Lab 10E DC Motor Control

This laboratory assignment accompanies the book, Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154, by Jonathan W. Valvano, copyright © 2018.

Goals
- To interface a DC motor and tachometer,
- To implement background processing with periodic interrupts,
- To develop an integral controller.

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 TIP120,
- Valvano Section 6.6 about integral control
- Valvano Figure 8.20 about interfacing a sensor with a saturated mode op amp
- Video demonstration http://youtu.be/ug-ZlOSbb4M (this video has the wires reversed)

Starter files
- **`PWM_4C123 PeriodMeasure_4C123 PeriodicTimer0AInts_4C123`**
- **`Lab10E_Artist.sch`**

Background

The advantages of DC motors include low cost, high speed, and high torque. However, they need sensors and closed-loop controllers in order 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 at a constant speed using an integral controller, see Figure 10.1. Two switches will allow the operator to increase or decrease the desired speed. A tachometer will be used to measure the motor speed. A background periodic interrupt will execute the three steps of a digital 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 in an attempt to drive the error to zero

The software will set the power delivered to the motor using pulse-width-modulation (PWM). The essence of an integral controller is the amount you add to the power is linearly related to the error. If the motor is spinning much too slowly, you will increase power a lot. If the motor is spinning just a little bit too slowly, you will increase power a little bit. If the motor is spinning much too quickly, you will decrease power a lot. If the motor is spinning just a little bit too quickly, you will decrease power a little bit. The input capture input and the PWM output must be written at a low level, like the book, without calling TivaWare driver code. Other code (LCD, timers, GPIO, and PLL) can use TivaWare driver code.

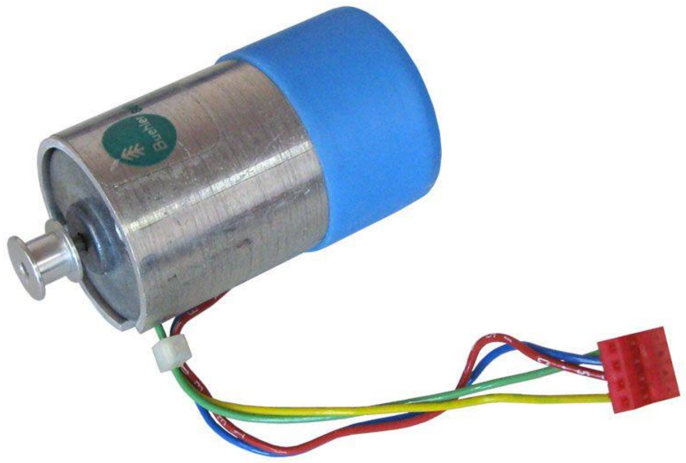

Figure 10.1. DC motor with tachometer, CAT# DCM-408, http://www.allelectronics.com (the two wires located near the shaft are the tachometer, and the two wires going under the plastic cap are the brushed DC motor).

Required Hardware

EK-TM4C123GXL, www.ti.com	$12.99
Sitronix ST7735 Color LCD, http://www.adafruit.com/products/358	$19.99
DC Motor, DCM-408, http://www.allelectronics.com	$6.25

Required Parts

Parts	www.Mouser.com	www.element14.com	www.digikey.com	Price
TIP120 N-channel Darlington	**512-TIP120**	**TIP120**	**TIP120-ND**	**$0.68**
1N914B 4ns switching diode	**512-1N914B**	**1N914B**	**1N914B-ND**	**$0.10**
OP2350 op amp	**595-OPA2350PA**	**OPA2350PA**	**OPA2350PA-ND**	**$5.18**

The DC motor in Figures 10.1, 10.2, and 10.3 has a tachometer built-in. You can do this lab with any DC motor (see www.jameco.com). However the controller needs a tachometer, and you could build one with a patterned disk and an IR reflectance sensor, like a QRB1134 (see Figure 10.9). The LM4F120 does not have hardware PWM. So if you have an LM4F120 LaunchPad you will need to use one of the timers and software to create the PWM output.

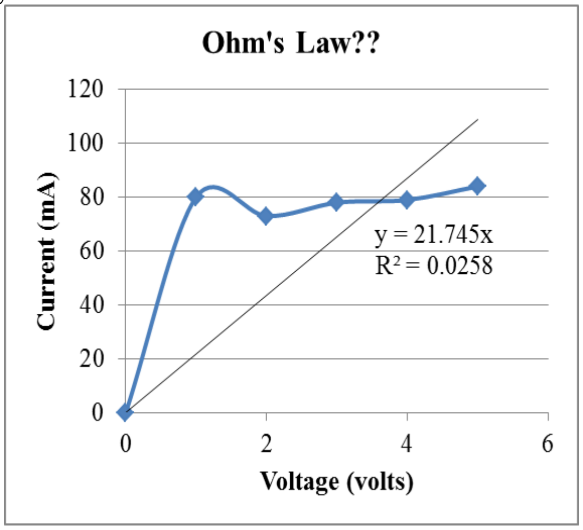

Figure 10.2. The current to the DC motor is plotted versus applied voltage. Notice the DC motor does not behave like a resistor. This response will also be a function of the mechanical torque (or friction) applied to the shaft.

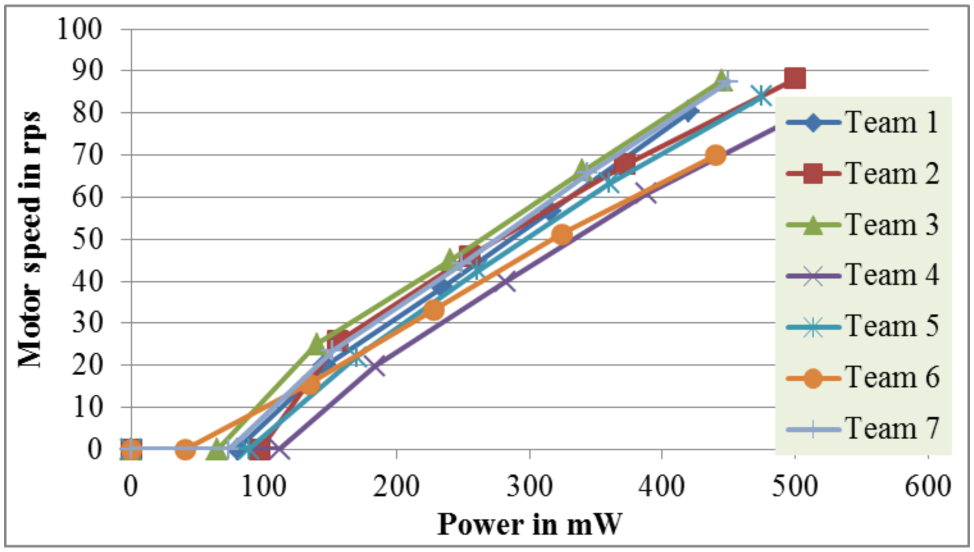

Figure 10.3. The motor speed is plotted versus applied power on seven different motors. Power is motor voltage multiplied by motor current multiplied by PWM duty cycle. Motor speed (rps) is tachometer frequency (Hz) divided by 4. Notice at 100 mW, these DC motors do not spin at all. Also notice the variability between these seven motors. This response will also be a function of the mechanical torque (or friction) applied to the shaft.

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 LaunchPad. There will be two switches that the operator will use to specify the desired speed of the motor. The system will be built on a solderless breadboard and run on the usual USB power. The system may use the on board switches or off-board switches. A hardware/software interface will be designed that allows software to control the DC motor. There will be at least five hardware/software modules: tachometer input, switch input, motor output, LCD output, and the motor controller. 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 A will build and test the sensor system. Student B will build the actuator and switch input. Both students will work on 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, both 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 in Figure 10.1. The wiring connector for the DC motor is described in the PCB Artist file `Lab10E_Artist.sch`. It will be powered using the USB cable. **You must use a +5V power from the lab bench, but please do not power the motor with a voltage above +5V. Do not connect this bench supply to Vbus (LaunchPad +5V). However, you must have a common ground.**

1.5. Terminology: Define terms used in the document.

For the terms integral 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 the team to keep its EE445L lab solutions secure.

2. Function Description

2.1. Functionality: What will the system do precisely?

If all buttons are released, then the motor should spin at a constant speed. If switch 1 is pressed and released, the desired speed should increase by 5 rps, up to a maximum of 40 rps. If switch 2 is pressed and released, the desired speed should decrease by 5 rps, down to a minimum of 0 rps. *(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. Remember the story about the Texas sharpshooter?)*

Both the desired and actual speeds should be plotted on the color LCD as a function of time similar to Figure 10.4 *(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.

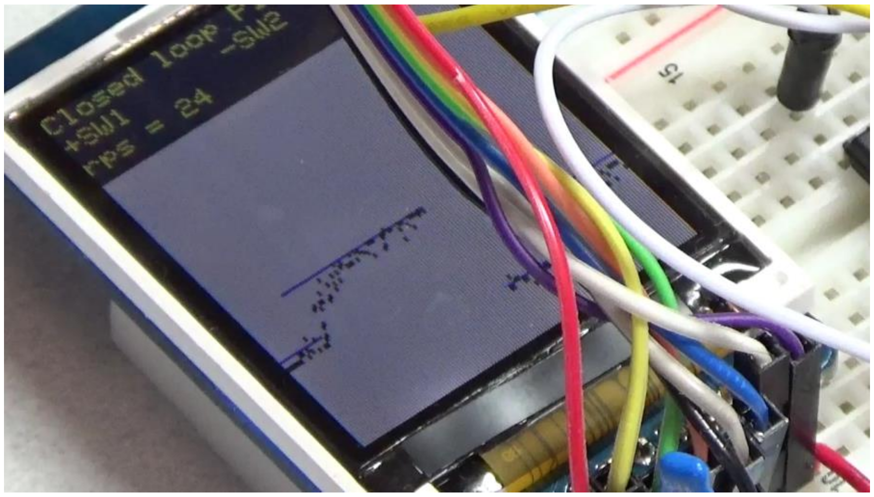

Figure 10.4. Measured and desired speeds are plotted versus time as the set point is increased from 20 to 25 rps.

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 an integral 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 rps will be measured. The average error should be less than 5 rps. Second, the step response is the time it takes for the new speed to hit 60 rps after the set point is changed from 40 to 60 rps. 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.

There will be two switch inputs. The tachometer will be used to measure motor speed. The DC motor will operate under no load conditions,

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

Figure 10.2 shows that under a no load condition, the motor current will be less than 100 mA. However, under heavy friction this current could be 5 to 10 times higher. Therefore, please run the motors 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).*

Preparation (do this before your lab period)

1) Copy and paste the requirements document from this document. Edit it to reflect your design.

2) Edit the **`Lab10E_Artist.sch`** file showing the interface to the motor and to the tachometer. Make sure your TA approves the design before connecting it to the microcontroller. Include pin numbers and resistor/capacitor types and tolerances. Be sure the interface circuit (e.g., TIP120) you select can sink enough current to activate the coil. Power the motor from the +5V (VBUS) and not +3.3V. Also, verify the driver can supply (I_{CE}) the required current for the DC motor. Collect all the parts you need before lab starts. This interface will require a 1N914 snubber diode. If do not properly connect this diode, the back EMF will destroy your microcontroller.

3) Design three software modules that implement the digital controller: tachometer input, motor output, and the motor controller with two switches. A "syntax-error-free" version of the header files is required as preparation. This will be checked off by the TA at the beginning of the lab period. If you wish to see more timer and PWM examples, see the EE445M sensor board and motor board starter projects

https://www.dropbox.com/s/vtjc5j6hz9b15dj/SensorTestProject.zip?dl=1
http://users.ece.utexas.edu/%7Evalvano/arm/MotorTestProject.zip

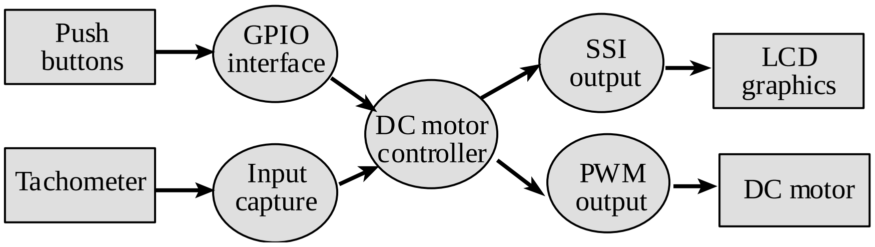

Figure 10.5. Data flows from the tachometer and from the switches to the DC motor and to the LCD.

Figure 10.6 shows a possible call graph of the system. Dividing the system into modules allows for concurrent development and eases the reuse of code.

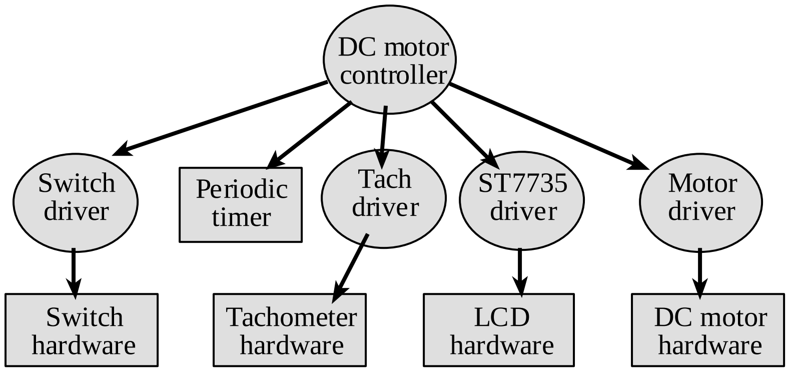

Figure 10.6. A call graph showing the modules used by the DC motor controller.

A tachometer device driver means you create **`tach.h`** and **`tach.c`** files, separating mechanisms (how it works) from policies (what it does). For example, you could implement two public functions: initialize the tachometer and return the most recent measurement in 0.1 rps units. It is a poor design practice to use public globals to pass data from one module to another. Similarly, make **`motor.h`** and **`motor.c`** files. For example, you could implement three public functions: initialize the PWM, set the duty cycle, and turn off the motor. A switch device driver means you create **`switch.h`** and **`switch.c`** files For example, you could implement two public functions: initialize GPIO and return the current position of the two switches. Design the software in a manner that makes it easier to understand, debug, modify and reuse in other projects. Design the system in a manner that would allow you to reassign the pin connections of the interface (e.g., moving I/O from one port to another), by making changes to **`tach.c`** **`switch.c`** and **`motor.c`** without requiring modifications to **`main.c`** **`switch.h`** **`tach.h`** or **`motor.h`**.

Procedure (do this during your lab period)

1) With an ohmmeter, measure the resistance of the motor coil. Using the bench supply, apply 1, 2, 3, 4, and 5V across the coil and simultaneously measure both voltage and current. Do this under no load and without any connections to the LaunchPad. Create a graph similar to Figure 10.2, and include the *I* vs *V* line that defines the effective resistance.

2) Build the hardware interface to the motor, and the interface from the tachometer, but do not connect it yet to the microcontroller, as shown in Figure 10.5. Use a switch as inputs for the motor interface. Connect the tachometer output to a scope. When the switch is pressed the motor should spin and when the switch is not pressed the motor should stop. Measure the base-emitter current, and the collector-emitter current while the motor is spinning. Compare the measured values with theoretical design equations from the book (are the currents more or less than you expected?) Using a scope, verify the tachometer interface generates a digital (0V and 3.3V) signal that could be safely connected to a microcontroller input. The very fast turn-off times of the digital transistors can easily produce 100 to 200 volts of back EMF, so please test the hardware before connecting it to the microcontroller. Make sure your TA checks your hardware diagram before connecting it to the microcontroller.

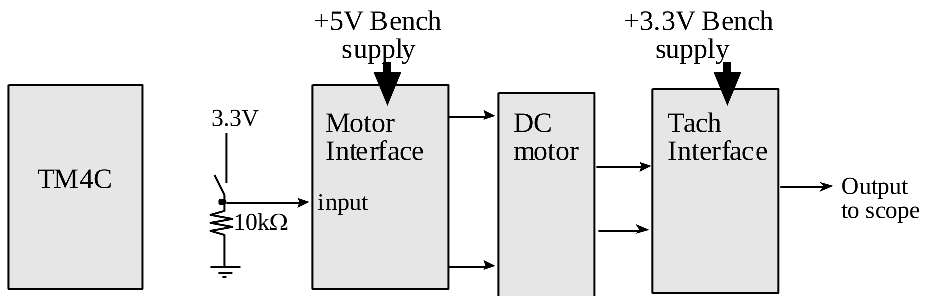

Figure 10.7. Hardware test procedure, microcontroller not connected at all. +5V and +3.3V power is supplied by a bench supply.

3) After you are sure the motor and tachometer interfaces are operating properly, replace the switch in Figure 10.7 with a PWM output of the microcontroller, as shown in Figure 10.8. If you are at all unsure about this lab ask your TA for help. Once you are sure the hardware is operating properly, run a simple software function that rotates the motor at a slow velocity to verify the hardware/software interfaces. In particular, debug the tachometer input and PWM output drivers separately before attempting to run the controller. Connect both the PWM output and motor voltage to the scope and take a screen shot. Identify the action of the snubber diode on the motor signal. Connect both the tachometer signal and digital connected to the input capture pin to the scope and take a screen shot.

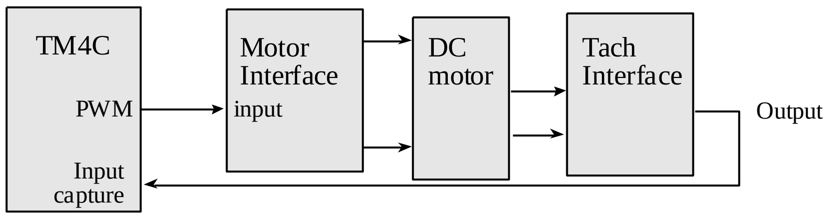

Figure 10.8. Hardware block diagram. +5V and +3.3V power is supplied by the LaunchPad.

4) Implement and debug the motor controller. Document clearly the operation of the routines. Interrupting input capture will measure the period of the tachometer and calculate motor speed. Consider calculating speed as a decimal fixed point number in 0.1 or 0.01 rps. The periodic interrupt-driven background thread will execute the digital controller, while the foreground thread initializes the system then periodically checks for switch input. Figure 10.5 shows the data flow graph of the DC motor controller. An integral controller is pretty stable. If the speed oscillates wildly (all stop- full speed) then decrease the controller constant. If it is VERY slow to response, increase the constant. Measure the maximum time to execute each of your ISRs. What is the average controller error (difference between desired and actual speed)? What is the approximate time it takes the speed to stabilize after the set point is increased from 20 to 25 rps? You do not need to tweak it to minimize error or response time. Your system performance should be similar to Figure 10.4 and the YouTube video, http://youtu.be/ug-ZlOSbb4M.

5) Remove the USB cable and carefully power your DC motor system using a lab power supply directly to the +5V pin on the board. Set the voltage to +5V and <u>measure the required current to run the DC controller system. Take a measurement with and without the motor spinning.</u> Double-check the positive and negative connections before turning it on.

Deliverables (exact components of the lab report)

A) Objectives (final requirements document)

B) Hardware Design (PCB Artist file)

DC motor and tachometer interfaces, showing all external components

LCD and switch interfaces, showing all external components

C) Software Design (upload your files as instructed by your TA)

Make sure to include units on all your software variables

Make a clear distinction between variables used for debugging and variables needed in the controller

If you organized the system different than Figure 10.3 and 10.4, then draw its data flow and call graphs

D) Measurement Data (<u>underlined sections of the lab manual</u>)

Procedure 1) Give the voltage, current, and resistance measurements

Procedure 2) I_{BE} and I_{CE} while spinning.

Procedure 3) Two screen shots of the hardware in operation.

Procedure 4) Specify the maximum time to execute once instance of the ISR

Procedure 4) Specify the average controller error

Procedure 4) Specify the approximate response time

Procedure 5) Measurements of current required to run the system, with and without the motor spinning

E) Analysis and Discussion (give short 1 or two sentence answers to these questions)

1) What is torque? What are its units?

2) Draw an electrical circuit model for the DC motor coil, and explain the components. Use this circuit model to explain why the current goes up when friction is applied to the shaft

3) Explain what parameters were important for choosing a motor drive interface chip (e.g., TIP120 or 2N2222). How does your circuit satisfy these parameters?

4) You implemented an integral controller because it is simple and stable. What other controllers could you have used? For one other type of controller how would it have been superior to your integral controller.

5) It the motor is spinning at a constant rate, give a definition of electrical power in terms of parameters of this lab? Research the term "*mechanical power*". Give a definition of mechanical power. Are the electrical power and mechanical power related?

Checkout (show this to the TA)

You should be able to demonstrate correct operation of the DC motor system. Be prepared to describe how your motor interface works. Explain how hysteresis was or could have been used in your tachometer interface. Demonstrate debugging features that allow you to visualize the software behavior.

A software and report files must be uploaded as instructed by your TA.

Hints

1. Be sure the interface driver (e.g., the 2N2222 or TIP120) has an I_{CE} large enough to deliver the needed coil current. Although many motors will operate at voltages less than the rated voltage, the torque is much better when using the proper voltage. Remember to actually measure the coil current rather than dividing voltage by resistance.
2. You could use a periodic interrupt to determine if the motor is stopped or spinning very slowly.
3. The USB +5V regulated supply is specified to 500 mA total current. Your motor requires almost all of this current. We strongly suggest you do not mechanically load the motor. It runs 150 mA unloaded, but over 500 mA if you load the motor. If your PC disconnects you, then you will need to power cycle your computer and restart the OS.
4. See the data sheets for the devices used in your interfaces at
http://users.ece.utexas.edu/~valvano/Datasheets/

OPA2350

1N914

B3F-1059, B3F-switch

TIP120

CarbonFilmResistors,CarbonFilmResistors2