Assignment for Credit

Course Code : ENG 5220	Course Name : *Real Time Embedded Programming*
Type : Technical Report *Oral Presentation & Public relations*	Title of Assignment : Development, design, construction and promotion of a product requiring realtime operation
% of final course mark : *100%*	Lecturer : *Bernd Porr & Nick Bailey*

Marking

1. 10% will be awarded for the initial pitch of the project (every team has 5 mins and two slides) and initial github pages up and running (special timeslot, see moodle). We assess here the originality/usefulness of the work, if it has a solid realtime requirement and the quality of the presentation.

2. 25% of the credit for the final submitted work will be based on the way the code is structured, that it is divided up in classes allowing encapsulation of data, using data structures in a failsafe way, receiving data and releasing data in a safe way and in general guaranteeing high reliability and ease of maintenance.

3. 25% of the credit for the final submitted work will assess the realtime responsiveness of the software and how this has been achieved. This includes whether processing of events has been achieved by waking up threads and in general employing event driven code using timers, signals, threads and/or kernel space interrupt driven coding in preference to polling or other less suitable methods.

4. 25% of the credit for the final submitted work will be based on the use of revision control, committing, branching, creating releases, testing and project planning. Marks will be awarded for demonstrating clear division of labour and documentation of the work. Has git been used as a revision control system or just to ``upload'' code? Has git been used to do revisions, track bugs and has there been a release strategy? Has the issue tracker system been used? Have unit tests been used?

5. 15% of the marks are devoted to the promotion of the work: has the project been properly presented on github so that it catches the eye of a potential user? Is the hard/software described in a way that other people can reproduce it? Has the project been advertised on social media and has it been picked up by online publications such as hackaday? Has a social media account been created and has it created a buzz around it? Has the project a license?

All items above will marked on the 22 point scale, according to the performance indicators written overleaf. Consideration will be given to the inclusion of Aims and Objectives and clarity of presentation.

University of Glasgow | School of Engineering

Submission & Return

The submission is online via moodle where you submit the link to the github page which contains your report, code, hardware and links to social media. In addition every team should be submit a single page just containing the names of the team members and a link to their github to confirm that the submission is all your own work.

Make sure that each group member's area of responsibility is clearly marked.

Note that University policy on late submission of work without good cause is that the grade will be reduced by two secondary bands (e.g. from 'B1' to 'B3 or 'A5' to 'B2') for each working day, or part of a working day, after the submission deadline. Work submitted more than five days after the deadline will receive an 'H' grade. If you are unable to submit work on time due to good cause, you should contact me as soon as is possible to seek a deferral.

Submission deadline : 17 April 2023

Results & Feedback

Feedback & results about the initial pitch will be available after the presentation and will come from both lecturers.

You will receive feedback about your final work via email. This feedback will be structured according the 4 marking criteria above covering the final work and will comment on every section.

1. Presentation

Grade range **Aggregation Score**	**A1, A2** **22, 21**	**A3, A4, A5** **20–18**	**B1, B2, B3** **17–15**	**C1, C2, C3** **14–12**	**D1, D2, D3** **11–9**	**E1, E2, E3** **8–6**	**F, G, H** **5–0** **(maybe CR)**
Delivery	Could present at a conference with no further training	Confident delivery, clear speech, no hesitation, held attention	Good delivery, only minor flaws such as hesitation	Significant lapses in delivery but satisfactory overall	Hard to follow significant parts of the talk	Couldn't make out anything without difficulty	Impossible to learn anything
Slides	Of professional conference quality	Excellent slides, attractive appearance, information well presented	Good slides, only minor flaws such as poor layout or plots with illegible axes	Some slides had illegible text or incomprehensible illustrations	Poor slides, hard to read or deduce content	No effort made to prepare appropriate slides	No slides (consider CR)
Originality	A novel product idea with clear market appeal	Impressive idea which is genuinely novel	Idea appropriate to the brief	Indea generally satisfactory but not clear what is original here	Idea not clear and hard to judge	Generally inadequate or incorrect content	No worthwhile idea(consider CR)
Realtime	Professional, quantitative realtime assessment	Clear case for realtime processing	Satisfactory case for realtime processing. Mostly qualitative.	Realtime demands not completely clear.	Poor case for realtime procesing, lacking major aspects	Minimal understanding of realtime processing.	No understanding of realtime processing.
Response to questions	Supervisor learnt from response to questions	Confident and informed response to all questions	Good response to questions but occasionally unconvincing	Satisfactory response to most questions	Had difficulty answering most questions	Required prompting for any answer	Unable to answer any questions satisfactorily

2. Structure of the code

Grade range Aggregation Score	**A1, A2 22, 21**	**A3, A4, A5 20–18**	**B1, B2, B3 17–15**	**C1, C2, C3 14–12**	**D1, D2, D3 11–9**	**E1, E2, E3 8–6**	**F, G, H 5–0 (maybe CR)**
Optimal choice of classes (SOLID)	Classes have clear responsibilies, interfaces are segregated to be client specific, dependency inversion, obey the Liskov Substitution Principle and documented in an intutive way.	Classes have clear responsibilies, interfaces are segregated to be client specific, dependency inversion, obey the Liskov Substitution Principle. Minor issues but still professional production standard.	Generally following the SOLID principles but either one is violated or documentation does not demonstrate that they have been taken into consideration.	Some SOLID principles haven't been applied and/or there are violations of the principle. Documentation has flaws which makes it hard to see if/how they have been applied.	Serious flaws in the implementation of SOLID and most principles haven't been applied. There is little mention in the documentation about the class choices.	Not clear whether SOLID has been applied or not. Some aspects appear to be applied but there is no direct evidence or documentation which makes it clear.	No application of SOLID or little to mark at all.
Encapsulation of data in classes and safe use of getters and setters and data management.	Clear public interfaces are defined, the data is private and getters and setters provide a safe interface to the client. Internal data structures are efficient and getters/setters provide fast acccess / compuation.	Public interfaces are defined, the data is private and getters and setters provide a safe interface to the client. However, some minor flaws for example in terms of safety, timing and choice of internal data structures.	Generally data is encapsulated and the internal storage of data is appropriate but there smaller issues with the getters / setters, not checking for fault conditions or the internal data storage could be more efficient.	Significant problems with encapsulation such as public variables and no fault checking. Data storage/management is inefficient.	Serious flaws in encapsulation with public variables being accessed, no clear getter/setter interfaces and data is stored in not appropriate structures.	No encapsulation in the classes used but classes work by accessing variables and calling member functions. No use of public / private variables & members.	No classes used, use of global variables or class-less coding.
Failsafe memory management	Memory management is completely leak free.	Memory management is leak free but uses new/delete where it could be avoided.	Excessive use of new/delete where C++ instances and copy constructors could be used.	Clearly there is a lack of care of tidying up memory allocations.	Serious flaws of memory management with eventual crash.	Serious flaws in memory management leading to out of memory.	No memory management at all or nothing to mark.

3. Realtime responsiveness

Grade range **Aggregation Score**	**A1, A2** **22, 21**	**A3, A4, A5** **20–18**	**B1, B2, B3** **17–15**	**C1, C2, C3** **14–12**	**D1, D2, D3** **11–9**	**E1, E2, E3** **8–6**	**F, G, H** **5–0** **(maybe CR)**
Assessment of latencies in the application context and appropriate design decisions	Professional quantitative assessment and tolerances leading to clear coding decisions	Good quantitative assessment of the realtime demands leading to good coding decisions with small omissions.	Correct assessment of requirements but smaller shortcomings and resulting smaller issues in terms of coding decisions.	Assessment of the latencies partially wrong or not completely considered and the propose coding framework is not well thought through.	Latencies not seriously assessed and thus no justification of the realtime coding strategy.	Almost no effort to research in into latencies and their knock on effect on coding.	Achieved virtually nothing (consider CR)
Realtime coding	Production level realtime coding using threads/timers/signals and kernel interrupts	Perfectly working prototype but minor shortfalls in structure, doc or reliability.	Solid realtime coding but with smaller coding issues causing small noticeable latencies.	Realtime coding has shortcomings in responsiveness, timing and sampling of signals.	Significant shortcomings in the realtime coding resulting in long latencies.	Design shows major weaknesses in realtime processing utilising delays / blocking code..	Showed few or none of the skills expected of a graduate (consider CR)

4. Revision control and project management

Grade range **Aggregation Score**	**A1, A2** **22, 21**	**A3, A4, A5** **20–18**	**B1, B2, B3** **17–15**	**C1, C2, C3** **14–12**	**D1, D2, D3** **11–9**	**E1, E2, E3** **8–6**	**F, G, H** **5–0** **(maybe CR)**
Revision control	Professional use revision control with regular commits, branching & merging	Good use of revision control with detailed commits	Use of revision control but shortcomings in commits and development on master	Only work on master without any safeguards and shortcomings in commits	Only few commits on the master branch with generic comments.	Used github only as an upload site with no collaborative effort	Achieved virtually nothing (consider CR)
Project management	Exemplary; could not have done better with the time and resources available	High-quality planning, made excellent use of time and resources available	Good planning and use of resources with only minor deficiencies	Satisfactory planning but could clearly have made better use of resources.	Poor planning and use of resources; did not always follow directions	All over the place; required continual direction from supervisor	Did only what the supervisor told him or her, if tha
Reliability / Testing / Bug fixing	Professional testing approaches with unit tests, issue tracking, fixing	Good test scenarios which unit tests	Satisfactory testing and debugging but smaller shortcomings	Testing only in some cases but clearly some are left out.	Poor testing just in a qualitative manner,	No explicit testing but just report of success.	Achieved virtually nothing (consider CR)

5. Documentation and PR

Grade range Aggregation Score	**A1, A2 22, 21**	**A3, A4, A5 20–18**	**B1, B2, B3 17–15**	**C1, C2, C3 14–12**	**D1, D2, D3 11–9**	**E1, E2, E3 8–6**	**F, G, H 5–0 (maybe CR)**
Quality of the content	Professional level of documentation comparable to other github prof projects	Comprehensive coverage with no significant omissions	Good coverage with only minor omissions	Covered much of the project but with significant omissions	Major omissions; large parts of project not covered	Only a little material relevant to project	Nothing of substance (consider CR)
Quality of argument	Could stay on github without further work	Arguments well presented; results clear and accessible	Results critically assessed	Discussion of results lacks insight	Discussion of results at only a very low level	Discussion perfunctory	No discussion apparent
Illustrations and video content	Worthy of publication	Well-chosen, illuminating and attractively formatted illustrations and excellent video	Good illustrations that enhance the report and an eye catching video	Illustrations satisfactory but could be drawn or chosen better; too few illustrations. Video could have clearer message.	Poor illustrations or mostly from WWW. Video film has low quality in terms of narrative and presentation.	Images only from WWW. The video has a poor quality.	No illustrations (consider CR) No video.
PR / social media strategy / release strategy	Perfectly devised strategy on all channels and targeting the right audience	Well devised strategy covering all relevant channels and target audience.	PR strategy reflects a good amateur project but has shortcomings for a prof product	PR OK for a local group of friends and followers but has shortcomings reaching beyond it	Poor PR just involving a few last minute posts on social media. No clear strategy.	PR strategy just limited to github.	No PR (consider CR)

University of Glasgow | School of Engineering

§1 Task Overview

Aims

Development and promotion of a product requiring realtime operation.

Objectives

- Propose a product which requires realtime processing and solves a real world task
- Select hardware connecting to a Raspberry PI as proof of concept
- Develop realtime software in C++ (web-pages/mobile apps in scripting languages)
- Create, maintain, schedule and document the project using git version control, tests and quality management
- Promote the final product via github, social media and live demos

§2 Task Requirements

The task is to present an end user product which requires realtime processing. This will be build around Linux on a Raspberry PI. It needs to be a project which solves a real world problem, for example, watering plants while away on holiday or a mattress which senses if a person sleeps well.

In technical terms this means that the Linux system needs to measure physical values, plot them on the screen, allow mouse interaction to change parameters and that it generates meaningful outputs. All this in realtime. At the end you should have a standalone embedded application which boots up and performs your chosen task.

Your task is to use data acquisition hardware, for example the sound card or on the Raspberry PI sensor boards and digital sensors.

Main coding language must be C++. The operating system must be Linux. Code must be written in an object oriented fashion with a testing framework i.e. unit testing. Only web pages and mobile phone apps are permitted to be written in a scripting language (PHP, js, Python, JAVA, swift, ...).
The code must be event driven -- either in userspace with signals and/or waking up threads and/or interrupt driven in kernel space.

Form groups of four and every person should have distinct roles. On moodle is a wiki where every team enters their names, matric numbers and links to github where their entire project is hosted.

Whilst creative lateral thinking is always welcome in Masters level courses, it is possible to take shortcuts in creating an application which mean that it is no longer realtime, or is otherwise trivial in nature, and thus does not show mastery of the Intended Learning Outcomes of the course. We set out here requirements for the work, which if you ignore will

ensure that your project does not fulfil the brief and is liable to receive few if any marks. In particular the following criteria pose a strong risk that the group will receive zero marks:

- program goes into wait state and becomes unresponsive
- using wait statements to establish timing instead of switching threads or load balancing
- not using callbacks to process events
- trivial work selling just with public relations but no substance
- no indication of version control
- not using C++ as the main coding language (remember scripting is only allowed for web services and apps)

Do not hesitate to discuss with the course co-ordinator any original approaches to the assignment you are worried might be off-topic and thus could attract a very low grade.

§3 Formal contact hours and independent work

You'll spend 33 hours in the lab under supervision. There are also 11 hours of lectures you need to attend. In addition you'll need to work both independently in the lab and do independent study in the remaining 156 hours allocated to this class. This work requires a high degree of independent work while the lab sessions shall be used to get advice, guidance and feedback from both the academics and teaching assistants.

§3 Hardware purchases

The budget is £45 per team for orders via the electronics store and/or technician.