# Stepper motor driver and circuit documentation

By

Will McKelvey
wmckelv99@gmail.com
601-701-7148

Mississippi State, Mississippi

TODO:

Put in more specific controls/commands stuff

Put in more Arduino IDE stuff, CP210x drivers, how to flash esp32, et.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

## 1.1 Overview of Project

This project was developed to support the Institute for Clean Energy Technology's (ICET) research into electrospinning filter materials. Electrospinning is a critical process in creating high-performance filter media, and precise control over the dispensing of fluids is essential for ensuring the quality and consistency of these materials.

Commercial syringe pumps, which are commonly used in this process, can cost upwards of $500, making them expensive for many research and development applications. To address this issue, the project aims to design and build a cost-effective syringe pump using readily available components, including a stepper motor, ESP32 microcontroller, and a TMC2209 driver.

This custom syringe pump is designed to provide the same level of precision and control as commercial units but at a fraction of the cost. By utilizing open-source hardware and software, this project also offers flexibility for customization and adaptation to various research needs, making it an invaluable tool for ongoing and future projects within ICET.

## 1.2 Objectives

The primary objectives of this project are as follows:

1. **Design and Develop a Cost-Effective Syringe Pump:**

- Create a syringe pump that delivers precise control over fluid dispensing using a stepper motor and open-source hardware, significantly reducing the cost compared to commercial alternatives.

2. **Enhance Research Capabilities:**
    - Support ICET's research in electrospinning filter materials by providing a reliable and customizable tool that meets the specific requirements of electrospinning processes.
3. **Ensure Precision and Accuracy:**
    - Implement control algorithms and calibration procedures to ensure that the syringe pump operates with high precision, enabling consistent and repeatable results in fluid dispensing.
4. **Provide Flexibility and Customization:**
    - Design the system with modular components and open-source software, allowing for easy customization and adaptation to various research scenarios, including the ability to adjust flow rates, syringe sizes, and other parameters.
5. **Facilitate Easy Assembly and Use:**
    - Develop clear assembly instructions and user-friendly software interfaces, ensuring that the syringe pump can be easily constructed, operated, and maintained by researchers with varying levels of technical expertise.
6. **Integrate Safety Features:**
    - Incorporate essential safety mechanisms, such as an emergency stop function and limit controls, to ensure the safe operation of the syringe pump in a research environment.

7. **Promote Future Scalability and Improvements:**
    - Lay the groundwork for future enhancements by designing a system that can be scaled or modified for larger applications or additional features, such as advanced control options or integration with other laboratory equipment.

# CHAPTER II

# SYSTEM ARCHITECTURE

## 2.1 Block Diagram

The following block diagram in fig. 2.1 provides a high-level overview of the system architecture for the syringe pump. It illustrates the major components involved in the operation of the pump, their interactions, and the flow of control and power within the system.

At the core of the system is the ESP32 microcontroller, which serves as the central control unit. It interfaces with the user through a graphical user interface (GUI) that allows for setting syringe parameters, controlling the flow rate, and starting or stopping the pump. The ESP32 processes these inputs and communicates with the TMC2209 stepper motor driver to control the movement of the stepper motor, which in turn drives the syringe plunger to dispense fluid.

The user controls everything from the laptop, which is connected to the ESP32 via USB. Communication between the ESP32 and the laptop is over serial. The ESP32 is also connected to the TMC2209 using a serial interface. The ESP32 sends commands to the TMC22209 to control the motor.

Additional safety and monitoring features, such as limit switches and an emergency stop button, are integrated into the system to ensure reliable and safe operation. The power supply provides the necessary energy to all components, ensuring consistent performance of the syringe pump.

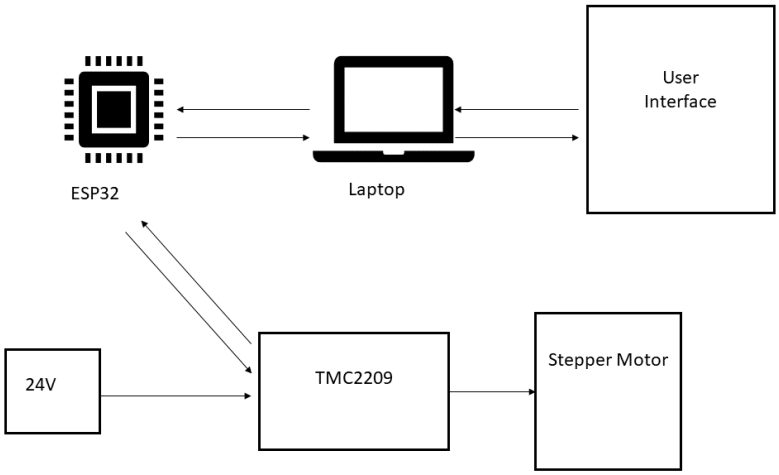


Figure 2.1 Block diagram of stepper controller

## 2.2 System Components

The syringe pump system is composed of several key components, each playing a crucial role in its operation. Below is a detailed overview of these components and their respective functions:

### 2.2.1 ESP32 Microcontroller

- **Function:** The ESP32 serves as the brain of the system, managing the control logic, processing user inputs from the GUI, and sending control signals to the stepper motor driver.
- **Key Features:**
    - Dual-core processor for efficient multitasking.
    - Built-in Wi-Fi and Bluetooth for wireless communication.
    - Multiple GPIO pins for interfacing with sensors, drivers, and other peripherals.

### 2.2.2 TMC2209 Stepper Motor Driver

- **Function:** The TMC2209 driver controls the stepper motor by converting digital control signals from the ESP32 into precise electrical pulses that dictate the motor's movement.
- **Key Features:**
    - StealthChop™ technology for quiet operation.
    - StallGuard™ for sensorless homing and load detection.
    - Configurable via UART interface for easy tuning and setup.

### 2.2.3 Stepper Motor

- **Function:** The stepper motor converts electrical pulses from the TMC2209 driver into mechanical movement, driving the syringe plunger to dispense fluid at controlled rates.
- **Key Features:**
    - High torque and precise step increments for accurate control.
    - Compatible with various motor drivers, including the TMC2209.
    - Suitable for applications requiring fine control over movement.

### 2.2.4 Syringe

- **Function:** The syringe holds the fluid to be dispensed. It is mechanically linked to the stepper motor, which controls the movement of the plunger to regulate fluid flow.
- **Key Features:**
    - Available in various sizes to accommodate different volumes.
    - Made of durable materials compatible with a range of fluids.
    - Easily replaceable for different applications.

### 2.2.5 User Interface (GUI)

- **Function:** The GUI provides an intuitive interface for the user to interact with the system. It allows for the input of syringe parameters, flow rate settings, and real-time control of the syringe pump.
- **Key Features:**
    - User-friendly design with clear visual feedback.
    - Real-time monitoring of system status and operation.
    - Customizable settings to accommodate different research needs.

### 2.2.6 Serial Communication Interface

- **Function:** Facilitates communication between the ESP32 microcontroller and the GUI. This interface transmits user commands to the ESP32 and relays system status and feedback to the GUI.
- **Key Features:**
    - Reliable and fast data transmission.
    - Supports both wired and wireless communication options.
    - Ensures synchronization between user commands and system operation.

### 2.2.7 Power Supply

- **Function:** The power supply provides the necessary electrical power to all system components, ensuring stable and reliable operation.
- **Key Features:**
    - Sufficient power output to support the stepper motor.
    - Stable voltage and current regulation.

# CHAPTER III

# HARDWARE DESIGN

## 3.1 Circuit Schematic

Figure 3.1 below shows the circuit diagram of the stepper controller. It includes the ESP32, the TMC2209, two screw terminals to make connections, and a 1kΩ resistor.

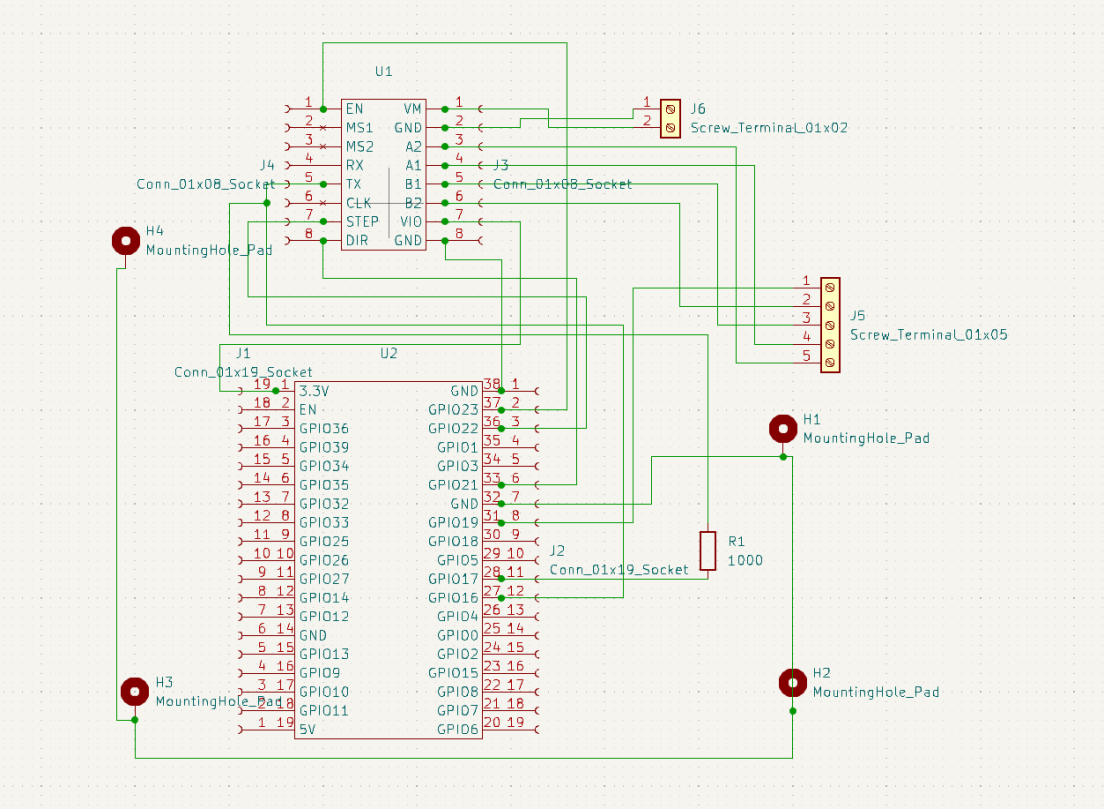


Figure 3.1 Circuit diagram

## 3.2 Component Selection

The components used in the circuit were selected due to their price and familiarity with them. The ESP32 is a lightweight, cheap, powerful, and easy to use microcontroller that is used in many Internet-of-Things (IoT) devices. It offers 32 GPIO connections, 3.3V and 5V outputs, and multiple grounds. The TMC2209 was chosen again for price and familiarity. It is commonly used in 3D printer mainboards to move the axes. It includes technology to use less power and to keep the motor quieter than other stepper drivers. A pinout of both the ESP32 and TMC2209 are shown below in fig. 3.x and fig. 3.x, respectively. Both components were bought from amazon, specifically the HiLetgo ESP-WROOM-32D development board and the BIGTREETECH TMC2209 V1.3.

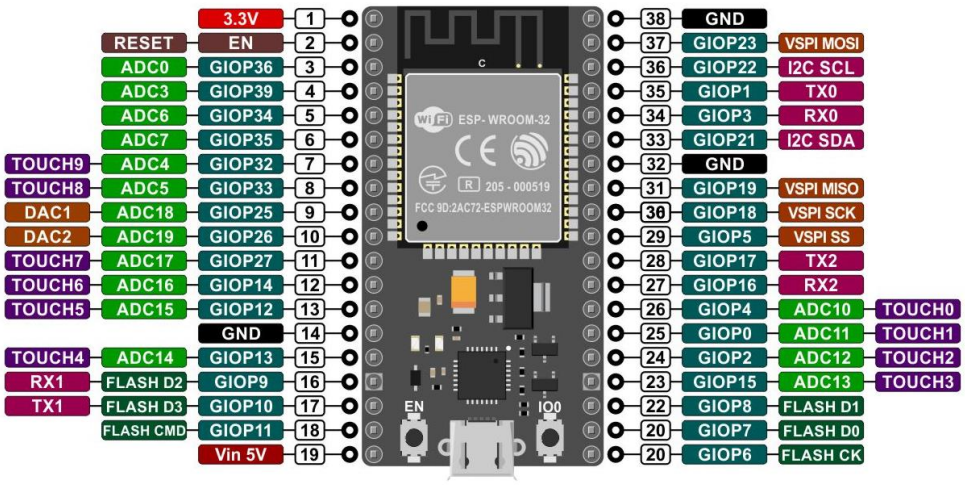


Figure 3.2 ESP32 pinout

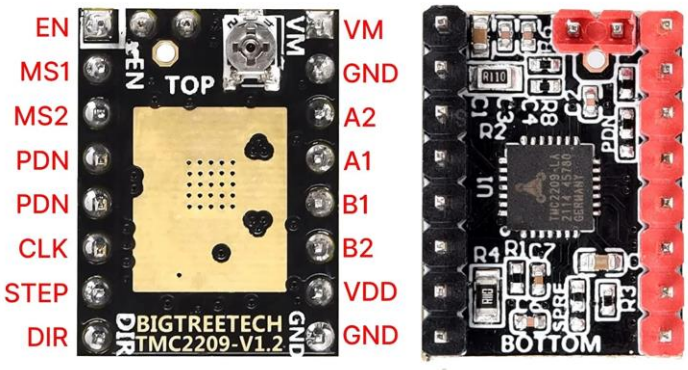


Figure 3.3 TMC2209 pinout

### 3.3 PCB Layout

Based on the circuit layout in section 3.1, the following PCB layout was created using KiCAD. Below in fig. 3.x is the top layer of the PCB. It shows connections from the ESP32 to the TMC2209, along with spots for the resistor and screw terminals. There are also four holes that are meant to screw the board down. These four holes are connected to the ground pin on the ESP32. These are used to connect the endstop to ground so it can signal the ESP32 when it is pressed. The smaller holes in the board that are side-by-side are meant to accept 19-pin and 8-pin headers that are to be soldered on, however the ESP32 and the TMC2209 can also both be soldered on directly. I would recommend that headers be used so components can be easily switched out. Fig. 3.x also shows the PCB in 3D with all the components connected.

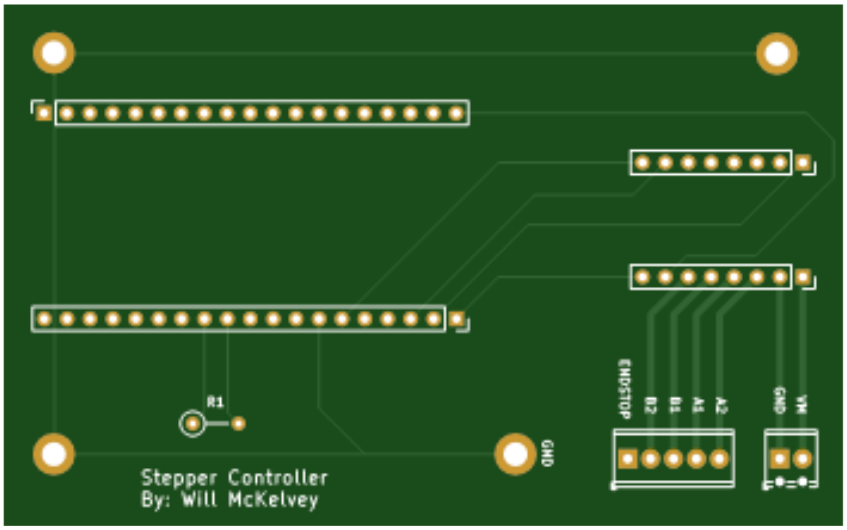


Figure 3.4 Top view of PCB.

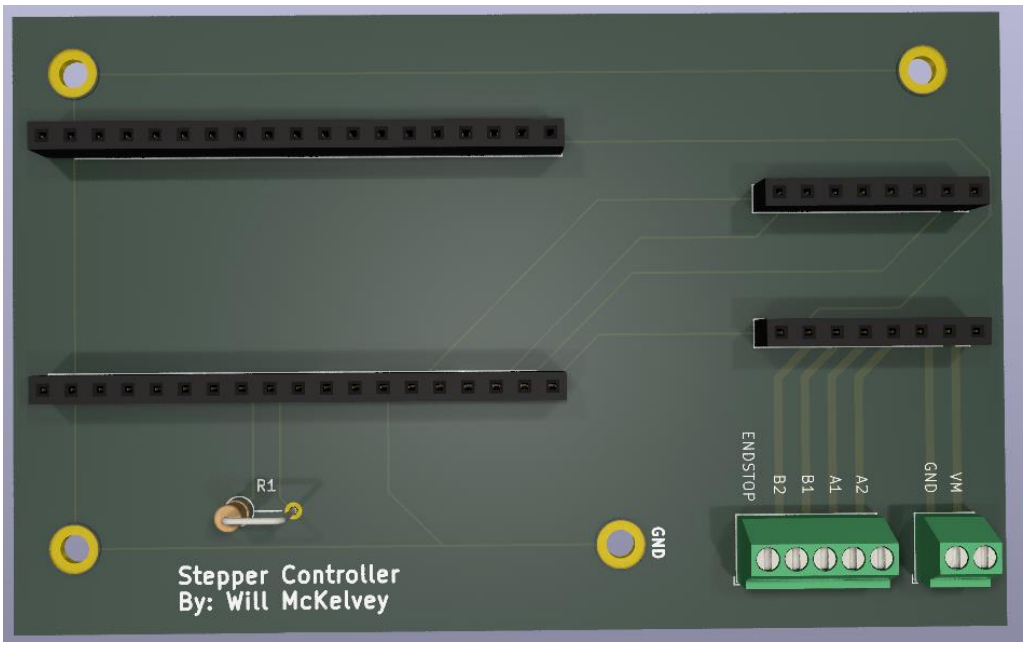


Figure 3.5 3D view of PCB

# CHAPTER IV

# SOFTWARE DESIGN

## 4.1 Firmware Overview

The firmware for the syringe pump system is designed to control the stepper motor with precision, manage user inputs, and ensure safe and reliable operation. The firmware runs on the ESP32 microcontroller and serves as the central control logic for the entire system. Below is an overview of the key components and functionalities of the firmware:

### 4.1.1 System Initialization

The firmware begins by initializing all necessary peripherals and system components. This includes configuring the GPIO pins for various inputs and outputs, such as limit switches and status LEDs. The UART interface is set up for communication with the TMC2209 stepper motor driver, ensuring that control signals can be transmitted accurately. Timers and interrupts are also initialized to provide precise timing for motor control and other critical operations.

### 4.1.2 User Input Handling

The firmware continuously monitors and processes inputs from the GUI, which include settings such as flow rate, syringe parameters, and control commands like start, stop, and emergency stop. These inputs are validated to ensure they are within acceptable ranges and are used to update the system's internal parameters accordingly. The firmware then provides real-

time feedback to the GUI, confirming received commands or alerting the user to any errors or issues that need attention.

### 4.1.3 Stepper Motor Control

At the heart of the firmware's functionality is the control of the stepper motor, which drives the syringe plunger. The firmware calculates the exact number of steps required to achieve the user-defined flow rate and sends these control signals to the TMC2209 driver. The firmware constantly monitors the motor’s position and adjusts the control signals as necessary to maintain accurate fluid dispensing. To ensure smooth and efficient operation, acceleration and deceleration profiles are implemented, preventing abrupt movements that could affect the precision of the fluid flow.

### 4.1.4 Communication with TMC2209

The firmware maintains constant communication with the TMC2209 driver via the UART interface to control the stepper motor. This communication involves sending configuration commands, such as setting current limits or enabling stealth mode for quieter operation. The firmware also reads status information from the TMC2209, including stall detection and motor position, allowing it to make real-time adjustments to the motor control parameters based on feedback from the driver.

### 4.1.5 Feedback and Status Reporting

Real-time feedback is essential for the effective operation of the syringe pump. The firmware provides continuous status updates to the GUI, including the current position of the syringe plunger, the operational status of the motor, and any errors or warnings that arise during

operation. These updates ensure that the user is fully informed of the system's performance and can respond quickly to any issues. Additionally, the firmware updates the status indicators, such as LEDs, to reflect the current state of the system, providing a visual confirmation of its operation.

## 4.2 Serial Communication Protocol

The serial communication protocol is a critical component of the syringe pump system, enabling seamless interaction between the ESP32 microcontroller and the Graphical User Interface (GUI). This protocol ensures that user commands are accurately transmitted to the ESP32, and that real-time feedback is provided to the user, allowing for control and monitoring of the syringe pump's operation.

Serial communication between the ESP32 and TMC2209 is done using a one wire interface. The RX pin of the TMC2209 is connected to the RX pin of the ESP32. The RX pin of the TMC2209 is also connected to the 1kΩ resistor which is then connected to the TX pin of the ESP32.

### 4.2.1 Communication Setup

The serial communication between the ESP32 and the GUI is established using a UART (Universal Asynchronous Receiver-Transmitter) interface. The ESP32 is configured to communicate over a specific baud rate, typically 115200, to ensure a balance between speed and reliability. The communication channel is initialized during the system startup, allowing the ESP32 to listen for incoming commands from the GUI and send status updates in return.

### 4.2.2 Command Structure

The protocol uses a structured format for sending and receiving commands. Each command sent from the GUI to the ESP32 follows a predefined structure, starting with a unique code that identifies the type of command, followed by one or more parameters depending on the command. A special character is used to signal the end of the command, ensuring that the ESP32 correctly interprets the complete instruction. For example, a command might instruct the ESP32 to set a specific flow rate.

### 4.2.3 Command Parsing and Validation

Upon receiving a command, the ESP32 firmware parses the input to extract the command identifier and any associated parameters. The firmware then validates the command, checking for correct syntax and ensuring that the parameters fall within acceptable ranges. If the command is valid, the firmware executes the corresponding action, such as adjusting the motor speed to achieve the desired flow rate. If the command is invalid, the ESP32 sends an error message back to the GUI, indicating the issue.

### 4.2.4 Feedback and Acknowledgment

After processing a command, the ESP32 sends an acknowledgment back to the GUI to confirm that the command was successfully received and executed. This acknowledgment may include the current status of the system, or any changes made as a result of the command. For instance, the ESP32 might confirm that the flow rate has been set to the desired value. If there was an error, the response would indicate the nature of the issue, such as an invalid flow rate.

### 4.2.5 Continuous Status Updates

In addition to responding to specific commands, the ESP32 periodically sends status updates to the GUI, providing real-time information about the syringe pump's operation. These updates might include the current position of the syringe plunger, the operational status of the motor, and any errors or warnings detected. These updates help the user monitor the system's performance and take corrective action if necessary.

## 4.3 Motor Control

Motor control is at the core of the syringe pump system, enabling precise movement of the stepper motor to accurately dispense fluid. The firmware's motor control logic is designed to ensure smooth operation, with a focus on precision, reliability, and safety.

### 4.3.1 System Initialization and Configuration

During system initialization, the ESP32 configures the TMC2209 stepper motor driver, setting parameters such as current limits, microstepping, and operation modes (e.g., StealthChop™ for silent operation). This initial configuration is crucial for ensuring that the motor operates within its optimal performance range and can handle the required load without overheating or stalling.

### 4.3.2 Step Calculation and Motion Planning

When the user sets a desired flow rate through the GUI, the firmware calculates the corresponding number of steps the stepper motor must take to move the syringe plunger at that rate. The calculation considers factors such as the syringe diameter and the desired volume of

fluid to be dispensed. The firmware then plans the motor's motion, determining the sequence of steps needed to achieve the most accurate movement possible.

### 4.3.3 Acceleration and Deceleration Profiles

To prevent sudden starts and stops that could lead to inaccuracies or mechanical stress, the firmware implements acceleration and deceleration profiles. These profiles gradually ramp up the motor’s speed at the beginning of a movement and gradually slow it down at the end. This not only improves the precision of fluid dispensing but also extends the lifespan of the motor and mechanical components by reducing wear and tear.

### 4.3.4 Real-Time Motor Control

During operation, the firmware continuously monitors the motor’s position, adjusting the control signals in real time to maintain the desired speed and position accuracy. This is important in applications where precise fluid volumes must be dispensed, as any deviation in motor position could lead to incorrect dosing. The firmware ensures that the motor follows the planned motion profile exactly, making real-time corrections as needed based on feedback from the motor driver.

# CHAPTER V

# USER INTERFACE DESIGN

## 5.1 GUI Overview

The Graphical User Interface (GUI) for the syringe pump system is a critical component that allows users to interact with and control the pump with ease and precision. Developed using C# with Windows Presentation Foundation (WPF), the GUI offers a user-friendly interface tailored to the needs of researchers and operators. The GUI can be seen below in fig. 5.x.

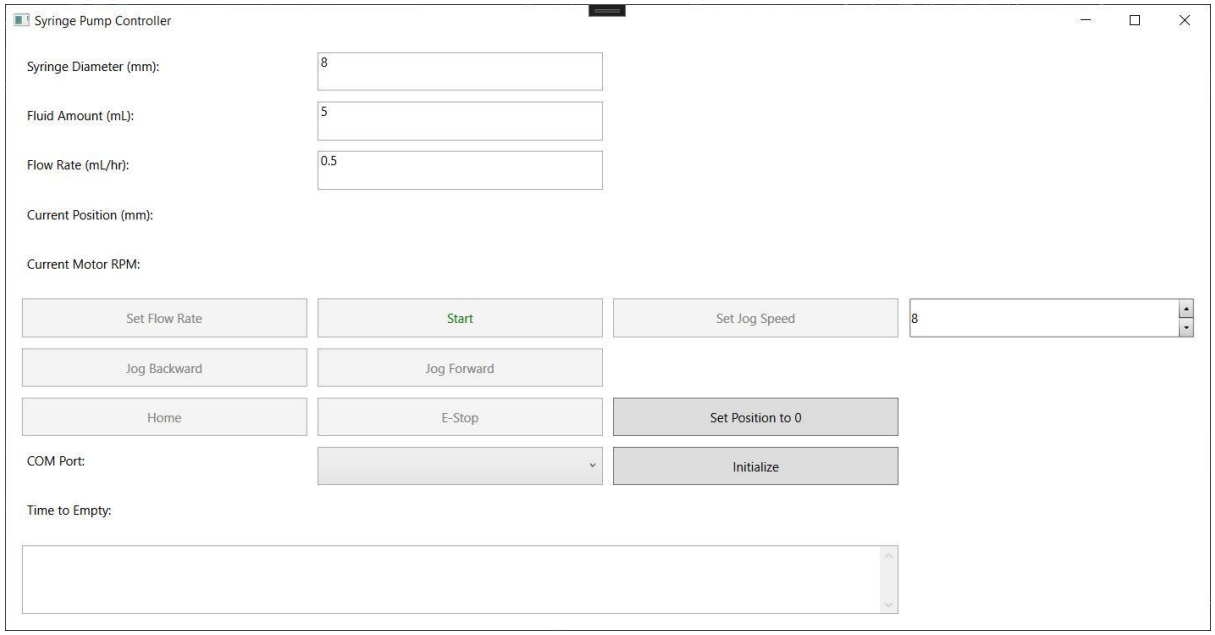


Figure 5.1 GUI

The GUI provides a clear and intuitive layout where users can input parameters such as syringe diameter, fluid volume, and desired flow rate. It also includes controls for starting and

stopping the pump, as well as for adjusting settings like motor speed. Communication between the GUI and the ESP32 microcontroller is handled through serial communication. When a user enters a command or changes a setting within the GUI, the software packages this input into a structured command message, which is then transmitted via the serial interface to the ESP32. The ESP32 processes these commands in real time, adjusting the motor's operation accordingly or performing other tasks as required.

The GUI is also responsible for receiving and displaying feedback from the ESP32. This feedback includes real-time status updates, such as the current position of the syringe plunger, the motor's operational state, and any error messages or alerts. These updates are displayed within the GUI in an accessible format, enabling users to monitor the system's performance and make informed decisions during operation.

To ensure reliability and user safety, the GUI includes several built-in safeguards. For example, the interface may include warning prompts if a user attempts to set a parameter outside of the safe operating range. Additionally, critical functions like the emergency stop are prominently displayed and can be activated immediately from within the GUI, allowing users to halt the pump's operation instantly if needed.

Overall, the GUI's combination of WPF's graphical capabilities and C#'s backend functionality makes it a powerful tool for controlling the syringe pump system. It bridges the gap between the user and the underlying hardware, providing an efficient means of managing the pump's operation while ensuring precision, and ease of use.

## 5.2 Communication with Microcontroller

The communication between the GUI and the ESP32 microcontroller is the backbone of the syringe pump system, enabling precise control and real-time feedback. This communication is facilitated through a serial interface, where commands and status updates are exchanged in a structured and reliable manner.

When a user interacts with the GUI—whether by setting a parameter, starting the pump, or issuing any other command—the action triggers the creation of a command message. This message is formatted according to a predefined protocol that ensures both the GUI and the microcontroller can correctly interpret the instructions. Each command message typically consists of a command identifier, which specifies the action to be taken, followed by any necessary parameters, such as flow rate or syringe size. The command is then terminated with a special end character, signaling the completion of the message to the microcontroller.

Consider the homing command as an example. Homing is essential for resetting the syringe plunger to a known position, typically at the start of the syringe. When the user clicks the "Home" button on the GUI, the software generates a homing command and sends it to the ESP32. Upon receiving this command, the microcontroller initiates the homing routine, driving the stepper motor in a specific direction until a limit switch is triggered, indicating that the plunger has reached the home position. The ESP32 then stops the motor and updates the system's internal position counter to reflect the new, known position of the syringe plunger.

After executing the command, whether it’s setting a flow rate or performing homing, the ESP32 generates a response message to be sent back to the GUI. This response typically includes an acknowledgment of the received command, along with any relevant status updates or error

messages. For example, after a successful homing operation, the response might confirm that the homing process is complete and indicate that the system is ready for the next operation.

In addition to handling individual commands, the communication protocol also supports continuous status updates. These updates allow the GUI to receive real-time data from the microcontroller, such as the current position of the syringe plunger, motor status, and any alerts. The ESP32 periodically sends these updates without requiring a specific request from the GUI, ensuring that the user always has the latest information on the system's performance.

The communication framework between the GUI and the microcontroller allows for smooth operation of the syringe pump system. By ensuring that commands are transmitted and executed accurately, and that feedback is promptly provided to the user, this communication link is essential for maintaining the precision and reliability required for advanced fluid dispensing tasks.

# CHAPTER VI

# ASSEMBLY AND TESTING

## 6.1 Assembly Instructions

There are multiple components that need to be put in place before the pump is ready for use. There are three main pieces to the syringe pump. These include the linear slide and lead screw, the V-blocks that hold the syringe, and the motor and motor controller. All three are mounted to aluminum extrusion to hold everything in place.

### 6.1.1 Lead Screw and Linear Slide

The linear slide and lead screw were bought off amazon, and the lead screw is a T8 thread, and can be seen in fig 6.x. This means that for every one revolution of the screw, the threaded block moves 8mm.

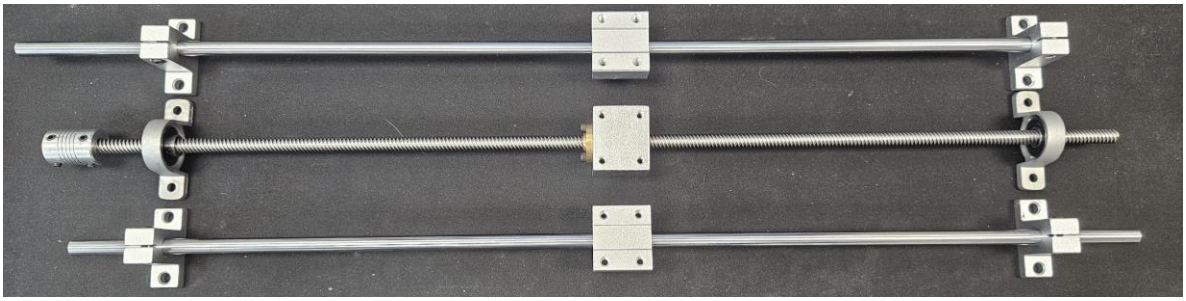

Figure 6.1 Lead screw and linear slides.

### 6.1.2 V-Block

The V-block was 3D printed and consist of four distinct parts as seen in fig. 6.x. There is the stationary block with its holder plate, and the moving block with its holder plate. Both blocks have holes within them to accommodate threaded inserts. These can be pressed into place with a soldering iron to ensure a good fit. The holder plates can then be screwed into place using the threaded insert and an M4 screw. This will hold the syringe while pumping is taking place. The stationary block also has a place for threaded inserts on the bottom side of it so it can be held in place while pumping. The moving block can then be screwed into the bearings and threaded block on the linear slide.

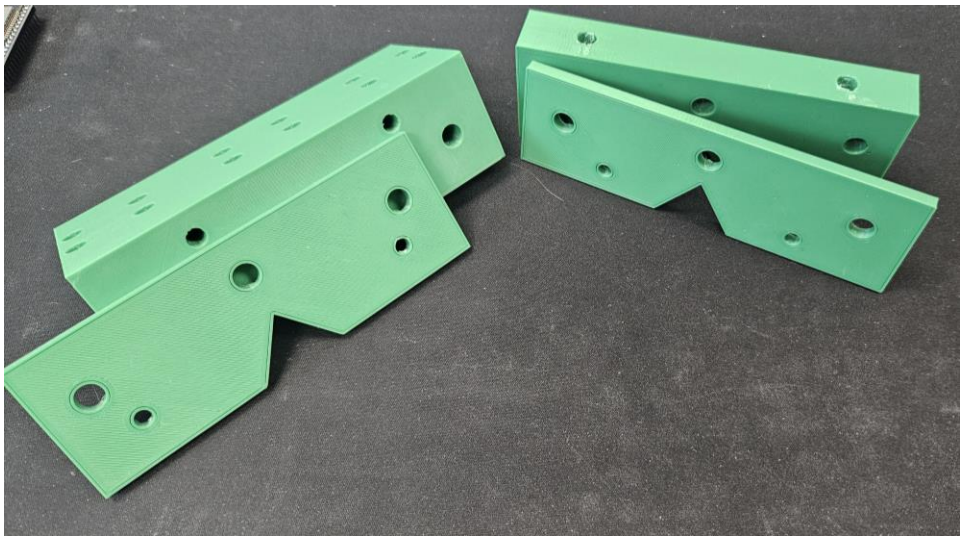

Figure 6.2 V-Blocks

### 6.1.3 Motor and Circuit

The last main part of the syringe pump is the motor and its associated circuit. The syringe pump motor was designed to be a NEMA 17, 1.8deg per step, stepper motor, however this can be changed to any number of different stepper motors if it is a bidirectional motor. Steppers can come with many different gearboxes if slower flowrates or higher step accuracy is required. The

circuit and stepper can be seen below in fig 6.x and fig 6.x, respectively. The motor is connected to the lead screw using a shaft coupling.

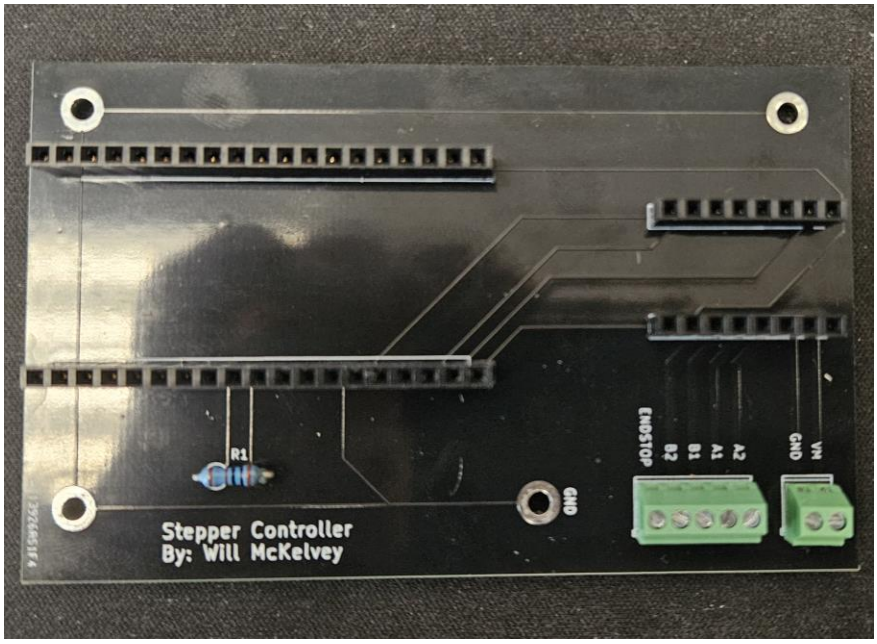


Figure 6.3 Stepper circuit

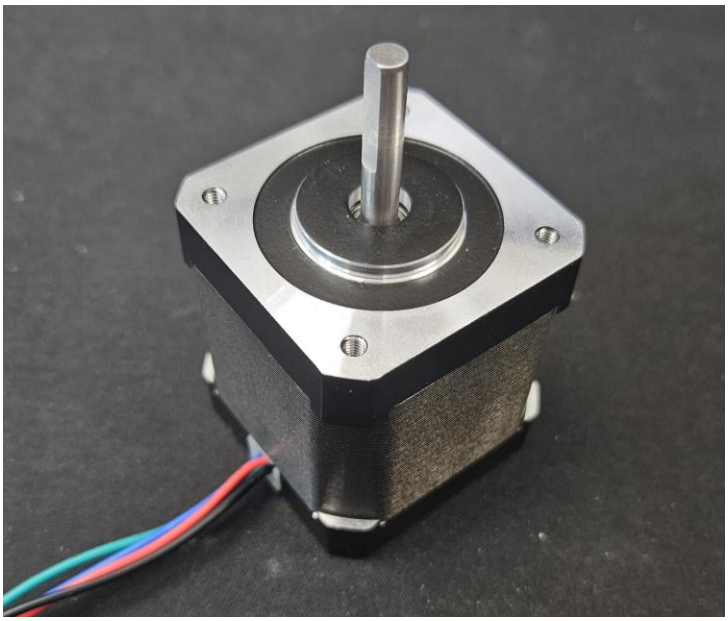

Figure 6.4 Stepper motor

## 6.2 Troubleshooting Guide

Some common problems that you may run into include, the motor spinning the wrong direction, the motor vibrating and making a loud noise but not spinning, the motor being loud and intermittently spinning, etc. This section should help alleviate some of those problems if they should occur.

### 6.2.1 Motor Spinning the Wrong Way

After first wiring up the motor, you may encounter the motor spinning the wrong direction. One of the main reasons for this is the wiring is wrong. Many companies use different color wires for each coil on the stepper motor. The motor in the figure above has four different wires, green, black, blue, and red. These each correspond to half of a coil, which is determined by the manufacturer. In the case of the aforementioned stepper, black is A+, green is A-, red is B+, and blue is B-. The A and B plus or minus correspond to the A1 and A2, and B1 and B2, on the stepper controller circuit board. One must take care to check the manufacturer specifications sheet to make sure they have everything wired correctly to the circuit board. If a pair of wires is switched, it can cause the motor to spin the wrong direction.

Table 6.1 Full Step 2-Phase example

| Step | A+ | B+ | A- | B- |
|---|---|---|---|---|
| 1 | + | + | - | - |
| 2 | - | + | + | - |
| 3 | - | - | + | + |
| 4 | + | - | - | + |

Moving down each step makes the motor turn clockwise, and moving up each step makes the motor move counter clockwise.

### 6.2.2 Motor Vibrating but not Spinning

Another common issue is the motor will vibrate and make loud noises but will not spin. This can be caused by two issues; one is a wire may have come loose from the motor to the circuit board, and the other is the speed is set too high. These can be alleviated relatively easily by making sure all the connections are tight and secured, or by simply lowering the speed or setting a lower max speed in firmware.

### 6.2.3 Motor vibrating and spinning intermittently

Like above in section 6.2.2, one cause for this issue is the motor speed being set too high. The speed is right on the cusp  of being low enough to not effect the motor, but the step count is just too high for the motor. This can be resolved by setting the speed lower in the firmware.

### 6.2.4 Endstop triggering without being pressed

A common issue with configuring the endstop for homing is mistakenly setting it to trigger when it reads LOW on the GPIO pin, when it should trigger when it reads HIGH. The endstop switch is Normally Closed (NC), meaning that when the endstop isn’t pressed, the circuit is complete, and the GPIO pin reads LOW. When the endstop is pressed, the circuit opens, causing the GPIO pin to read HIGH.

To correct this issue, simply invert the logic so that homing triggers on a HIGH reading instead of a LOW one. This adjustment ensures that the homing process is triggered correctly by the endstop's state.

# CHAPTER VII

# FUTURE IMPROVEMENTS

## 7.1 Potential Enhancements

While the current design of the syringe pump system offers precise control and reliable operation, there are several potential enhancements that could further improve its performance, functionality, and versatility. These enhancements could be implemented in future iterations of the project, providing additional capabilities that cater to more advanced research and operational needs.

### 7.1.1 Closed-Loop Stepper Motor Control

One of the most significant upgrades would be integrating a closed-loop stepper motor control system. In the current setup, the system uses open-loop control, where the microcontroller sends step commands to the motor without direct feedback on the motor's actual position. By incorporating a closed-loop system with an encoder, the exact position of the stepper motor can be continuously monitored and verified. This would allow for real-time corrections if the motor misses steps or encounters unexpected resistance, significantly increasing the accuracy and reliability of the fluid dispensing process.

### 7.1.2 Advanced Feedback and Monitoring Systems

Another potential enhancement could involve adding more sophisticated feedback and monitoring systems. This might include integrating sensors to monitor the fluid pressure,

temperature, or even the viscosity of the fluid being dispensed. Such sensors would allow the system to make real-time adjustments based on the properties of the fluid, improving the consistency and quality of the output, especially in complex or sensitive applications.

### 7.1.3 Integration with IoT and Cloud Services

Another potential improvement is integrating the syringe pump system with IoT (Internet of Things) and cloud services. This would enable remote monitoring and control of the pump from any location, as well as the ability to log data to the cloud for long-term storage and analysis. Such integration would be particularly useful in research settings where tracking and analyzing large amounts of data over time is critical.

# REFERENCES AND RESOURCES

# APPENDIX A

# GLOSSARY

**Closed-Loop Control:** A control system that continuously monitors and adjusts its output based on feedback from the system to ensure accuracy and stability. In the context of a stepper motor, it refers to using an encoder to track the motor's position and make real-time corrections.

**Encoder:** A device that provides feedback on the position, speed, or direction of a motor or other moving component. In a closed-loop control system, an encoder helps ensure precise positioning by reporting the motor's actual position back to the controller.

**ESP32 Microcontroller:** A powerful, low-cost microcontroller with built-in Wi-Fi and Bluetooth capabilities, used for managing control logic, processing inputs, and communicating with other devices in the system.

**Graphical User Interface (GUI):** A visual interface that allows users to interact with the system, input parameters, and control operations. In this project, the GUI is developed using C# with Windows Presentation Foundation (WPF).

**Homing:** A process in which a machine or device returns to a known reference position, often at the start of operation. In this project, homing involves moving the syringe plunger to a predefined start position using a stepper motor and limit switch.

**Internet of Things (IoT):** A network of physical devices connected to the internet, allowing them to collect and exchange data. Integration with IoT in the syringe pump system would enable remote monitoring and control.

**Limit Switch:** A type of switch that detects the physical limits of movement in a machine or device. When triggered, it signals the system to stop or reverse movement to prevent damage or overtravel.

**Motor Driver:** An electronic component that controls the power and movement of a motor. In this project, the TMC2209 stepper motor driver converts digital control signals from the microcontroller into precise electrical pulses to drive the stepper motor.

**Normally Closed (NC) Switch:** A type of switch that completes a circuit when it is in its default, unpressed state. Pressing the switch opens the circuit, causing it to break the connection.

**Open-Loop Control:** A control system where the controller sends commands to the motor without receiving feedback on the actual position or status of the motor. Unlike closed-loop control, it does not adjust based on the system's current state.

**Serial Communication:** A method of communication where data is transmitted one bit at a time over a single communication line. In this project, serial communication is used between the GUI and the ESP32 microcontroller.

**Stepper Motor:** A type of motor that moves in discrete steps, allowing for precise control over its position. It is commonly used in applications requiring accurate positioning, such as driving the syringe plunger in this project.

**TMC2209 Stepper Motor Driver:** A motor driver used to control the stepper motor in this project. It features advanced technologies like StealthChop™ for quiet operation and StallGuard™ for sensorless homing and load detection.

**UART (Universal Asynchronous Receiver-Transmitter):** A hardware communication protocol used for asynchronous serial communication between devices. It is used in this project for communication between the ESP32 microcontroller and the TMC2209 motor driver.

**Windows Presentation Foundation (WPF):** A graphical subsystem for rendering user interfaces in Windows-based applications. WPF is used in this project to develop the GUI for the syringe pump system.

# APPENDIX B

LINKS

**Circuit Pieces**

- HiLetgo ESP32
- BIGTREETECH TMC2209
- 19-pin female header
- 8-pin female header
- 5-pin screw terminal
- 2-pin screw terminal
- PCBWay for circuit board

**Mechanical Pieces**

- STEPPERONLINE Nema 17 Stepper Motor
- Lead Screw and Linear Rod Kit
- Shaft Coupler