## Day 1: Test Connectivity and Functionality of Device, Test of MQTT protocol

**Duration: 5 hours**

After encountering issues with earlier hardware (ESP32, SIM800 module, GPS NEO-6 module) that prevented connectivity, I decided to switch to the **LilyGO7000G**. The initial issues likely stemmed from **2G/GPRS** limitations. My SIM card was unable to connect to the network, send, or receive data.

**Objective:**
Today I focused on testing the general functionality of the device, confirming its ability to connect to the network, use **LTE**, test **APN**, and verify MQTT protocol functionality.

**Hardware:**
- LilyGO7000G
- LTE antenna
- USB-C cable
- SIM card from Comviq

**Software:**
- VS
- Platformio (framework Arduino)
- MQTT explorer
- Broker:
    - broker.hivemq.com
    - test.mosquitto.org

**1. Test Connectivity**

First, I verified that the device was able to establish a network connection:
- SIM card connectivity
- APN settings
- LTE connection

The **"Network_Test"** code provided the following initial insights:
- The modem is powered on and successfully communicating with the network.
- PDP Context was activated, indicating that the modem established a data session with the network.
- IP address was successfully assigned.

**2. Test Sending Data to/from Device Using MQTT Protocol**

Initially, I attempted to use the HTTP response method, but the device failed to connect to the webhook server. As a result, I switched to testing the MQTT protocol, which worked seamlessly.

**a) Publishing from MQTT Explorer to Device**

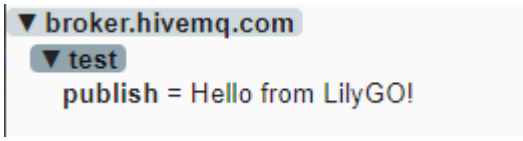
In the **"MQTT_publish"** code, I published messages from **MQTT Explorer** to the device. The device successfully received them, and I saw the following messages in the **Serial Monitor**:

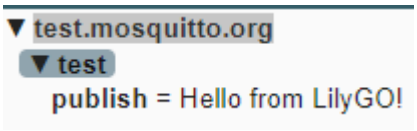
```
Initializing modem...
Waiting for network...
Network connected.
Connecting to the internet with APN: internet.tele2.se
```

```
Internet connected.
Connecting to MQTT broker: broker.hivemq.com
Connected to MQTT broker.
Subscribed to topic: test/subscribe
Message arrived [test/subscribe]: 1
Message arrived [test/subscribe]: "Hello beauty"
Message arrived [test/subscribe]: "I am happy it works :)"
Message arrived [test/subscribe]: "II am happy happy "
Disconnected from MQTT broker. Reconnecting...
Connecting to MQTT broker: broker.hivemq.com
Connected to MQTT broker.
Subscribed to topic: test/subscribe
Reconnected to MQTT broker.
```

### b) Subscribing to a Topic on MQTT Explorer from the Device

In the **"MQTT_subscribe_MQTTexplorer"** code, I tested if the device could send data to MQTT Explorer. I created a topic test/subscribe in MQTT Explorer, where I successfully saw messages from the device:




**Challenges:**

- The **initial signal** was weak, likely due to the setup process. The device needed extra time to establish a connection.
- The **HTTP protocol** failed to work, but **MQTT** proved to be reliable, leading me to conclude that MQTT will be used for data transfer in future steps.
- The device could connect to **broker.hivemq.com** and **test.mosquitto.org**.
- The **Mosquitto broker** on my local machine needs proper configuration to allow the device to connect to it.

**Next Steps:**

- Test **GNSS** functionality (GPS, GLONASS, BeiDou).
- Insert a **LiPo 18650 battery** into the device (the LilyGO has a battery holder for this).
- Attempt to get the device's position using **GNSS** (GPS, GLONASS, and BeiDou). The device does not support **Galileo**, which is the European GNSS system.
    - If GNSS data cannot be retrieved, mock data will be used for further testing.
- Create an **Adafruit IO** account to visualize **battery status** and **location**.
- If time permits, create a simple app for visualization (though this is secondary to the main project goal).

**Final Thoughts:**

The **LilyGO7000G** device is working well with the **MQTT protocol**, and I am able to connect to remote brokers and send/receive messages. The next step involves implementing **GNSS data retrieval**, and possibly visualizing the device's location and battery status on **Adafruit IO**.

**Suggestions for Improvement:**

- **Technical Details**: When describing challenges and limitations (e.g., issues with HTTP), it may be useful to describe the **error messages** or **specific reasons** why something failed, so it's easier to troubleshoot in the future.
- **Code Sections**: Referencing specific code snippets by name is great. For future work, you might include a brief **code explanation** or highlight which parts of the code were crucial to success, especially when connecting to MQTT brokers.

## Day 2: Test Device with Adafruit account

**Duration: 8 hours**

### a) Test device to send simple MQTT message to Adafruit Feed (via USB-C connection)

| Created at | Value | Location | |
|---|---|---|---|
| 2024/12/31 05:11:54PM | Hello from LilyGO! | | × |
| 2024/12/31 05:10:56PM | Hello from LilyGO! | | × |
| 2024/12/31 05:09:47PM | Hello from LilyGO! | | × |

This works as expected.

### b) Test battery level to make sure device can operate without connection to PC

I encountered a problem with the battery holder for the 18650 LiPo battery; it seems to be a factory fault. To work around this, I used an external LiPo battery connected directly to the device.
When using the battery holder, the device consistently showed a battery level of 0.0, even though I tested three different batteries, all of which were fully charged according to the multimeter. Furthermore, the device would not operate without being connected to the PC.

| | | |
|---|---|---|
| 2024/12/31 07:46:42PM | 0.00 | × |
| 2024/12/31 06:30:03PM | 0.00 | × |
| 2024/12/31 06:05:51PM | 0.00 | × |

However, when I switched to a separate LiPo battery, the device operated without the need for a PC connection, but the battery readings remained faulty.

| | | |
|---|---|---|
| 2024/12/31 07:49:13PM | 0.00 | × |
| 2024/12/31 07:48:43PM | 0.00 | × |
| 2024/12/31 07:48:13PM | 0.00 | × |

***Since this is not a critical part of the project, I will continue troubleshooting if time allows.***

### c) Test device to send a simple MQTT message to Adafruit Feed without the USB cable

| | | |
|---|---|---|
| 2024/12/31 07:42:45PM | Hello from LilyGO remote! | × |
| 2024/12/31 07:42:15PM | Hello from LilyGO remote! | × |
| 2024/12/31 07:41:44PM | Hello from LilyGO remote! | × |
| 2024/12/31 07:41:14PM | Hello from LilyGO remote! | × |

Code works as expected.

### d) Test GPS functionality

The GPS functionality is crucial for this project, as it will allow us to track the device's position.

Due to cold weather, I used the **"DollaTek Vattentät GPS aktiv antenn 28dB 3-meter SMA-port för NEO-6M U-BLOX bil"** GPS antenna, which has a long cable, allowing me to keep the LilyGo device inside while placing the GPS module outside.

After some initial testing, I successfully obtained the location and published it to the feed.

| | |
|---|---|
| 2024/12/31 09:00:05PM | 59.579670,17.511856 |
| 2024/12/31 08:59:35PM | 59.579666,17.511858 |
| 2024/12/31 08:59:04PM | 59.579662,17.511860 |
| 2024/12/31 08:58:34PM | 59.579659,17.511864 |

Date / Time / Latitude / Longitude

**Challenges:**

- The device's battery holder was faulty, preventing the use of the 18650 LiPo battery. I used a separate "LiPo 3.7V 1500mAh" battery connected to the 3V and GND pins instead.
- The included GPS antenna was insufficient due to the cold weather and short cable. To overcome this, I used an outdoor-rated GPS antenna with a longer, waterproof cable.

**Next Steps:**

- Try to apply the data on a map.

**Final Thoughts:**

- Adafruit IO proved to be an excellent platform for MQTT data transfer during this phase of testing.
- While dealing with a faulty device wasn't part of my initial plan, I was able to find a solution and continue the project.

## Day 3: Get data on the map

**Duration: 4 hours**

**a) Visualise Data with Serial Monitor**

I needed to adjust code, so the data are stored under **"Location"** and not under **"Value"** to get position on map. To do this I needed to add speed and altitude into parameters, which are required.

Feed looks like this:

| | | | |
|---|---|---|---|
| 2025/01/01 03:19:57PM | 0.000000 | 59.579784, 17.511822, 0 | × |
| 2025/01/01 03:14:01PM | 0.000000 | 59.579758, 17.511988, 0 | × |
| 2025/01/01 03:13:31PM | 0.000000 | 59.579754, 17.511988, 0 | × |
| 2025/01/01 03:13:05PM | 0.000000 | 59.579796, 17.511963, 0 | × |

This format gives me position on map on Dashboard:

Position is not that much accurate, my GPS module is on the windowsill, but I got approximate position, which is good enough.

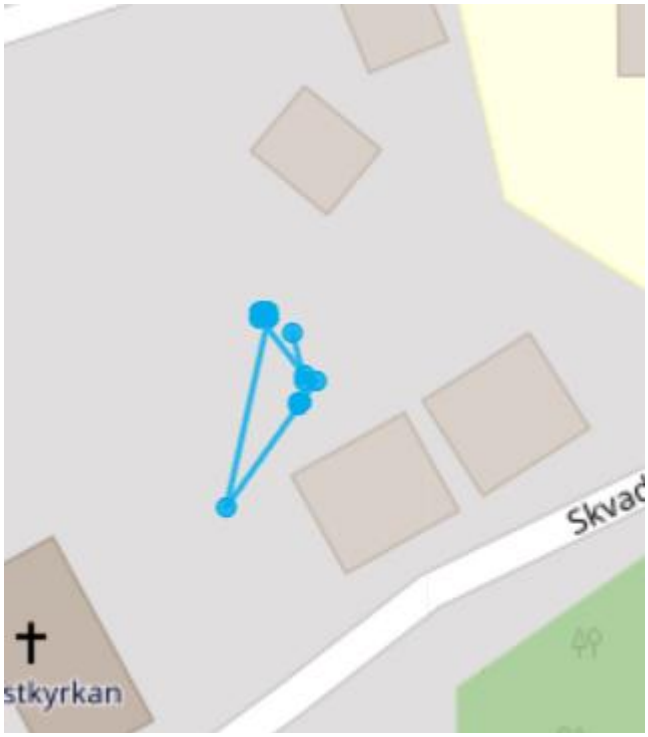
This code is tested with serial monitor to ensure proper storage of data and visualisation. Code works as expected.

**b) Visualise Data without Serial Monitor**

I needed to remove functions of Serial Monitor, so I could test it only with battery.
I got same results even without connection to PC and Serial Monitor:

| | | | |
|---|---|---|---|
| 2025/01/01 03:51:28PM | 0.000000 | 59.579735, 17.51198, 0 | × |
| 2025/01/01 03:50:58PM | 0.000000 | 59.579731, 17.511976, 0 | × |
| 2025/01/01 03:50:28PM | 0.000000 | 59.579735, 17.511978, 0 | × |
| 2025/01/01 03:49:58PM | 0.000000 | 59.579754, 17.512011, 0 | × |
| 2025/01/01 03:49:30PM | 0.000000 | 59.57975, 17.512327, 0 | × |

Map:




Code works as expected.

**c) Test device on the road**

Device on the road was able to get only 2 positions before it lost connection. This could be due to several reasons:

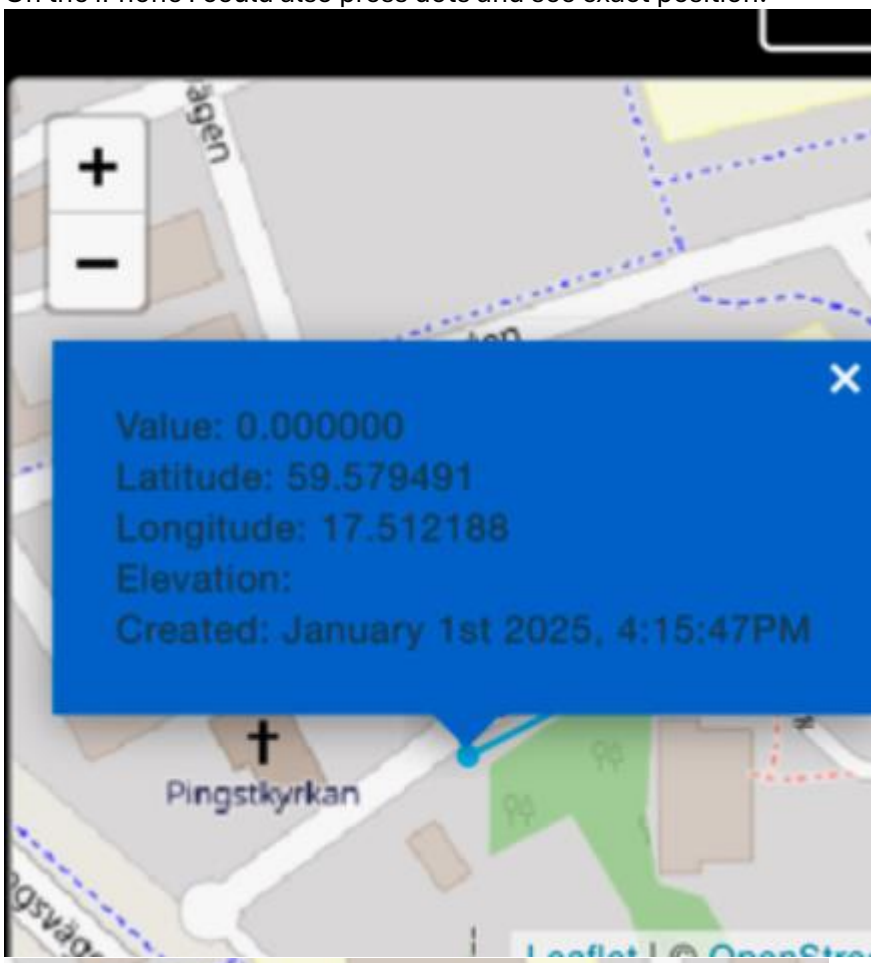
- Snow and clouds - when tested it started to snow, and it was very cloudy
- I lost LTE signal, so I needed to restart device several time
- Jumpers connecting battery to the device detached, which also led to restart of device
- Short time on the road – because of snow and clouds I shortened road trip and will try another day with better weather conditions

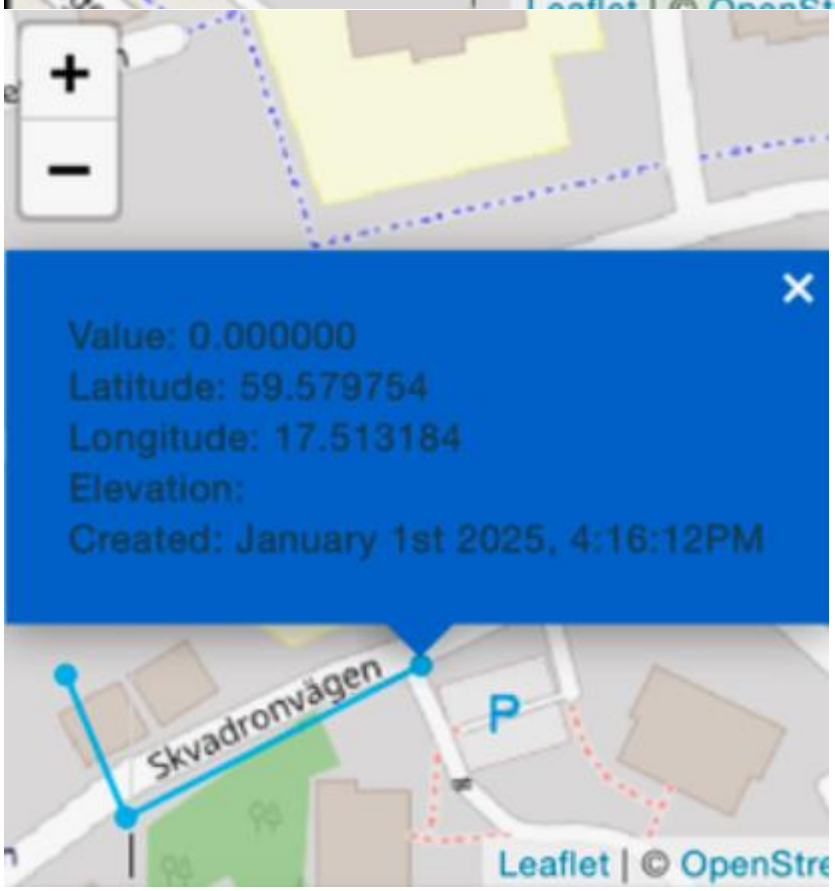
I got these positions:

| | | | |
|---|---|---|---|
| 2025/01/01 04:16:12PM | 0.000000 | 59.579754, 17.513184, 0.0 | × |
| 2025/01/01 04:15:47PM | 0.000000 | 59.579491, 17.512188, 0.0 | × |

Which on map showed like this:

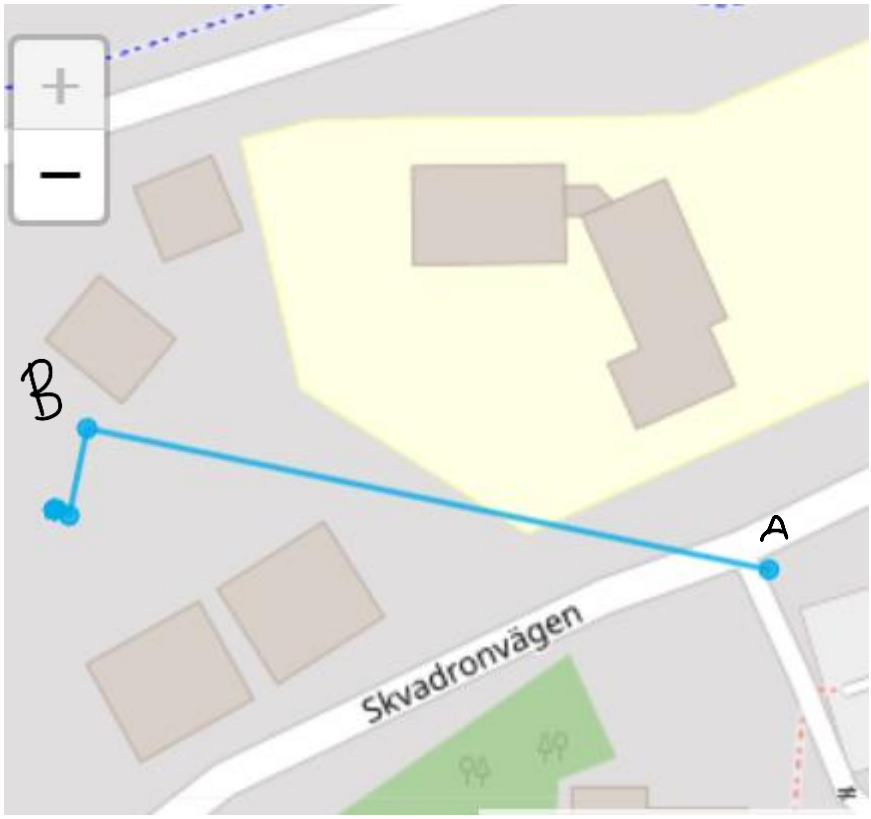
On the iPhone I could also press dots and see exact position:



When returned home, and position GPS again on windowsill, I got again on track with positioning.

| | | | |
|---|---|---|---|
| 2025/01/01 04:58:24PM | 0.000000 | 59.579807, 17.511908, 0 | × |
| 2025/01/01 04:57:54PM | 0.000000 | 59.579807, 17.511908, 0 | × |
| 2025/01/01 04:56:54PM | 0.000000 | 59.579807, 17.511908, 0 | × |
| 2025/01/01 04:56:24PM | 0.000000 | 59.579807, 17.511913, 0 | × |
| 2025/01/01 04:55:54PM | 0.000000 | 59.579803, 17.511934, 0 | × |




This shows path from last known position and current measurements.

A = last position on the road
B = position on windowsill

**Conclusion:**

Device can obtain position and send data to Adafruit IO feed via MQTT protocol. Issues lies on locking GPS position during bad weather and unstable connections to the battery.

**Next:**

- Test during better weather conditions
- Try to secure more stable connection to battery
- Make several tests to get data on the road

## Day 4: More testing

**Duration: 5 hours**

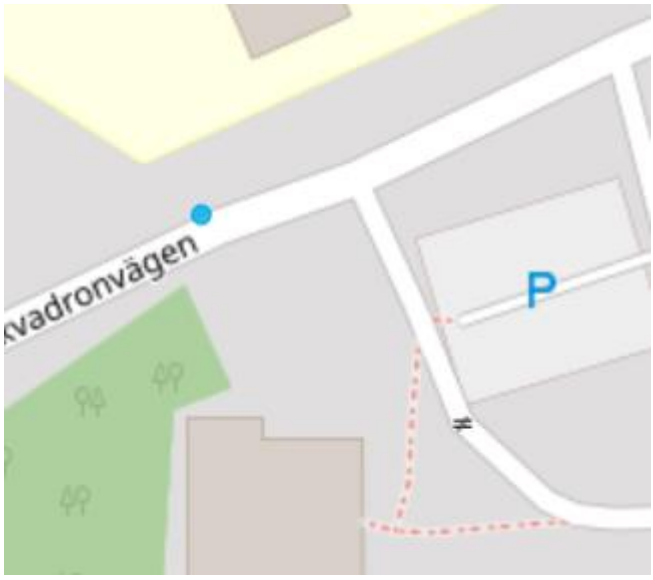
**a) Testing outdoors**

**Test 1:**
I tested device outdoor during nice weather, however device could send around 6 readings before it stopped working, probably because in cold weather device can malfunction.
I also got readings only on feed, but on the map is only reading visible.

Readings:

| | | |
|---|---|---|
| 2025/01/02 01:52:41PM | 0.000000 | 59.579739, 17.512856, 0.0 |
| 2025/01/02 01:52:11PM | 0.000000 | 59.579594, 17.512314, 0.0 |
| 2025/01/02 01:51:41PM | 0.000000 | 59.579575, 17.512138, 0.0 |
| 2025/01/02 01:51:11PM | 0.000000 | 59.579556, 17.512037, 0.0 |
| 2025/01/02 01:50:41PM | 0.000000 | 59.579556, 17.512037, 0.0 |
| 2025/01/02 01:50:11PM | 0.000000 | 59.579536, 17.512093, 0.0 |
| 2025/01/02 01:49:40PM | 0.000000 | 59.579597, 17.512196, 0.0 |

Map:



**Test 2:**
Another short walk with device got no readings. When GPS-module is placed at windowsill and device indoors, it gets data again.

| | | |
|---|---|---|
| 2025/01/02 03:42:40PM | 0.000000 | 59.579758, 17.511986, 0 |
| 2025/01/02 03:42:10PM | 0.000000 | 59.579758, 17.511992, 0 |
| 2025/01/02 03:41:40PM | 0.000000 | 59.579765, 17.512011, 0 |

**Conclusion of testing:**

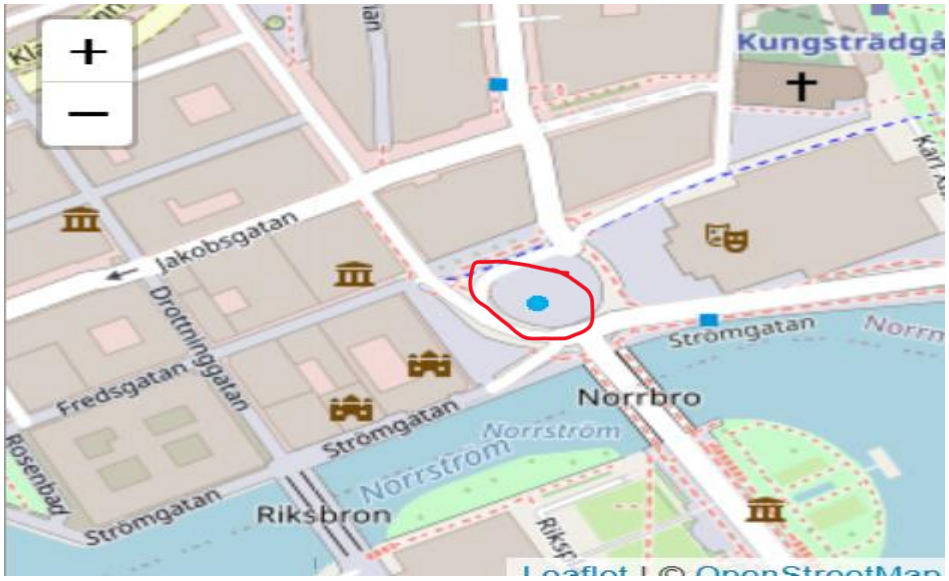
GPS is not reliable because of weather conditions, instead I will try to use combination of GPS positioning and Wi-Fi positioning to try to get more precise location and functionality of device.

**Resolving of problem:**

I use for now **Wi-Fi positioning** only to test functionality. I must use metadata for scanning and then change these data into location because device has trouble with direct HTTP data transfer.

Device could find several Wi-Fi hotspots, but the result location was faulty. On other hand, I know that device can find Wi-Fi hotspots and send data to Adafruit feed, as well as publish location on another feed which shows location on map.

| | | |
|---|---|---|
| 2025/01/02 05:53:02PM | 0.000000 | 59.3293, 18.0686, 0 |
| 2025/01/02 05:52:24PM | 0.000000 | 59.3293, 18.0686, 0 |
| 2025/01/02 05:51:46PM | 0.000000 | 59.3293, 18.0686, 0 |



**Next:**

- Test several code modifications to obtain real location
- Test device on the road
- Implement both Wi-Fi and GPS location for better functioning

**Final Thoughts:**

**GPS positioning is not sufficient** for this project to be used as only solution, mainly because of weather condition (clouds, cold, snow etc). Also, device has **major problem to work with HTTP protocol**, which can be also factory fault. To go around of using this protocol, I implemented **Wi-Fi location**, which is working in order of functionality, but gives faulty position. GPS when working, gives more precise location, however it is not working properly on the road.
To be able to get right position with Wi-Fi location, means also that GPS can be used as back up, or not at all. To skip GPS module could be beneficial for the animal, because this means device is only module which must be attached to the collar. Next goal is to get right location as well as implement GPS as back up.

## Day 5: Testing MQTT request, more Wi-Fi positioning, fixing location data

**Duration: 7 hours**

### A) More testing with MQTT using real location

In previous test I used mock data to publish 😊. This was mistake which I try to make right now. So next test is to use real metadata. However, when tried to convert this metadata to location, it did not work.
For now, I need to make sure that metadata part works.

After short walk I obtained few readings. However, position feed on Adafruit has problem to convert meta-data to location.

### B) Retrieve metadata from Adafruit IO feed

Next step is to retrieve metadata from Adafruit feed into server.js. This works as expected.


```
{
  id: '0FS6Z32KX59KJFE2ZR6H22QY23',
  value: 'B0:0A:D5:D8:32:BF,-45;C0:C9:E3:92:28:DE,-77;6E:A6:E6:B5:E3:0A,-86;C4:F8:39:87:6C:35,-87;18:31:BF:07:4B:D0,-90;7C:10:C9:AF:53:88,-91;C0:C9:E3:92:28:CA,-92;76:97:79:42:59:CC,-93;6E:A6:E6:B5:D3:42,-94;'

  feed_id: 2977122,
  feed_key: 'wifi-metadata',
  created_at: '2025-01-03T14:42:54Z',
  created_epoch: 1735915374,
  expiration: '2025-02-02T14:42:54Z'
},
{
  id: '0FS6Z2PZ3GYP5S8JVMT7GZTZM0',
  value: 'B0:0A:D5:D8:32:BF,-46;C0:C9:E3:92:28:DE,-76;6E:A6:E6:B5:E3:0A,-88;18:31:BF:07:4B:D0,-89;B6:39:56:6D:5D:D8,-91;C0:C9:E3:92:28:CA,-93;',
  feed_id: 2977122,
  feed_key: 'wifi-metadata',
  created_at: '2025-01-03T14:42:16Z',
  created_epoch: 1735915336,
  expiration: '2025-02-02T14:42:16Z'
},
```


### C) Convert metadata into location

After few testings I successfully converted all data from metadata feed, converted into location and sorted from oldest to latest.

Code worked as expected:

```
Location for entry ID 0FS6ZP6KATBRPEJMXKKJ4TG9DS:
Latitude: 59.5796831
Longitude: 17.512016
Location for entry ID 0FS6ZXSQ4HK42NSVBT2887H8G6:
Latitude: 59.5796853
Longitude: 17.5120196
Location for entry ID 0FS6ZM1363ADKM5RJ2Q37W46GZ:
Latitude: 59.579689
Longitude: 17.5120053
```

### D) Store converted locations in JSON file

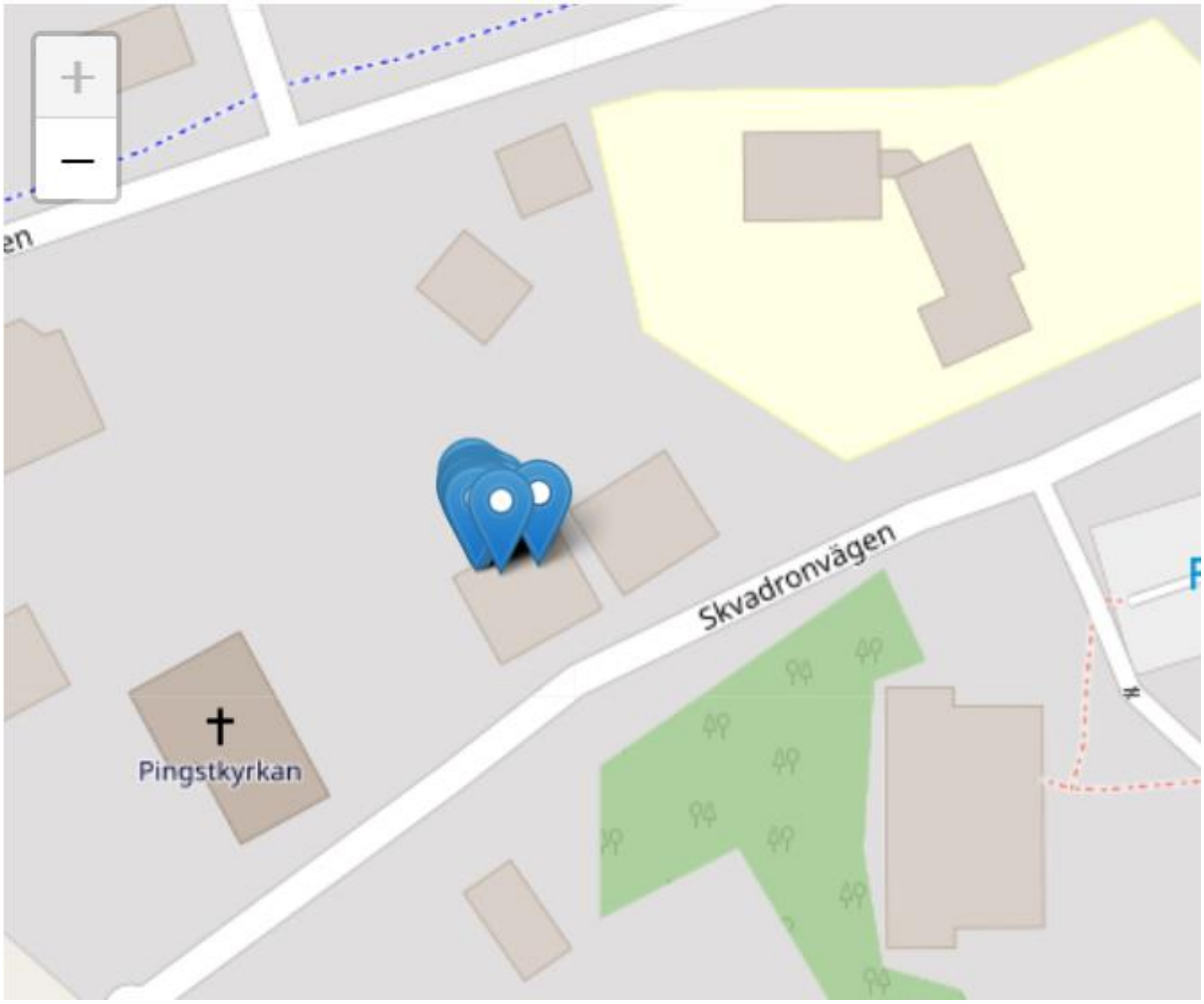
Next step is to store converted locations into JSON file, which I later use for visualisation


```
{
  "entryId": "0FS6YZE93HSWFT558SYXS0V7XZ",
  "latitude": 59.5796917,
  "longitude": 17.5120193,
  "timestamp": "2025-01-03T16:22:39.443Z"
},
```

**E) Show locations on map**

I used Leaflet for visualisations:

# Leaflet Map with Locations




Code works as expected.

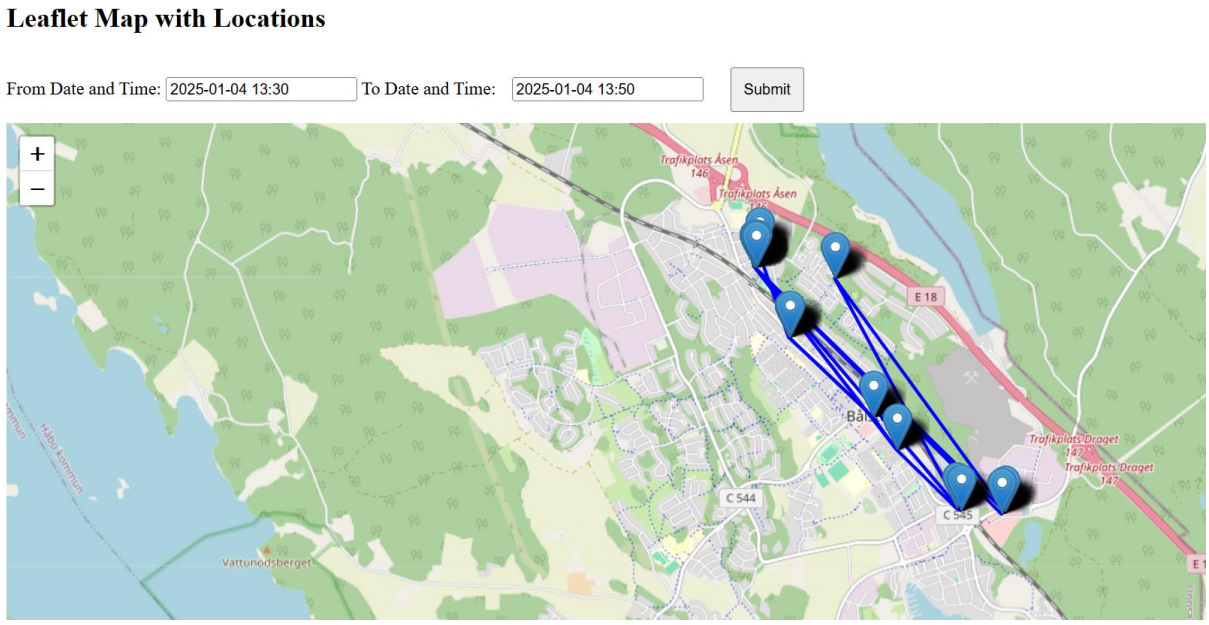
**Next:**

- Modify codes for real time
- Test to obtain more positions to obtain metadata
- Continuously retrieve metadata from Adafruit feed, convert them into location, store them and place them on map
- Create nicer front end for users benefit

## Day 6: Testing Wi-Fi-positioning and front-end

**Duration: 4 hours**

Today I tested Wi-Fi-positioning. I went to affair with device to check if all is working.
I got many data entries and got them converted and placed on map, however I needed to adjust code for filtering data, because map is constantly downloading and changing. To avoid this, I try to change code just to show date and time frame from which I want data to show.




Code works as expected.

**Conclusion:**

Wi-Fi positioning is better solution for tracking the device. It doesn't depend on GPS satellites, weather condition or obstructs in the way. It also means that additional GPS-antenna doesn't need to be attached which increases comfort for the animal.

Now when I have successfully achieved functionality I aimed for, I can focus now on easier implementations for the user.

## Day 7: MongoDB storage

**Duration: 5 hours**

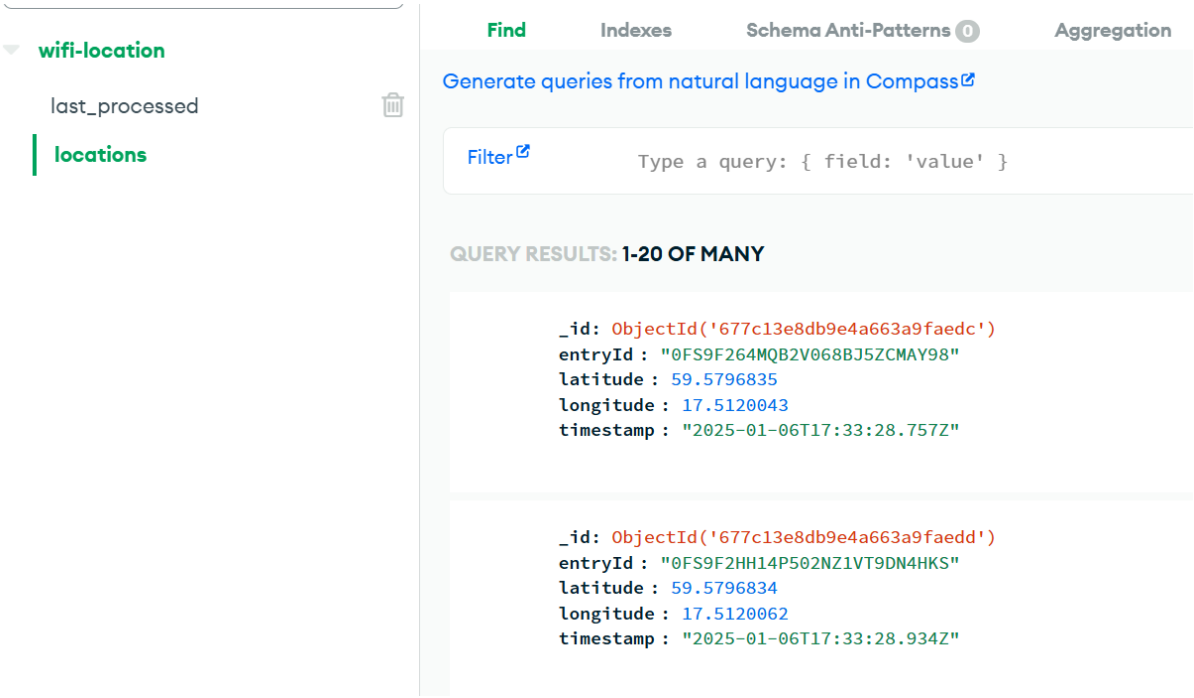
Basic of my project tis done and because there is time left; I decided to use this time for implement more realistic features into it. Firstly, focus was on use of external database, which allows run project from anywhere. For this I created account on MongoDB. Because MongoDB doesn't support MQTT protocol, and because my device doesn't support HTTP protocol, I needed to make some way around it.

**Schema I work with looks like this:**

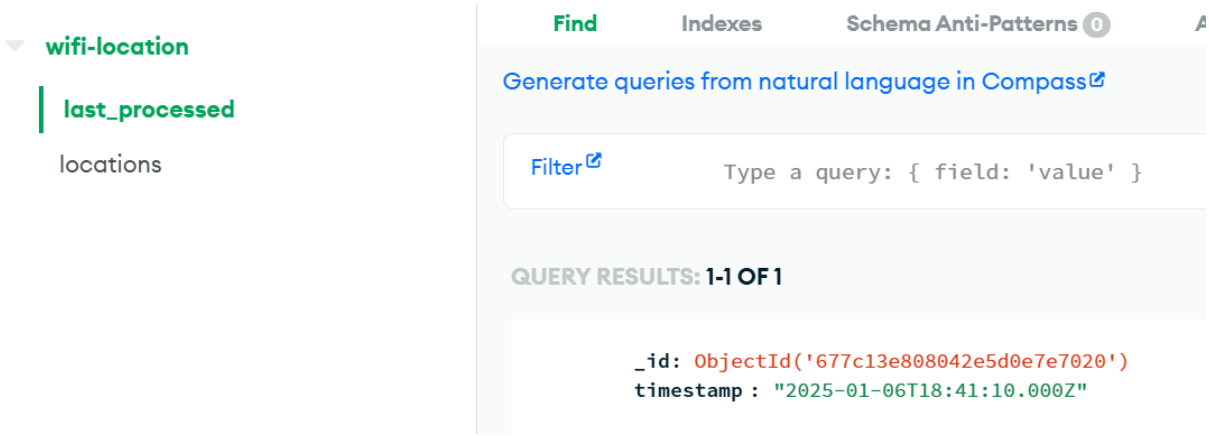
1. Device sends Wi-Fi hotspots via MQTT protocol into Adafruit feed (wifi-meta).

| Created at | Value | Locati |
|---|---|---|
| 2025/01/06 07:41:10PM | ViewB0:0A:D5:D8:32:BF,-43;... | |
| 2025/01/06 07:40:32PM | ViewB0:0A:D5:D8:32:BF,-43;... | |
| 2025/01/06 07:39:55PM | ViewB0:0A:D5:D8:32:BF,-43;... | |

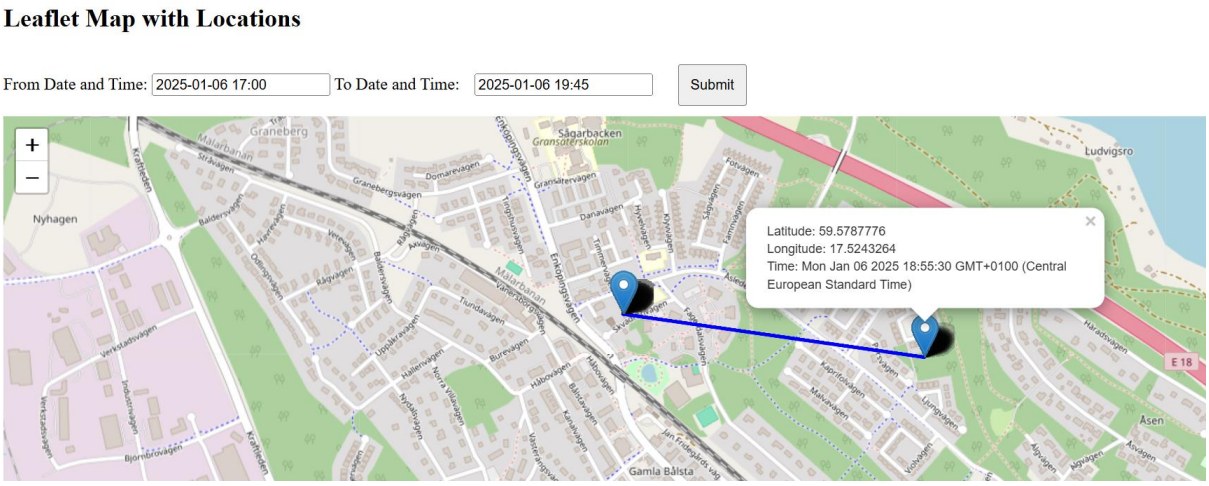
2. Server.js retrieve these data via HTTP protocol, convert them into location and save these locations on cluster in MongoDB.



3. Server.js also stores last processed data. This makes code to resume and store data from last entry point, not all of data again. In other words, it continues to retrieve data from Adafruit feed.

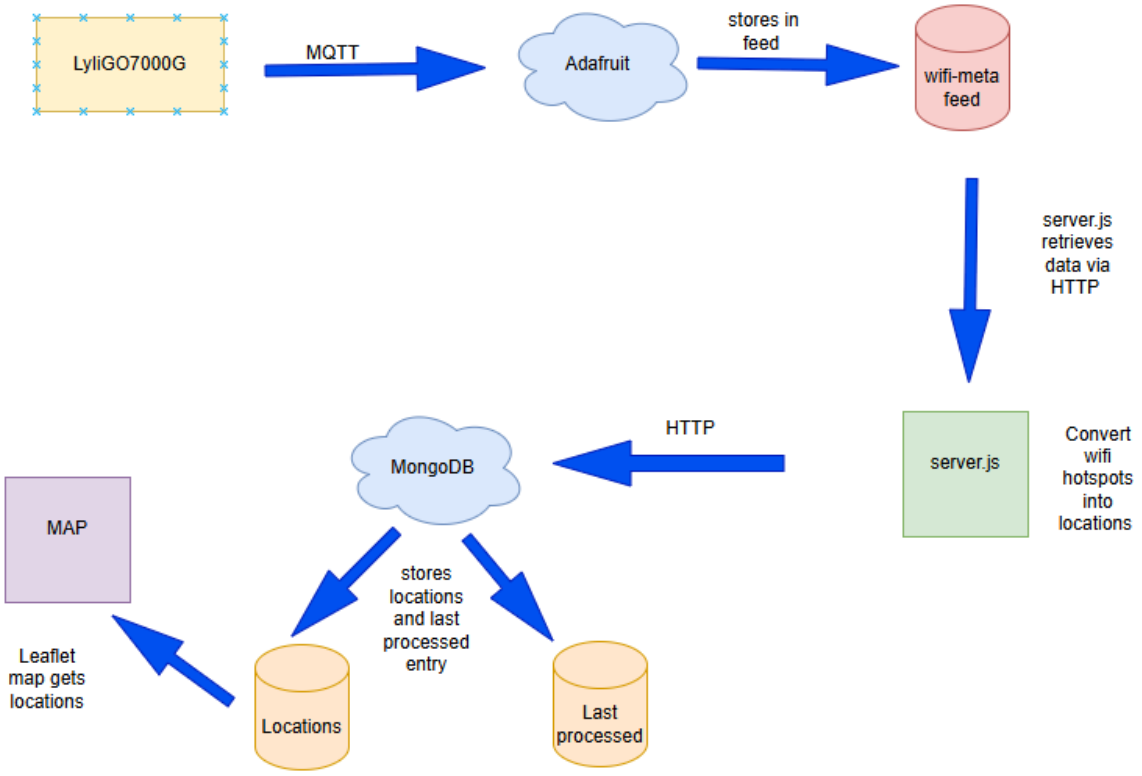



4. Map.html then retrieves locations from MongoDB and visualise them on Leaflet map.




With this approach I solved problem with protocols, which allowed me to show position on map successfully.

**Schema of communication:**




**Challenges:**

- I needed to combine both MQTT and HTTP protocol, which is due faulty device (probably factory fault, in documentation it should support HTTP, however with testing I was not successful)
- Find right syntax for retrieving data
- Fix timestamp on MongoDB – on Adafruit feed I got right time, however on MongoDB it showed 1 hour less on timestamps. I could fix timestamps on "locations" database, but not on "last_processed" database. This means that if I look on last processed entry, I just know it shows 1 hour less then is original time. This is not big problem for project, because timestamp inside of location is right.

**Next:**

- Make more testing from different locations to make sure all is working
- Implement geofence
- Implement notifications
- Make nicer webpage
- Deploy webpage to my domain
- Make documentation
- Fix GitHub for project with codes excluded of private information (such as Google API key, Adafruit key etc)

## Day 8: Geofence, more data

**Duration: 6 hours**

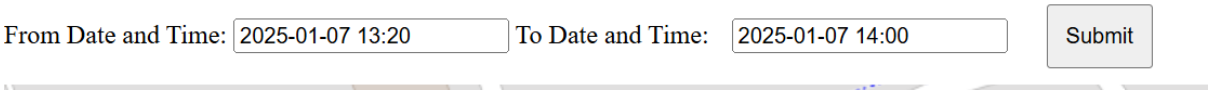
Today I wanted to implement geofence into my project. This geofence is around my house visible on map:

After several tries and errors, radius 13.5 meters is optimal.

```
// Geofence configuration: 13.5 meters around my house (can be adjusted)
const geofenceCenter = [59.5796970, 17.51206];  // Geofence center (my house)
const geofenceRadius = 13.5;  // Geofence radius in meters
```

With this approach I could test to obtain data from other locations and see when device was outside of bounds.

With initial testing, I got problem with time zone. In MongoDB "locations" I have already adjusted time, but on map I got +1 hour in comparison of timestamps.

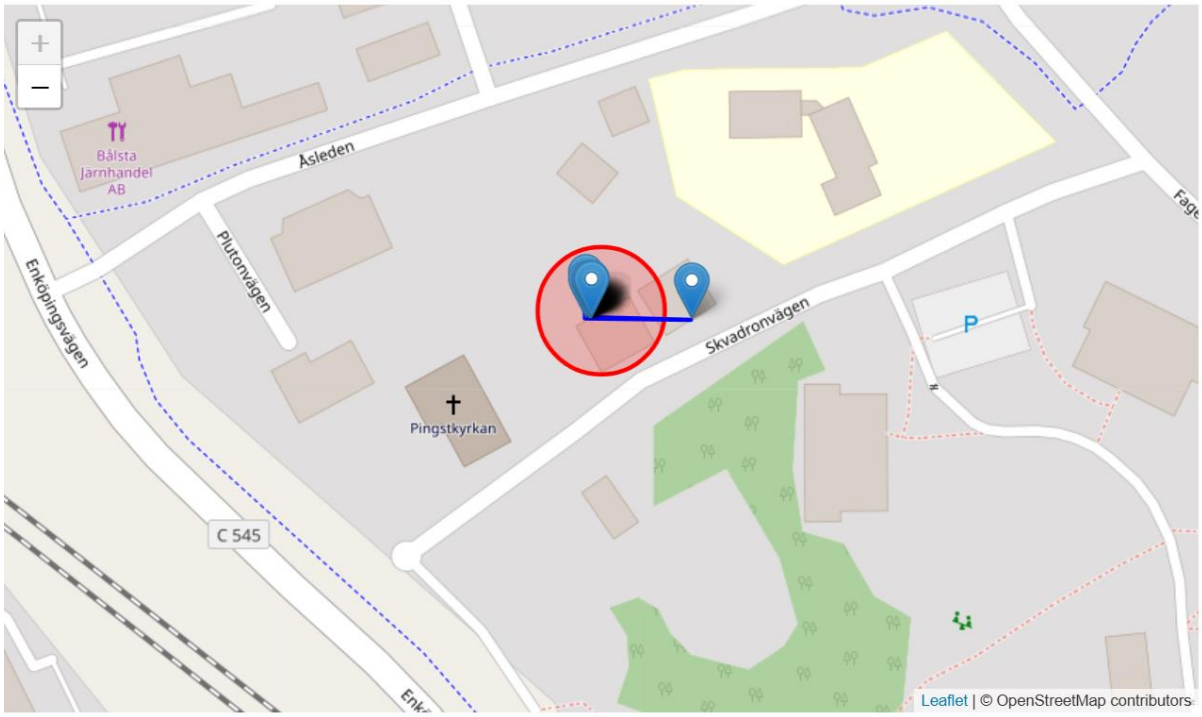



This is not correct time, but I needed to add 1 hour to real time.

After correcting code, I got correct time and data on map:

## Leaflet Map with Locations and Geofence

From Date and Time: 2025-01-07 12:15 To Date and Time: 2025-01-07 13:00 Submit




Here I can also see that geofence works, and with short walk outside of bounds I got satisfying result.

**At this point all desired functionality is implemented:**

- Sending Wi-Fi metadata via MQTT protocol from device to Adafruit feed
- Retrieve metadata from Adafruit feed to server.js via HTTP protocol
- Convert data into location (latitude, longitude)
- Send locations into MongoDB cluster (“locations”)
- Save last processed location, so when restarted program, it continues from last saved entry
- Implementing geofence on map
- Visualisation of location on map, where it is visible when device is outside of boundaries

**Next step:**

- Make nicer webpage,
- Write report
- Make GitHub repository for publishing (i.e. remove sensitive information before publishing)
- Create PowerPoint for presentation
- Submit

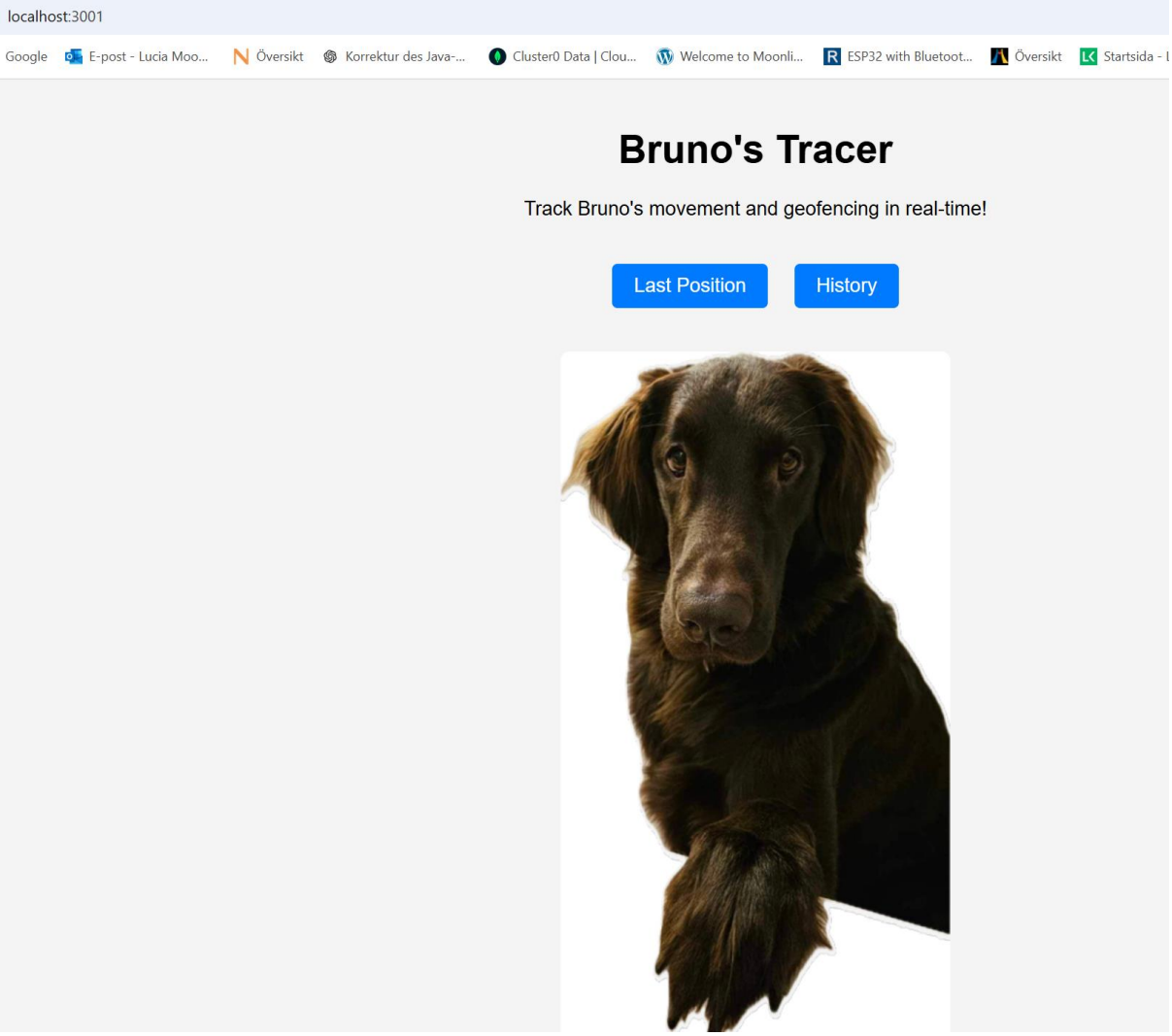
**Nicer Front End:**

At this point I played little bit with front end, so it is nice experience for the user. I created folder "**public**" which contains:

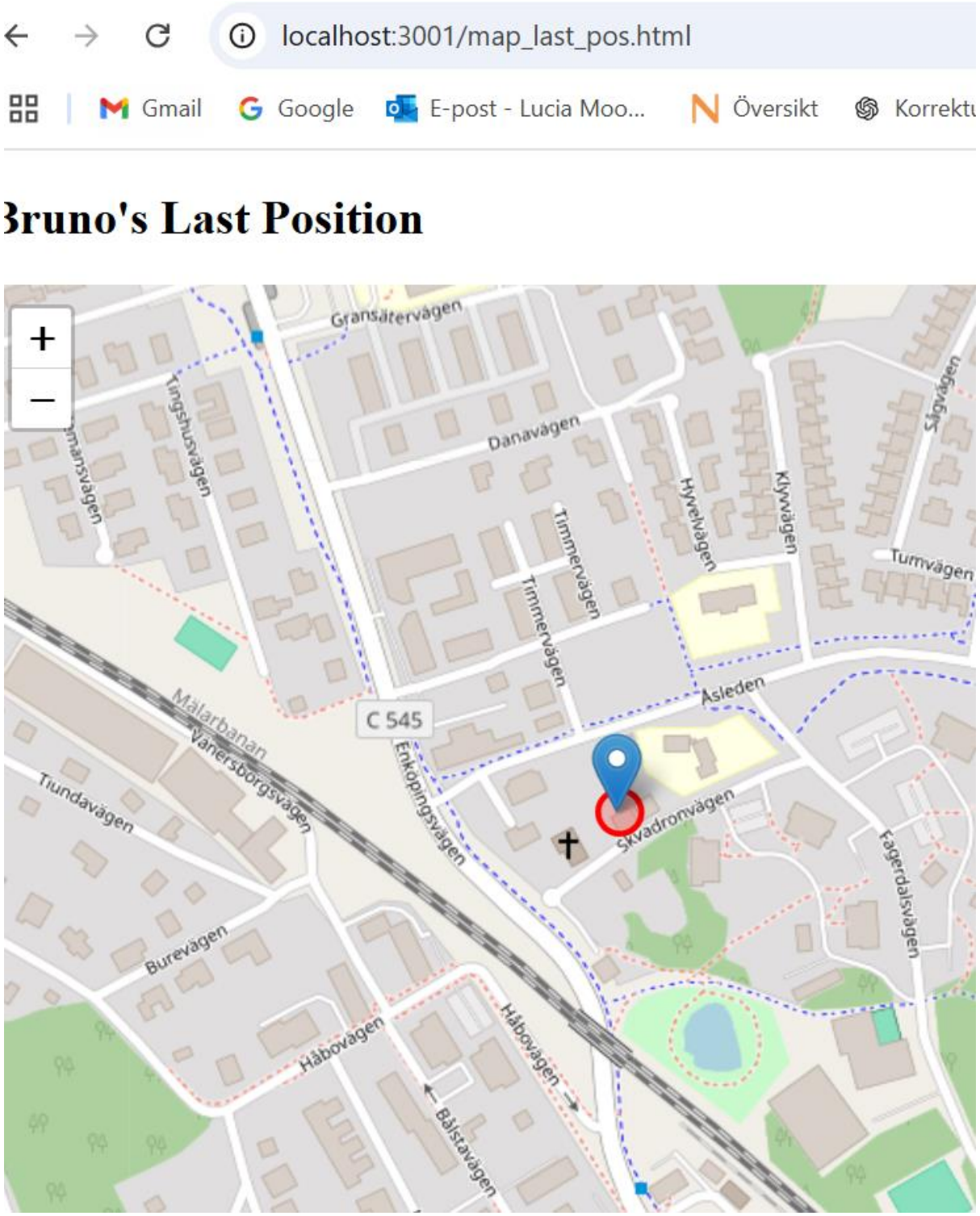
- index.html
- map_history.html
- map_last_pos.html
- images folder which contains picture of my dog Bruno

It looks like this:
When start ***node server.js***

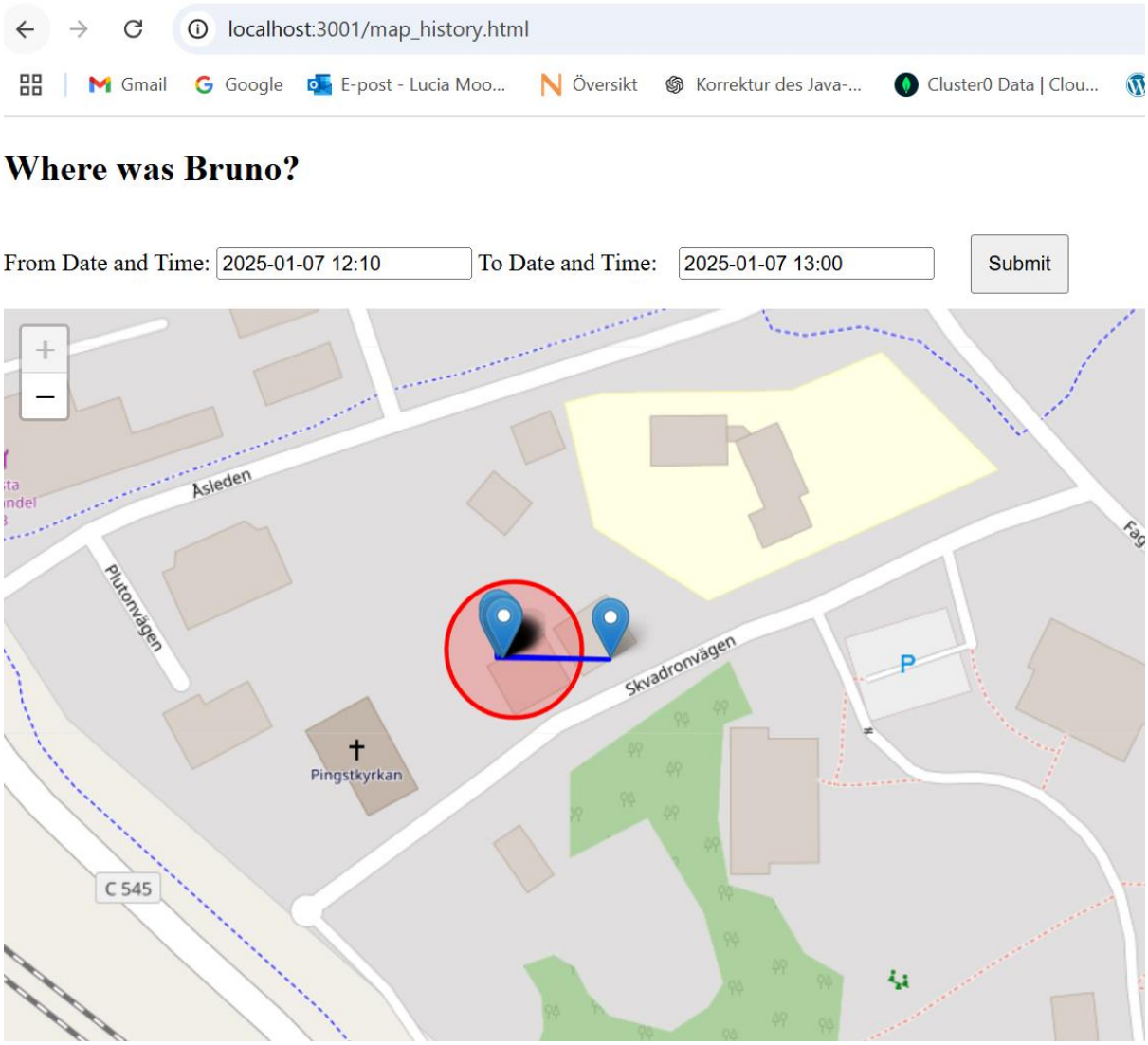
**First comes index.html**



**If chosen Last Position button:**



**When chosen history:**




**Achievements:**

- Successfully send data via MQTT protocol to Adafruit feed
- Successfully retrieved data from Adafruit feed, convert them into location
- Save locations into MongoDB database together with last location
- Successfully retrieve locations and implement visualisation
- Nice front end with options and picture
- Geofence on both maps
- Map with last position
- Map with history with option to choose date and time

**Project took 44 hours to complete.**