Multi-Segment Parallel Continuum Manipulator

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for

the Degree in Bachelor of Science

in

Robotics Engineering

By:

Sarah Chamberlain

Steven Kordell

Cody Wall

Date: April 30, 2015

Approved:

Kenneth Stafford, Advisor

Worcester Polytechnic Institute

Major Qualifying Project

Multi-Segment Parallel Continuum Manipulator

Authors:
Sarah Chamberlain
Steven Kordell
Cody Wall-Epstein

Date: April 30, 2015
Adviser: Prof. Stafford
Project: MQP KZS COMA

Abstract

Continuum manipulators are a type of robot arm that resemble biological tentacles and trunks. They have a flexible and compliant structure, which may allow them to out-perform rigid-link designs in cluttered workspaces or in environments that contain people. While most continuum manipulators are required to have constant curvature along the length of each segment, a new design known as a parallel continuum manipulator removes this restriction and inherits some properties of parallel rigid-link robots such as greater stability, precision, strength, and maneuverability. Until now, only single segment forms of these manipulators have been created. This project expands this manipulator design concept by creating the first multi-segment parallel continuum manipulator.

Contents

List of Figures

List of Tables

1 Acknowledgements

Throughout this project, many individuals have provided guidance, supplies, and other Aid.

First, we would like to thank Worcester Polytechnic Institute's Professor Kenneth Stafford for advising this project and providing us with guidance and feedback on a weekly basis. We would also like to thank Professor William Michalson, who provided us with lab space in Atwater Kent Laboratories for storing and working on the project. Finally, we would also like to thank WPI's Washburn shops where all of our manufacturing took place.

Second, we would like to thank Derrill Hartley from the Pololu team who generously discounted the stepper motors and drivers used on the manipulator and for his assistance in the selection process.

Finally, we would like to thank Anthony Aragon and the Advanced Circuits Team who generously fabricated and donated the electronic control board designed for this project.

2 Executive Summary

Research and development in soft robot manipulators have led to advances in continuum manipulators. More recently, parallel continuum manipulators have been developed, which attempt to combine the benefits of continuum manipulators with parallel rigid-link manipulators. This project expands this manipulator design concept by creating the first multi-segment parallel continuum manipulator.

Initial background research was used to set the project's scope and goals for workspace dimensions, speed, compliance, precision, loading, and software control. A plan was established for working towards achieving these goals over the course of an academic year. The plan was carried through to completion using a number of engineering skills to design, build, and test the manipulator and all three sub-domains of robotics (mechanics, electronics, and software) were involved in the project's design and construction.

The resulting two-segment parallel continuum manipulator was tested against the initial project goals and met all goals except for those related to the manipulator's maximum speed.

3 Introduction

3.1 Motivation

The first robot manipulators were introduced by General Motors in 1961 [21] for mass manufacturing. In the nearly six decades since, robots have been transforming industry by vastly increasing the efficiency with which goods can be produced. The robots of today can easily perform a great variety of tasks with speed, precision, and repeatability, but despite these benefits and many years of research, robots have still failed to become ubiquitous within human homes. This failure can be attributed to a large number of causes such as high cost, safety, lack of software intelligence, and the motion limitations of traditional manipulators.

The robots of today commonly consist of a large number of rigid links connected by rotational or prismatic joints. These rigid links can place motion constraints on these manipulators, which can make it difficult for the arm to reach certain positions or closely follow complex trajectories. Even when the manipulator is capable of reaching every desired point within the defined work area, joint limits and singularities may prevent it from doing so along a continuous path, making strict trajectory-following difficult. Often positioning a workpiece in an environment such that a specific trajectory is valid requires a trial and error methodology where one tries moving the workpiece into several possible poses until one is found which allows the robot to generate a desirable path. Obviously, such a methodology is not feasible outside of a choreographed industrial environment, making it difficult for average people to use.

Making matters worse, when working in complex, obstacle-ridden environments, objects can easily obstruct otherwise valid trajectories. Even when a valid trajectory can be found, it may require placing the robot in awkward positions where it is unable to take full advantage of the available torque. Additionally, today's manipulators are often heavy, bulky, and dangerous, and are usually separated from humans by protective fences to limit human contact as much as possible for fear of accidents or fatalities. Such robots are clearly not suitable for working in home environments or around large groups of people.

3.2 Objective

Ultimately, the lack of versatility in traditional manipulators may be a fundamental flaw keeping robot manipulators and the advantages they offer from performing tasks outside of industry and entering human homes. To approach this problem, this project has constructed a manipulator with significantly fewer motion constraints which could operate safely around humans. Having fewer motion constraints means less intelligence will be needed to find valid and safe trajectories for the manipulator to move through, opening up opportunities for robotic manipulators to move into these environments.

4 Background

4.1 Definitions

There are various types of manipulators, each defined by certain characteristics and parameters. Configuration space, or "C-Space", of a manipulator is the space of all possible joint states. The number of configuration space variables for a particular robot is equal to the number of degrees of freedom (DOF) the robot has. The number of degrees of freedom needed to perform a particular task defines the task space. In two dimensions, this value would be three, two degrees of lateral motion and one degree of rotational motion. In the three dimensional world, this number is usually six, where three degrees of lateral motion and three degrees of rotational motion are the minimum required for most handling tasks. Finally, the workspace is the bounded physical area that can be reached by a manipulator's end-effector.

4.2 Rigid-link Manipulators

Traditionally, most robot manipulators have been rigid-link manipulators used in industry for mass-producing goods. According to the International Federation of Robotics (IFR), in 2012 between 1,235,000 and 1,500,000 industrial robots were in operation with a worldwide market estimated to be $26 billion [17]. IFR classifies industrial robots into one of 5 categories: articulated robots, cylindrical robots, linear robots, parallel robots, and SCARA robots [18]. All of these robots rely upon the traditional rigid link structure used since the inception of industrial manipulators.

According to the Occupation Safety and Health Administration (OSHA), since 1984 there have been thirty-three industrial accidents involving robots, with 24 of them resulting in fatality [20]. This number is small considering the large number of industrial robots in operation, but this is not because the robots used are inherently safe. Rather, it is because these robots are kept away from people in closed and controlled environments. The majority of industrial robot accidents occur when there is a breach of these safety measures and humans enter the robot's proximity. Today's rigid-link robots, though great for performing tasks repeatedly with accuracy and precision, are just not meant to be in an environment where they might collide with humans, making them generally infeasible for use in non-industrial environments.

4.2.1 Redundant Manipulators

It is common to see six DOF rigid-link manipulators since the task space in which most robot manipulators work is a six DOF space and it is economical to match the number of DOFs needed in the task space with the number of DOFs in the configuration space. There are, however, a number of circumstances when having additional degrees of freedom can be extremely useful. Manipulators that have more degrees of freedom than required for a particular task are termed redundant manipulators. Such manipulators benefit by being able to better maneuver around obstacles, avoid joint limits, and escape singularities. This can be beneficial in certain tasks like trajectory-following. A non-redundant manipulator following a trajectory may at some point along the path encounter a joint limit or obstacle that would require the end-effector deviate from this path and approach it from another direction. In certain tasks where this extra motion could be considered unacceptable, a redundant manipulator is particularly handy. While a non-redundant manipulator might only have a single solution to the inverse kinematic problem, a redundant manipulator often has infinite solutions and is often able to move from one to the next without having the end-effector deviate from the path.

In many cases, to achieve more agility with a manipulator, designers will simply add more links to the arm. While this can be effective, a better approach might be to make agility a primary goal of the design and change the entire nature of the manipulator to better suit this goal. This is the case with continuum manipulators.

4.3 Hyper-redundant and Continuum Manipulators

There exists a subset of tasks which require more agility than that offered by rigid-link manipulators. These tasks, such as surgery, working through small access holes, or in complex obstacle-ridden environments, can be handled by a class of *hyper*-redundant manipulators, manipulators which have many more degrees of freedom within their configuration space than within the task space. One such manipulator is called a continuum manipulator.

A continuum manipulator, also called a snake-arm robot or elephant's trunk robot, is a type of hyper-redundant manipulator composed of non-rigid links. By being kinematically redundant, a continuum manipulator can work well in complex environments which would place additional constraints on the motion of the manipulator. For example, the high kinematic redundancy would make it easier for the end-effector to trace complex trajectories, as portions of the arm could be moved into a more ideal tracing posture without effecting the position of the end-effector, whereas a traditional rigid link manipulator could often find this task difficult or even impossible. Finally, a continuum manipulator can be designed to have the additional benefit of being structurally compliant, making it better suited to working in environments containing people.

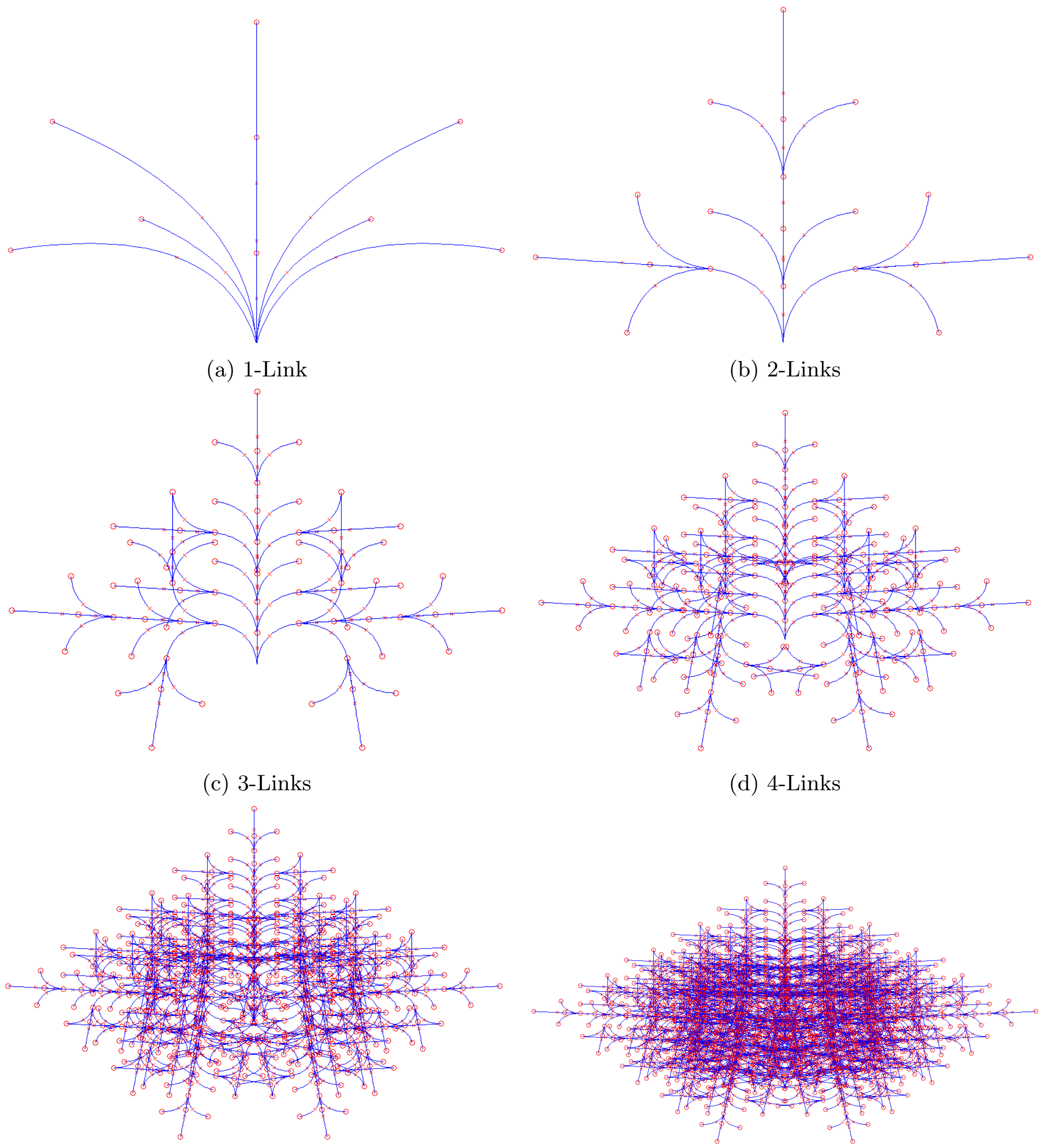
In certain continuum designs, all the actuators for the arm are built into its base. By offloading the actuators, the arm itself can be lighter and have lower inertia than an intrinsically actuated manipulator. This lower inertia makes it safer to use in environments with people. These manipulators are also usually compliant by nature, so the risk of damage or human injury is reduced in any cases where there is a high risk of collision. Natural prismatic abilities can create a larger workspace than a manipulator with a similar number of degrees of freedom and the manipulator has the benefit of being easily scaled to small sizes.

4.3.1 Workspace

A typical continuum manipulator in 2D has a work area like that shown in figure 1. Though this design creates a very large amount of maneuverability, especially when using a very large number of links, it is not without its limitations. These limitations are most easily seen by examining the 1-link case in figure 1a. In this figure, one can see the rotation of the end-effector is coupled with the translation of the end-effector. For a Cartesian coordinate system, the 1-link case is "always in a singularity with respect to a two dimensional positioning requirement" [11]. As links are added to the system, the Constrained Manipulability Ellipsoid (CME), which for the 1-link case is a line perpendicular to the end-effector, expands and approaches the Global Manipulability Ellipsoid (GME), which encompasses all possible changes in position, even those not physically reachable [11]. In other words, imagine the 1-link manipulator attempting to lift a full glass of water and placing it in another nearby location. This manipulator would be unable to accomplish this task for two reasons. First, the manipulator would need to lift the glass using an arc motion, causing spilling. Second, to maintain the glass's original orientation, the arm would be constrained to placing the glass in a location along the same radial vector as it was previously unless the manipulator had an additional degree of freedom at its base, allowing it to rotate the arm about its primary axis.

4.3.2 Existing Manipulators

Continuum manipulators are still considered an emerging field, but already many different varieties exist. Some are based upon pneumatics or muscle-wire, while others are tendon-based. Some are only capable of curving, while others have some prismatic ability, where the continuum links are capable of stretching or retracting to achieve even greater maneuverability within the workspace.


(a) 1-Link (b) 2-Links

(c) 3-Links (d) 4-Links

(e) 5-Links (f) 6-Links

Figure 1: Workspace of a Traditional Continuum Manipulator with n-links.

4.3.2.1 Tendon-Based Tendon-based manipulators involve the use of extrinsically or intrinsically activated tendons that bend a continuous structure [24]. Most tendon-driven designs have some form of elastic backbone along their length and are biologically inspired by snakes, which have backbones consisting of vertebrae linked together with tendons and muscles [9]. Examples of continuum manipulators typically seen in this type are made of plates that are interconnected with either cables or rods and are actuated in such a way that these plates can pivot to make the structure move in interesting ways. Although a tendon design may seem structurally weak as compared to traditional rigid manipulators, different implementations can employ backbones with varying levels of stiffness to suit the application at hand. Backbones can include springs that encompass the rods or cables being driven, rods that provide support in the center of the arm, and pneumatic muscle backbones where the stiffness can be regulated by air pressure [13].

The first continuum manipulator developed used a tendon-based system. This system was developed by the navy for underwater applications. The "Tensor Arm" as it was called used motors that wound up cables connected to plates to create a pivoting action between each of the plates [22]. A disadvantage of using a cable driven system like the "Tensor Arm" was the fact that you cannot push a rope and therefore you can only rely on the pulling retraction of the cable for movement.

Several different advancements were made on this particular model of actuation. Festo designed an arm in 2011 that utilized a roller mechanism and belt system that extrinsically actuated and deflected rods to form a continuous bending structure. This design removed the problem seen with the "Tensor Arm" in that it has the ability to do both pushing and pulling actions to achieve a greater range of motion. This particular design was able to move payloads of 400g and is lightweight, compact, and energy efficient [2].

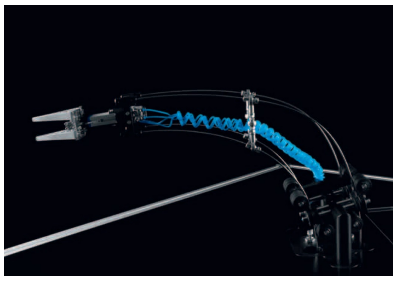


Figure 2: Festo's Bionic Tripod 3.0

OCRobotics in the UK is a company that specializes in the development of continuum manipulators. OCRobotics has designed both extrinsically actuated and intrinsically actuated manipulators. The extrinsically actuated systems use cables to drive arms much like the "Tensor Arm" [8]. The intrinsic actuation design utilizes three actuators connected in parallel between two plates. The differences in length when actuated causes the bending of the structure [6]. An advantage of intrinsically actuated

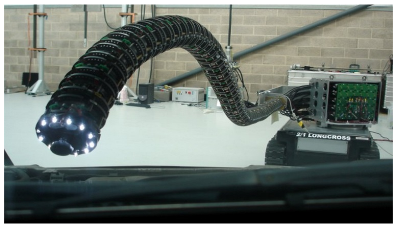


Figure 3: OCRobotics' Snake Arm

designs is that they can provide greater strength and structure to the arm, but are also bulkier and can be unnecessarily complicated.

4.3.2.2 Pneumatics Other forms of intrinsically actuated continuum manipulators employ pneumatics as their mode of actuation. Examples of these robots include Festo's "Bionic Handling Assistant" and Clemson University's "Octarm". Both implementations of the arm derived from models in nature. The bionic handling system is taken from the example of an elephant's trunk and the "Octarm" design is derived from an octopus's flexible arms. Both of these systems use pneumatically controlled muscles that expand and contract based on airflow. Both designs connected these pneumatic muscles to plates to create separate sections that can be controlled independently to make the arm move [23] [1].

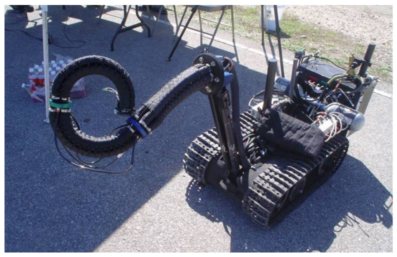


Figure 4: Clemson University's Octarm

4.3.2.3 Concentric Tube There are also continuum manipulators that use concentric tubes to act as a backbone and provide a rigid structure. The arrival of this design was due to the need for greater torsion and extension [24]. A concentric tube manipulator uses a series of concentric tubes that are interconnected and are typically intrinsically actuated. In this design, it is more common to have the outer rods be more rigid and the inner rods more flexible. The intent of a notable concentric tube design published by the IEEE was to develop an easier way of varying the stiffness of the arm such that the user can select which tip or joint to vary the stiffness of [15]. This design is interesting in the way it works; however, it does not provide for sufficient bending of the backbone and therefore makes the structure less flexible. The use of precurved tubes managed to solve a part of this problem, but causes other complications [24].

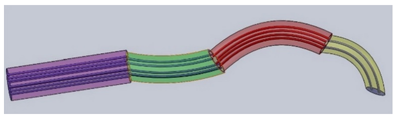


Figure 5: CAD of Concentric Tube

4.3.2.4 Soft Robotics Other forms of continuum manipulators have been seen in the field of soft robotics. Many of these variations are concentric tube or tendon-based, however the difference between them and traditional tendon-based and concentric tube manipulators lies in their actuation. Soft robotic continuum manipulators most often use smart materials such as muscle wire or electro-active polymers for actuation. Concentric tube designs used NiTinol memory alloy tubes to allow for greater flexibility and provide an easier way to control the stiffness of manipulators in a compact way [4]. These modes of actuation are not well suited for our application because shape memory alloys are actuated very slowly due to the fact that they need to be heated and cooled to be manipulated in different ways and electro-active polymers do not provide much structure and are relatively slow compared to designs with pneumatics and/or motors [14].

4.3.2.4.1 Interlocking Fibers A recent development in the field of soft robotic continuum manipulators involves the use of interlocking fibers for the backbone and a tendon-like structure. This design uses two or more elastic beams that interlock with each other to form a dovetail-like mechanism that prevents the beams from moving apart laterally, but allows them to slide past each other along their axis. When forces are applied to the base of each beam, the beams deform to form a curve. This design is different from previous continuum designs in that it provides a stronger manipulator within a smaller diameter that is more compact than tendon-based designs. It also has benefits over concentric tube designs in its use of interlocking fibers, which can bend in two axes without worrying about which direction to twisting which is what makes the concentric tube design more complicated. Although these benefits are numerous, the downside to this design lies in the cost. This design uses materials that are very rare and expensive with difficult fabrication processes [16].

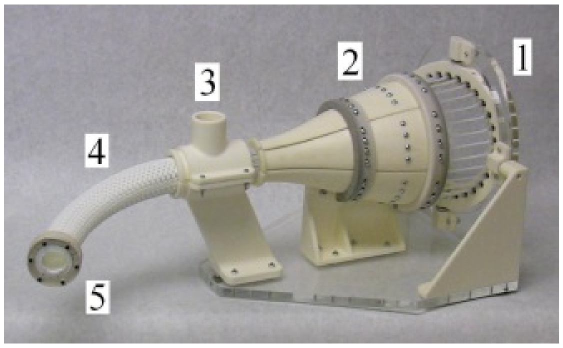


Figure 6: Johns Hopkins University Applied Physics Laboratory Prototype of the Interlocking Fiber Manipulator; 1) Gimbal, 2) Guide Funnel, 3) Support Tube and Air Fitting, 4) Interlocking Fibers, 5) End-cap

4.4 Parallel Continuum Manipulators

Recently, the Reach Lab at The University of Tennessee published a paper [7] about a new form a Continuum Manipulator termed a *parallel* continuum manipulator. Parallel continuum manipulators first appeared in a patent application in 2008 [25], but don't appear to have been studied much until recently. Most continuum manipulators are required to have constant curvature in which each segment must bend along a single continuous arc. Parallel continuum manipulators are special because they do away with this requirement. In addition to bending along their length and translating along their primary axis, they can also translate laterally and rotate about their primary axis. In this way they can be likened to a flexible Gough–Stewart as they have a similar number of actuators (for the single segment case) in a similar configuration. By configuring the actuators in this fashion, parallel continuum manipulators can achieve more mobility in a smaller number of segments, reducing the overall complexity of the system.

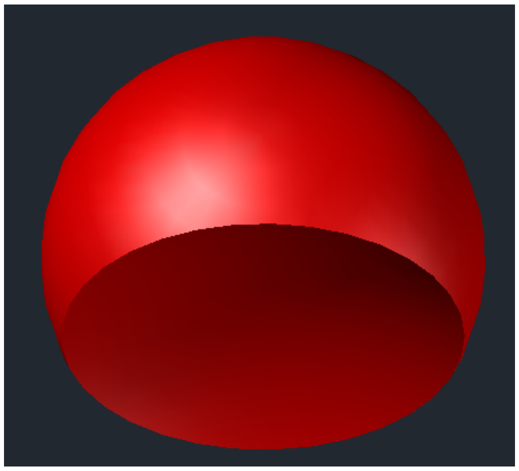
5 Design Requirements and Specifications

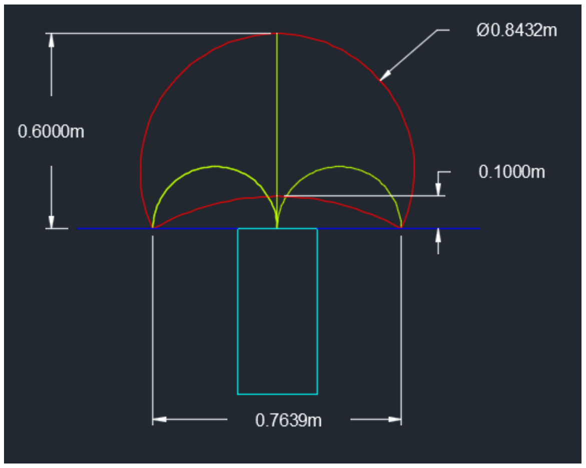
Background research reveals that continuum manipulators may offer many advantages over traditional rigid-link manipulators. In particular, continuum manipulators should perform better within obstacle-ridden environments, environments that contain people, and in situations where it is undesirable to deviate from strict trajectories. Parallel continuum manipulators may offer additional stiffness and strength while extrinsically actuated manipulators offload the weight of the actuators to create a lighter, safer arm. Since the only parallel continuum manipulator in existence today has only a single six degree of freedom segment, this project sought to advance this area by creating a multi-segment extrinsically actuated parallel continuum manipulator.

5.1 Itemized Requirements

Below is an itemized list of project goals.

- Minimum of two continuum segments
- Workspace dimensions of at least those shown in Figure 7b
- Within the workspace, capable of rotating at least 120° about the axis of the manipulator perpendicular to the base
- At least one of the continuum segments must use a parallel Gough–Stewart actuator configuration, allowing the segment to move in six degrees of freedom
- Capable of applying at least 5 N of force throughout its workspace excepting singular configurations
- End-effector is capable of withdrawing from a position and returning to that position within ± 6.25 mm
- Minimum no-load end-effector speed of 400 mm/sec
- The manipulator will have some compliance, but will not displace more than 20 mm when applying a 5 N load
- Software will include:
  - Agility demonstration
  - Inverse kinematic control
  - Basic path planning


(a) Specified Workspace of the Manipulator



(b) Dimensioned Cross-Section of Workspace

Figure 7: Workspace Requirements

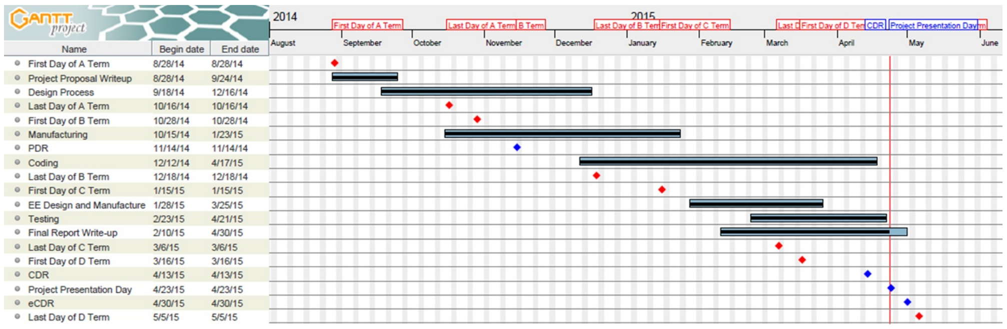
6 Design and Analysis

This section describes the design process used throughout the project including mechanical, electrical, and software design as well as relevant mathematical calculations and overall project planning.

6.1 Project Planning and Logistics

6.1.1 Scheduling

A detailed Gantt chart was made for planning and tracking the project's progress. Figure 8 shows when the major portions of the project were worked on and completed.



Name	Begin date	End date
First Day of A Term	8/28/14	8/28/14
Project Proposal Writeup	8/28/14	9/24/14
Design Process	9/18/14	12/16/14
Last Day of A Term	10/16/14	10/16/14
First Day of B Term	10/28/14	10/28/14
Manufacturing	10/15/14	1/23/15
PDR	11/14/14	11/14/14
Coding	12/12/14	4/17/15
Last Day of B Term	12/18/14	12/18/14
First Day of C Term	1/15/15	1/15/15
EE Design and Manufacture	1/28/15	3/25/15
Testing	2/23/15	4/21/15
Final Report Write-up	2/10/15	4/30/15
Last Day of C Term	3/6/15	3/6/15
First Day of D Term	3/16/15	3/16/15
CDR	4/13/15	4/13/15
Project Presentation Day	4/23/15	4/23/15
eCDR	4/30/15	4/30/15
Last Day of D Term	5/5/15	5/5/15

Figure 8: Project Gantt Chart

The first few weeks of the academic year were spent performing background research and deciding upon the goals the project should focus on. After completing the project proposal, the project's design phase began which included initial calculations, material and budget selection, electronic design, and prototyping. The manufacturing phase began about a month after starting the design phase. Software development began as the electrical and mechanical design phases neared completion. The end-effector was designed and built after completing the manufacturing and assembly of the manipulator. Testing took place during the last couple months of the academic year along with the writing of the final report. Some important milestones are placed on the gantt chart with red indicating the beginning and ends of academic terms and blue indicating presentations and key project deadlines.

6.1.2 Budget

The budget was projected to be around $2,000. This includes all prototyping expenses purchased near the beginning of the term as well as large purchases such as actuators and stock material purchased during construction of the final manipulator.

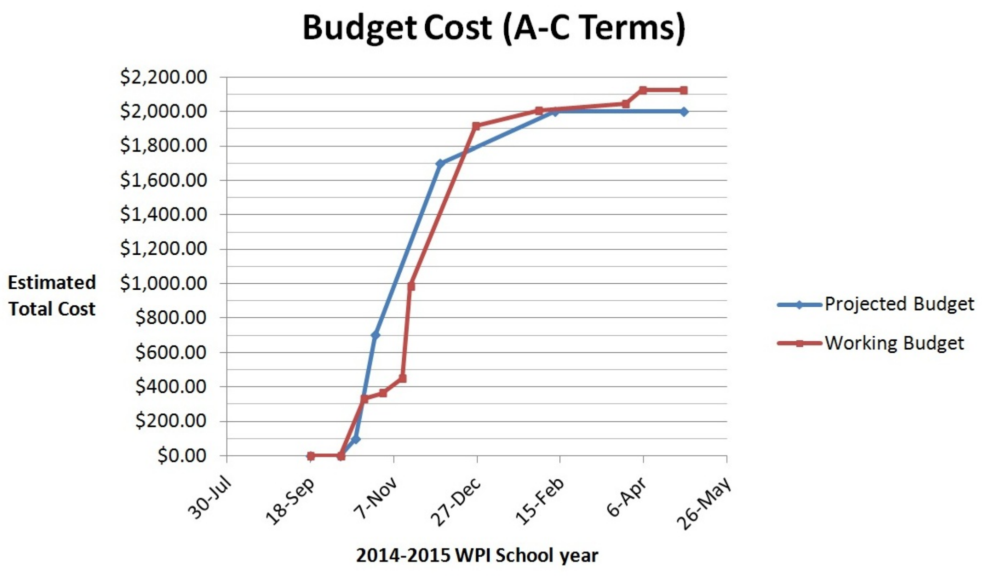


Figure 9: Projected and Resulting Budget

The final project cost was $2,120.81, which is only slightly above the original projection. Figure 9 compares the projected budget with the actual budget. Discounts and sponsorships from Pololu and Advanced Circuits helped to reduce the overall cost of the project. Pololu supplied the team with a 20% discount on stepper motors and drivers and Advanced Circuits fabricated the printed circuit board (PCB) for free.

6.2 Mathematical Discussion

6.2.1 Kinematics

The REACH (Robotics, Engineering, Applied Continuum Mechanics, and Healthcare) lab at the University of Tennessee established a method for solving the inverse kinematics of single segment parallel continuum manipulators using Cosserat rod mechanics [7]. REACH models each leg of the manipulator using the differential equations seen in equation 1. Derivatives are with respect to the arc length of the rod. Table 1 describes the variables referenced by the Cosserat rod equations.

$$\begin{array}{ll} \boldsymbol{p'}_i = R_i \boldsymbol{v}_i, & \boldsymbol{v}_i = \boldsymbol{v}_i^* + K_{se,i}^{-1} R_i^T \boldsymbol{n}_i \\ R_i' = R_i \hat{\boldsymbol{u}}_i, & \boldsymbol{u}_i = \boldsymbol{u}_i^* + K_{bt,i}^{-1} R_i^T \boldsymbol{m}_i \\ \boldsymbol{n'}_i = -\boldsymbol{f}_i & \\ \boldsymbol{m'}_i = -\boldsymbol{p'}_i \times \boldsymbol{n}_i - \boldsymbol{l}_i & \end{array} \tag{1}$$

Description	**Equation**
Derivatives with respect to arc length	$s_i \in \mathbb{R}$
Leg position	$\boldsymbol{p}_i(s_i) \in \mathbb{R}^3$
Leg orientation	$R_i(s_i) \in SO(3)$
Internal forces	$\boldsymbol{n}_i(s_i) \in \mathbb{R}^3$
Internal moments	$\boldsymbol{m}_i(s_i) \in \mathbb{R}^3$
External forces and moments	$\boldsymbol{f}_i$ and $\boldsymbol{l}_i$
Local kinematic variables (shear, extension, bending, torsion)	$\boldsymbol{v}_i(s_i) \in \mathbb{R}^3$ $\boldsymbol{u}_i(s_i) \in \mathbb{R}^3$
Kinematic variables in a stress free reference state	$\boldsymbol{v}_i^* = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ $\boldsymbol{u}_i^* = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$
Mapping from $\mathbb{R}^3$ to $\mathfrak{so}(3)$, $\vee$ is the inverse mapping	$\hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$
Cross-sectional area	A_i
Young's modulus	E_i
Shear modulus	G_i
Second area of moment	I_i
Polar area moment	J_i
Constant Matrices	$K_{se,i} = \text{diag}(A_iG_i, A_iG_i, A_iE_i)$ $K_{bt,i} = \text{diag}(E_iI_i, E_iI_i, J_iG_i)$

Table 1: Cosserat Rod Variable Description

The Cosserat rod equations describe an individual rod in the system. By enforcing various geometric constraints, it is possible to solve for the unknown values shown in Table 2. The geometric constraints are represented by equations 2 through 6. Equation 2 states that an individual leg cannot support a torsional load. Equations 3 and 4 state that the system must be statically stable with the sum of the forces and moments equaling zero. Equations 5 and 6 state that the distal ends of the rods must line up with the bolt pattern in the plate and must be perpendicular to the plate.

Unknown	**Physical Significance**
$n_{xi}(0), n_{yi}(0)$	Internal shear forces at base of i^{th} rod
$n_{zi}(0)$	Internal axial force at base of i^{th} rod
$m_{xi}(0), m_{yi}(0)$	Internal bending moments at base of i^{th} rod
L_i	Arc length of the i^{th} rod

Table 2: Geometric Constraint Unknowns

$$m_{iz}(0) = 0 \tag{2}$$

$$\sum_{i=1}^{n}[\boldsymbol{n}_i(L_i)] - \boldsymbol{F} = \boldsymbol{0} \tag{3}$$

$$\sum_{i=1}^{n}[\boldsymbol{p}_i(L_i) \times \boldsymbol{n}_i(L_i) + \boldsymbol{m}_i(L_i)] - \boldsymbol{p}_d \times \boldsymbol{F} - \boldsymbol{M} = \boldsymbol{0} \tag{4}$$

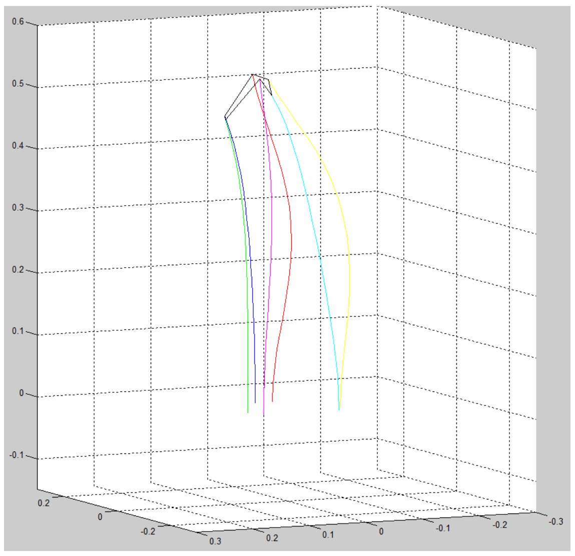
$$\boldsymbol{p}_d + R_d\boldsymbol{r}_i - \boldsymbol{p}_i = \boldsymbol{0} \text{ for } i = 1...n \tag{5}$$

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\end{bmatrix}[log(R_i^T(L_i)R_d)]^V = \boldsymbol{0} \text{ for } i = 1...n \tag{6}$$

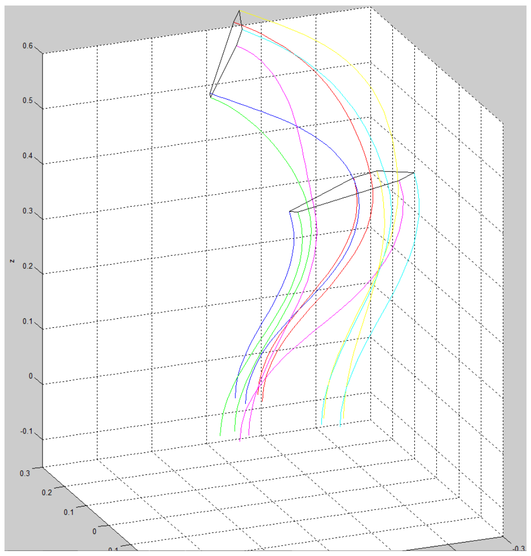
This model works equally well for a multi-link manipulator with only minor modifications. One simply combines two six DOF manipulators serially and solves for them simultaneously, using the final positions of the bottom link (averaged, then arranged to match the bolt pattern) as the initial positions for the rods in the top link.

With six non-linear differential equations for each rod and twelve rods to solve for, the final system has a total of seventy-two simultaneous differential equations with seventy-two unknowns. This system can be solved using the Runge–Kutta shooting method which is an iterative process in which a guess for the unknowns is made, then the legs are integrated over their lengths and a residual is evaluated using the geometric constraints to determine how close the guess is to a correct solution. A jacobian matrix is then built showing how the individual residual terms change with respect to changes in the guess. This jacobian can be used to produce a better guess. This process repeats until the system converges within some desired tolerance on the solution.

This project did not focus on writing a simultaneous differential equation solver, but rather used MATLAB's built in fsolve function for initial testing and later Google's Ceres solver [3] for the final C++ implementation. The below figures compare the MATLAB model for REACH's one-link manipulator to the two-link version developed in this project. Both models were given the same desired pose. The one-link model shown in Figure 10a was not able to find a solution and stopped with a residual of 0.117. The two-link model shown in Figure 10b found a solution with a residual of 9.096×10^{-16}, demonstrating some of the benefits of using an additional link.



(a) One-Link Kinematic Model



(b) Two-Link Kinematic Model

Figure 10: MATLAB Kinematic Models

6.2.2 Stroke Length

The relationship between maximum manipulator bend angle, manipulator width, and linear stroke size is shown in equation 7 and illustrated in figure 11. Δs is the length difference between two parallel legs, w is the width of the manipulator, and θ is the resulting angle. This relationship was used for determining how much linear actuation was needed to achieve the workspace specified in the design goals. The final design used a linear stroke of twenty inches.

$$\theta = \frac{\Delta s}{w} \tag{7}$$

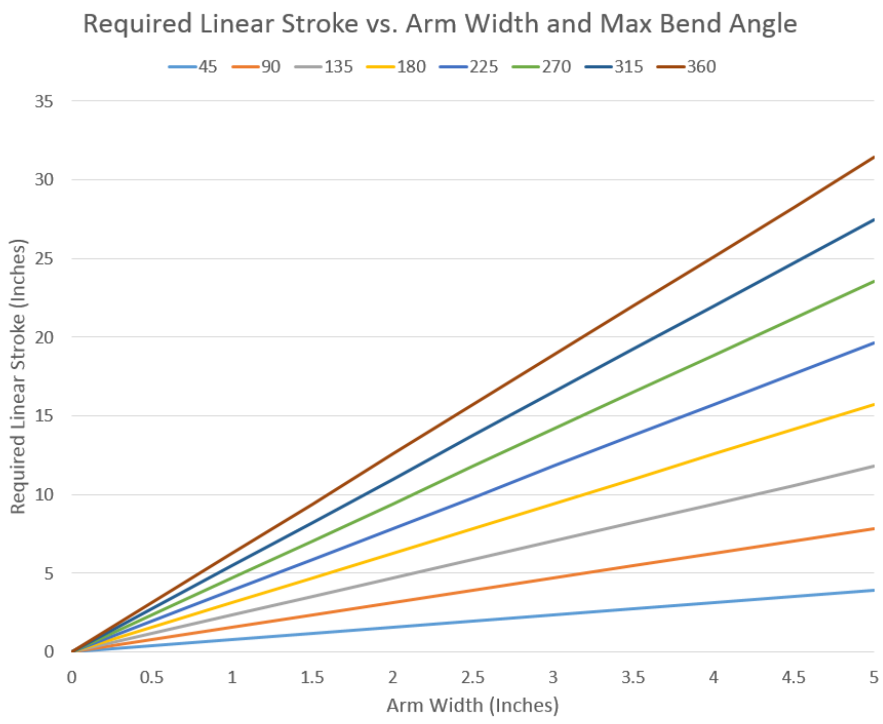


Figure 11: Required Linear Stroke For Various Arm Widths

6.2.3 Bending Torques

The potential energy stored in a bent rod is equal to the area under the stress strain curve. It can be computed using equations 8, 9, and 10. Table 3 describes the variables used in these equations [10].

$$U = \frac{1}{2}V\sigma\epsilon = \frac{1}{2}VE\epsilon^2 \tag{8}$$

$$\epsilon = \frac{y}{\rho} \tag{9}$$

$$\rho = \frac{s}{\theta} \tag{10}$$

Description	**Variable**
Stress	σ
Strain	ϵ
Radius of Curvature	ρ
Volume	V
Young's Modulus	E
Potential Energy	U
Bend Angle	θ
Arc Length	s
Effective Distance	y

Table 3: Bent Rod Potential Energy Variable Description

Using these equations, it was possible to determine the rod diameter needed to ensure that enough energy was stored to lift the minimum required weight as detailed in the project goals as well as the amount of power the motors need to output to bend the rods over the maximum bend angle. At least three joules of energy are required to lift an object weighing five newtons to the top of the workspace and doing so at a speed of 0.4 m/s requires two watts of power. The selected rod diameter was about 1.8 mm, which over a 180° bend stores 3.5 joules of energy and moving at the required speed uses 2.3 watts of power. This is enough to lift the desired load.

6.2.4 End-Effector

6.2.4.1 Gripping Force Calculations The project requirements set a lifting load of 5 N. Equations 11 and 12 were used to calculate the gripping force (F_n) required by the end-effector to effectively manipulate the 5N load, taking in the load and the coefficient of friction as parameters.

$$\begin{gathered} W = ma + mg \\ W = 5N \end{gathered} \tag{11}$$

The static coefficient of friction, μ, of rubber on plastic is 0.5. The gripper was coated in rubber after it was printed to better grip the objects that it would hold. Thus this coefficient of friction was used for calculating the required gripper force of $\frac{5}{4}N$.

$$F_n = \frac{W}{2}\mu$$
$$F_n = \frac{5}{2}0.5 = \frac{5}{4}N \tag{12}$$

6.2.4.2 Wrist Force Calculations In order to actually rotate the wrist of the end-effector, the forces required to move a 5N object from the wrist had to be calculated.

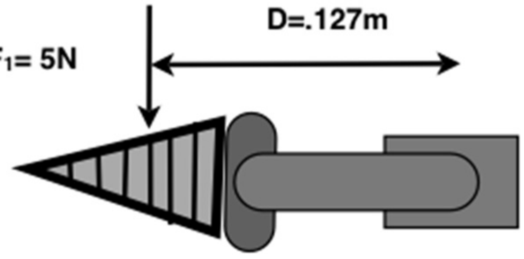


Figure 12: End-Effector Analysis

$$\tau = F_1 D = (5)(.13) = 0.64Nm \tag{13}$$

Equation 13, complimented by Figure 12, was used to find the amount of torque needed to lift a 5N load at the end of the gripper, with the minimum amount of torque needed to lift a 5N load being 0.64 Newton-meters for our end-effector.

6.2.5 Mechanical Analysis

6.2.5.1 Linear Slide Binding Calculations Equations 14 through 20 were used to calculate the maximum distance between the bushing holes of the linear assembly in order to prevent binding. The variables used in these equations are described in Table 4. The Free Body Diagram (FBD) used for the binding analysis can be seen in Figure 13.

Description	**Variable**
Applied Force	F_1
Resulting Force 1	F_2
Resulting Force 2	F_3
Dragging Force 1	F_4
Dragging Force 2	F_5
Coefficient of Friction	μ
Length of Bushings	L_1
Distance Between Bushings	D_1

Table 4: Binding Ratio Equations Variable Description

Using these equations, it was possible to determine the maximum distance between the bushing holes within the brackets of the linear actuator assembly. These calculations were needed to ensure the bushing holes of the bracket were positioned close enough to prevent binding while actuating the linear assemblies. The depth of the bushing holes was defined as 1 inch to make sure the bracket did not limit the stroke length of the actuator. The maximum distance between the bushing holes was calculated to be 3.6 inches as seen in equation 20, however to compensate for the unwanted forces of the spring steel rod being slightly off center, the distance was kept at 1.9 inches to add a factor of safety.

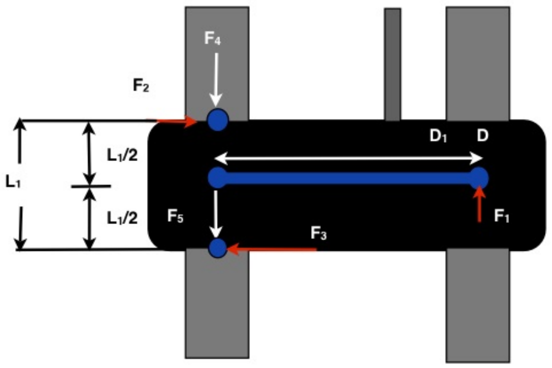


Figure 13: FBD of Bracket

The bracket accelerates when the applied force (F_1) is greater than the sum of the drag forces (F_4 and F_5). Equation 14 shows how this system can be solved for the minimum applied force (F_1) needed to accelerate the bracket.

$$\begin{aligned} F_1 > F_4 + F_5 &\rightarrow \text{Acceleration} > 0 \\ F_4 &= \mu F_2 \\ F_5 &= \mu F_3 \\ F_1 > \mu F_2 + \mu F_3 &\rightarrow \text{Acceleration} > 0 \\ F_1 > \mu(F_2 + F_3) &\rightarrow \text{Acceleration} > 0 \end{aligned} \tag{14}$$

For these statics equations, point C was chosen as the fulcrum. All forces and moments were summed around this point. Figure 13 shows the lever arms resulting from the applied forces (F_1) and the resulting force (F_2 and F_3) with an origin of point C. Since the system was designed to be rotationally stable, the sum of the moments (about point C) was equal to zero. Equation 15 shows this system solved for the resulting forces F_2 and F_3.

$$\begin{aligned} \text{Rotation Stable } &\rightarrow \sum(M_z) = 0 \\ \sum(M_z) &= F_1 D_1 - \tfrac{F_2 L_1}{2} - \tfrac{F_3 L_1}{2} \\ F_1 D_1 &= \tfrac{F_2 L_1}{2} + \tfrac{F_3 L_1}{2} \\ F_1 D_1 &= L_1 \tfrac{F_2 + F_3}{2} \\ \tfrac{2 F_1 D_1}{L_1} &= F_2 + F_3 \\ \text{Acceleration} = 0 &\rightarrow F_1 = \mu(F_2 + F_3) \\ F_1 &= \tfrac{2\mu F_1 D_1}{L_1} \end{aligned} \tag{15}$$

Equation 16, shows the relationship between the coefficient of friction (μ), bushing hole distance, and moment arm distance. Equation 16 was used to derive the conditions that must be met for the linear actuator to be capable of motion. Equations 17 through 19 will separate the maximum allowable coefficient of friction (μ), the maximum allowable moment arm distance (D_1), and the minimum allowable bushing length (L_1). The final value of 3.6 inches is seen by equation 20.

$$1 = \frac{2\mu D_1}{L_1} \tag{16}$$

$$L_{1-min} = 2\mu D_1 \rightarrow L_1 > 2\mu D_1 \tag{17}$$

$$D_{1-max} = \frac{L_1}{2\mu} \rightarrow D_1 < \frac{L_1}{2\mu} \tag{18}$$

$$\mu_{max} = \frac{L_1}{2D_1} \rightarrow \mu < \frac{L_1}{2D_1} \tag{19}$$

$$D_{1-max} = \frac{L_1}{2\mu} \rightarrow D_1 < \frac{1}{2 \times .14} = 3.6 \text{ in.} \tag{20}$$

6.3 Mechanics

This section discusses the manipulator's mechanical design including initial design concepts, material selection, CAD development, and prototyping. Manufacturing will be discussed in section 7.1.

6.3.1 General Concepts

The base structure of the manipulator was designed to be robust and capable of supporting itself as well as containing the actuators and linear slides for creating the arm's motion. Three structures were explored for supporting the linear motion: slotted aluminum extrusion (80/20) slides with roller bearings, smooth (drill) rods with linear bearings or bushings, and THK Linear Slides. Table 5 shows a pro/con analysis of the different structures. Both the 80/20 and the THK brand slides proved to be too expensive and bulky and THK brand slides are also very heavy. The smooth rod and linear bearings option met the needs of the project as they were the least expensive, most compact, and lightest, but did require some custom machining to create supporting components, which could be easily done in house.

Structure Type	**Pros**	**Cons**
80/20	- Easy to work with in terms of adjustability and prototyping - Relatively inexpensive - Previous experience working with it	- May be too bulky
Smooth Rods with Linear Bearings	- Less bulky than 80/20 - Previous experience working with it	- Imperial linear bearings more expensive than metric - Possible issues with mixing metric and imperial
THK Linear Slides	- Very smooth linear system	- Very heavy - Very expensive

Table 5: Pro/Con Analysis on Various Structures

Three types of linear actuators were considered, a comparison can be seen in Figure 6. Screw drives which are used on most CNC machines because of strength and stability, pneumatics which could potentially provide more force, and a belt driven system using steppers. The screw drive mechanism was ruled out of the design because, although it has the potential to provide a lot of mechanical advantage, it could not easily meet the required speed goals. Pneumatics were ruled out from a control perspective as they would require a complex valve system to achieve positional control. The third option of a belt driven system with stepper motors was selected because steppers have enough power to effectively move the linear actuators, are precise, and are easily controlled.

Actuator Type	**Pros**	**Cons**
Belt and Pulley System	- Relatively inexpensive - Compact - Easy to work with	- Possible inconsistencies with belt tension Possible weaknesses in belt (snapping)
Screw Drive	- Provides system with increased mechanical advantage	- Slow movement - Quality lead screws expensive
Pneumatics	- A lot of force output for small size	- Needs an on-board compressor - Needs elaborate valve system for linear translation - Have minimal experience working with it

Table 6: Pro/Con Analysis on Various Actuators

The final design decision for linear motion was to use a belt driven system with linear slides consisting of smooth (drill) rod and Delrin bushings.

6.3.2 CAD Modeling: Pre-Prototype

With design concepts planned and materials chosen, the manipulator began being modeled using SolidWorks.

The initial design of the manipulator had been a continuum design, rather than parallel continuum as shown in Figure 14. Each rod that actuated the wire to control the movement of the manipulator was equidistant in a circular pattern. The manipulator had four major areas of design. The bottom stage in the base housed the electronics and Power Supply Unit (PSU).The second stage of the base housed the twelve stepper motors that were positioned to compliment each linear guide assembly. Above the motors was the main base containing the linear guides that would vertically translate the spring steel rod that made up the manipulator. Twelve limit switches were located on the bottom of each linear guide assembly for homing. The upper part of the assembly above the base contains the actual "arm" of the manipulator, where the wire came out and was separated into segments. A plate marked the top of each segment, with the end-effector mounted to the top link plate. The bottom stage that housed the electronics and PSU as well as the EE are not shown in Figure 15.

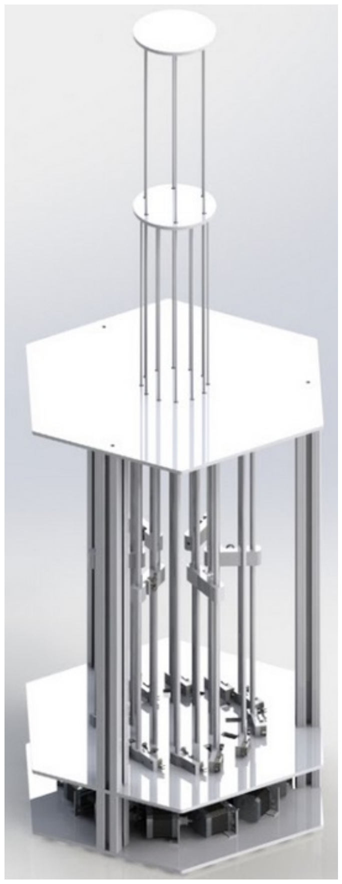


Figure 14: Early Design of the Manipulator

6.3.3 Material Selection

When determining the materials needed for manufacturing, materials were chosen based on cost, material properties, weight, durability, and machinability. The base of the arm was made out of various materials including steel, aluminum, Delrin, and polycarbonate. The bottom base plates were made of 6065 Aluminum so that it would be relatively lightweight and would be able to accept the weight of the motors and the linear actuators with minimal deflection. The other components that were static within the system were made out of aluminum as well. The aluminum was readily available and the weight of these components did not need to be greatly limited. As stated earlier the linear brackets were machined out of Delrin due to its low coefficient of friction on steel, and its durability and low weight. The pulley components at the top plate of the base were machined out of Delrin as well in order to make assembly easier by being able to snap them on and screw them in to the top plate.The bottom plate covering the motors was made out of polycarbonate because the plate was not carrying significant loads therefore, polycarbonate was the more cost effective approach in comparison to aluminum or Delrin. The top plate of the base was machined out of 0.5 inch thick Delrin. The thickness allowed for more support for the wires as they passed through the bushing. The link plates were also made out of Delrin which was again chosen for its properties of low coefficient of friction and light weight. These properties reduced the load on the arm induced by its own weight, as well as made it possible for the spring steel rods controlling the second plate to pass through the first plate without binding.

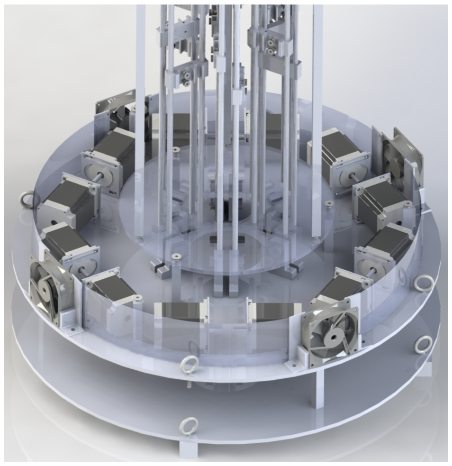
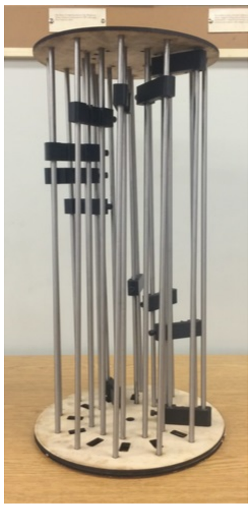


Figure 15: Motor Placement and View in mid design

The final end-effector consisted of combination of machined, 3D printed, and laser-cut parts. The base of the end-effector consisted of a laser-cut acrylic adapter plate. The gears for rotation of the assembly and for opening and closing of the gripper were 3D printed out of ABS to allow for the gear attachment points to be customizable and easily manufacturable. The fins for the gripping component of the end-effector were 3D printed using a flexible material called NinjaFlex which allowed for the fish tail effect. Other components that were manufactured included the adapter brackets for a gimbal attachment for the wrist of the end-effector. These were made out of 1/8” thick aluminum to make the parts light and sturdy. If these parts were 3D printed, there would have been an added risk of them breaking. This material choice allowed for the end-effector to have a more compact design without sacrificing strength and durability.

6.3.4 Prototyping

The first prototype constructed utilized wooden plates that were designed and cut for what would be the base of the manipulator that would consist of the drill rod and eventually the spring steel rods used for actuation. Using the drill rod purchased that would be kept through to the final build, brackets were also 3D printed to go on the drill rod. This helped us visualize the basic structure of the manipulator, shown by Figure 16a, and test the linear actuators within the system.


(a) First Prototype Constructed

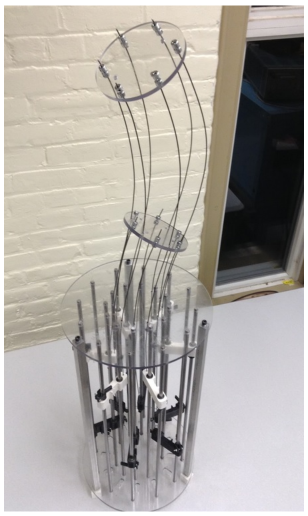

(b) Second Prototype Constructed

Figure 16: Prototypes Made

During testing with this prototype, we had issues with how smooth the brackets vertically translated along the smooth rod. The main verdict came with the plates and that the rods themselves were not parallel to each other within a reasonable tolerance, which caused racking making the linear bearings bind and hindered movement of the linear guides.

The second prototype for this project, Figure 16b, was used to determine which areas in the mechanical design of the manipulator had to be improved. The prototype was a mock-up of the base with the linear guides and the wire of the manipulator connected with the guides on the smooth rod. For the prototype the smooth rod was machined to size in order for the arm to have the proper stroke length of 20 inches. The top and bottom plates of the base as well as the link plates were machined out of polycarbonate, and the linear brackets to be connected to the spring steel rod were 3D printed. The prototype showed that changes to the mechanical design were needed.

6.3.5 Post Prototyping Design Stage

After the prototyping stage of the project many issues were identified and resolved. The hole pattern for the top plate and the arrangement of the linear actuators and wires had to be shifted in order to implement the Stewart-Gough platform. Without the Stewart-Gough platform, the arm would not be able to meet the rotation requirements of the project. However, to gain the additional rotation some prismatic motion was sacrificed. Additionally, in order to achieve the workspace and compliance requirements for the arm, in terms of strength and flexibility, the maximum diameter of the arm remained at 5 inches for the outside diameter of the bottom link and the top link remained at 4.65 inches in diameter. In order to fit the linear actuators within the relatively compact system, major design changes were necessary. The size of the brackets for the linear actuators were decreased by approximately 25 percent and the linear bearings could no longer fit without the walls of the brackets becoming too thin to maintain its structural integrity. In order to retain work that had already been done attaining and machining the drill rod, a new bracket was designed. The brackets were machined out of Delrin and instead of using linear ball bearings the Delrin acted as its own bushing. The holes that originally were going to contain the linear bearings were drilled and reamed to achieve a close running fit. The close running fit allowed for smooth actuation, but also maintained minimal play in holes to prevent unwanted forces on the part which could cause racking and hinder its motion. This design change greatly decreased the size of the part and eliminated the need and cost of ball bearings, as well as, allowing the diameter requirements of the arm to be met. Having the parts machined out of Delrin instead of another material not only allowed for the part to have built in bushings, but provided a sturdy, lightweight material perfect for this application. Another change which occurred in the bracket design having the potential to cause a problem, was the relocation of the holes for the spring steel wire to go into the bracket. The movement of the holes on the bracket was only slightly off center, but this still had the potential to cause racking. To reduce the potential for racking, induced by the forces resulting in differing alignment of the rods, the bushing holes were made slightly longer and the distance between the two drill rods for the linear actuator was moved closer together. The change maintained the maximum ratio of the moment arm distance to bearing length to prevent binding. This greatly decreased the forces on the bracket to a point where they were insignificant and the bracket would still be able to move smoothly. The calculations to determine the maximum distance between the bushings can be seen in equation 18 from Section 6.2.5.1.

Another potential problem identified during the prototyping stage was the danger of the arm undergoing unwanted bowing at the bottom of the base under the top base plate of the arm and at the links. The position of the wire had to be controlled such that the arm would meet the precision requirement set, and move with minimal bowing. Calculations were done in order to determine the forces that occurred where the wire started to come out of the top base plate, and the first link plate at the sharpest bend possible. Several approaches to this problem were taken into account in order to reduce the forces acting directly on the opening holes on the top base plate. Originally we had holes that were a close running fit for the wire to go through. This had worked for the prototype, however, the adoption of the Stewart-Gough platform changed the forces on the holes on the top plate. The wires were constantly being bent at a 20 degree angle which made it virtually impossible to continue with just a normal hole in polycarbonate. Originally it was decided that the top plate should be made from Delrin because it was a material that was relatively soft and had a low coefficient of friction. Other possible solutions considered included a roller bearing system that the wire could run through that could guide the wire at the bend instead of just having the wire wear down the hole, the use of spherical bearings, or using a bushing with a countersunk and ideally filleted hole. A roller system was prototyped and was made small enough to fit with two stages to constrain the wire in order to have the wire have controlled motion in the bending, although the design seemed feasible, the time needed to manufacture twelve custom machined two-stage brackets that could withstand the calculated loads would have been outside our time constraints. The use of spherical bearings also came up. Sample bearings were tested, but it was realized that the use of the spherical bearings could drastically change the kinematic model and make control of the manipulator a more difficult task than it already was. The use of spherical bearings for each of the wires would have also been expensive and difficult to customize for the small diameter wire. In order to have prevented further play in the system than what spherical bearings already introduced, a quarter of an inch diameter cylinder would have to have been machined, drilled with a 0.072 inch hole,and pressed into each of the spherical bearings.

The original simpler approach to the design was taken and bushings with countersunk edges were chosen. The material choice was the next step to fixing the issue. Polycarbonate, although a strong, lightweight material is not very smooth after it is machined. Many other materials were considered and lubricated bronze seemed like a viable solution. It had a very low coefficient of friction against steel and would not machine away as fast as plastics would with repeated sliding motions against the surface of the bushing through the motion of the arm. These bushings also had to be custom machined because of the constraint of the extremely small diameter of the wire. These bushings also employed the use of a close running fit. After machining and testing it out on the manipulator it was found that this had not worked as expected. The arm had trouble moving and another solution was needed. Instead of using lubricated bronze for the bushing material, Delrin was used and countersunk in the same way which reduced the friction and eliminated the racking. It was understood that repeated use, would cause the Delrin to wear away much faster than bronze. To allow for this the bushings were made to be easily replaceable. Although there was some bowing in our final design, the arm was able to move as required by our project criteria.

Most of the challenge for the rest of the components within the design were making the parts small enough to fit within a certain envelope of space, as well as keeping the price to a reasonable level. The linear brackets had to be brought down to size as described previously. Additionally, the motors and other pulley components also had to fit in the space in the base due to constraints in manufacturing sizes that needed to be taken into account in order to make it manufacturable with the resources given. The bottom plates were limited to 20 inches in diameter. Initially the motors were NEMA 17 stepper motors, but then the decision was made to increase the size of the motors to NEMA 23 which are larger but provide more power and torque. In order to fit all of the motors within the existing space they had to be placed at the edges of the plate. This changed how the pulley system needed to be set up. In order to connect the belt to the motors and direct the belts to their desired positions, a pulley block was designed and implemented. The pulley block consisted of 2 rotational ball bearing idler pulleys offset from each other, attached to a machined aluminum block. The pulley block redirected the belts from the linear bracket, to the motor and back. This proved to be an effective solution that allowed for the use of the larger motors.

6.3.6 CAD Modeling: Post-Prototype

The final iteration of the arm shown in Figure 17 takes into account the parallel nature of the manipulator and implements the Stewart-Gough Platform for parallel manipulators. To achieve this,the hole pattern for the top base plate and the alignment of the drill rod and spring steel rods had to be changed which resulted in a more compact design.

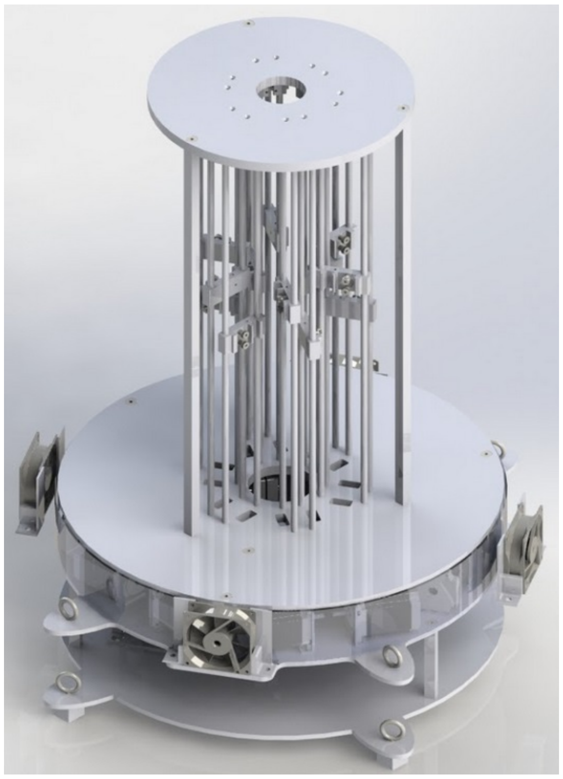
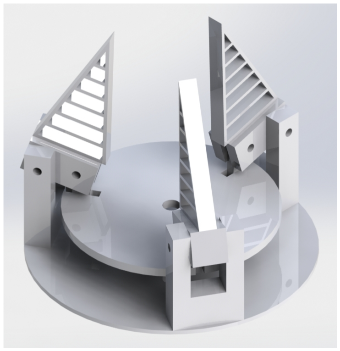


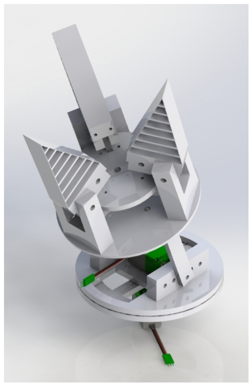
Figure 17: Final Design of Base

6.3.7 End-Effector

Several different types of grippers were proposed when choosing the end-effector for the manipulator. The final design utilized flexible fish fin fingers as the gripper of choice. The flexible fingers allowed us to maintain a compliant structure for the gripper and allowed for the manipulation of various objects with the nature of the fish fin design. The fish fin design used fingers with internal supports that would bend the finger inward when a force was applied to them through flexible buckling allowing for a tight grip on different shaped objects, this design also reduced the weight of the gripper.



(a) Initial Gripper Design



(b) Initial End-Effector Design

Figure 18: Prototypes Made

The end-effector for the arm was designed to add two additional degrees of freedom to the arm. The original design for the end-effector as seen in Figure 18a employed the use of a screw drive driven by a continuous motor that would actuate a plate up or down up to certain points using feedback from limit switches.When the plate was actuated to its highest and lowest heights, this would pivot brackets attached to flexible fingers to open and close the gripper. This design also had a wrist and a rotational component at its base. Although this gripper could have served its purpose in being able to grasp a bottle, it was very bulky, required several parts, and its motors were not strong enough to lift it. A second design was made that made it slightly less bulky, but tried to salvage as many parts from the original as possible, however, this proved to be more difficult than redesigning it fully, therefore, the decision was made to fully redesign the effector instead. The final design of the gripper in Figure 19 was based off of a camera gimbal used for quad rotors. The gimbal has two degrees of freedom and was manufactured out of aluminum instead of being 3D printed and was much more compact. The gripper in the original design used a screw drive in order to actuate 3 gripper fins that were equally spaced in a circular pattern. The final design was a standard angular end-effector that had slightly curved fins. This greatly decreased the size of the gripper needed to be moved by the servo motors for the wrist and the rotational component. The lever arm for the end-effector wrist was also greatly decreased in length, making the forces creating the moment about the motor axle much less significant. The motors for the new design were chosen after various calculations on the moments and force needed for closing the gripper.Three metal geared micro servos with 1.36 Newton-meters of torque were chosen for this application after the calculations seen in equations 12 and 13 in Section 6.2.5.1. The use of a third servo motor rather than a continuous motor, thus also eliminating the need and use of limit switches.

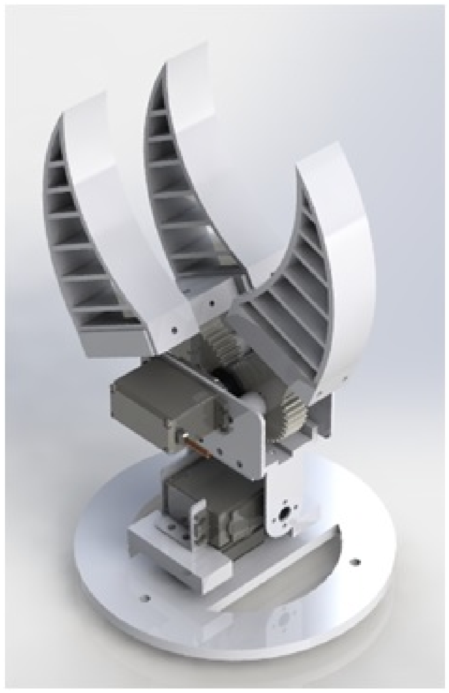


Figure 19: Final End-Effector Redesign

6.4 Electronics

6.4.1 General Concepts

The final manipulator has twelve mid-sized stepper motors, twelve limit switches, three servos, and serial communication to an attached computer. To achieve the necessary speeds and torques, each stepper motor was operated at thirty volts and drew approximately one amp. Due to the large number of devices needed to be controlled and the heavy power requirements, a custom controller board was designed. An overview of the control system is shown in figure 20. High level control operations such as interpreting control inputs and finding solutions to the inverse kinematics model were computationally expensive and were handled by an external computer running Linux and ROS (Robot Operating System). The computer produced the desired positions for the stepper motors and servos and sent these positions over a serial communication interface using an FTDI chip to an embedded microcontroller. The microcontroller keeps track of the current state of the manipulator and sends necessary step commands to onboard stepper drivers which operate the steppers. The board also monitors the end stops during the homing process. A complete bill of materials for the PCB can be found in Appendix B.1.

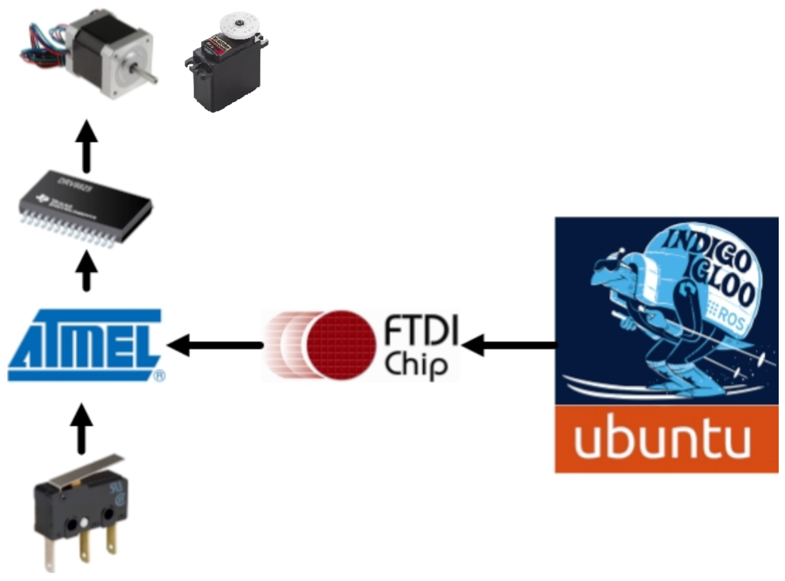


Figure 20: High Level Electronics Overview

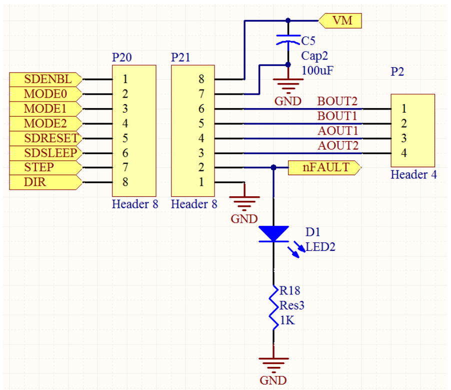
6.4.2 Motor Selection and Power Considerations

Section 6.2.3 discussed the energy and power requirements needed to create the necessary bending motions. To meet these requirements, NEMA 23 bipolar stepper motors were selected. The steppers have a voltage rating of 8.6 V with 1 A current draw per phase and 200 steps per revolution. A Motor driver IC (DRV8825) was selected to match the requirements of the motors. The driver uses a chopper drive to regulate the voltage and current supplied to the motor. Higher input voltages to the driver allow the current supplied to the motor to reach steady state faster, resulting in higher step rates. In an attempt to meet the project's speed requirements, it was estimated that a 30 V supply would be sufficient. The twelve step drivers in total will draw 12 A peak current, but because of the chopper drive usually stabilizes around 6 A while at stall and lower currents while stepping. A power supply was selected capable of meeting these needs. Stepper motors are typically controlled with open loop algorithms and require only endstops to home their initial position. In testing of the real manipulator, at averaged operating speeds it was found that the belts on the mechanism slip before the steppers do, so no benefits could be gained by using closed loop control (e.g. encoders) on the stepper output shafts (though closed loop control for positioning of the entire manipulator could be beneficial, see section 10.6).

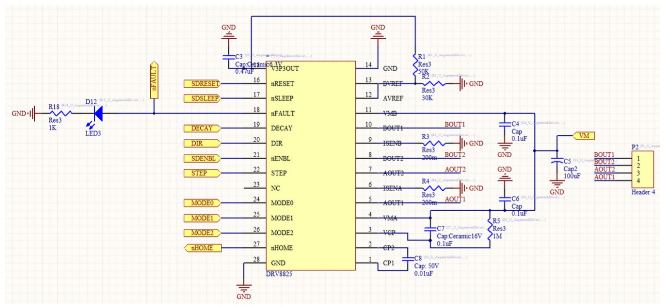
6.4.3 PCB Design

Altium was selected for designing the PCB as it commonly used in commercial environments and has good support for mutli-channel designs, a feature useful in this project since the the stepper driver circuit will be reused multiple times with minor variations in signal bus connections. Initially, the PCB was designed to use entirely surface mount (SMD) components which have the benefit of taking up less space on the completed PCB. Using surface mount motor driver ICs is also beneficial because it means the PCB itself can be used as a heat sink, helping to move extra heat away from the drivers. This design was ultimately abandoned for a number of reasons. First, due to the large amount of power flowing through the board, the PCB is expected to heat up by a significant amount and thus probably would not be particularly helpful in moving heat away from the the motor drivers, possibly having the opposite of making them warmer or risking overheating the PCB itself. Additionally, motor driver ICs require a large amount of supporting components creating increased complexity both in the design of the board and in its assembly as well as possible points of failure. While surface-mount components are cheaper and easier to assemble robotically for large board quantities, through hole-components are simpler for one-off hand assembled boards like those used for this manipulator. For these reasons, all ICs used on the board including the motor drivers used the standard DIP (Dual In-line Package) instead of an SMD version. Such ICs can also be easily replaced should they fail and are easy to prototype on a breadboard.

The motor driver IC selected for this project was the DRV8825 created by Texas Instruments. It can support supply voltages between 8.2 and 45 V and can handle currents up to 2.2 A. Making it perfect to control the steppers selected for this project. It also has built in over-current and over-temperature shutdown with under-voltage lockout and offers several microstepping resolutions which could be used to achieve smoother operation at the cost of motor output torque. Pololu offers a breakout board for this IC [19]. This is a well documented driver board which places all the features offered by the Texas Instruments driver IC into a modular package for simple step control. It includes a potentiometer for easily tuning of the maximum stepper current to take best advantage of the stepper motor's available power without causing damage and is built on a four-layer PCB with extra copper for improved heat dissipation. Built in voltage regulators means no additional logic supply is needed. Figure 21a shows the completed board schematic for using this component. One can also compare this to an older version of the schematic shown in Figure 21b which uses the surface-mount IC instead of the Pololu breakout board. The old version has a larger amount of components and the driver cannot be removed and replaced if needed as easily as it can in the new version.



(a) The Final Version



(b) A Previous Version

Figure 21: Scehmatics of the Motor Driver Circuit

The schematic in Figure 21a shows two rows of headers where the stepper driver is inserted. The stepper driver outputs, A0 through B1, are connected to a header where the motors will be plugged in. Power for the driver is drawn from a dedicated high power net (VM) and a 100μF capacitor provides protection against current spikes. The driver fault pin is connected to an LED to which will turn off if the stepper driver overheats or draws more current than is safe. This driver fault protection pin will later connect to logic used for detecting motor driver errors and eliciting a response by the control software. The control pins on the left are all connected off-schematic. The SDENBL, SDRESET, and SDSLEEP pins are all used to place the driver in various states of operation and are connected directly to the microcontroller. The MODE pins are used for selecting the microstepping mode. These will be connected to a set of header pins to be easily configured by moving a a few jumpers. The stepper motor steps every time it detects a rising edge on the STEP input pin. It moves in the direction corresponding to the signal on the DIR pin. These pins are connected to a network of shift registers discussed later.

The next step after creating the schematic for a single driver was to create the logic to send the step commands to the drivers from the microcontroller. Each driver requires at least two control pins for transmitting digital step and direction pulses as well as additional pins for enabling or disabling the driver and detecting faults. With twelve drivers plus the additional needed peripherals such as endstops, servos, and communication, a standard microcontroller would not have enough pins for controlling all the devices. Additionally, since a microcontroller can only toggle a single GPIO (General Purpose Input Output) pin at any instant in time, their would be a very minor time difference between each stepper receiving its commands, possibly creating some synchronization issues. To solve these problem, rather than connecting the control pins of all twelve stepper drivers directly to the microcontroller, the pins are attached to a the latched output pins from a set of shift registers. One data pin is attached from the shift registers to the microcontroller. On each clock cycle, the microcontroller places a control signal for a single stepper driver onto the data pin. This control signal is then shifted down the row of output pins as new control signals are shifted in until every signal is aligned with the appropriate stepper driver. At that point, the microcontroller toggles the latch pin, moving all the step commands simultaneously into the stepper drivers and causing the desired stepping motion to occur. In this way, a small number of pins can be used to control a very large number of stepper motors. Rather than connecting the input to the shift registers to a GPIO pin on the microcontroller, for faster operation the input is connected to the SPI (Serial Peripheral Interface) bus of the microcontroller. The software on the microcontroller can then place all the step commands into the data register for the SPI bus and have all of them sent to the drivers at speeds up to 384 KHz, while it is working on other tasks, such as computing the next set of step instructions or handling communications from the host computer[5]. The schematic in Figure 22 shows how this system is wired up. In the schematic, one can also see how we are taking advantage of Altium's mutlichannel support to repeat our previously designed stepper driver schematic twelve times and automatically wire all the drivers to the control buses.

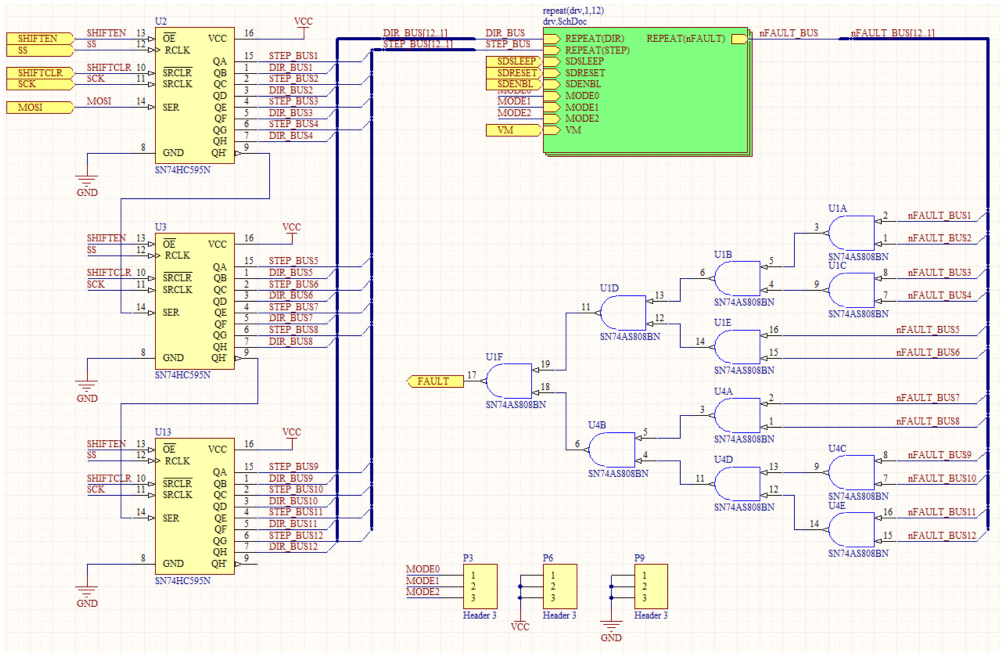
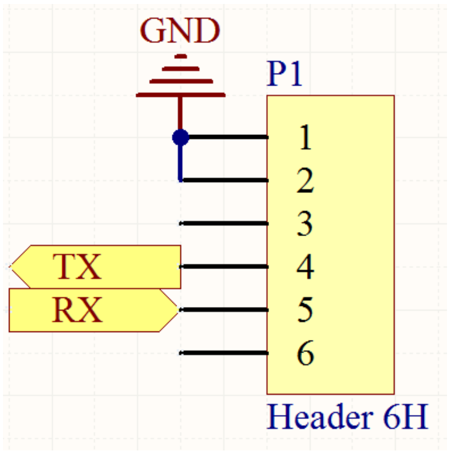


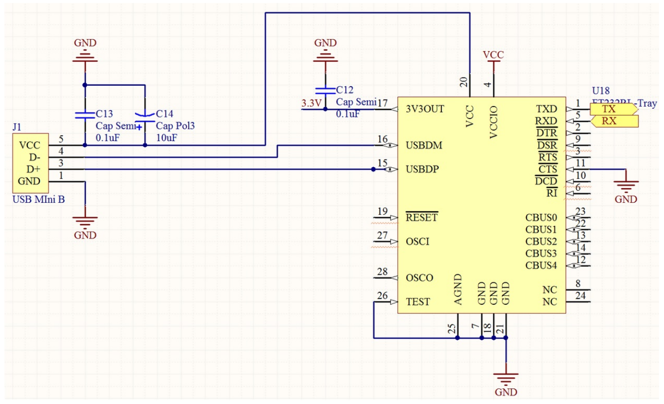
Figure 22: Completed Motor Control Circuit

The Pololu stepper driver breakout boards automatically cut power to any stepper driver which reports an overheat or over-current condition. While this built in protection is a wonderful feature, it is also desirable to alert the software system to such faults. Since the built in protection will handle the faulure of a single stepper, its not necessary to detect in software which stepper driver is the source of the fault, thus we can save a large number of GPIO pins on the microcontroller by combing the fault signals from all the stepper drivers together using a series of AND gates. Should any of the stepper drivers fail, the resulting signal from this sequence of AND gates will be a low signal, triggering an alert in software on the microcontroller which will later be sent to the host computer so that the appropriate action can be taking. The LEDs connected to the fault pins on the individual steppers can allow a human to easily examine the exact point of failure to take the appropriate action. The schematic also shows the header pins used for selecting the microstepping mode.

With the circuity for the stepper drivers designed, focus shifted to the other components. The board required a way to communicate with the host computer. This can be accomplished easily through the use of a microcontroller's on board USART (Universal Synchronous Asynchronous Receiver Transmitter) which is a simple communication interface which requires only two wires for data (one for transmitting and one for receiving) as well as a common ground signal. To use the USART with the host computer requires some way to convert these signals to those more commonly found on an average PC such as USB. The common solution to this is to use an FTDI (Future-Tech Devices Inc.) chip. Wiring the chip to the board only requires connecting the RX and TX signal wires as well as the sharing the common ground. It is also possible to use the breakout board as a source of 5V power for logic. The schematic for these connections is very basic and just includes a header pin for the connection, Figure 23a. In an earlier design, rather than using a breakout board for this operation, a surface mount FTDI chip to be used along with the necessary augmentation as shown in Figure 23b. This design was abandoned for the simpler one for all of the same reasons mentioned for swapping out the motor driver ICs for breakout boards.



(a) Current Version



(b) Prior Version

Figure 23: Communication Schematic

The microcontroller controls all the devices on the embedded board. It receives serial commands from the host computer, processes them, sends control signals to the stepper drivers and servo motors, and performs all other operations required for the decontrolling the low level features of the manipulator. The schematic for the microcontroller chosen is shown in Figure 24. An ATMega1284 microcontroller was selected for its feature set including enough GPIO pins for controlling all needed devices, the necessary peripherals (SPI and USART), enough available flash for storing the program and RAM for storing in use data and communication buffers, and can operate at the speeds needed for controlling the stepper drivers at the needed speeds. As with the other chips on the embedded board, a DIP package was selected for this chip instead of a surface mount package for easy replacement should any damage occur. All of the GPIO pins are broken out and attached to connectors. This allows for easy access to all signals manipulated or received by the microcontroller for debugging as well as connections for offboard components like end stops and limit switches. The ATmega1284 microcontroller has a built in 8MHz oscillator. This internal oscillator is very energy efficient but also inaccurate and slow. To better meet the needs of this project, an external 18.432 MHz crystal oscillator was attached to the microcontroller. This is the fastest frequency which can be used with the ATmega1284 which is also divisible by all the common communication baud rates. An accurate clock divisible by the common communication baud rates is necessary for ensuring low error rates in data transmission to and for the host computer. The high speed of the clock ensures that step commands can also be processed and sent at a high speed, ensuring that the microprocessor is not limiting factor for manipulator's maximum speed.

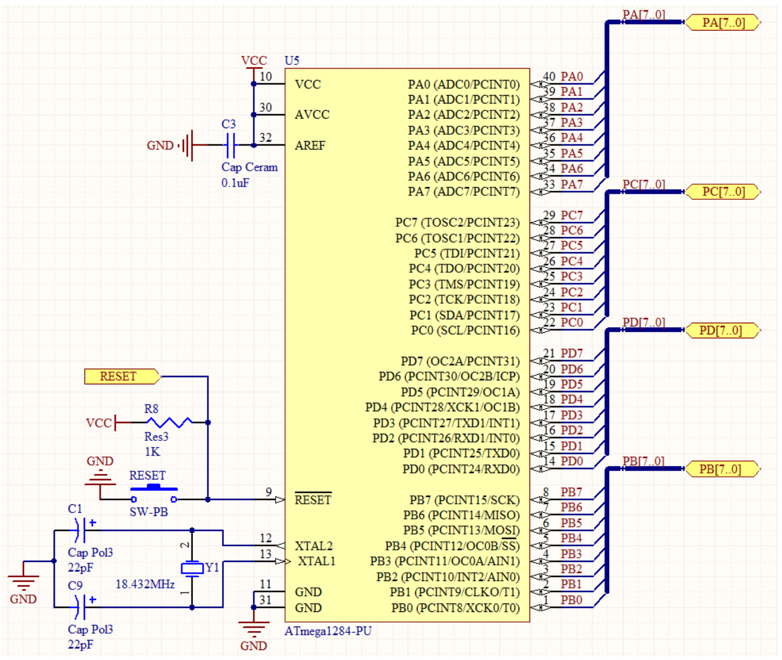


Figure 24: Microcontroller Schematic

The next schematic shown in Figure 25 shows all the off-board connections for things such as power, programming, servos, and endstops. All the GPIO pins for the microcontroller are broken out into header pins where the limit switches and servos are connected. A six pin connector connects to the in-system programming port on the microcontroller where an Atmel AVRISPmkII (or similar) programmer can be connected to upload code to the board. The power inputs for both the stepper motors and logic are protected against current spikes with large capacitors.

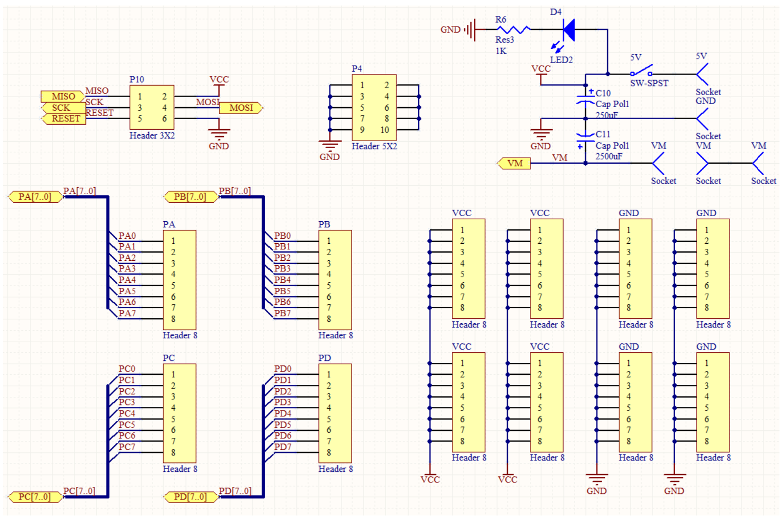


Figure 25: Schematic Showing Off-board Connectors

Finally, Figure 26 shows the top level design of the board. On this schematic one can see how all the previous schematics for the microcontroller, connectors, stepper drivers, and communication connect together.

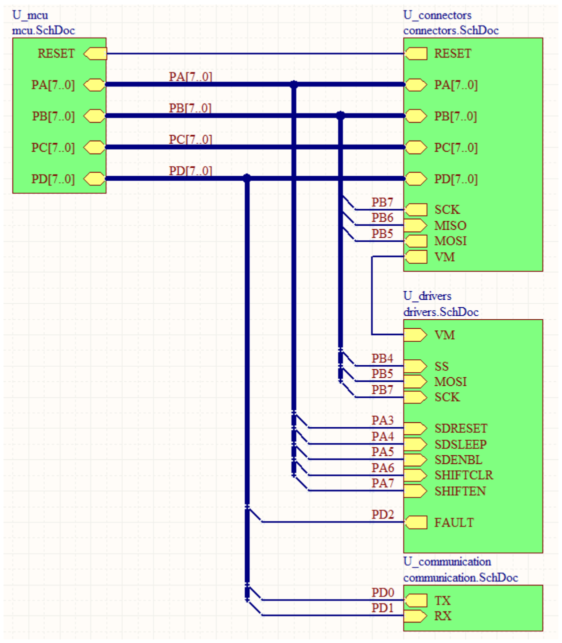


Figure 26: Top Level PCB Design Schematic

Like the schematic, the board layout went through a number of design iterations. The largest problem of creating the board layout was finding an optimal component placement that would allow easy access to off board connections and user interface elements, as well as being physically routable with no short circuits or broken connections while minimizing board size to save on costs. The first full PCB route can be seen in Figure 27 and another earlier version of the PCB can be seen in figure 28. Both these version were before the switch from SMD components to through-hole components. The final PCB layout can be seen in Figure 29.

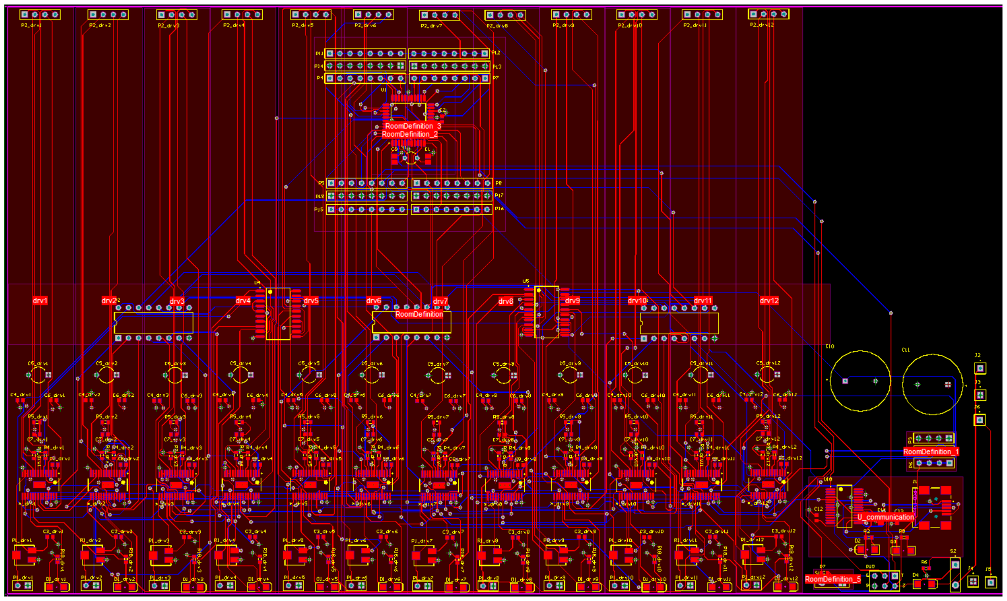


Figure 27: The First Full Routing of the PCB

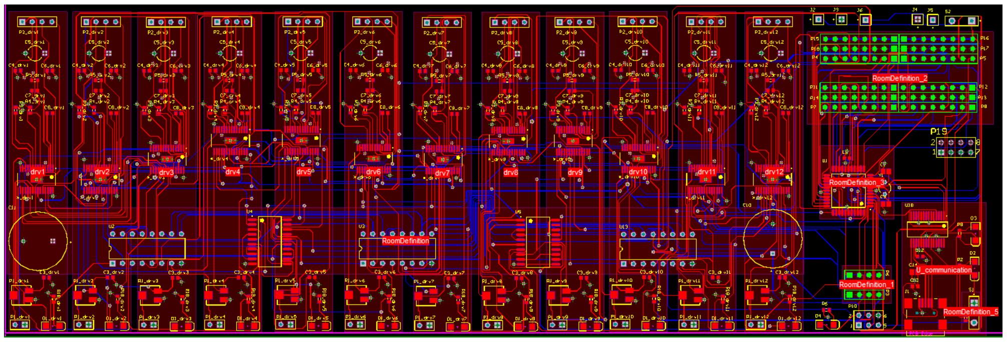


Figure 28: Another Early Routing During the Design Process

Due to the large power requirements of the motor drivers, the PCB was designed to have four layers. The top and bottom layers were used for routing signals and logic level power while the middle two layers were dedicated power and ground planes to handle the incoming 360 Watts (12A @ 30V) needed by the stepper motors. The traces carrying power from the individual stepper drivers to the stepper motors were also made wide enough to handle the motor current draw in accordance with IPC-2221 PCB standards [12].

The board was laid out to be convenient to use with the manipulator. All the motor attachment points are placed along the top of the board and all the indicators were placed along the bottom. The stepper drivers were placed to be as close to the corresponding motor connector as possible and the fault logic and shift registers were placed to be near the stepper drivers. Connections for limit switches, power, and servos were also placed at the top of the board for easy connection to the manipulator. Serial communication and the programming port were placed at the bottom right of the board for easy attachment to the host computer. Board layout is where Altium's multichannel feature becomes extremely useful, as it was possible to make changes to the layout of one of the drivers, and have the change repeated in the other driver blocks, making it easy to rearrange the components to achieve optimal routing of the traces and ensuring that everything is spaced evenly over the board. With the components of the board laid out, Altium's autoroute was used to generate the traces. It's possible to manually draw the traces, but with this project autorouting worked just fine, though some minor cleaning and design rule changes were required.

After routing the board, some silkscreen was added to the top and bottom of the board to label the connectors and add other useful information. The final board dimensions are 185 mm wide by 90 mm tall.

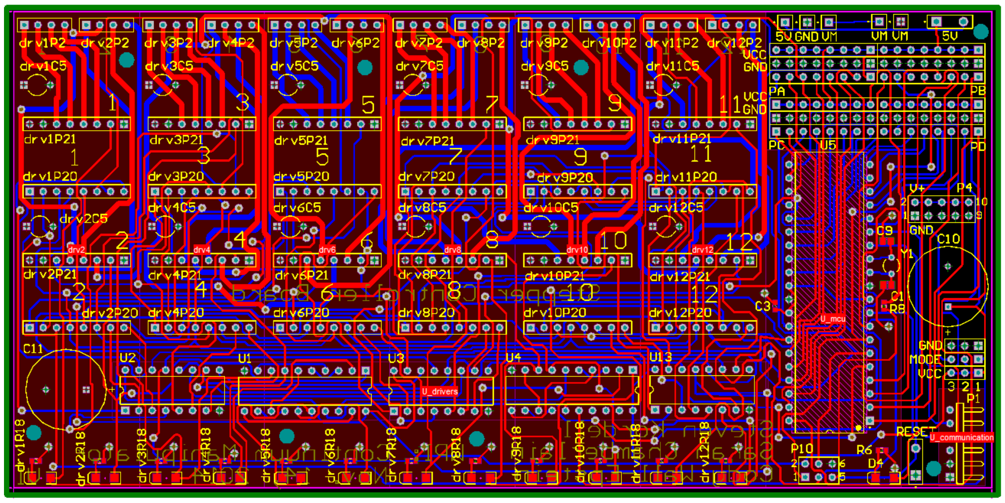


Figure 29: The Final PCB Layout

6.5 Software

6.5.1 General Concepts

The software for this project is broken into two main components. There is the embedded software written in C for use on the custom control board and their is the high level control software written in C++ used on the host computer. MATLAB was also used while developing the kinematic model. Git was used throughout the software development process for version control and to enable easy collaboration among team members. Some code snippets are show throughout this section to highlight important areas. Full source is available from the github repositories located at `https://github.com/Spkordell/coma_embedded` and `https://github.com/Spkordell/coma_ros`. Github automatically generates a few graphs for visualizing development progress over time which are include in Figures 30 through 33 for those interested.

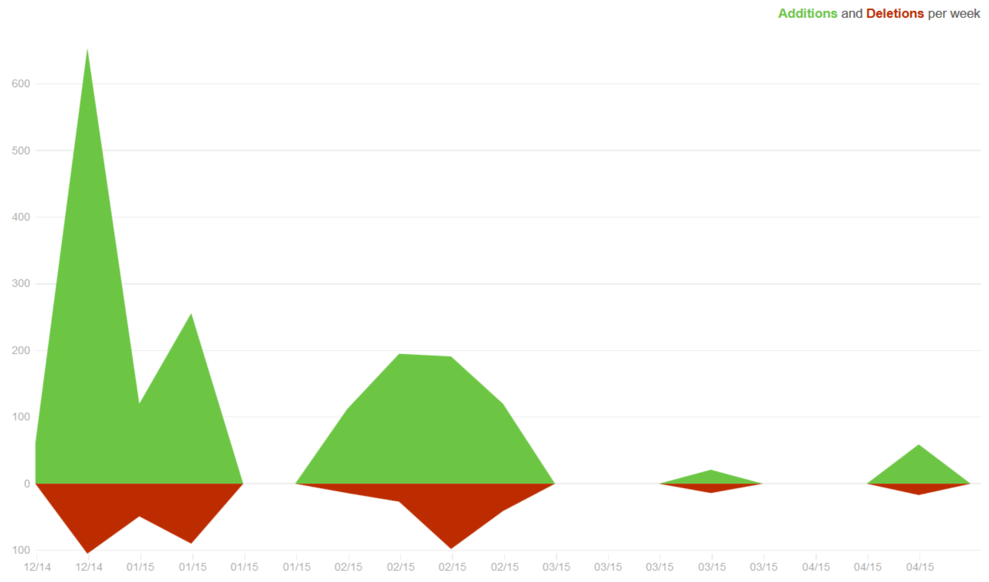


Figure 30: Code Frequency for coma_embedded

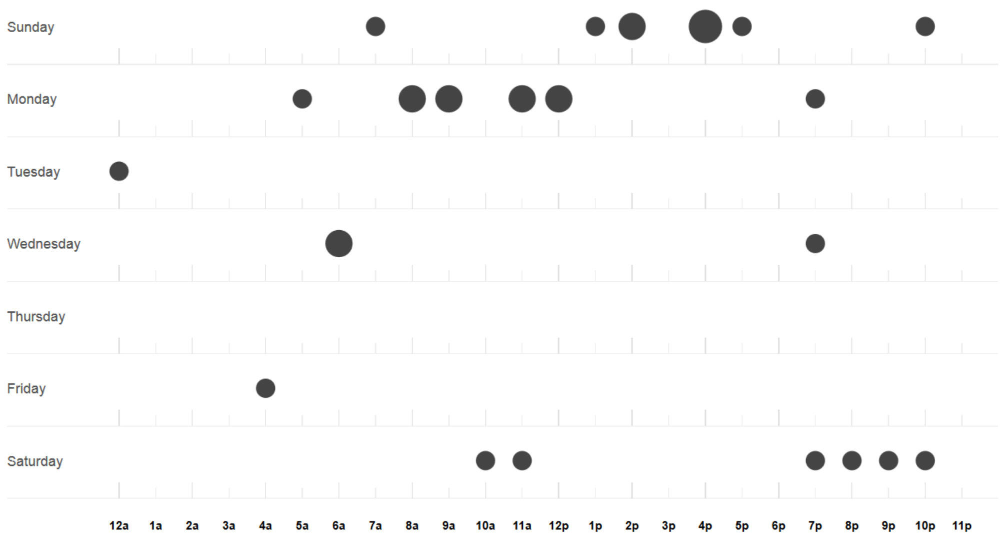


Figure 31: Punch Card for coma_embedded

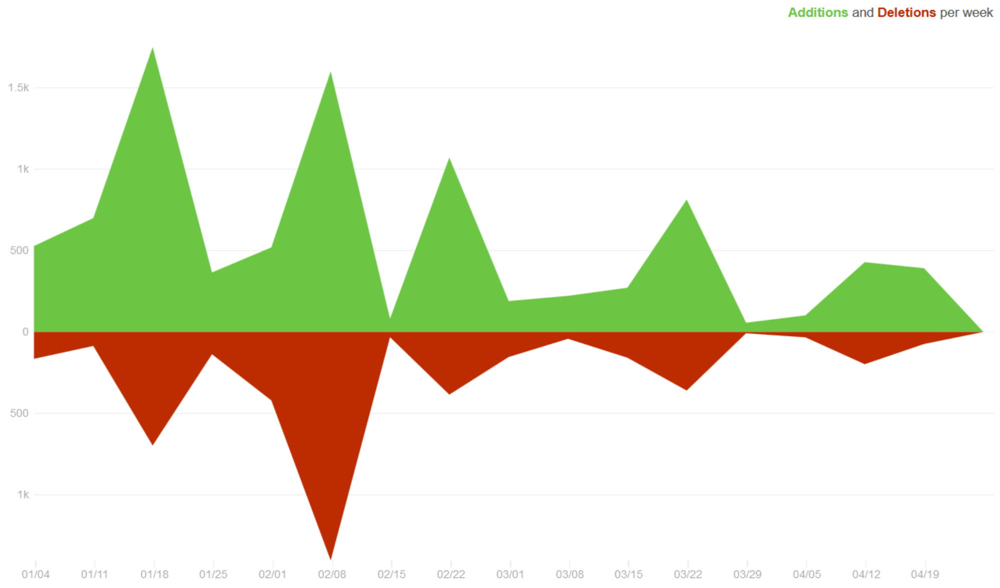


Figure 32: Code Frequency for coma_ros

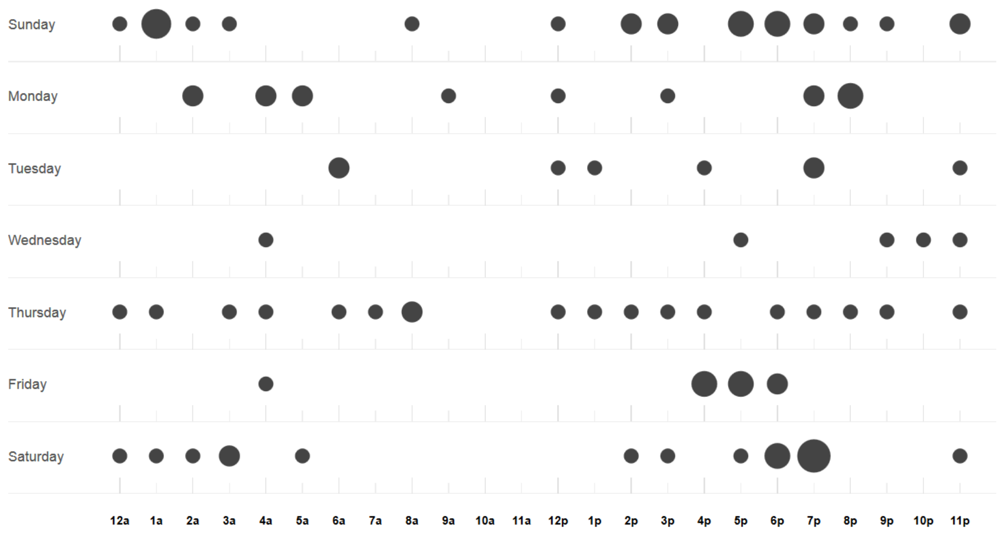


Figure 33: Punch Card for coma_ros

6.5.2 Embedded Software

Atmel's AVR studio was used to developed the embedded software as it was created by the manufacturing of the microcontroller and has the best supported feature set. Once the PCB was constructed, some simple test programs were created to ensure code could be uploaded to the microcontroller successfully. Upon verifying this, the first step step in embedded software development was to get serial communication running so the board could talk to the host computer. Not only is this required for receiving motion commands, but it also makes it significantly easier to debug other code issues as the program can print status information to a terminal running on the host computer.

The USART was initialized in asynchronous UART 8-bit mode with 1 stop bit, no parity, Tx and Rx interrupts enabled, and a baud rate of 115200 (The fastest common baud rate). Two FIFO (first in, first out) buffers were created for storing incoming and outgoing serial information. FIFO buffers are used as they allow asynchronous access to the data held within the buffer, preventing race conditions when accessing the data from different parts of the program, particularly when the interrupt service routines are called. The interrupt service routines for handling incoming and outgoing messages were created as were a number of helper functions for to aid in communication. The helper functions performed operations such as adding characters or strings to the outgoing buffer or reading data from the incoming buffer. The interrupt service routines for transmitting and receiving data over the UART are shown below. Once the incoming data from the host computer has been saved into the UART buffers, parsing it to extract the desired stepper positions is trivial.

```
/**
 * \brief UART data register empty interrupt handler
 *
 * This handler is called each time the UART data register is available for
 * sending data.
 */
ISR(UART0_DATA_EMPTY_IRQ) {
   char data;
   //if there is data to send, fetch it and send it
   if (fifo_get(&out_buffer, &data)) {
      UDR0 = data;
   } else {
      // no more data to send, turn off the data ready interrupt
      UCSR0B &= ~(1 << UDRIE0);
   }
}

/**
 * \brief Data RX interrupt handler
 *
 * This is the handler for receiving UART data.
 */
ISR(UART0_RX_IRQ) {
   fifo_put(&in_buffer, UDR0);
}
```

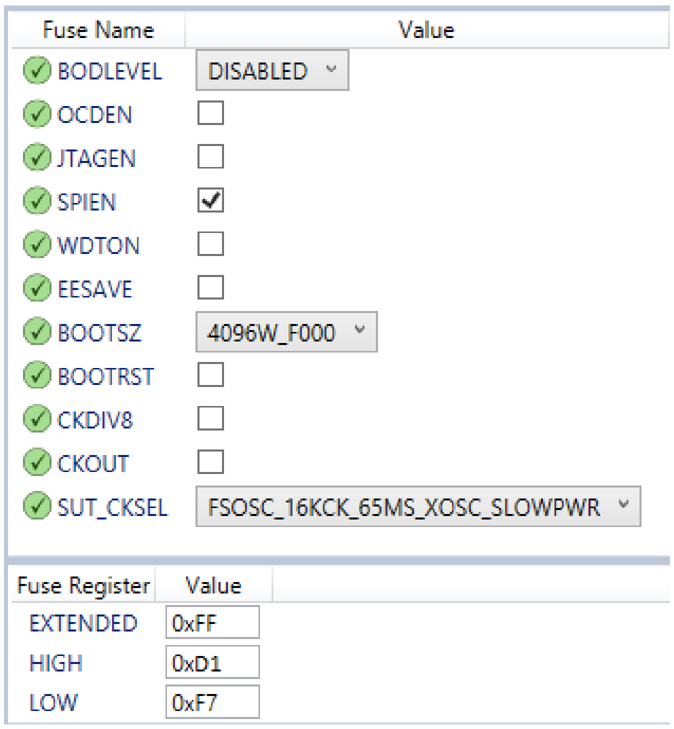
Once serial communication was running, initialization code for the other needed peripherals was written. This includes the SPI interface for sending commands to the stepper motors, the GPIO pins for for reading the state of the limit switches, and the timers for use in servo control. The initialization routine for the steppers is shown below.

```
void init_steppers(void) {
  DDRB |= (DATA | LATCH | CLOCK);  //Set shift control pins as outputs
  PORTB &= ~(DATA | LATCH | CLOCK); //Set shift control pins low
  DDRA |= (STEPPER_RESET | STEPPER_SLEEP | STEPPER_ENABLE | SHIFT_ENABLE |
      SHIFT_CLEAR);  //set pins as output
  PORTA |= (STEPPER_RESET | STEPPER_SLEEP | SHIFT_CLEAR);               //set pins high
  PORTA &= ~(STEPPER_ENABLE | SHIFT_ENABLE);                            //set pins low
  DDRD &= ~(STEPPER_FAULT);                                                 //set
      fault pin as input

  init_SPI();

  //initialize stepper counts to 0
  for (unsigned int i = 0; i < STEPPER_COUNT; i++) {
    currentStepperCounts[i] = 0;
  }
}
```

With the peripherals configured, it was time to attempt to modify the fuse settings on the microcontroller to attempt to use the external crystal oscillator. This can be a dangerous procedure as failing to set the fuse bits properly can permanently “brick” the microcontroller. The proper fuse settings are shown in Figure 34. While being faster (18.432 MHz vs 8 MHz), the external oscillator is also evenly divisible by common communication frequencies and is better at keeping synchronized time then the internal oscillator, resulting in more reliable communication. In additional to changing the selected oscillator, it was also necessary to disable the JTAG interface. JTAG is a debugging tool not being used in this project which uses a few of the microcontroller’s pins needed for other purposes.



Fuse Name	Value
✓ BODLEVEL	DISABLED
✓ OCDEN	☐
✓ JTAGEN	☐
✓ SPIEN	☑
✓ WDTON	☐
✓ EESAVE	☐
✓ BOOTSZ	4096W_F000
✓ BOOTRST	☐
✓ CKDIV8	☐
✓ CKOUT	☐
✓ SUT_CKSEL	FSOSC_16KCK_65MS_XOSC_SLOWPWR

Fuse Register	Value
EXTENDED	0xFF
HIGH	0xD1
LOW	0xF7

Figure 34: Microcontroller Fuse Configuration

With all devices on the board working, it was time to move on to sending step commands to the motors. This portion of code can be a little complicated due to the methods used to control the stepper drivers, so it warrants some additional explanation. The send step function receives an array of desired stepper positions. The function iterates over the values in this array and compares them with the values in a global array containing the current step positions. When a mismatch is found, the microcontroller prepares to send a step signal to update all the steppers until the current step positions match the target step positions. This is done by first clearing any data currently contained in the shift registers, then toggling the latch pin on the shift registers so data can be shifted in without effecting the output until all the data has been moved into the shift registers. Several bytes are constructed to specify which steppers need to step and what direction they should step. These bytes are transmitted over SPI into the shift registers. While SPI is transferring one byte, subsequent bytes can be computed. When all the data has been moved into the shift register, the latch pin is toggled again, causing all the step and direction commands to be simultaneously sent to the motor drivers which perform the steps. The cycle then repeats until the motor positions match the desired positions.

```
void send_step(unsigned long* stepperTargets) {
   unsigned char spi_buffer;
   for (unsigned int stepper = 0; stepper < STEPPER_COUNT; stepper++) {
      while (currentStepperCounts[stepper] != stepperTargets[stepper]) {
         //clear shift register
         PORTA &= ~SHIFT_CLEAR;   //write low
         PORTA |= SHIFT_CLEAR;    //write high
         //Toggle latch to copy data to the storage register
         PORTB |= LATCH;
         PORTB &= ~LATCH;

         SPDR = ((unsigned char) /*compute and transmit first instruction */
         ( ( currentStepperCounts[11] > stepperTargets[11]) << 7) |
             ((currentStepperCounts[11] != stepperTargets[11]) << 6)
         | ((currentStepperCounts[10] < stepperTargets[10]) << 5) |
             ((currentStepperCounts[10] != stepperTargets[10]) << 4)
         | ((currentStepperCounts[9] > stepperTargets[9]) << 3) |
             ((currentStepperCounts[9] != stepperTargets[9]) << 2)
         | ((currentStepperCounts[8] < stepperTargets[8]) << 1) |
             ((currentStepperCounts[8] != stepperTargets[8]) << 0));
         spi_buffer = ((unsigned char) /*while waiting for first transfer to finish,
             compute second instruction*/
         ( ( currentStepperCounts[7] > stepperTargets[7]) << 7) |
             ((currentStepperCounts[7] != stepperTargets[7]) << 6)
         | ((currentStepperCounts[6] < stepperTargets[6]) << 5) |
             ((currentStepperCounts[6] != stepperTargets[6]) << 4)
         | ((currentStepperCounts[5] > stepperTargets[5]) << 3) |
             ((currentStepperCounts[5] != stepperTargets[5]) << 2)
         | ((currentStepperCounts[4] < stepperTargets[4]) << 1) |
             ((currentStepperCounts[4] != stepperTargets[4]) << 0));
         while(!(SPSR & (1<<SPIF))); //Wait for first SPI transfer to finish
         SPDR = spi_buffer; /*send second instruction*/
         spi_buffer = ((unsigned char) /*while waiting for second transfer to finish,
             compute third instruction*/
```

```
        ( ( currentStepperCounts[3] > stepperTargets[3]) << 7) |
            ((currentStepperCounts[3] != stepperTargets[3]) << 6)
        | ((currentStepperCounts[2] < stepperTargets[2]) << 5) |
            ((currentStepperCounts[2] != stepperTargets[2]) << 4)
        | ((currentStepperCounts[1] > stepperTargets[1]) << 3) |
            ((currentStepperCounts[1] != stepperTargets[1]) << 2)
        | ((currentStepperCounts[0] < stepperTargets[0]) << 1) |
            ((currentStepperCounts[0] != stepperTargets[0]) << 0));
        while(!(SPSR & (1<<SPIF))); //Wait for second SPI transfer to finish
        SPDR = spi_buffer; //send third instruction
        while(!(SPSR & (1<<SPIF))); //Wait for third SPI transfer to finish

        //Toggle latch to copy data to the storage register
        PORTB |= LATCH;
        PORTB &= ~LATCH;

        //update current stepper counts
        for (unsigned int i = stepper; i < STEPPER_COUNT; i++) {
           currentStepperCounts[i] += (currentStepperCounts[i] != stepperTargets[i]) *
               (currentStepperCounts[i] < stepperTargets[i] ? 1 : -1);
        }

        _delay_ms(STEPPER_DELAY);

      }
   }
}
```

The homing routine uses code similar to that used for sending step commands, except instead of sending step signals until a desired target position is met, step signals are sent until the limit switch corresponding to the stepper is pressed. To avoid collisions in the mechanism, the bottom link is homed first, followed by the top link.

Servos for the end-effector are controlled using the ATmegas's timers in phase-correct PWM mode. In this mode, one can compute various trigger values on a step-wise waveform which can automatically trigger the toggling of an output pin at a desired frequency and duty cycle. In this manner, one can “set-and-forget” the desired position of a servo and not have to about tending to any interrupts or other processor consuming tasks as the control is delegated entirely to the hardware. The code for initializing the servo and updating the duty cycle to move the servo to a desired position is shown below.

```
void init_servos() {
  //set OC0A for gripper servo to output
  DDRB |= (GRIPPER_SERVO_PIN);
  //set OC1A/B for rotation/flex servos to output
  DDRD |= (WRIST_ROTATE_SERVO_PIN | WRIST_FLEX_SERVO_PIN);

  //initialize timer 0 in phase-correct PWM mode for the gripper
  //Initialize timer count to 0
  TCNT0 = 0;
  //configure timer 0 for phase correct PWM mode, prescaler = clk/1024
  TCCR0A = (1 << COM0A1) | (1 << WGM00); //enable non-inverting PWM output on pin OC0A
      (not using OC0B)
  TCCR0B = (1 << CS02) | (1 << CS00); //prescaler = clk/1024
  OCR0A = 13; //set the duty cycle

  //initialize timer 1 in phase-correct PWM mode for the wrist rotation/flex
  //initialize timer count to 0
  TCNT1 = 0;
  //configure timer 1 for phase correct PWM mode, prescale = 8
  TCCR1A = (1 << COM1A1) | (1 << COM1B1) | (1 << WGM11); //use ICR1 as top and
      non-inverting PWM output
  TCCR1B = (1 << WGM13) | (1 << CS11); //phase-correct, prescale = 8
  TCCR1C = 0;
  //set top comparison value // period is 20 ms with a 18432000 hz clock and prescale
      = clk/8
  ICR1 = 23040; // ((0.02 * F_CPU)/8)/2 //division by 8 because of prescale, division
```

```
        by 2 because of phase correct mode

    //set servos to 0 position by setting duty cycle to 5% (1 ms pulse) (max servo range
        occurs at 12.5% duty (2.5 ms pulse)
    OCR1A = ICR1*0.05;
    OCR1B = ICR1*0.05;
}

void set_servo(int servo, double angle){
    static char TOTOP = 180;
    switch (servo) {
        case GRIPPER_SERVO:
            OCR1B = (uint16_t)(ICR1 * (0.075*(angle/180)+0.03)); //compute the duty cycle
            break;
        case WRIST_FLEX_SERVO:
            OCR1A = (uint16_t)(ICR1 * (0.075*(angle/180)+0.03)); //compute the duty cycle
            break;
        case WRIST_ROTATE_SERVO:
            OCR0A = (uint8_t)(TOTOP * (0.075*(angle/180)+0.05)); //compute the duty cycle
            break;
    }
}
```

6.5.3 Host-Computer Software

The high level control software is written in C++ and uses the ROS (Robot Operating System) framework. This framework provides a standardized method for organizing code into different nodes encapsulated within various packages. The nodes can pass information between each other using a common messaging system. ROS's framework makes it easy to reuse existing code reducing development time as commonly used features do not have to be implemented from scratch.

This project split the high-level code into eight ROS packages: coma_ros, coma_bringup, coma_serial, coma_teleop, coma_demo, coma_kinematics, coma_rviz, and coma_simple_planner.

6.5.3.1 coma_ros The coma_ros package is a "metapackage", a type of specialized ROS package that does not contain code or other files typically found in other ROS packages. Instead, a metapackage references related packages and loosely groups them together so they can be thought of as a single unit with shared dependencies. The coma_ros metapackage contains references to the other seven ROS packages used for this project.

6.5.3.2 coma_bringup The coma_bringup packages contains eleven different launch files used for starting the system in different ways by launching and configuring the other ROS nodes. The launch files and their associated purposes are shown in Table 7.

Launch File	**Purpose**
demo	2 Launches the demo node used for testing and demonstration
teleop_full	Launches all nodes necessary for controlling the real manipulator with an attached joystick
teleop_full_path	Same as teleop_full, except the IK solver will interpolate between control points to build smoother paths
teleop_full_rviz	Same as teleop_full except will also graphically show IK solutions in rviz (Robot Visualizer)
teleop_full_rviz_path	Same as teleop_full_path except will also graphically show IK solutions in rviz
teleop_full_fake_ik	Launches nodes to control the real manipulator using an attached joystick but does not use the full kinematic solver
teleop_ik	Launches teleoperation nodes and IK solving nodes in simulation (does not control real arm)
teleop_ik_path	Same as teleop_ik except interpolates points between desired positions
teleop_ik_rviz	Same as teleop_ik with the addition of rviz for displaying kinematic solutions
teleop_ik_rviz_path	Same as teleop_ik_path with the addition of rviz for displaying kinematic solutions
telop_fake_ik	Same as teleop_ik except does not use the full IK solver

Table 7: ROS Launch Files

6.5.3.3 coma_serial This package is responsible for handling all communication between the host computer and the embedded control board. It uses the boost ASIO libraries for opening the serial port and launches threads for handling communication over the port. It receives messages from other ROS nodes (such as the teleop or demo nodes) containing information about the desired state of the manipulator. It then transmits this information over the serial interface in a manner the embedded board can understand. It also alerts other nodes to incoming communication from the embedded board, such as when when the manipulator is ready to receive the next command.

6.5.3.4 coma_teleop This node listens to messages broadcast on the
joy topic by the joy_node (A standard ROS package for communicating with attached joystick devices). These messages contain information about the state of the attached controller, what buttons are pressed and what positions the joystick axis are in. These input commands are interpreted and sent to the kinematic solver then to the serial node for transmission to the manipulator. Changes to the control inputs can also be interpreted directly as desired changes to the manipulator and can be used to teleoperate the manipulator without running the inverse kinematics solver which can produce poor solutions if using the C++ implementation. This was the primary method used for testing the manipulator.

6.5.3.5 coma_demo This package broadcasts predetermined manipulator configurations to the serial node for easy testing, debugging, and demonstration.

6.5.3.6 coma_kinematics This package contains the C++ implementation of the inverse kinematic solver. It uses the Eigen libraries for performing matrix operations, the boost odeint library for performing numerical integrations along the length of the rods, the boost multithreading library for running solving tasks in parallel, and Google's Ceres solver for solving the system of equations representing the kinematics of the manipulator. Outside nodes can query the solver by making a service call containing a desired manipulator pose and receive a response containing the leg lengths necessary to achieve that pose.

6.5.3.7 coma_rviz When then kinematic solver finds a solution, it also publishes the the position results of the rod integrations. The node within the coma_rviz package takes these positions and plots them in rviz (A standard ROS visualization package) to show what the solution should look like graphically.

6.5.3.8 coma_simple_planner The simple planner receives a message containing a start and goal pose. It interpolates between these poses and sends points to the kinematic solver which solves for each positions. The node returns the list of leg lengths to the calling node which can then command the manipulator to follow the path. Quaternions are used in the interpolation of rotation to ensure smooth rotational paths are created rather than the seemingly chaotic paths produced by interpolating between Euler angles.

7 Manufacturing

7.1 Mechanical

This section goes over the manufacturing and assembly involved with the manipulator. This includes the base of the manipulator, the end-effector, and the electronics control board.

7.1.1 Base Manufacturing

All of the parts were designed for manufacturability and minimal stock waste. The CAM for all of these parts was done in ESPRIT CAM software. The various parts were designed to be done in no more than 3 operations each with standard tooling to make the manufacturing process go much more smoothly. The cycle time for the parts was optimized through the use of automated probing cycles and optimized feeds and speeds for the materials used. Although designed to be made more faster with few complicated parts, there were a few issues that arose and were promptly solved. Within the manufacturing process, some parts were more complicated to machine than others, not due to the complexity of the part, but the complexity of fixturing for the parts and the size of the parts.The final base (not including the end-effector) consisted of 102 machined parts. Some of the various parts included in the base are in Figure 35.

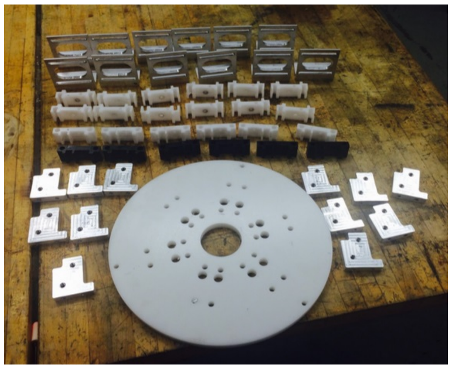


Figure 35: Manufactured Parts

One of the more difficult parts to machine was the second stage base plate that contained the motors that can be seen in Figure 36. As stated in the materials section, second stage base plate was 20 inch in diameter and made from 0.25 inch thick aluminum. The main challenge encountered with the second stage base plate was its size and fixturing.

In order to manufacture it, adjustments had to be made in the design. The plate was originally designed to be 24 inch in diameter. However, manufacturing the 24 inch diameter design proved to be impossible with the machines available, because the machine tool travel dimensions were 20 inches in y and 24 inches in x. This led to the need to redesign the plate both to accommodate the limitations of the machine tool and to fit all of the components on the plate within the limited space. Although the workpiece was now redesigned to fit within the y axis travel limit of the machine tool, a challenge remained in fixturing. The part had to be affixed in such a way that it did not crash the machine or allow the part to detach from the machine tool. This was resolved by affixing the working plate to a sacrificial plate which allowed an effective fixturing configuration.

The outside contour of the plate was machined in two operations; half of the plate was machined and then the plate was flipped and the other half was machined. This could be done because the features of the plate were symmetrical. In order to make sure that the plate was lined up properly when flipped to machine the other side, dowel pins were used to align the hole pattern.

Another feature on the plate that made it more challenging to machine was the configuration of 12 slots. Slots put more load on the tool during machining because the tool is fully engaged in the workpiece. The feeds and speeds needed to be adjusted accordingly for the loads on the tool. After the design changes, fixturing, and programming, this part was successfully completed.

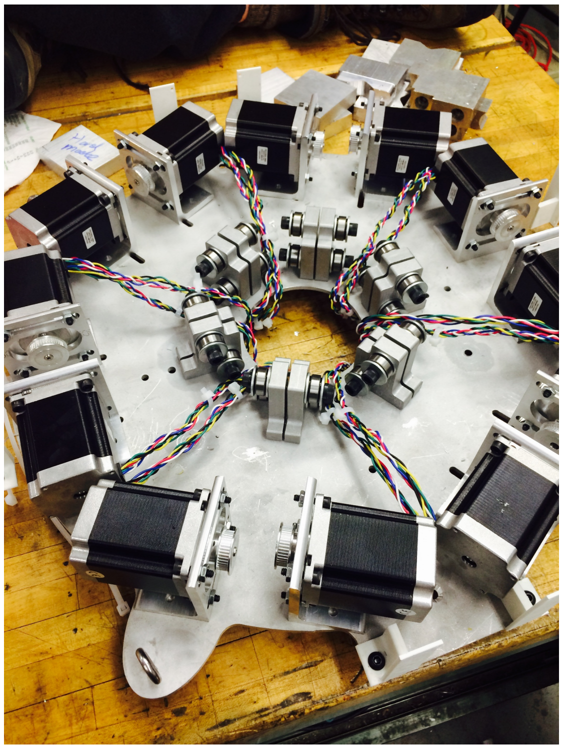


Figure 36: Actuator Plate Assembly

7.1.2 End-Effector Manufacturing and Material Selection

The end-effector design was based off of a gimbal and instead of machining the two brackets for the gimbal, off the shelf components were used because it was the fastest and most cost effective way of ensuring that the end-effector worked and was done in a timely manner. The bottom rotational component was attached to the end of the arm through a laser-cut adapter plate that utilized the fixturing holes from machining the top link. The adapter plate housed a servo that was then connected to a gear that meshed with another gear that rotated the base of the gimbal attachment. Parts were then made to attach the gripper fingers to the end of the gimbal. Two plates were fastened to the end of the gimbal and housed another servo for opening and closing the gripper. The servo was connected to an ABS 3D printed gear gripper attachment. When the servo moved to an extreme in one direction it would mesh with another gear gripper attachment part and would in turn open and close the gripper. The gripper fins described earlier were attached to the 3d printed gear components. The fins were 3D printed out of a flexible material called NinjaFlex allowing for the fish tail effect. The use of this material for 3d printing simplified its manufacturability because otherwise it would have to have been made using a mold of sorts.

7.1.3 Assembly

The parts manufactured in this project were made with assembly in mind and came together much more easily than the prototype. Although, the design was made to be assembled more easily it was still a time consuming process because many more components were added to the overall system compared to the prototype. The assembly now contained the motors, pulleys, and belts. The assembly of the base can be seen in Figure 37.

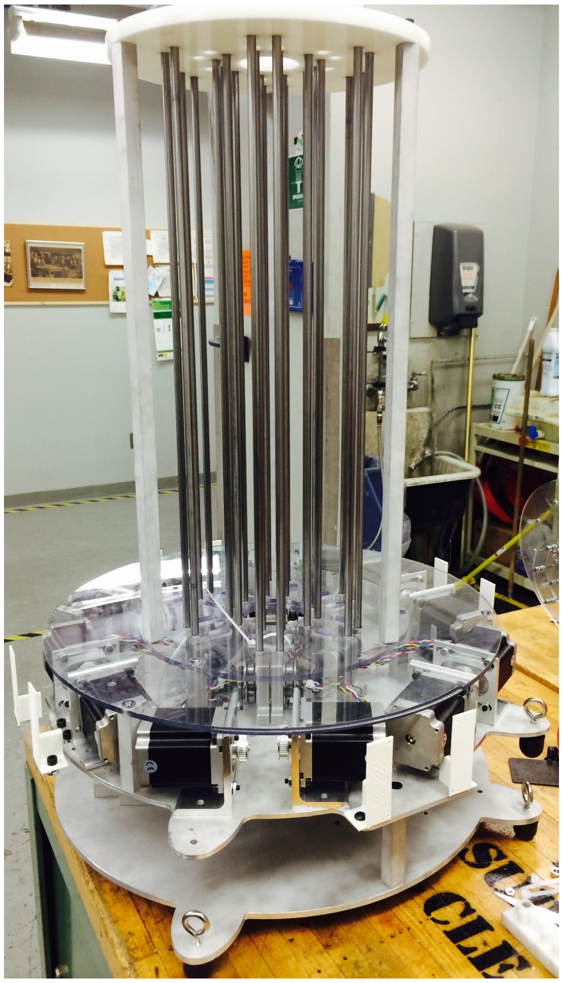


Figure 37: Manipulator Base Assembly

7.2 Electrical

After designing the PCB, the layout was sent to a fabrication house to create the board and the other electronic components were purchased from retailers such as DigiKey, Pololu, and Sparkfun. Advanced Circuits agreed to sponsor the project and fabricate the board free of charge. When the board and other electrical components arrived. they were first inspected by checking all the connections with a multimeter, then applying power to the board with none of the components installed to be absolutely certain there were no shorts. The front and back of the board before component installation can be seen in figures 38 and 39.

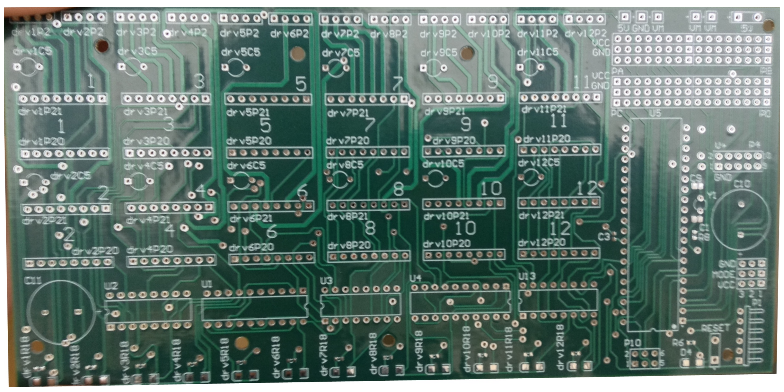


Figure 38: The Front of the Unpopulated PCB

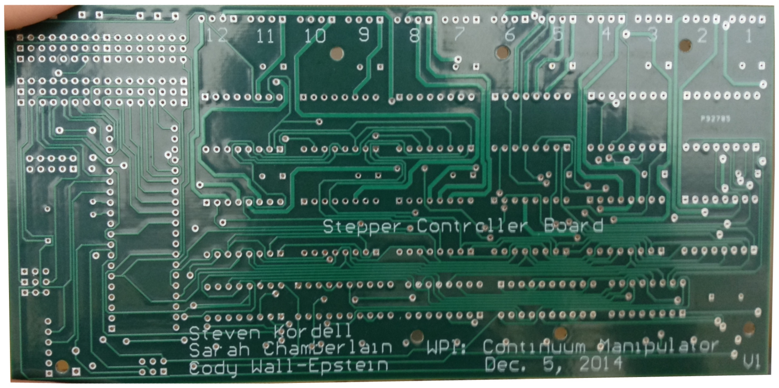


Figure 39: The Back of the Unpopulated PCB

Confident that the board arrived as designed, assembly could begin. Assembly usually begins by installing the smallest components first as these usually require more dexterity and are hard to place with larger components in the way. For this board, the smallest components are the SMD LED's used for indicating power and the state of the stepper drivers and the corresponding resistors. There are also a few small capacitors and resistors near where the microcontroller will be placed. Figure 40 shows the board with the LEDs installed and power applied to one of them to ensure it was soldered in correctly.

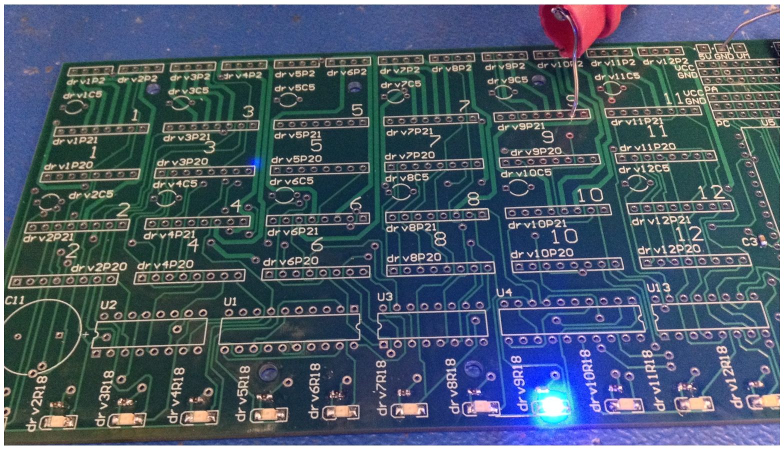


Figure 40: The PCB with LEDs Installed

With the small components installed, assembly continued with the installation of the sockets for the ICs, motor drivers, and connectors. Figure 41 shows the board with the sockets installed.

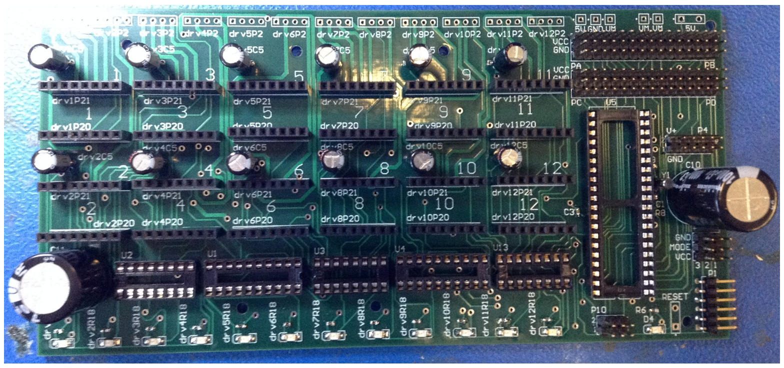


Figure 41: The PCB with Sockets Installed

With the sockets in place, all the ICs and motor drivers were installed, the motors were attached, and software development for the board as described in section 6.5.2 occurred. Images of the board with stepper drivers in place and motors connected are shown in figures 42 and 43.

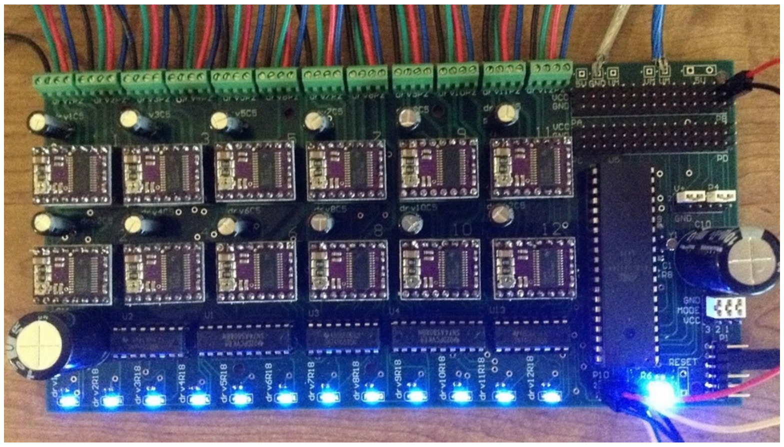


Figure 42: The PCB with Drivers Installed

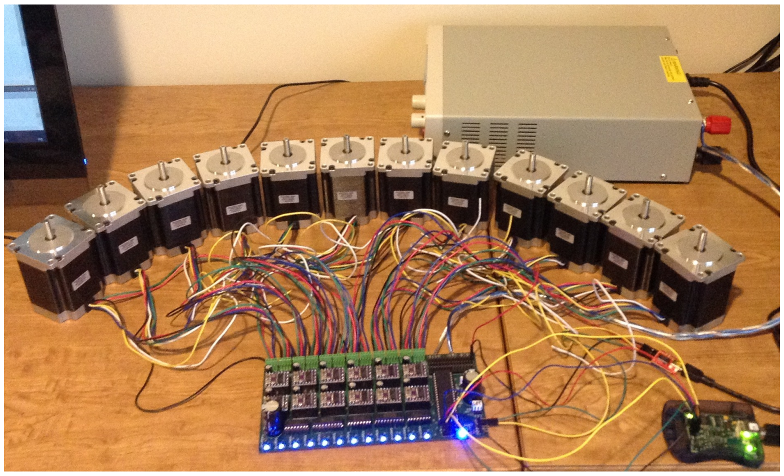


Figure 43: The PCB With Stepper Motors Attached

8 Testing and Results

8.1 Testing

The manipulator was put through a series of tests to check how well it met the stated project goals outlined in section 5.

8.1.1 Speed

The manipulator's max speed requirement was 400 mm/sec with no load. For this test, the manipulator's end-effector was removed as it is considered a load. The worse-case speed of the manipulator should occur during linear motions along the z-axis so this was where testing was focused. Software commanded the steppers to move the manipulator between its minimum and maximum height. Times for each rise and fall were recorded and divided by the displacement to compute the max speed at around 250 mm/sec. Disappointingly, this does not reach the projects speed goal. The electronics were designed to handle sending step commands at high rates and testing revealed this not to be the limiting factor in the manipulator's speed. The goal was most likely missed due to a failure to account for some friction and forces in the mechanical structure which prevent the stepper motors from completing a full step before the next occurs, causing steps to be missed. Higher voltages could be used to help achieve this goal as it would allow the current drawn by the stepper to reach steady state faster, resulting in a higher chance of a successful step. The motor drivers can handle voltages up to 45V but the selected power supply has a max voltage of 30V and the filtering capacitors on the embedded control board are only rated 30V as well.

8.1.2 Rotation

The rotation requirement for the manipulator required that it was capable of rotating perpendicular to its base at least 120 degrees. To test this, the team teleoperated the manipulator into multiple points along the rotation path and stored the positions. These were combined together into a full motion path which was executed in a single smoot motion. The manipulator was able to rotate beyond 120 degrees to about 170 degrees and there should be nothing preventing the manipulator from completing a full rotation or multiple rotations. Figure 44b shows the test.

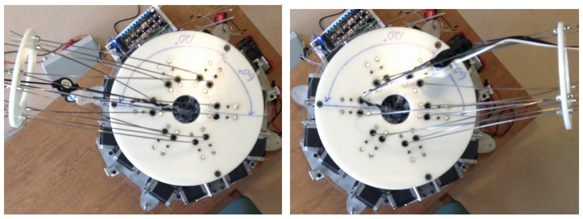

(a) Start of Rotation Test (b) End of Rotation Test

Figure 44: Results of Rotation Test

8.1.3 Precision

The manipulator's precision requirement was for the end-effector to repeatably hit the same point ±6.25 millimeters. For this test, we taped a pointer to the end-effector. Placing paper on the wall directly behind the manipulator, it was programmed to go to a single point, withdraw, and then return to that point. When the end-effector reached the paper, a mark was made just above it. This was done multiple times with the final result estimating a precision of ±3 millimeters as shown in figure 45.

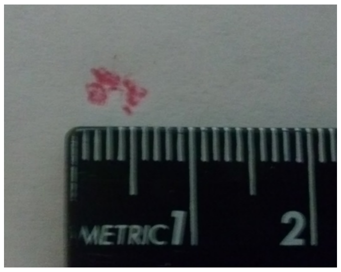


Figure 45: Results of Precision Test

8.1.4 Weight

The requirement was to be able to lift an object weighing at least 5 Newtons. The end-effector itself weighs over 5 Newtons. The manipulator was teleoperated to lift a bottle containing some water successfully.

8.1.5 Compliance

The compliance requirement was for the manipulator to displace no more than 20 mm when a 5 Newton load was applied within the workspace. To test this, the end-effector was removed and the manipulator was moved to the edge of the goal workspace where the applied weight would have the largest effect. A filled water bottle, weighing 5 Newtons, was set at the end of the manipulator where the end-effector attaches. The distance between the top plates before and after the bottle was placed is measured by taking images of each and then using the base plate of 0.30m as a scale reference as shown in figure 46. The manipulator was stiff enough to support the weight in this test.

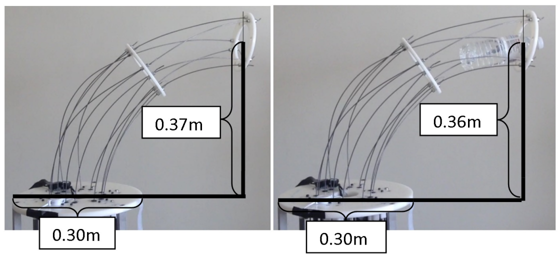
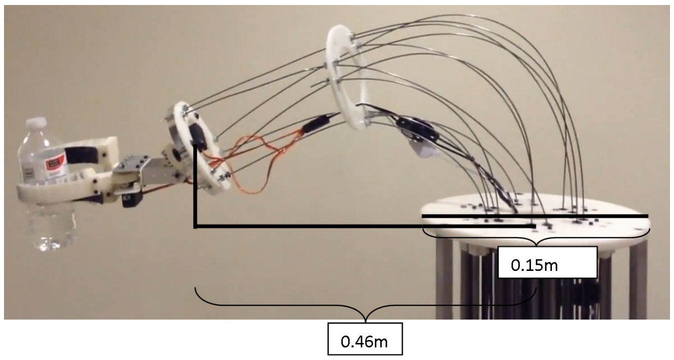


Figure 46: Results of Compliance Test

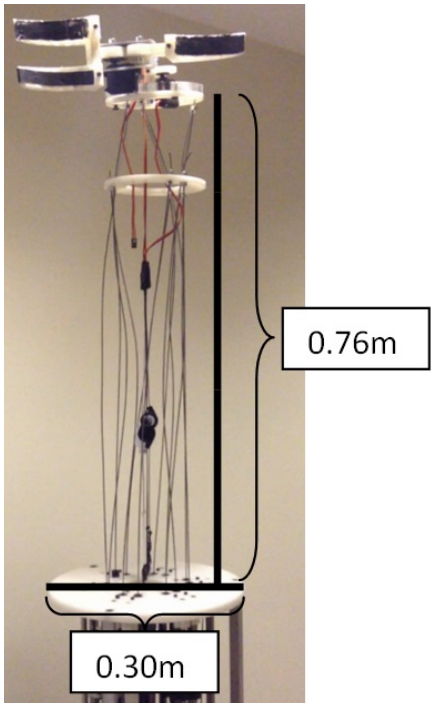
8.2 Requirement Evaluation

8.2.1 Workspace Dimensions

The workspace defined placed the manipulator 0.38 meters from the base with a max height of 0.60 meters.



(a) Max length of the manipulator from base



(b) Max height of manipulator from base

Figure 47: Workspace Dimensions

Using the top of the base plate as a scaled reference point of 0.30 meters, the distance of the arm's maximum length and height from the center of the base can be found. Figure 47a shows the maximum length to be about 0.46 meters while Figure 47b shows the maximum height to be about 0.76 meters. Both of these measurements exceed the original workspace goals, thus meeting this requirement.

8.2.2 Evaluation Summary

Table 8 summarizes the project requirements and compares them against the results of the tests. All the project requirements except for the speed test were successful and the project's final budget was near what was estimated.

Requirement	**Goal**	**Result**
Minimum number of Segments	2	2
Links with Gough-Stewart Configuration	1	2
Workspace dimensions	Figure 7	Section 8.2.1
Rotation perpendicular to base	120°	120°+
Minimum lifting weight	5 N (1.13 lbs)	> 6 N
Precision (Repeatability) of End-Effector	± 6.25 mm	± 3 mm
Compliance	400 mm/sec	250 mm/sec
Software	Inverse Kinematics Agility Demonstration Basic Path Planning	Section 6.5.3.6 Section 6.5.3.5 Section 6.5.3.8

Table 8: Requirements and Results

9 Social Implications

Robot manipulators are often found in university research labs and industrial manufacturing plants, but even after decades of development, remain absent from the homes of ordinary citizens. The lack of domestic robots can be attributed to many factors including lack of software intelligence, high cost, and safety concerns. Continuum manipulators offer a solution to at least one of these problems. If a traditional rigid-link manipulator hits a human, especially those used within today's industrial environments, the human will most likely sustain serious injury and possibly death. However, if a continuum manipulator, which is both mechanically compliant and has all its heavy actuators offloaded into an unmoving base, hits the human, they would be unlikely to sustain any injuries. More than just speculation, the manipulator built in this project had been demoed in a crowded hallway with people directly interacting with it for a number of hours without a single incident. Even during the testing phase of this project, the team was working in close quarters with the manipulator with no barriers or walls yet no one on the team sustained any form of injury.

Additionally, many robot manipulators cost several thousand dollars but the manipulator built in this project was constructed for only $2,000. Since continuum manipulators can be built without a large amount of heavy and expensive materials, they may become affordable to the average consumer in a shorter amount of time. Furthermore, while the inverse kinematics problem was one of the most difficult problems faced in this project, it is still more approachable then many other problems in robot motion planning. It is easier to create a continuum manipulator having many degrees of freedom then it is to do the same with a rigid-link manipulator and it could also reduce the difficulty of finding valid paths through complex environments, meaning domestic robots could possible get by with less intelligence. For all these reasons, continuum manipulators or manipulators, which use similar actuation techniques, have the potential to be among the first commercially available domestic robots and could have a very large impact on the use and availability of robots within society.

10 Conclusion

This project was successful in reaching nearly every design requirement, and even exceeding a few. It required strong cooperation among mechanical, electric, and software system designs to meet these goals. Overall the preliminary estimates for project scope and budget proved true, though the project did take more time than initially expected.

10.1 Mechanical

During sub-system testing the only mechanical issue that occurred was the plastic deformation of the spring steel after repeated use. After the fact, this could have been improved by changing the design with a metal that whose plastic deformation limit is slightly higher, for example nitinol wire could potentially be used in future iterations to solve this problem and help keep the robot easily maintainable. Other than that issue the robot performed well and met the majority of the requirements and the mechanical structure served its purpose.

There is still room for improvement in the design for future iterations. A minor improvement that could have been made is an improvement on the belt tensioning system, if given more time, a cam tensioning system would have been implemented for the belts to solve this problem. Better clamps for the belts could also improve the performance of the robot for repeated use. These improvements could help prevent some of the slipping of the belts which could in turn improve overall performance of the manipulator.

Overall, the arm is mechanically sound and met or exceeded the majority of design goals related to manipulation. Improving the minor issues described above would produce an increase in performance and longevity if implemented in future iterations.

10.2 Electrical

The custom control board for this project worked beautifully. It was able to handle the high power draw from all of the actuators and was able to control all the actuators despite having limited IO pins on the microcontroller. It had enough processing power and storage space to handle every task which needed to be performed and held up even after some accidental damage to a couple on board ICs (which were easily replaced by hand in seconds). The board could have used some better pin labeling and improved connections to off-board power, but these caused only minor inconvenience and didn't interfere with the operation of the manipulator. It should be noted that there was a noticeable increase in temperature of the board after running the stepper motors for prolonged period of time, but it stayed within a safe operating temperature during testing and demonstration.

10.3 Software

The ROS framework chosen for this project both out of familiarity and robustness proved useful. It allows the code to be split effectively into smaller working components and allowed the team to take advantage of some existing ROS nodes to perform operations such as reading from an attached controller. Most of the software performed as desired. There were a number of issues which arose while trying to solve the inverse kinematics problem. These to be due to Google's Ceres solver getting trapped within local minima while searching for valid solutions, producing bad solutions as compared with those acievhed from the MATLAB model, which takes far to long to run in a real system. Should this project continue, the team would probably recommend trying to find a different solver better suited to this purpose or possibly writing their own. One could also attempt to apply non-analytic techniques to solving the inverse kinematic problem, such as neural networks or fuzzy logic. Despite the difficulties encountered, the project was still able to find a working solution by means of "fake IK" to perform the needed tests by using some properties inherent to the manipulator's mechanism.

On the low-level embedded software side, there were no noteworthy problems which would have prevented the project from meeting it's goals.

10.4 Project Logistics

This project followed the timeline as set by a Gantt chart which was updated weekly. Originally this project was to span for the first three terms of a four term year at WPI. However, due to the complexity of the manipulator's design and programming, certain deadlines had to be pushed which lead to continued work during the fourth term which primarily focused on testing and evaluation of our requirements set. Our Gantt chart adapted with every push, and the other areas to be worked were also changed to appropriately accommodate the new working schedule. Even with the extra time added, all final deadlines were met and all but one requirements met.

Our budget planning also followed suit similarly to how we actually spent it. There were a couple deviations however, for example we spent the majority of our budget later than initially planned due to the construction and testing with a second prototype rather than just the first. We were also \$120.81 over budget, but not higher thanks to sponsors such as Pololu for giving us discounted motors and drivers, as well as Advanced Circuits for making the control board designed for free. Going over budget however, is likely to do with the unexpected second prototype that had to be built as well as other unexpected issues such as finding bearings that would work on the smooth rod and servo motors for the gripper. Some of our initial planning accounted for some errors, but not to not be over budget. Though we did go over budget, it was not by much and stayed close to our \$2,000 expected spending.

10.5 Lessons Learned

This capstone project required a large amount of previous knowledge to complete, but also required learning a number of new skills along the way. The following will certainly not be a comprehensive list but should touch on a few primary lessons. First, a lot of general design and implementation skills were practiced, things such as performing calculations in advance to check the feasibility of a solution or project goal, and how to prototype effectively. Team members became more familiar with design tools used in industrial environments such as Altium for designing PCBs, MATLAB for experimenting with mathematical models, Solidworks for designing mechanical components, and ESPRIT for planning manufacturing operations. The team had to gather research in an area where information is lacking (parallel continuum manipulators) and had to learn how to work with the little information that could be found. Building an electrical systems to meet high power requirements is something no team member has done previously as well as figuring out how to control a very large number of actuators. The team also learned more about stepper motors and how they differed from more traditional electric motors. Additional practice in programming both low-level embedded systems and high-level control systems using C, C++, and ROS was garnered while writing the algorithms to operate the manipulator. The team learned about robot manipulators in general and about how to approach solving the mathematics of complicated robot structures, particularly those which involve flexible components. Other logistical and organizational skills were practiced such as how to manage the workload of a large project involving multiple team members with different backgrounds and skill sets; planning deadlines, selecting project goals, and documenting expenses; and talking with company representatives for sponsorships.

10.6 Future Work

The manipulator constructed in this project is believed to be the very first of its kind, a multi-segment parallel continuum manipulator. As such, future projects could advance the technology in many different ways. Future projects could work on improving the quality of the inverse kinematic solutions returned, the time it takes to find a solution, or try to find solutions using smaller amounts of processing power to eliminate the need for a large host computer to run the algorithms on. One could also develop higher level control algorithms which take advantage of the manipulators structure for performing complex tasks or model the dynamics of the manipulator rather than handling only static forces. One could also develop closed loop control algorithms by adding sensors to monitor the true position of the arm, allowing the control systems to compensate for disturbances or achieve even better precision.

On the hardware side, one of the largest problems is the tremendous size and weight of the manipulator. Finding a better way to actuate the manipulator without having such an extremely tall base would make the manipulator far more portable and increasing the arm's max lifting weight while decreasing the weight of the base could make this style of manipulator more useful in real world applications. While on the topic of real world applications, one could explore the possible uses of this style of manipulator by attempting to perform tasks traditional rigid-link manipulators find difficult, like working in small-access holes or cluttered workspaces. The manipulator could be scaled to different sizes or the effects of different levels of compliance and stiffness could be examined. Additional links could be added to the arm enabling even more manoeuvrability in the work space. Ultimately, now that it is known that is is possible to build a working multi-segment parallel continuum manipulator, future projects can explore all the potential benefits this design could offer.

References

[1] Bionic Handling Assistant, publisher=Festo AG & Co. KG, year=2012, month=April

[2] Bionic Tripod 3.0. April 2011.

[3] Sameer Agarwal, Keir Mierle, and Others. Ceres Solver. `<http://ceres-solver.org/>`.

[4] Shaoping Bai and Chuhao Xing. Shape Modelling of a Concentric-tube Continuum Robot. *2012 IEEE International Conference on Robotics and Biomimetics*, December 2012.

[5] Brian Benchof. CONTROLLING SHIFT REGISTERS VIA SPI. `<http://hackaday.com/2011/11/05/controlling-shift-registers-via-spi/>`.

[6] Ian D. Walker Brian A. Jones. KINEMATICS AND IMPLEMENTATION OF CONTINUUM MANIPULATORS.

[7] Caroline E. Bryson and D. Caleb Rucker. Toward Parallel Continuum Manipulators. *IEEE International Conference on Robotics and Automation*, 2014.

[8] Rob Buckingham and Andrew Graham. Reaching the Unreachable- Snake Arm Robots.

[9] Jonathan Fildes. Snake-Arm Robots Slither Forward . *BBC News*, September.

[10] Kurt Gramoll. Mechanics - Theory: Bending Strain and Stress. `<http://www.ecourses.ou.edu/cgi-bin/ebook.cgi?topic=me&chap_sec=04.1&page=theory>`.

[11] Ian A. Gravagne and Ian D. Walker. Manipulability, Force, and Compliance Analysis for Planar Continuum Manipulators. *IEEE Transactions on Robotics and Automation*, 18(3):263–273, June 2002.

[12] IPC. IPC-221A Generic Standard of Printed Board Design, 2003. `<http://www.sphere.bc.ca/class/downloads/ipc_2221a-pcb%20standards.pdf>`.

[13] Frank Grabert Jan Schmitt and Annika Raatz. Design of a Hyper-Flexible Assembly Robot Using Artificial Muscles. *2010 IEEE International Conference on Robotics and Biomimetics*, pages 897–902, December 2010.

[14] Zheng Li and Ruxu Du. Design and Analysis of a Biomimetic Wire-Driven Underactuated Serpentine Manipulator. *Transaction Series on Engineering Sciences and Technology (TSEST)*, 1(6):250–258, October 2012.

[15] P.E. Mahvash, M.; Dupont. Stiffness Control of a Continuum Manipulator in Contact With a Soft Environment. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*, October 2010.

[16] M.D.M.; Ma H.; Armand M. Moses, M.S.; Kutzer. A Continuum Manipulator Made of Interlocking Fibers. *Robotics and Automation(ICRA), 2013 IEEE International Conference*, May 2013.

[17] International Federation of Robotics. Industrial Robot Statistics. `<http://www.ifr.org/industrial-robots/statistics/>`.

[18] International Federation of Robotics. Types of Industrial Robots. `<www.ifr.org/industrial-robots/products/>`.

[19] Pololu. DRV8825 Stepper Motor Driver Carrier, High Current. `<https://www.pololu.com/product/2133>`.

[20] Occupational Safety and Health Administration. Accident Search - Robot, 2014. `<https://www.osha.gov/pls/imis/AccidentSearch.search?acc_keyword=%22Robot%22&keyword_list=on>`.

[21] Carnegie Mellon University. Robot Hall Of Fame - Unimate, 2014. `<http://www.robothalloffame.org/inductees/03inductees/unimate.html>`.

[22] Ronald C. Horm Victor C. Anderson. Tensor Arm Manipulator. *IEEE Transactions on Robotics*, February.

[23] M. Csencsits D. Dawson I.D. Walker B.A. Jones M. Pritts D. Dienno M. Grissom W. McMahan, V. Chitrakaran and C.D. Rahn. Field Trials and Testing of the OctArm Continuum Manipulator. *2006 IEEE International Conference on Robotics and Automation*, May 2006.

[24] Ian D. Walker. Continuoous Backbone "Continuum" Robot Manipulators. *ISRN Robotics*, June 2013.

[25] Zhenqi Zhu, Hongliang Cui, and Kishore Pochiraju. Flexible Parallel Manipulator for Nano-, Meso-, or Macro-Positiong with Multi-Degrees of Freedom, October 2008. `<https://www.google.com/patents/US20080257096>`.

A Authorship

Section	**SC**	**SK**	**CW**
Abstract		X	
Executive Summary		X	X
Introduction		X	
Background	X	X	X
Design Requirements and Specifications		X	X
Project Planning and Logistics			X
Mathematical Discussion - Kinematics, Stroke Length, Bending Torques		X	
Mathematical Discussion - End-effector, Mechanical Analysis	X		
Design - Mechanics	X		
Design - Electronics		X	
Design - Software		X	
Manufacturing - Mechanical	X		
Manufacturing - Electrical		X	
Testing and Results			X
Social Implications		X	
Conclusion - Mechanical	X		
Conclusion - Electrical, Software, Future Work		X	
Conclusion - Lessons Learned	X	X	X
Conclusion - Project Logistics			X
Appendices and Bibliography			X

Table 9: Section Leads

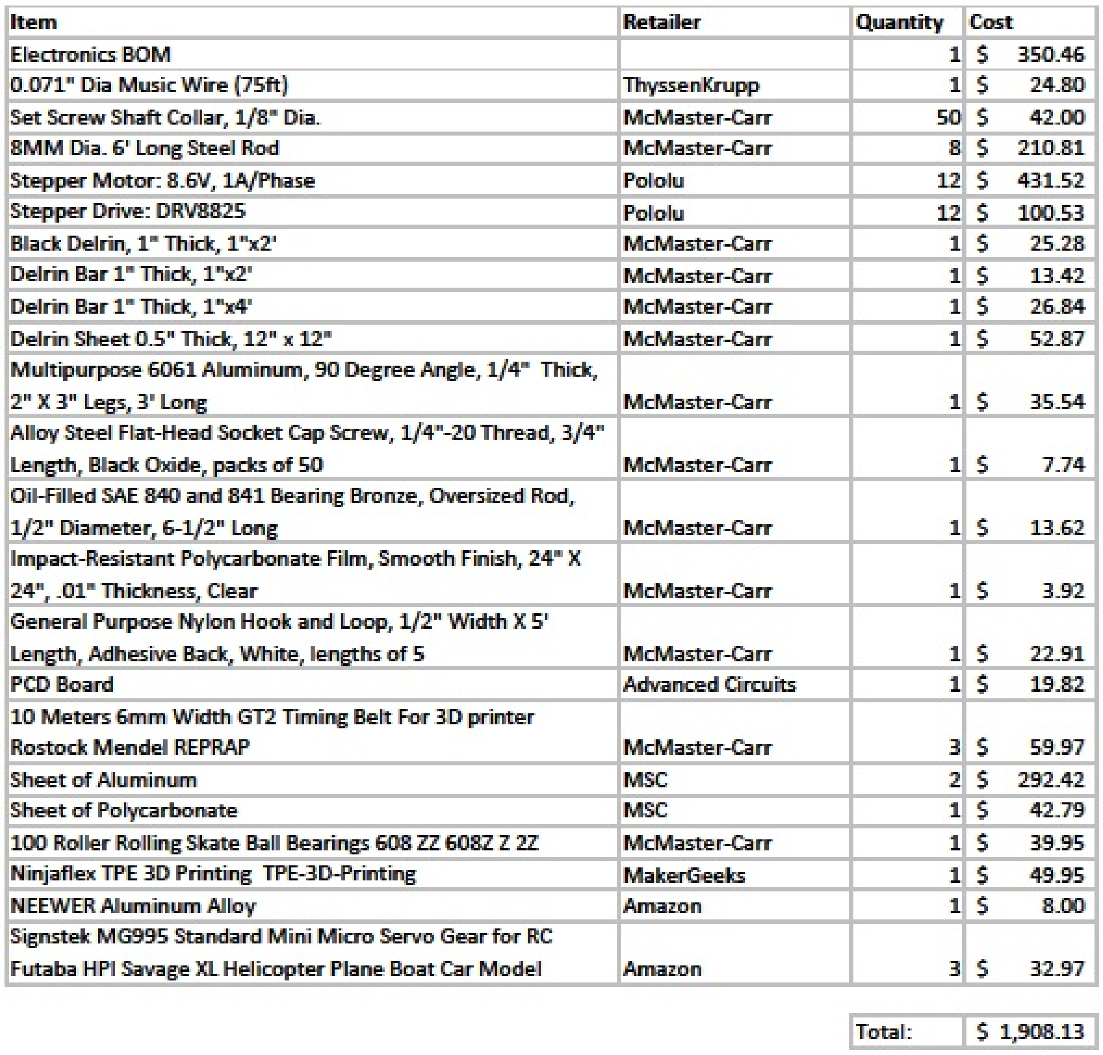
B Bill of Materials (BOM) and Expenses

B.1 Electronics BOM

Item	Part #	Retailer	Quantity	Cost
30V 30A Bench Power Supply	CSI3030SW	Circuit Specialists	1	$ 249.00
MCU	ATmega1284-PU	Digikey	1	$ 7.75
MCU 2x20 pin DIP socket	ED3048-5-ND	Digikey	1	$ 0.44
595 Shift Register	SN74HC595N	Digikey	3	$ 1.89
Shift 2x8 pin DIP Socket	A16-LC-TT	Digikey	3	$ 0.75
fault AND gate	SN74AS808BN	Digikey	2	$ 11.48
AND fate 2x10 pin DIP Socket	ED20DT	Digikey	2	$ 0.48
18.432 MHz Crystal Oscillator	ECS-184-18-9X	Digikey	1	$ 1.02
22 pF >5V Ceramic Capacitor	C0805C220J5GACTU	Digikey	2	$ 0.20
0.1 uF >5V Ceramic Capacitor	GRM188R71C104KA01D	Digikey	1	$ 0.10
1000uF >30V Polarized Capacitor	UVR1J102MHD	Digikey	2	$ 1.92
100uF > 30V Polarized Capacitor	EEU-HD1V101B	Digikey	12	$ 8.53
300 Ohm Resistor	ESR03EZPJ301	Digikey	14	$ 1.11
LED (Blue)	LTST-C150TBKT	Digikey	13	$ 4.80
Limit Switches	SS-3GL13PT	Digikey	6	$ 6.66
2-Pin Jumper conectors	63429-202LF	Digikey	5	$ 1.35
Solder	SMDSW.031 2OZ	Digikey	1	$ 6.30
FTDI Breakout Board	DEV-09716	Sparkfun	1	$ 14.95
Male Headers - Right Angle	PRT-00553	Sparkfun	1	$ 1.95
Male Headers - Straight	PRT-00116	Sparkfun	4	$ 6.00
8-pin Female Header - Straight	PRT-11895	Sparkfun	24	$ 12.00
Power Switches		Sparkfun	2	$ 5.90
Heat Sinks	PRT-11510	Amazon	1	$ 5.88
PCB		Advanced Circuits	1	$ -

Total:	$ 350.46

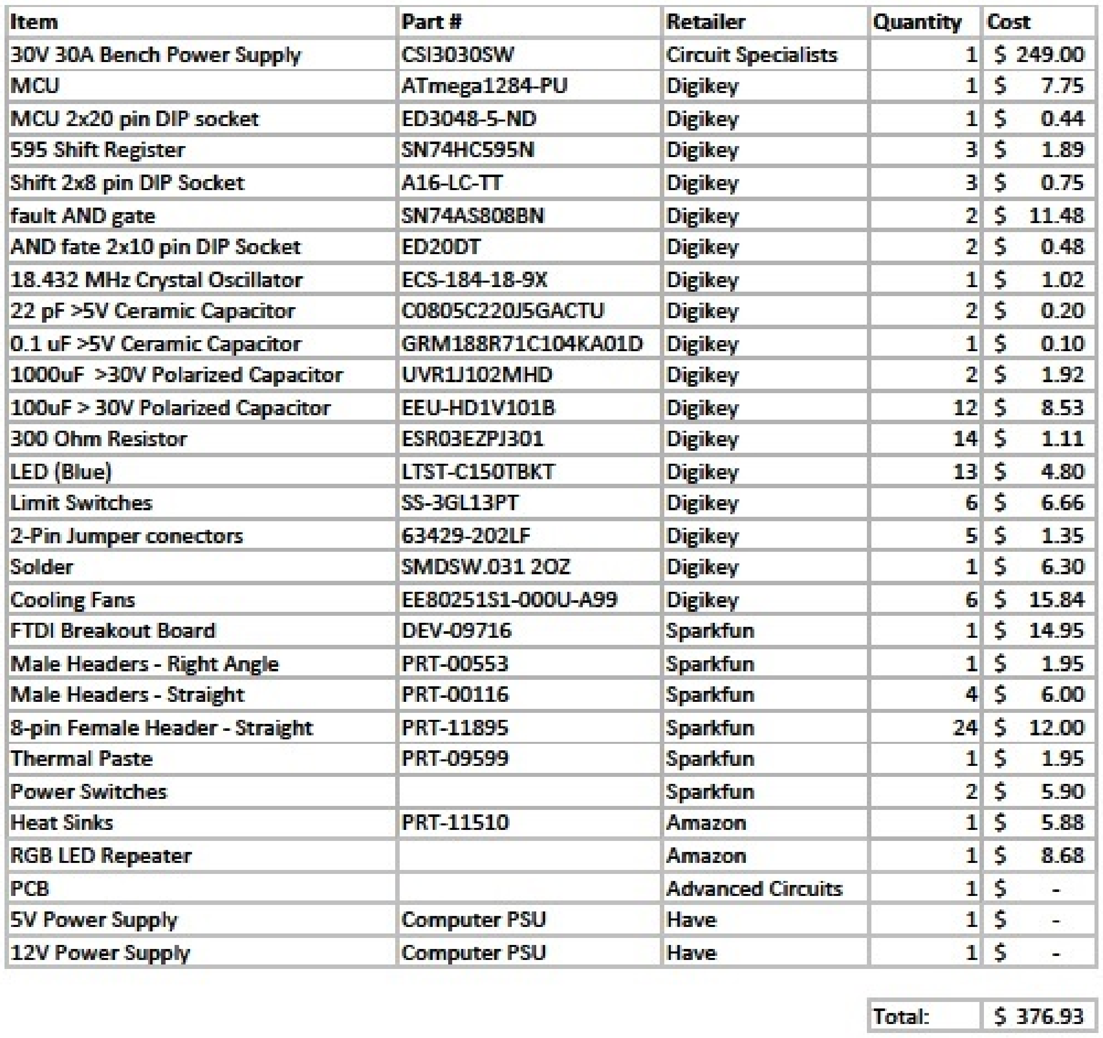
B.2 Project BOM



Item	Retailer	Quantity	Cost
Electronics BOM		1	$ 350.46
0.071" Dia Music Wire (75ft)	ThyssenKrupp	1	$ 24.80
Set Screw Shaft Collar, 1/8" Dia.	McMaster-Carr	50	$ 42.00
8MM Dia. 6' Long Steel Rod	McMaster-Carr	8	$ 210.81
Stepper Motor: 8.6V, 1A/Phase	Pololu	12	$ 431.52
Stepper Drive: DRV8825	Pololu	12	$ 100.53
Black Delrin, 1" Thick, 1"x2'	McMaster-Carr	1	$ 25.28
Delrin Bar 1" Thick, 1"x2'	McMaster-Carr	1	$ 13.42
Delrin Bar 1" Thick, 1"x4'	McMaster-Carr	1	$ 26.84
Delrin Sheet 0.5" Thick, 12" x 12"	McMaster-Carr	1	$ 52.87
Multipurpose 6061 Aluminum, 90 Degree Angle, 1/4" Thick, 2" X 3" Legs, 3' Long	McMaster-Carr	1	$ 35.54
Alloy Steel Flat-Head Socket Cap Screw, 1/4"-20 Thread, 3/4" Length, Black Oxide, packs of 50	McMaster-Carr	1	$ 7.74
Oil-Filled SAE 840 and 841 Bearing Bronze, Oversized Rod, 1/2" Diameter, 6-1/2" Long	McMaster-Carr	1	$ 13.62
Impact-Resistant Polycarbonate Film, Smooth Finish, 24" X 24", .01" Thickness, Clear	McMaster-Carr	1	$ 3.92
General Purpose Nylon Hook and Loop, 1/2" Width X 5' Length, Adhesive Back, White, lengths of 5	McMaster-Carr	1	$ 22.91
PCD Board	Advanced Circuits	1	$ 19.82
10 Meters 6mm Width GT2 Timing Belt For 3D printer Rostock Mendel REPRAP	McMaster-Carr	3	$ 59.97
Sheet of Aluminum	MSC	2	$ 292.42
Sheet of Polycarbonate	MSC	1	$ 42.79
100 Roller Rolling Skate Ball Bearings 608 ZZ 608Z Z 2Z	McMaster-Carr	1	$ 39.95
Ninjaflex TPE 3D Printing TPE-3D-Printing	MakerGeeks	1	$ 49.95
NEEWER Aluminum Alloy	Amazon	1	$ 8.00
Signstek MG995 Standard Mini Micro Servo Gear for RC Futaba HPI Savage XL Helicopter Plane Boat Car Model	Amazon	3	$ 32.97

Total:	$ 1,908.13

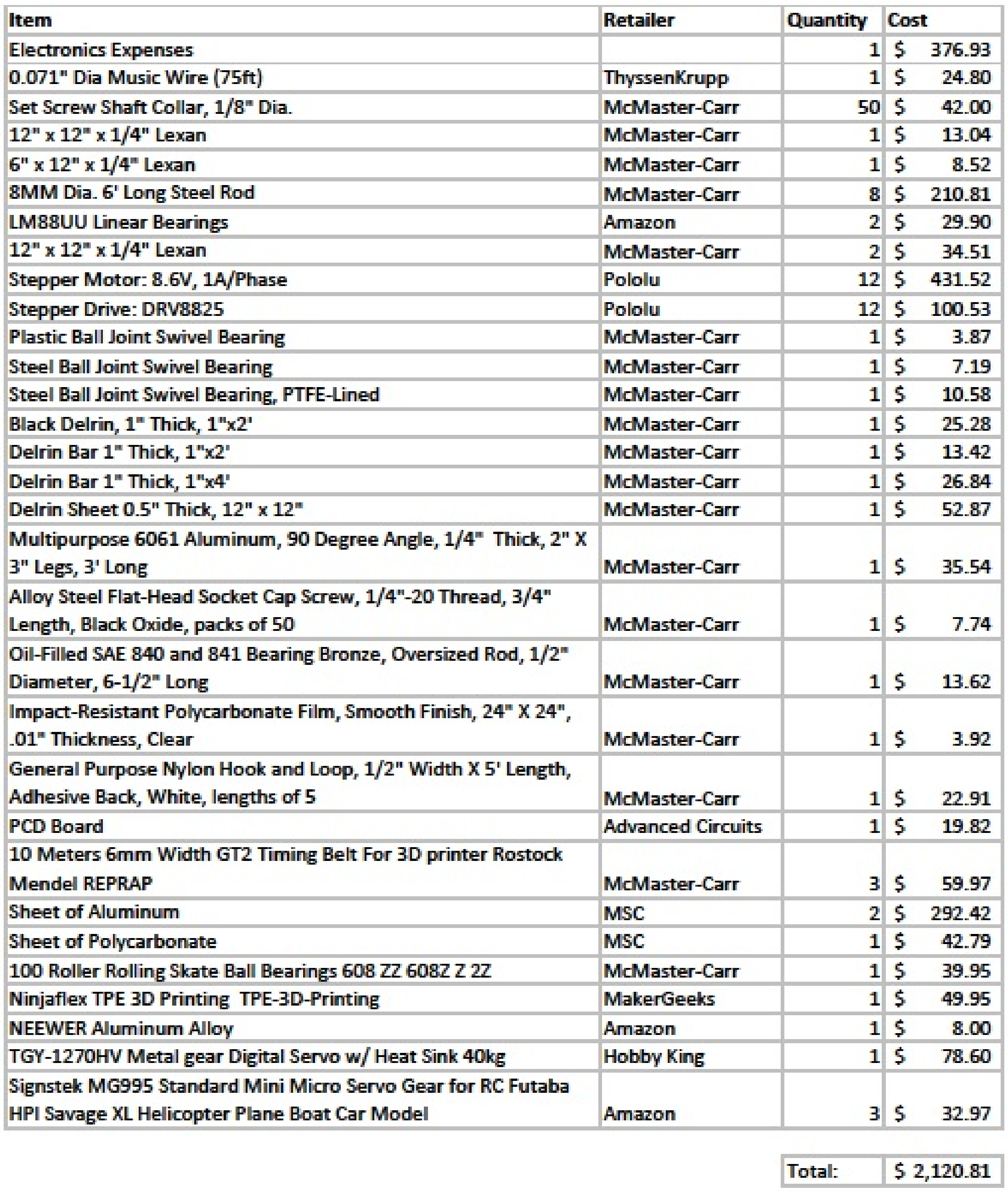
B.3 Electronic Expenses



Item	Part #	Retailer	Quantity	Cost
30V 30A Bench Power Supply	CSI3030SW	Circuit Specialists	1	$ 249.00
MCU	ATmega1284-PU	Digikey	1	$ 7.75
MCU 2x20 pin DIP socket	ED3048-5-ND	Digikey	1	$ 0.44
595 Shift Register	SN74HC595N	Digikey	3	$ 1.89
Shift 2x8 pin DIP Socket	A16-LC-TT	Digikey	3	$ 0.75
fault AND gate	SN74AS808BN	Digikey	2	$ 11.48
AND fate 2x10 pin DIP Socket	ED20DT	Digikey	2	$ 0.48
18.432 MHz Crystal Oscillator	ECS-184-18-9X	Digikey	1	$ 1.02
22 pF >5V Ceramic Capacitor	C0805C220J5GACTU	Digikey	2	$ 0.20
0.1 uF >5V Ceramic Capacitor	GRM188R71C104KA01D	Digikey	1	$ 0.10
1000uF >30V Polarized Capacitor	UVR1J102MHD	Digikey	2	$ 1.92
100uF > 30V Polarized Capacitor	EEU-HD1V101B	Digikey	12	$ 8.53
300 Ohm Resistor	ESR03EZPJ301	Digikey	14	$ 1.11
LED (Blue)	LTST-C150TBKT	Digikey	13	$ 4.80
Limit Switches	SS-3GL13PT	Digikey	6	$ 6.66
2-Pin Jumper conectors	63429-202LF	Digikey	5	$ 1.35
Solder	SMDSW.031 2OZ	Digikey	1	$ 6.30
Cooling Fans	EE80251S1-000U-A99	Digikey	6	$ 15.84
FTDI Breakout Board	DEV-09716	Sparkfun	1	$ 14.95
Male Headers - Right Angle	PRT-00553	Sparkfun	1	$ 1.95
Male Headers - Straight	PRT-00116	Sparkfun	4	$ 6.00
8-pin Female Header - Straight	PRT-11895	Sparkfun	24	$ 12.00
Thermal Paste	PRT-09599	Sparkfun	1	$ 1.95
Power Switches		Sparkfun	2	$ 5.90
Heat Sinks	PRT-11510	Amazon	1	$ 5.88
RGB LED Repeater		Amazon	1	$ 8.68
PCB		Advanced Circuits	1	$ -
5V Power Supply	Computer PSU	Have	1	$ -
12V Power Supply	Computer PSU	Have	1	$ -

Total:	$ 376.93

B.4 Project Expenses



Item	Retailer	Quantity	Cost
Electronics Expenses		1	$ 376.93
0.071" Dia Music Wire (75ft)	ThyssenKrupp	1	$ 24.80
Set Screw Shaft Collar, 1/8" Dia.	McMaster-Carr	50	$ 42.00
12" x 12" x 1/4" Lexan	McMaster-Carr	1	$ 13.04
6" x 12" x 1/4" Lexan	McMaster-Carr	1	$ 8.52
8MM Dia. 6' Long Steel Rod	McMaster-Carr	8	$ 210.81
LM88UU Linear Bearings	Amazon	2	$ 29.90
12" x 12" x 1/4" Lexan	McMaster-Carr	2	$ 34.51
Stepper Motor: 8.6V, 1A/Phase	Pololu	12	$ 431.52
Stepper Drive: DRV8825	Pololu	12	$ 100.53
Plastic Ball Joint Swivel Bearing	McMaster-Carr	1	$ 3.87
Steel Ball Joint Swivel Bearing	McMaster-Carr	1	$ 7.19
Steel Ball Joint Swivel Bearing, PTFE-Lined	McMaster-Carr	1	$ 10.58
Black Delrin, 1" Thick, 1"x2'	McMaster-Carr	1	$ 25.28
Delrin Bar 1" Thick, 1"x2'	McMaster-Carr	1	$ 13.42
Delrin Bar 1" Thick, 1"x4'	McMaster-Carr	1	$ 26.84
Delrin Sheet 0.5" Thick, 12" x 12"	McMaster-Carr	1	$ 52.87
Multipurpose 6061 Aluminum, 90 Degree Angle, 1/4" Thick, 2" X 3" Legs, 3' Long	McMaster-Carr	1	$ 35.54
Alloy Steel Flat-Head Socket Cap Screw, 1/4"-20 Thread, 3/4" Length, Black Oxide, packs of 50	McMaster-Carr	1	$ 7.74
Oil-Filled SAE 840 and 841 Bearing Bronze, Oversized Rod, 1/2" Diameter, 6-1/2" Long	McMaster-Carr	1	$ 13.62
Impact-Resistant Polycarbonate Film, Smooth Finish, 24" X 24", .01" Thickness, Clear	McMaster-Carr	1	$ 3.92
General Purpose Nylon Hook and Loop, 1/2" Width X 5' Length, Adhesive Back, White, lengths of 5	McMaster-Carr	1	$ 22.91
PCD Board	Advanced Circuits	1	$ 19.82
10 Meters 6mm Width GT2 Timing Belt For 3D printer Rostock Mendel REPRAP	McMaster-Carr	3	$ 59.97
Sheet of Aluminum	MSC	2	$ 292.42
Sheet of Polycarbonate	MSC	1	$ 42.79
100 Roller Rolling Skate Ball Bearings 608 ZZ 608Z Z 2Z	McMaster-Carr	1	$ 39.95
Ninjaflex TPE 3D Printing TPE-3D-Printing	MakerGeeks	1	$ 49.95
NEEWER Aluminum Alloy	Amazon	1	$ 8.00
TGY-1270HV Metal gear Digital Servo w/ Heat Sink 40kg	Hobby King	1	$ 78.60
Signstek MG995 Standard Mini Micro Servo Gear for RC Futaba HPI Savage XL Helicopter Plane Boat Car Model	Amazon	3	$ 32.97

Total:	$ 2,120.81