VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**MULTIDISCIPLINARY PROJECT - CO3109**

---

**Topic:**

# Bach Khoa Smart Office

---

|  |  |  |
|---|---|---|
| Lecturer: | Phan Trung Hieu |  |
| Student: | Huynh Nhat Quang | 1952212 |
|  | Doan Hong Hieu Kien | 2052555 |
|  | Nguyen Hong Quan | 2052228 |
|  | Pham Thien Dang | 1952653 |
|  | Nguyen Tien Trung | 2052353 |

HO CHI MINH CITY, 12/2023

# Contents

## 1 Introduction

In the era of industrial revolution 4.0, when technology makes everything smarter: smartphones, smartwatches, smart cars, or even smart homes. Human have successfully used the idea of **smart homes** for a very long time to make their life simpler and better. With smart homes, people do not have to worry about wasting energy when they go out and forget to turn off the lights, or they just relax and smart homes will do anything for them, etc.

Nowadays, not only homes can apply smart technology, many different spaces can also use the modern technology to provide ease of access for working individuals working and living within the space and office is one of them.

Like Smart Homes, Smart Office uses modern technology to increase and boost employee productivity, employee productivity, experience, and efficiency by optimizing the office environment while simultaneously keeping the office space environmentally friendly and cost-efficient. This is usually managed in two different ways, firstly by introducing tools that support employees in finding, using, and collaborating in their office space, secondly by providing analytic capabilities and tools for facility managers to optimize space usage with smart systems. In both ways, an IoT (Internet of Things) device will be used as it connects users via Internet to exchange and manipulate information or data.

In this project, our team will design a office with some IoT devices to make an office smarter, provide better workspace for people working in the office.

Figure 1: A visualization of Smart Office

# 2 User requirements

## 2.1 Functional requirement

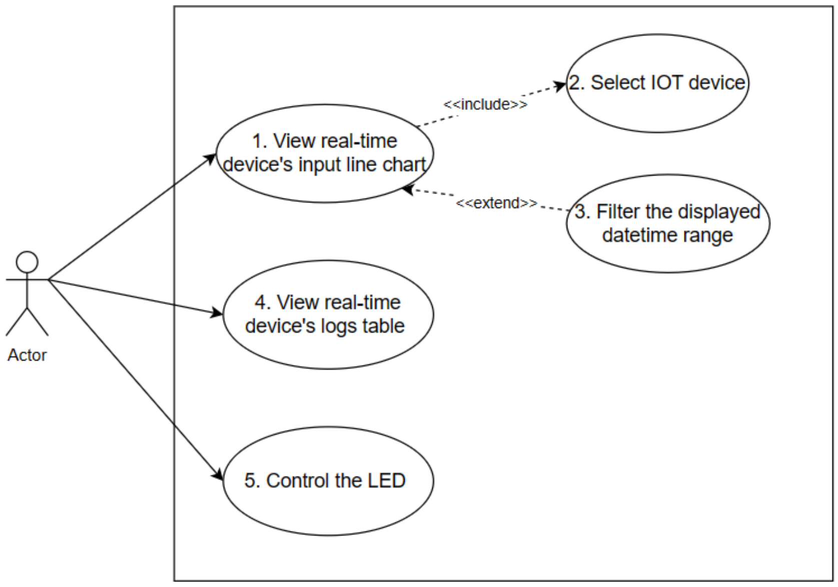
- Users can view temperature, humidity, led status, and light intensity by the line chart and the logs table in real-time manner.
- Users can set the range of date time for viewing a specific time as they wish.
- Users can turn on/off the LED Automation, which is controlled by infrared sensor.

## 2.2 Non functional requirement

- **Performance**: the system must perform well, with fast response times and minimal latency, to ensure a smooth and seamless user experience. Information is updated from the sensor continuously and fully operational 24/24.
- **Scalability**: The system must be scalable to accommodate future expansion or modification of the smart office system, such as addition of new sensors or devices.
- **Reliability**: The system must be reliable and operate without errors, ensuring user safety and security. Multiple device interactions must work properly with low errors, and sensor information can have errors but should not exceed 1 seconds.
- **Usability**: The system must be user friendly and easy to use, with a clear an intuitive interface, to ensure that users can interact with the system easily and efficiently.
- **Compatibility**: The system must be compatible with a wide range of devices and platforms to ensure seamless integration and users accessibility.
- **Maintainability**: The system must be easy to maintain, with clear documentation and support, to ensure that any issues or errors can be addressed quickly and efficiently.

# 3 Use-case

## 3.1 Use-case diagram for the whole system


2. Select IOT device
<<include>>
1. View real-time device's input line chart
<<extend>>
3. Filter the displayed datetime range
4. View real-time device's logs table
5. Control the LED
Actor

## 3.2 Use-case description

### 3.2.1 View real-time device's input line chart

| | |
|---|---|
| ID | 1 |
| Name | View real-time device's input line chart |
| Actor | User |
| Description | The user view the line chart of device's input in real-time manner |
| Pre-condition | 1. The user has been connected to the Internet.<br>2. The database has been connected. |
| Include use-case | Use-case 2: Select IOT device |
| Normal Flow | 1. The system triggers the use-case Select IOT device.<br>2. (Extension-point: Filter the displayed datetime range).<br>3. The system displays the device's input value in the line chart in the real-time manner corresponding to the selected device and filtered datetime range. |

| | |
|---|---|
| Exception Flow | Exception 1: The websocket is broken.<br>Exception 2: The connection to MQTT is broken.<br>Exception 3: The connection of the device to MQTT is broken. |

### 3.2.2 Select IOT device

| | |
|---|---|
| ID | 2 |
| Name | Select IOT device |
| Actor | User |
| Description | The user selects the IOT device for viewing |
| Pre-condition | 1. The user has been connected to the Internet.<br>2. The database has been connected. |
| Normal Flow | 1. The system selects the Humid sensor as default.<br>2. The user chooses a specific device from the dropdown list of available devices (Humid, LED, Temp, Moved, Lumos) as they wish. |

### 3.2.3 Filter the displayed datetime range

| | |
|---|---|
| ID | 3 |
| Name | Filter the displayed datetime range |
| Actor | User |
| Description | The user chooses a specific range of datetime for the line chart |
| Pre-condition | 1. The user has been connected to the Internet.<br>2. The database has been connected. |
| Extend use-case | Use-case 1: View real-time device's input line chart. |
| Normal Flow | 1. The system specifies the start datetime about 1 hour before the current time and the end datetime about 1 hour after the current time of the range as default.<br>2. The user chooses the specific range of datetime as they wish.<br>3. The system displays the values in line chart corresponding to the filtered range. |

### 3.2.4 View real-time device's logs table

| | |
|---|---|
| ID | 4 |
| Name | View real-time device's logs table |

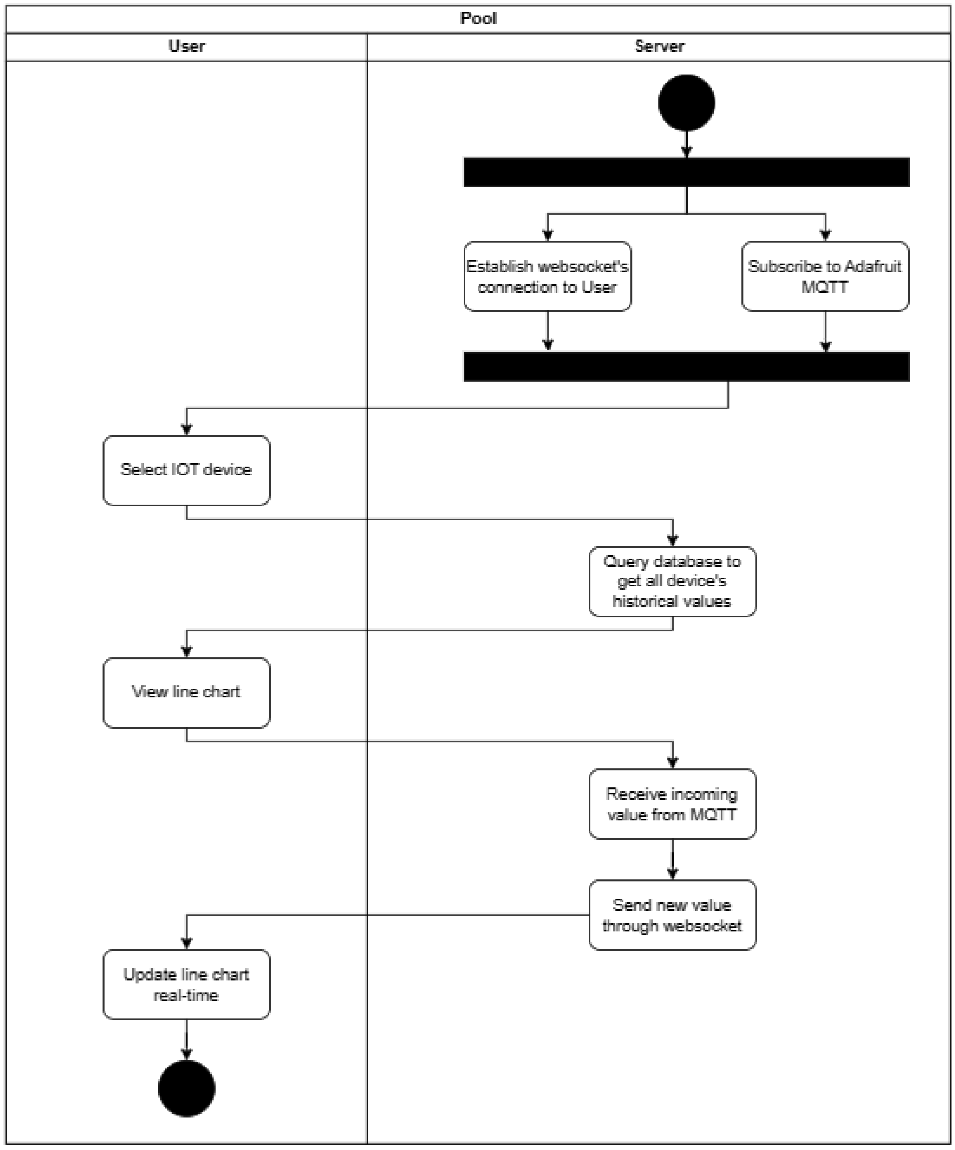
| | |
|---|---|
| Actor | User |
| Description | The user views the device's logs table in the real-time manner |
| Pre-condition | 1. The user has been connected to the Internet.<br>2. The database has been connected. |
| Normal Flow | 1. The system displays the corresponding status of the LED.<br>2. The system updates the status of the LED in the line chart and the logs of the LED in real-time manner. |
| Exception Flow | Exception 1: The websocket is broken.<br>Exception 2: The connection to MQTT is broken.<br>Exception 3: The connection of the device to MQTT is broken. |

#### 3.2.5 Control the LED

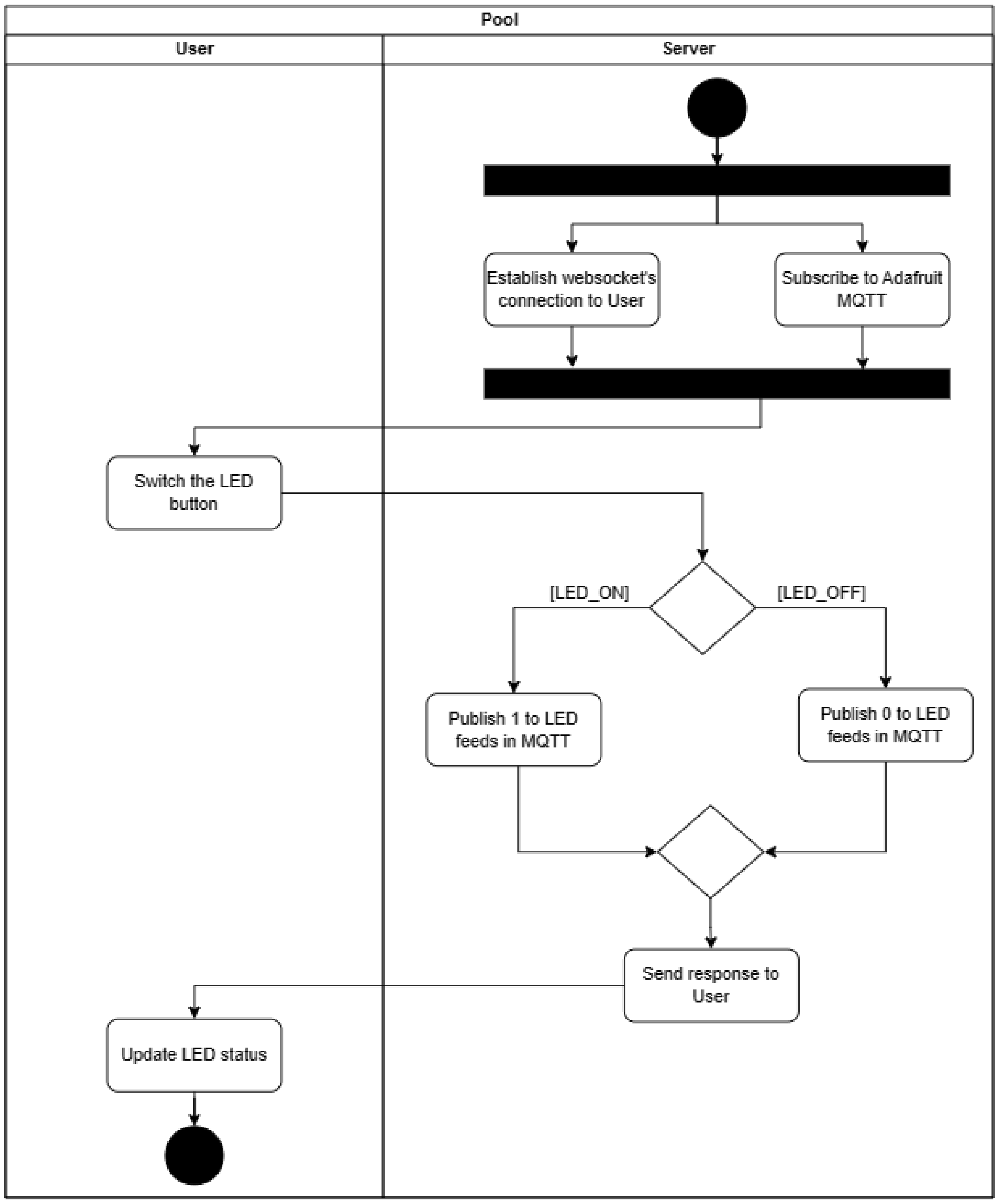
| | |
|---|---|
| ID | 5 |
| Name | Control the LED |
| Actor | User |
| Description | The user turns on/off the LED manually for warning |
| Pre-condition | 1. The user has been connected to the Internet.<br>2. The database has been connected. |
| Trigger | The user clicks on the LED button. |
| Normal Flow | 1. The system displays the logs corresponding to the selected device in real-time. |
| Exception Flow | Exception 1: The websocket is broken.<br>Exception 2: The connection to MQTT is broken.<br>Exception 3: The connection of the device to MQTT is broken. |

# 4 Activity diagram

## 4.1 View real-time device's input line chart


Pool
User
Server
Establish websocket's connection to User
Subscribe to Adafruit MQTT
Select IOT device
Query database to get all device's historical values
View line chart
Receive incoming value from MQTT
Send new value through websocket
Update line chart real-time

### 4.2 For viewing dashboard


Pool
User
Server
Establish websocket's connection to User
Subscribe to Adafruit MQTT
Switch the LED button
[LED_ON]
[LED_OFF]
Publish 1 to LED feeds in MQTT
Publish 0 to LED feeds in MQTT
Send response to User
Update LED status

# 5 System devices

## 5.1 List of devices

| INPUT DEVICES | OUTPUT DEVICES |
|---|---|
| Light sensor | Groove Shield |
| PIR sensor | Module 4 LED RGB |
| DHT20 | LCD 16x2 |

## 5.2 Input devices

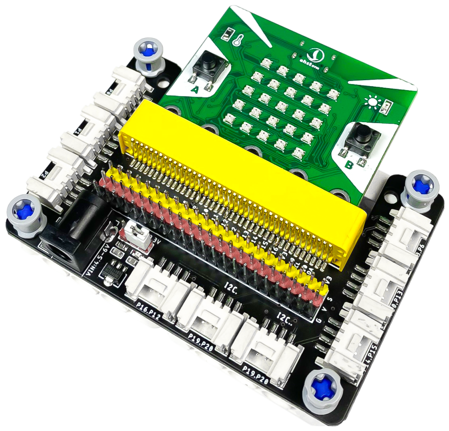
1. **Light sensor**: is a device used to detect and measure the intensity of the light in the surrounding environment. It is suitable for basic applications such as detecting light, knowing whether it is day or night and many other interesting applications.

2. **PIR sensor**: PIR (Passive InfraRed Sensor) is used to detect the movement of objects that emit infrared radiation (such as human or animal movement, heat-emitting objects, etc). This sensor is also known as motion sensor or motion detector. You can adjust the sensitivity of the sensor to limit the range of motion detection at a far or close distance, as well as the intensity of the desired object's radiation. PIR sensor can be used in automation door, light automation, etc.

3. **DHT20**: The DHT20 temperature and humidity uses the I2C output protocol. This sensor has high accuracy and low cost, making it suitable for applications that require temperature and humidity measurement. Applications of this sensor can be used in automated control projects, to record data on temperature and humidity in the surrounding environment, for dehumidifiers, and many other projects.

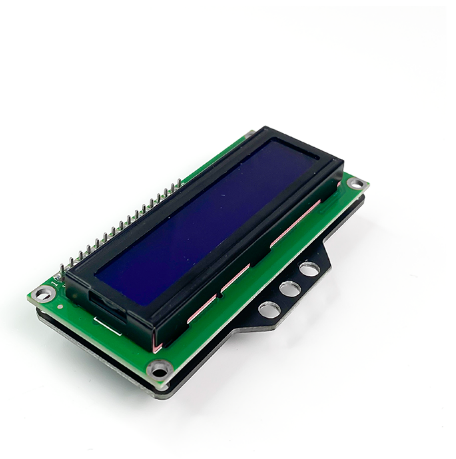
## 5.3 Output devices

1. **Grove shield**: The Grove shield is an expansion shieldcompatible with Yolo: Bit, which allows you to expand an additional 9 Grove standard connection ports, making it easy for you to connect Yolo: Bit with external sensors.

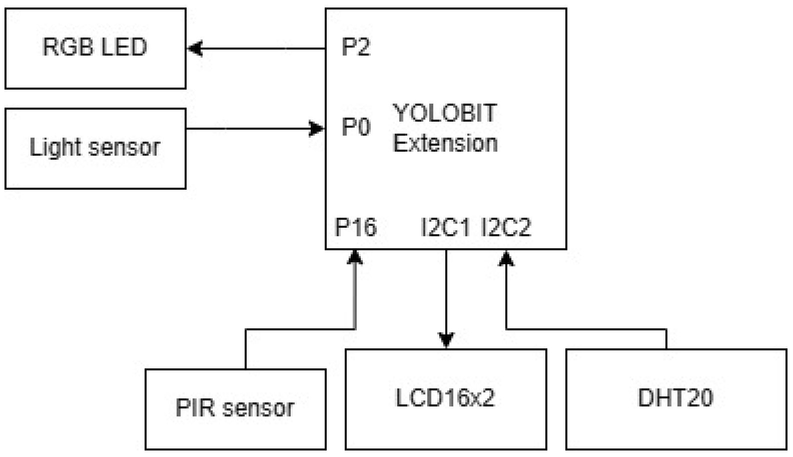


2. **Module 4 LED RGB**: The 4 RGB LED module consists of 4 ws2812 RGB LED lights, each capable of displaying a full range of colors. With an integrated chip, you can control each LED individually or all of them at once.

3. **LCD 16x2**: The LCD 16x2 screen comes with an I2C module that uses the HD44780 driver. This module has the ability to display 2 lines with 16 characters per line. The LCD 16x2 has high durability and is very popular (with many sample codes available). LCD 16x2 is suitable for beginner working on IoT project because of its ease of use. The LCD screen integrated with the I2C communication module makes communication much easier and faster.



## 5.4 Smart Home Connection

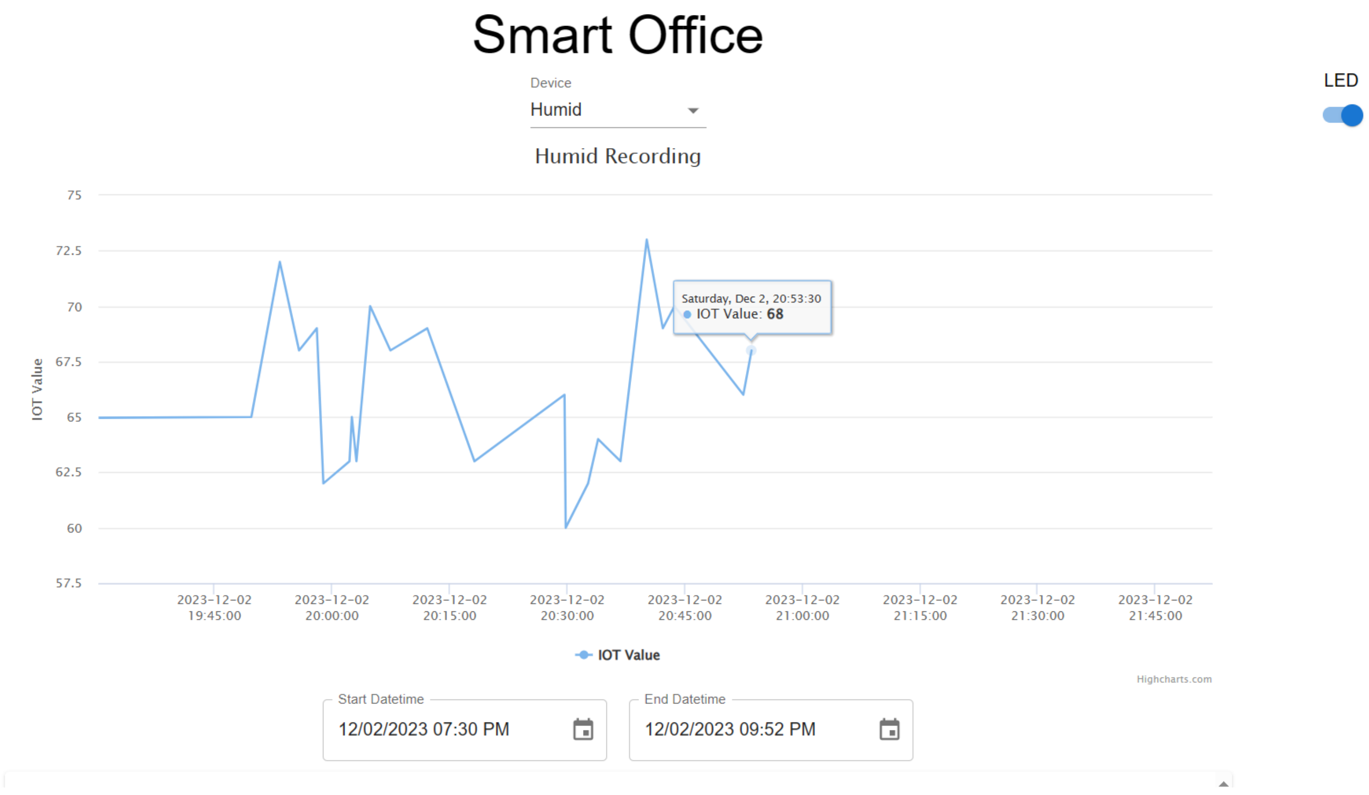


The Yolobit MCU is attached to the expansion Grove shield to connect to input and output devices. Inward arrows represents the inputs, and the outward ones such as LCD 16x2 or LED display the output of Yolobit. The LCD 16x2 and DHT20 are required to connect via I2C, and the rest devices (PIR sensor, light sensor, LED) are connected via normal ports.

## 6 User Interface

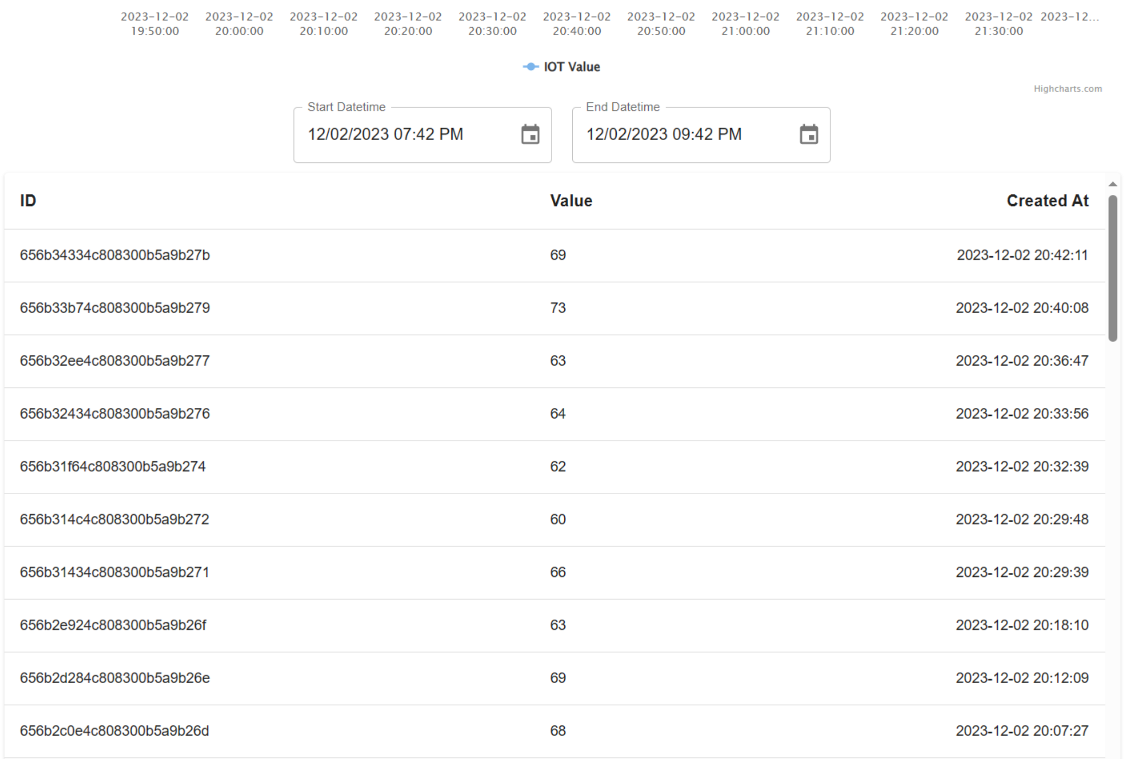
### HOME SCREEN

The home screen of the Bach Khoa Smart Office, a product of the Ho Chi Minh City University of Technology and our team, is simple and user-friendly. After users type in the web page domain name (in this case, is localhost:3000), users will be greeted with the friendly user-interface web applications in order to help users feel easier to control their IoT devices in their office.



In the top of the web page, there is a drop-down list of devices for users to choose from. They can view the measurement of humidity, light intensity and temperature level of the office in the given time (from Start Date time to End Date time) These readings were updated continuously by the sensor and displayed on the home screen. Monitoring these readings can help users adjust their smart devices more accurately.

Users can also choose to turn on or off the LED Automation (in the top right corner) to turn on or off the PIR sensor. In other words, if LED Automation is ON, whenever there is an appearance of human, the light will automatically turn on, and users can choose to turn off this Automation to use light switch as usual.



Start Datetime: 12/02/2023 07:42 PM

End Datetime: 12/02/2023 09:42 PM

| ID | Value | Created At |
|---|---|---|
| 656b34334c808300b5a9b27b | 69 | 2023-12-02 20:42:11 |
| 656b33b74c808300b5a9b279 | 73 | 2023-12-02 20:40:08 |
| 656b32ee4c808300b5a9b277 | 63 | 2023-12-02 20:36:47 |
| 656b32434c808300b5a9b276 | 64 | 2023-12-02 20:33:56 |
| 656b31f64c808300b5a9b274 | 62 | 2023-12-02 20:32:39 |
| 656b314c4c808300b5a9b272 | 60 | 2023-12-02 20:29:48 |
| 656b31434c808300b5a9b271 | 66 | 2023-12-02 20:29:39 |
| 656b2e924c808300b5a9b26f | 63 | 2023-12-02 20:18:10 |
| 656b2d284c808300b5a9b26e | 69 | 2023-12-02 20:12:09 |
| 656b2c0e4c808300b5a9b26d | 68 | 2023-12-02 20:07:27 |

# 7 MQTT protocol

## 7.1 What is MQTT protocol?

MQTT is a standards-based messaging protocol, or set of rules, used for machine-to-machine communication. Smart sensors, wearables, and other IoT devices typically have to transmit and receive data over a resource-constrained network with limited bandwidth. These IoT devices use MQTT for data transmission, as it is easy to implement and can communicate IoT data efficiently. MQTT supports messaging between devices to the cloud and the cloud to the device.

### 7.2 MQTT Components

1. MQTT Client: An MQTT client is any device from a server to a microcontroller that runs an MQTT library. If the client is sending messages, it acts as a publisher, and if it is receiving messages, it acts as a receiver. Basically, any device that communicates using MQTT over a network can be called an MQTT client device.

2. MQTT Broker: The MQTT broker is the backend system which coordinates messages between the different clients. Responsibilities of the broker include receiving and filtering messages, identifying clients subscribed to each message, and sending them the messages. It is also responsible for other tasks such as:

   - Authorizing and authenticating MQTT clients.
   - Passing messages to other systems for further analysis.
   - Handling missed messages and client sessions.

3. MQTT Connection: Clients and brokers begin communicating by using an MQTT connection. Clients initiate the connection by sending a CONNECT message to the MQTT broker. The broker confirms that a connection has been established by responding with a CONNACK message. Both the MQTT client and the broker require a TCP/IP stack to communicate. Clients never connect with each other, only with the broker.

In this project, AdafruitIO will be our MQTT Broker.

## 8 How to install our web application?

Since this project runs locally (localhost), before we jump into instructions, please install the following things:

- MongoDB
- Python
- Node.js

*Note: Make sure to add pip and npm into environment PATH*

We also recommend users to install Visual Studio Code because of its simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

Here are some steps to install our web application

1. Using git to clone these GitHub repositories:

    - https://github.com/QuanKunNguyen2711/Smart-Office-FE
    - https://github.com/QuanKunNguyen2711/Smart-Office-BE

   Or download zip files in each GitHub repository and extract both zip files.

2. Run MongoDB to connect to localhost with port 27017.

3. Open folder "Smart-Office-FE" (Front End part) and "Smart-Office-BE" (Back End part) in Visual Studio Code:

    - In FE part: run `npm install --force` to install node_modules folder, then run `npm start` in Visual Studio Code's terminal
    - In BE part: run `pip install -r requirements.txt` to install all the modules in the file txt, then run `uvicorn app.main:app --reload` to run the backend server.