ME218B

Laboratory Assignment Number 7 for ME 218b

Pre-Lab Due by 11:59 pm on January 28, 2026
Due by 5:00 pm on January 30, 2024

Goals:

In this lab you will explore the behavior of a DC motor being driven by Pulse Width Modulation (PWM) using a Drive-Brake mode. You will also gain experience implementing closed loop speed control of the DC motor.

Submitting your Report

All lab reports will be submitted electronically. Your report should be a single PDF file, based on the Word document template file provided on Canvas in the folder associated with this lab assignment. Any requested schematics, or graphs should be embedded into the document. This template file includes sections for everything that you need to submit for the report. Please be sure that your answers are in the correct sections of the template document. When calculations are requested, the calculations should be included as scans of your neatly handwritten calculations (use a physical scanner, or smartphone scanner app. Please do not submit a jpeg file, they can be difficult to read.). When including required schematics in your reports, be sure to use the procedure outlined in the Lab 0 assignment section Supplemental KiCad Info to copy from KiCad and paste into Word. Limit the size of the area being transferred so that the resulting images fit on the page in the report. When the report document is complete, print it to a PDF file, in color.

When copying your code into your report, please use the Highlight program to prepare an RTF file using the "bright" color theme. You can then open the RTF file in Word and copy and paste the code into your report and it will maintain the color highlighting. Look at the code that you paste into your report. If the indenting is messed up anywhere, it means that you had tab characters in your source files. If the indenting is messed up, stop, go back and fix it before running Highlight again and submitting your report.

You submit your report by uploading the report PDF file to Gradescope and identifying in your document where the answers to the various sections are located. In order to get your reports graded and the feedback back to you as soon as possible, it is essential (read that as required) that you use the template document as the starting point for your report and correctly associate the sections of the assignment in Gradescope to the correct pages in your report document. I have instructed the TAs not to search for your answers if they are not in the correct portion of the template document or if you have not properly associated the pages to the sections in the Gradescope assignment. If you submit your report after the due date, please inform us.

When labs require you to write code, the plain text .c files will be uploaded to Gradescope separately from the report so that they can be processed by MOSS (Measure Of Software Similarity) to insure that each of you has written the code for yourselves.

Part 0: Pre-Lab

Background:

The pre-lab should be completed after you have read through the entire lab assignment, but before you go into the lab to begin your work there. Completing the pre-lab, which can be done without any special tools or resources, will allow you to be most efficient with your time in the lab. We judge the pre-lab submission based on a reasonable effort in pre-lab document submitted.

Assignment:

- ☐ 0.1) Design a circuit to drive the DC motor bi-directionally using Drive-Brake mode using an L293NE (or SN754410) as the drive stage with a motor voltage of 5V. The schematic should be created in KiCad.
- ☐ 0.2) Figure out how you will measure the electrical time constant of the motor as requested in Part 1.1. Include a circuit diagram of the measurement circuit.
- ☐ 0.3) Design the software that you will use for Part 1. Include pseudo code for program(s).
- ☐ 0.4) Design the software that you will use for Part 2. Include pseudo code for program(s).

In the report:

Include your answers to all of the sections of the pre-lab.

Part 1: Driving a DC Motor Using PWM

Reading:

Sections 13, 15 & 16 in PIC32 Data Sheet (Canvas), Section 14 in PIC32 FRM Timers (skip 14.5) (Canvas), Section 16.3.3 in PIC32 FRM Output Compare(Canvas), COK: Chapter 8, section 8.3 & Chapter 23

Components Required:

PIC32MX170F256B, bench-top Power Board, 1 ea. L293NE (or SN754410), 4 ea. 1N4935/6 diodes, 1 ea. 10K potentiometer and assorted other components of your choosing. See a TA for any parts you need that are not in your parts kit.

Assignment:

You are to design the necessary circuitry and software to connect the PIC32 to drive the supplied DC motor using the L293NE (or SN754410) as the power stage implementing the Drive-Brake drive mode and PWM using the hardware PWM capability of the PIC32. Do not use the PWM library. The speed at which you run the motor should be determined by the setting of an external potentiometer which your software reads using the A/D converter in the PIC32. The direction in which your motor spins should be set by reading the state of a digital input pin. Your program should periodically (10 Hz) read the voltage set by the potentiometer and set the motor speed directly proportional to the voltage, as well as read the state of the direction input pin, setting the motor direction correspondingly. Your code should use an Input Capture (Input Edge-Time mode) interrupt response routine to measure the period of the encoder signal and pass it to a non-interrupt driven routine that will periodically (not at the same 10Hz rate) report the speed of the motor (in RPM) to the terminal window.

❑ 1.1) To get the best performance out of the motor, it is useful to know the electrical time constant. For minimum current ripple, you will want to set the PWM frequency to at least a frequency of one over the electrical time constant. Implement the necessary software to help you to measure the electrical time constant of the motor on the lab bench. What is the time constant?

❑ 1.2) Implement your DC motor driver design from Part 0.1 and Event Driven software design to read the A/D converter connected to the potentiometer and adjust the speed. Hook it up to the motor and get the motor to run with the speed controlled by the potentiometer and the direction set by the digital input pin. Set the PWM frequency based on the time constant that you determined in part 1.1

❑ 1.3) Map the Duty Cycle vs RPM transfer function for the motor/driver combination. Step through drive duty cycles from 0 to 100% in 10% increments and record the speed of the motor at each duty cycle. Be sure to wait until the motor speed stabilizes before recording the speed. For regions where there was a large change in speed between two adjacent 10% duty cycle points, go back and add two more duty cycle points in between them. Create a graph to visualize the data.

❑ 1.4) Demonstrate your working software to a coach/TA/Ed and get signed off on the check-out sheet.

In the report:

Include

- The schematic diagram of the circuit you used, design calculations confirming the suitability of all components used. Be sure to indicate on the schematic which bits of which ports of the PIC32 you used, and please include pin numbers.
- A description of your measurement technique and results from your time constant measurement in Part 1.1,
- A graph of the DC vs. Speed data from Part 1.3.
- A Highlighted version of the source code to the program.
- Be sure to be signed-off by a coach or TA.

This quarter, we will be using Gradescope for your code submissions as well. Look for a Gradescope assignment entitled Lab 7 Code and submit the raw .C files (not Highlighted). Note, you will complete 1 code submission (including the files for all parts) when you submit, so save this step to the last. Please make sure that your files are named such that we can tell which files go with which parts of the lab.

Time Spent:

Preparing outside of the lab_____

In the lab working this part_____

Part 2: Implementing Closed Loop Speed Control

Reading:

COK: Chapter 28, Canvas: PID Without a Ph.D., The Control Loop; Sections 13, 15 & 16 in PIC32 Data Sheet (Canvas), Section 14 in PIC32 FRM Timers (skip 14.5) (Canvas), Section 16.3.3 in PIC32 FRM Output Compare(Canvas).

Components Required:

PIC32MX170F256B, bench-top Power Board, 1 ea. L293NE (or SN754410), 4 ea. 1N4935/6 diodes, 1 ea. 10K potentiometer and assorted other components of your choosing. See a TA for any parts you need that are not in your parts kit.

Assignment:

In this part of the lab, you will build on your code from Part 1 to implement closed loop speed control on the DC motor. **Bi-directional closed-loop control is *not* required.** Get your closed loop speed control working with the motor spinning in only one direction.

- ☐ 2.1) Using your code from Part 1 as a base, add a periodic interrupt response routine set to fire at 2 ms intervals. In the interrupt response routine, implement a minimal proportional plus integral control law with anti-windup to calculate and program a new duty cycle every 2 ms. Be sure that you structure your code so that it does not cause you to lose an input capture interrupt (hint: make the control law interrupt a lower priority than the encoder interrupt). For calculating your control law, it is OK to use floating-point variables.
- ☐ 2.2) Tune the gains on your system to provide a quick and stable startup and maintain speed in the presence of an external disturbance (dragging a finger on the wheel attached to the motor). See Appendix B for suggestions on how to gather data to implement Ziegler-Nichols tuning.
- ☐ 2.3) Add code to your program that will take the A/D converter input as a speed command.
- ☐ 2.4) Explore the ability of the system to maintain a constant speed in the face of disturbances across the speed range.
- ☐ 2.5) Add a few instructions to your code from Part 2.4 to raise an I/O line immediately upon entry to the ISR and lower it immediately before exiting. How long did it spend in the ISR? What fraction of the execution time does that represent?
- ☐ 2.6) Demonstrate the working system to a TA/Coach/Ed. They will want to see the ability to vary the commanded speed and to maintain that speed in the presence of a disturbance.

In the report:

Include a Highlighted version of the source code to the program that you wrote for Part 2.1 and the measurements from 2.5. Be sure to be signed-off by a coach or TA. This quarter, we will be using Gradescope for your code submissions as well. Look for a Gradescope assignment entitled Lab 7 Code and submit the raw .C files (not Highlighted). Note, you will complete 1 code submission (including the files for all parts) when you submit, so save this step to the last. Please make sure that your files are named such that we can tell which files go with which parts of the lab.

Time Spent:

Preparing outside of the lab_____
In the lab working this part_____

Time Summary

Please add the time that you spent preparing the report to the times that you logged in each section and enter the data from the table into the Time Summary for Lab 7 assignment on Canvas.

While this information is being gathered solely to produce statistical information to help improve the lab assignments, it is a required part of the lab assignment.

Appendix A

Motor Stand Connections

Stepper Ph B	1	20	NC	
Stepper Ph B	2	19	NC	
Stepper Ph A	3	18	NC	
Stepper Ph A	4	17	NC	Ribbon Cable
DC	5	16	NC	
DC	6	15	NC	
Encoder A	7	14	NC	
Encoder B	8	13	NC	
Encoder +5	9	12	NC	
Encoder Gnd	10	11	NC	

Encoder gives 512 pulses per revolution on each channel 90 degrees out of phase from one another. There is a 5.9:1 gearbox between the motor and the output wheel.

For the small motors, the encoder gives 3 pulses per revolution on each channel, 90 degrees out of phase from one another. The small motor has a 298:1 gearbox whose output drives the wheel.

Appendix B

Gathering the Data to Apply Ziegler-Nichols

In order to apply the Ziegler-Nichols tuning method, you need to be able to generate the necessary data. That is, a time history of the motor speed in response to a step function change in duty cycle. There are at least two different methods that you can use to gather this data.

Method #1, The Pseudo-DAC

With this method, you will set up what I call a Pseudo-DAC (pseudo- Digital to Analog converter). To achieve an analog output proportional to duty cycle, you set up a PWM channel running at a high frequency, then low pass filter the PWM signal (with a corner frequency at least ten times lower than the PWM frequency) to generate an analog voltage that is proportional to the duty cycle pf the PWM waveform. In your code, you can use a scaled version of the motor speed to set the duty cycle output. The filtered PWM signal will be an analog voltage proprtional to the motor speed. You can then connect this analog voltage to the oscilloscope to capture the transient changes in speed when you make a step function change in duty cycle. On the oscilloscope you will use single trace mode to capture the transient, then save the data to a USB thumb-drive (see pg 81 of the User Manual for Siglent SDS1052DL+ on the Canvas, Data Sheets page). This will work best if you dedicate a single digital output pin to acting as the trigger for the oscilloscope. You connect this digital line to the second channel on the oscilloscope and use it as the trigger for the single trace capture. In your code, when you make the step function change in duty cycle, you raise the digital output line providing the triggr to the oscilloscope.

Method #2, Capturing the data on the PIC32

In this method, you will log the necessary data into a RAM buffer on the PIC32 and then, after the buffer is full, print the time:speed pairs out to TeraTerm. TeraTerm has the ability to capture all incoming characters to a file (see File:Log…). I would suggest that you create an interrupt that fires every mill-second. In the ISR, log to a statically allocated array the most recently measured period (a single 16 bit value) and increment the array index. When the buffer has filled, post an event to a service that has access to the RAM buffer and in that service use the logged period to calculate the current RPM and write out a comma seperated pair of the array index,RPM to the terminal. Since the interrupt is firing at 1ms intervals, the array index is also the time. You should kick off the interrupt at the same time that you make the step function change in duty cycle.

Once the data has been captured (using either method) you can use Excel to plot the data for analysis. You will want to experiment with different duty cycle step sizes to see which step size gives you a time history that most closely matches what the Ziegler-Nichols algorithm is expecting.