University of California, Riverside

Lab #5 Report

Stepper Motor Control

EE 128 Section 21, Fall 2025

Name: Muneer Al Jufout

Student ID: 862410258

Lab Partner: Sameer Anjum

Lab Partner ID: 862422332

TA: Johnson Zhang

Abstract

The objective of this lab was to design and implement a stepper motor control system using the FRDM-K64F microcontroller and the L298N dual H-bridge driver. The system enables four operation modes: clockwise (CW) and counterclockwise (CCW) rotations at two different angular speeds controlled by a DIP switch. The microcontroller outputs sequential logic signals to the L298 module to power the motor in the correct order, producing smooth rotation in both directions. A simple delay-based loop determines the motor speed. The implemented system successfully demonstrated direction and speed switching without too much time in between.

Experiment System Specification

For this lab, Port C was used to drive the stepper motor, which was connected to the L298 motor driver's output. We also used Port B for the switch to control the direction and the speed of the stepper motor.

Flowchart

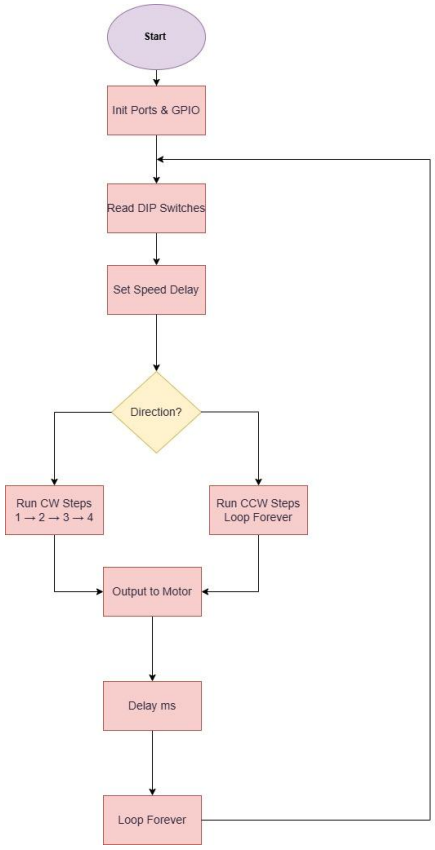
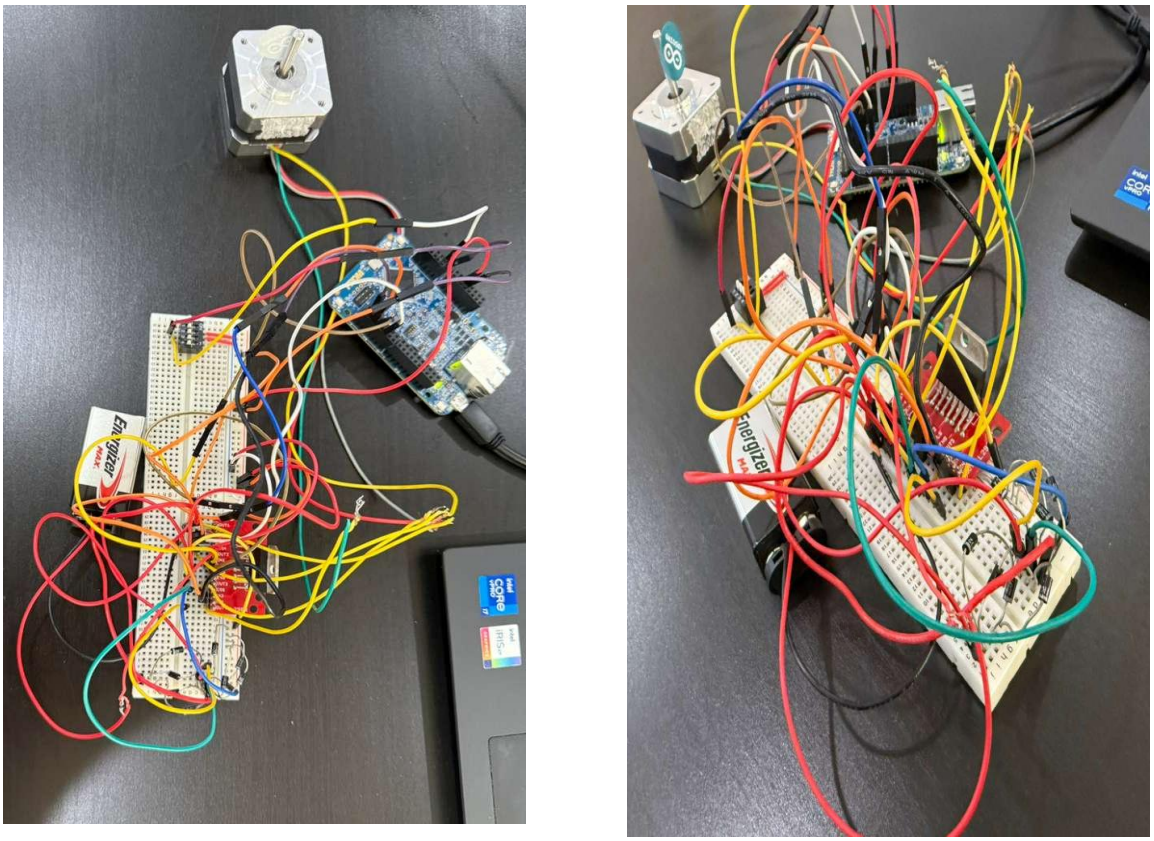


Fig. 1: The Program Flowchart

Hardware Design

The L298N motor driver was connected to the breadboard, where the K64F microcontroller and the stepper motor were connected to it. The K64F was connected to the EDA and EDB, and each IN, while the stepper motor was connected to each output by using two diodes for each stepper motor wire. A 4-bit DIP switch was connected to the K64F to control the direction and speed.



a) b)

Fig. 2: Photos of the Setup

Schematic Diagram

This schematic included:

- MK64FN1M0VLL12 microcontroller
- Ground (GND)
- 4-bit DIP switch (piano type)

- 2n4001 diodes
- L298N motor driver
- 2-phase, 12V DC bipolar stepper motor

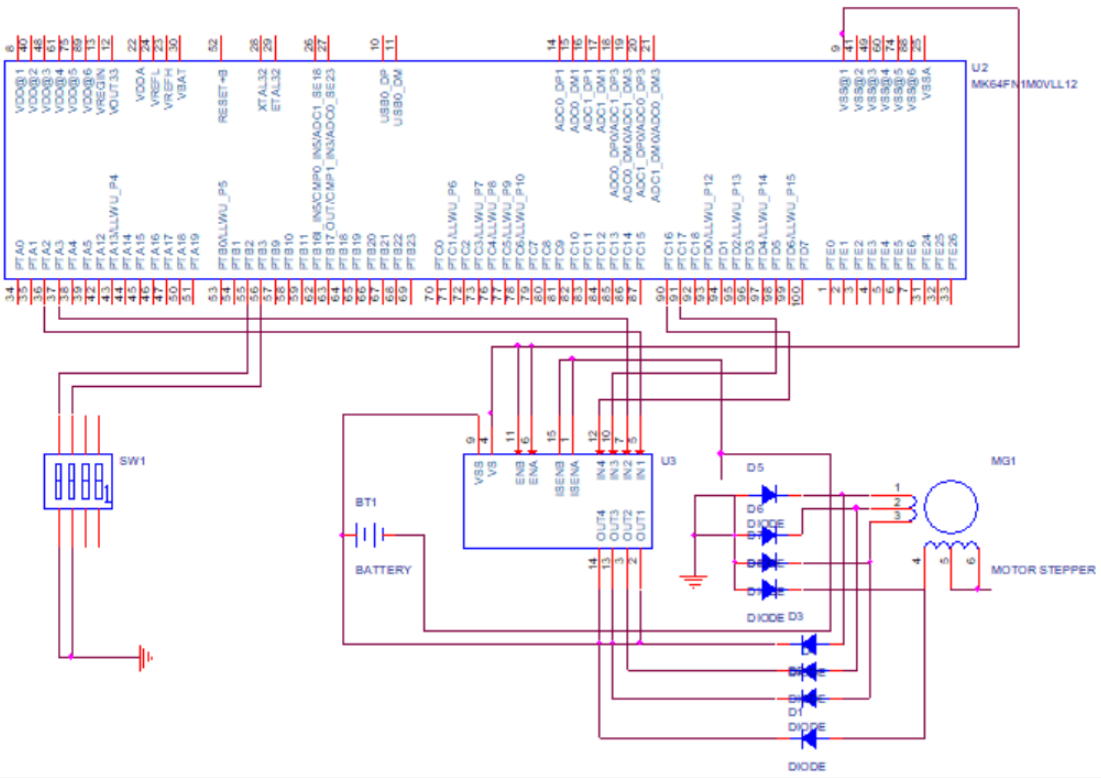


Fig. 3: Schematic Diagram

High-Level Description

The program was designed to control a bipolar stepper motor using the K64f microcontroller and the L298N motor driver. It begins by enabling the necessary port clocks and configuring specific GPIO pins as outputs for the motor driver inputs and as inputs for the DIP switch used to select direction and speed. In the main loop, the microcontroller continuously reads the DIP switch to determine whether the motor should rotate either clockwise or counterclockwise and whether it should operate at a certain speed. These settings control which activation sequence occurs, and the delay applied between steps, effectively determining the rotational direction and angular velocity. The four-step sequence energizes the coils in the correct order to produce smooth

motion in either direction. The system runs in open-loop mode, relying solely on software-controlled GPIO signals to reach consistent stepper motor performance.

Program Listing (Main Sections)

FRDM-K64F Code

```
#include "fsl_device_registers.h"

void delay_ms(int loops) {
    for (volatile int i = 0; i < loops; i++) {
    }
}

int main(void) {
    // Enable clocks for used ports
    SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;
    SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK;
    SIM_SCGC5 |= SIM_SCGC5_PORTC_MASK;

    // L298 input pins
    PORTA_PCR1 = PORT_PCR_MUX(1);
    PORTA_PCR2 = PORT_PCR_MUX(1);
    PORTB_PCR9 = PORT_PCR_MUX(1);
    PORTC_PCR17 = PORT_PCR_MUX(1);

    GPIOA_PDDR |= (1 << 1);
    GPIOA_PDDR |= (1 << 2);
    GPIOB_PDDR |= (1 << 9);
    GPIOC_PDDR |= (1 << 17);

    // DIP switch inputs
    PORTB_PCR2 = PORT_PCR_MUX(1);
    PORTB_PCR3 = PORT_PCR_MUX(1);
    GPIOB_PDDR &= ~(1 << 2);
    GPIOB_PDDR &= ~(1 << 3);

    // Directions and speed
    while (1) {
        int direction = 0;
        int speed = 0;
        int delay_val = 0;

        if (GPIOB_PDIR & (1 << 2)) {
            direction = 1;
        }
```

```
else {
   direction = 0;
}

if (GPIOB_PDIR & (1 << 3)) {
   speed = 1;
}

else {
   speed = 0;
}

if (speed == 1) {
   delay_val = 5000;
}

else {
   delay_val = 40000;
}

// if direction = 0 (Clockwise), if direction = 1 (counter clockwise)
if (direction == 0) {
   // Clockwise
   GPIOA_PSOR = (1 << 1);
   GPIOA_PCOR = (1 << 2);
   GPIOB_PSOR = (1 << 9);
   GPIOC_PCOR = (1 << 17);
   delay_ms(delay_val);

   GPIOA_PCOR = (1 << 1);
   GPIOA_PSOR = (1 << 2);
   GPIOB_PSOR = (1 << 9);
   GPIOC_PCOR = (1 << 17);
   delay_ms(delay_val);

   GPIOA_PCOR = (1 << 1);
   GPIOA_PSOR = (1 << 2);
   GPIOB_PCOR = (1 << 9);
   GPIOC_PSOR = (1 << 17);
   delay_ms(delay_val);

   GPIOA_PSOR = (1 << 1);
   GPIOA_PCOR = (1 << 2);
   GPIOB_PCOR = (1 << 9);
   GPIOC_PSOR = (1 << 17);
   delay_ms(delay_val);
}

else {
   // Counterclockwise
   GPIOA_PSOR = (1 << 1);
```

```
            GPIOA_PCOR = (1 << 2);
            GPIOB_PCOR = (1 << 9);
            GPIOC_PSOR = (1 << 17);
            delay_ms(delay_val);

            GPIOA_PCOR = (1 << 1);
            GPIOA_PSOR = (1 << 2);
            GPIOB_PCOR = (1 << 9);
            GPIOC_PSOR = (1 << 17);
            delay_ms(delay_val);

            GPIOA_PCOR = (1 << 1);
            GPIOA_PSOR = (1 << 2);
            GPIOB_PSOR = (1 << 9);
            GPIOC_PCOR = (1 << 17);
            delay_ms(delay_val);

            GPIOA_PSOR = (1 << 1);
            GPIOA_PCOR = (1 << 2);
            GPIOB_PSOR = (1 << 9);
            GPIOC_PCOR = (1 << 17);
            delay_ms(delay_val);
        }
    }
}
```

Technical Problems Encountered and Solutions

One major technical problem we encountered was the stepper motor vibrating back and forth rather than moving in one direction. The solution we found was to switch the diode's polarity, as the positive terminal was facing the stepper motor wire and not towards ground.

Questions

1. Can a stepper motor change its speed from zero to a high value instantly? Also, can a stepper motor switch its direction while running at high speed? Answer with a brief explanation. (3 points)

   No, a stepper motor cannot go from zero to a high value instantly since it has inertia that prevents it from doing so. You would have to apply some force to the motor to speed it up to a high value. It would be hard to switch direction at high speed because you would have to slow down one way, then speed up another.

2. Suppose that there is a 4-phase stepper motor, as shown in the right figure. The rotor magnet is assumed to have 2 poles.

   a) Write a table of clockwise stepper control steps in “one phase on, full step” mode. (4 points)

Step	A1	A2	B1	B2	C1	C2	D1	D2
1	+	-	-	-	-	-	-	-
2	-	-	+	-	-	-	-	-
3	-	-	-	-	+	-	-	-
4	-	-	-	-	-	-	+	-

b) Write a table of clockwise stepper control steps in “two-phase on, full step” mode. (4 points)

Step	A1	A2	B1	B2	C1	C2	D1	D2
1	+	-	+	-	-	-	-	-
2	-	-	+	-	+	-	-	-
3	-	-	-	-	+	-	+	-
4	+	-	-	-	-	-	+	-

c) Write a table of clockwise stepper control steps in “half-stepping” mode. (4 points)

Step	A1	A2	B1	B2	C1	C2	D1	D2
1	+	-	-	-	-	-	-	-
2	+	-	+	-	-	-	-	-
3	-	-	+	-	-	-	-	-
4	-	-	+	-	+	-	-	-
5	-	-	-	-	+	-	-	-
6	-	-	-	-	+	-	+	-
7	-	-	-	-	-	-	+	-
8	+	-	-	-	-	-	+	-

Conclusions

In this lab, a complete stepper motor control system was successfully developed and tested using the K64F board and L298N module. The program accurately controlled motor direction and speed through DIP switch inputs and proper step sequencing. The experiment enhanced motor control and delay-based timing using a K64F microcontroller. Overall, the design achieved reliable performance in all operating modes, showing control of both directions of the stepper motor.

Team Contribution Summary

We worked together in most parts of the lab. Both of us helped set up the circuit and test the hardware. I (Muneer) worked on part of the K64F code, and Sameer worked on another part of the code. We also both helped with debugging and verifying the motor. Sameer worked on the Report.