# IoT HW4

# Setup Instructions

# Subscriber (Laptop)

* **laptop.py**

This Python script functions as the subscriber in an MQTT communication setup. It establishes a connection to the MQTT broker by using the Paho MQTT client library and retrieves configuration details from the 'config.yml' and 'secrets.yml' files. The MQTT client is configured with a client ID and credentials for authentication.

Upon connecting to the broker, the script subscribes to a specific topic ('imu\_sensor') where messages related to the door status are expected to be published. When a new message arrives, the **on\_message** callback function is triggered. This function extracts the status information from the received message, writes it to a 'status.txt' file, and publishes the status change to another topic ('status\_change').

The Flask web application is initiated, providing a route ('/') that renders an 'index.html' template. The template displays the current door status, which is read from the 'status.txt' file. Users can access this information by navigating to the specified hostname and port displayed when the Flask application is run.

To run the script, execute 'main.py'. This will start the Flask web application and establish the MQTT subscriber, allowing users to monitor and interact with the door status through the provided web interface.

* **Template index.html**

The **index.html** file serves as the HTML code for a straightforward web page. This page is designed to showcase the current status of a door, with the status information dynamically updated through data received from the MQTT broker via the **laptop.py** script.

* **refresh\_topic.py**

This Python script utilizes the Selenium library to automate the Chrome browser. It begins by loading configuration details from the 'config.yml' and 'secrets.yml' files, which contain information such as the MQTT broker settings.

The **on\_message** function is defined as an MQTT message callback. When a message is received on the 'status\_change' topic, the payload is checked. If the payload is 'refresh', indicating a request for a page refresh, the script triggers a refresh of the opened webpage.

The MQTT client is initialized, connecting to the specified MQTT broker and subscribing to the 'status\_change' topic. The script then enters a loop to continually check for new messages.

Next, the Selenium webdriver is configured with Chrome options, including the path to the ChromeDriver executable. A webdriver instance is created, and the script opens the Flask-rendered page by navigating to the provided URL.

Finally, an infinite loop is initiated with a one-second delay between iterations. Within each loop, the Selenium webdriver refreshes the webpage, ensuring that any updated content is reflected in the browser.

To execute the script, make sure to replace "/path/to/chromedriver.exe" with the actual path to your ChromeDriver executable. Additionally, provide the correct Flask app URL (e.g., "<http://127.0.0.1:5000/>") to the **driver.get()** function. Running this script will automate the periodic refresh of the specified webpage, facilitating the viewing of real-time updates.

# Library Installations

1. Paho MQTT Client:

*Install using: pip install paho-mqtt*

1. Flask:

*Install using: pip install Flask*

1. Selenium:

*Install using: pip install selenium*

1. PyYAML:

*Install using: pip install pyyaml*