ME 3210 Mechatronics II Laboratory

Lab 1: Stepper and Servo Motors

Introduction

In this laboratory exercise, you will explore some of the properties of stepper and servo motors. These actuators are very useful when you desire a particular angular position from a motor output. They are relatively easy to control, and can provide interesting capability to any design. You may find them useful in the design of the robot for your project.

Brief Background

A stepper motor is a wound synchronous motor capable of rotating in discrete steps. A permanently magnetized rotor sits within an envelope of stator windings, as seen in Figure 1. When current is introduced to the windings, a magnetic field develops according to Faraday's law. The poles of the magnet attached to the rotor line up with the induced magnetic field, resulting in a net torque on the rotor. Stepper motors would not be very interesting or useful if their rotors could only assume one position and stay there when the current is switched on. To improve their usefulness, an external circuit is used to switch the current between the windings in a circular pattern to create a rotating magnetic field. The rotor constantly realigns itself to the rotating magnetic field resulting in a net speed proportional to the switching rate. Most stepper motors have two sets of windings or phases, but some have three.

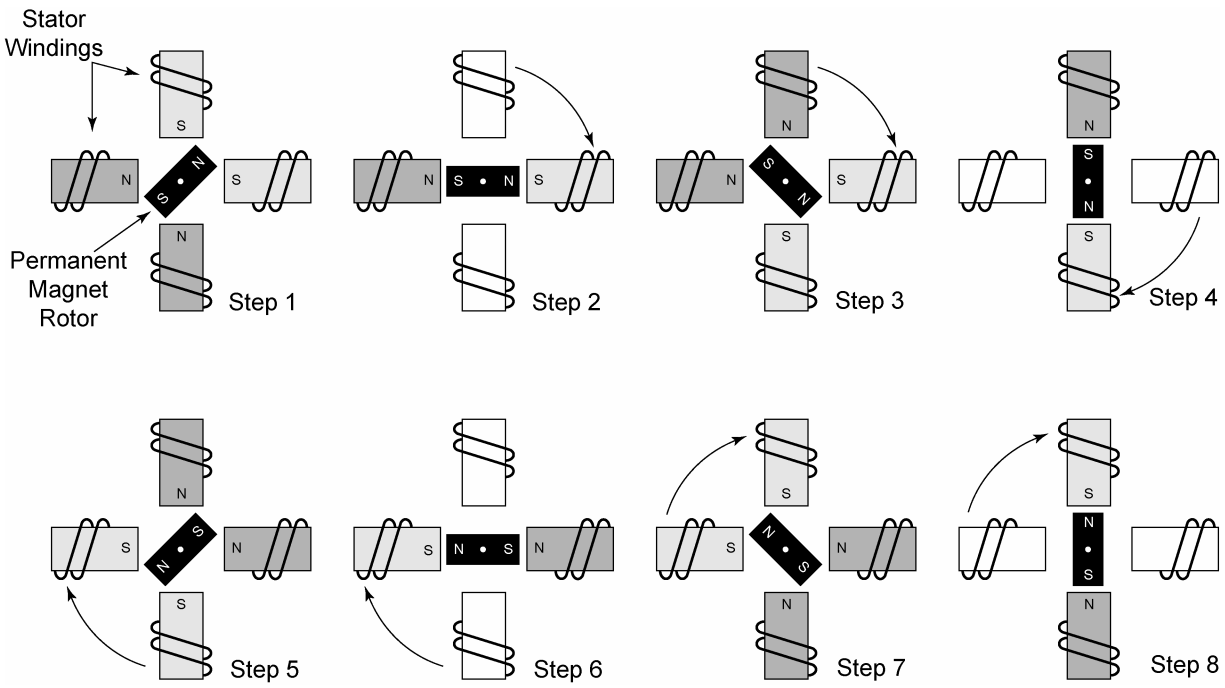

Figure 1: Schematic of a stepper motor with two sets of windings and a single pole rotor taking half steps.

The stepper motor of Figure 1 does not have a very desirable angular resolution since the rotor has only one pole pair; each full-step will rotate the rotor 90°. To increase angular resolution, the rotors of most stepper motors have many more pole pairs. The addition of these pole pairs reduces the angle the rotor turns each step, thus improving the angular resolution of the stepper motor. The angular resolution of a stepper motor in full-step mode is defined according to the following equation:

$$\Delta\theta = \frac{360°}{m \cdot N} \tag{1}$$

where $\Delta\theta$ is the resolution, m is the number of stator windings, and N is the number of rotor teeth (twice the number of pole pairs). Equation 1 applies only for full-step mode; the half-stepping doubles the full-step resolution. Note that each pole pair has two teeth. Thus, for the stepper motor depicted in Figure 1, $m = 2$, $N = 2$ and the full-step and half-step resolutions of the stepper motor are 90° and 45° respectively.

The motor used in this laboratory exercise is a two-phase stepper motor. An external circuit using the Motorola MC3479P stepper motor driver chip drives the stepper motors in this laboratory exercise. This IC chip requires a TTL input clock signal (i.e. square wave whose amplitude switches between zero and five volts) and produces control signals that excite the two stator windings of the stepper motor. Stepping configuration and rotation direction are controlled by setting pins nine and ten, respectively, to either zero or five volts. The individual stepper motor coils are connected between pins two and three and fourteen and fifteen, respectively. For more detailed information on this circuit, please refer to the Motorola MC3479P data sheet by following the link in the "Lab Handouts" page of the class website.

Servo Motors

Servo motors are another useful motor to use in situations where angular position is critical. Servomotors are DC motors whose output shafts are coupled to a potentiometer that provides position feedback to a control circuit. The control circuitry drives the motor to the angle specified by the input signal until the difference between the potentiometer signal and the input reference is zero. This control circuit guarantees that the output shaft of the servo will achieve the desired angular position.

The servos used in this lab are the same servos used in radio controlled airplanes and cars. These servos require a pulse width modulated (PWM) input signal that communicates the angle command to the servo's control circuit. A PWM signal is a special kind of square wave characterized by a repeated pulse, the duration of which is less than the period of the signal, T (see Figure 2). The term "pulse width modulation" refers to the fact that the width of the signal is variable while the frequency of the signal is held constant. In the case of the servos used in lab, the width of the pulse determines the angle of the servo output.

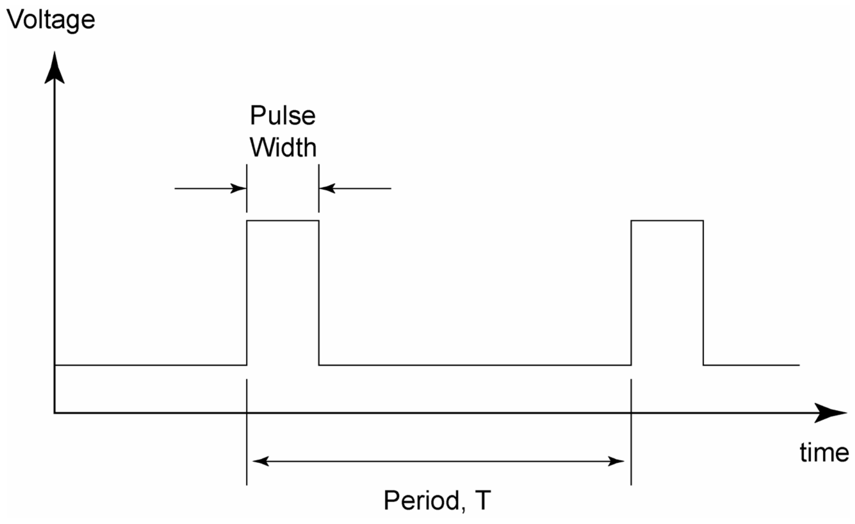

Figure 2: PWM signal

Interfacing and controlling the servo with the Handy Board is very simple. Each servo has three color-coded input wires: the white wire for the PWM signal, the red wire for a regulated 4–6V input, and the black wire is grounded. To connect the servo to the Handy Board simply connect the red wire to the five volt power bus, the black wire to the ground bus, and the white wire to either digital input nine or timer output three (TOC3) (see port TOC3 on the schematic located on page 58 of the Handy Board Technical Reference).

Once the servo is connected to the Handy Board, download the binary source file(s) appropriate for the connections of the servo(s): servo_a7.icb controls the digital nine output, and servo_a5.icb controls TOC3. These files add built-in functions and variables to the Handy Board operating system that allow you to control up to two servo motors. The two control commands of interest are `int servo_a7_init(int enable)` and `int servo_a7_pulse`. These commands control the servo attached to digital nine. To control a servo using the TOC3 output, change the 7 to a 5 in these filenames. The command `servo_a7_init(1)` enables the control of the servo connected to digital input nine: `servo_a7_init(0)` disables it. Once the servo is initialized, redefining the value of the global variable `servo_a7_pulse` changes the position of the servo output (`servo_a7_pulse = 150` for example). This variable is an integer that defines the pulse width (duration) of the PWM output in clock ticks of the Handy Board's processor. Since there the variation between the maximum and minimum pulse width for is different for each servo, determining these limits experimentally is recommended.

Laboratory Exercise

Required Materials/Equipment

- An electronic breadboard
- A Motorola MC3479P motor driver chip
- A stepper motor
- Solid core wire
- Resistors
- Banana cables
- Alligator clips
- BNC to banana connectors
- BNC cables
- Wire cutters/strippers
- A Handy Board
- An RC Servo
- Digital multimeter
- stepper_motor.vi

Stepper Motors

1. Build the stepper motor control circuit shown in Figure 3.

2. Connect pin two to the top left banana port of the stepper motor, pin three to the top right banana port, pin fifteen to the bottom left banana port, and pin fourteen to the bottom right banana port.

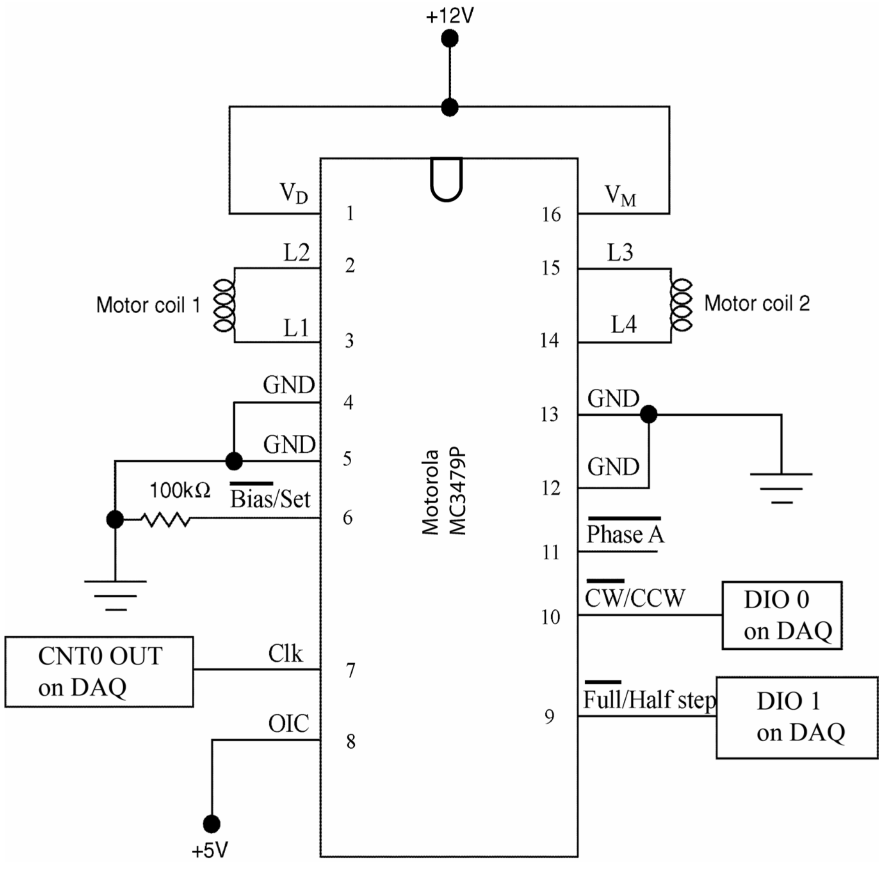

Figure 3: Pin out schematic for the driver chip.

3. Using a piece of solid core wire, connect the clock pin (pin 7) to CTR0 OUT on the DAQ. CTR0 OUT is located on the green block of digital I/O ports. Press in the orange button below the port using a pen or small screwdriver and insert the wire into the hole. In a similar manner, connect DIO 0 to pin 10 to control direction of rotation and DIO 1 to pin 9 to switch between full and half steps. Connect DGND2 to the common ground of the circuit.

4. Connect CTR1 OUT on the DAQ to PFI9, also on the DAQ, with a piece of solid core wire.

5. Open the stepper_motor.vi from the LabView folder on the computer desktop.

6. Set the direction switch to full steps, and enter 1 for number of steps and 3 Hz for the step frequency. Click the white run arrow to initiate the program, and press the 'Send Pulse Train' button to send a clock pulse to the driver chip. The motor should execute one step and stop.

7. Using the paper compass on the face of the stepper, record the angle traversed in one full step. If you find it difficult to measure a single step, you can command more steps and divide to find the angle of one step.

   Full Step Angle = __________ degrees

8. Now set the direction switch to half steps. Using the angle traversed in one full step, calculate how many half steps are required to rotate 120°. Enter that number of steps into the LabView program and send the pulse train. How many degrees did the motor actually rotate?

   Number of Half Steps = _____________ Total Rotation = _____________ degrees

9. Change the frequency to 500 Hz. Rerun the program with the same amount of half steps (be careful to release the button quickly or the command will be executed more than once). How far did the motor rotate this time? If the rotation was the same, increase the frequency by 50 Hz and repeat until you notice a difference.

   Total Rotation at 500 Hz = _____________ degrees

   Why is there an angle difference from the 3 Hz frequency?

10. Determine the number of rotor pole pairs of your stepper motor by using the angle you measured in step 7 and Equation 1 to solve for the number of rotor teeth, *N*, and dividing by two. Record the result in the space below:

    Number of pole pairs = _________________

11. Determine what input frequency makes the stepper motor stall by entering 100 as the number of steps, setting the switch to full steps, and slowly increasing the step frequency until the rotor does not rotate when you send the pulse train.

    Stall frequency = ________________Hz

12. Unplug the inputs to the stepper motor, connect them to ACH0 and ACH1 of the DAQ breakout box, press the 'Display Chip Output' button, set the number of steps to 5 at 10 Hz, and press the 'Send Pulse Train' button. The signals being sent to the motor coils will be graphed on the lower screen: draw a picture of the signals in the table provided.

13. Repeat step 12 for the remaining combinations of rotational direction and stepping mode and record your observations in the table provided (sketches could be helpful).

	Clockwise	Counter-clockwise
Full-stepping		
Half-stepping		

14. Breakdown the stepper motor setup and return the connecting wires to the appropriate racks.

Servo Motors

15. Locate an RC Servo and a Handy Board and interface it with Interactive C at your lab station.

16. Open Windows Explorer, navigate to C:\IC\libs\RCservo, and copy the following files onto your disk: servoExample.c, servo_a5.icb, and servo_a7.icb. These files are all you need to control the RC Servo

17. Open your copy of servoExample.c and examine the code to understand what the code does.

18. Connect the RC Servo to your Handy Board by connecting the white lead of the servo to digital port nine (top input bus), the red lead to the +5V supply bus (middle bus), and the black lead to the ground bus (bottom bus).

19. Download your copies of servoExample.c, servo_a5.icb, and servo_a7.icb to your Handy Board and reset your Handy Board. The servo should respond by moving to the center of its angular travel.

20. Rotate the knob of the Handy Board to actuate the servo back and forth. As you rotate the knob position the display shows the number of clock ticks that determine the width of the input pulse to the servo.

21. Determine the maximum and minimum values of the `servo_a7_pulse` variable that correspond to the maximum and minimum angles the servo can travel before it hits its stop by modifying the values of min and max in your copy of the servoExample.c file, reloading the code, and using the screen output and the knob to determine the limits on the pulse width.

    Caution: The servo can break the mechanical stop if you command the servo to go past the limits imposed by its mechanical stop. Before you load the modified code, make sure that the knob is somewhere in the middle of its travel so that it doesn't accidentally try to move past the stop. Then carefully determine the maximum and minimum travel by moving the knob toward its maximum and minimum positions respectively.

22. Record the values you determined in the previous step and modify the servoExample.c program to reflect these values.

    Max: ______________________ Min: ______________________

23. Place your finger on one of the legs of the rotor and oppose the movement of the servo to qualitatively measure the torque output of the servo.

24. Connect the digital nine input to A_CH0 of the DAQ (make sure the DAQ will measure the output relative to the ground of the Handy Board). Observe the output using the 2_channel_oscope.vi VI, turn the knob to its maximum and minimum positions, and measure the width of the clock pulse for each. Record these values below

    Max Pulse = ______________________sec Min Pulse = ______________________sec

25. Clean up your station, close LabView, and answer the questions at the end of the lab.

Questions

1. List at least two possible applications of stepper motors.

2. How can you determine the absolute position of the stepper motor shaft?

3. Explain why the timing of the waveforms in the table of step 13 is different for the various combinations of direction and step-mode. What do the timing differences do?

4. When you would use a stepper motor versus a DC motor?

5. List at least two applications of servo motors.

6. What are two benefits of using an servo in your robot project?

7. What are two limitations of using a servo in your robot project?

8. What is required to change a DC motor into a servo motor?