MECH 421 Lab 5: PCB Assembly and Stepper Motor Control

Student Name 1 Student Name 2

November 21, 2025

Abstract

This report details the assembly and testing of a custom Printed Circuit Board (PCB) for mechatronic applications, specifically focusing on stepper motor control. The lab is divided into three exercises: PCB assembly, single-axis stepper motor control, and dual-axis gantry control. The objective is to provide a comprehensive guide for future students to replicate the assembly and control logic without external references.

Contents

1 Introduction

The purpose of this lab is to gain hands-on experience in PCB assembly, soldering surface-mount components, and implementing real-time control for stepper motors using a microcontroller. The final system is a 2-axis gantry capable of drawing complex shapes, demonstrating the integration of hardware (PCB, motors, mechanics) and software (firmware, PC interface).

2 Exercise 1: PCB Assembly and Soldering

2.1 Objective

The goal of this exercise is to populate a bare PCB with surface-mount and through-hole components to create a functional motor control board. This board includes a microcontroller (MSP430), USB interface (FTDI), power regulation, and motor drivers.

2.2 Theory and Circuit Description

The PCB consists of several key functional blocks:

- **Power Supply**: Converts external 12V DC input to 5V and 3.3V logic levels using linear regulators (NCP1117). 5V is used for the USB interface and some logic, while 3.3V powers the MSP430 microcontroller.
- **USB Interface**: Uses an FT230XS chip to convert USB signals from a PC into UART (Serial) signals compatible with the microcontroller. This allows for data logging and control commands.
- **Microcontroller**: The MSP430FR5739 is the brain of the board. It executes firmware to generate PWM signals for motors, read sensors, and communicate via UART.
- **Motor Driver**: The DRV8841 is a dual H-bridge driver capable of driving DC motors or bipolar stepper motors. It handles the high currents required by the motors, controlled by low-power logic signals from the MCU.

2.3 Parts List

The following components are required for assembly. Ensure all parts are accounted for before starting.

Component	Description	Qty
MSP430FR5739	Microcontroller (TSSOP-38)	1
FT230XS	USB-to-UART Bridge (SSOP-16)	1
DRV8841	Motor Driver (HTSSOP-28)	1
NCP1117-3.3	3.3V Regulator (SOT-223)	1
NCP1117-5.0	5.0V Regulator (SOT-223)	1
Resistors	Various (0603/1206 SMD)	Kit
Capacitors	Various (0603/1206 SMD)	Kit
Connectors	USB Mini-B, DC Jack, Headers	Kit
LEDs	Green (1206)	6

Table 1: Major Components List

2.4 Assembly Procedure

The assembly should follow a specific order to ensure testability at each stage.

2.4.1 Step 1: Underside Resistors

Solder the 150 Ω resistors on the bottom side of the PCB. These limit current to the MCU pins.

- **Tip**: Use flux and a clean iron tip. Tin one pad, slide the resistor in, then solder the other side.

2.4.2 Step 2: Power Supply

Install the DC jack, diodes, and voltage regulators (3.3V and 5V).

- **Verification**: Plug in a 12V supply. Measure the output of the regulators with a multimeter. You should see stable 3.3V and 5V rails. The Power LED should light up.

2.4.3 Step 3: USB Interface

Solder the FT230XS chip, USB connector, and associated passives.

- **Caution**: The FT230XS has fine-pitch pins. Use plenty of flux and drag solder if necessary. Check for bridges.
- **Verification**: Connect to a PC via USB. The PC should recognize the device (e.g., as a COM port). If the chip gets hot, disconnect immediately.

2.4.4 Step 4: Microcontroller

Solder the MSP430FR5739 and the JTAG header.

- **Verification**: Connect the JTAG programmer. Attempt to flash a "Blink LED" program. If successful, the MCU is operational.

2.4.5 Step 5: Motor Driver

Solder the DRV8841 and the large bulk capacitors.

- **Note**: The thermal pad under the chip must be soldered for heat dissipation (via the hole on the back).

2.5 Results and Discussion

PLACEHOLDER FOR: Assembled PCB (Front)

Insert a clear photo of the fully assembled PCB (Front View).

Figure 1: Assembled PCB (Front)

PLACEHOLDER FOR: Assembled PCB (Back)

Insert a clear photo of the fully assembled PCB (Back View).

Figure 2: Assembled PCB (Back)

Challenges Encountered: (Example: We encountered a solder bridge on the FTDI chip pins, which prevented USB recognition. This was resolved using solder wick and additional flux.)

3 Exercise 2: Stepper Motor Control

3.1 Objective

To control a bipolar stepper motor using the assembled PCB. This involves generating precise PWM waveforms to drive the motor phases in a specific sequence (half-stepping) and implementing a UART interface for velocity control.

3.2 Theory

3.2.1 Stepper Motor Operation

A bipolar stepper motor has two coils (Phase A and Phase B). By energizing these coils in a specific sequence, the rotor aligns with the magnetic field, moving in discrete steps.

- **Full-Stepping**: Energizing phases in sequence (A+ $\rightarrow$ B+ $\rightarrow$ A- $\rightarrow$ B-).
- **Half-Stepping**: Inserting intermediate states where both phases are energized, doubling the resolution (A+ $\rightarrow$ A+B+ $\rightarrow$ B+ ...).

3.2.2 Current Control (PWM)

To prevent overheating, the voltage applied to the coils is modulated using Pulse Width Modulation (PWM). A duty cycle of roughly 25% is sufficient for no-load operation.

3.3 Firmware Implementation

The firmware uses a state machine and a look-up table to determine the coil excitation for each step.

- **Look-up Table**: An array storing the PWM duty cycle values (or enable bits) for the 8 half-step states.
- **Timer Interrupt**: A timer interrupt triggers the next step. The frequency of this interrupt determines the motor speed.
- **UART Command**: The PC sends a velocity command (byte). The MCU maps this byte to a timer period (CCR0 value).

PLACEHOLDER FOR: Stepper State Table

Insert the Half-Step State Table used in your code.

Figure 3: Stepper State Table

3.4 Software Interface (C#)

A C# GUI was developed to send commands. It features:

- **Slider**: Sets velocity (Center = 0, Left = CCW Max, Right = CW Max).
- **Buttons**: Trigger single steps.

3.5 Results

PLACEHOLDER FOR: C# GUI Screenshot

Insert a screenshot of the C# application controlling the motor.

Figure 4: C# GUI Screenshot

PLACEHOLDER FOR: Packet Structure

Insert a diagram of the UART packet structure (Start Byte, Command, Data, Checksum).

Figure 5: Packet Structure

PLACEHOLDER FOR: Electrical Schematic

Insert the schematic of the minimum components to drive the stepper (MCU + Driver + Motor).

Figure 6: Electrical Schematic

Speed Measurements:

- **Max No-Load Speed**: [Value] steps/s
- **Max Loaded Speed**: [Value] steps/s

Discussion: The maximum speed is limited by the inductance of the motor coils, which opposes rapid current changes. As speed increases, the current doesn't have enough time to reach the target level within a step period, reducing torque until the motor stalls.

4 Exercise 3: 2-Axis Control with Dual Stepper Motors

4.1 Objective

To extend the system to 2 axes (X and Y) for a gantry stage. This requires driving a second stepper motor using an external H-bridge driver wired to the PCB.

4.2 Procedure

4.2.1 Wiring the Second Motor

Since the PCB has only one DRV8841, a second external driver is used.

- **Connections**: The external driver inputs are connected to available GPIO pins on the MCU.
- **Power**: Shared 12V rail.

4.2.2 Coordinated Motion

To draw shapes, both motors must move simultaneously.

- **Algorithm**: The MCU receives a target coordinate (X, Y). It calculates the number of steps for each axis. To move in a straight line, the step rates are synchronized such that both axes finish at the same time.

4.3 Results

PLACEHOLDER FOR: Standard Shapes Plot

Insert photo of the paper with the 6 test points/lines drawn.

Figure 7: Standard Shapes Plot

PLACEHOLDER FOR: Creative Shape Plot

Insert photo of the creative shape drawn by the gantry.

Figure 8: Creative Shape Plot

PLACEHOLDER FOR: Creative Shape Design Overlay

Insert photo of the drawn shape with the intended design overlaid for comparison.

Figure 9: Creative Shape Design Overlay

Discussion on Accuracy: Deviations between the design and the result can be attributed to:

- **Backlash**: Play in the belt/pulley system.
- **Missed Steps**: If the motor torque was insufficient for the acceleration.
- **Resolution**: The discrete nature of stepper motors (finite step size).

5 Conclusion

This lab successfully demonstrated the complete process of building a mechatronic controller, from soldering the PCB to implementing low-level motor control firmware and high-level PC software. The final 2-axis gantry system was able to draw complex shapes, validating the integration of all subsystems.

A Microcontroller Code (C)

```
// Insert your C code here
// Example:
// void main(void) {
//     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
//     init_clocks();
//     init_uart();
//     while(1) {
//         // Control loop
//     }
// }
```

Listing 1: MSP430 Firmware Main Loop

B PC Interface Code (C#)

```
// Insert your C# code here
// Example:
// private void serialPort1_DataReceived(object sender,
    SerialDataReceivedEventArgs e) {
//     // Handle incoming data
// }
```

Listing 2: C# Serial Communication Handler