Networking for Software Developers

This is the final project of COMP216. It is a group effort that will replace your second test. This builds on the previous week’s labs, which may require some tweaking to assemble into a complete IoT solution.

# Before you start.

This is a large undertaking considering the time constraints and the number of components required. One person would not do able to and should not be doing all the work because of the volume and the scope. Because the time is not enough sufficient for one person to adequately complete all the prescribe tasks.

I suggest short (not more than 15 minutes) meeting frequent (at least 5 time per week), where designs are finalized, tasks are farmed out and reported back on. You can also examine the milestones and see if more effort is need to bring any lagging task up to the expected completeness.

Make use of your favourite collaboration platform whatapps™, discord™, slack™, signal™, telegram™, teams™ etc. and relegate e-mails to formal correspondence.

## Intellectual Property

Software ownership is intellectual property. It is critical to understand that you may not take credit for something that you do not own/create. If you use code or algorithm or ideas from somewhere, it is ethical and the right thing to acknowledge the owner of the intellectual property. For this course if this happens, then you are committing plagiarism and punishment can range from a reprimand to zero to expulsion.

## Technical Constraints

You will only use the libraries that were covered in classes. You will use Python version 3.6.5 or later. The only external frameworks/libraries allowable are requests, flask and mosquito. You may use the libraries in a standard python distribution with restraints.

You must use 90% of the code from your previous labs.

# Overview.

We will implement an end-to-end IoT solution that will satisfy your client needs (this is based on your previous labs). We will use the MQTT protocol for this implementation. There will be publisher clients that send data to a broker as well as subscriber clients that receive data from the broker. The broker sends data to the appropriate subscriber.

You will design and build both types of clients according to specifications. The diagram shows an overly simplified architecture of the intended system.

Broker

Subscriber

Publisher

Publisher

Publisher

Subscriber

Each component is described more fully below along with the weight contribution towards the final grade. Each component must be implemented be well designed class.

# Summary rubric

The weight includes design, implementation, and demonstration of that particular component.

No written documentation is required, only your python code files. Your source files should have only one class per file.

|  |  |
| --- | --- |
| Item | Marks(%) |
| Broker | 5 |
| Publisher | 40 |
| Subscriber | 45 |
| Quality of code | 5 |
| Video (demonstration) | 5 |
| Total | 100 |

# Detailed description

**5 marks**

## Broker

You will install the Eclipse mosquito broker and ensure that it is working as expected.

**40 marks**

## Publisher

The publisher will generate data to send to the broker at regular intervals. The data value must be random with a pattern (I know that this is a contradiction). Think of the value of a particular stock on the stock exchange or the outdoor temperature around your home.

This must be implemented as gui and must include an interface to change the various parameters of this publisher.

Corrupt data?

Mutate value

Get value from generator

Sleep

Send data?

Package value as dict

Connect to broker  
Send data  
Disconnect from broker

No

Yes

No

Yes

You will configure and run multiple publisher clients to simulate multiple devices.

### Publisher – Value generation

17 marks

This must be implemented in a class in a separate file. (Just import the filename without the py extension in the file where you want to use the logic). This must be based on previous labs. You must not limit the number of values generated.

The design should be such that it is easy to use and flexible enough to be able to give the data value in the required pattern.

### Publisher – Packaging the above values

8 marks

The above value must be tagged with at least a time stamp packet id and packaged as a json object before transmission. You decide what other features you need to encapsulate in your package.

### Publisher – Sending data to broker

10 marks

* You will send the above packaged data to the broker under an agreed topic. [See lab13 **publisher.py**]
* You must miss transmission with a frequency of about 1 in very 100 transmissions. This must not be deterministic!

### Publisher – Extras

These extras are for bonus points and will only be considered if all the normal specifications are satisfied adequately.

* To simulate a real-world scenario, occasionally skip blocks of transmissions (or sets of transmission). This condition must not throw the subscriber into confusion.
* Transmit “wild data” something that is completely off the chart. Again your subscriber should be able to handle this.
* Anything that will add value to your project. You must make me aware of these.

**45 marks**

## Subscriber

The subscriber accept data from the broker and process it. It will decode the data and decide how to process it.

This is best implemented as a GUI application.

You will configure and run multiple subscriber clients to simulate multiple devices.

### Subscriber – Receiving data from the broker

10 marks

* You will listen to messages from the broker under an agreed topic.
* You will decode the message and decide how to handle the data

### Subscriber – Handling the data/absence of data

30 marks

This section is important, because this is where you imbue your personality on this project. You have to decide what is out of range data. You also have to be able to detect missing transmission.

* Handle sensible data. Display data both in text and visual formats.
* Detecting and handling out of range (erroneous) data.
* Detecting and handling missing data.

## Quality of Code

**5 marks**

These 5 points are to lose. You automatically start with full points and as the instructor notices code aberration these 5 points will evaporate. Code aberration will include design flaws and implementation blunders.

## In person demo/presentation of Project

**5 marks**

You make a 5-10min video that will demonstrate each of the item in the rubric table. If an item is not clearly demonstrated, there might be a possibility that you might not get a point for that item.

You need not demonstrate quality of code.

#### Submission

1. You will bundle all your files (except the video) into a single zip file (not wrar or anything else). The name of the file will be group\_«your\_group\_number».zip e.g. group\_1.zip.
2. Your publisher code files must be called group\_«your\_group\_number»\_publisher.py e.g. group\_1\_publisher.py.
3. Your subscriber code files must be called group\_«your\_group\_number»\_subscriber.py e.g. group\_1\_subscriber.py.
4. All your other code files must be prefixed by you group number e.g. group\_1\_data\_generator.py.
5. Must be uploaded to course drobox.
6. The video file must be uploaded directly in the drop box following the same naming conventions. Make sure the video has MP4 extension or if you are uploading on YouTube please send a link for it.