CSE4355/5355 Electromechanical Systems and Sensors LAB 2

Anaf Mahbub

1001952603

Yasasvi Vanapalli

1002182593

1. Introduction

In this lab, we explored stepper motor control using a TM4C123GH6PM microcontroller and a BTS7960 H-bridge module. The goal was to understand how varying voltage and current affect the torque of the motor and implement motor control using a microcontroller to drive the stepper motor at full, half, and quarter and 32nd steps. The experiment also involved using an optical encoder for position calibration.

The primary objective was to:

- Calculate the holding torque at different voltages.
- Develop code to rotate the motor using different stepping modes.
- Calibrate the motor's position using an optical switch.
- Implement micro stepping to achieve smooth and fractional rotation.

2. Experimental Procedure

2.1 Holding Torque Measurement

The stepper motor was driven by different voltages (10.4V, 9.4V, 8.4V), and the corresponding current and torque were measured. A weight was applied to the beam, and the motor's holding torque was calculated based on the current, mass of the weight, and distance from the motor shaft to the weight.

The formula used to calculate torque was:

Torque = mass × gravity × distance

2.2 Stepper Motor Control

Using the TM4C123GH6PM microcontroller, coil A and coil B were connected and driven by the appropriate GPIO and PWM signals. The motor was configured to rotate using a full step, half step, and quarter step patterns. Additionally, PWM control was implemented for micro stepping, aiming for less than 1.8° per step.

2.3 Calibration with Optical encoder

The optical encoder was interfaced with the microcontroller. The motor rotated until the switch detected the blocking tab, which served as a calibration point. The motor was then rotated empirically counterclockwise until the beam was level.

3. Results

3.1 Torque Calculation

From the experimental readings (shown in Table 1), the holding torque was calculated at different voltages.

Voltage (V)	Current (A)	Power (W)	Distance (cm)	Mass (g)	Torque (N-m)
10.40	0.298	3.099	11.5	225	0.025
9.40	0.269	2.528	10.9	225	0.024
8.40	0.239	2.007	9.8	225	0.021

Table 1: Measured and Calculated Torque Values:
Torque values were calculated using the mass of 225g and the varying distances from the center of the motor shaft to the weight. A gradual decrease in torque was observed as the voltage decreased.

3.2 Code Implementation and Observations

The code was successfully implemented to rotate the motor using full steps. However, issues were encountered with half-step and quarter-step modes:

- **Full stepping**: The motor successfully rotated in a forward and backward direction, with a clear 1.8° step between rotations.
- **Half stepping**: the motor failed to rotate the full 0.9° step.
- **Quarter stepping**: The motor failed to rotate the full 0.45° step, suggesting issues in generating accurate direction control signals for micro stepping.
- **32nd stepping**: the motor rotated successfully in the clockwise and counterclockwise direction with a clear step angle of 0.05625° in a smooth and jitter free manner.

The issue with half-step and quarter-step modes could be due to imprecise PWM signal generation or incorrect coil activation sequences. Further debugging is required to refine the stepping logic and PWM control.

The variation of the direction control pins to 1 and 0 to energize the coils was identified to be the issue in the case of half and quarter stepping, where the sin(theta) and cos(theta) where responsible in determining which poles of Coils A and B where to be energized. The PWM signals were generated based on the motor factor which was used to determine whether the stepper motor had to rotate for full, half, quarter or 32nd steps using the formula fabs(sin(angle)) and fabs(cos(angle)).

Code Overview

The code was designed to control the rotation of a stepper motor using the TM4C123GH6PM microcontroller. The motor is driven using a series of digital pulses also known as Pulse Width Modulation or PWM that activate the motor coils in a specific pattern. The main operations of the code are:

- Configuring the necessary GPIO pins to control the motor.
- Driving the motor coils in full step, half step, and quarter step and 32nd step modes.
- Using PWM signals to the Enable pins to achieve micro stepping.
- Calibrating the motor's position using the optical encoder.

Key Components

1. **GPIO-Configuration**:
   The GPIO pins are configured to drive the stepper motor coils. Each coil of the motor is controlled by two outputs, and the direction of rotation is determined by the sequence in which these outputs are toggled.
2. **Step-Sequence**:
   The motor is driven by alternating current between the coils in a specific sequence. This sequence determines the step size (full, half, quarter or 32 steps) and direction (clockwise or counterclockwise). The basic sequences for full and half steps are stored in an array or handled through conditional logic.
3. **PWM-Control**:
   Pulse-width modulation (PWM) is used to control the enable inputs of the motor. By varying the duty cycle of the PWM signal, the motor can be driven with less than the full current, allowing for fractional steps (micro stepping).
4. **Optical-Encoder**:
   The optical encoder is interfaced with the microcontroller to provide feedback on the motor's position. When the beam blocks the light in the optical switch, the code uses this signal to determine a reference position for calibration.

2. Full-Step Logic

In full-step mode, the motor rotates in 1.8° increments. The logic works as follows:

- Each step requires a combination of two coils being energized in a particular order.
- The program cycles through a predefined set of states to drive the coils, with each state corresponding to a 1.8° step.
- Coil activation sequence:
  - **Step 1**: Coil A is powered.
  - **Step 2**: Coil B is powered.

    - **Step 3**: Coil A is powered in reverse.
    - **Step 4**: Coil B is powered in reverse.
- By looping through this sequence, the motor rotates clockwise (CW). To reverse the motor's direction (counterclockwise, CCW), the sequence is reversed.

Example of Coil Activation:

For example, a full-step rotation sequence might look like this in terms of coil activation:

Step	Coil A (WHITE-BLACK)	Coil B (GREEN-YELLOW)
1	High	Low
2	Low	High
3	Low	High (reverse)
4	High (reverse)	Low

The program cycles through these steps, applying a delay between each step to ensure the motor has enough time to rotate to the new position.

3. Half-Step Logic

In half-step mode, the motor should rotate in 0.9° increments, providing more precision than full-step mode. The logic is similar to full-step mode but includes additional steps where only one coil is powered at a time. This results in the following sequence:

- **Step 1**: Coil A is powered.
- **Step 2**: Both Coil A and B are powered.
- **Step 3**: Coil B is powered.
- **Step 4**: Only Coil B is powered in reverse.
- **Step 5**: Both Coil B and A are powered in reverse.
- **Step 6**: Only Coil A is powered in reverse.

Issues with Half-Step Mode:

The problem you are experiencing with half-step mode may stem from timing issues or improper sequencing of the coil activation. If the PWM signal is not being applied correctly or if the delays between steps are inconsistent, the motor may not be receiving a clean signal for each half step.

4. Quarter-Step Logic

In quarter-step mode, the motor attempts to rotate in increments of 0.45°. This requires more precise control of the current flowing through the coils, which is achieved by varying the PWM signal. The

logic involves applying fractional power to each coil, with the current being a function of the sine and cosine of the desired angle.

The PWM duty cycle for each coil would be calculated based on the following formula:

- Coil A: PWM_A = abs(A * cos(angle))
- Coil B: PWM_B = abs(A * sin(angle))

Issues with Quarter-Step Mode:

Quarter-step mode requires precise timing and current control. The motor may jitter or not move correctly if:

- The PWM signals are not correctly synchronized.
- The sine/cosine values are not calculated or applied accurately.
- The microcontroller does not handle the precise timing needed for quarter-step control, resulting in inconsistent movements.

5. 32nd step Logic

In 32nd -step mode, the motor attempts to rotate in increments of 0.05625°. This requires more precise control of the current flowing through the coils, which is achieved by varying the PWM signal. The logic involves applying fractional power to each coil, with the current being a function of the sine and cosine of the desired angle. The timing of the waitMicroseconds function was also observed to be critical in ensuring the timing of the stepper motor Coil powering, which could potentially make the movement jittery and cause the motor to cog back to the previous pole if the timing was too fast.

The PWM duty cycle for each coil would be calculated based on the following formula:

- Coil A: PWM_A = abs(A * cos(angle))
- Coil B: PWM_B = abs(A * sin(angle))

Issues with 32nd Step Mode:

32nd Step mode requires precise timing and current control. The motor may jitter or not move correctly if:

- The PWM signals are not correctly synchronized.
- The sine/cosine values are not calculated or applied accurately.
- The microcontroller does not handle the precise timing needed for 32Nd step control, resulting in inconsistent movements.

5. Calibration with Optical encoder

The optical encoder is used to determine a known position (the "home" position) of the motor. When the motor rotates, the optical switch detects when a tab on the motor blocks the light, which signals

the microcontroller that the motor has reached a specific position. This position is then used to calibrate the motor and establish a reference point.

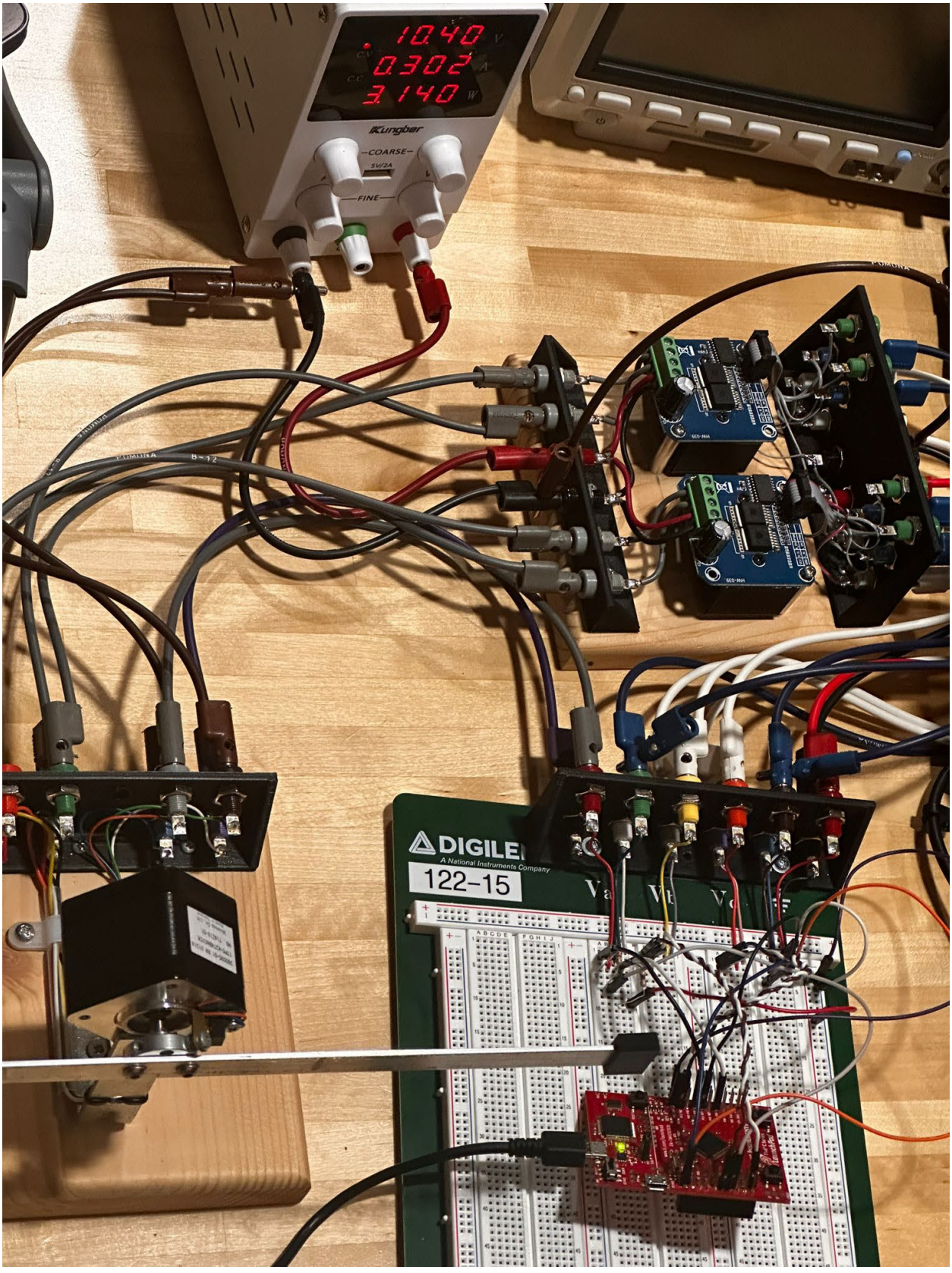
How It Works:

- The optical encoder has an IR LED and a phototransistor. When the beam of light from the LED is blocked, the phototransistor sends a signal to the microcontroller.
- The code monitors the signal from the optical switch. When it detects a high signal on the GPIO pin(indicating the light is blocked), it uses this as a calibration point.
- After calibration, the motor can rotate from this position in both CW and CCW directions, with the step count tracked.

3.3 Calibration with Optical Switch

The optical encoder was correctly interfaced, and the motor rotated until the switch indicated the tab was blocking light. This provided a reliable calibration point for the motor, though some fine-tuning in terms of step count for leveling was necessary.

4. Pictures


10.40
0.302
3.140
Kungber
COARSE
5V/2A
FINE
DIGILE
A National Instruments Company
122-15

5. Conclusion

This lab provided valuable insight into the control of stepper motors using a microcontroller. While the full-step rotation was successful, there were issues with half-step and quarter-step modes, which need further investigation. The calibration mechanism using an optical switch was effective, though improvements could be made to fine-tune the motor's movement during micro stepping.