ME 3200 Mechatronics Laboratory

Lab Exercise 9: Stepper and Servo Motors

Introduction

In this laboratory exercise, you will explore some of the properties of stepper and servomotors. 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

The term "stepper motor" is a short descriptive name for a wound synchronous motor. These motors operate on the principle of a rotating magnetic field. 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 chases the 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.

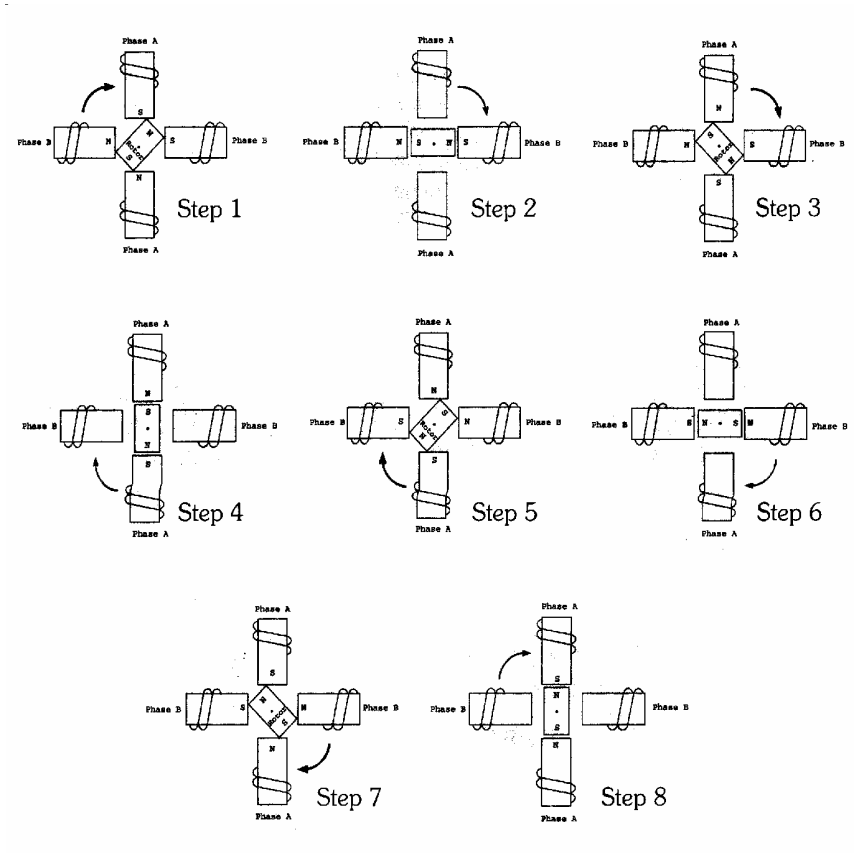

Figure 1: Schematic of a stepper motor with two sets of windings and a single pole rotor taking half steps.

The stepper motor of Figure 1 does not have a very desirable angular resolution since the rotor has only one pole pair. To increase angular resolution, the rotors of most stepper motors contain many 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 is defined according to the following equation:

$$\Delta\theta = \frac{360°}{m \cdot N} \qquad (1)$$

where $\Delta\theta$ is the resolution, m is the number of stator phases, and N is the number of rotor teeth. Equation 1 applies only for full-step configurations. The half-step configuration 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 built around the Motorola MC3479P stepper motor driver chip drives the stepper motors used 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

Servomotors 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.

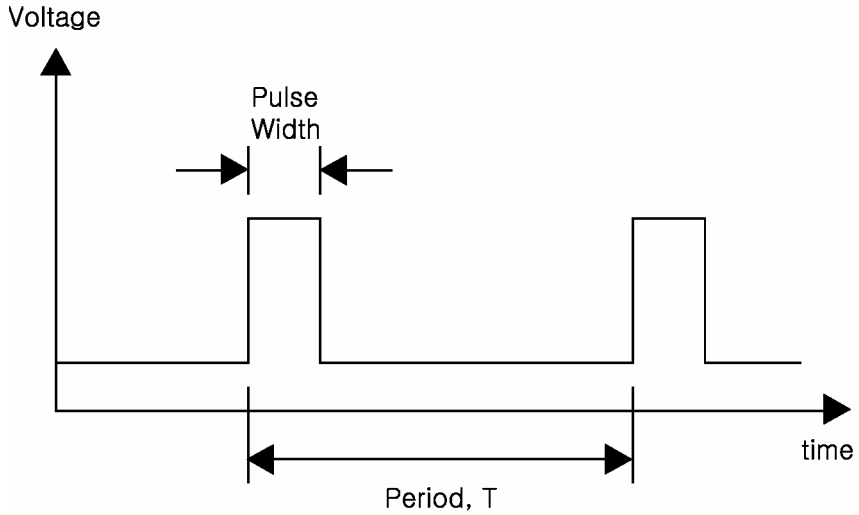

Figure 2: PWM signal

The servos used in this lab are the same RC servos used in radio controlled airplanes and cars. These RC servos require a pulse width modulated (PWM) input 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 RC servos, the width of the pulse determines the angle of the servo output.

Interfacing and controlling the RC servo with the Handy Board is very simple. Each servo has three input wires: the white wire accepts the PWM output of the Handy Board, the red wire requires a regulated +5V input, and the black wire is the ground connection for the internal circuitry. 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, 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 the 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 allow us to control the servo attached to digital nine (please note that changing the 7 to a 5 in these filenames will alter the servo behavior on TOC3). Calling `servo_a7_init(1)` enables the control of the servo connected to digital nine port: `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. 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 is a lot of variation between the maximum and minimum pulse width 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
- Wire cutters/strippers
- A Handy Board
- An RC Servo
- Digital multimeter
- 2_channel_oscope.vi

Stepper Motors

1. Build the stepper motor control circuit shown in Figure 3 making sure that you can easily change the voltage on pins nine and ten.

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.

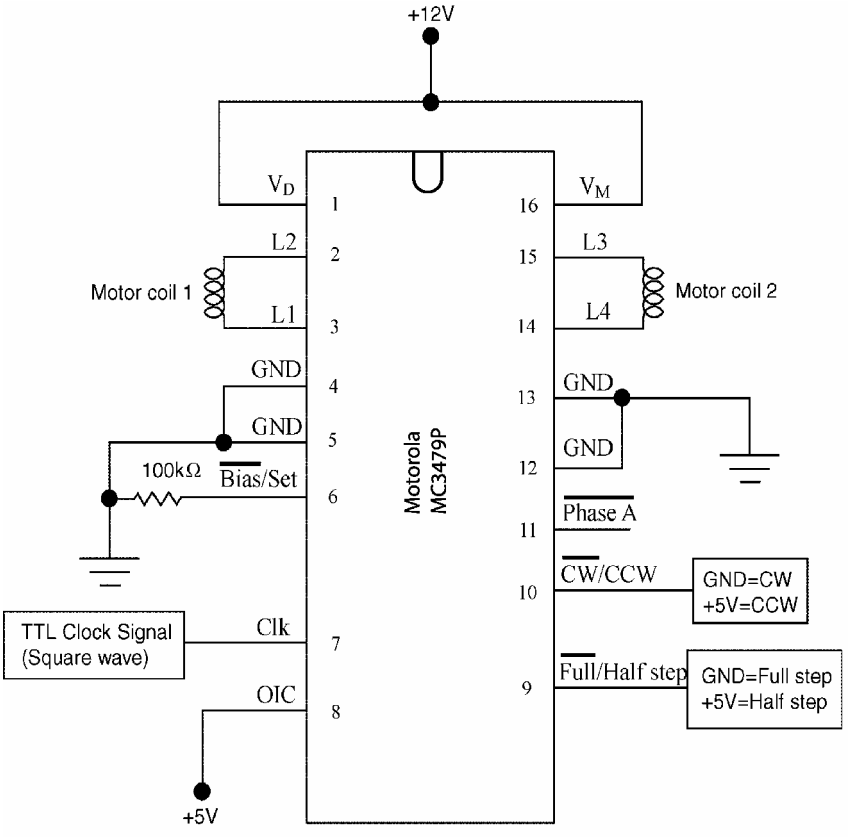

Figure 3: Pin out schematic for the driver chip.

3. Open the 2_channel_oscope.vi from the LabView folder on the computer desktop and connect the output of the signal generator to the A_CH0 input of the DAQ breakout box.

4. Using the 2_channel_oscope.vi, adjust the output of the signal generator to a 2.5V, 2Hz square wave that oscillates between zero and five volts.

   Note: To meet the amplitude requirement for the step above, the DC offset of the signal generator must be modified to 2.5V. The DC offset is adjusted by pulling out the Offset knob and rotating the knob clockwise to increase and counterclockwise to decrease the offset.

5. Connect the output of the signal generator to the Clk input of the driver circuit (the stepper motor should begin to turn). Try different voltage inputs for pins nine and ten to change the stepping method and the direction of rotation, respectively.

6. Unplug the inputs to the stepper motor, connect them to the first two input channels of the DAQ breakout box, and use the 2_channel_oscope.vi to observe the two control signals for different combinations of rotational direction and stepping mode. Record your observations in the table below (sketches could be helpful).

	Clockwise	Counter-clockwise
Full-stepping		
Half-stepping		

7. With the input frequency of 2Hz, measure the stepping torque of the motor with your finger. What happens to the stepping torque when the input frequency is increased?

8. Determine the number of rotor pole pairs of your stepper motor by counting the number of steps it takes to complete one revolution in full-step mode, and using 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 = _____

9. Determine what input frequency makes the stepper motor stall, by slowly increasing the input frequency until the rotor stops rotating. You may note that the motor misses steps before it completely stalls. Why?

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

Servo Motors

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

12. 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

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

14. 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).

15. 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.

16. 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.

17. 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.

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

    Max: _____ Min: _____

19. 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.

20. 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 VI you opened previously, turn the knob to its maximum and minimum positions, and measure the width of the clock pulse for each. Record them below

    Max Pulse = _____sec Min Pulse = _____sec

21. Clean up your station 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. What are some potential disadvantages of having more windings in a stepper motor?

4. Compare and contrast stepper motors to DC motors.

5. List at least two applications of RC servomotors.

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

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

8. Compare and contrast a servomotor to a DC motor.