Real-Time Location Tracking System

Submitted by: Dipen Gaihre

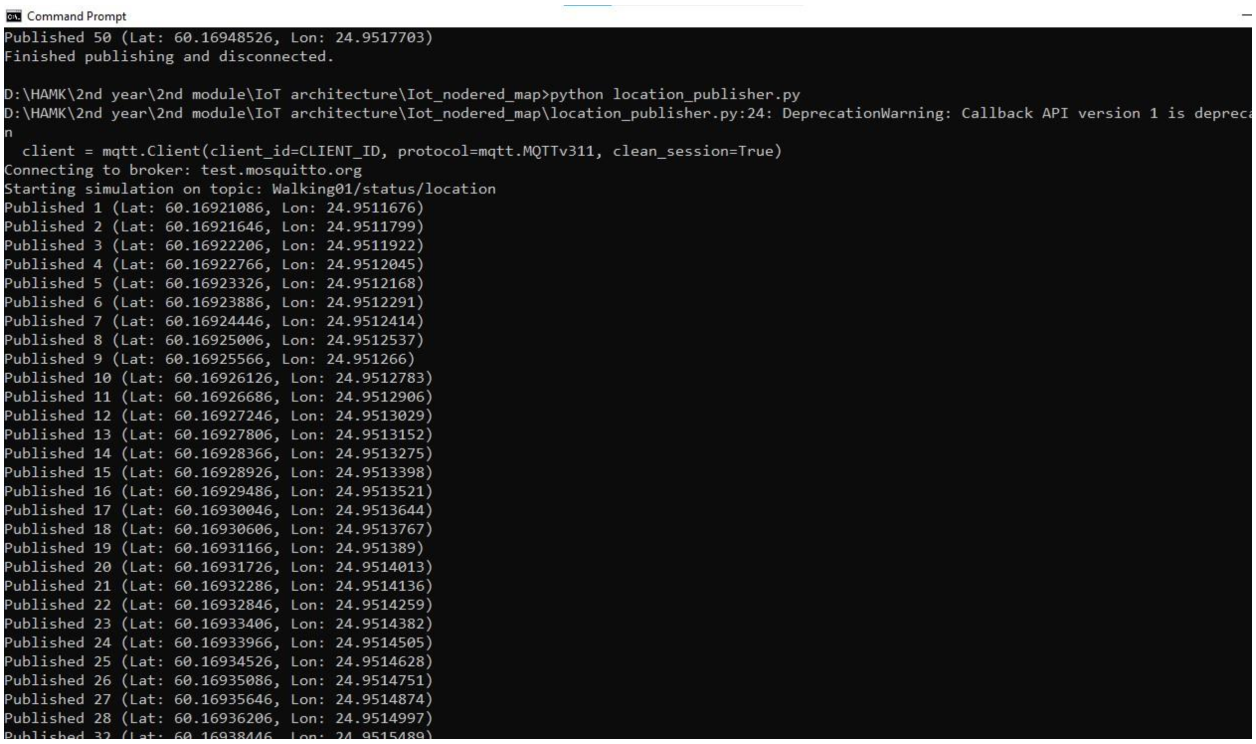
Overview

This document summarizes the basic implementation of an IoT location tracking system. The goal was to simulate a person walking using a Python script and visualize their real-time movement on a web-based map using Node-RED and MQTT.

Implementation & Evidence

1. Data Generation (Python)

created a Python script that calculates new coordinates every second to simulate a person walking. This data is converted into a JSON object and published to a public MQTT broker.



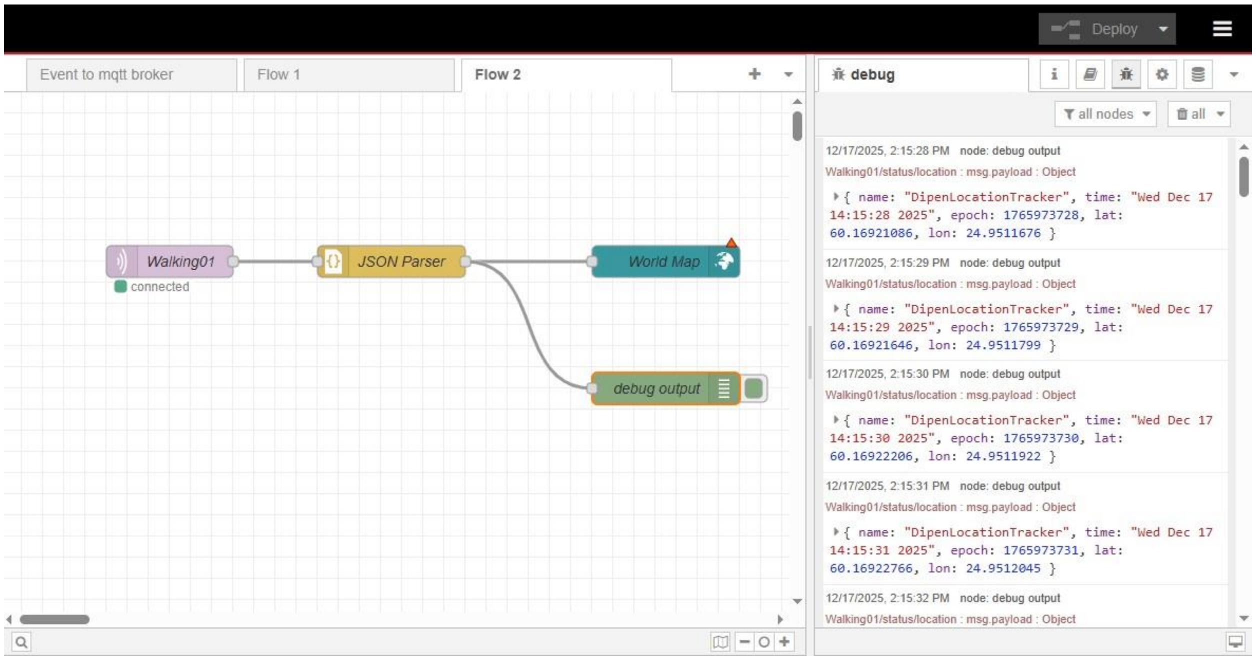
```
Published 50 (Lat: 60.16948526, Lon: 24.9517703)
Finished publishing and disconnected.

D:\HAMK\2nd year\2nd module\IoT architecture\Iot_nodered_map>python location_publisher.py
D:\HAMK\2nd year\2nd module\IoT architecture\Iot_nodered_map\location_publisher.py:24: DeprecationWarning: Callback API version 1 is depreca
n
  client = mqtt.Client(client_id=CLIENT_ID, protocol=mqtt.MQTTv311, clean_session=True)
Connecting to broker: test.mosquitto.org
Starting simulation on topic: Walking01/status/location
Published 1 (Lat: 60.16921086, Lon: 24.9511676)
Published 2 (Lat: 60.16921646, Lon: 24.9511799)
Published 3 (Lat: 60.16922206, Lon: 24.9511922)
Published 4 (Lat: 60.16922766, Lon: 24.9512045)
Published 5 (Lat: 60.16923326, Lon: 24.9512168)
Published 6 (Lat: 60.16923886, Lon: 24.9512291)
Published 7 (Lat: 60.16924446, Lon: 24.9512414)
Published 8 (Lat: 60.16925006, Lon: 24.9512537)
Published 9 (Lat: 60.16925566, Lon: 24.951266)
Published 10 (Lat: 60.16926126, Lon: 24.9512783)
Published 11 (Lat: 60.16926686, Lon: 24.9512906)
Published 12 (Lat: 60.16927246, Lon: 24.9513029)
Published 13 (Lat: 60.16927806, Lon: 24.9513152)
Published 14 (Lat: 60.16928366, Lon: 24.9513275)
Published 15 (Lat: 60.16928926, Lon: 24.9513398)
Published 16 (Lat: 60.16929486, Lon: 24.9513521)
Published 17 (Lat: 60.16930046, Lon: 24.9513644)
Published 18 (Lat: 60.16930606, Lon: 24.9513767)
Published 19 (Lat: 60.16931166, Lon: 24.951389)
Published 20 (Lat: 60.16931726, Lon: 24.9514013)
Published 21 (Lat: 60.16932286, Lon: 24.9514136)
Published 22 (Lat: 60.16932846, Lon: 24.9514259)
Published 23 (Lat: 60.16933406, Lon: 24.9514382)
Published 24 (Lat: 60.16933966, Lon: 24.9514505)
Published 25 (Lat: 60.16934526, Lon: 24.9514628)
Published 26 (Lat: 60.16935086, Lon: 24.9514751)
Published 27 (Lat: 60.16935646, Lon: 24.9514874)
Published 28 (Lat: 60.16936206, Lon: 24.9514997)
Published 32 (Lat: 60.16938446, Lon: 24.9515489)
```

SS 1: Python Publisher

A screenshot of the Command Prompt showing the Python script running and messages 1–50 being successfully published.

2. Data Processing (Node-RED)

configured a Node-RED flow to subscribe to the MQTT topic, parse the incoming JSON data, and pass it to the World Map node.



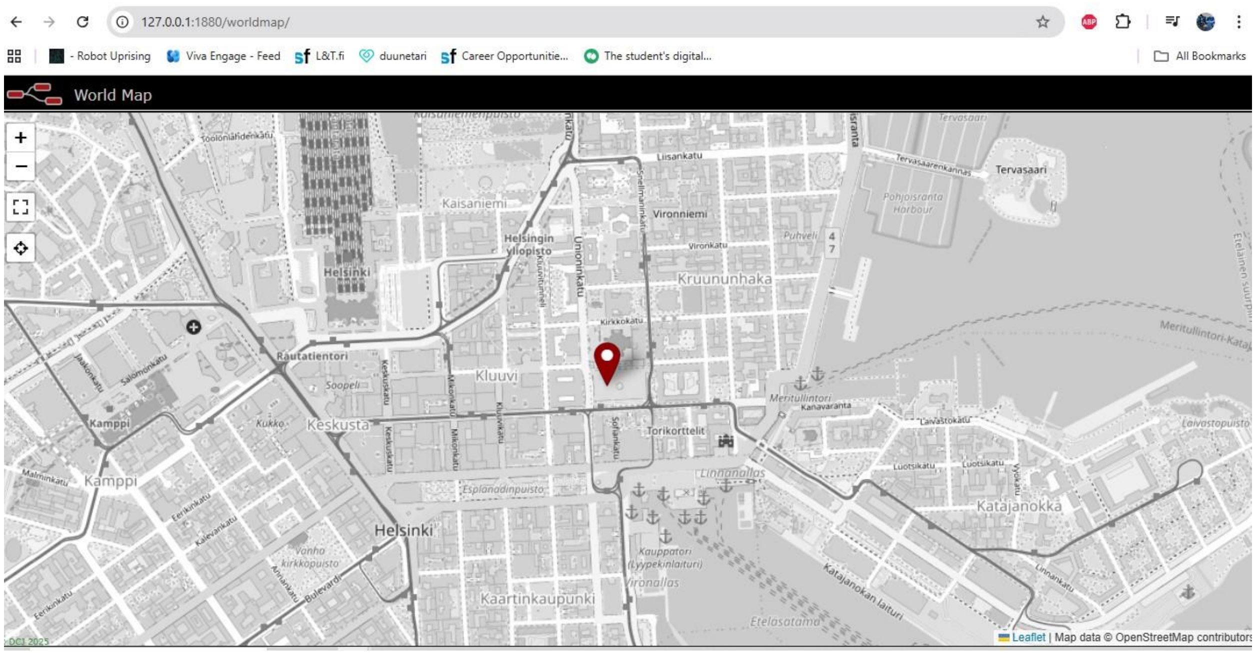
```
12/17/2025, 2:15:28 PM   node: debug output
Walking01/status/location : msg.payload : Object
▸{ name: "DipenLocationTracker", time: "Wed Dec 17
14:15:28 2025", epoch: 1765973728, lat:
60.16921086, lon: 24.9511676 }

12/17/2025, 2:15:29 PM   node: debug output
Walking01/status/location : msg.payload : Object
▸{ name: "DipenLocationTracker", time: "Wed Dec 17
14:15:29 2025", epoch: 1765973729, lat:
60.16921646, lon: 24.9511799 }

12/17/2025, 2:15:30 PM   node: debug output
Walking01/status/location : msg.payload : Object
▸{ name: "DipenLocationTracker", time: "Wed Dec 17
14:15:30 2025", epoch: 1765973730, lat:
60.16922206, lon: 24.9511922 }

12/17/2025, 2:15:31 PM   node: debug output
Walking01/status/location : msg.payload : Object
▸{ name: "DipenLocationTracker", time: "Wed Dec 17
14:15:31 2025", epoch: 1765973731, lat:
60.16922766, lon: 24.9512045 }

12/17/2025, 2:15:32 PM   node: debug output
Walking01/status/location : msg.payload : Object
```

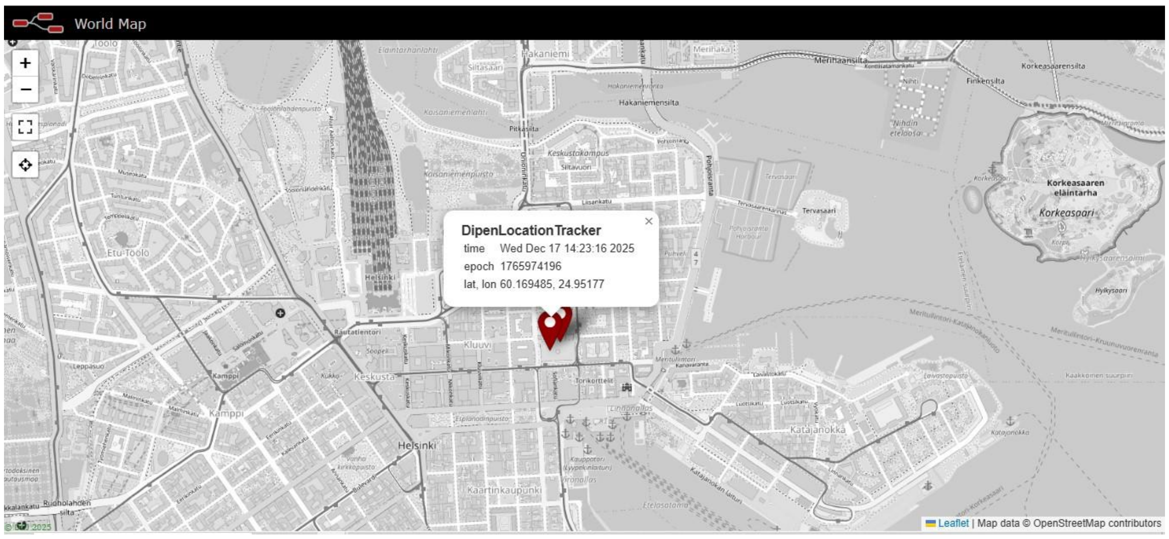
SS 2: Node-RED Flow and Debug Sidebar

A screenshot of the Node-RED workspace showing the MQTT input, JSON parser, and World Map node connected and the Node-RED debug window showing the parsed coordinates (lat, lon) and the device name "DipenLocationTracker".

3. Map Visualization (World Map)

The final step was displaying the data on a map. The World Map node uses the incoming coordinates to place a marker at the exact location.





DipenLocationTracker
time Wed Dec 17 14:23:16 2025
epoch 1765974196
lat, lon 60.169485, 24.95177

SS 3: Live Map

A screenshot of the browser at 127.0.0.1:1880/worldmap showing the marker in Helsinki.

Conclusion

The basic task was successful. The system correctly transmits data from a local Python simulation to a cloud-based MQTT broker and visualizes it locally on a map without any data loss.

The end