RSA Lab #5: Stepper Motors

# Part A: Stepper Motor Controller: A “Dry Run” Using LEDs

For this part I was able to make a program to depict a series of LEDs blinking in sequential order using full steps, in preparation for applying this program to the stepper motor.

1. Design a stepper-motor drive circuit for the Jameco 2138812 stepper motor (see the motor data sheet posted on the course Blackboard site). Hand in a circuit drawing of your complete circuit with your lab report. Show your resistor selection calculation.
2. Write a program to control the motor using the full-step high-torque commutation as shown on the motor spec sheet. This will be used to light the LEDs in this step; the motor will be added in part C. Include a hardcopy with your lab report. You must have comments written into the program; it is not sufficient to add comments in later by hand.

Text

Description automatically generated

# Part B: One Small Step: Control the LEDs Using a Button

This section imitates the previous section, however now I used half-steps to essentially half the increment time.

1. Create a new version of your program to use the half-step increments. Keep the same 500ms delay between increments. Test your program and hand in a hardcopy. You must have comments written into the program; it is not sufficient to add comments in later by hand.

Graphical user interface, text, application

Description automatically generated

# Part C: One Giant Leap: Control the Stepper Motor Using a Button

For this section I applied the program from the previous two sections into the stepper motor to cause a rotation of various degrees.

1. Create a new version of your program so that the motor turns a full circle (360 degrees) using full steps each time the button is pressed. Use a 50ms delay between increments. Test your program and hand in a hardcopy. You must have comments written into the program; it is not sufficient to add comments in later by hand.

Graphical user interface, text, application

Description automatically generated

1. Now modify the delay time between steps. (Note: make sure you don’t have any Serial.print statements that will slow things down.) What is the shortest delay the motor can keep up with using full steps? What causes this limitation? Explain briefly.

Upon testing smaller and smaller delay durations, I found that the shortest delay the motor and keep up with is approximately 4ms, since at 3ms the motor wasn’t able to complete a full rotation. This limitation is caused by the discrepancy between the speed at which the Arduino sends the signal to the motor and the time it takes for the motor to rotate.

1. Use your fingers to feel for the stall torque (more precisely for a stepper motor, the “pullout torque”) of this motor at this voltage. What happens when you exceed this maximum? Does the motor “catch up” when the obstruction is released? (In other words, when you release it, does the motor finish the full 360°?) Explain briefly.

The motor does not catch up; once you exceed the maximum stall torque, the motor just makes partial revolutions.

1. Now modify the delay time between steps. What is the shortest delay the motor can keep up with using half steps? How does this compare with the minimum delay for full steps? Explain briefly.

After changing the code to increment in half steps, I found the new shortest delay to be approximately 6ms, which is two milliseconds more than when it rotated with full steps. This makes sense because a smaller step size should

# Part D: More Control: Three-Button User Interface

Finally, in this part I was able to combine the programs from the previous section to control how the motor rotates from three buttons.

1. Finally, make a version of your stepper-motor control program that will use 3 buttons for a user interface for the motor. Hand in a circuit drawing of your complete circuit with your lab report. Test your program and hand in a hardcopy of your program. You must have comments written into the program; it is not sufficient to add comments in later by hand.

Text

Description automatically generated