# **Introduction**

In this lab you will implement the simple version the Task API from lab 4a adapted to use MQTT. Tasks will have an ID, state and description represented using JSON. For example,

|  |
| --- |
| {“id”: 1234, “state”: “open”, “description”: “example task”} |

Task states are governed by the same rules as in prior labs.

* New tasks are initially in OPEN state.
* OPEN tasks may be ASSIGNED or CANCELLED.
* Once someone begins work on an ASSIGNED task they set the status to PROGRESSING.
* A PROGRESSING task is either set to DONE once completed or is CANCELLED.

# **Part (a) Design decisions. Deadline Monday 30 November.**

1. There is no single “server” in a publish-subscribe system to store all tasks. Where should tasks be stored in a pub-sub system such as this?
2. Who should generate task IDs? How should they be generated to avoid collisions?
3. How can we represent the Task API ADD and DELETE operations in a pub-sub system like MQTT? Under what assumptions would the LIST operation be required?
4. When implementing the Task API operations on MQTT, comment on if, and how you would use the following MQTT features.
5. Quality of service (0, 1, 2).
6. Clean session flag on topic subscription.
7. Retained flag on message publication.
8. Two students are arguing about how to structure topics
9. Student a suggests OPERATION/ID/STATE
10. Student b suggests STATE/ID/OPERATION

Either choose one of these proposals or suggest your own scheme stating your reasons.

1. Student c suggests including description in the topic as well. Argue for or against this suggestion.

# **Part (b) Implementation: Deadline Friday 4 December.**

1. Implement the ADD and DELETE task operations.
2. Assuming exactly once delivery is available, implement the EDIT operation to change task state.