ECE271: Microcomputer Architecture and Applications — University of Maine

Lab #5: Stepper Motor in C

Week of 28 February 2022

Goals

1. Understand the limits of GPIO current.
2. Learn to use Darlington transistor arrays to drive devices needing higher current loads.
3. Understand the usage of full and half stepping to control the speed and position of a stepper motor.

Pre-lab

1. Complete the pre-lab before attending lab. The pre-lab is in a separate pdf file, found on the website.
2. Be sure to bring a breadboard and wires to the lab. The stepper motor and driver board will be provided.

Lab Procedure

The end goal of this lab is to be able to exhibit fine-grained control over the position of the stepper motor.

You will need to be able to rotate the motor shaft exactly 360 degrees counter-clockwise via both half and full stepping.

Part A – Connect the Stepper Motor

1. A stepper motor and control board will be provided, which looks like Figure 1.
2. You will connect this to your STM32L4Discovery board. You will probably want to do this via a breadboard. You might also want to keep your keypad connected as well, as it can be useful for various of the "something cool" options.
3. Connect the motor to the controller board by snapping together the white connectors (see Figure 1). The connector is keyed so it should only connect in one direction.
4. You will now need to connect six wires from the controller board to your STM32L4 board. You will be provided with some female/male wires that might make interfacing with the breadboard easier. A diagram is shown in Figure 2.
   (a) Connect PB2 to IN1 on the controller board.
   (b) Connect PB6 to IN2 on the controller board.
   (c) Connect PB3 to IN3 on the controller board.
   (d) Connect PB7 to IN4 on the controller board.
   (e) Connect one of the GND pins on the STM32L4 to the - (minus) pin on the controller board.
   (f) Connect 5V on the STM32L4 to the + (plus) pin on the controller board.

Figure 1: Stepper motor and control board.

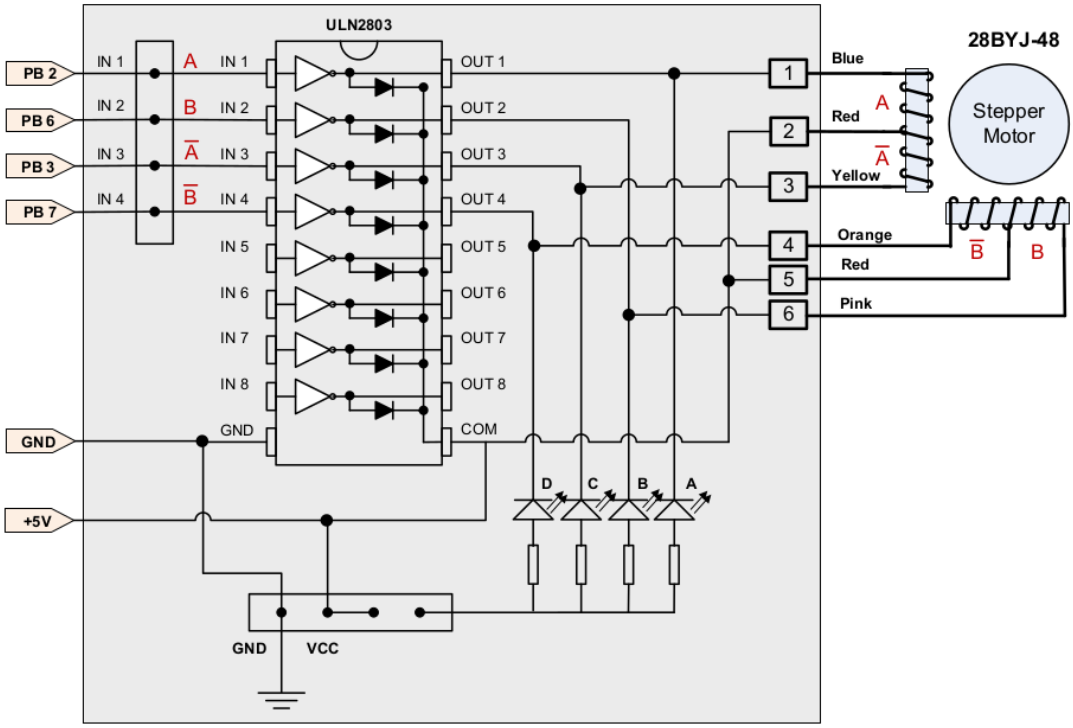
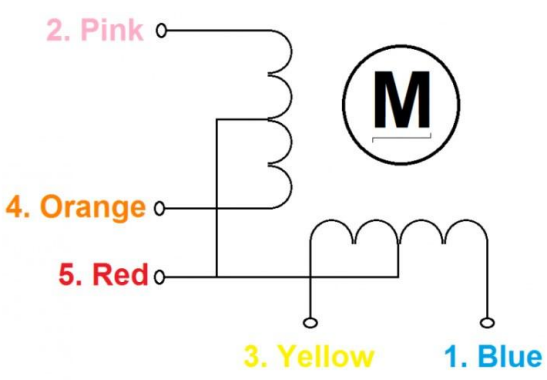
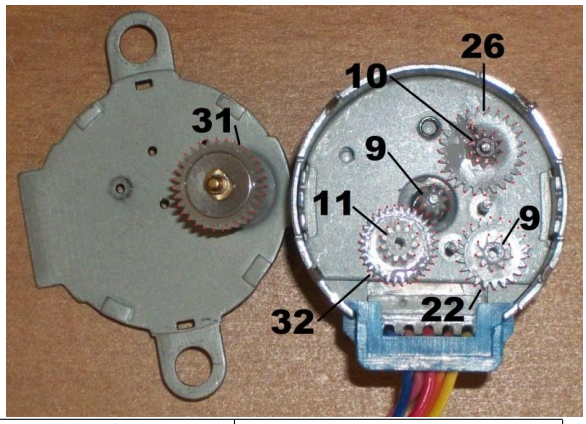


Figure 2: Motor board connections.





Motor Model	28BYJ-48	Number of Phases	2
Rated Voltage	5V DC	Geared reduction ratio	1/64
DC Resistance per phase	$50\Omega \pm 7\%(25^oC)$	Pull in torque	$> 300gf.cm/5VDC100pp$
Darlington Array	ULN2003	Gear Ratio	$\frac{31\times32\times26\times22}{11\times10\times9\times9} = 63.68395$

Figure 3: Some hardware info on the stepper motor. If the output shaft rotates 1 revolution (gear with 31 teeth in the figure) the internal shaft (gear with 9 teeth) rotates approximately 64 revolutions.

5. Note that the controller board/motor will draw power while powered up and might get hot. If you are not actively using or debugging the motor you might want to disconnect the 5V wire.
6. It can be hard to see the position of the motor while testing. A piece of tape stuck to the shaft can make it easier to see what direction it is pointing.
7. Some hardware info on the motor can be found in Figure 3

Part B – Code – Initialization

1. This lab will be done in C.
2. Use your code from Lab#3 as a base. Modify `main.c`
3. Add a `Stepper_Pin_Init()` function.
   (a) Be sure the GPIOB clock is enabled.
   (b) Set GPIOB pins 2, 3, 6, and 7 as digital outputs. (We didn't do this in the prelab, but you should be good at doing this kind of thing by now).
4. Be sure your main function calls this `Stepper_Pin_Init()`

Part C – Code – Full Stepping

1. Set up the full-stepping code. As a summary:
   - The internal motor has 32 steps per revolution
   - Gear reduction of 1/63.68395, or approximately 1/64

- Thus it takes 32*64 = 2048 steps for one full turn of the input shaft.

2. The textbook has some reference code in Example 16-1 but we are doing things a bit differently so don't be misled by it.

3. Create a function,

```
void Stepper_Full_Step(int angle);
```

that rotates the motor shaft by `angle` degrees counter-clockwise.

4. Use the hex values you calculated in the pre-lab for the four steps, updating the `GPIOB->BSRR` register for each step.

5. You will want to delay between each step. A simple loop will work. The textbook uses a loop to 6000, which seems to be a reasonable value to use.

6. It takes 2048 steps, or 512 repeats of the 4-step pattern, to complete a rotation. Your function should do the math to convert the value in degrees to a number of steps, and then do the rotation.

7. Call this function with various angles and be sure it does the right thing. We will have you demo 360 degrees.

8. Be careful when doing the angle calculations in C. Remember that something like (90/360) using integer math in C will get truncated to zero.

Part D – Code – Half Stepping

1. Set up the half-stepping code. As a summary:

   - The internal motor has 64 half steps per revolution
   - Gear reduction of 1/63.68395, or approximately 1/64
   - Thus it takes 64*64 = 4096 half steps for one full turn of the input shaft.

2. The textbook has some reference code in Example 16-2 but we are doing things a bit differently so don't be misled by it.

3. Create a function,

```
void Stepper_Half_Step(int angle);
```

that rotates the motor shaft by `angle` degrees counter-clockwise.

4. Use the hex values you calculated in the pre-lab for the eight steps, updating the `GPIOB->BSRR` register for each step.

5. You will want to delay between each step. A simple loop will work. The textbook uses a loop to 6000, which seems to be a reasonable value to use.

6. It takes 4096 half-steps, or 512 repeats of the 8-step pattern, to complete a rotation. Your function should do the math to convert the value in degrees to a number of steps, and then do the rotation.

7. Call this function with various angles and be sure it does the right thing. We will have you demo 360 degrees.

Part E – Something Cool

Do something cool! You can come up with something on your own, but here is a list of ideas you can use.

1. Output the current rotation angle to the LCD.
2. Get input from the keypad and rotate that number of degrees.
3. Change rotation from counter-clockwise to clockwise based on the keypad.
4. Rotate the motor based on the joystick button (i.e., press right and it rotates 90 degrees clockwise) (note: remember if you want to use the joypad you have to disconnect the keypad)

Lab Demo

1. Submit your code

   - Complete a README with the post-lab answers.
   - Make sure the code is properly commented.
     This includes a header at the top of your main.c with your name and a brief summary of the lab.
   - Check your code and README into your gitlab tree.

2. Demo your implementation to your lab TA.

   (a) Rotate the motor 360 degrees counter-clockwise using full stepping.

   (b) Rotate the motor 360 degrees counter-clockwise using half stepping.

   (c) How fast can you rotate using full-stepping? (Reduce your delay loop until it no longer turns a full 360 degrees)

   (d) How fast can you rotate using half-stepping? (Reduce your delay loop until it no longer turns a full 360 degrees)

   (e) Is the highest frequency of the half-stepping higher than that of full-stepping? Why?

Post-Lab

- Place your answers to the question in a file Readme.md
- Submit with your code via the gitlab server.
- Questions:

  1. The Darlington array can only provide 500mA of current. If you needed a larger current, what could you use instead of the Darlington array?
  2. Is it possible to rotate the motor less than 1/2 step? (Hint, see Chapter 16.6 of the textbook)