Pulse Width Modulation (PWM) Lab

BME554L - Fall 2025 - Palmeri

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

2025-11-03

Table of contents

Learning Objectives

- Implement a steady-state PWM output to control "intensity" over a finite dynamic range.
- Implement a time-series-modulated PWM output.

Git Best Practices

- 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.

Firmware Expectations

- All firmware should be written using the **State Machine Framework**.
- Choose your states for each part as a firmware engineer would, using what you have learned so far this semester.
- Timers, work queues, callbacks, and interrupts should be used as appropriate.
- All good coding practices developed this semester should be followed.
- Use logging to display state information and other relevant information, warnings, and errors. Debugging log messages can remain in the code, but the logging level should be submitted at the `INF` level.
- Include a state diagram in your repository (`state_diagram.png`) using UML (`state_diagram.puml`) or some equivalent.

> **Tip**
>
> Do all development for this lab on section-specific development branches.

Steady-State PWM Output

Do this section on a development branch called `pwm-steady-state`.

Firmware Functional Specifications

- Using the ADC lab functionality to read a DC voltage on `AIN0`, map this read DC voltage to scale the maximum brightness of `LED2` using a PWM output.

> **Example**
>
> - If `AIN0` = 0 V, then `LED2` should be off (0% duty cycle).

- If `AIN0` = 1.5 V, then `LED2` should be at 50% brightness (50% duty cycle).
- If `AIN0` = 3 V, then `LED2` should be at maximum brightness (100% duty cycle).

- Update your state diagram to include the new functionality.

Testing

- Quantify the linearity of the maximum brightness of `LED2` as a function of `AIN0` voltage ranging from 0-3 V.
- Present your data and analysis in the technical report Jupyter notebook called `testing/testing_pwm.ipynb`.

> **Do Things Looks Inverted?**
>
> Remember that LEDs on the DK are `ACTIVE_LOW`.

Commit-n-Merge Steady-State PWM

- Merge your completed `pwm-steady-state` branch into your `main` branch using a Merge Request on Gitlab.
- Create an annotated tag of your `main` branch with all part of this lab merged in called `v3.0.0`.

Sinusoidal Modulation of PWM Output

> **Tip**
>
> Do this section on a development branch called `pwm-sinusoid`, branched off of `main` after `pwm-steady-state` was merged.

Firmware Functional Specifications

- Modulate the brighness of `LED3` to match sampling a 2-second, 10 Hz differential sinusoidal voltage on `AIN1` and `AIN2` after pressing `BUTTON2`.

- `LED3` should have its brightness modulated with as little latency as possible with respect to the input voltage. In other words, the brightness of `LED3` should follow the input sinusoid as closely as possible in time.

> **Important**
>
> The blocking nature of a syncronous buffered acquisition scheme is not ameable to real-time modulation of the PWM output. Consider using the aynchronous ADC sampling approach with a callback to update the PWM duty cycle after each sampling event or a timer-based, single-sample acqusition approach.

> **Tip**
>
> - Be cognizant of your sampling rate, PWM on-time period update rate and the PWM clock period relative to your input signal.
> - There are several ways to implement the 2-second sampling period, including using a monostable timer or counting the number of samples taken at a known sampling rate. Make a purposeful decision in how you implement this functionality.

- Set the minumum and maximum sinusoidal brightnesses to be PWM duty cycles of 0 and 100%, respectively.
- Update your state diagram to include the new functionality.

Testing

- Using the oscilloscope, concurrently measure your input sinusoidal signal (what is differentially input to `AIN1` and `AIN2`) and the output PWM signal on `LED3`.

> **Tip**
>
> You will need to low pass filter your modulated PWM output to see a smooth sinusoidal output. You can do this using the LPF Math mode on the oscilloscope with an appropriate cutoff frequency (relative to the frequency of the input sinusoid) and sampling window (capturing multiple cycles of the sinusoid). You can also build a simple RC low pass filter circuit to connect to the oscilloscope probe.

- Save this acquired oscilloscope data to a CSV file using a USB memory stick.
- In your Jupyter notebook, in a new section, plot the input and output signals.
- Calculate the frequency of your PWM signal and the phase difference between the PWM sinusoid and your input signal.

- Present your data and analysis in your Jupyter notebook.
- Discuss the accuracy of your PWM frequency and the latency (phase lag) of your system and how you could improve it in the future.

Commit-n-Merge Modulated PWM

- Merge your completed `pwm-sinusoid` branch into your `main` branch using a Merge Request on Gitlab.
- Create an annotated tag of your `main` branch with all part of this lab merged in called `v3.1.0`.
- Create an Issue for your TA to review this PWM lab, assigning it the `Review` label.

How to 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.