Name: \_\_Replace with your name(s)\_\_\_\_

EID: \_\_Replace with your EID(s)\_\_\_\_\_

Semester: Spring 2024

Course: ECE445L

1. ***Requirements Document:***
2. I have completed the Project Requirements Document at the end of this lab document (Type yes if true): \_\_\_\_\_

B) ***Objectives*:**

1. In a few sentences, summarize the goal of this lab.

B) ***Hardware Design*:**

1. Updated schematic showing the DC motor interface.

C) ***Software Design Deliverables:***

1. I have pushed my code to GitHub for grading (Type yes if true): \_\_\_\_\_
2. Briefly describe the system design. Alternatively, create a system call graph.

D) ***Measurement Data:***

1. Deliverable 1: I-V graph of motor characteristics
2. Deliverable 2 Scope capture of tachometer output
3. Deliverable 3: Motor speed versus applied power graph
4. Deliverable 4: Graphs of 4 PI configurations
5. Deliverable 5: CPU utilization to run the PID controller
6. Deliverable 6 (15pt EC): Performance measurements

E) ***Analysis and Discussion Questions:***

1. What is torque? What are its units?
2. You implemented an integral controller because it is simple and stable. What other controllers could you have used? How would another type of controller have been superior to your integral controller?
3. Why do we need to use a PID loop (Or PI loop as the case may be) to control the motor? Would we have needed a PID loop to control a servo or a DAC? What is the difference?
4. Explain what parameters were important for choosing a motor drive interface chip (eg. IRLD024 or IRLD120). How does your circuit satisfy these parameters?
5. How did you tune your PID controller? What was your process? Is there a better way?

F) ***Project Requirements Document:***

1. Overview

1.1. 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.

1.2. 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 at 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.

1.3. 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 design build and test the CLI interface. All students will work performance measurements and tuning the controller. *(note to students: you are expected to make minor modifications to this document in order 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.)*

1.4. 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).

1.5. 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.

1.6. 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.**

2. Function Description

2.1. 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 CLI will have the capability to handle the BS and DEL keys, where the character entered is deleted from the input buffer and the display refreshed with the reentered value
* The CLI will ignore CR only entries and keep the previous values (Figure 10.8(a))

A picture containing graphical user interface

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Figure10.8(a) Figure 10.8(b)

* The following Control-Key capabilities will be provided:
* CTRL-C: Cancel the entry sequence and redisplay all the current motor control register values: New\_Speed, Kp1, Kp2, Ki1and Ki2 (Figure 10.8(b))
* CTRL-S: Enter the continuous status display mode (Figure 10.8(c). In this mode the following must be displayed:
* Xstar & RPM Values
* PI Error Value (E)
* PWM Duty Cycle Control Value (U)

Table

Description automatically generated with medium confidence

Figure 10.8(c)

The PI control equation may be in this form:

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 like Figure 10.12 The actual speed should also be graphically shown on the Blynk App. (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).

2.2. 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.

2.3. 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.

2.4. 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.

2.5. Usability: Describe the interfaces. Be quantitative if possible. Describe the how the CLI will control the motor.

2.6. Safety: Explain any safety requirements and how they will be measured.

We expect the motor current to be less than 120 mA. Please place a current probe on the motor interface the first couple of time you run the motor to make sure you have wired it correctly. However, under heavy friction this current could be 2 times higher. Therefore, please run the motor unloaded. Connecting or disconnecting wires on the protoboard while power is applied will damage the microcontroller. Operating the circuit without a snubber diode will also damage the microcontroller.

3. Deliverables

3.1. 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.

3.2. 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.

3.3. 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).