Objectives

- Gain experience with the MQTT protocol.
- Learn to build a simple IoT Geoweb app

Overview

In this project, you'll build a simple web application that uses the javascript geolocation API and turn any smartphone into an IoT sensor. You will also build a simple web mapping application to visualize the location of the smartphone sensor.

Getting Started

MQTT

MQTT stands for Message Queuing Telemetry Transport which is known as one of the most used protocols in IoT networks. MQTT is designed to enable Machine to Machine communication in a publish/subscribe architecture. You will no longer need to stack your messages in the queue and wait for a response to send other messages to a server. MQTT is a lightweight protocol that is able to work in low-bandwidth networks as well as networks with various latency levels. One of the most popular applications of MQTT is to get the latest changes of devices/sensors' status and visualize them on a web client application. In this project, you will get the latest location of your device by MQTT protocol and visualize it on the map. You can watch this short video to get familiar with MQTT terms and concepts.

Leaflet

For this project, you'll need to use Leaflet.js as the JavaScript library for interactive maps. Leaflet is one of the most popular web mapping APIs. You will find the API Docs here, Tutorials here, and some very powerful Plug-ins here.

MQTT over WebSockets JavaScript client

Web browsers, often, do not support built-in MQTT protocols. Therefore, we recommend using WebSockets to handle communication between clients and an MQTT message broker. WebSocket is a persistent bidirectional protocol that makes the client/server communication. In this project, you will need to make an MQTT javascript client with WebSockets. We recommend using Paho as a well-known MQTT client library for handling MQTT messages in your web browser.
You should use this script tag to use the Paho js library.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js" type="text/javascript"></script>
```

This short video presents a summary of the Paho functions and methods. To get more detailed information, this video provides you with a simple example of building connections, publishing, subscribing and reading MQTT messages in a web browser. As an example, you might download this web form example and try it on your local web browser to see how it works. You can use

test.mosquitto.org as the MQTT message broker host and 8080 (or 8081) as ports. You are free to use any MQTT topic you want.

To test your web application, you will need to use an MQTT client software like MQTTX. Download and install the software on your local machine, then try to publish your messages from your client browser to the message broker and see your messages by subscribing to the same host and topic in the MQTTX. Also, you might want to test your browser subscription functionality by publishing messages from the MQTTX and seeing the messages in your web browser console. You will need MQTTX in your demo presentation. If you have issues with connecting to test.mosquitto.org in MQTTX, you can use the MQTT protocol and port 1883 for your connection settings.

Geolocation Javascript API

From Mozilla.org, "the Geolocation API allows the user to provide their location to web applications if they so desire. For privacy reasons, the user is asked for permission to report location information." You can learn more about it here: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API

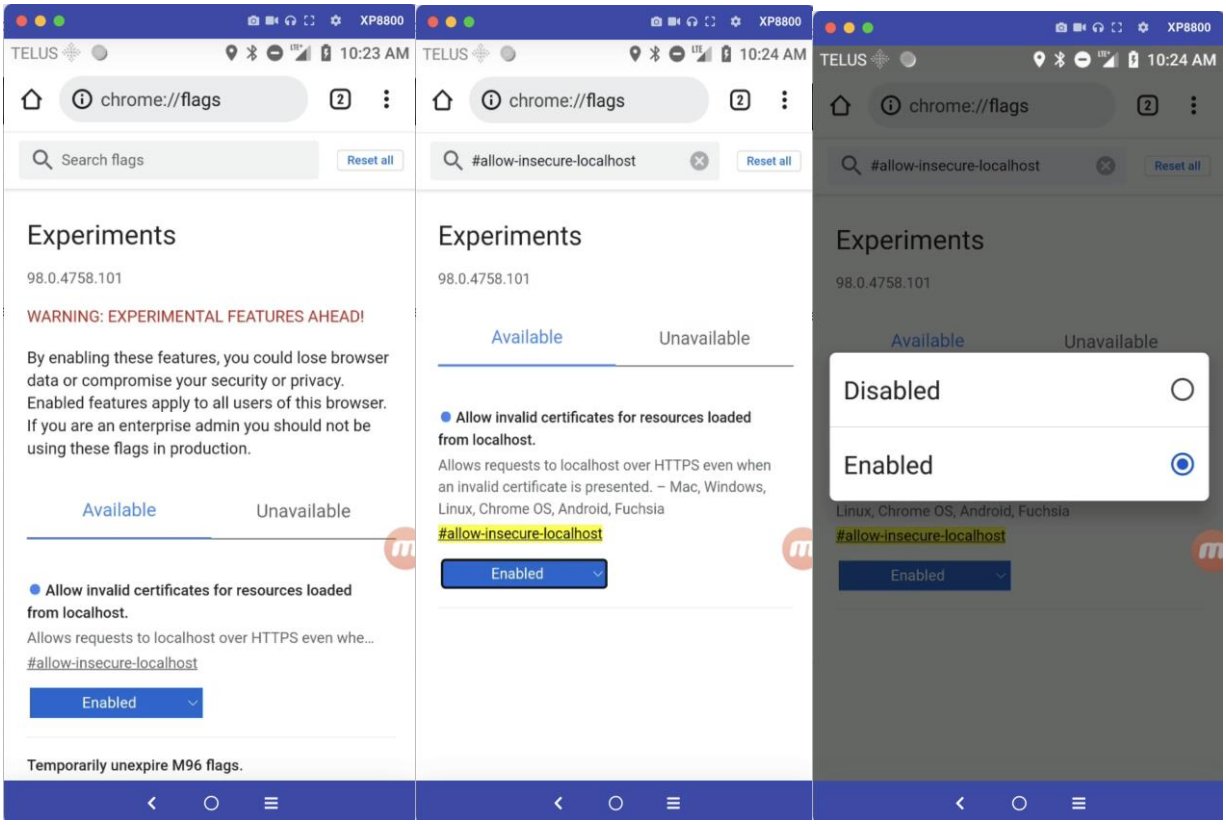
Combining Geolocation Javascript API and MQTT, we will be able to send real-time locations to an MQTT broker, and the subscribers will be able to receive the locations in real-time.

Build your single-page website on GitHub pages

As you will need to develop a web application to be used on smartphones, it will be more convenient to host your web application somewhere and access it from different devices. GithubPages lets you host your web application from your GitHub repo. All you need to do is to follow the instructions here.

Set up your smartphone browser to handle the WebSocket protocol

You can use Edge browser on your smartphone if you do not want to apply extra security settings to enable WebSockets on your device. If you have an android phone and would like to see your web application in Chrome, you should apply the following settings.



If you are using Safari on your IOS device, you will need to enable LBS access to Safari on your phone settings.

NOTE: WebSockets on Chrome browsers on IOS devices have not been tested yet.

Requirements

Alright, it's time to actually build your first Geoweb application! Here are the requirements:

1. Users should be able to determine the MQTT message broker host and port
2. The web application must have a Start/End button to establish/finish a connection with the MQTT message broker. If the user pushed the start button, he would no longer be able to determine host and port values unless he/she clicks on the End button.
3. In case of disconnection, users should receive a proper message and the web application should automatically re-establish the connection.
4. Users should be able to publish any messages to any topics they want and you should show in your demo if MQTTX can subscribe to the topic and read the message that users have just published.
5. You should include the "share my status" button in your app. When a user pushes the button, a Geojson message is generated. The Geojson includes your current location and a random value for the temperature. Then your app should publish it as an MQTT message. You should show in your demo if MQTTX can subscribe to the topic and read the message that users have just published.
    - **Note**: Your MQTT topic should be like this pattern: <your course code>/<your name>/my_temperature. Please use "_" instead of space in your topic.
6. Your map should show your current location by subscribing to the MQTT message broker. When the user clicks on your location icon, she/he should see the current temperature by subscribing

to the message broker with the "<your course code>/<your name>/my_temperature" topic (you should use leaflet popup to show the temperature value). Your location icon color should be changed based on the current temperature. [-40,10) blue. [10,30) green. [30,60] red.

7. In your demo, you should publish the Geojson message from MQTTX and your map should automatically be updated by subscribing to the "<your course code>/<your name>/my_temperature" topic.
8. You are supposed to run your web application on a browser on your smartphone while you are using GPS for your location report. Therefore, your demo should include a recording of your mobile screen. You might use free screen recorder apps for iphone and android in the market or use Vysor to record your demo on your PC or laptop.