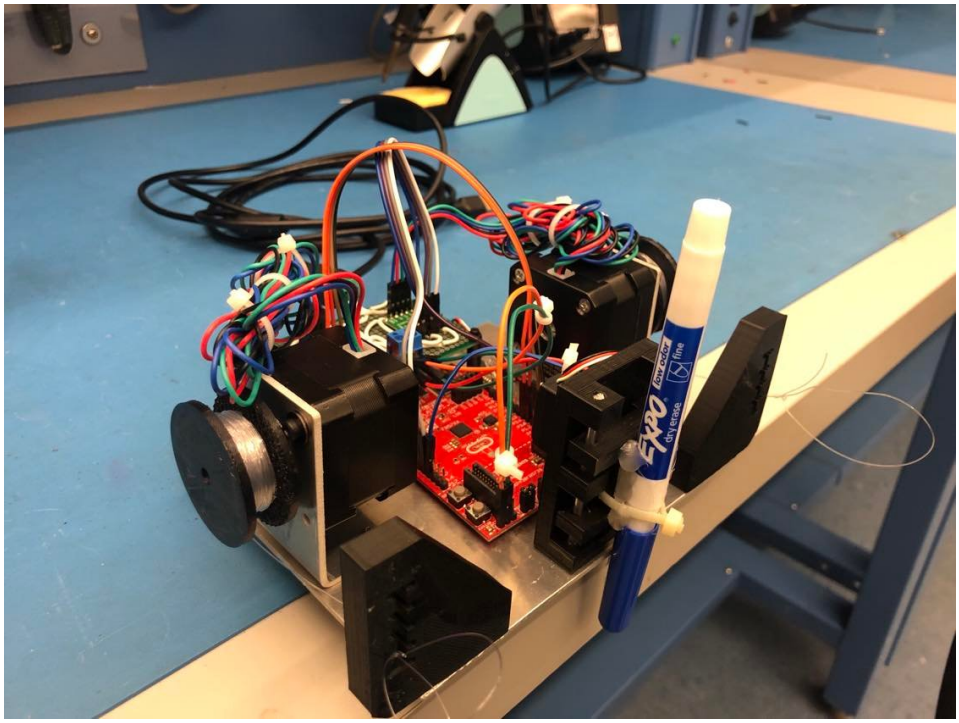
Mech 423: Wall Drawing Robot

Project Report



Gurmehak Pannu (gm.pannu41@gmail.com)
Anoush Sepehri (anoushsepehri@gmail.com)

December 24, 2021

Abstract

The goal of this project was to design and build a 2D wall drawing robot. This type of device would be valuable for a number of applications including murals, advertisements, and classroom applications for drawing and writing notes. The wall drawing robot will automate repetitive tasks and free up time and resources for others to work on more important and challenging problems.

The robot was designed to be suspended from two strings at both corners of a board to be drawn on. The strings are retracted or extended in a precise combination to maneuver the robot around to draw. The development of the robot can be split into four categories, the mechanical design, electrical design, firmware development, and front-end/User-interface development. The mechanical design consists of two motor mounts to house stepper motors for controlling the string lengths, string guides to ensure the strings are aligned at all times, a servo/marker mechanism to extend and retract the marker for drawing, as well as the entire enclosure to house the electrical and mechanical components. The electrical aspect of the robot consists of a protoboard with two motor drivers to supply 12 V to the stepper motors based on signals from the MSP430 microcontroller. The firmware includes all the software that is deployed on an MSP430 board for parsing and determining correct string lengths and actuation patterns to successfully maneuver the robot around the board. The front-end design involves the User interface that was developed in C# to enter different drawings or patterns to draw. A unique G-Code packet structure was also developed that could store all the necessary information/commands that the firmware on the MSP430 could parse and understand.

Table of Contents

1. Objectives

The goal of this project was to design and build a "Wall Drawing Robot" which can create 2D designs on a vertical surface such as a wall or whiteboard. During the planning stage, the target device was a mobile robot supported by 2 strings attached to fixed supports on the wall. This conceptualized setup is shown in Figure 1 below. After going through the design and build process, we were able to produce a robotic system which satisfied the initial concept to a significant level. Our final device shown on the title page was able to perform the desired task of creating 2-D drawings using the string-support mechanism shown in Figure 1. 2 simple designs drawn using the robot are shown in Figures 11 and 12 in the System Evaluation section below. Despite the overall functional success of the device, there were some features we were not able to accomplish. These included the ability to draw on a fully vertical surface (all tests conducted were on a 45 degree plane), draw using multiple colors, and load custom images to be drawn. For future iterations and improvements of our device, we would focus on these aspects.

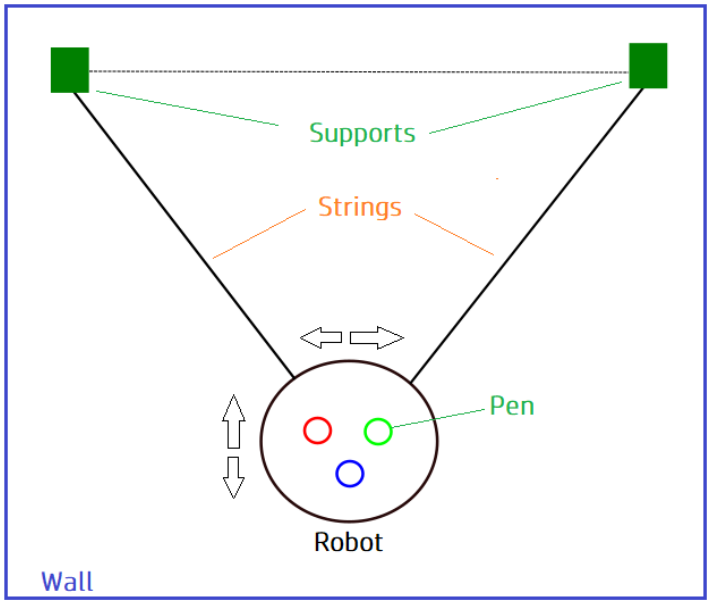


Figure 1. Conceptual sketch of the robot drawing system on a wall

2. Rationale

The target device in this project would be a useful tool to create custom drawings or text on vertical surfaces. Some potential use-cases include creating murals on public surfaces and creating art/informational displays in home or business spaces. For example, some coffee shops and restaurants display their menus with handwritten text and drawings on a large wall. An example of this can be seen in Figure 2. To achieve this, they need to hire an artist or spend a significant amount of their time and effort to create the display. Our device would be able to do

this autonomously and would not require any artistic expertise from the user. Furthermore, it will ensure high quality drawings each time due to its high repeatability, thus eliminating human error. Our device would also free up the time of artists or employees who would have had to do this before.

Figure 2. Example of a large menu that is handwritten in a coffee shop

3. Functional Requirements Summary

Our functional requirements can be seen in Table 1. It should be noted that these requirements are slightly different than those listed in our project proposal as we think these are more representative of what we focused on during the design process. One of the requirements (FR#6) from the proposal is omitted - This functional requirement described an optional feature of converting a .PNG image into a drawpath and sending it to the MSP board. Due to time limitations, we were not able to satisfy this requirement in the final device. This is further discussed in the conclusion of this report.

The functional requirements interact as follows. Series of commands from the User interface are sent to the MSP430 via UART. Once stored in the MSP430 code, they are analyzed to determine suitable commands to the servo motor (for the marker) as well as the stepper motors (for controlling the string lengths). The motor commands for the stepper motors are sent to the drives to supply the necessary voltages using standard digital IO. The command to the servo is sent directly from a pin with a PWM signal with varying duty cycle. The actuation of the motors maneuver the robot and entire physical system around on the drawing board to create different shapes and drawings.

Table 1. Functional Requirements for our wall drawing robot

Functional Requirements	% Effort	Responsible Person
FR#1: C# UI and code to generate robot drawpath and transmit to MSP board	20	Anoush/Gurmehak
FR#2: MSP430 code to read and parse drawpath from FR#1 and send signals to motor drivers for actuation	30	Anoush
FR#3: Motor drivers electrical design and wiring layout	10	Anoush
FR#4: Design and integration of Pen Extension Mechanism	10	Gurmehak
FR#5: Mechanical Design of Robot/ Pulley Systems	30	Gurmehak

4. Functional Requirement Analysis

Functional Requirement #1
C# UI and code to generate robot drawpath and transmit to MSP board

---

Approach and Design

The objective of this FR was to develop a User Interface in C# which the user of the robot could use to interact and send commands to the robot.

To address this FR, a GUI was developed using the Windows Forms template in C# - a screenshot of the interface is shown in Figure 3 below.

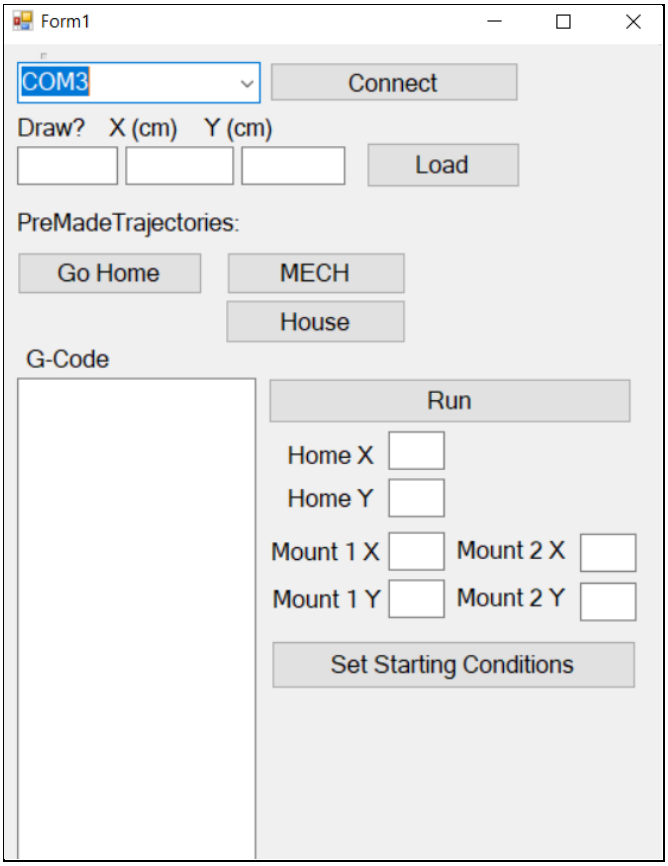


Figure 3 - C# User Interface for communicating with the robot

Using this interface, the user can first connect to the MSP430 board via UART. Once connected, the user has a few different functionalities to choose from. The robot can be jogged to a particular x,y coordinate by filling in the textboxes in the second row of the form. The “Draw?” textbox specifies whether the pen should be extended or retracted when the robot is moving to the specified coordinate. Once the text boxes are filled in by the user, the “Load” button is pressed and the resulting serial command is shown in the “G-Code” box on the bottom left of the form. The user can load multiple commands prior to sending these to the robot. Once the user is satisfied with the loaded commands, they can press the “Run” button to send these to the robot. Once the “Run” button is pressed, the robot will parse and execute these commands.

The user interface also features some pre-made trajectories which make the robot draw a particular design on the drawing surface. For example, pressing the “House” button will load a trajectory which draws a simple house and the “Mech” button will load a trajectory which writes the letters “MECH” onto the drawing surface. The “Go Home” is used to bring the robot to its home position which is set at power-on. The home position as well as the position of the fixed mounts can be changed as required based on the type of setup the robot is being used on. This feature is included underneath the “Run” button on the form.

Inputs and Outputs

The inputs and outputs of this functional requirement can be seen in Table 2.

Table 2. Inputs and Outputs to the electronic system

Inputs	Commands inputted by user
Outputs	Serial Packets to MSP board

The packet structure used in the serial output is described in FR #2 below. It is referred to as the "G-Code" packet.

Parameters

The main parameters for this FR are listed below:

1. Home position coordinates
2. Mount 1 and Mount 2 position coordinates
3. G-Code Commands

All the parameters above are dependent upon the setup in which the robot is being used and are all supplied by the user prior to running the robot.

Testing and Results

The testing of this FR was done in conjunction with FR #2 which encompasses the MSP code developed to run the robot. This included preliminary testing to verify communications between the MSP board and PC using the CCS debugger. This was followed by more comprehensive testing which involved mounting the robot on a whiteboard and then sending simple commands through the user interface. Finally, in the last phase of testing, complete trajectories for simple shapes were loaded onto the robot. This testing is discussed in further detail in FR#2 below.

Functional Requirement #2
MSP430 code to read and parse drawpath from FR#1 and send signals to motor drivers

Approach and Design

The objective of this function was to read all the commands from the UI and convert them into the appropriate signals to the stepper motors to move the robot and draw different objects.

The MSP430 code was designed as follows. A set of "G-Code" packets were sent from the UI to the MSP430 where they were stored in a buffer. G-Code commands were stored until a specified start byte was sent to the MSP430. When the start byte was received, each G-Code command packet is parsed and converted from coordinates to commands for the stepper motors and marker. Once the steppers completed their actuation, the next G-Code command was parsed. This was continued until the buffer was empty, meaning all G-code commands were executed. A more detailed description of determining stepper commands from a G-Code packet can be seen below.

The first step in determining the stepper signals is to calculate how much to extend or retract each of the pulleys. This was done by analyzing the kinematics of the robot. A diagram of the kinematics can be seen in Figure 4. The XYZ coordinates of the marker were converted to XYZ coordinates of the pulley (X_{pulley} Y_{pulley} Z_{pulley}) by using the calculated offsets presented in the parameters section. This was also done with the desired coordinates that were extracted from the G-Code packets. With the real and desired coordinates, the new string length was calculated as follows. $X_m Y_m Z_m$ correspond to the mount locations on the wall. These are defined in the MSP430 Code and are further discussed in the parameters section.

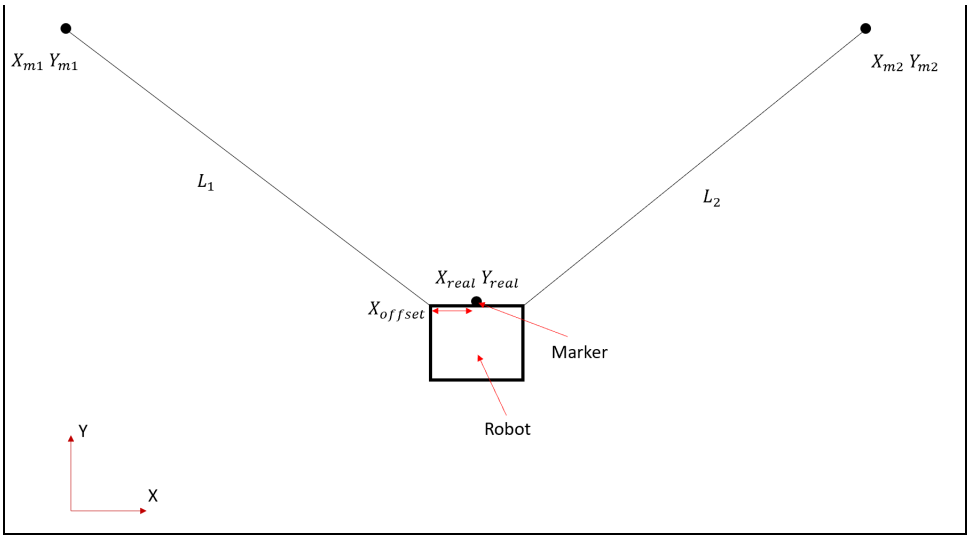


Figure 4. Basic kinematic setup of robot to calculate string lengths. Some key parameters are included in the figure and are further discussed in the parameters section.

$$L_{string} = \sqrt{(X_m - X_{pulley})^2 + (Y_m - Y_{pulley})^2 + (Z_m - Z_{pulley})^2}$$

$$L_{desired} = \sqrt{(X_m - X_{desired})^2 + (Y_{m1} - Y_{desired})^2 + (Z_m - Z_{desired})^2}$$

$$\delta L = L_{string} - L_{desired}$$

A negative change in length corresponds to a retraction and is noted in the MSP430 code. A positive change in length corresponds to an extension. The change in length was calculated for both strings.

With the change of lengths, the time needed to operate the steppers is calculated. This is done by dividing the change in length by the target speed defined in the parameters. Furthermore, the number of steps required to turn the steppers was calculated using the stepper parameters (half-stepping, 0.9 degree increments), as well as the pulley geometry.

The create commands to send to the stepper, a "scaling factor" was calculated to be used for determining when to step the motors. This was done by dividing the total time required to actuate the motors by the number of steps. To ensure proper synchronization between motors, the motor that required more time to actuate was used when determining the scaling factors for both pulleys.

$L_{factor} = \delta L_{time} * 1000/\delta L_{steps}$ (provides a scale factor with units ms/step)

Once the scaling factors are determined for each string, a clock is started that interrupts every millisecond. For each interrupt, a counter is incremented. Once the counter reaches the scaling factor the stepper is moved one step (in either the extension of retraction direction). The counter is then reset to 0. This is continued until both steppers have moved the correct number of steps. In essence, this ensures that for every L_{factor} milliseconds, the stepper moves one half step. Once both motors have moved the correct number of steps, the clock is paused and the next G-Code packet is parsed. The pseudocode walking through the entire parsing and analysis process can be seen below.

```
If start byte received:
    Extract G-Code packet from buffer
    Calculated δL for both strings
    Calculate the time and number of steps to retract of extend by δL for both strings
    If (string 2 takes longer to move):
        //use string 2 time to calculate scale factors for both strings
        Calculate L_factor
    Elseif (string 1 takes longer to move):
        //use string 1 time to calculate scale factors for both strings
        Calculate L_factor
    Turn on clock with 1 ms interrupts
    Run each stepper every L_factor milliseconds for each string until completed
    If (both strings completed actuation):
        Turn off clock
        Extract next G-Code packet (if buffer is not empty)
```

Figure 5. Pseudocode for determining stepper commands

Inputs and Outputs

Each G-Code packet is delivered as follows:

Start Byte	**X coordinate**	**Y coordinate**	**Activate Pen**
255	8 Bit Unsigned Int	8 Bit Unsigned Int	Bool

The start byte is to keep track of the start of a G-Code packet. The next two bytes are the X and Y coordinates to maneuver to. The final byte is a boolean to determine if the pen should be

activated (for drawing) or if the pen should be disabled (for traveling). Do note that because this is a 2-dimensional drawing robot, the Z coordinate is always set to zero.

The output from the MSP430 code from calculating the scaling factors is sent to the stepper motors. A state machine that keeps track of the pattern of coils to active/deactivate for proper half stepping was implemented in the software. Digital signals of 3.3V or 0V were sent to the motor drives to produce 12V signals that are necessary for the stepper motors to operate. More description of the electrical components can be found in Functional requirements #3.

Parameters

Several parameters in the software were defined and tuned to successfully convert the G-Code into stepper commands. They are listed in Table 3. The velocity of the robot was used to limit the speed that the motors could operate at. This was an important parameter which had clear trade offs in performance. A faster robot meant it could draw faster, however this would lead to larger inertial forces and torque requirements, thus leading to higher chances of skipping steps or motors stalling. Furthermore, faster speeds led to higher operating temperatures of the stepper motors, which was undesirable due to the fact that the pulley was fabricated using PLA and was at risk of melting. Through trial and error, a velocity of 4 cm/s was determined to be the most optimal to maximize speed while staying within safe torque limits and operating temperatures. The XYZ offsets were used to determine the location of the pulleys relative to the marker. They were measured by hand after manufacturing the robot. Because of symmetry, the X offset for one motor was 8 cm while the other motor was -8 cm. The Z offset is 1 cm because of the size of the roller bearings that were installed on the robot to ensure smooth motion along the board. The pulley diameter and stepper increment size were used to determine how many steps are necessary for the stepper to extend or retract the string a fixed length, These values were determined while designing the pulleys and by going through the datasheet of the motor (to identify increment size of the stepper). The XYZ mount coordinates were determined while installing the robot onto a wall. These coordinates are for calculating the string lengths and are the position of the C-clamp mounts for hanging the robot. The Z coordinate of the mounts are always set to zero.

Table 3. Key parameters in MSP430 Code

Parameter	**Selected Value**
Velocity	4 cm/s
X offset	8 cm
Y offset	0 cm
Z offset	1 cm

Pulley Diameter	1.5 cm
Stepper increment size	0.9 deg
X mount 1 (X_{m1})	0 cm
Y mount 1 (Y_{m1})	119 cm
X mount 2 (X_{m2})	86 cm
Y mount 2 (Y_{m2})	119 cm

Testing and Results

The first tests conducted were to ensure that the UART was working as expected. Data packs were sent from the UI to the MSP430 and without any analysis conducted. The extracted values were validated in the debugger to ensure that they were being properly stored and retrieved from the buffer.

The next tests were to ensure that the correct commands were being sent to the stepper. To test this, single commands were sent to the robot. The robot was attached to a whiteboard and commands to move the robot vertically and horizontally were sent to the MSP430 microcontroller. The resulting lines were measured and compared with the desired commands. In all cases, the lines were within 1 cm and straight enough to ensure satisfactory performance.

The final tests were to draw basic shapes. Sequences of G-Code commands to draw squares, rectangles, and triangles were used and sent to the robot. The side lengths were measured and validated against the input commands. Some of the tests can be seen in Figure 6 and 7. For all tests conducted, the measurements were within 1 cm and were thus satisfactory to move forward and create more complex drawing. More complicated drawings are presented in the system evaluation section.

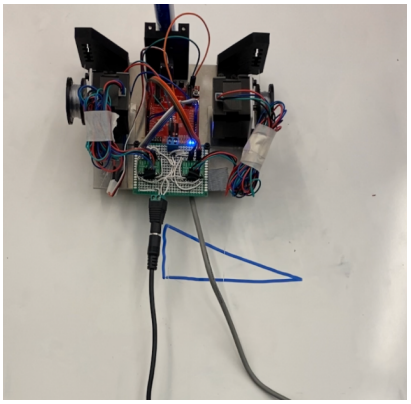


Figure 6. Drawing a basic right triangle

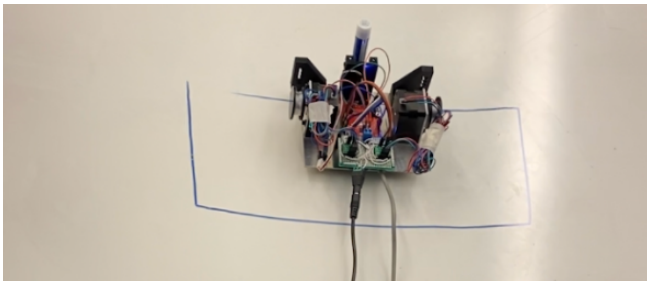


Figure 7. Drawing a basic rectangle.

Functional Requirement #3
Motor Drivers Electrical Design and Wiring Layout

Approach and Design

The objective of this FR was to design a protoboard and appropriately connect the key electronic components used in the robot. This includes 2 NEMA 17 stepper motors to control the position of the robot and a SM-S2309S servo motor to control the pen mechanism.

To address this FR, the first step was to select appropriate motors to power the robot. The following estimation procedure was followed to select the appropriate motor and driver:

Robot Mass = 2 kg max
Pulley Diameter = 1.5 cm

Calculate peak torque required when one of the strings almost vertical and is carrying most of the robot's weight:

$$T_p = Fr = mg\frac{d}{2}$$

From the above values, this gives us a required peak torque:

$$T_p = 0.196 \text{ Nm}$$

After looking at potential options, it was found that a NEMA 17 motor would be appropriate for handling the above torque requirements. Based on the specs of the chosen motor, the TB6612FNG motor driver chip was chosen to drive the motor. 1 Chip is used per motor.

Once the key components were selected, they were connected as per the schematic in Figure 8 below. The motor drivers were soldered onto a proto-board and this board was installed on the robot chassis using 3D printed standoffs. All connections to the stepper and servo motors were made using jumper wires, similar to those provided in the Lab 3 kit.

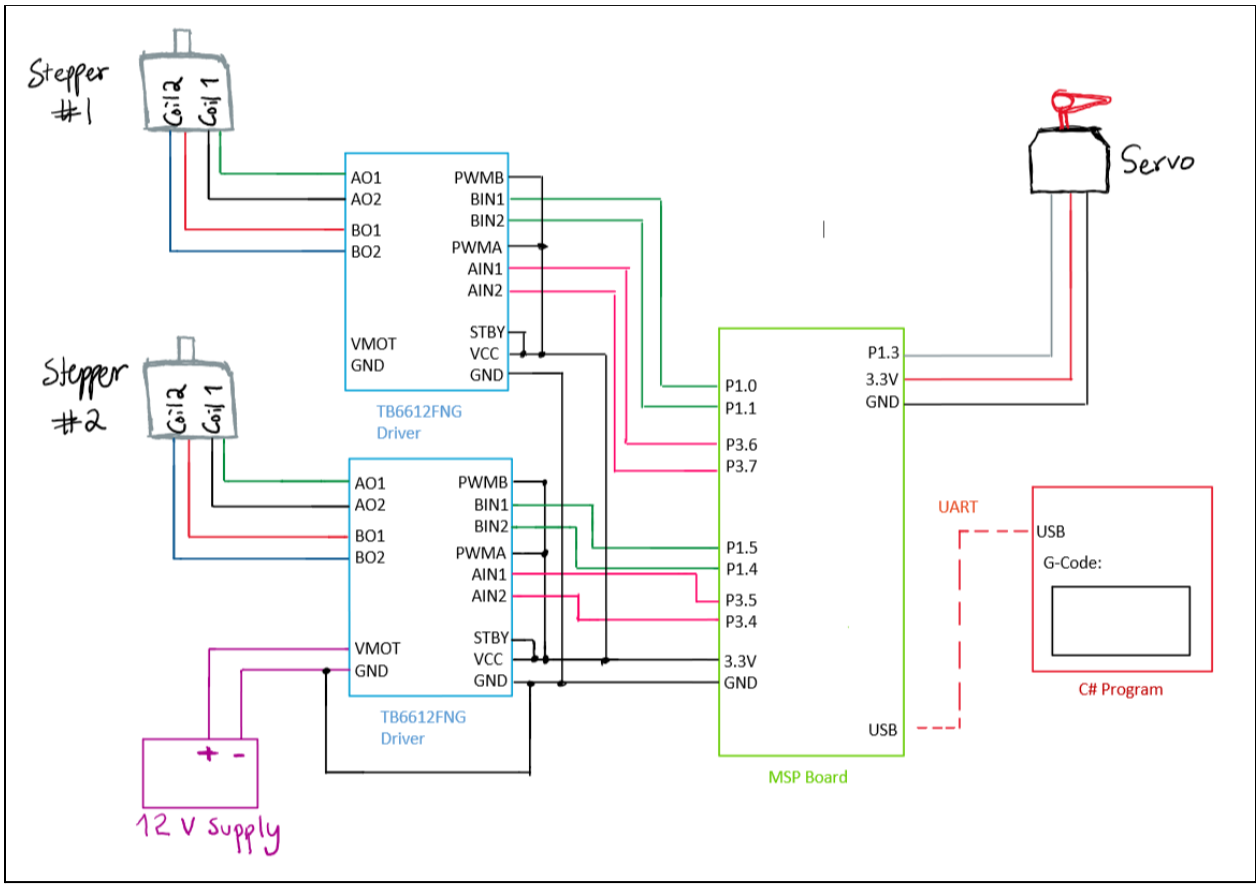


Figure 8 - Wiring Layout of all electrical components of the robot

Inputs and Outputs

The inputs and outputs of this functional requirement can be seen in Table 4.

Table 4. Inputs and Outputs to the electronic system

Inputs	Digital IO commands from MSP at 3.3V or 0V
Outputs	Analog voltages to Motors at 12V (Stepper) and 3.3V (Servo)

Parameters

The only parameter which needed to be "tuned" for this module was the PWM signal range supplied to the servo motor to achieve the desired range of travel. This tuning was done in parallel with the testing of the pen extension mechanism by trying out different PWM values until a suitable range was found.

Testing and Results

To test the wiring of the system, tests were done to ensure that the stepper motors would run as expected when commanded by the MSP and C# programs. First, all pins were confirmed to be properly connected using a standard multimeter. Then, a simple MSP and C# program were written which commanded the stepper motors to rotate a certain number of steps, as specified by the user in a text box in the C# program. By visually observing the range of travel (eg. full turn, half turn, quarter turn), we were able to verify if the stepper was rotating the correct number of steps. Once this core functionality was verified, we could move on to test more complex paths supplied to the motor which is further described in FR #2.

Functional Requirement #4
Design and integration of Pen Extension Mechanism

Approach and Design

The objective of this FR was to design and build a mechanism which could be used to extend and retract the pen or marker used in the robot for drawing.

To address this FR, a spring loaded mechanism was designed which translated the marker in the vertical direction and applied the marker onto the wall with a constant force. The mechanism is

actuated using a [model name] servo motor which is controlled by the MSP board. The final subsystem is shown in Figure 9 below.

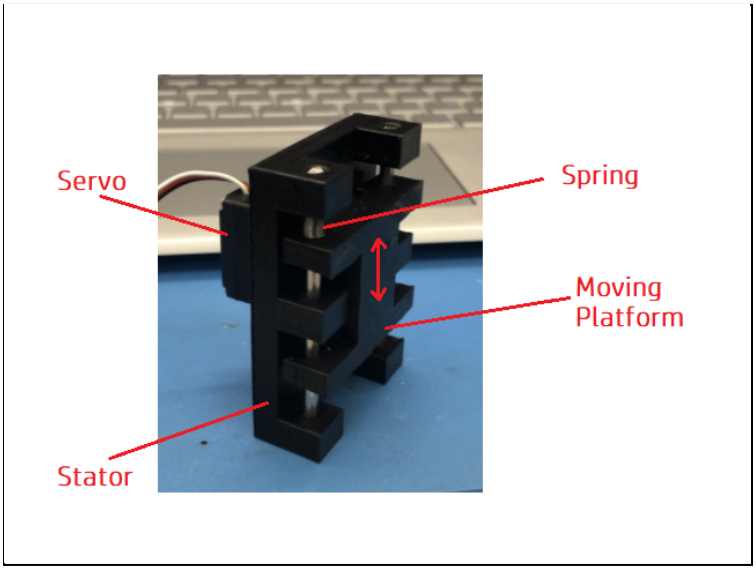


Figure 9 - Pen Extension/Retraction Mechanism

The marker is omitted in the above picture, but in the final device, the marker was attached to the "Moving Platform" using hot glue. There is a servo arm hidden behind the moving platform which makes contact with the platform to move it up and down in the direction shown above. There are only two positions the platform can be at - "UP' or "DOWN".

Inputs and Outputs

The inputs and outputs of this functional requirement can be seen in Table 5.

Table 5. Inputs and Outputs to the electronic system

Inputs	PWM command from MSP (0% - 50%)
Outputs	Translation of Servo/Pen to appropriate position (UP or DOWN)

Parameters

The key parameters for this FR are listed below:

1. Spring stiffness
2. Length of travel

In the first prototype of the pen extension mechanism, 2 parallel springs were being used. Each spring was taken out of a regular ball-point pen which corresponds to a stiffness of roughly 250N/m. Therefore, the combined stiffness of the system was about 500 N/m. After doing some

preliminary functional tests, it was found that this stiffness was too high and resulted in a rather large force applied by the pen onto the drawing surface. To remedy this, one of the springs was removed and the resulting stiffness seemed to be adequate for the system based on qualitative observations. The mechanism was designed with a travel length of about 1cm between its “UP” and “DOWN” positions. After preliminary tests, this travel distance was found to be adequate.

Testing and Results

To verify functionality of the pen extension mechanism, preliminary testing was first done by 3D printing and assembling all the pieces of the system and then moving the mechanism by hand to see if it performed as expected. Next, the system was connected to an MSP board and a simple firmware program was written for the MSP which activated the servo motor based on pressing of the push buttons. Finally, the subsystem was integrated into the overall device to see how well it performed when drawing on a surface. Results from all the tests were mainly observational and are summarized in Table 6 below.

Table 6 - Tests and Observations

Test	Observations
Test #1: All components assembled (No electronic actuation)	● Mechanism worked as expected; length of travel was adequate, spring stiffness seemed a bit high
Test #2: With Servo Actuation using MSP	● Mechanism was moving vertically as expected - removed one of the springs for lower stiffness
Tes #3: With full system (Pen Attached)	● Pen was first attached using a zip-tie. With this setup, the pen kept slipping upwards whenever contact was made with the drawing surface. Hot glue was then used to fix the pen onto the moving platform which fixed the issue.

Functional Requirement #5
Mechanical Design of Robot and Pulley Systems

Approach and Design

The objective of this FR was to design and build the mechanical subsystem of the robot. This subsystem includes the following aspects.

1. Winch Mechanism to retract/extend string
2. Enclosure/Chassis to house all internal components

Note that the marker extension and retraction mechanism in covered in FR #4.

To address this FR, a 3D model of the desired system was first developed based on structural requirements of the system. This included the ability to mount all actuators and electronics on a compact chassis frame which would be supported by 2 strings coiled onto pulleys attached to the actuators (NEMA 17 stepper motors). The final 3D model is shown in Figure 10 below.

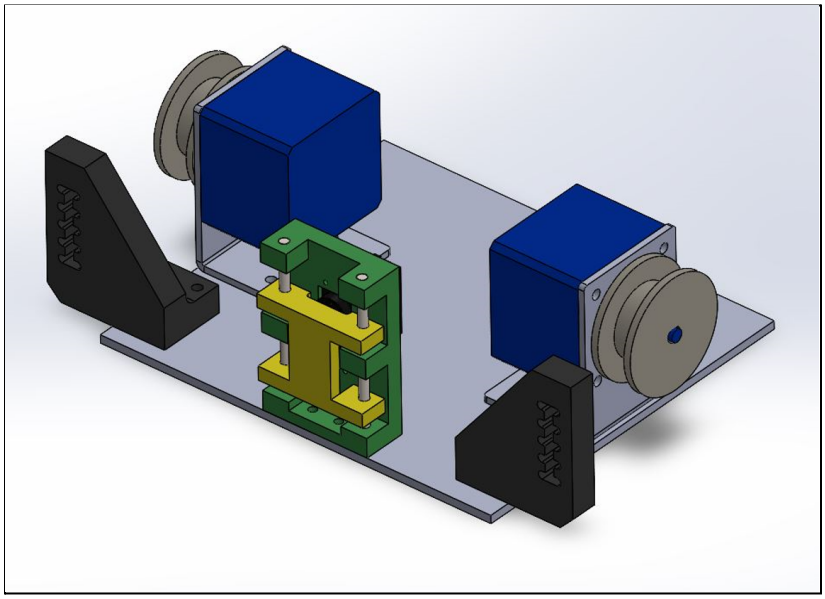


Figure 10 - 3D Model of Mechanical System without electrical components

Inputs and Outputs

The inputs and outputs of this functional requirement can be seen in Table 7.

Table 7. Inputs and Outputs to the electronic system

Inputs	Voltage Commands to Steppers from MSP Board
Outputs	Rotation of pulleys to achieve string extension/retraction

Parameters

The main parameters which needed to be adjusted for the mechanical subsystem is the spacing of the string guides. In the initial prototype, we found that the strings were getting caught on the pulleys when actuated. As a potential solution, the spacing between the two string guides was

increased such that the “pivot” point of the string was roughly on top of the pulley. We saw substantial improvement with this change and kept the spacing at this level in the final device.

Testing and Results

Testing for the mechanical subsystem was mostly integrated into the overall system evaluation testing which is further discussed below. Nevertheless, there were some small mini-tests which were conducted to verify fit and compatibility of different components. For example, to fit the pulleys onto the stepper motors, a specific hole profile was required which had a flat section on a circular shaft. Additionally, since there are inherent errors when 3D printing small features such as holes, we first printed out some “sample” profiles to test with the motor shaft. By qualitatively comparing the fit of the different samples printed, we selected the sample which provided the most snug fit while allowing for assembly by hand. The design of this sample was then used in the final pulley design.

5. System Evaluation

The final tests conducted to validate the robots performance were to draw full drawings. This would validate all aspects of the system (User interface, data transmission, G-Code parsing, marker control etc). Two drawings were selected to test the robot’s performance. These were a house and the word “MECH”. A sequence of G-Code commands (approximately 20 to 30 commands) were programmed in the User Interface to send to the robot. The commands were delivered and the robot subsequently created the drawing. The final drawings can be seen in Figure 11 and 12. The results were satisfactory as each aspect of the system was working as expected. The robot was able to generate an accurate representation of the drawings using the G-Code commands that we wrote and sent to the MSP430 using the User Interface.

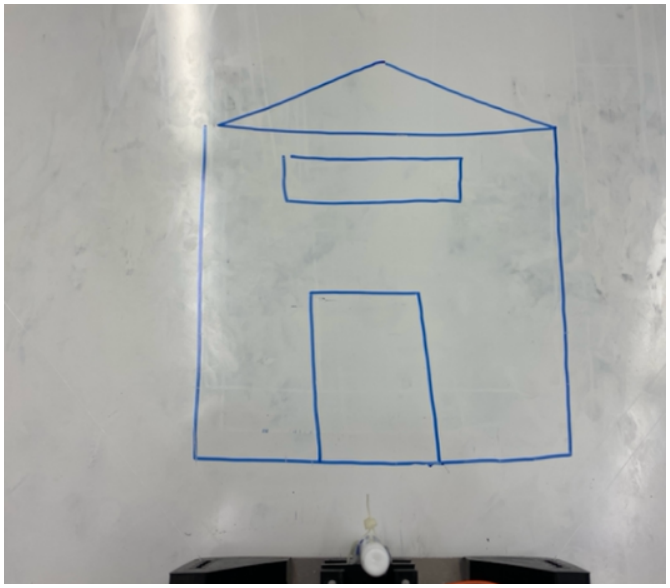


Figure 11. Final drawing of the house

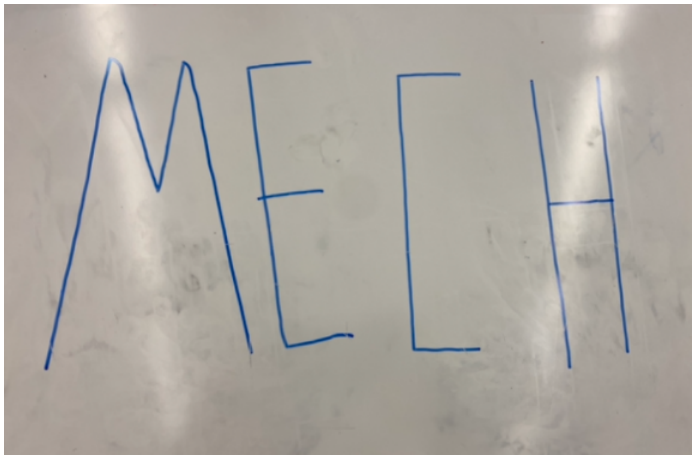


Figure 12. Final drawing of “MECH”

6. Reflections

What worked and what didn’t work? Why? What would you do differently if you could do it again?

In this project, the aspects which turned out well include the overall mechanism used to move the robot as well as the path planning algorithm used to control it. Given that we were able to achieve satisfactory accuracy on the shapes drawn, the method of using strings to move the robot was validated. However, a key drawback in the project was that we were not able to draw on a fully vertical surface and had to do our testing on a whiteboard angled at 45 degrees from the vertical. This mainly resulted from the robot not having enough horizontal force against the

whiteboard which would keep the robot in contact with it. If we could do the project again, we would experiment with increasing the offset between the pulleys and the drawing surface (whiteboard). A higher offset would result in a larger "z-component" of the force from each string. This "z-component" acts in the direction perpendicular to the drawing surface and hence, a larger value of this force component would result in more force pushing the robot towards the white board.

An additional improvement we would like to implement include using a DC Motor servo control loop to coil/uncoil the string as opposed to the stepper motors. We found that the inherent jerking motion that comes from starting and stopping motion of the stepper motors results in vibrations of the robot and hence decreases accuracy of the drawings. A DC Motor control loop would likely provide more smooth motion and would be worth experimenting with.

Identify 3 things you learned in MECH 423 that you consider the most useful and why.

Throughout MECH 423, we believe learning the following things were most useful:

1. **Firmware development for Microcontrollers:** As microcontrollers are used to control a majority of mechatronic systems, having a good knowledge base of how they operate and how to program them is a vital skill for any mechatronics engineer. After MECH 423, we now have a strong understanding of key microcontroller components such as Clocks, Timers, ADC, and Communications. With this knowledge as well as the experience of programming microcontrollers, we now have the capability to implement microcontroller software for other mechatronic systems outside MECH 423.

2. **C# GUI Development:** Another key item learned in MECH was the design of GUIs using the C# Windows Forms template. After developing a wide variety of GUIs in labs as well as the project, we have a good command of the features available in this framework and the areas in which it could be applied. Knowing how to create simple GUI programs using C# and interfacing them with hardware is a key skill because it allows the development of user interfaces relatively quickly compared to other GUI development frameworks such as Qt in Python. In fact, we have already applied this skill in another course called CPEN 333 where we need to develop an automated software system for a warehouse. We are sure this skill would come in handy in future projects as well.

3. **Driving and Controlling DC/Stepper motors:** Finally, on the hardware side, we learned to drive both a Stepper and DC motor in MECH 423. Additionally, we also implemented closed-loop control on a DC motor gantry system to achieve servo-style position control. In industry, many mechatronic systems such as CNC machines and motion control

systems utilize similar closed-loop control systems using DC or stepper motors. Therefore, having this experience in how to set up a simple control system on a DC motor or controlling a Stepper motor to achieve a similar purpose would be a key asset in our mechatronics knowledge.

What are some limits of your knowledge and expertise as a mechatronic engineer? Identify 3 things you would like to learn going forward. What is your strategy to acquire knowledge in these areas?

As mechatronics engineers, we have a broad skill set ranging from Mechanical Design to Electronic design to firmware and software development. However, there definitely are some limits to our expertise and gaps in some areas of our knowledge. The following are just 3 out of many things we would like to enhance our knowledge in.

1. Image Processing and Computer Vision
2. Deeper Programming Knowledge: Web Development, Further knowledge in Microcontroller programming (I2C communication, content not covered in MECH 423 etc.)
3. Machine Learning

The items identified above are highly relevant to the technologies used in industry today. Therefore, having a deeper knowledge base in each of these things would enhance our ability to solve problems as Mechatronics Engineers. To acquire more knowledge in these areas, the best way is likely to get involved in projects which require the implementation of 1 or more of the identified topics. Furthermore, there are a variety of online courses through websites such as coursera and udemy that offer training in these areas. Having a problem to solve or a project to build is usually the biggest motivator in learning new skills. Therefore, it would make sense for us to seek projects which involve the above items, whether it be at a workplace or on personal side projects as well as take an online course covering some of the content.