ECE 445L Lab 10

DC Motor Control

This laboratory assignment accompanies the book, [*Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154*](https://www.amazon.com/Embedded-Systems-Real-Time-Interfacing-Microcontrollers/dp/1463590156), by Jonathan W. Valvano, copyright © 2021.

# Table of Contents

[Table of Contents 1](#_Toc1375437197)

[Team Size 1](#_Toc393115276)

[Goals 2](#_Toc1406307881)

[Review 2](#_Toc262344142)

[Starter Files 2](#_Toc75705100)

[Required Hardware 2](#_Toc1865176043)

[Lab Overview 3](#_Toc1580832077)

[DC Motors and Tachometer 3](#_Toc533112057)

[PID Controllers 4](#_Toc99392623)

[Requirements Document 5](#_Toc1589680837)

[Preparation 8](#_Toc1376602629)

[Procedure 8](#_Toc186592395)

[Motor Interface 9](#_Toc626172091)

[Tachometer Interface 10](#_Toc1661818506)

[Digital Controller Design 10](#_Toc1610961725)

[Display 11](#_Toc1308934245)

[Lab Checkout 12](#_Toc293414696)

[Lab Report 12](#_Toc482793528)

[Deliverables 12](#_Toc1565043857)

[Analysis and Discussion Questions 13](#_Toc2129030293)

[Extra Credit 13](#_Toc47654589)

[Hints 13](#_Toc1564896371)

# Team Size

The team size for this lab is **4**.

# Goals

* To interface a DC motor using PWM so software can adjust delivered power.
* To interface a tachometer so the software can measure speed.
* To implement the data acquisition with input capture interrupts.
* To implement the digital controller processing with periodic interrupts.
* To study the behavior of the system as parameters to the proportional-integral (PI) controller are adjusted.

# Review

* Valvano Section 6.1.3 about using input capture to measure period.
* Valvano Section 6.3 about generating PWM outputs.
* Valvano Section 6.5 about interfacing a DC motor with a IRLD024 or IRLD120.
  + [IRLD120 data sheet](https://www.dropbox.com/s/ulcq0selng2aqfx/IRLD120.pdf?dl=1) [IRLD024 data sheet](https://www.dropbox.com/s/u49pvrn8jz70aru/sihld24.pdf?dl=1)
* Datasheet on the Pololu motor <https://www.pololu.com/product/3675> .
* Valvano Section 6.6 about integral control (you will need to add [proportional control](https://www.et.byu.edu/~tom/classes/436/ClassNotes/Class07-PI.pdf)),

# Starter Files

* Starter project:
  + Lab 10 template provided on GH classroom repo.
* Example projects:
  + PWM\_4C123
  + PeriodMeasure\_4C123
  + PeriodicTimer0AInts\_4C123
  + EE445M <http://users.ece.utexas.edu/%7Evalvano/arm/MotorTestProject.zip>

# Required Hardware

|  |  |  |  |
| --- | --- | --- | --- |
| Parts | Datasheet | Price | Source (**price source)** |
| EK-TM4C123GXL | [EK-TM4C123GXL datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/TM4C_Datasheet.pdf) | $16.99 | **TI** |
| Two 10k resistors | N/A | N/A | EER Checkout Desk |
| 1N914B 4ns switching diode | N/A | N/A | EER Checkout Desk |
| Pololu Gearmotor and Encoder assembly (P/N: 3675) | [Datasheets](https://www.pololu.com/product/3675) | N/A | EER Checkout Desk |
| IRLD120 or IRLD024 MOSFET | [IRLD120 data sheet](https://www.dropbox.com/s/ulcq0selng2aqfx/IRLD120.pdf?dl=1) [IRLD024 data sheet](https://www.dropbox.com/s/u49pvrn8jz70aru/sihld24.pdf?dl=1) | $9.61 | EER Checkout Desk  Or **Mouser**, Digikey |

# Lab Overview

The advantages of DC motors include low cost, high speed, and high torque. However, they need sensors and closed-loop controllers to operate them with predictable responses. Applications of DC motors include electric cars, robotics, industrial machines (pumps, drills, mills, and lathes), consumer products (blender, washing machine, and AC/heaters) and medical devices (pumps).

In this lab, you will spin a brushed DC motor (Figure 10.2) at a constant speed using a proportional-integral (PI) controller. You will be able to control it via a serial monitor on your PC; you should be able to set the desired speed and change controller constants to adjust how the controller behaves. The motor speed will be captured using a tachometer which informs the controller, and the results will be displayed on the ST7735.

## DC Motors and Tachometer

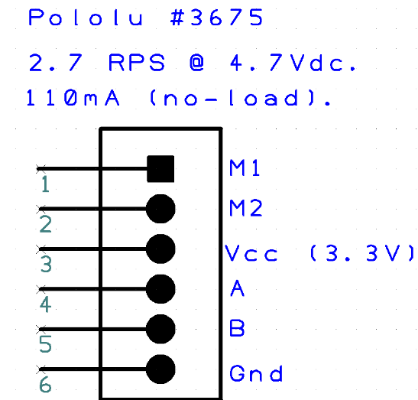
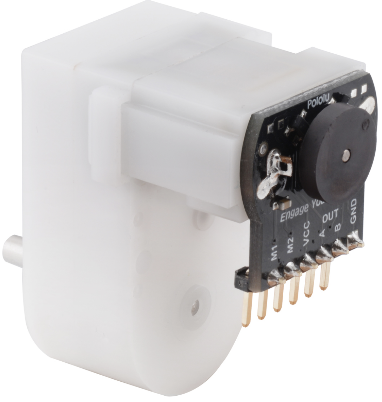


Figure 10.2. Pololu Item 3675 and Pinout.

The Pololu 3675 is a DC gearmotor and encoder assembly that represents the DC motor and tachometer. M1 and M2 drive the motor, much like the speaker circuit in Lab 3. Applying a positive voltage across the terminals causes the shaft to move in one direction, and a negative voltage causes the shaft to move in the opposite direction. You will use PWM to drive a FET to control the speed of your motor.

We recommend that students choose Port-B[6] on the TM4C123 for the PWM output. Use a 10k resistor at the gate to reduce transients. Use a 1N914 (any fast diode is ok) to remove back EMF caused by the motor (see Figure 10.3, 10.4). Connect M2 to +5V and use the IRLD024 or IRLD120 to sink current from the M1 to ground.

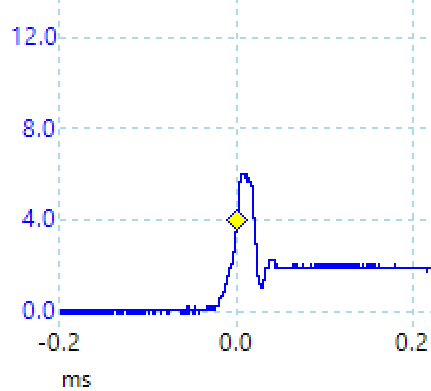
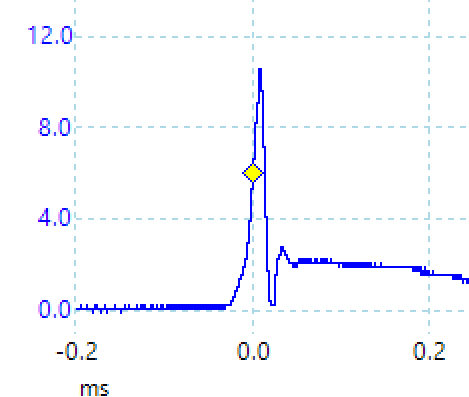


Figure 10.3, 10.4. Voltage from drain to source without and with the shunt diode

The other four pins represent the tachometer: it requires logic level power and ground, and then the A and B signals will output a train of square waves representing the angular movement of the shaft. Do not ground A or B, they are outputs and need 10k pullup resistors to work correctly. A and B are out of phase; which one shows up first from idle shows the direction of movement. You will want to connect A and B to TM4C pins that support the input capture interface.

The motor has a gearing of 120:1; it takes 120 internal rotations to make a full shaft rotation. There are 12 tachometer pulses per internal rotation (for each output). Note that if we used both A and B, and OR’d them together, we could double our measurement resolution to 24. Figure 10.5 shows the tachometer output of 973.4 Hz. We can calculate the shaft angular velocity in RPM using the following dimensional analysis:

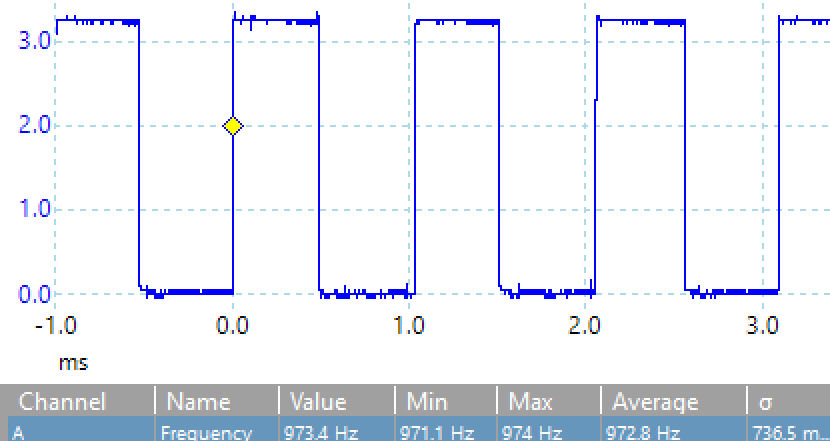


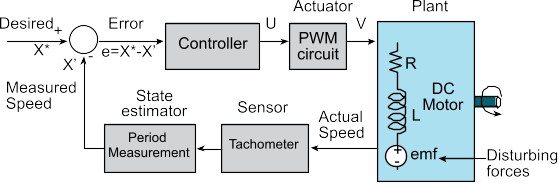
Figure 10.5. Tachometer pulse train at 973.4 Hz.

## PID Controllers

A closed loop feedback is required because not all motors are created equal. In Figure 10.9 we see the results from 8 different motors (the ones from the lab board, not this one however). One of the first steps in designing a digital controller is to make an estimate of motor speed in RPM versus PWM duty cycle. What is your range of acceptable values that bound your controller? This curve will help estimate the range of speeds that your controller will be able to manage.

A background periodic interrupt will execute the following three steps of a digital PID controller:

1. Calculate error as the difference between actual and desired speed
2. Execute a control equation to determine the next output
3. Adjust the power to the actuator to drive the error to zero



*Figure 10.6. Lab 10H Block Diagram.*

Wikipedia defines the full PID control equation as follows:



In this lab, we expect you to develop a P and then a PI closed loop controller (the first two terms). The design progression is as follows:

1. Proportional (stable but slow)
2. Integral (stable, a little faster)
3. Proportional-integral (may be unstable, fast)

## Requirements Document

1. **Overview**
   1. **Objectives**: Why are we doing this project? What is the purpose?

The objectives of this project are to design, build and test a brushed DC motor controller. The motor should spin at a constant speed and the operator can specify the desired set point. Educationally, students are learning how to interface a DC motor, how to measure speed using input capture, and how to implement a digital controller running in the background.

* 1. **Process**: How will the project be developed?

The project will be developed using the TM4C LaunchPad. The user will use switches or the UART to specify the desired speed of the motor as well as the “K” terms for the Proportional and Integral components of the PI controller. The system will be built on a solderless breadboard and run on the usual USB power. A hardware/software interface will be designed that allows software to control the DC motor. There will be four hardware/software modules: motor controller output, tachometer input, digital controller, and user interface with switches, UART and LCD. The process will be to design and test each module independently from the other modules. After each module is tested, the system will be built and tested.

* 1. **Roles and responsibilities**: Who will do what? Who are the clients?

EE445L students are the engineers, and the TA is the client. Student 1 will design, build, and test the motor controller output. Student 2 will design, build, and test the tachometer input. Student 3 will design, build, and test the digital controller. Student 4 will integrate the systems. All students will work on performance measurements and tuning the controller. *(Note to students: you are expected to make minor modifications to this document to clarify exactly what you plan to build. Students are allowed to divide responsibilities of the project however they wish, but, at the time of demonstration, all students are expected to understand all aspects of the design.)*

* 1. **Interactions with existing systems**: How will it fit in?

The system will use the microcontroller board, a solderless breadboard, and the DC motor shown above in Figure 10.2. The wiring connector for the DC motor is described in [**aLec42\_DC\_Motors.pptx**](https://www.dropbox.com/s/ez9kxdaawj3j3xy/aLec42_DC_Motors.pptx?dl=1).

* 1. **Terminology**: Define terms used in the document.

For the terms Proportional-Integral (PI) controller, PWM, board support package, back EMF, torque, time constant, and hysteresis, see textbook for definitions.

* 1. **Security**: How will intellectual property be managed?

The system may include software from TivaWare and from the book. No software written for this project may be transmitted, viewed, or communicated with any other EE445L student past, present, or future (other than the lab partner of course). **It is the responsibility of each team to keep its EE445L lab solutions secure.**

1. **Functional Description**
   1. **Functionality**: What will the system do precisely?

The CLI will provide the following capabilities:

* Entry to set the desired motor speed
* Two entries to set the Kp1 (multiplier) and Kp2 (divider) values for the Proportional term
* Two entries to set the Kp1 (multiplier) and Kp2 (divider) values for the Integral term
* The CLI will have the capability to display the modified value after it has been entered

The PID controller or the variant used may consist of a form like the following:

**MotorSpeed = rps/TBD; // Set the Motor Speed (NEED to set the denominator)**

**P = (Kp1 \* E)/Kp2; // Proportional term**

**if(P < 300) P = 300; // Minimum PWM output = 300**

**if(P >39900) P = 39900; // Maximum PWM output = 39900**

**I = I + (Ki1 \* E)/Ki2; // SUM(KiDt)**

**if(I < 300) I = 300; // Minimum PWM output = 300**

**if(I >39900) I = 39900; // Maximum PWM output = 39900**

**U = P + I; // Calculate the actuator value**

**if(U < 300) U=300; // Minimum PWM output**

**if(U >39900) U=39900; // 3000 to 39900**

**PWM0A\_Duty(U); // Send to PWM**

Note that the proportional and integral control values in the PI loop each have two sub-values, one is the multiplicand, and the other is the divisor. To prevent underflow, we always multiply first and then divide. This provides the ability to get P and I values that are effectively less than 1.00.

The motor speed should start out at zero RPS. Once the desired motor speed is entered the motor should start. (Note to students: feel free to change how the set point is established, and feel free to increase or decrease the maximum speed in accordance to how it actually works.)

Both the desired and actual speeds should be plotted on the color LCD as a function of time. The actual speed should also be graphically shown. (Note to students: feel free to specify exactly how the data is displayed. For example, you could but do not have to add numerical outputs).

* 1. **Scope**: List the phases and what will be delivered in each phase.

Phase 1 is the preparation; phase 2 is the demonstration; and phase 3 is the lab report. Details can be found in the lab manual.

* 1. **Prototypes**: How will intermediate progress be demonstrated?

A prototype system running on the LaunchPad and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

* 1. **Performance**: Define the measures and describe how they will be determined.

The system will be judged by three qualitative measures. First, the software modules must be easy to understand and well-organized. Second, the system must employ a PI controller running in the background. There should be a clear and obvious abstraction, separating the state estimator, user interface, the controller and the actuator output. Backward jumps in the ISR are not allowed. Third, all software will be judged according to style guidelines. Software must follow the style described in Section 3.3 of the book (note to students: you may edit this sentence to define a different style format). There are three quantitative measures. First, the average speed error at a desired speed of 60 RPM will be measured. The average error should be less than 5 RPM. Second, the step response is the time it takes for the new speed to hit 100 RPM after the set point is changed from 50 to 100 RPM. Third, you will measure power supply current to run the system. There is no particular need to minimize controller error, step response, or system current in this system.

* 1. **Usability**: Describe the interfaces. Be quantitative if possible. Describe how the CLI will control the motor.
  2. **Safety**: Explain any safety requirements and how they will be measured.

1. **Deliverables**
   1. **Reports**: How will the system be described?

A lab report described below is due by the due date listed in the syllabus. This report includes the final requirements document.

* 1. **Audits**: How will the clients evaluate progress?

The preparation is due at the beginning of the lab period on the date listed in the syllabus.

* 1. **Outcomes**: What are the deliverables? How do we know when it is done?

There are three deliverables: preparation, demonstration, and report. (Note to students: you should remove all notes to students in your final requirements document).

# Preparation

1. **Edit the requirements document** to reflect your design.
2. **Draw the circuit** (in KiCAD) that integrates your motor, tachometer, and other IO to the TM4C. Please power the motor from 5V and preferably from an off-board power supply rather than the TM4C. Your circuit will look similar to the Lab 3 speaker circuit and has the same vulnerabilities: the back EMF of the motor/speaker can generate voltages high enough to destroy your MCU and possibly your laptop. Use a snubber diode to suppress this voltage spike.
3. **Collect components and breadboard the system** to reflect your schematic.
4. **Design the system diagram and software modules** for your implementation.

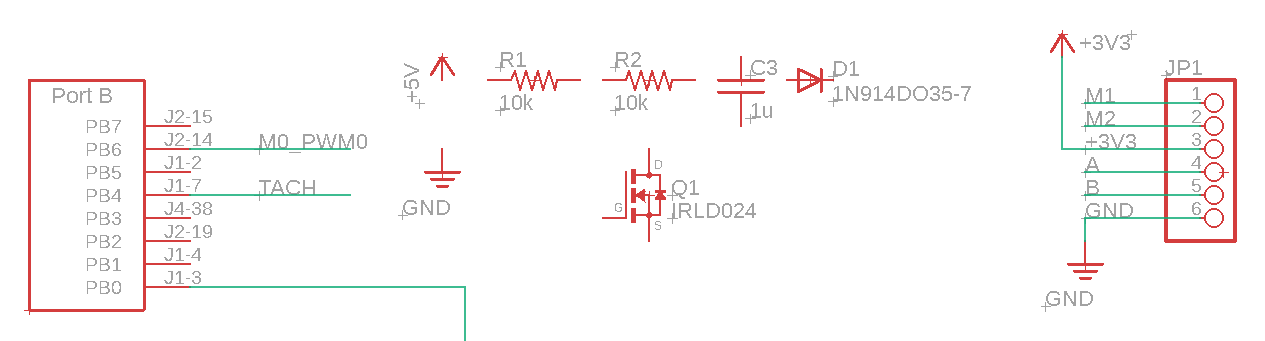


Figure 10.7 Components that could be used for Lab 10.

# Procedure

The procedure may be best split into four tasks, one for each member of the group. They are:

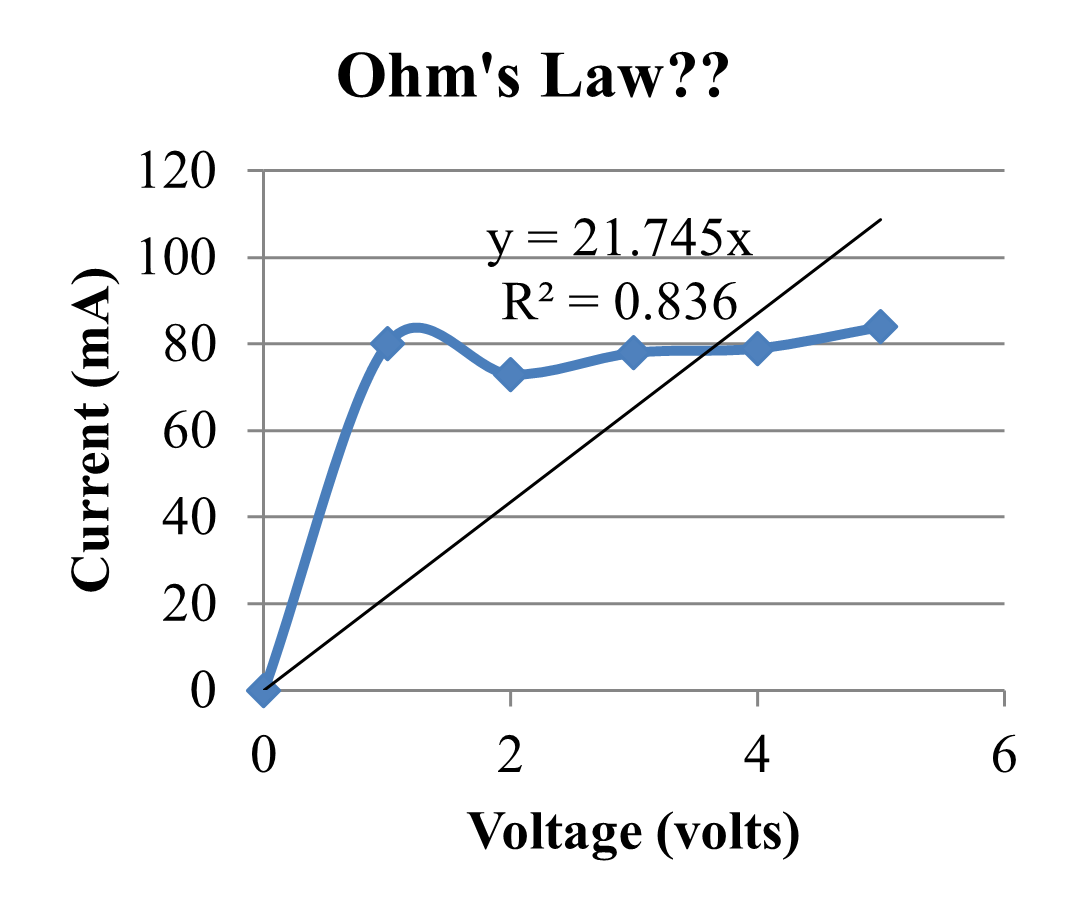
* Interfacing and driving the motor,
* Interfacing and reading from the tachometer,
* Implementing the PID controller,
* Interfacing and visualizing time series on the display.

Team members will need to get together at the culmination of these tasks to integrate the parts – so proper division of tasks, strong understanding of their API, and liberal documentation and unit tests (as usage reference) will make this job easier.

## Motor Interface

1. Make sure you get your TA to check your motor circuit before connecting it to the MCU to prevent *incidents*.
2. **Measure the resistance of the motor**.
3. **Simulate a PWM output of 100% duty cycle** (FET is shorted/ON) to drive your motor at full speed.
   1. **Measure the current draw through the motor** with an ammeter or using the power supply display. Please contact your professor or TA if this current is vastly different from **120mA**.
   2. **Measure the voltage drop across the motor** with a voltmeter or using the power supply display. Call this full speed motor voltage . The voltage drop across the MOSFET should be much less than 1V. Note: should equal 5V.
4. **Repeat this process at various PWM duty cycles.** Assume the PWM period is 1ms: see the Hints section on why you might want to change this.
5. **Reconstruct Figure 10.8** to make an I-V graph of the motor. The average motor voltage can be assumed as the . Draw the line corresponding to the measured resistance.

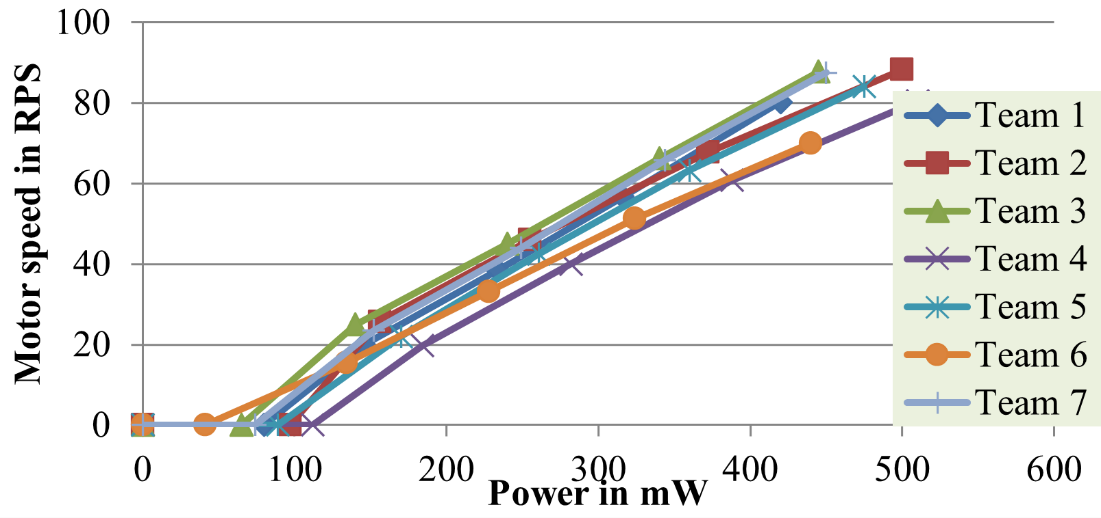
Figure 10.8 was measured with the lab board motor, so expect your measurements to be different. However, the motor does not behave like a simple resistor. The response will also be a function of the mechanical torque (or friction) applied to the shaft. Make sure your experiment is performed at no load.



*Figure 10.8: Current to the lab board DC motor versus applied voltage*

## Tachometer Interface

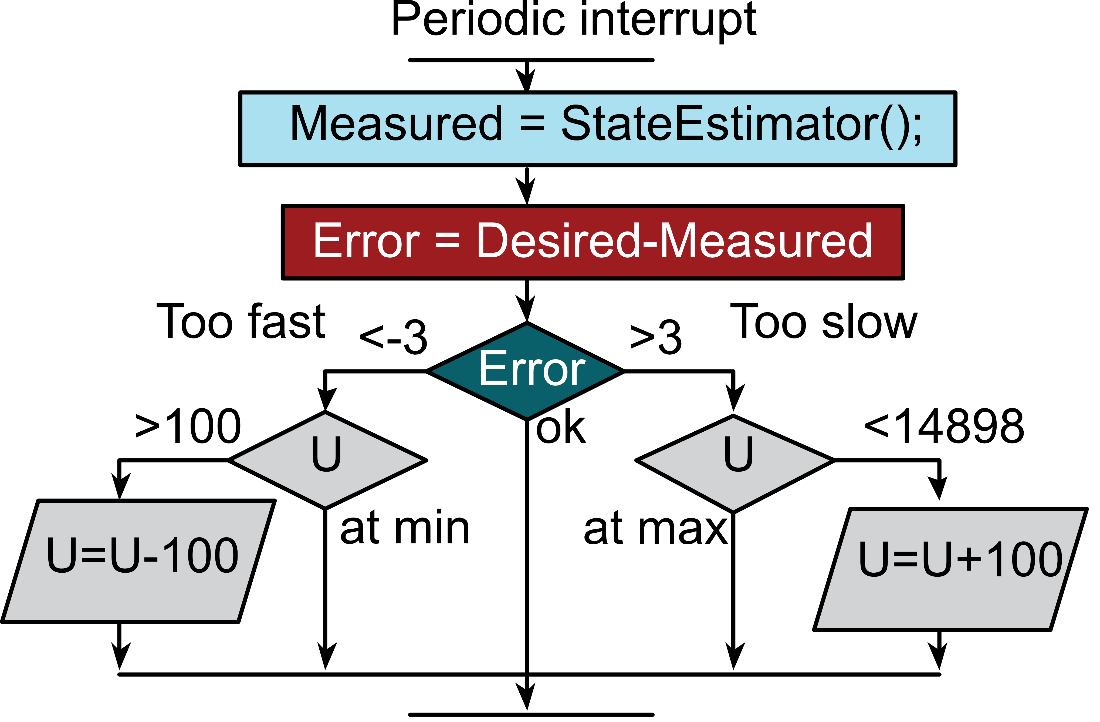
1. **Verify the tachometer interface** generates a digital (0V and 3.3V) pulse train signal using an oscilloscope, at 100% duty cycle and no load. Measure the RPM using the pulse train spacing.
2. **Verify the tachometer software** returns a rate averaged over time that matches what is seen in the oscilloscope with the same setup. Speed measurement must occur in the background using an input capture and periodic interrupt.
3. **Measure the motor speed at various PWM duty cycles.** Combine the current and voltage data from *Motor Interface* procedure to estimate the average motor power (). Create a graph similar to Figure 10.9 below. See the background for calculating the motor speed.



*Figure 10.9: Motor speed versus applied power on seven different motors.*

## Digital Controller Design

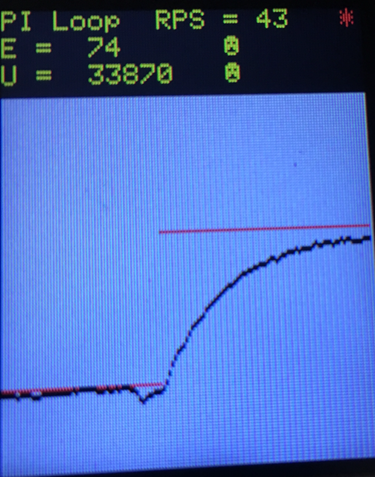
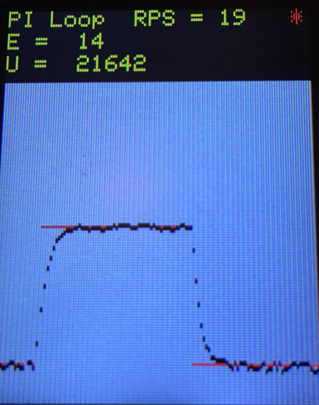
1. **Develop a simple proportional (incremental) controller** similar to Figure 10.10. Choose a PWM period, a range of usable speeds, and a controller rate consistent with the data collected in the above two procedures. Verify this works with the motor and tachometer interfaces.
2. Then **add an integral controller** to make a PI controller. You will need to tune this such that you develop coefficients for the four types of controller damping (critically damped, over damped, under damped, and oscillating). See Figures 10.11 - 10.14.



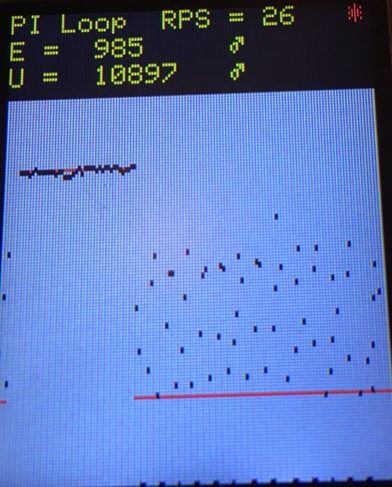
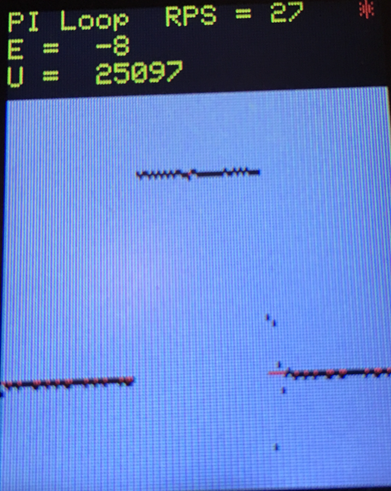
*Figure 10.10: Incremental controller. Change the +3, -3, +100, -100, 14898 constants.*

## Display

1. Create a mechanism to visualize and control the system (a method to control the system has been created for you in the form of **pid\_controller\_parser.c**). In the images below measured and desired speeds are plotted versus time as the set point is increased from 20 to 50 RPS. Ensure that the error, integral error, and current RPS is shown. You may find some useful ST7735 routines in Valvanoware.



*Figure 10.11, 10.12. Critically damped and Over damped PI Loop.*



*Figure 10.13, 10.14. Under-damped and Unstable PI loop.*

## Deliverable 1

I-V graph of motor characteristics (with measured resistance line).

## Deliverable 2

Include in your report a scope capture of tachometer output. Explain what you are seeing on the scope, and how this helps you determine the speed of the motor.

## Deliverable 3

Create a motor speed versus applied power graph. This can be done by using a varying duty cycle from the TM4C to control the motor while using a bench power supply to measure and supply power to the motor. Include a graph showing the power in watts on the X axis, and the motor speed in RPM on the y axis. Your graph should have at least 5 reasonably spaced data points.

## Deliverable 4

Take pictures of graphs of 4 damping. Specifically, include in your report four images showing the following conditions:

1. Critically damped PI loop
2. Over-damped PI loop
3. Under-damped PI loop
4. Unstable PI loop

This can be done by using the ST7735 or some other device that allows you to plot the actual and required speeds of the motor. You will need to change what PID configuration you are using to control the motor for this deliverable.

## Deliverable 5

Measure the CPU utilization to run the PID controller. Similar to what you have done in previous labs, this can be performed using any debugging technique you prefer. Be sure to show your work for the math done in your report.

## Deliverable 6 (15pts Extra Credit)

You may (for up to a +15% bonus) perform the following three performance measurements for a single, best-effort, critically damped PI controller. Each measurement is worth 5% each. Please explain why we should know these measurements and how they might affect your system design.

1. Average steady state controller error
2. Average response time until steady state convergence
3. Average percentage overshoot/undershoot

Ensure that all metrics are calculated from a standardized experiment (e.g. 5 RPS set to 20 RPS, 5 runs each) and specify your methodology and results in your lab report. You may write your own software routines to capture this data and analyze it for you automatically.

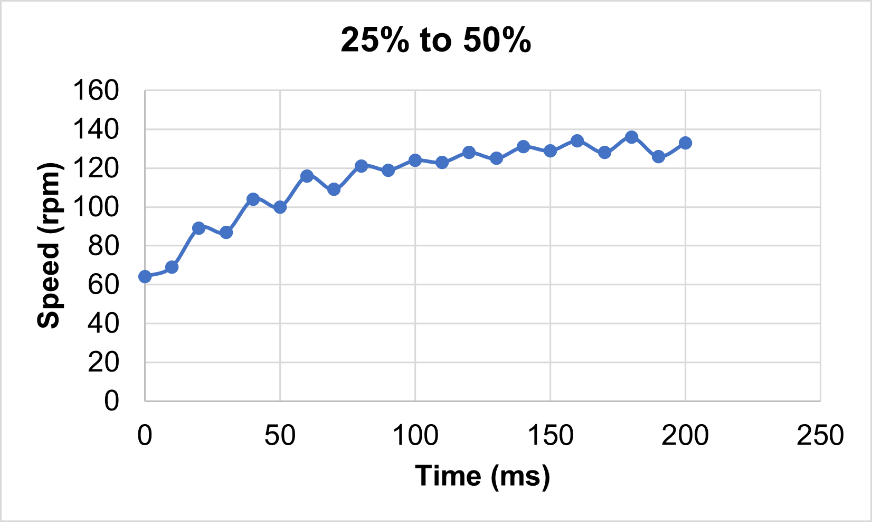
# Lab Checkout

For lab checkout, the team should be able to demonstrate working PID control of the motor by setting the speed of the system and visualizing the RPS over time using the display. The team will be asked to demonstrate four types of PID controller damping:

1. Critically damped PI loop
2. Over-damped PI loop
3. Under-damped PI loop
4. Unstable PI loop

Students will be asked questions regarding the operation of components that their partners may have implemented.

# Hints

1. The USB +5V regulated supply is specified to 500mA total current. We strongly suggest you do not mechanically load the motor. It runs 120mA unloaded, but much larger if you load the motor. If your PC disconnects you, then you will need to power cycle your computer and restart the OS. In the worst case, your USB hub in your computer will be blown and can no longer be used properly.
2. To choose a PWM period you will need to know the time constant of your motor. Figure 10.15 measures speed as a function of time as PWM is increased from 25% to 50%. You can estimate the time constant of this motor to be 50 ms. Therefore, if the PWM period is 1ms, the motor will only respond to the average power (duty cycle), and not to the individual high and low signals of the interface.
3. 

*Figure 10.15: Step response of the DC motor used to simulate the time constant. Expect your measured speeds to be slower (this data was collected at 7.2V)*

1. The tachometer interface may require filtering to improve SNR. However, be careful to minimize time delay. Time delays in the digital controller will cause the system to be unstable. Examples of various filters (Rolling mean, median, Kalman) may be provided in the starter code.
2. You may need a separate periodic timer interrupt to determine if the motor is spinning too slow to be accurate or not spinning at all.
3. The discrete version of an integral is a sum.