://raspberrypi-guide.github.io/electronics/using-usb-webcams)
- Using DHT11 (source: https://projecthub.arduino.cc/arcaegecengiz/12f621d5-055f-41fe-965d-a596fcc594f6)
- Servo Motor basics with Arduino (source: https://docs.arduino.cc/learn/electronics/servo-motors)
- How to set up a keypad on an Arduino (source: https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/)

Appendix

File system:



```
│   SWE30011-G2-103488117-103488007.mov
│   SWE30011-G2-103488117-103488007.pdf
│
├───DoorNode
│   │   discord-bot.py
│   │   GetData.py
│   │   RecognitionData.py
│   │   TrainData.py
│   │   upload-img.py
│   │
│   ├───dataset
│   ├───known
│   │       Known.jpg
│   │
│   ├───locking_system
│   │       locking_system.ino
│   │
│   ├───recognizer
│   └───unknown
│           Unknown.jpg
│
├───WeatherNode
│   │   discord-bot.py
│   │   gate_subcriber.py
│   │   index.html
│   │   main.py
│   │   window_subcriber.py
│   │
│   ├───css
│   │       styles.css
│   │
│   ├───js
│   │       Chart.min.js
│   │       jquery.min.js
│   │       script.js
│   │
│   ├───php
│   │       alert.php
│   │       data.php
│   │       security.php
│   │
│   └───weather_sensor
│           weather_sensor.ino
│
└───WindowNode
    │   discord_bot.py
    │   window_guard.py
    │
    └───window_sensor
            window_sensor.ino
```

Figure 14: Submission file system