Your final project is to design and build a simple system that solves a problem you find interesting. The final project is intended to bring together the course topics in a form that can be shared with prospective employers.

The project was described in the Final Project video (see Week 1 Lecture - Final Project Overview. The slides have the essential information.

Keeping up with the course exercises will help you finish the project without undue effort at the end of the course. That said, students are expected to spend 10-15 hours on this project in addition to the exercises. That is not a lot of time so you'll have to be efficient and focused.

While the lectures have talked about grandiose ideas, please keep your system simple, explainable, and reuse code.

Minimum Project Requirements

The project must:

(a) Use a Cortex-M processor
(b) Have a button that causes an interrupt
(c) Use at least three peripherals such as ADC, DAC, PWM LED, Smart LED, LCD, sensor, BLE
(d) Have serial port output
(e) Implement an algorithmic piece that makes the system interesting
(f) Implement a state machine

You are not required to use a HAL but it is encouraged. In fact, write as little code as possible, reusing whatever you can, while keeping in mind licensing.

Cortex-M Processor

Any Cortex-M processor can be used. The STM32 and the Discovery boards were recommended but any Cortex-M is good. Students may use a processor/board simulator instead of physical hardware (please notify a mentor if you intend to use a simulator).

Button

The system should have a button that causes an interrupt. What the interrupt does is up to you. The goal here is to show a simple GPIO interrupt handler being used.

Peripherals

The system should interact with at least three peripherals. The peripherals suggested include ADC, DAC, PWM LED, Smart LED, LCD, SPI sensor, and BLE. However, any peripherals are

acceptable as long as they are not controlled by a simple GPIO. The peripherals may be internal to the microcontroller (like an on-board ADC) or external (SPI Flash, I2C sensor, PWM speaker). There should be three different peripherals used.

Serial Port

In addition to the peripherals, the system should have a serial port configured for debugging. This may be simple logging or it may include a command line interface.

Algorithm

The system should have an application, it should do something. Exactly what it does is up to you, hopefully it is something that interests you.

Many examples were given in the final project lecture (slides). You may use one of those or develop your own idea.

State Machine

The system should include a basic state machine with at least three states. This may be part of your algorithmic element. Also, using a command line on the serial port fulfills this requirement.

Deliverables

The final project will be delivered as:

(a) Video of the system working as intended (link to mp4 or youtube)
(b) Write up of the system (PDF or Google docs report).
(c) Link to the code

These will be submitted to assignment-submissions in Discord or emailed to Thomas Fuller (thomas.fuller@cohorts.classpert.com). Please submit links to all three deliverables as part of one message.

Bonus points are available by including one of these:

- An analysis of the power used in the system (expected vs actual)
- Implementation of firmware update with a description in the report of how it works
- A description of profiling the system to make it run faster

Additional bonus points are available by having a version control history showing the development process.

Video

A two to five minute video will be used to give an introduction and show the functionality of the system. This does not need to have high production values, it is intended as a simple check of the system. Talking about the system as you film is great. We will have a live session where you have the option to present your final project to the class. This recording can be used to fulfill the video requirement.

Report

The written part of the assignment is an introduction to the system for another programmer. The write up should include:

- Application description
- Hardware description
- Software description
    - Describe the code in general
    - Describe the parts you wrote in some detail (maybe 3-5 sentences per module)
    - Describe code you re-used from other sources, including the licenses for those
- Diagram(s) of the architecture
- Build instructions
    - How to build the system (including the toolchain(s))
        - Hardware
        - Software
    - How you debugged and tested the system
    - How you powered it (and how you might in the future)
- Future
    - What would be needed to get this project ready for production?
    - How would you extend this project to do something more? Are there other features you'd like? How would you go about adding them?
- Grading
    - Self assessment of the project: for each criteria, choose a score (1, 2, 3) and explain your reason for the score in 1-2 sentences.
    - Have you gone beyond the base requirements? How so?

Link to the Code

The code will be reviewed for clarity and that it meets the minimum project goals. Please make it clear which files are written or significantly modified by you.

Grading

The instructor and mentors will review the projects and critique them according to this rubric.

For each category, students with an assignment that truly exceeds expectations will be given the maximum score (3). A “Meets Expectations” will receive a 2, while a “Needs Improvement” would get 1.

The scale is flexible and an assignment may be between levels.

Students with more than half the points on the final project will pass the course.

Criteria	1 - Needs Improvement	2 - Meets Expectations	3 - Exceeds Expectations
Project meets minimum project goals	Any project goals are not met.	All project goals are met. The state machine may be basic.	Additional sensors, actuators. Well documented and implemented state machine. Comprehensive command line on serial port.
Completeness of deliverables	Lacks report, video or code. Report does not cover all sections listed. Code has obvious errors that would cause it not to compile.	Report covers all sections but some are answered incompletely leaving questions for the reader. Code is readable given the report as a description. Video shows code working.	Code is readable on its own, without the report. Report addresses each point thoroughly, demonstrating understanding as it relates to the course. Video demonstrates the project and is explanatory.
Clear intentions and working code	What the system is supposed to do (based on the report or code) doesn’t seem to be what the system does in the video.	The system performs approximately as described in the report and code.	The system performs as described in the report in a manner that is professionally polished. The code shows how it works in a way that is easy for a maintainer to see.

Reusing code	No code was used from other sources or it is unclear what code was used from other sources.	Student code was identified.	Versioning of reused code was included along with a license document that describes the license for the student's code and the reused code as well as shipping implications. Reader is confident they could rebuild the student's system.
Originality and scope of goals	The student did the bare minimum to meet the goals, showed no originality.	Some areas of interest were noted in the report but they were minor extensions of the existing examples.	The student has gone far beyond the requirements to make something novel and awesome.
Self-assessment (Mentor category only)	Self-assessment was significantly different from mentor assessment.	Self assessment was +/- 25 % of mentor assessment.	Self assessment was +/- 10% of mentor assessment.
Bonus: Power analysis, firmware update, or system profiling	None	Described	Described, has graphs, and is accurate
Bonus: Version control was used.	None or a single commit.		The log shows the project being built, though the messages may be terse but should be descriptive.