ELEC143 Coursework P1: Environmental Control System

GROUP WORK AND PRESENTATION

SEMESTER 1: 2020

In the coursework, you are to work in a small team to develop a simple temperature and light monitoring system for a plant irrigation system. You are to write the code collectively, submit all your work online and demonstrate your system during a presentation at the end.

Task Outline

Plants grown in controlled conditions need to be monitored. This can be error prone and would benefit from a warning system. Conditions that need to be detected include:

- Light levels –some plants do react well to too much light
- Temperature – some plants will suffer when too hot or cold
- Pressure – if pressure consistently drops, this can be a predictor of rain

Your "module support board" includes the following sensor input devices:

- Temperature and pressure environmental sensors
- Light sensor

These are to be used to monitor the environmental conditions. It also has the following output devices:

- Buzzer
- LCD Screen
- Serial interface (connected to a PC)
- LEDs (traffic lights)

Your system should monitor and detect the following conditions:

Table 1. Software requirements and mark weighing for the technical achievements

SENSOR	CATEGORIES	ACTIONS	WEIGHT
LIGHT	DARK LOW DAY INTENSE	Detect when the category changes and report via the serial terminal. Proportionally adjust LCD backlight brightness with PWM[1]	25%
TEMPERATURE	FROST COLD WARM HOT	Periodically sound an alarm if near freezing and flash Red LED Red LED ON Yellow LED ON Green LED ON Detect when the category changes and report via the serial terminal. Use hysteresis.	25%
PRESSURE	FALLING	Display the pressure on the LCD screen Detect consistently falling pressure (to predict rain) If a rapid pressure fall is detected, write a message to the serial port and LCD	50%
ALL THE ABOVE	For higher marks, you should do the following:	For all the parameters above, avoid using single measurements. Instead, use arrays to calculate average values to reduce the impact of noise and other artefacts. For thresholds, use hysteresis to avoid jitter near category boundaries. Consider a sensible sampling rate. All these parameters change slowly. For testing, you might want to adjust this.	

[1] The darker the environment, the brighter the backlight. The LCD is visible in day light, so it should be off to save battery.

IMPORTANT NOTES

1. You are given a starter project that demonstrating how to read all the parameters above. You must modify this project and write your own solution.
2. For the different category boundaries, **you should use hysteresis** to stabilize the outputs. If you use hard thresholds, you can only achieve half-marks for the category.
3. For the different measurements, work with average values. For maximum marks, calculate a "moving average" (you might want to research the meaning of this).
4. You are required to test your code and present (i) your test methodology; (ii) your results; (iii) any problems you had. You may need to be creative about how you test your system.
5. When a category change is detected, it should only be reported once (not repeatedly).
6. For detecting falling pressure, research what is considered a significant fall in air pressure in advance of a low-pressure system. Consider writing functions and testing them separately with simulated data. You can also run some experiments of your own (by logging data over many days and keeping a record of when it rains). *This is a challenging requirement designed to push those who are ambitious.*

Assessment

There are essentially two aspects to the assessment: (A) coding and (B) scientific method and evidence.

A - SUBMIT YOUR SOFTWARE

Starting with the sample project (see the DLE), write your code to achieve as many of the requirements listed in Table 1 as you can.

You are to submit all your software project as a single zip file before the deadline.

1. Please ensure you delete the mbed-os and BUILD folders
2. Zip up your project folder
3. Submit online

TIP - Note this is the code you will use in your presentation. Every year, at least one group submits the wrong code and loses marks. To avoid this, get one member to submit the work and another to download the submitted work on a different computer, build and test.

B – DEMONSTRATE YOUR WORK AND EVIDENCE

To evidence the extent to which the requirements in Table 1 (over the page) have been met, you will need to give a presentation to the tutor(s) which should include a working demo of your system.

Arrange a time to present your work to the tutor(s). Your presentation should cover the following items:

1. Present a top-level flow-chart of your solution. This can be broken into smaller flow-charts if preferred.
2. Give a live demo during the presentation, showing which requirements you implemented and tested.
3. Present **how** you tested each of the requirements (this is known as **methodology**)
4. Present your test results of each criteria (known as **test results**). Be critically honest about the efficacy of your system.
5. Describe the challenges and problems you faced and what you would do to improve your work and how you might organize yourselves differently if you did this again (known as a **reflective account**)

Assessment Criteria

The overall project assessment criteria is shown below:

Table 2. Assessment Criteria

Assessment Criteria	
1. **Correctly drawn flow-chart**	5%
2. **Demonstration of technical achievements (of all criteria in Table 1).**	70%
3. **Test Methodology**	10%
4. **Presentation Test Results**	5%
5. **Reflective Account**	5%
6. **Code layout and commenting**	5%

For item 6., consider the following:

- Check code indentation is correct
- Comment all functions
- Comment at line level if the purpose is not self-evident
- Use meaningful names for variables, functions and constants
- Use preprocessor directive #define for any constants that occur more than once in your code
- Factor code into functions as appropriate, especially where it facilitates testing
- Split your code into multiple C or CPP files as appropriate
- Avoid unnecessary use of global variables or global functions
- Try and avoid deep nesting
- Consider polling / state-machine model (see lab notes)
- Use enumerated types to make code clearer

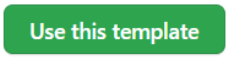
A key judgement the examiner will consider is this: “Can someone else read your code and understand it”.

Template Code

A project to get you started is provided on GitHub

https://github.com/UniversityOfPlymouth-Electronics/ELEC143-P1-2020

Click the link above and click the following button:



This allows you to create your own private repository. You MUST create a private repository.

You can then add your group as members so you can work collaboratively if you wish.