Lab 4: Pulse Width Modulation and Servo Motor Control

Goals

1. Understand the concept of Timer Output PWM mode
2. Learn how to configure and start a timer
3. Use PWM to control the LED brightness
4. Use PWM to control a servo motor

Pre-Lab Assignment

1. Watch Youtube Tutorials
    - Timers in PWM mode https://youtu.be/qAZjdx71ePc (37 min)
    - Timer PWM output https://youtu.be/zkrVHIcLGww (17 min)

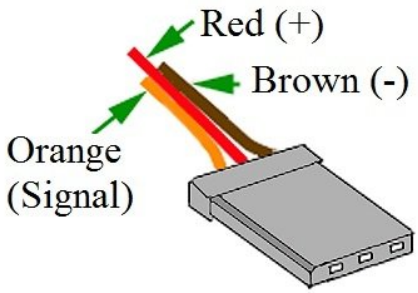
Lab Demo

1. Use PWM signals to periodically dim an LED
2. Use a 50-Hz PWM signal to control a servo motor

Background: Servo Motor

The key difference between a servo motor and a stepper motor is that the servo motor is controlled by a closed-loop system that uses position feedback to perform self-adjusting. A servo motor has an internal position sensor, which continuously provides position feedback. However, a stepper motor typically is an open-loop system which no built-in position sensor.

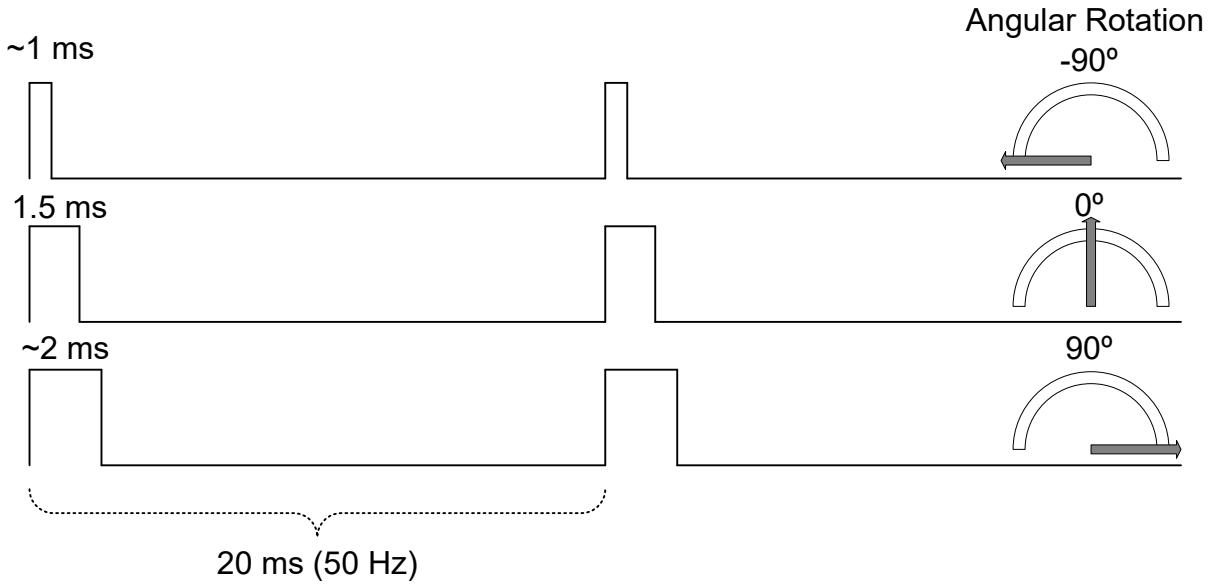
A servo motor is preferred in applications that require higher speed. A stepper motor is often used in applications that need higher holding torque.



In this lab, we will use FS90 servo motor. It can rotate approximately 180 degrees, with 90 degrees in each direction. The operating speed is 0.12sec/60 degrees (4.8V, no load). The motor has three wires, red wire for +5V, brown for ground, and orange for the PWM control signal. The position is controlled by the pulse width. As shown below,

- a pulse of 1.5 *ms* makes the motor return to the middle (0 degree)
- a pulse of ~1 *ms* makes it rotate all the way to the left (-90 degrees)

- a pulse of ~2 *ms* makes it rotate all the way to the right (90 degree)



Part A: Get an LED controlled by PWM output

Light an LED with 75% brightness using PWM output timer mode.

Part B: Make an LED pulse

1. Now put an infinite loop in your main.c function that makes the LED PWM duty cycle change from 20% to 100% and back (causing it to pulse).
2. You do this by changing the **`TIMx->CCR1`** value with a short delay between each update.

Part C: Get a Servo Motor controlled by TIMx

1. Modify your code so it can control the servo motor. Be sure you preserve the LED pulsing code so you can demonstrate it.
2. Modify your TIMx initialization code so the **ARR** period happens at 50Hz (20ms)
3. Calculate the values you need in the **CCR1** register to end up with a duty cycle of 1ms, 1.5ms, and 2ms. Create a function you can call that sets the **CCR1** register in this way so you can rotate to 90, 0, and -90 degrees.
4. You will demo the pattern of 0 degrees, wait 1 second, move to 90 degrees, wait 1 second, move to -90 degrees, wait 1 second, then move to 0 degrees.

5. You may find that for your servo 1ms / 2ms might not be the exact right values to rotate 90 degrees. Adjust the values until you get a good 90-degree rotation and document the values you used in the Lab demo section.

Part D: Submission

The grading for this challenge will mainly be based on your documentation. The reader should be able to verify the design process and the design choices based on objective arguments, test results, measurements, etc. Use pictures and diagrams to clarify your setup / measurements and design. Fully functional code without the required documentation has not much value.

Your submission is a zip file that consists of:

1. All code
2. Proper documentation consisting of:
   2.1. Title page with date and name
   2.2. Short introduction
   2.3. Your research: Show all relevant research steps and elaborate on the choices made
   2.4. Your design: all diagrams with short description and well justified design choices
   2.5. How you tested your implementation including proof (how you validated the results and so on)
   2.6. Reflection on what you have learned
   2.7. Bibliography (sources) in APA style (https://www.bibme.org/citation-guide/apa/)
3. A short video (max 4 min) where you demonstrate and clearly explain your solution