Overview

This document defines the specification for the embedded software/hardware project for CS4447. The marks awarded for this project will form 25% of the overall marks for the block

Summary

For the end-of-block project, you are asked to form teams and to build an Internet of Things (IoT) device that demonstrates a real-time software system running on embedded hardware.

Scoring

Projects will be scored on a team basis but where interviewers feel contributions were not made equally individuals within the team may score higher or lower than other members of the team. Projects will be scored over a number of categories as follow:

15%	20%	20%	30%	15%
Project Demonstration	Real-time implemention	Comms protocol and resilience	Innovation	Code structure and quality

Each category scored out of 5 points by each interviewer. The points score will be converted to a percentage before weighting and averaging. The points and their criteria are as follows:

5	100%	A1	Excelled at goal
4	76%	A2	Solid implementation that is largely right with reservations
3	62%	B2	Acceptable execution but flawed
2	48%	C3	Basic attempt but minimal demonstration of acceptable results
1	35%	D2	Some idea but nothing to show for it
0	15%	F	Didn't even try

Please note that ALL team members must be able to answer questions about ANY part of the code.

Requirements

- Students must group themselves into 12 teams with ideally 3 students per team.
- Teams of 4 are permitted but there will be a higher expectation of functionality.

- Your project must run on the FreeRTOS operating system. You may use either:
    - The ESP-IDF framework Running on the Firebeetle-ESP32 platform
    - The Maxim SDK framework running on the MAX32655 feather board from ADI with the ADI ISE I/O board
- The code must be compiled and should be written in C or C++ but may be written in Rust or another compiled language if you prefer.
- The system must demonstrate real-time data acquisition and control and must also communicate with a cloud back-end for data aggregation and to accept commands.
- The cloud back-end must run on the ISE debian cloud server (alderaan.software-engineering.ie - DNS TBD).
- An MQTT broker back-end (mosquitto) will be provided on the cloud server.
    - Data can be published to topics defined by the team
    - Commands can be accepted by subscribing to a topic defined by the team
    - Grafana, NodeRED or other tool can be used to visualise data and send commands.
- Alternatively, a back-end HTTP API may be created that accept data reports from the embedded device and the device should periodically poll for commands from the back-end (start, stop, set temperature, etc).
    - The HTTP API can use python, nodeJS or other
- The embedded software must demonstrate
    - Control / data acquisition functionality and the agent functionality must run in different threads running at different priorities.
    - A bi-directional queuing mechanism must be implemented to allow data to be transferred between threads.
    - The project must have a real-time control aspect. This is left unspecified and can be chosen by the student. It could be a simulated heating or air-conditioning control system, an LED or LCD display animation, a motor positioning control system, or any other application that demonstrates real-time control.
- An intermediate agent may be used (for example, a laptop) to connect to the embedded device over USB, WiFi, bluetooth or other protocol available to the agent platform.
- Students should invent a proprietary protocol for communication between the agent and the embedded device (simple binary message, JSON payload, etc).
- The real-time control functionality must always execute correctly whether or not the agent is attached. It is recommended that your IoT device is tolerant of agent disconnects and reconnects and can buffer some acquired data.

Deliverables and Submission

This assignment is for submission of three artifacts for the end of block combined project. The artifacts to be submitted before the deadline are:

1) zipped code for embedded (firebeetle or feather platform)

2) zipped code for cloud backend

3) powerpoint presentation for the interview

A collaborative source code repository (e.g. github) is strongly recommended and a link to the github should be included in the submission.

Code should be well-designed, modular, commented and readable.

Project interviews will take place on the last two days of the block. Each student will have 10 minutes to demonstrate their system, do a brief walkthrough of their code and answer questions about the design and implementation.

Extra hardware

The project focus should be on software deliverables and marks will given for software deliverables. However, external hardware can be used to add realism to the system. The Arduino zero to Hero kits can be used for components if required. These kits contain push buttons, temperature sensors, buzzers, sound sensors, ambient light sensors and other components that could be used to build a control system, as well as breadboards and jumper wires.

If any soldering is needed (e.g. Firebeetle header pins), please let me know and I will arrange.