Zephyr: Threads & Kernel Events Lab

BME554L - Spring 2026 - Palmeri

Dr. Mark Palmeri, M.D., Ph.D.

2025-09-29

Table of contents

Git Version Control

- Use best practices for version control (branching, commit messages, etc.).
- Do all development on a dedicated branch that is merged into `main` once it is functional.
- Commits should be very specific to the changes/additions you are making to your code. This will help you and others understand what you did and why you did it.

- On a given development branch, try to implement one small piece of functionality at a time, commit it, and then move on to the next piece of functionality.

> **!** Important
>
> You do not want one, monolithic git commit right before you submit your project.

Refactor: Heartbeat LED (25% Duty Cycle)

You will be refactoring your code from the timers lab to use a thread to control the heartbeat LED. Do all of the refactoring below on a new branch called `heartbeat_refactor`.

- Refactor your code to use a thread instead of a kernel timer to control the heartbeat LED.
- Instead of having your heartbeat blink at 1 Hz with a 50% duty cycle, change it to have a 25% duty cycle (i.e., on for 250 ms, off for 750 ms).

Test the Accuracy of the Heartbeat LED Timing

- Create a new Jupyter notebook for this lab's technical report called `testing/thread_testing.ipynb` that analyzes the accuracy of your 25% duty cycle heartbeat LED.
- Perform these measurements using the oscilloscope, and save all raw data in dedicated CSV files.
- Choose what to measure and how to analyze the data to verify that the heartbeat LED thread is working as expected.
- Discuss the results of your measurements and how they compare to the nominal specifications for the heartbeat LED.

Merge-n-Tag this Refactored Heartbeat LED Code and Analysis

> **Tip**
>
> Make sure that your Jupyter notebook reads in data from relative paths for the repository, **not** absolute paths on your local machine.

- Push your `heartbeat_refactor` branch to GitLab.
- Create a new Merge Request to merge `heartbeat_refactor` into your forked repository's `main` branch.
- Merge your `heartbeat_refactor` branch into your `main` branch, and create an annotated tag for that merged commit called `v1.2.0`.

Refactor: Kernel Events

- Do the following refactoring on a new branch called `kernel_events_refactor`, branched after the `v1.2.0` commit.
- Refactor all of your code to remove Boolean variables being used in your ISR callback functions to indicate that buttons have been pressed to now use kernel events in a single bit array called `button_events`.
- You should use 1-bit for each button, meaning the bit array is 4-bits.
- Implement a `k_event_wait()` in the appropriate state(s) of your main thread to wait for button events to occur. This will be a blocking wait, meaning the `main` thread `while` loop will not be running while waiting for button events.

Verify that your LED Timing Still Works

- Create a new Jupyter notebook called `testing/kernel_events_validation.ipynb` to verify that your refactored code still works as expected.
- It is up to you to now determine how to verify that your LED timing is still working with regard to what data to collect, how to analyze it, and how to present it in your notebook.

Merge-n-Tag this Refactored Kernel Events Code and Analysis

- Create a Merge Request to merge `kernel_events_refactor` into your `main` branch.
- Merge your `kernel_events_refactor` branch into your `main` branch.
- Create an annotated tag for that merge commit called `v1.3.0`.
- Create an Issue assigned to Dr. Palmeri to review your code for both `v1.2.0` and `v1.3.0`.

How do I ask for help?

1. If you have a general / non-coding question, you should ask your TAs / Dr. Palmeri on Ed to allow any of them to respond in a timely manner.
2. Push you code to your GitLab repository, ideally with your active development on a non-`main` branch.
3. Create an Issue in your repository.
    - Add as much detail as possible as to your problem, and add links to specific lines / section of code when possible.

- Assign the label “Bug” or “Question”, as appropriate.
- Be sure to specify what branch you are working on.
- Assign the Issue to one of the TAs.
- If your TA cannot solve your Issue, they can escalate the Issue to Dr. Palmeri.

4. You will get a response to your Issue, and maybe a new branch of code will be pushed to help you with some example syntax that you can use `git diff` to visualize.