Introduction

We are coming to the end of our semester in PES. For your final project, I want to allow you to pursue an area of particular curiosity and express your creativity.

For the final project, therefore, you should define your own functionality. To give you an idea of what I'm looking for, I have some ideas below, but I encourage you to be original.

The only requirement is that you develop some code to run on the KL25Z in bare-metal mode or using FreeRTOS. Your project should be complex enough to demonstrate your mastery of the course material, but not so complex that it is difficult to achieve in the time you have. In general, I am looking for something on roughly the same order of complexity as our every-two-weeks assignments; this will probably take more time, however, because I expect you might have to do more research since you will be defining the project.

As with the previous assignments in this class, the final project is an individual project that should be completed independently by each student. You may, of course, work with each other to perform peer code reviews or otherwise support each other.

You may choose to add hardware to the Freedom board, if you have the ability to do so. You may also submit a software-only project. In either case, your grade will only be based on the software portion of your project.

As we have done throughout the project, you will develop a proper GitHub project for this assignment, complete with code, documentation, and README. If your project requires special hardware, please take steps to submit documentary evidence (videos, photos, and screenshots), since it is unlikely the graders will be able to duplicate your hardware environment.

Considerations for the Final Assignment

You might consider the following technologies for inclusion in your project:

- Circular buffers
- State machines
- I2C
- SPI
- UART
- Command Processing
- Data compression
- DMA
- Low power modes
- ADC
- DAC
- Analog comparator
- Write some ARM Assembly
- GPIO lines
- Interrupts
- Detailed configuration of the system clock
- Pulse width modulation

- A test framework such as uCUnit (see http://www.ucunit.org/) or cmocka (see https://cmocka.org/)
- FreeRTOS

Your project should use several of the technologies from the list above. Of course, you may use other technologies of interest, as well. For instance, if you are interested in motor control, by all means work that into your project. I want to challenge you: Impress me with your originality!

If you are looking for ideas, consider browsing the Sparkfun website (https://www.sparkfun.com/); again, you do not need to buy hardware, but if you do want to do so, you can normally pick it up within a day from Sparkfun if you are in the Boulder area.

Possible Project Ideas

Below, I have listed a handful of project ideas for you to consider. You are welcome to select from this list, or to use one of the ideas here as a launching-off point to put your own unique spin on something.

1. Use the DAC to play musical notes in response to commands typed in your active prompt. For instance, type "A300,G300,F300" to get 300 msec of an A, followed by 300 msec of a G, followed by 300 msec of an F. See https://pages.mtu.edu/~suits/notefreqs.html for a start at mapping notes to frequencies. To accomplish this, you would need to hook up the DAC's output on the FRDM-KL25Z to a means of hearing the sound. Once you have basic notes playing, add at least one DSP effect (e.g., echo, delay, reverb, …), controllable from the interactive prompt.

2. Tie into the MMA8451Q accelerometer over I2C, and implement the logic for a digital angle gauge (something akin to https://www.rockler.com/wixey-digital-angle-gauge-with-backlight without the LCD screen). Change the LED colors in response to the device orientation to allow a user to set the saw blade on a table saw to exactly 45° relative to the table. Note that in order to do this, you will need to provide some ability to calibrate the 0° setting – perhaps the user could press a button connected to GPIO to indicate 0°, or the user could type commands over the serial port.

3. Develop a UART implementation to implement profiling, including writing scripts on PC side to interpret program counters. There are two ways to do this, one of which does not require any additional hardware:

    a. Multiplex both debug printouts and profiling info over UART0. Send the profiling information in binary to compress it a bit. Add a header so that you can separate profiling from debug information in the code you write on the PC side.

    b. Light up a second UART; send debug printouts over UART0 as usual, and profiling data over the second UART

    In either case, write the script on the PC side to convert program counter addresses into function names.

4. Implement compression to speed debug printouts over UART. Down in the UART code, prior to transmitting a particular string over the serial line, use Huffman coding (see https://en.wikipedia.org/wiki/Huffman_coding) to compress the debug message. On the PC side, write the complementary code to decompress and display the debug printouts.

5. Build a USB-controllable heating pad for warmth in the coming months. Cut open up an inexpensive heating pad[1] and figure out how to connect it to your Freedom board. Then implement a thermistor connected to the ADC, or an I2C temperature sensor, and develop a control loop to drive the heating pad to a value defined by the user.

Testing

You should consider carefully how you will prove the correctness of your implementation. What is your testing strategy? Consider how you will prove correctness both for the “happy case” (when there are correct inputs and no errors) and for any error cases and corner cases. You should develop automated tests if at all possible; for areas where test automation is not feasible, you should write up a step-by-step manual test outline, and you should plan to submit both the outline and the test results with your final submission.

Project Approval

In order to ensure you are on the right track, please develop a brief proposal for what you intend to do with your project, and submit it to me via Canvas by **Tuesday, November 16 at 5:00 pm.** (Either PDF or Microsoft Word format is fine.) This proposal should answer the following questions:

- What functionality will your project demonstrate?
- What technologies will you use? For those areas we have covered in class, how will you demonstrate deeper knowledge than what we covered in the biweekly homework assignments?
- What do you anticipate needing to learn in order to develop your project? What sources (KL25Z Reference Manual, Internet sites, etc.) do you plan to use to figure out how to do whatever it is you are attempting?
- Does your project require any additional hardware? If so, what will you acquire, and what is your plan for assembly? (Again, my focus is on the software you develop. I am asking about your hardware plans just so I can ensure that whatever you are planning in this area is relatively straightforward.)
- Finally, what is your testing strategy for your project? Will you develop automated tests, will you use manual tests, or will you use a mixture of both?

[1] For illustration only, something like https://www.amazon.com/Sunbeam-Electric-UltraHeat-Technology-Machine-Washable/dp/B00006IV4N/.

Final Project Submission

Your project must be submitted via GitHub and Canvas by **Monday, December 13, at 11:59 pm.**

Grading

Points will be awarded as follows:

Points	Item
10	Did you submit the project proposal on time (5 pm on Nov. 16)? Was it clearly written, and did it address the questions outlined above?
2	For the final project submission, did you include the GitHub URL in your Canvas submission, and did you include appropriate documentation (video, photos, screenshots) if needed?
28	Overall elegance: Is your code professional, well documented, easy to follow, efficient, and elegant? Is each module located in its own file, with an appropriate .h file to expose the interface?
20	Does the project as submitted match the functionality outlined in your November 16 proposal?
20	Does the project show your mastery of embedded software development?
20	Testing: How did you prove that your code is correct? Consider automated tests, writing a test plan, or other possible approaches. Did you cover the "happy case" functionality? What about error cases and corner cases?

Good luck and happy coding!