THE UNIVERSITY OF TEXAS AT ARLINGTON

COMPUTER SCIENCE AND ENGINEERING

MECHATRONICS
LAB 2 REPORT

ELECTROMECHANICAL SYSTEMS & SENSORS

Submitted toward the partial completion of the requirements for CSE 5355-001

Submitted by,

Jennifer Hernandez &

Servando Olvera

Date 09/23/2024

2. Measure the mass of the weight and the distance from the center of the motor shaft to the weight to calculate the holding torque in N-m. Note the current for the power supply.

Weight:	225 g	Torque = F * r * sin(90)
Distance:	10.5 cm	Torque = (m*g) * d * 1
Current:	0.246 A	Torque = (0.225 kg * 9.8 m/s^2) * 0.105 m * 1
		Torque = 2.205 kgm/s^2 * 0.105 m
		Torque = 2.0205 N * 0.105 m
		Torque ~ 0.232 N-m

Throughout this step the current on the power supply remained a constant 0.246 A. However, it was noted that upon finding the holding torque and going over it, there was a small downward spike in current that would quickly settle back up to 0.246 A.

3. Decrease the voltage to 9.4V and repeat step 2.

Weight:	225 g	Torque = (0.225 kg * 9.8 m/s^2) * 0.102 m
Distance:	10.2 cm	Torque = 2.205 kgm/s^2 * 0.102 m
Current:	0.246 A	Torque = 2.0205 N * 0.102 m
		Torque ~ 0.225 N-m

The same observation as on step 2 can be made for step 3, regarding the behavior of current.

4. Decrease the voltage to 8.4V and repeat step 2.

Weight:	225 g	Torque = (0.225 kg * 9.8 m/s^2) * 0.099 m
Distance:	9.9 cm	Torque = 2.205 kgm/s^2 * 0.099 m
Current:	0.246 A	Torque = 2.0205 N * 0.099 m
		Torque ~ 0.218 N-m

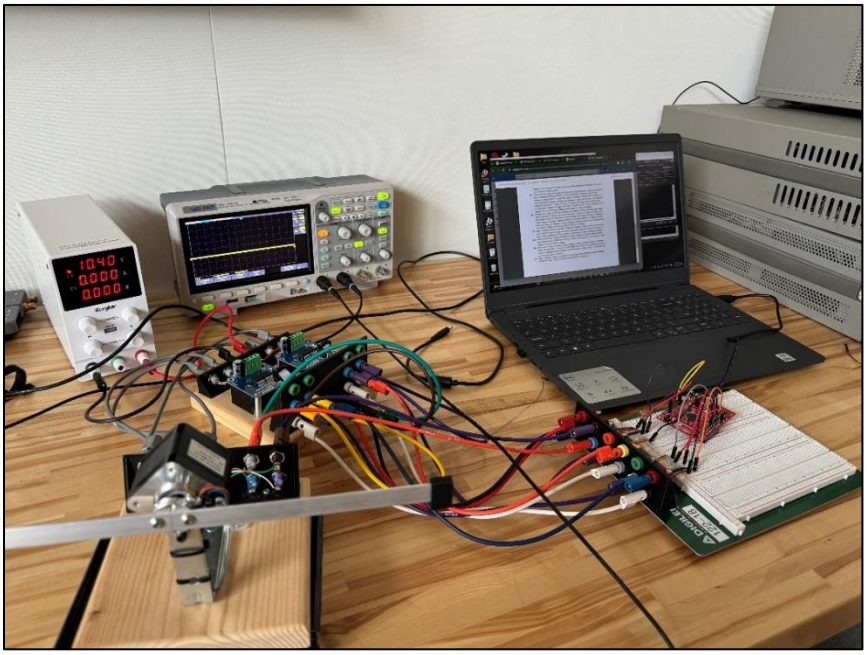
It seems that current and torque are proportional to one another. The previous steps showed that upon finding the holding torque and going over it, the current would spike down. It seems that decreasing current would decrease torque, and increasing current would increase torque.

5. Now, remove the weight and power the 4 coil wires in a + - * *, * * + -, - + **, * * - + pattern and note that the motor turns.

The pattern makes the motor turn clockwise.

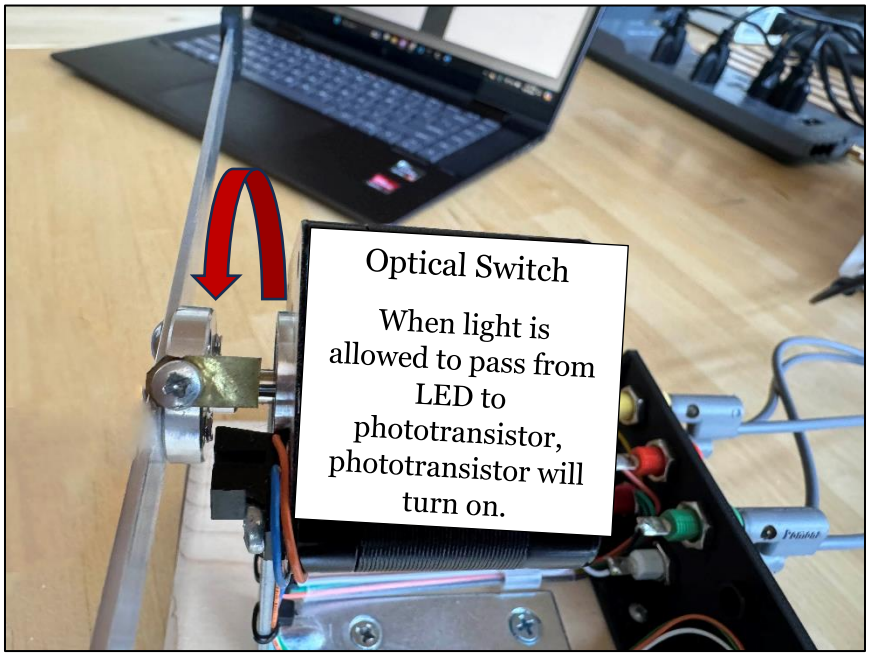
Reversing pattern makes the motor turn counterclockwise.

9. **Write a simple loop to configure the 6 pins as simple GPIO and write a simple loop to rotate the motor 16 * 1.8 degrees CW and then 16 * 1.8 degrees CW in a rocking pattern.**



The code was designed to toggle the direction inputs of the motor driver using the GPIO of the microcontroller. The direction bits coincided with the pattern used in step 5 to turn the motor clockwise and counter-clockwise.

11. Next, interface the optical slotted switch with your software. Modify your code so that the software will rotate until the collector goes high, indicating that the brass tab is blocking light. Use this as a calibration point and then rotate CCW until the beam is level (empirically determine the number of CCW steps to be level).



The code was designed to monitor the status of the phototransistor and execute steps in the clockwise direction until the phototransistor was blocked from receiving light. Then, the program drove the motor in the counter-clockwise direction for a set amount of steps until the beam was leveled. The number of CCW steps to level the motor fluctuated around 20-26 mechanical steps, meaning 5.5 to 6.5 full electronic steps, depending on the motor being used and the value in *waitMicrosecond()*.

14. Now, write code to microstep the stepper motor so that less than 1.8 degree steps are possible. You will vary the PWM to be either the abs(A*cos(angle)) or abs(A*sin(angle)) and the other coil will be driven high or low. Try to see how small a step you can create.

FULL STEPS TABLE SAMPLE

i	θ	Coil A PWM	Coil B PWM	Coil A		Coil B	
		$\cos\left(i * \frac{\pi}{2}\right)$	$\sin\left(i * \frac{\pi}{2}\right)$	DIR1	DIR2	DIR1	DIR2
0	0°	1	0	1	0	0	0
1	90°	0	1	0	0	1	0
2	180°	-1	0	0	1	0	0
3	270°	0	-1	0	0	0	1

The basic idea of the code was to calculate the above values based on the type of steps that the code was executing. The type of steps ranged from full steps (1), half steps (1/2), quarter steps (1/4), one-eighth steps (1/8), one-sixteenth steps (1/16) and one-thirty-second steps (1/32). The type of step used when running the program was determined beforehand. As the steps became smaller, the table grew larger. This correlated to more iterations through the code to move the beam to the desired location.