Server Startup Guide

Intro

A brief guide and pointer how to start up the server receiving MQTT messages from the conduit gateway, and starting the web server that displays the received data. Progress, current status, and future improvements will be discussed. This guide is

MQTT Server

To receive messages from the conduit gateway, two applications need to be running: a broker (hbmqtt) and a client (mqttClient.py). The broker acts as a server, and establishes connections to clients through TCP/IP. The client (mqttClient.py) connects with the broker, and listens for messages sent by other connected clients. The client handles the received message, prints the data to terminal, stores data for the webserver and stores the file to a text file.

Startup:

- Open cmd.exe
- Type in `hbmqtt`

The broker should now start, leave it running.

- Open a new cmd.exe
- Change directory by inserting `cd C:\Users\stiabo\mysite`
- Run the client by inserting `python mqttClient.py`

This is all that's need to start the MQTT server.

Future work/improvement:

Broker/Server (hbmqtt)

- The broker hbmqtt is put in a debug mode by default, and makes it difficult to read useful messages, like client connected etc. This can be changed by creating a config file and adding its path when executing from terminal. For more info, investigate the "hbmqtt documentation" link.

Client (mqttClient.py)

- Decode the code type to identify what tag the message is coming from.
- Add the topic/DEVeui to identify which TBR the data was received from, add it to text file and web server.
- Storage to text files is currently only separated by data type. Adding additional hierarchy and separating the files by for instance date, sensor tag, location etc. can be beneficial.

Useful links:

- A guild to help understanding MQTT: http://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment
- Github location of hbmqtt: https://github.com/beerfactory/hbmqtt
- Hbmqtt documentation: http://hbmqtt.readthedocs.io/en/latest/

Web Server

The web server is implemented using Django and Python. Information about Django is probably more accurate and in depth from their website. Whenever you make changes to the files and the server is running, you don't have to restart the server, the changes will be automatically applied.

Startup:

- Open a new cmd.exe
- Change directory by inserting `cd C:\Users\stiabo\mysite`
- Start the server by inserting `python manage.py runserver 129.241.10.159:8000`
- The demo can be viewed by typing in a browser: http://129.241.10.159:8000/demo
- The admin page is accessed from: http://129.241.10.159:8000/admin/ username is *admin* and password is *ttk8108-password* . Here you can view and edit received data, but is currently not easily readable, e.g. you have to enter each unnamed message individually to view the content of it.

The startup is dependent on the host's IP address. It's recommended to try to avoid changing the IP address, as you will be required to edit the Conduit Gateway. To change the IP on the server computer you need to do the following steps:

- Edit C:\Users\stiabo\mysite\mysite\settings.py. Change ALLOWED_HOSTS to include your new IP address
- In the Conduit Gateway, edit the lora-sample-app.js (vi /home/lora-sample-app.js) and change *var url* to your new IP address
- Proceed with the same steps in startup, only change the IP address

Mysite files

Mysite is the name of the folder where the website is built, currently located at C:\Users\stiabo\mysite. There is currently one web application there; demo. The demo currently logs and graphs all stored SNR and Temperature messages received. What each notable file in the mysite folder will be briefly undergone here:

../mysite/manage.py

The manage file is used to start the server and execute commands regarding the server. It shouldn't be needed to edit.

../mysite/mysite/settings.py

General setting regarding the entire website. Noticeable variables to edit is ALLOWED_HOST if you want to publish your website, and INSTALLED_APPS to add additional applications (similar to the demo app). Debug mode is currently on.

../mysite/mysite/urls.py

Edit this to add more urls to navigate to new applications.

../mysite/demo/models.py

This is where the stored data is defined. Use this to add different classes of data (for instance a fish position class?). To add new changes you need to run `python manage.py makemigrations` and then `python manage.py migrate`

../mysite/demo/views.py

This is where the data is processed and sent to an html script. Add different functions there to correspond to different urls. You need to work and edit both the html script and this python script to get the message parsing and format correct.

../mysite/demo/urls.py

Edit this to add more sub-urls under the demo directory that corresponds to different functions made in view.

../mysite/demo/templates/demo/home_test.html

This is one example of an html file, using google charts and JavaScript. Create new ones or edit for additional functions in views.py. The data sent from views.py is extracted using `var TEMP = {{series.temp|safe }};` and `var SNR = {{series.snr|safe }};` in the code.

Future work/improvements:

- Convert the timestamp to date-time. Can be converted in both the python script and the JavaScript, but transferring an int between the two languages is usually easier than a date-time object.
- Select different timeframes, and from different sensors
- Implement a graph displaying the fish position or movement over time.

Useful links:

- Django, documentation and tutorials/guides: https://www.djangoproject.com/
- Examples of using Google charts in html file: https://google-developers.appspot.com/chart/interactive/docs/