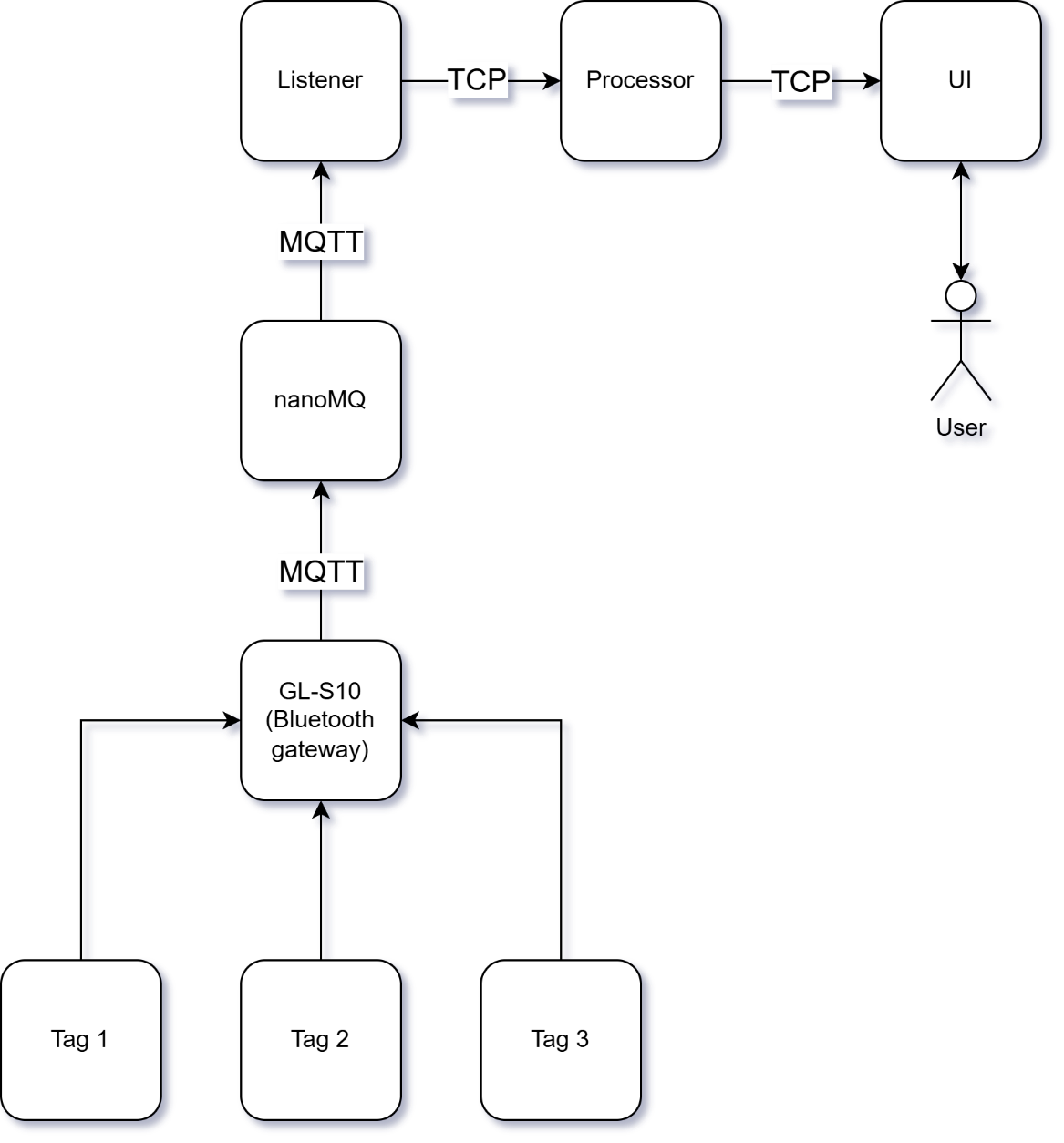
**Healthcare Bluetooth Gateway**

# **I. Description**



The system has a function to detect the location of the person who is carrying the tag. For example, if person 1 is carrying tag 1, the Bluetooth Gateway will receive the RSSI from the tag and send it to the listener via nanoMQ. The listener will then extract the data and send it to the processor to calculate the score and determine the tag’s nearest location. The nearest gateway for the tag will be displayed on the UI, which helps the server keeper track the person’s activity.

* **Tag 1, Tag 2, and Tag 3**: These are Bluetooth tags that communicate with the GL-S10 Bluetooth gateway to send RSSI data, which is used to detect the tag.
* **GL-S10 (Bluetooth Gateway):** This component is a Bluetooth gateway that connects to Bluetooth tags to receive RSSI data from the tags.
* **nanoMQ:** This is an MQTT broker that facilitates the transmission of messages between the Bluetooth gateway and the listener module.
* **Listener:** This module listens for incoming data, possibly from Bluetooth devices.
* **Processor:** The data received by the listener is processed in this component, where it may undergo analysis or transformation before being sent to the UI.
* **UI (User Interface):** The processed data is then displayed on the user interface, allowing the user to interact with the system and view the information.

# **II. Testing Model**

* There are three rooms and one tag used for testing detection. I will carry **Tag 1** and move through all three rooms. When I am in **Room 1**, the UI (Beacon page) will show that Tag 1 is detected by **GW1**.
* As I move to **Room 2**, the location of Tag 1 will be updated and shown as detected by **GW2**, and similarly, when I move to **Room 3**, it will be detected by **GW3**.
* If I stand between **Room 1** and **Room 2**, the tag will be detected by the gateway that is closer.
* This model has also been tested with multiple tags being detected simultaneously. However, for simplicity, I am only explaining how the system works with a single tag.

A diagram of a room

AI-generated content may be incorrect.

# **III. How to setup**

## **1. Install and Setup nanoMQ as MQTT Bridge**

* **Step 1: Download the NanoMQ repository**

|  |
| --- |
| curl -s https://assets.emqx.com/scripts/install-nanomq-deb.sh | sudo bash |

* **Step 2: Install NanoMQ**

|  |
| --- |
| sudo apt-get install nanomq |

* **Step 3:** **Start nanoMQ**

|  |
| --- |
| nanomq start --log\_level debug |

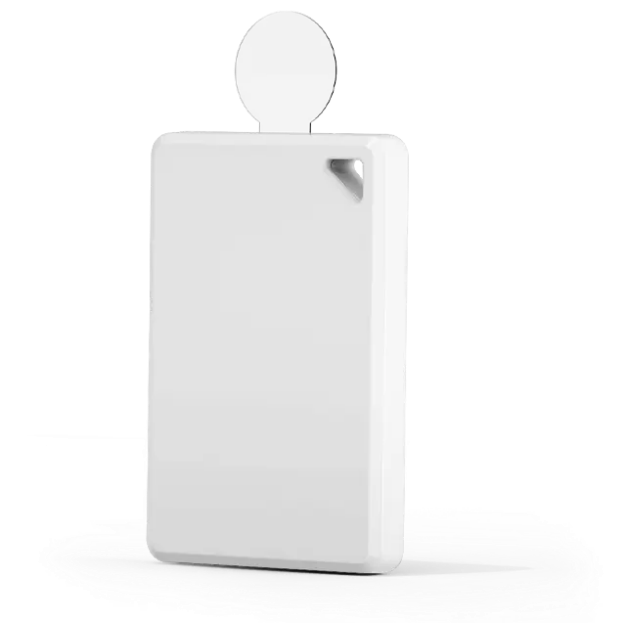
If you start nanoMQ and see the log below, it means that nanoMQ has detected and received data from the GL-S10 (It will show log as below when you setup all of gl-s10 and tag).

A computer screen shot of text

AI-generated content may be incorrect.

### **2. Configuration bluetooth tag**

* We will config the E8 Asset Tracking Tag using the BeaconSET Plus app.

 A screenshot of a phone

AI-generated content may be incorrect.

* **Open the App**
  + Launch the **BeaconSET Plus** app on your phone.
  + Make sure **Bluetooth is turned on**.
* **Find the Tag**
  + The app will automatically show nearby tags.
  + Look for the tag you want to configure and tap on it.

A screenshot of a phone

AI-generated content may be incorrect.

* **Enter Password**
  + When prompted, enter the **default password**: minew123.

A screenshot of a phone

AI-generated content may be incorrect.

* **Configure iBeacon Settings**
  + Go to the slot where **Frame Type** is set to **iBeacon.**
  + In the **Adv Content** section, set Major: 25442

A screenshot of a phone

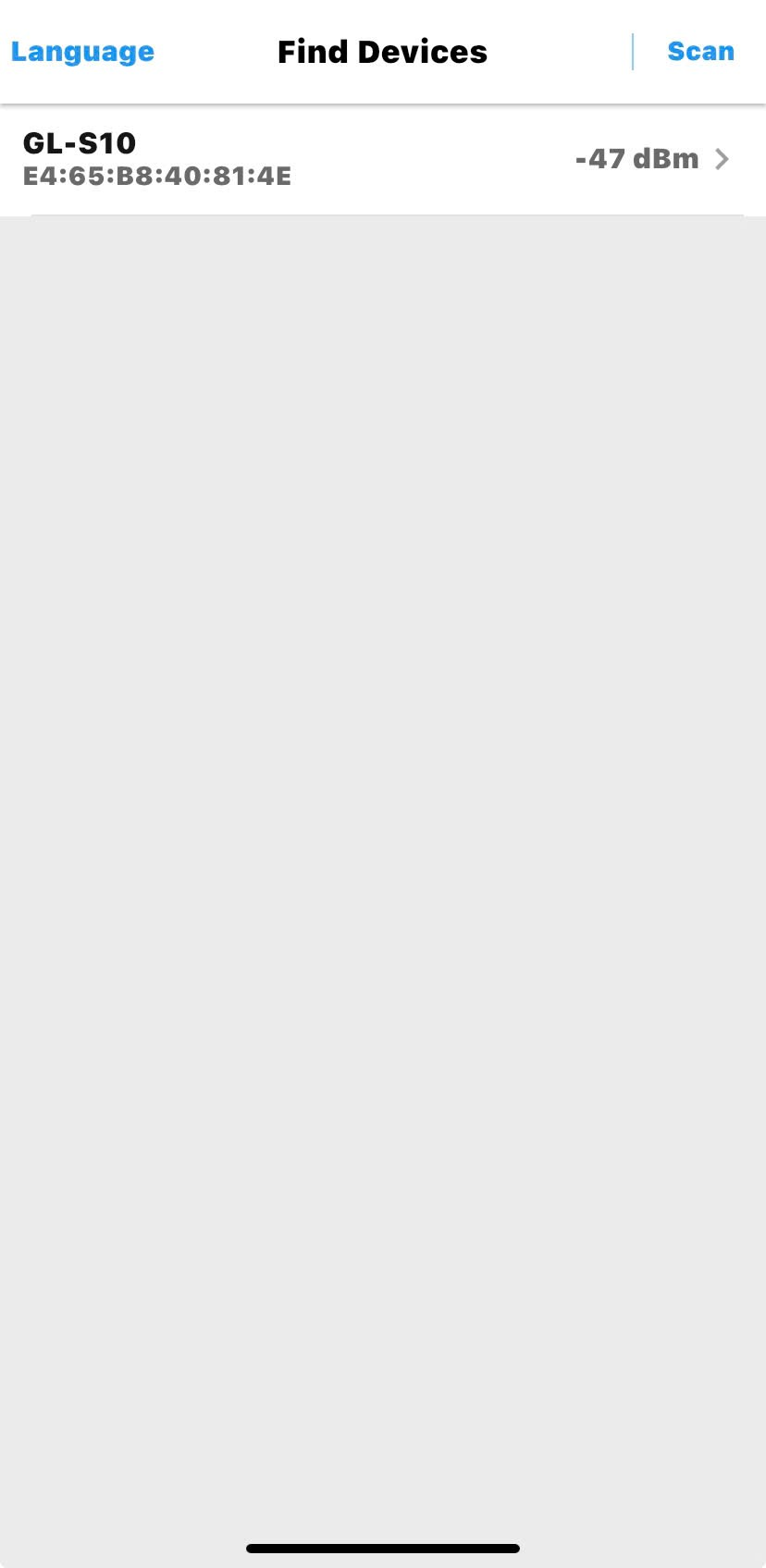
AI-generated content may be incorrect.

* **Save Settings**: Tap the **Config** button to save and apply the new settings.

### **3. Configuration bluetooth gateway (GL-S10)**

* To connect with GL-S10, we need to download an app named “GL-S10” from the manufacturer. Before using the app for configuration, set the GL-S10 into its pairing mode by pressing the reset button of the GL-S10 when the power is off, then turn on the power. Release the button when you see the BLE indicator in the middle flash, and your GL-S10 will enter its pairing mode. After all, press “scan” to search for the device.

A white background with black and white clouds

AI-generated content may be incorrect. 

* After successfully connected with the device, to connect and config Wi-Fi, choose “WiFi” → “Scan WiFi”, and select your WiFi SSID. Click "Done" on the upper right corner to confirm the setting as shown below.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

* We will setting MQTT server because GL-S10 will communicate with nanoMQ via MQTT. Firstly, check the ip of your local server with the command.

|  |
| --- |
| ipconfig |

A screenshot of a computer

AI-generated content may be incorrect.

* Now open application GL-S10 on your phone and choose “MQTT-Server”.

A screenshot of a computer

AI-generated content may be incorrect.

* To connect to the local server via nanoMQ, enter the host of the local server (for my server: 192.168.10.7) and the default port of nanoMQ (1883). Then, click “Done” to apply configuration.

A screenshot of a computer

AI-generated content may be incorrect.

* With the Bluetooth gateway (GL-S10), it will receive all devices that explore Bluetooth nearby. To receive only the tag whose RSSI we want to know, we can set a filter to only receive data from tags with **Major = 25442**, which is set in the **Manufacturer Specific Data**. This ensures that the Bluetooth gateway will only receive data from the tag we are interested in. Go to “**Report Configuration**”.
* Since the tag uses the decimal value **25442**, and the GL-S10 requires hexadecimal, we will enter **6362** (the hex equivalent of 25442) in the filter.

A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a white screen

AI-generated content may be incorrect.

* To connect to the server, the GL-S10 must change its topic to bluetooth/**GWID**/data, where **GWID** is any identifier you choose. For example, if you define the device as **GW2**, then you would use bluetooth/**GW2**/data.

A screenshot of a phone

AI-generated content may be incorrect. A white background with black dots

AI-generated content may be incorrect.

* After all, press “Reboot” in the main screen to GL-S10 turn off the mode configuration.

Note: Ensure your bluetooth gateway in version 3.0.3. If it in 3.0.4. Please press “Upgrade OTA" and choose version 3.0.3. Then, press “Upgrade OTA” and wait until it finished.

### **4. Setup and run server**

* **Step 1: Check if Python 3.10 is installed**

Verify whether Python is installed on your system by running:

|  |
| --- |
| python3 --version |

If Python 3.10 is not installed, use the following commands to install it:

|  |
| --- |
| sudo apt update && sudo apt install python3.10 -y |

* **Step 2: Create a Virtual Environment**

Set up a virtual environment inside the project folder

|  |
| --- |
| python3 -m venv .venv |

* **Step 3: Activate the Virtual Environment**

Activate the virtual environment using the command below

|  |
| --- |
| source .venv/bin/activate |

* **Step 4: Clone the Project Repository**

Since this repository is private, you need to set up an SSH key to authenticate and clone it. Follow these steps carefully.

* + **Open a New Terminal and Generate an SSH Key.**

|  |
| --- |
| ssh-keygen -t rsa -b 4096 -C "your\_email@example.com" |

Replace "your\_email@example.com" with your GitHub email.

When prompted:

* + - Press Enter to accept the default location (~/.ssh/id\_rsa).
    - Optionally, set a passphrase for added security or press Enter to skip
  + **Copy Your Public Key.**

|  |
| --- |
| cat ~/.ssh/id\_rsa.pub |

Copy the output (it starts with ssh-rsa).

* + **Add It to GitHub.**
    - Click the **"add a new public key"** link from your screenshot.
    - In the **Title** field, enter something like "My Laptop SSH Key".
    - In the **Key** field, paste the copied SSH key.
    - Click **Add SSH Key.**
  + Clone the repository.

|  |
| --- |
| git clone git@github.com:CloudBurst-Australia/ble-gateway-server.git |

* **Step 5: Install Required Libraries**

Install the necessary dependencies listed in the **requirements.txt** file:

|  |
| --- |
| pip install --no-cache-dir -r requirements.txt |

* **Step 6: Install redis server**

|  |
| --- |
| sudo apt install redis-server |

* **Step 7: Modify port of redis server**

Since both the **vitals server** and the **BLE-gateway server** run on the same machine, we need to configure Redis to run on two different ports to avoid data conflicts between the servers.

* **Copy the default Redis config file** (which uses port 6379 by default).

The **BLE-gateway** server will use Redis on port 6380.

|  |
| --- |
| sudo cp /etc/redis/redis.conf /etc/redis/redis6380.conf |

* **Edit the copied config file to change the port**.

|  |
| --- |
| sudo nano /etc/redis/redis6380.conf |

* **In the file, find and modify the port setting**.

|  |
| --- |
| port 6380 |

* **Start both Redis instances on different ports**.

Each time you start the servers for the first time, run the following commands.

|  |
| --- |
| sudo redis-server /etc/redis/redis6380.conf # Port 6380 for BLE-gateway server |

* **Step 8: Run the application**

Note: Make sure you already run nanomq before running server and redis server.

We will need to run 3 files: listener.py, processor.py and ui.py. I recommend run these files in unique terminal to easy follow the data.

|  |
| --- |
| python3 listener.py  python3 processor.py  python3 ui.py |

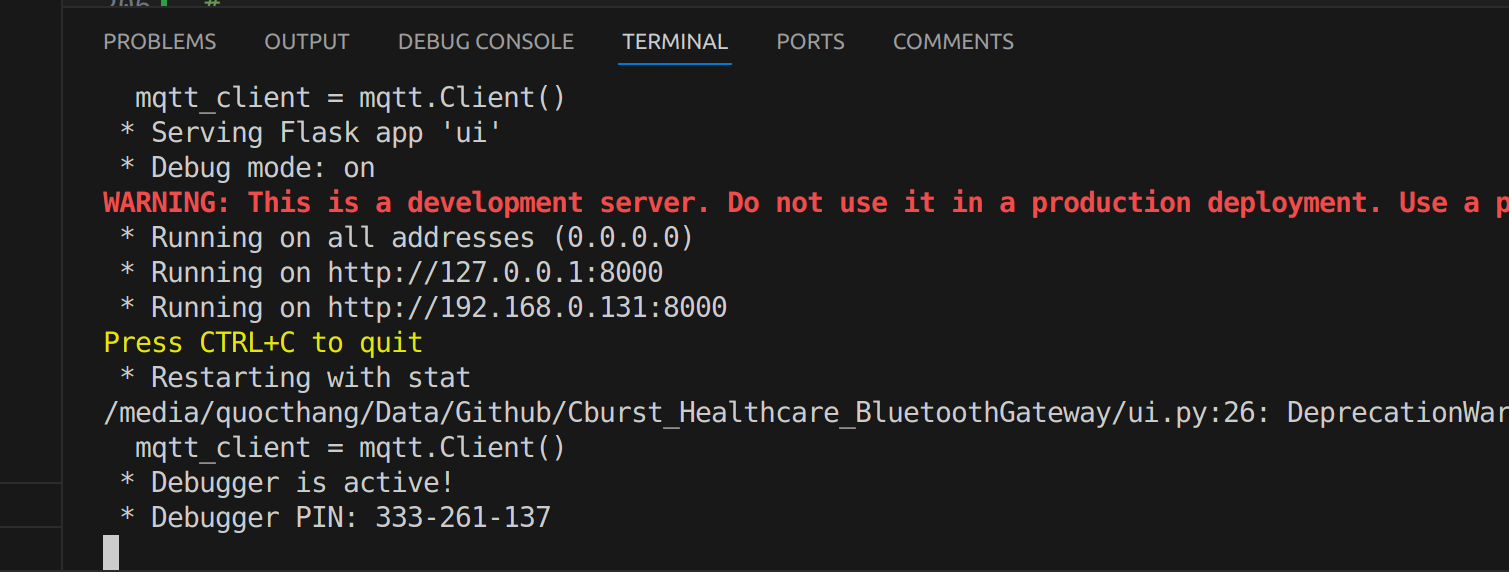
The **listener.py** (left terminal) will receive and push data to the processor, which will calculate scores, as shown in the right terminal.

A black screen with white text

AI-generated content may be incorrect. A black screen with white text

AI-generated content may be incorrect.

When you run **ui.py**, it will display the IP address of your server. When you click on it, it will show an image.



# **IV. User interface functions**

* At first, you will see the login page. Please log in with the account: **admin** and the password: **123.**

A screenshot of a computer

AI-generated content may be incorrect.

In the UI, we will have 7 pages:

* **Dashboard**: Used to determine which gateways are connected to the server and which are online. It also shows the number of beacons detected.
* **Gateways**: A list of gateways connected to the server. You can check the status of gateways and see how many beacons each gateway has detected. The function to control gateways is not developed.
* **Beacons**: This page shows logs of all detected beacons. You can also check the current location of beacons based on the Gateway ID. For example, beacon ID **C300003561E2** has been detected by **GW2**. **"Last seen"** means the gateway still detects the beacon at that time. The **"Beacon Detection Logs"** table shows the first time a beacon was detected.
* **Config**: Not developed
* **Logs**: Not developed
* **Change Password**: Not developed
* **Logout**: Used to log out and return to the login page.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.