Windigipet 2025 – MQTT integration

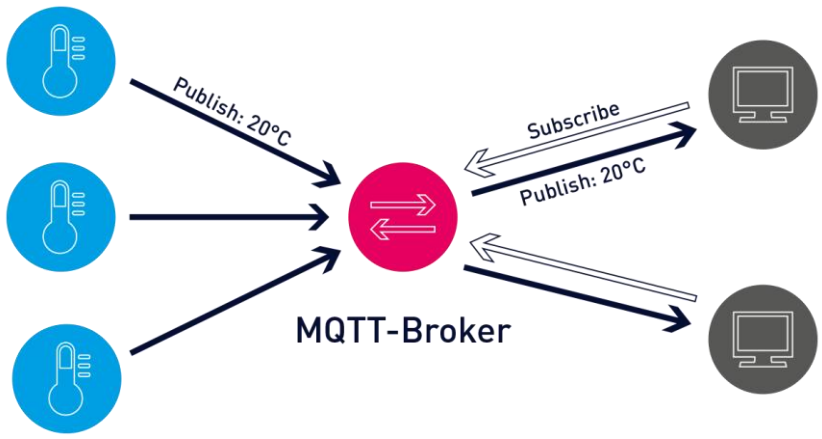
7-7-2025 – Jens Krogsgaard, Copenhagen - Denmark

Table of contents

1. Summary

In the latest version of the program Windigipet 2025 for controlling a model railway, direct integration with MQTT has been introduced. This is actually a really good idea, as this integration makes it possible to communicate directly from Windigipet with, for example, an Arduino or NodeMCU, which many people use on their model railway. I have tested the integration with a small, simple example, and in this document, I describe how you can get started testing the integration. In other words, I'm sharing my experiences to make it easier for others who are interested in trying it out.

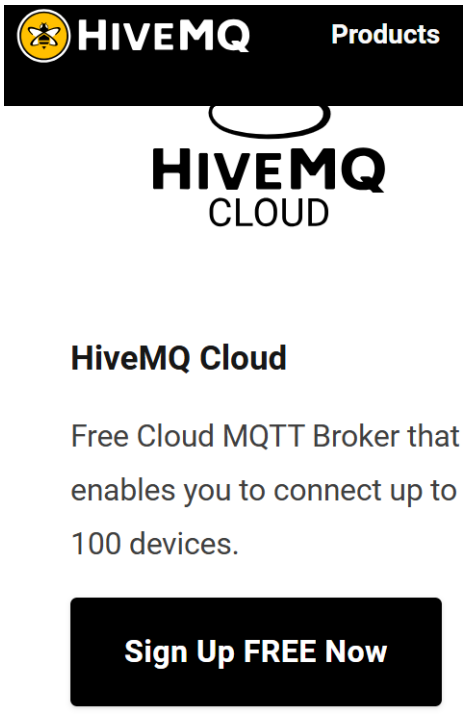
2. Install MQTT-Broker



MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol often used in IoT. Devices (called clients) communicate via a central broker. For example, a thermometer can publish temperature data to a topic, and a PC can subscribe to that topic to receive updates. Clients can also publish commands or data themselves. The broker handles all message distribution between clients.

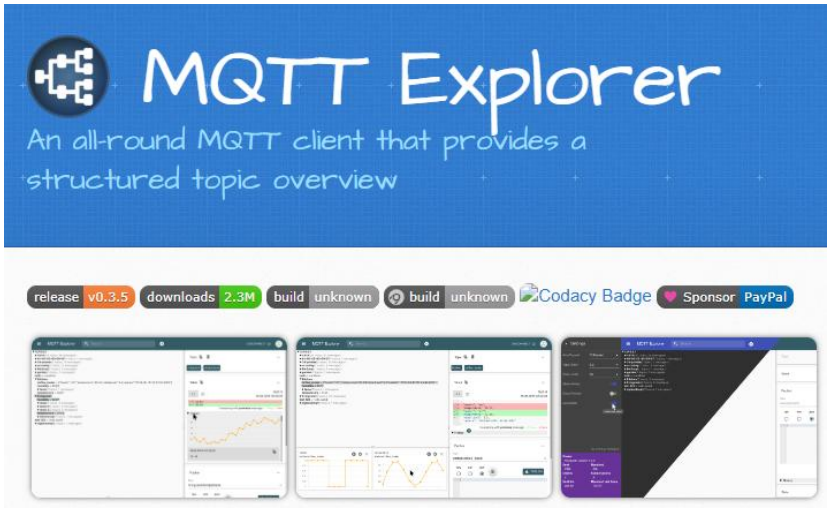
The first step is to get access to an MQTT broker. You can either install one locally on your own PC or subscribe to a cloud-based broker. If you prefer to install it locally, there is a good guide in this document:

If you prefer to try a cloud-based MQTT broker, HiveMQ can be an option. You can create a free account. I have tested this, and it works well.



3. Install MQTT-Broker

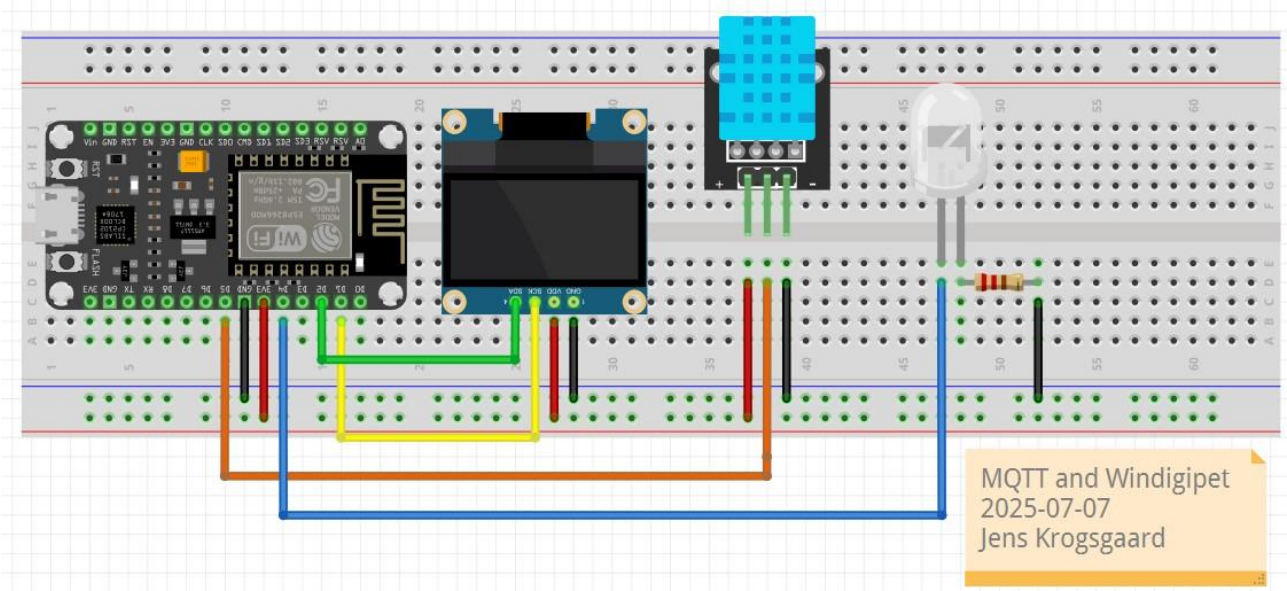
To be able to monitor what is happening on the MQTT broker, it is useful to install an MQTT Explorer. There are several options to choose from – I have chosen this free version:



4. My test project.

I already had a project based on an ESP8266 (NodeMCU) that measured temperature and humidity and displayed the values on a small OLED screen. This project has now been expanded so that the data is sent to my MQTT broker. At the same time, an LED has been installed on the roof of the container. This LED can now be turned on and off from Windigipet (via MQTT). With this simple setup, I can test both publishing and subscribing of data to and from Windigipet.

Here is a diagram of my test project:



You can download my test project from here. In the file config.h, you need to insert your actual values for WiFi and the MQTT broker. **Disclaimer:** This code is provided *as is* and without any warranties. Use it at your own risk.

Download ESP8266 project

A Topic in MQTT is like an address or a channel that devices use to send and receive messages. When a device publishes data, it sends it to a specific topic. Other devices can subscribe to that topic to receive the messages. Topics are organized in a hierarchy using slashes (/), for example: home/livingroom/temperature

When using MQTT with Windigipet, there are a few rules for how the topics must be structured.

When sending data **to** Windigipet, the topic should look like this:

- **WDP/Evt/SD/{Solenoid Address}/State**

Here, {Solenoid Address} is the same address normally used in Windigipet – for example, 110.

Similarly, when sending data **from** Windigipet, the syntax must also be:

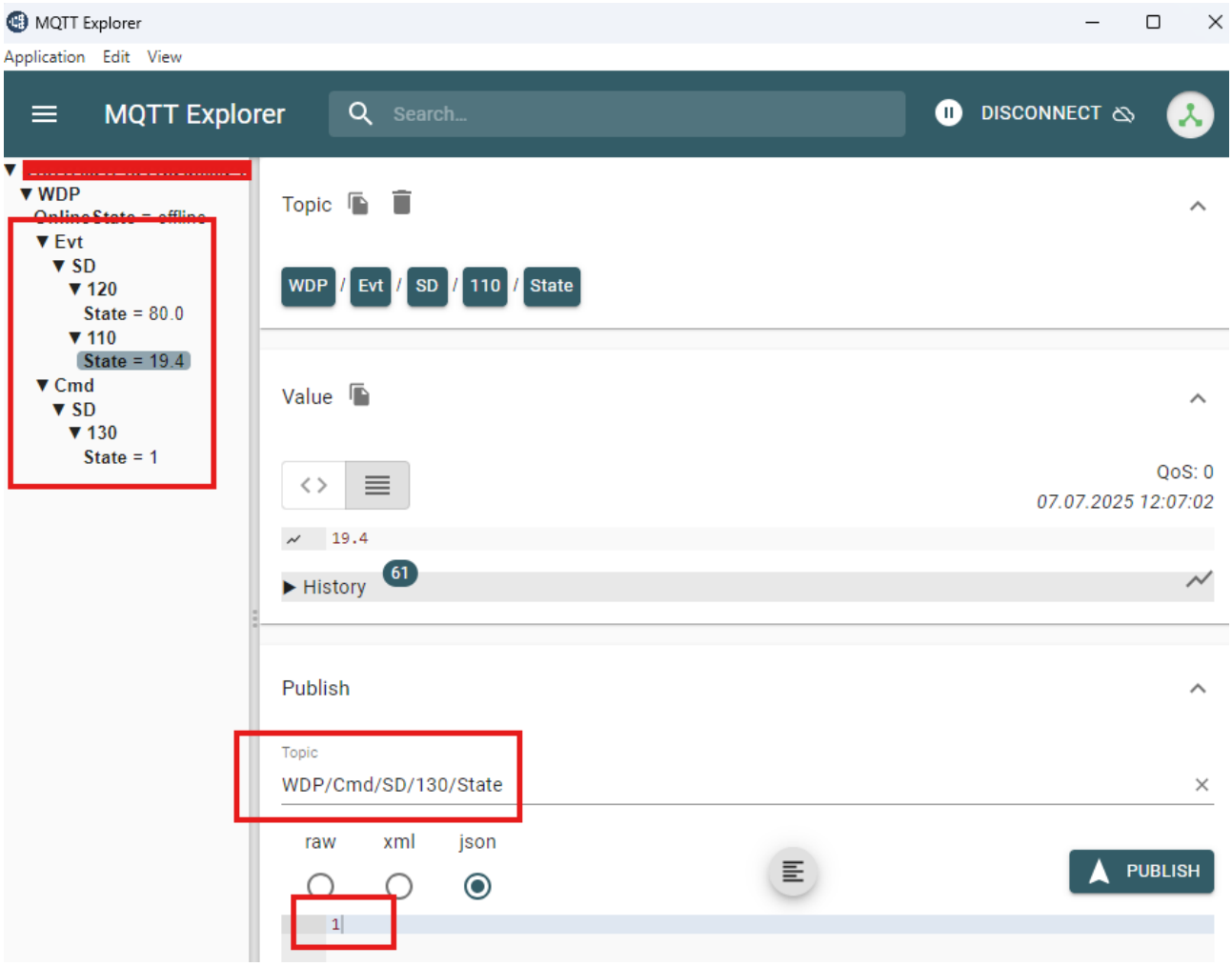
- **WDP/Evt/SD/{Solenoid Address}/State**

In my example, we receive temperature and humidity data on addresses 110 and 120, respectively. Likewise, we send an on/off command to the LED at address 130.

It looks like this in the code:

```
// Temperature MQTT Topics
#define MQTT_PUB_TEMP "WDP/Evt/SD/110/State"
#define MQTT_PUB_HUM "WDP/Evt/SD/120/State"
// Publish on/off LED
#define MQTT_PUB_LED "WDP/Cmd/SD/130/State"
```

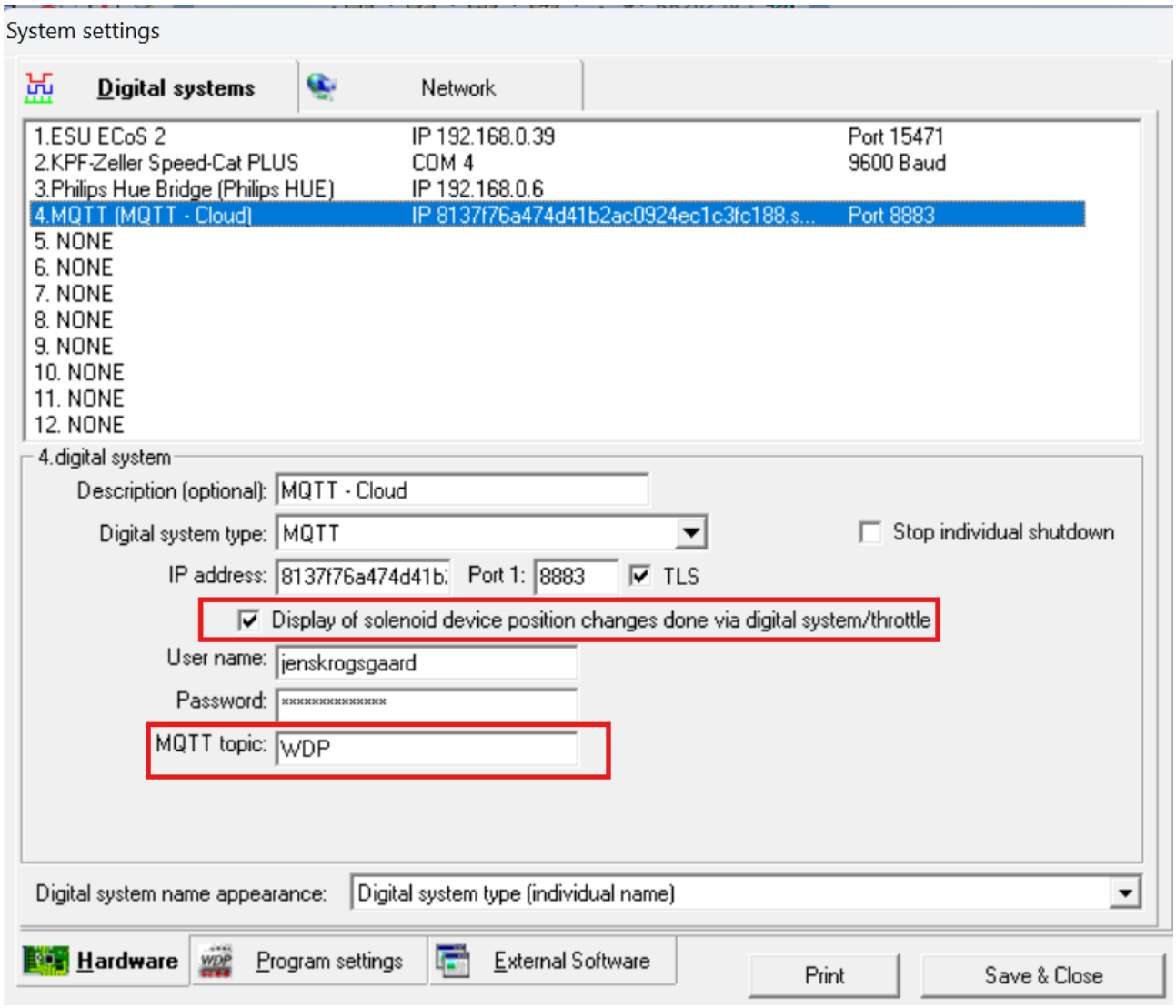
5. Testing in MQTT Explorer



We can now check if data is being sent from the test project to our MQTT broker. As shown here, the data is being received – the temperature is 19.4°C and the humidity is 80%. That's quite high, but it's cool in the basement during the summer, and it's raining outside. Normally, it's lower :)

You can also turn the LED on and off by publishing 1 or 0 to WDP/Evt/SD/130/State.

6. Windigipet 2025 – connect to MQTT

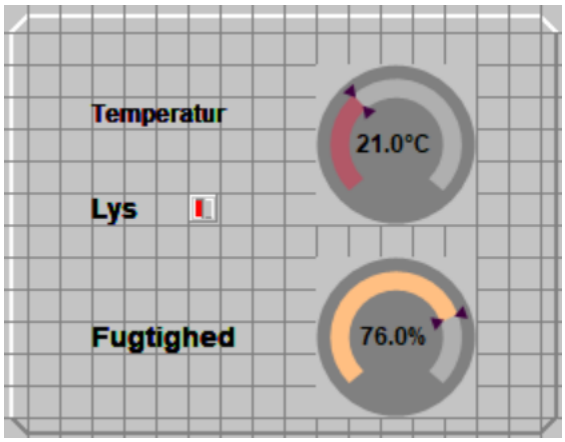


In Windigipet, we connect to MQTT in the usual way. It is important to check the box "Display of solenoid device position", and also to specify WDP under "MQTT topic:".

Username and password only need to be entered if they have been defined on your MQTT broker.

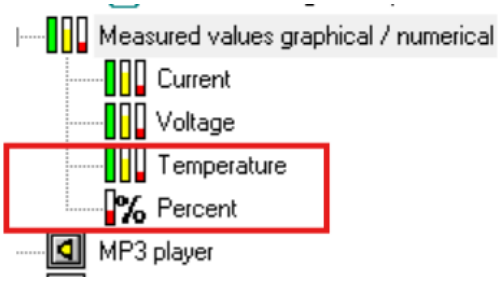
As you can see, I have checked "TLS" – this is not necessary if you install MQTT locally. However, since I'm using HiveMQ in this case, it is required.

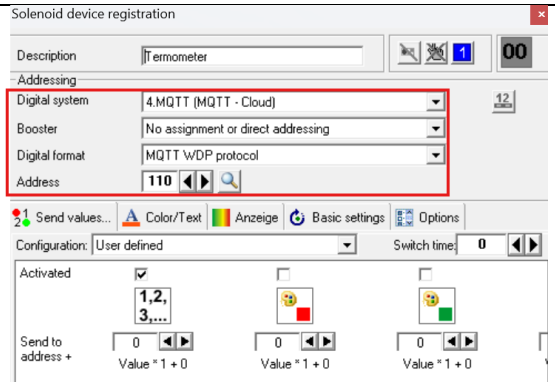
7. Windigipet 2025 – define Solenoid

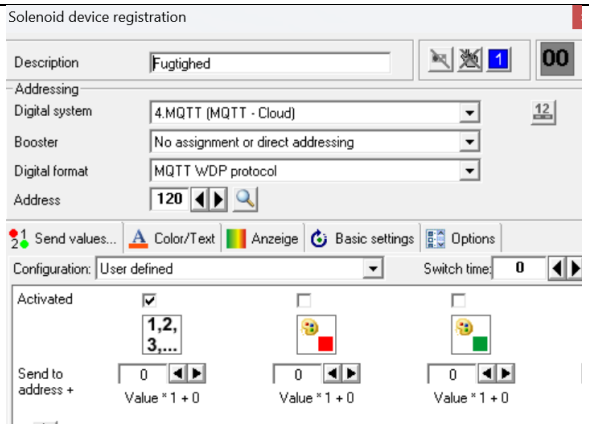
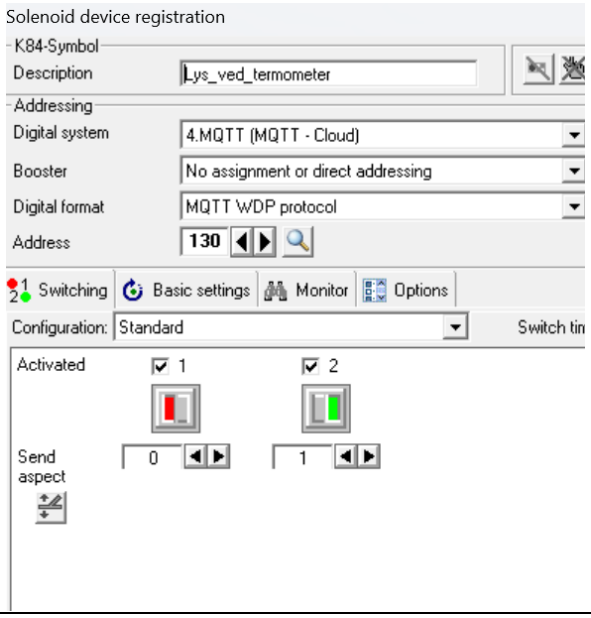


In Windigipet, we create a window with a gauge for temperature and humidity, as well as a button to turn the light on and off on the roof of the container.

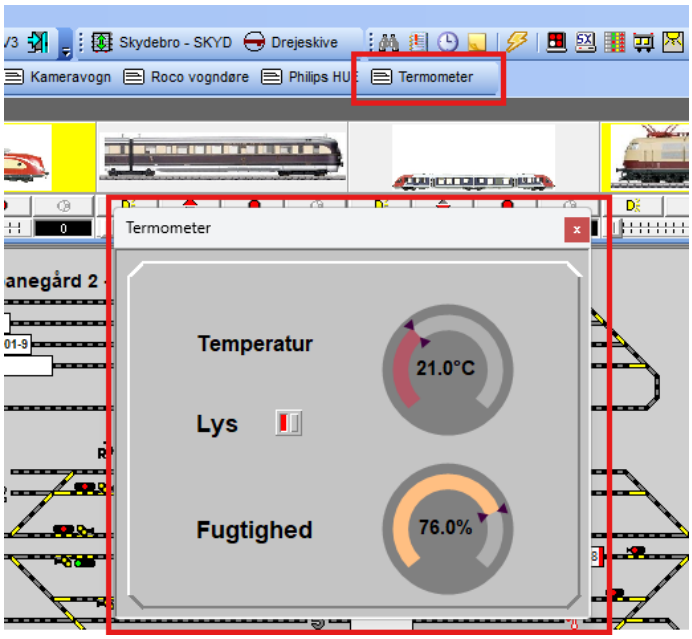
We use these two gauges:



Define Solenoid Temperature. Adress 110	

Define Solenoid Humidity. Adress 120	
Define Solenoid On/Off - LED Adress 130	

8. Windigipet 2025 – Test



We can now test that everything is working from Windigipet. We can compare the readings with what we see on the small OLED screen and in the MQTT browser. And we can test that the light can be turned on and off.

This is a very simple example that I've described here, but hopefully it provides a basic introduction to the topic. It will be exciting to follow the Windigipet Forum and see the creative ideas shared on the subject. I definitely plan to convert my small arrival signs with real-time times to use MQTT instead, now that I'm currently using the Windigipet Mobile protocol.