Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.115 Microprocessor Project Laboratory Spring 2022

Laboratory 4
Electromechanical Actuators and Feedback Control

Issued: March 30, 2022 Due: April 11, 2022

GOALS: Continue learning how to connect peripherals to the R31JP.
Use an A/D converter and LCD screen.
Learn about electromagnetic actuators.
Understand, design, and construct a digital feedback controller.
Practice writing C code for the PSoC.
PRIOR to starting this lab:
Read: LM18293/L293D datasheet, pgs 429-449, 833-845 in Scherz 3rd ed. (and other sections as indicated).

Our overall goal in this laboratory is to understand common electromagnetic actuators: relays, AC and DC motors. These actuators are ubiquitous in consumer products including kitchen appliances, CD players, and automobiles with powered windows, steering, brakes, and locks. They are used in other products in the home, office, and manufacturing and industrial facilities. Electromagnetic actuators are "power'' devices that may require substantially more current or voltage than the microcontroller or DAC can provide. We will need to use a bank of high current buffers that can drive electromagnetic actuators.

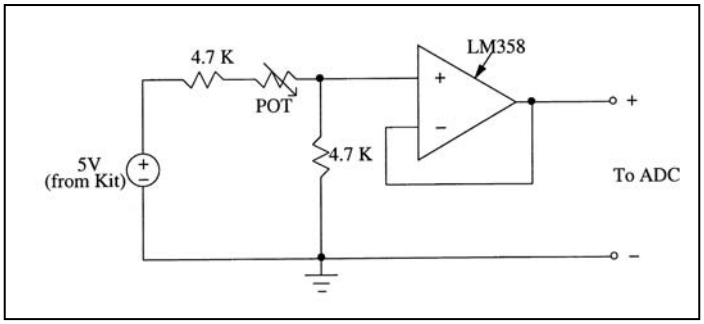
Achieving precise positions or speeds with actuators may prove challenging, as we will see. We use feedback loops to help get the actuators to do what we want. A feedback loop is a system that responds to an error or difference. Specifically, a feedback loop looks at the difference between the desired value of a system attribute (like speed or temperature or voltage) and the actual, measured value of the system attribute. Control actions are made in response to this difference. Therefore, to make a **closed**-loop system, we need more than just a command source (like a DAC and amplifier) to drive a "plant" (like a motor) with commands. This distinguishes closed-loop systems from open loop systems. For a closed loop, we will need some means to measure or sense what the actuator is doing. In this lab, we will add an analog-to-digital converter (ADC) for making measurements of analog sensors. Our hope when we construct a feedback loop is that we will be able to achieve system performance that is "robust.'' For example, we hope that the system will be minimally or unnoticeably affected by reasonable disturbances in the environment and errors in our modeling.

EXERCISE 1: Add more useful peripherals to your lab kit

To make some of the experiments easier to understand, it will be helpful to have a convenient "readout" or display that you can use to indicate gain settings, command set points, measured values, etc. So, let's add a flexible display that's become relatively inexpensive these days: a liquid crystal display. We will issue you a special LCD for the R31JP, with pins that fit (be careful!) into a kit breadboard strip – this is NOT the LCD in the PSoC Big Board kit! The LCD is a small embedded system! Besides the actual liquid crystal screen, the display contains a simple microcontroller with on-board memory. To use the display, you first "program" it for the display mode of your choice, and then load data to be displayed in a memory bank in the LCD. The suggested reading section in Scherz will tell you what you need to know to connect and "program" the LCD display. As discussed in lecture, connect the LCD directly to the 8255

on your kit from Lab 3. Use MINMON to "play" with the LCD, poking instructions and data at the display through the 8255 until you are comfortable with it.
Please do the following:

- Connect your LCD to the 8255 on your kit. Do not use the LCD backlight. Do make sure you adjust the LCD contrast so that you can see the letters and numbers on the screen. Test the LCD carefully, first with MINMON "W" commands, to make sure that you understand how the LCD works. Then, write a simple subroutine that you can re-use later that loads a string onto the display. You can use any approach you like. One method might be to have the subroutine load a fixed length string of character bytes from a location in memory. You could test this routine using your MINMON "V" command. Alternatively, you could have the subroutine always write a fixed string to the screen and recompile your code to change the text. Test your display with a program that writes "6.115 Rules!" to the LCD (You can be sure we'll want to see a demo of this).
- Connect your ADC0804 in the memory-mapped I/O space on your kit. Configure your ADC so that it accepts input voltages in the range from 0 to 5 volts, mapping these to digital bytes in the range from 00h to FFh. Make **VERY** certain that you never apply less than 0 volts or more than 5 volts to the ADC. These chips are expensive. Always check signals you apply to the input before you connect them to the ADC to make **sure** they remain in the acceptable range. Test your ADC by writing a program for the R31JP that samples the input to the ADC and displays the sampled byte on the Port 1 lights. Use the op-amp circuit shown below to drive the analog input of the ADC (create the variable resistor by using two leads on a potentiometer).



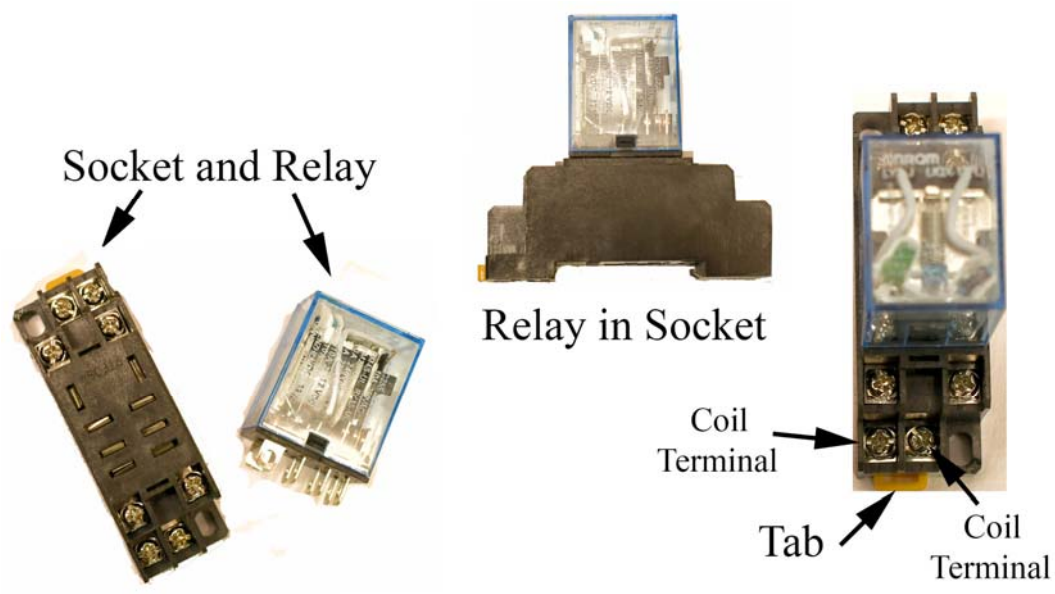

**Use ground and your 5 volt power supply to power the LM358**, ensuring that it's output voltage will be safe (below 5 volts). If you pick a reasonable pot, you should be able to get a variable voltage source connected to the ADC that cannot go above 2.5 volts (a conservative choice for testing purposes), and which can get close to zero volts. Apply some DC levels to the analog input of the ADC and then compare the byte on Port 1 to the measurement of a real voltmeter connected to the input of the ADC0804. If everything is working, you should, for example, see a light pattern corresponding to 128 decimal on the port 1 lights when the voltmeter indicates 2.5 volts applied to the ADC input.
- Write a program for the R31JP that samples the input to the ADC and displays the voltage on the LCD screen. That is, make a simple voltmeter. Your voltmeter should be sensitive to one digit after the decimal, which is well within the resolution of your ADC. For example, if you apply 2.10 volts exactly to the ADC input, your voltmeter should display "2.1" on the LCD panel. Test your ADC by applying some DC levels to the analog input and then comparing your LCD reading to a real voltmeter connected to the input of the ADC0804. Once again, use this op-amp circuit shown above to drive the analog input of the ADC.
- READ the L293D datasheet. (NOTE: You may have an **LM18293 or an L293D** in your kit. They are the same for our purposes, and **we will refer to them interchangeably**.) You should have at least one of your three 8255 ports free and available. Pick 4 convenient 8255 output lines and wire these 8255 output lines to the inputs of the four buffers on an L293D driver IC, pins 2, 7, 10, and 15. The

L293D is a "power buffer" that provides more current than the 8255 can provide. Also, you get to select the output voltage of the L293D . Ground the four ground pins of the L293D. Connect L293D pins 1, 9, and 8 (two enable lines and a logic power input) to 5 volts. Connect the drive power input pin 16 to the +V variable power supply on your kit. By adjusting the variable voltage, you can set the output voltage of the L293D. When an input to the L293D is logic high, the output of that L293D buffer will be at +V. A low input brings the associated L293D output to ground. What else do you need to do? (Hint: BYPASS!)

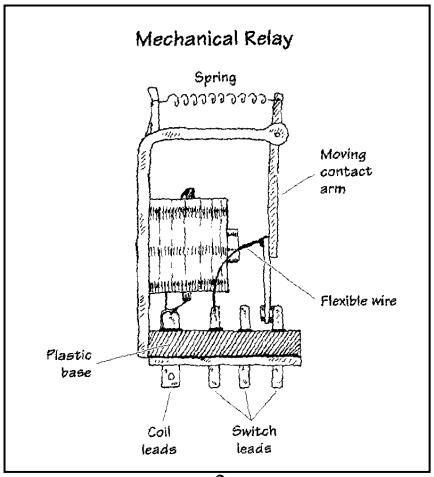
- For now, keep the variable kit power supply voltage limited to 5 volts. Test your system, e.g., use the LM18293/L293D to drive some suitable resistors to make sure that your system works. Again, use MINMON's "R" and "W" commands to write to the 8255 and thereby set the status of the L293D outputs.
- Check to make sure that everything else on the kit still works, e.g., the 8254, DACs, etc.

EXERCISE 2: A simple linear actuator: the relay

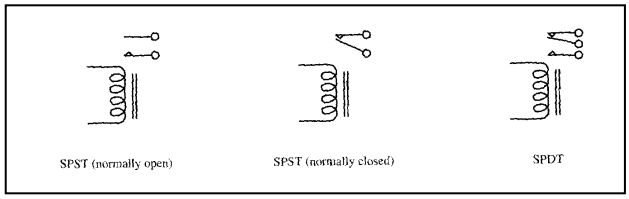
Please find your **relay** and socket in your kit, and put them together, as shown below:




A cartoon of a relay (from "Practical Electronics for Inventors" by Scherz) is shown below:



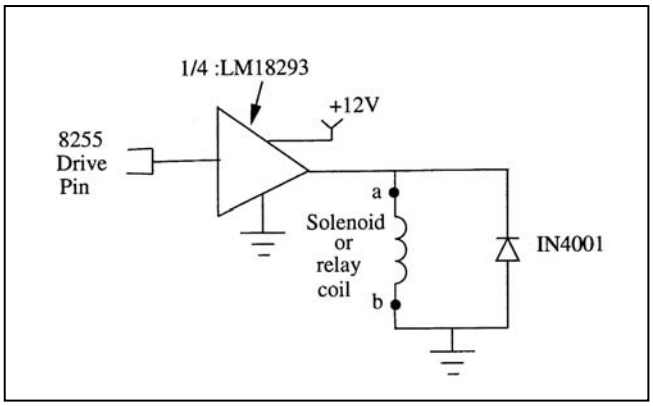
When energized, the electromagnet or coil "pulls" on the moving contact arm, overcoming the force of the spring. This causes the position of the contact on the contact arm to change. A number of possible contact arrangements are shown below:




(Aside: A solenoid is a linear actuator (something that can push or pull). It is generally similar to a relay mechanism, but does not necessarily have any switch contacts. That is, it's just a coil and a sliding rod that can move back and forth. Relays are wonderful electromechanical switches, bidirectional and useful for switching both AC and DC at low frequencies, e.g., on and off for an automobile ignition.)

Please do the following:

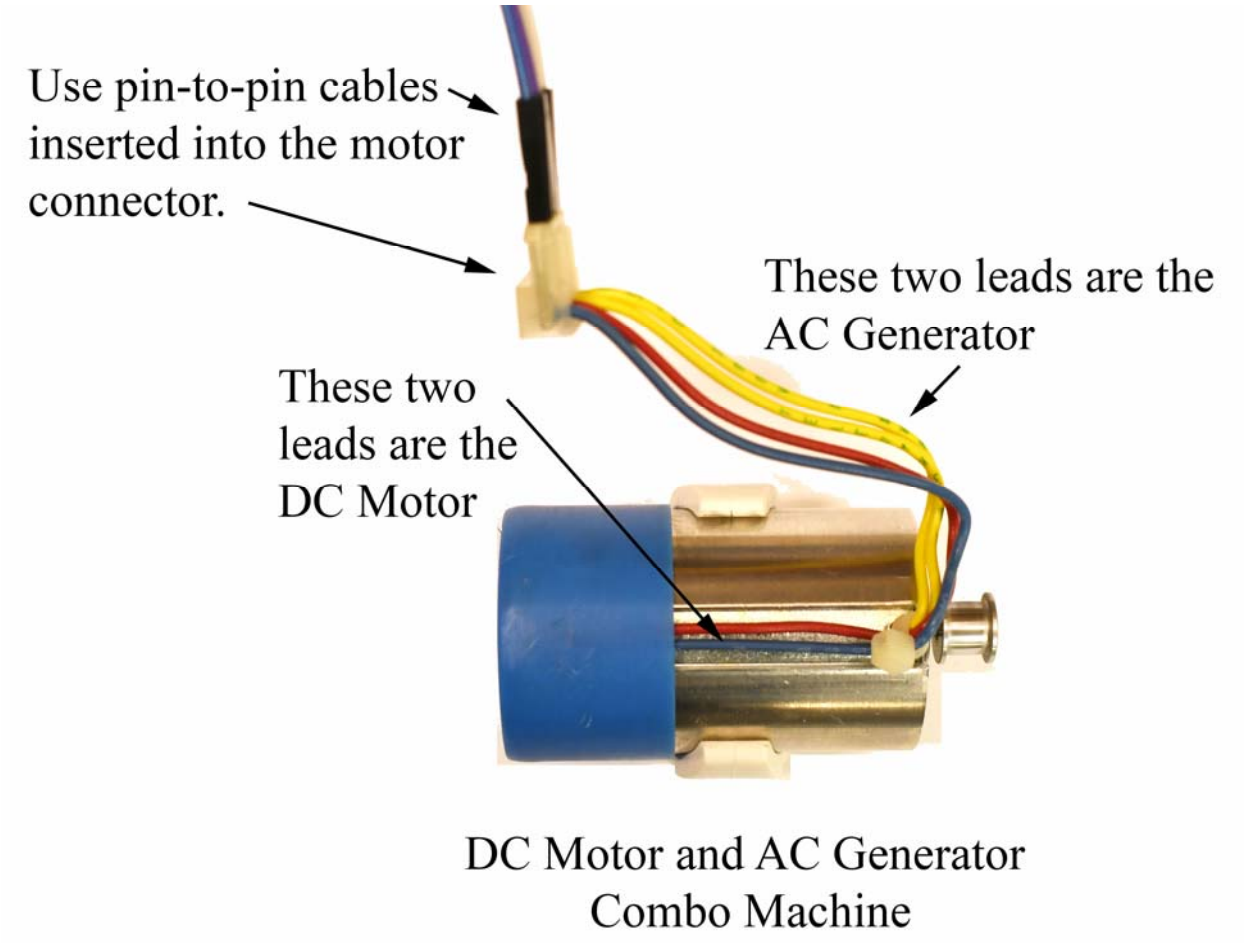
- Determine what type of relay you have (how many contacts, normally open, normally closed, etc.) by physically examining the relay.
- Connect the relay to your R31JP through your 8255/LM18293 combination as shown below:




Explain why the 1N4001 is absolutely essential and cannot be eliminated (!) from this circuit. Run your LM18293 from the +V variable power supply set to 12 volts. Write and test a simple program that turns the LM18293 on and off. You might make the program change the state of the LM18293 based on a key press, or have it toggle the LM18293 every 5 or 10 seconds. Use your multimeter (on the continuity or Ohms setting) to see the contacts of the relay changing connections. Reverse the leads of the relay (a and b), but leave all other connections the same. How does this change the behavior of the relay relative to the drive signal ?

EXERCISE 3: Get to know the DC motor

Please find the DC motor/AC generator combo machine in your kit. You can see a picture of the machine on the next page. This machine is really two machines in one. It has a conventional brushed DC motor, just like we discussed in lecture. Inside the machine, connected to the same shaft, it also spins a magnet disk in a coil; this is the AC generator. The AC generator will make a sine wave as the machine rotates. If you look with an oscilloscope, you will see the sine wave go through 8 full periods for every revolution of the shaft. So, you can use your oscilloscope as a tachometer. If you monitor the AC generator and measure the sine wave frequency in hertz, then that is eight times the actual mechanical rotation frequency in hertz.




DC Motor and AC Generator Combo Machine

Note that you can connect to the terminals of both the DC and AC machines. The connector has four socket holes. Use four pin-to-pin cables from your kit, as shown in the figure above, to make connections, two for the AC generator and two for the DC motor. Be sure you know which pair goes with which machine. If you monitor the AC generator pair with an oscilloscope and spin the shaft with your hand, you should see a sine wave on a properly adjusted scope.

This motor was used in a rotating storage drive for a computer product. The DC motor was controlled by a microcontroller to spin at a constant speed. The microcontroller measured the speed by observing the waveform from the AC generator, adjusting the DC motor input voltage and speed as necessary (feedback!). In this role, the AC generator serves as a "tachometer" or speed sensor.

In this exercise, you will use a kit power supply to directly drive the motor. Do NOT drive the motor using the LM18293. The DC motor does not "care" which way the voltage is applied to it. The polarity simply changes the direction of rotation (a convenient feature for implementing "forward and reverse"). So, a convenient hook-up for this exercise is to drive the DC motor using your "-V" variable supply on top of the kit connected to one DC motor terminal, and kit ground connected to the other DC motor terminal. Turn you –V variable supply all the way to zero volts before you start. Check with a multimeter! Do not wire with the power on. Make sure your motor is mechanically secure while energizing it. Don't leave it on top of the kit or allow it to spin into your microcontroller or other electronics! **As always, use your safety glasses.**

Please do the following:

- Use the "-V" variable kit supply to run the DC motor. We'll call this voltage source the "drive" voltage. Turn the voltage low, i.e., start at 0 volts. Turn the current limit all the way up (let the motor have all the current it can draw). Monitor the AC generator with an oscilloscope, and use the frequency readout measurement feature of the oscilloscope to determine the rotation speed of the shaft. Remember, the AC generator makes 8 sine wave periods for every one rotation of the mechanical shaft. You will need to convert to determine the actual shaft speed from the scope measurement. Gradually increase the motor voltage to 12 volts in steps of one volt, that is, run the motor at 0 volts, then 1 volt, 2 volts, etc. At each voltage step, let the motor settle to a steady-state speed. Record the AC generator voltage. Make a plot of shaft speed (in RPM, rotations per minute) versus drive voltage. Does your plot make sense to you?
- Use your multimeter, set to measure current at a proper setting, to measure the current going into the DC motor. Continue measuring speed by observing the AC generator with your oscilloscope. With the voltage at 12 volts, gently "load" the motor with your fingers (**Be careful, don't pinch yourself on the shaft or blister your finger; touch gently, or skip this step if you are not able to do so.**). Watch the current display. What happens to the current flowing into the motor as you lightly increase the mechanical load on the shaft? What happens to the shaft speed?
- Think about what you saw. Looking at your graph from the first part of Exercise 3, how does the DC motor's terminal voltage affect speed? Terminal **voltage** on the motor appears to roughly influence what mechanical quantity? What mechanical quantity does terminal **current** influence?

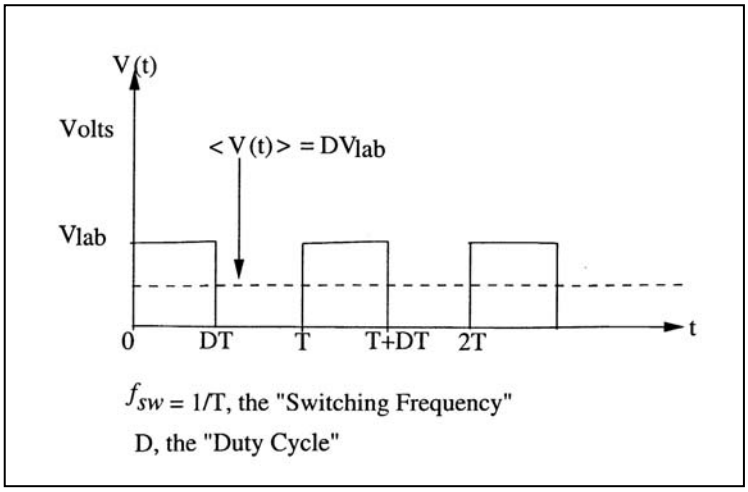
EXERCISE 4: Pulse Width Modulation

Some actuators, like the relay or the solenoid, are "discrete" in the sense that they have a limited number of electromechanical "states" they can assume. For example, the solenoid rod is either "out" or "in". The relay is either "closed" or "open". These actuators are relatively easy to drive. The actuator responds to voltage levels, "highs" or "lows", which are straightforward to make with a microcontroller and appropriate support circuitry like the LM18293/L293D.

Other actuators may be more complicated to drive. As you saw in Exercise 3, the DC motor can generate a potentially continuous range of speeds and shaft torques depending on the electrical drive. The LM18293 is not designed to provide variable output voltage, i.e., it makes a binary power-level output that is either high or low.

There are, however, all sorts of other ways to make a more varied range of drive voltages with your microcontroller. One familiar approach would be to use a DAC. However, our DAC is unable (like most DACs) to directly deliver significant amounts of electrical power. So, the DAC would have to be followed or "buffered" with a power amplifier capable of delivering suitable voltage and current levels to an actuator and capable of following the command voltage signal from the DAC. The DAC and amplifier combination tends to be expensive. Therefore, people have looked for other ways to provide the effect or appearance of variable levels using only "switched" or binary waveforms. We'll examine one such approach in this exercise, called "Pulse Width Modulation" or PWM.

A PWM waveform is a square wave that varies between a high and a low level. For now, let's imagine a waveform varying between zero volts and some positive voltage, Vlab. A typical PWM waveform is shown below:




The waveform repeats periodically every T seconds. For part of this switch period, the waveform is "high". For the rest of the period, the waveform is "low". The fraction of the switch period T for which the waveform is high is a variable called the "duty cycle", denoted by the variable D. The average value of the waveform over a period, indicated by the carets <> in the figure, is D*Vlab. The waveform is said to have a "switching frequency," the inverse of the switch period, T.

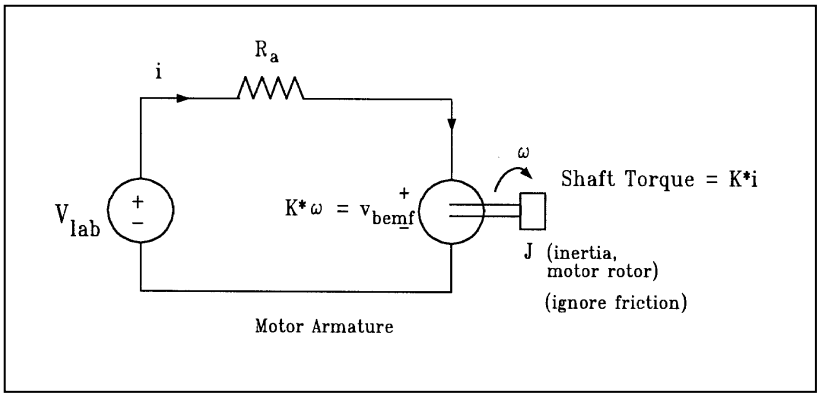
We can sometimes use a PWM waveform to approximate the effects of a variable analog voltage drive. Instead of varying the voltage level, the microprocessor varies the easily controlled duty cycle. The **average** voltage varies smoothly as a function of D. If the load driven by the PWM waveform is "low-pass" in character, it may respond predominantly to the average value of the drive, "ignoring" the high frequency ripple at the switch frequency and beyond. Part of the trick to make this scheme work is that the actuator or load must respond to the average or low frequency part of the waveform while ignoring the high frequency components; hence, careful selection of the switch period T for different applications is essential.

Please do and answer the following:

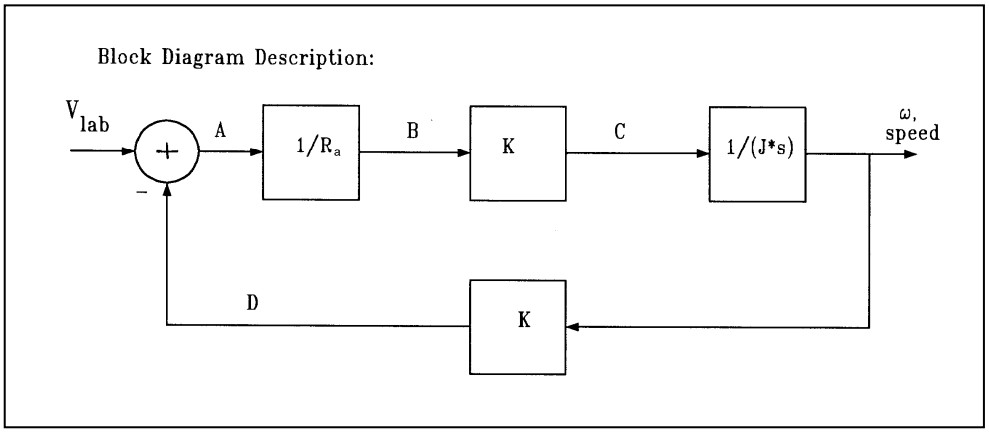
- Write two programs for the R31JP. One should create a PWM waveform at the output of one of your LM18293 channels with a duty cycle of 50% and a switch period of two seconds (0.5 Hz switch frequency). We'll call this the "low frequency" program. Also write a "high frequency" program that makes a PWM waveform with a duty cycle of 50% and a switch period of 0.2 milliseconds (switch frequency of 5000 Hz). Continue using the LM18293 with the "+V" variable kit supply voltage, set to 12 volts for this exercise, and apply the output voltage of your LM18293 to an **LED in series with a current limiting resistor**. Now, drive the LED with the low frequency program, and also with the high frequency program. What do you see in each case? Explain your observations? What is the low-pass filter in this system?
- Remove the LED/resistor combination from the output of the LM18293, and replace it with the motor/tachometer combination you used in Exercise 3. Drive the DC motor with the low and high frequency programs. Capture the AC generator/tachometer voltage waveform (speed) and drive voltage to the motor for each case. How does the motor respond to these two different drive programs? Explain your results. What provides low-pass filtering in this motor system?
- For the high frequency program, note the average tachometer or speed reading. To what drive voltage did this tachometer reading correspond in Exercise 3 (use your graph of tach voltage versus drive voltage)? How does this drive voltage compare with the average voltage applied to your motor in this exercise?

EXERCISE 5: Model the DC motor

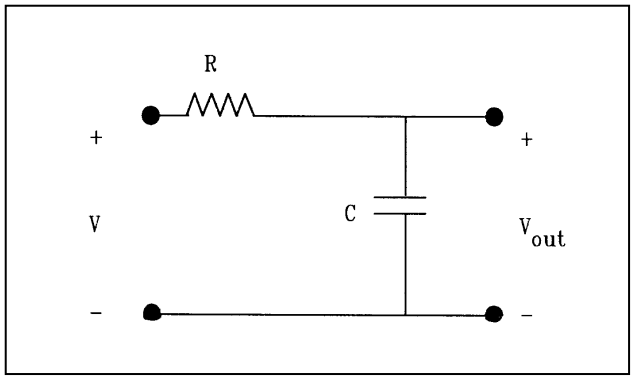
Motors are used to control the position or speed of all sorts of electromechanical devices: antenna in radar systems, wheels on an electric car, a rotating mirror in a grocery store laser scanner. An essential step in the development of a good controller for a motor is the determination of an accurate motor model. In this exercise, we'll think briefly about the model for the DC motor, following from our lecture discussions. A circuit schematic of a permanent magnet DC motor is shown below:




Several assumptions and simplifications have been made. The armature inductance has been ignored, as it is typically small. The motor shaft is connected to a rotating inertia. Friction is ignored. A block diagram that represents this circuit model of the DC motor might be:




In the questions below, we will also think about the following RC circuit, which you will compare to the DC motor:



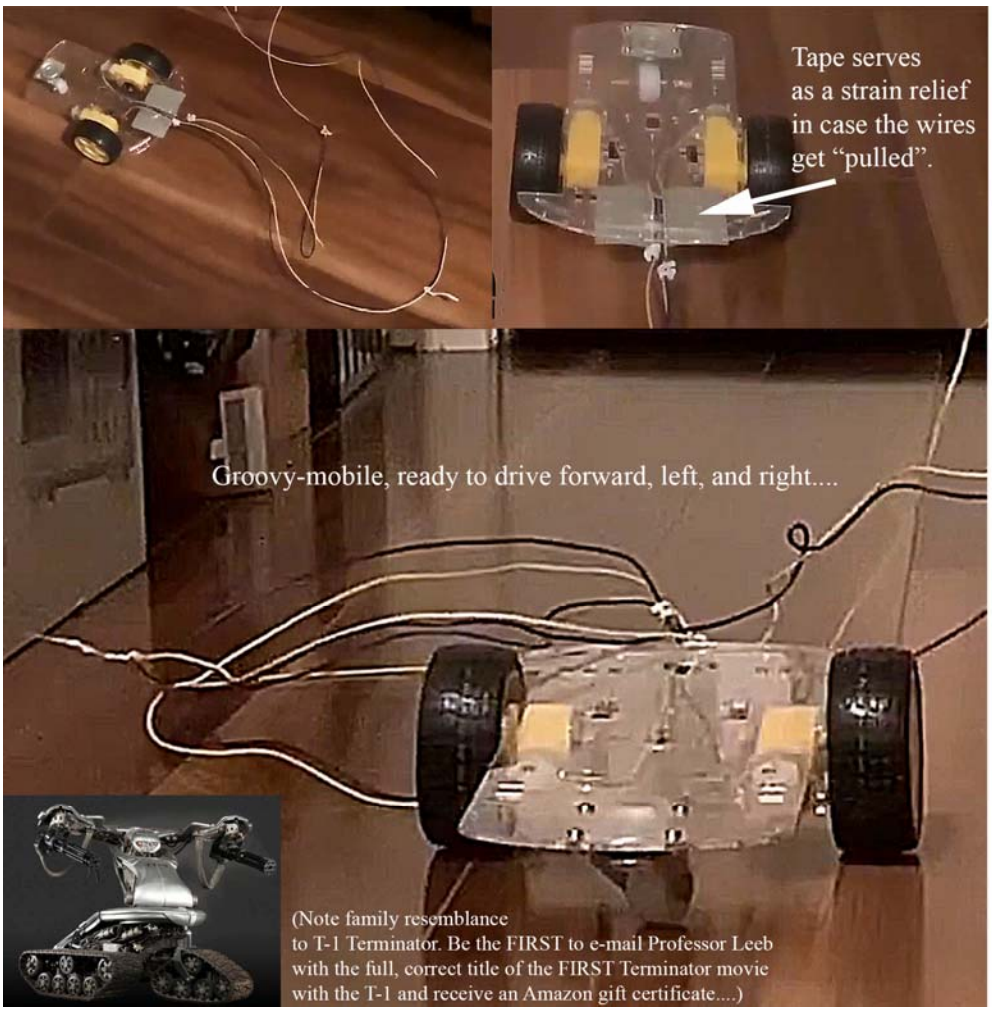
Please consider the following:

- In the block diagram above, determine the simplest description of the "unknown" signals indicated by the letters A, B, C, and D in terms of the DC motor circuit diagram.
- Determine a transfer function or equation that relates the motor shaft speed to the voltage Vlab applied to the DC motor.
- Determine a transfer function or equation that relates the output voltage Vout of the RC circuit to the input voltage V applied to the circuit.
- Compare and contrast the transfer relationships for the DC motor and the RC circuit.

EXERCISE 6: Motorized Drone Cart with PWM

Your mental model of the actuator is the starting point for designing electronics to drive and control your system. Recognize, however, that most actuators are connected to a mechanical system: the turntable in a microwave oven, the platters and head in a hard disk drive, the antenna in a radar unit, the arms or links in a robot arm, etc. A complete system model includes not only the actuators, but also the interconnected mechanical system, and often at least a guess at what disturbances or "unexpected" loads and conditions (like friction, which we ignored previously) might pop up in the field. Let's see how the DC machine behaves in a practical system, a motorized cart.

You will find a small robot cart kit in your box from us. Your box has 4 yellow gear motors for driving wheels; your robot only needs two. Two of the yellow motors have pre-attached wires, the other two do not. It will be easier if you use the two motors with the pre-attached wires. Be careful. Do NOT break the wires off the motors. Please build the car kit so that it can be used as a wire-tethered drone car




This student-built cart has a total of 4 wires, two for each of the two motors. The wires run back to the kit, so each motor can be driven by an L293D. NOTE the careful use of tape to provide a "strain relief" that keeps the motor connections from breaking if the cart pulls on the wires. An even better choice might be to **put a small scrap of breadboard on top of the robot** so that the wiring can be changed or reconnected as needed.

The wire tethers would be better if they had been built as twisted pairs that stayed together. Use **enough** wire so that the wires can reach your kit easily and your cart can drive around in a 6 foot or so circle on the floor. Do not operate your cart on a table top!

**Never drive the motors in the cart with more than 6 volts!**

READ all of the exercise bullets, below, before you start. We will ask you to modify and improve your cart control program in each bullet. You may want to know where the exercise is headed, so that you can plan your code for this entire exercise and avoid unnecessarily complex revisions.

Please do the following:

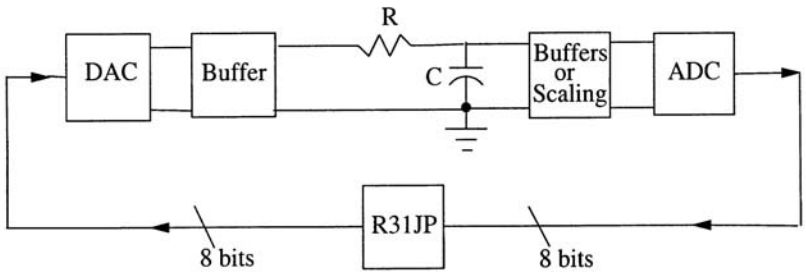
- Connect each of the two motors in the robot cart to an LM18293 drive on your kit. One wire from each motor should go to an LM18293 channel. The other motor wire should go to ground. The hookup is essentially identical to the one we used for the relay, and you should use a 1N4001 flyback diode, just as we did with the relay, for each motor. With this arrangement, you will only be able to run the motors "forward," which will be fine for our purposes. By controlling the two motors independently, you should be able to move forward, turn left, and turn right. Continue to use the +V variable kit supply to power your LM18293 outputs, set for a maximum of 6 volts! Write a short R31JP program that slowly scans through the 2 channels, activating both, then one, then the other. That is, the cart should roll forward, then left, then right.
- Use your keypad to drive the cart. Your program does not have to service multiple simultaneous key presses. That is, you may assume that the user will only move one button at a time. Pressing "1" should cause the cart to move forward in a straight line. Pressing "2" should give a left turn, and pressing "3" should give a right turn. The R31JP should keep moving the cart until the key is released. Use a timer interrupt to scan or poll the control key pad at regular intervals. Drive your cart around a little, carefully. Don't overstretch the wires, and do NOT operate the cart on a tabletop or anywhere where it could fall or bump into something, someone, or a pet.
- Modify your control program to include PWM for the wheel motors. That is, the user should be able to select the duty cycle of a PWM waveform used to drive the motors selected by the keypad. This need not be too fancy. Provide at least three duty cycle choices, D = 0.5, D = 0.75, and D = 1, selected by keypad keys "4", "5", and "6." Use your LCD to display the selected duty cycle. Use a timer interrupt to implement the PWM scheme. Continue to use another timer interrupt to control the polling of the control keypad. If you select the low duty cycle, the cart should be able to go forward, left, or right slowly. If you select a D = 1, the cart should also go forward, left, or right (as selected on the keypad), but relatively quickly. Play with your program – drive the cart, attempting to repeat simple driving patterns. Mark a starting spot on your floor and try driving to a few different end points at different speeds.
- Finally, write a "canned" program that runs the cart at a duty cycle of your choice and drives the cart through a path you select that includes at least one "straight," one "left", and one "right," any order. Pressing "7" on the keypad should cause the cart to run through the pattern and then stop automatically. Run the program, marking the start and end point of the cart. Run the program several times, with the cart always starting from the same start position. How reliable is the cart in getting to the "first end point" that you found? Explain why closed-loop feedback control is generally used in industrial robot arms. (In a closed-loop control, the wheel rotations and distance traveled would be actively sensed, tracked, and corrected using some sort of position sensor or collection of sensors.) Based on your experiments, how easy is it to get the cart to repeat its behavior using our "open loop" (no sensors) approach?

EXERCISE 7: Feedback Control for a First Order Plant

Under certain electrical drive and mechanical loading conditions, we know that the DC motor "looks like" a first order system, i.e., a single pole. In this exercise, you'll develop a feedback loop for a similar first order system, an RC circuit. We'll play with the RC circuit instead of the motor because the RC circuit

will not require a special power amplifier or speed sensor. However, keep in mind that the circuit you are playing with is a good model for many physical systems.

On your lab kit, please construct the physical layout for an R31JP feedback controller for your first order "plant", the RC circuit, shown below:




Use a value of R = 1100 Ohms. The value of C will change in our experiments, but you can start with a value of C = 100 microFarads. **If you use an electrolytic capacitor, be sure to observe the capacitor's polarity when you wire it into your circuit (minus leg to the ground connection).** Design appropriate circuits for the blocks labeled "Buffer" and "Buffers & Scaling". Remember why you need these blocks! The DAC may not be able to drive enough current for your plant, so the "Buffer" block is a mini-power amplifier that limits the load on the DAC. An OP-AMP buffer is a good choice here, ideally selected with a gain of 2 so that the drive to the RC circuit is limited to at most 5 volts. The "Buffers or Scaling" block serves several functions. First, we do not want to "load" the output of the RC plant with any additional circuitry, so the input to this block should appear to be a high impedance. Second, your DAC may produce more voltage than is safe for the ADC input. In this case, the "Buffers or Scaling" block needs to include an appropriate gain to reduce the output voltage of the plant.

In this exercise, you'll write a closed-loop feedback control program for the first order plant. We'll try different values for the plant time constant and control gain in order to understand how these quantities affect the performance of the controller. We are interested in the stability of the loop, steady state error, and transient performance.

Please do the following:

- Make sure to include a neat, complete circuit schematic of your feedback loop in your lab report, with values you selected for components in the "Buffer" and "Buffers or Scaling" blocks.
- Write a closed-loop feedback control program for your R31JP. Your goal is to regulate the output of the plant, which is the capacitor voltage. Your program should compute the difference between the desired capacitor voltage (reference) and the actual, sampled capacitor voltage. The reference is a number between 0 and 255 in your program that corresponds to a desired output voltage (as measured at the ADC) between 0 and 5 volts. Let's call this difference the "error". Multiply the error times a positive number or "gain" in your program. This product, the "command", should be written to the DAC as the drive voltage for the plant. Your program should include careful range checking. If you do not check values carefully **BEFORE** computing the error and then the command, your program will perform strangely. For example, if the plant output voltage is greater than the reference, the error would be negative. Since you cannot drive a negative voltage using your DAC as we've asked you to configure your circuitry, you should simply limit the command to be zero in the event of a negative error. Also, if either the gain or the error is very large in magnitude, the product of the gain times the error could be larger than 255. Since 255 is the largest command you can issue to the DAC, you should limit or "saturate" the command in the event of large gain/error products to be no more than

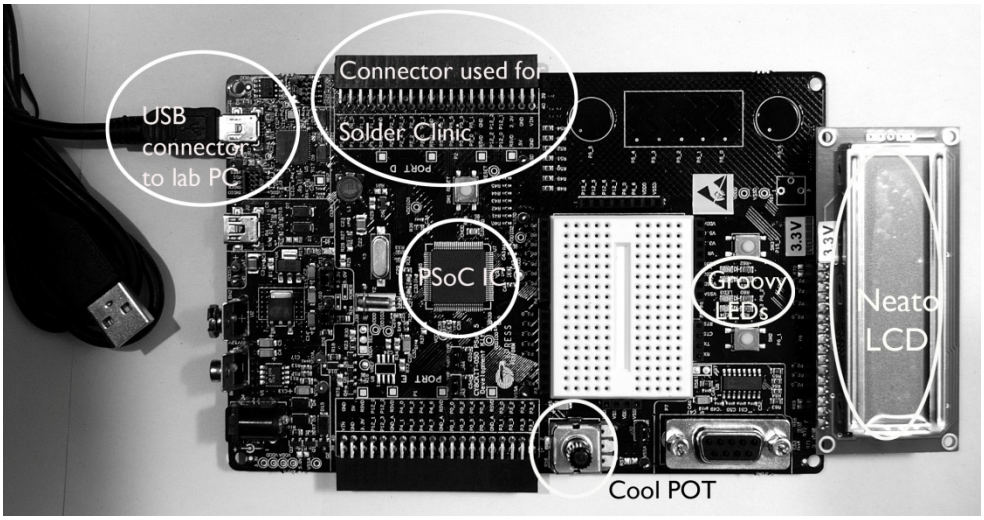
255. Develop your program using an 8051 timer and interrupt to achieve a sample rate of approximately 125 Hz. That is, every 1/125$^{th}$ of a second, you should sample the ADC, compute the error, multiply by the gain, produce an acceptable command limited to the range 0-255, and write this command to the DAC. Your program should switch between a reference of 0 and a reference of 128 approximately every 2 seconds. That is, you should attempt to regulate the output to 0 volts for 2 seconds, then switch and attempt to regulate the output to 2.5 volts (as measured at the ADC) for 2 seconds, and then repeat this process indefinitely. Get your program working with a gain K = 2 and C = 100 microFarads. Use your LCD screen to display your current reference and control gain. Use an oscilloscope to observe the drive voltage (output of the "Buffer" block) and the output voltage (at the input to the ADC). Sketch your observations, and do the following:

1. Observe the output voltage (input to the ADC) carefully. You will notice that the "rise time" when the reference switches to 128 is different from the "fall time" when the reference switches back to 0. Calculate the apparent time constants of the system, and explain the difference in rise and fall time.
2. When the reference switches to 128, your capacitor voltage should eventually settle to some "steady state" voltage level. What exactly is this level, as measured at the input to the ADC? What did you expect this level to be? Explain any observed differences quantitatively.
3. Repeat tasks 1 and 2 with C = 33 microFarads and again with C = 1 microFarad. Explain your results. How could the same observations you made with C = 33 microFarads and also with C = 1 microFarad have been seen qualitatively using only C = 100 microFarads? Explain qualitatively what factors limit the lower and upper sample rates appropriate for a digital controller.

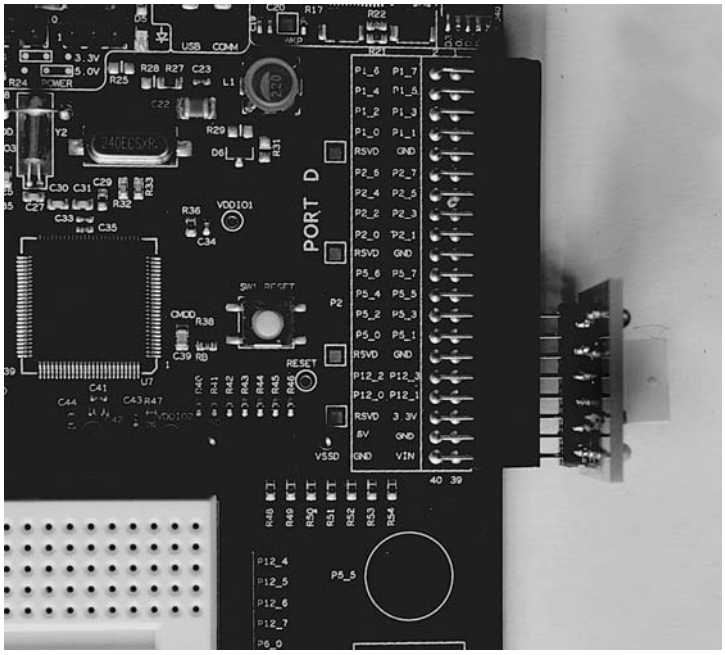
- Make sure that your program can be quickly reconfigured for gains in the range from 2 to 30. We may ask to see the performance of your program with other gains during check-off.

EXERCISE 8: But can I do it with PSoC?

YES! The LCD, ADC, and DAC tools are easy to use on PSoC. The PSoC has some astonishing abilities to support analog measurements and computations. Without going overboard redoing everything, let's scratch the surface and take a look at analog-to-digital conversion and techniques for creating the illusion of a "continuum" of values, e.g., for an LED. For this Exercise 8, use your **PSoC 5LP "Big Board"**:



Make sure that you have your Digit LED board correctly positioned and fully inserted before using the evaluation board. This picture shows the LED board in "mid-insertion":




Please do the following:

1. Go to the "PSoC" page on the 6.115 course website: http://web.mit.edu/6.115/www/page/psoc-information.html
2. Now, from the 6.115 course website, download the "Exercise 2" for the PSoC (on the webpage under the "Blinky" project). Complete Exercise 2 by following the instructions on the file page with a ".cysch" extension for Exercise 2. You now know how to use the LCD for displaying text, and how to use an ADC to read an analog voltage! Include pictures and documentation of your code and TopDesign in your lab report.
3. Now, from the 6.115 course website, download the "Exercise 3" for the PSoC (on the webpage under the "Blinky" project). Complete Exercise 3 by following the instructions on the file page with a ".cysch" extension for Exercise 2. Include pictures and documentation of your code and TopDesign in your lab report.
4. Now, create your own program using your skills and tools you have just seen. For this program, make use of the "cool pot," the Digit LED display, the "Neato LCD" display," and the two "user buttons" on the "Big Board" that are located on either side of the "groovy LEDs". Here's what your program should do: Following a reset (using the reset button near the "PSoC IC" on the "Big Board"), your user (you in this case) should be able to turn the pot. As the pot rotates, the pot position should select one of at least 8 different letters or numbers (your choice) on the LCD display. The LCD display should show the character in a fixed location, changing the character as the pot rotates. When the user presses one of the "user buttons," let's call it "button 1," the PSoC should record the letter or number that was chosen, and display it on the Digit LED Display. The next letter chosen by the pot rotation should be shown in the "next" location on the LCD, so that a running record forms on the LCD display of the user choices. That is, the user should be able to repeat the pot-and-press process, storing several letters (you pick a maximum, at least five) shown sequentially on the LCD display as a historical record, with the most recent selection shown on the Digit LED Display. When the user presses the "other" user button, let's call it "button 2", the PSoC should stop accepting letter entry from the pot, and instead display the

sequence of letters that was entered, one at a time, on the Digit LED Display. Each letter should be displayed for one second, and the "message" should repeat indefinitely, cycling along on the Digit LED Display. A "reset" should start the whole process over again. (You may find looking at Exercise 1 again, under "Blinky", to be helpful....). READ the datasheets for the PSoC LCD display, and any other needed components, to learn about the API commands you will want in order to do things like manipulate the LCD display. Include pictures and documentation of your code and TopDesign in your lab report.

5. NOTE: You do not have to do this for this exercise, but be aware that the PSoC Big Board offers capacitive-sense or "Capsense" buttons and software support for using them. You could, for example, for your final project, use the slider and "no click" Capsense buttons to implement useful controls.

IF YOU ARE **KOVID KAPABLE**: Please complete Exercise 9 in the 38-600 lab. This exercise will teach you how another actuator, a stepper motor, works. Exercise 10 is optional – a successful solution will earn you two bonus points.

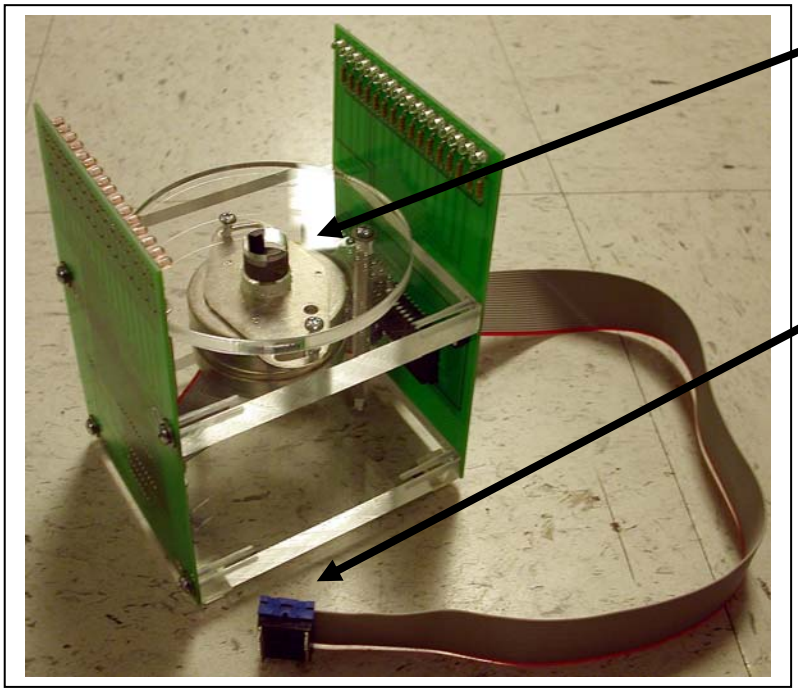
EXERCISE 9: Stepper motors

We've spent a lot of time with the DC motor. There are many other types of machines, which require different drive waveforms to make them turn. Induction machines and synchronous machines are two classes of motors that require AC or time-varying waveforms in order to turn. The precise nature of the motion produced by these machines is intimately coupled to the frequency, amplitude, and waveshape of the drive voltages and currents. We don't have time to study all of these machines carefully in 6.115, but we can pick one: the stepper motor. The stepper motor is a type of synchronous machine. This means that the electrical excitation frequency of the motor is an integer multiple of the mechanical rotation frequency of the shaft. Recognize the distinction between the stepper motor and a DC motor. A DC motor could be driven with a "flat", zero hertz DC waveform of a certain voltage, and the motor would turn at a (possibly large) frequency controlled by the voltage level. A stepper motor must be driven by time-varying waveforms, usually square waves. The frequency of this drive controls the rotation of the shaft. The stepper motor is attractive because it provides an inherent position control capability. The drive waveforms can be chosen to cause the rotor to move a specific number of "steps," each step corresponding to a certain number of degrees around a circle. The manufacturer determines the number of degrees per step when the motor is made.

We will explore the use of a stepper motor in a new lab station called "SpinDude". You will find SpinDudes attached to the benches in the 6.115 laboratory. SpinDude can be used to make a "tomographic" image of solid objects. Specifically, it can reconstruct the two-dimensional cross-section of an object or collection of objects from one-dimensional shadows. SpinDude illustrates the idea employed in medical imaging machines like MRI's and CAT scanners.

**Please treat them gently and with the utmost care!**

Complete schematics of SpinDude's phototransistor and LED boards are appended to the end of this lab. In this exercise, we will work first with the stepper motor in SpinDude. Then, we will use both the stepper motor and also the optical components in SpinDude to make images. SpinDude contains a stepper motor that rotates a clear plexiglass turntable. The stepper motor turns 15 degrees for each step. Our goal is to be able to make the turntable rotate through one full rotation of 360 degrees. That is, we seek to make the stepper execute 24 steps. A photograph of SpinDude is shown below:



SpinDude has a 12-volt stepper motor that turns a clear plastic turntable.

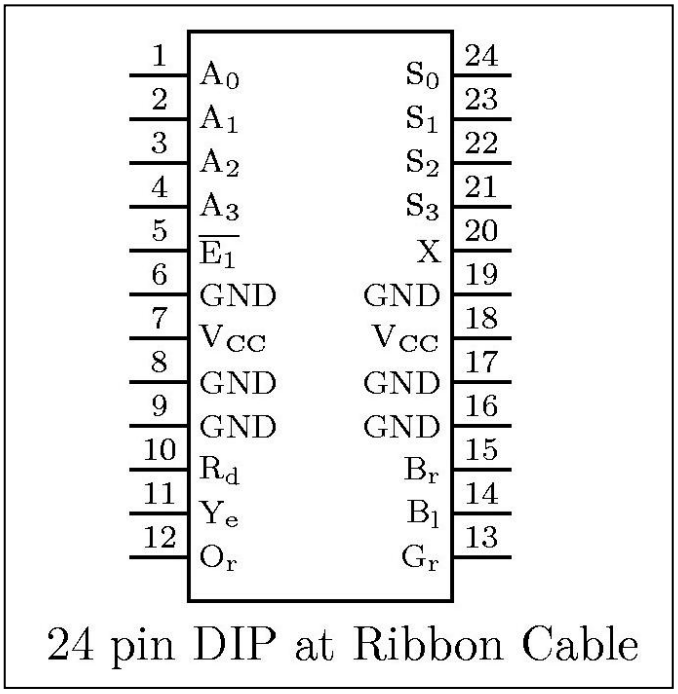
You will make all of your electrical connections to SpinDude through the 24-pin DIP connector on the end of this ribbon cable.

Please carefully examine the blue 24-pin DIP connector at the end of the SpinDude cable. You will notice that, on the back of the connector, there are numbers identifying each pin. The DIP connector can be plugged into your lab kit like any other 24-pin integrated circuit. Please make sure that you have found the pin numbers and understand them. If you apply stepper motor drive voltages to the optical components, you will destroy SpinDude. Be sure that you are making your connections correctly.

The pinout diagram at the right shows you which pins connect to which parts of SpinDude. For now, ignore everything except for pins 10, 11, 12, 13, 14, and 15. These six pins are connected to the stepper motor.

**Use NO OTHER pins for this laboratory exercise. Use ONLY the six pins 10-15.**

The pin numbers in the figure at the right correspond to the pin numbers on the blue 24-pin DIP connector at the end of the SpinDude ribbon cable.



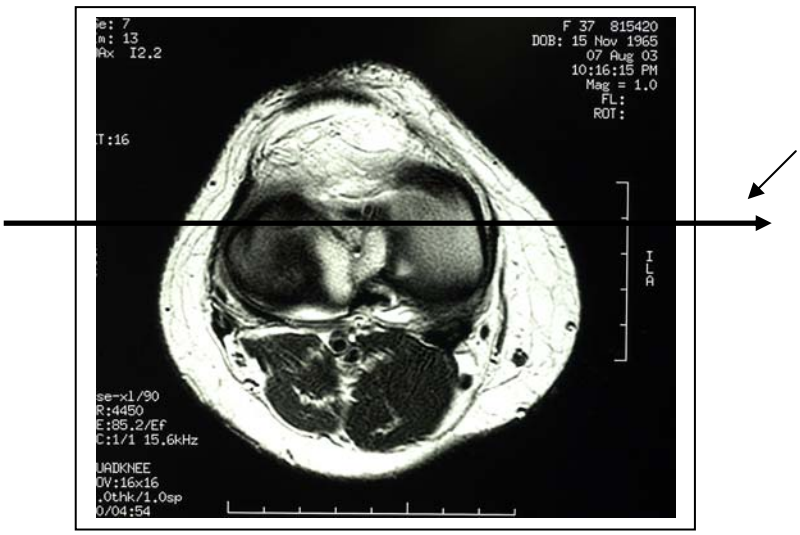
24 pin DIP at Ribbon Cable

Please make sure you've read the pages in Scherz covering stepper motors and then do the following:

- Reason out what type of stepper motor must be contained in SpinDude (unipolar, bipolar, etc.). Use a multimeter.
- Once you have determined the type of motor, connect it using pins 10-15 on the SpinDude connector to the necessary number of LM18293 channels on your kit, and to power and ground as needed. Run the LM18293's from a lab power supply set to 10 volts and current limited to 1 amp. Make sure and include adequate protection, i.e., 1N4001 freewheeling diodes and bypass capacitors! Also, make sure that your connection will support rotation in either direction.
- Write a program for the R31JP that makes the stepper turn. Your program should wait to receive a serial character from Hyperterminal over the R31JP serial port. When the R31JP receives the character, it should turn the SpinDude turntable through exactly 24 steps, waiting approximately one second between steps. It should not matter what character is transmitted, any character (space bar, letter, etc.) should make the program start turning the table. A single transmitted character – one key press on the personal computer – should cause SpinDude to turn through 24 steps, completing a full circle. Make it possible to change the delay between steps. Your delay between steps should be able to range between a tenth of a second and three seconds. Watch the motor carefully, don't let it or the LM18293's get too hot. If anything seems warm, check for mistakes. Also, lower the lab supply voltage to the lowest voltage that will permit the motor to turn.
- Compare the stepper and DC motors qualitatively. Why might one be preferred over the other, depending on the application?

FOR 2 BONUS POINTS: EXERCISE 10: Image Reconstruction with SpinDude (Don't panic, it's easy.)

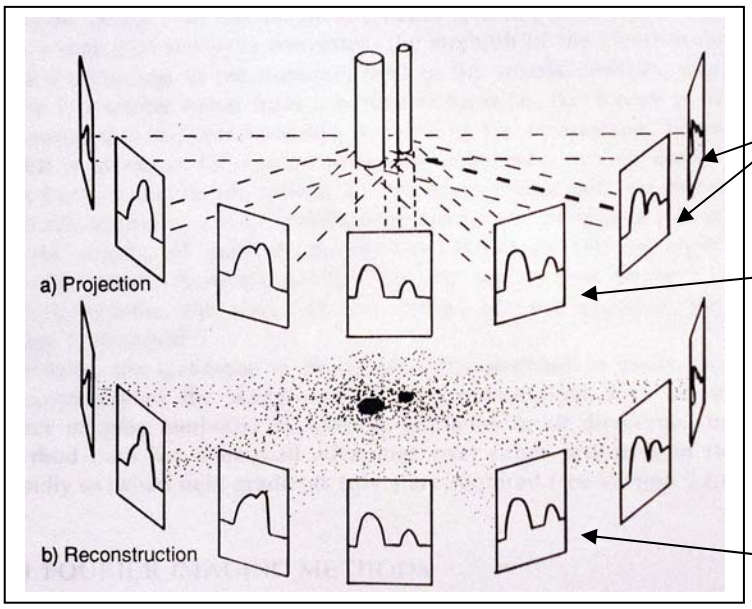
Now, let's use SpinDude to make a two-dimensional (cross-sectional) image of some objects from one-dimensional shadow projections. This is analogous to the process used by medical imaging machines to make two- and three-dimensional images of internal organs, bones, etc. For example, an image of a human knee is shown below:




To make a "tomographic" reconstruction, that is, a two-dimensional, cross-sectional view of this human knee (looking down on the patella), we shine one-dimensional beams of a penetrating radiation (e.g., an X-ray) or electromagnetic field through the knee at many different rotation angles in the plane of the desired image.

The "Radon" transform algorithm reconstructs the two-dimensional image given many one-dimensional "shadows" from the penetrating beams.

The cartoon below illustrates the basic idea behind the imaging algorithm:



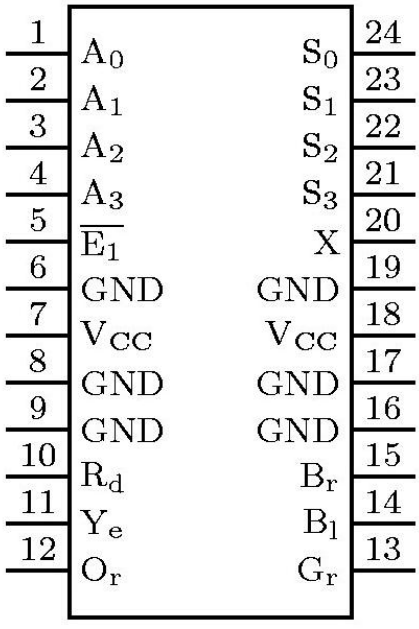

During the **projection** phase, a planar beam of light illuminates the dowels or cylinders in the center of the imaging plane. The photoreceptors are shown as squares that could be digital imaging chips like CCDs or photographic film. The image seen by the imaging element might look like this:

The 1-D image is dark where the dowels occlude the light beam. There is a dark region for the big dowel, and another for the small dowel. The graphs shown in a) to the left indicate high "amplitudes" for dark regions and low amplitudes for light regions.

**Reconstruction** of the image happens when we "backproject" light according to the observed pattern for each photoreceptor. We shine bright light where the shadows were dark. The sum of all the beams makes the brightest spots at the location of the dowels.

In this lab exercise we will gently and reversibly secure some dowels to the SpinDude turntable. These dowels must be tall enough to rise up into the imaging plane formed by the LEDs and phototransistors on SpinDude's two printed circuit board "walls." Your job is to create a two-dimensional cross-sectional image of the dowels in the SpinDude imaging plane. You will write an R31JP program that "runs" SpinDude, rotating the turntable, activating the LEDs, and collecting shadows from the phototransistors. You will provide this shadow information to a Matlab script that we will give you. The Matlab script will perform the reconstruction and show you an image of the dowels in the imaging plane. To gather the shadow data, you will need to use the other pins on the 24-pin SpinDude ribbon connector, shown again below for your convenience (this is the 24 pin DIP end of the cable that goes on your kit):

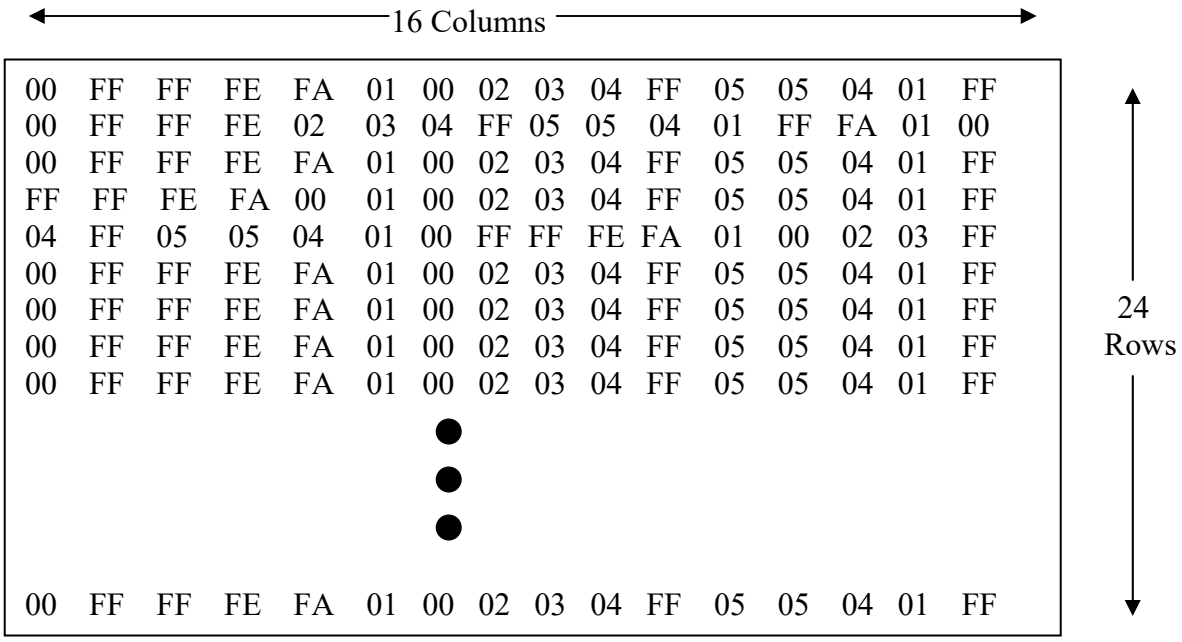
- Please leave pins 10 – 15 connected to your stepper driver circuit.
- GROUND pins 6,8,9, 16, 17, and 19.
- Please connect pins 7 and 18 to plus five volts on your lab kit.
- Pins 1 – 4 form a binary nibble that selects one of SpinDude's 16 LEDs for illumination.
- Similarly, pins 21 – 24 select one of SpinDude's 16 matching phototransistor channels, used to read the transmitted light from the chosen LED.
- Asserting pin 5 by bringing it low causes the selected LED to light up.
- Pin 20 provides an analog voltage between zero and five volts from the selected phototransistor, indicating how much light is received. You'll get near zero volts when the phototransistor is dark, and near five volts when it is illuminated.




24 pin DIP at Ribbon Cable

Please do the following:

- Write a program to collect all shadows at all rotations. Start with the stepper motor program you wrote to rotate SpinDude's turntable. In response to the reception of a single serial character (any character should do the trick, don't check for a specific character), your program should rotate the turntable 15 degrees. Next, it should illuminate each LED in turn. For each LED, your program should use your ADC to read the illumination intensity on signal X, pin 20 of SpinDude's ribbon cable (the 24-pin DIP connector on your kit). Send this illumination value back to Hyperterminal as a hex character (a number between 00h and FFh) followed by a space. When you have illuminated each LED and read each associated phototransistor in turn, send a carriage return and line feed back to Hyperterminal. You should then rotate the turntable another 15 degrees and scan the LED/Phototransistor pairs again. Do this a total of 24 times, for each 15-degree position of the turntable between 0 and 360 degrees. Be sure to leave enough time at each rotation to allow the turntable to stop "jiggling" and settle into position. Also be sure to leave the LED illuminated long enough at each step to allow the phototransistor output to settle. Use an oscilloscope to check this on pin 20 if you're not sure how long to wait. You should be able to complete the entire 360 degree scan in no more than a couple of minutes. Notice that you could use the same four digital output lines to drive both the LED select pins 1—4 and also the phototransistor select pins 21—24. In response to hitting spacebar in Hyperterminal, your R31JP should send back a 16 column by 24 row matrix of hex numbers that looks something like this (the details of this data are meaningless, it just shows the structure of what you should see in your Hyperterminal window):




  Examine your data for a test case with dowels on SpinDude's turntable. If you think about the data and the objects carefully, you should be able to verify that your program is performing correctly.

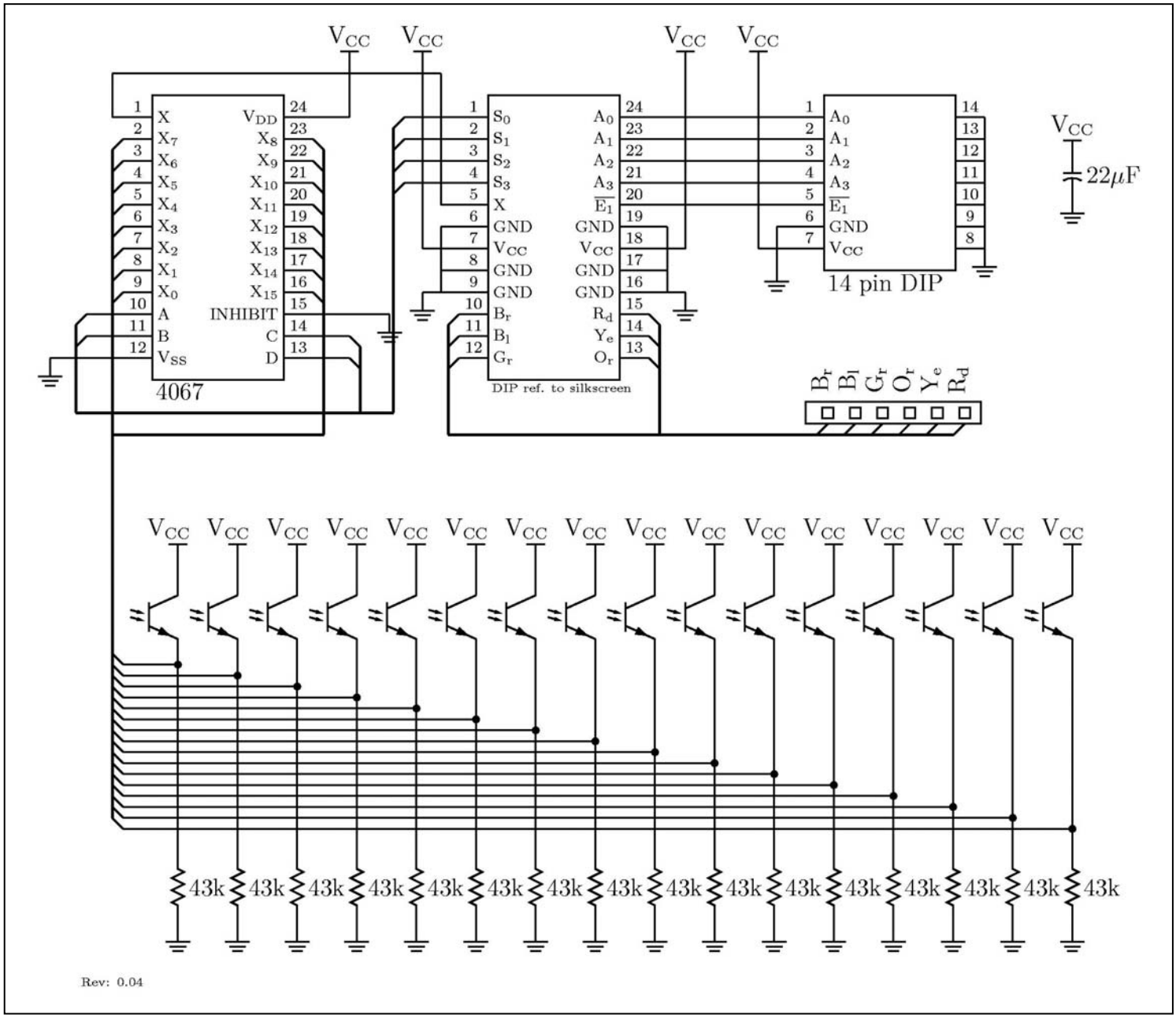
- We have written a Matlab script for you that will perform the image reconstruction algorithm. If your program works and you can see the 16 by 24 matrix in Hyperterminal, you are ready to have Matlab make an image out of your data. Exit Hyperterminal and run Matlab on the 6.115 personal computer you are using. At the Matlab command prompt, type:

```
>> spin_dude('a')
```

and hit return. Matlab will send a character to your R31JP to start your imaging program, collect the 16 by 24 data matrix, and use the backprojection algorithm to create a cross-sectional image of the objects on your SpinDude turntable.

- Think about how you might improve the image created by this Matlab program. What enhancements could you provide to give additional, useful data from your R31JP program? How would you make a three-dimensional scan of an object? This could be a good source of final project ideas.

**SpinDude Photodetector Board:**
(NOTE: The "DIP Reference" on the photodetector board schematic below is the mirror image of the 24 pin dip connector cable that goes to you kit. Use the pinout provided above in the lab for the dip connector cable when wiring SpinDude to your kit.)



**SpinDude LED Board:**