

# Final Report

**Team No. 19**

**Construction Marking Robot**



## **Members:**

Kelsey Howard (knh12d)

Justin Gibbs (jrg13f)

Brandon Roberts (bdr12)

Derrick Portis (dp11d)

Christian Baez (cob11b)

**Advisor: Dr. Nikhil Gupta**

**Sponsor: PSBI – Mark Winger**

**Instructor: Dr. Nikhil Gupta**

April 8, 2016

# Table of Contents

<b>Table of Figures .....</b>	<b>iii</b>
<b>Table of Tables .....</b>	<b>vii</b>
<b>Abstract.....</b>	<b>viii</b>
<b>Acknowledgments .....</b>	<b>1</b>
<b>1. Introduction.....</b>	<b>2</b>
<b>2. Background and Literature Review.....</b>	<b>4</b>
2.1 Construction Industry .....	4
2.1.1 Floor Plans .....	4
2.1.2 Layout of Floor Plans.....	5
<b>3. Concept Generation .....</b>	<b>7</b>
3.1 Initial Design Conceptualization and Selection .....	7
3.2 Initial Design Concept.....	9
3.3 New Design Concept – Gantry Marking Mechanism .....	10
3.4 Sourcing the Linear Actuators.....	13
3.5 Gantry Design Iterations and Fabrication .....	14
3.6 Improvements to Fabricated Assembly.....	16
3.7 Marker Holder .....	17
3.8 Obstacle Detection and Avoidance .....	17
3.9 Electronics.....	19
<b>4. Final Design .....</b>	<b>21</b>
4.1 Marking Mechanism .....	21
4.2 Marker Holder - Revolver.....	23
4.3 Robotic Total Station .....	23

4.4	Obstacle Detection .....	25
4.5	Electronics .....	26
4.5.1	Density Propagation Program.....	26
4.5.2	Stepper Motors .....	26
4.5.3	SmartLynx .....	27
4.5.4	Arduino Mega.....	27
4.5.5	Robotic Total Station.....	28
4.5.6	Batteries .....	29
4.5.7	Raspberry Pi 2 .....	29
4.5.8	Final Program Flow Chart .....	30
<b>5.</b>	<b>Design of Experiment .....</b>	<b>31</b>
<b>6.</b>	<b>Considerations for Environment, Safety, and Ethics .....</b>	<b>32</b>
<b>7.</b>	<b>Project Management.....</b>	<b>33</b>
7.1	Schedule .....	33
7.2	Resources .....	33
7.3	Procurement .....	34
7.4	Communication .....	37
<b>8.</b>	<b>Conclusion .....</b>	<b>38</b>
<b>9.</b>	<b>References.....</b>	<b>39</b>
	<b>Appendix A .....</b>	<b>a</b>
	<b>Appendix B – FMEA .....</b>	<b>b</b>
	<b>Appendix C - Design for Manufacturing and Reliability.....</b>	<b>d</b>
1.1	Assembly Time .....	d
1.2	Components.....	e

1.3 Design for Reliability ..... f

**Appendix D – Operations Manuel..... j**

Functional Analysis ..... j

**Project Specification ..... l**

**Project Assembly..... o**

**Operation Instruction..... p**

Software Instructions ..... p

Hardware Instructions..... s

**Regular Maintenance..... u**

**Appendix E – Specifications Sheets..... v**

**Appendix F – Operation Codes ..... y**

**Biography..... mm**

## Table of Figures

Figure 1: Example Floor Plan.....	5
Figure 2: Example Construction Marks.....	5
Figure 3: Initial Design Concept A.....	7
Figure 4: House of Quality.....	7
Figure 5: Initial Design Concept B.....	8
Figure 6: Initial Design Concept C.....	8
Figure 7: Preliminary Final Designs.....	9
Figure 8: Movement of Preliminary Final Design.....	10
Figure 9: Mounting of Gantry System to Pioneer.....	10
Figure 10: Gantry System Concept.....	11
Figure 11: Initial Gantry Design.....	11
Figure 12: Linear Motion of Gantry.....	12
Figure 13: Mounting of Gantry System.....	12
Figure 14: Assembling Linear Actuator Bundles.....	13
Figure 15: Linear Guide Rail.....	13
Figure 16: Connection of Linear Guide Rails.....	14
Figure 17: Finite Element Analysis on Linear Guide Rail.....	15
Figure 18: Entire System.....	15
Figure 19: Contact between Wheel and Guide Rail.....	16
Figure 20: Marker Holder Design.....	17
Figure 21: SICK LMS 200 LiDar.....	18
Figure 22: Example Construction Site.....	18
Figure 23: LiDar Mount Design.....	18
Figure 24: Distance of LiDar from Mount.....	19
Figure 25: Obstacle Avoidance Algorithm.....	19
Figure 26: Marking Mechanism on Pioneer.....	21

Figure 27: Initial Layout of Support Structures.....	22
Figure 28: Mounting of Guide Rails Together.....	22
Figure 29: Finite Element Analysis on Guide Rail.....	23
Figure 30: Final Marker Holder Design.....	23
Figure 31: Robotic Total Station.....	24
Figure 32: Prism for Tracking.....	24
Figure 33: SICK LMS 200 LiDar.....	25
Figure 34: Point Propagation through Pointor.....	26
Figure 35: NEMA 17 Stepper Motor.....	27
Figure 36: NEMA 23 Stepper Motor.....	27
Figure 37: SmartLynx Motor Driver.....	27
Figure 38: Arduino Mega.....	28
Figure 39: Robotic Total Station.....	28
Figure 40: LiPo Batteries.....	29
Figure 41: Raspberry Pi 2.....	29
Figure 42: Final Program Flow Chart.....	30
Figure 43: Budget allocation by subsystem.....	35

## Table of Tables

Table 1: Pugh Decision Matrix .....	8
Table 2: LiDar Settings .....	25
Table 4: Bill of Materials .....	36
Table 3: Price of parts by subsystem .....	36

## Abstract

Due to the inefficient and error prone way of currently laying out the floor plans of a construction site, Team 19's sponsor, Mark Winger of PSBI has tasked the team to create a robot that will mark out the floor plans of a construction project full scale on the concrete slab. This proof of concept robot should be able to make its marks within 1/2" accuracy, be easily portable, able to mark on concrete, able to mark across 100 sq. ft. within 10 minutes, and be able to navigate autonomously. The final design for this robot consists of the Pioneer 2-DX, which is a differentially steered, mobile research robot that holds a SICK LMS 200 LIDAR system for obstacle detection, with a gantry system and revolver-style marker holder for the marking mechanism which is mounted to the rear of the robot. Additionally, while full communication has not been achieved, significant work has been accomplished towards communication between the robot and the robotic total station, which was to be used for increased accuracy for localization. A raspberry pi 2 microprocessor was used as a communication hub between the Pioneer, the gantry, and the RTS and an Arduino Mega was used for controlling the gantry and marker holder's stepper motors. For increased precision the robot will communicate with a robotic total station, which aids in localization. While this project did take a significant step in the right direction towards developing this construction marking robot, it could be further improved by further developing the support system for the gantry, proper mounts for the electronics, and sourcing new batteries to power the system. However, in its current state the system can take in the CAD of a layout, convert it to useable coordinates, and communicate with the robot and gantry to move accordingly and make marks.



# Acknowledgments

Team 19 would like to thank Pro Steel Building Inc. for sponsoring the project and providing the team with an amazing, motivated liaison to the company.

Team 19 would also like to thank Dr. Gupta, Dr. Shih, and Dr. Collins, the academic advisors to the team, for providing them with the knowledge, criticisms, and critiques.

Team 19 would also like to thank the CISCOR group for the donation of the pioneer 2 mobile robot platform for use in our project.

Team 19 would also like to thank Rob Miller of Florida Building Point for instructing the team on the uses of Trimble's technology and for being a liaison to the company for the use of a robotic total station.

# 1. Introduction

Mark Winger of PSBI introduced the project of a construction marking robot as a way to lead technology into the construction industry. He believes that introduction of robotics in the construction industry will increase both efficiency and productivity in the work force. Currently there is an absence of robotics in the industry because of a lack of trust in automated processes. The construction industry is rooted in the work of tradesmen who learned and excelled at particular aspects of the construction process and did all work manually. Due to this, there is a greater trust in individual manual labor instead of robotics.

With this need in mind, Mr. Winger saw that the process of marking floor plans on the concrete slab of a construction site before installation of interior components was both slow and inaccurate because it is done manually. He proposes a robot that will take the 2D CAD floor plans and plot them on the concrete slab at full scale.

With this information the team found that the current need in the construction industry is a means of increasing efficiency and productivity as well as reducing the amount of time and error that goes into laying out floor plans manually. The goal to meet this need is to implement a proof of concept high precision marking robot that will lay out the preliminary floor plan of a construction site to increase efficiency and productivity in the layout process.

Objectives for the project were determined based on needs and goals determined by the team in the beginning stages of the project. These objectives are what the team hoped to accomplish by the end of the time allowed. The objectives for the completion of the construction marking robot were to:

- Add functionality to the robot to receive a CAD file of a floor plan and convert it into useable coordinates
- Design, fabricate, and implement a marking mechanism
- Make the robot able to navigate autonomously, avoid obstacles, and generate an error report

Design requirements for the construction marking robot were determined by the sponsor for the group. These requirements are what the sponsor wanted the robot to be able to accomplish in its final design. While these requirements were the ultimate goal for the prototype produced by the group, it was realized that the product developed by the team is a prototype and did not achieve all of the design requirements. The design requirements of the construction marking robot were that the final product must be able to:

- Make marks within 1/2" accuracy
- Be easily portable
- Mark on concrete
- Mark across 100 sq. ft. within 10 minutes
- Navigate autonomously

## 2. Background and Literature Review

The idea of a construction marking robot is a fairly recent idea in the construction industry. Currently there is research being done with the idea by companies such as Trimble and DPR, who are combining their specialties in GPS positioning products and construction to create an automated layout robot, or Laybot as they have termed it. This Laybot idea is similar to the group's construction marking robot in that it hopes to be able to mark multiple layouts on the ground, use downloaded 2D CAD files to input the layouts into the robot, and have communication with a robotic totaling station to ensure precise positioning<sup>1</sup>.

Aside from the Laybot idea, there is also a patent for a construction marking robot by Joseph M. Prouty with Totalmark Technologies<sup>2</sup>. While designed to perform the same function as both Team 19's design and the Laybot, the robot by Totalmark Technologies "is controlled via an included tablet pc which is accessible via the internet from anywhere in the world, so long as the job site has Wi-Fi access."<sup>3</sup> It also is linked with a robotic totaling station to track position, and is able to function in complete darkness.

The differences between Team 19's design and the others are the plan of a different marking mechanism and different sensors on the robot. With the resources provided to the group, and the materials available, the final design for the construction robot is guaranteed to be different than the products currently being researched and produced by other companies.

### 2.1 Construction Industry

#### 2.1.1 Floor Plans

Floor plans in the construction industry are the means of communicating how the structure is to be built to each person on the job site. These plans, which are simply 2D CAD drawings designed by an architect, show a scale diagram, as viewed from above, of each component in the structure. Different subsystems of the building, such as interior walls, sprinkler systems, plumbing, HVAC systems, etc. are shown at different levels on the plans for each respective subcontractor. The essential components of these CAD drawings are what the construction marking robot will be transferring onto the concrete slab of the building before internal construction begins. For

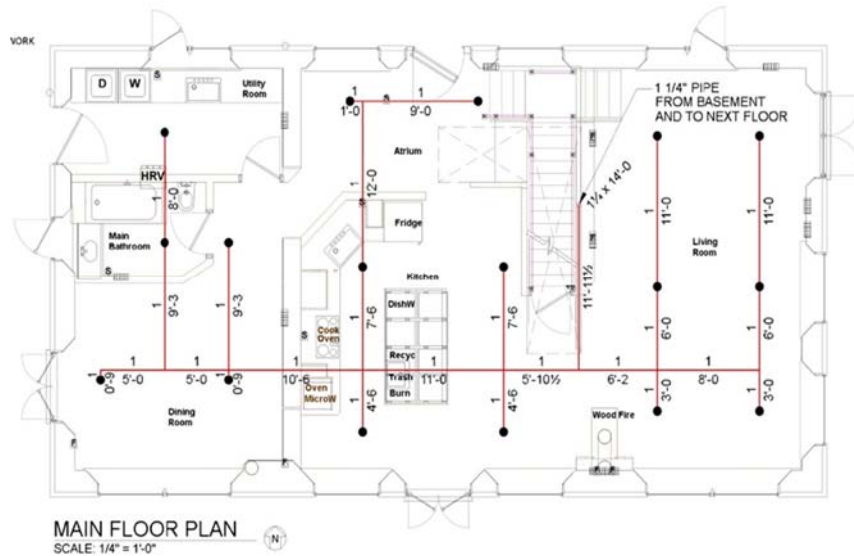


Figure 10 – Example Floor Plan

example, Figure 1<sup>4</sup> shows an example of a 2D CAD drawing for the sprinkler system in a residential building. The construction marking robot would be responsible for marking both the location of the sprinkler heads, as well as the piping leading to each.

## 2.1.2 Layout of Floor Plans

After the concrete slab has been poured, each of the interior components of the building must be marked on the slab to show where installation will occur. These components include locations of interior walls, electrical wall outlets, sprinkler systems, HVAC systems, etc. The current method of laying out floor plans on a construction site is manually. Points from the floor plans are marked on the ground after being measured from a known location on the plans with a ruler, and subsequently on the concrete slab using a measuring tape. If necessary, connections between these points are placed by using a chalk line, such as when laying out the interior walls of the building, as seen in Figure 2<sup>5</sup>, which shows an example of an interior wall marked on the slab. For other subsystems of the structure, such as sprinkler heads, electrical



Figure 11 – Example Construction Marks

wall outlets, and light switches, a simple mark might suffice. Each subcontractor is responsible for marking their own components on the concrete slab, as well as installation of their parts.

There are various ways in which issues can arise when marking the slab manually. For one, there is always error when work is done by humans. This human error can come from using a ruler and tape measure to measure the floor plans and the concrete slab, as well as taking those measurements and translating them to marks on the slab. Human error can also have an effect by taking measurements from different points of origin. This can quickly lead to error propagation throughout the building. Since the majority of the parts come pre-built, such as the interior walls and HVAC system, any small error is multiplied through the building because corrections to the structures are not easily made. For example, if a wall is measured and marked incorrectly by  $\frac{1}{4}$ " , and the next wall is measured from that wall and is also marked incorrectly by  $\frac{1}{4}$ " , it is easy to see how by the end of the building there has been a transmission of error and there is much greater deviation from what is displayed on the floor plans. Finally, with multiple subcontractors coming in to mark their respective components on the slab, there may be discrepancies between them. While the floor plans clearly show where each subsystem is meant to be placed, in actuality many subcontractors and workers rely on their expertise to decide where to place their parts. This means that parts often end up in relatively close, but not exactly where they appear on the floor plans and can cause disagreements when each subcontractor is attempting to do their work separate from the floor plans.

The purpose of the construction marking robot is to alleviate these problems. Since the robot will be working in conjunction with the robotic total station, it will know its exact location in real time. This eliminates the human error of ensuring the marks are in the right location, and also the error propagation because it is able to constantly be checking to ensure it is in the correct location. Finally, since the floor plans will be directly shown on the concrete slab there will be less discrepancies between workers in various subsystems because the exact locations of their respective components are clearly displayed.

### 3. Concept Generation

#### 3.1 Initial Design Conceptualization and Selection

For designing the marking mechanism, the team first constructed a House of Quality, as seen in Figure 3, based on the team’s own analysis of the situation as well as the sponsor’s input through a survey the team provided him with. From this, the team determined that functionality, speediness, and autonomy were important design aspects to focus on according to the HOQ. Feeling as if these results might not quite adequately capture the important aspects of this design,

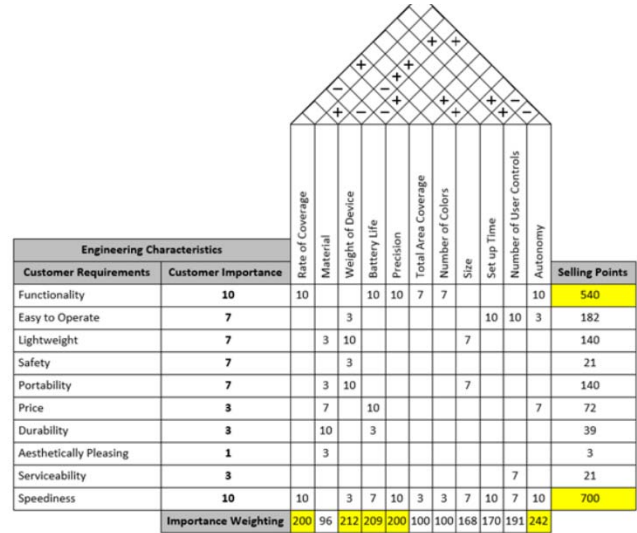


Figure 13: House of Quality

the team analyzed the project proposal and conferred with the sponsor and advisor, using the results of these conversations along with the HOQ to determine that functionality, safety, and ease of operation were the design aspects that the team would focus on the most for the design of the marking mechanism. Once this process had been completed, the team agreed to have a brainstorming session later that week where each team member was to propose at least one design concept keeping in mind the aforementioned key aspects. The three design concepts that can be seen in Figures 4, 5, and 6 below were then agreed upon as the main three starting ideas to consider. Design concept A was fairly simplistic in that it was just a lever arm which held the marker and was driven by a single motor that gave it an arc-like range of potential marking areas, concept B was a bit more complicated in that the marker holder was mounted to a moving pulley

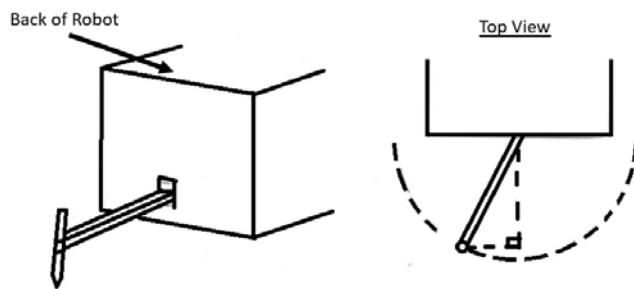


Figure 12: Initial Design Concept A

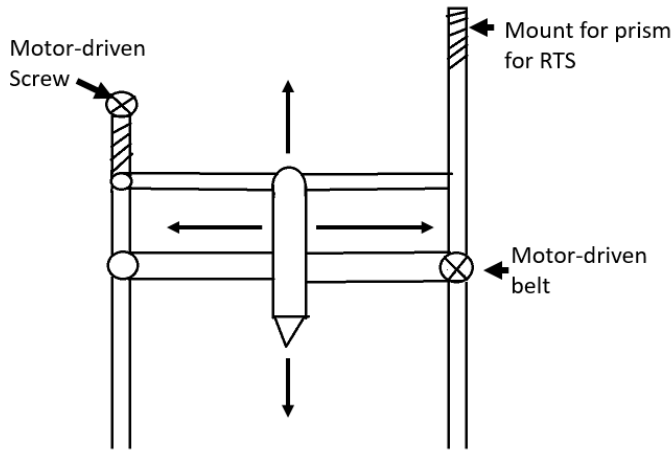


Figure 14: Initial Design Concept B

and could rotate from side to side while the other attached the first to the robot and was driven by a motor which allowed it to move along an axis horizontal to the robot. From here, the team used a Pugh Decision Matrix, as seen in Table 1, to compare these three concepts as well as a potential fourth that the team collectively came up with after evaluating the strengths of the other three designs. Through this, the team selected the fourth combined design concept as the one the team would proceed forward with because, while Concepts A and D had the same total score and A was slightly better than D in ease of operation, Concept A was determined to not be functional enough to be the chosen concept in terms of the types of motion the marker could achieve.

which would be driven by a motor and the entire structure itself could be moved up and down using a motor-driven screw, and concept C was basically a slightly more complicated version of concept A in that it consisted of two lever arms, one of which held the marker with grippers

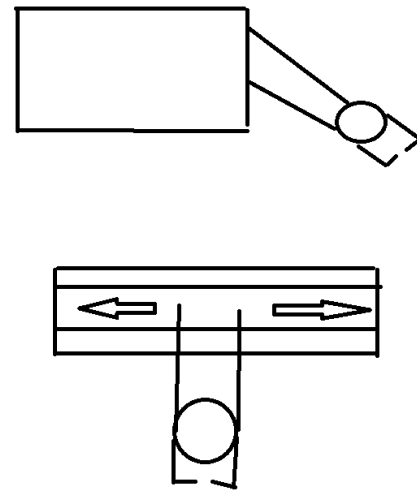


Figure 15: Initial Design Concept C

Table 1: Pugh Decision Matrix

Customer Requirements	Customer Importance	Target Values	Concept A	Concept B	Concept C	Concept D
Functionality	10	10	7	7	9	10



Easy to Operate	7	9	9	8	8	8
Portability	7	8	8	8	7	8
Safety	7	10	9	8	8	9
Price	3	7	9	8	7	8
Durability	3	7	9	7	8	8

### 3.2 Initial Design Concept

After further developing the chosen design concept, the team arrived at the design that can be seen in Figure 7. This design was one which encompassed what the team believed to be the best attributes from the earlier proposed designs. The primary goal of this design was to remain as simple as possible while still allowing the accuracy desired by the customer. This marking mechanism consisted of two servo motors attached to two lever arms, a stepper motor which drives a rack and pinion, and necessary supports and guide rails. The stepper motor-rack and pinion set is meant to allow translational motion along the horizontal plane of the robot, between the wheels, as depicted in Figure 7. As seen in Figure 8, the servo motor-lever arm pairs are meant to allow for rotational motion; the first for raising and lowering the marking arm, and the other for additional reach along the horizontal plane by arching the marking

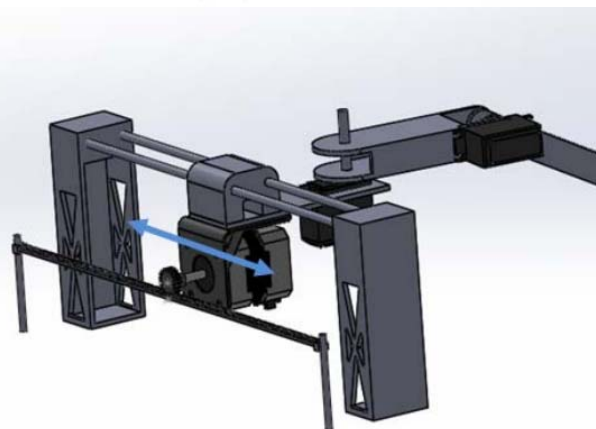


Figure 16: Preliminary Final Designs

arm outside the track width of the robot. However, upon further analyzing this design and consulting the team's advisor, it was agreed upon that the current design was not satisfactory in that it was quite possibly too simple to the point that it would cause the team issues further along the design process. First of all, the current arrangement did not account for being able to use multiple marking colors. While the team did want to originally focus on one color to first achieve functionality of the marking mechanism before moving into such improvements, it was agreed

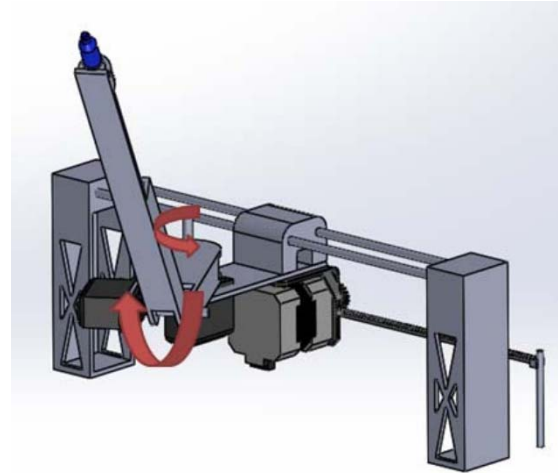


Figure 17: Movement of Preliminary Final Design

upon that the current design accounting for only one marker was too simplistic. In other words, while only designing for one marker would potentially be simpler; it could cause potential issues further along the project once the team tries to integrate a mechanism for switching colors. Additionally, the current marking mechanism is flawed for applying an appropriate force to the marker when marking. With the current design of basically dragging the marker, the resulting marks would most likely either be too light and not straight or too much force could be applied in a vertical configuration which would wear down and potentially break the tip of the marker. As a result, the team, while working on the other aspects of the project, tasked the lead Mechanical Engineer with coming up with a new design concept that would then be proposed to the team.

### 3.3 New Design Concept – Gantry Marking Mechanism

Keeping in mind the necessary design aspects, as well as the needed improvements from the initial design concept, the idea of making a marking mechanism similar to a 3-D printer was proposed.

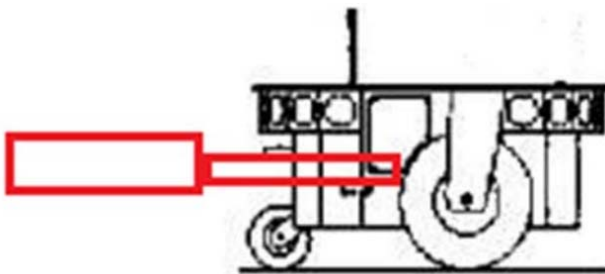


Figure 18: Mounting of Gantry System to Pioneer

The basic idea was to mount two lever arms to the sides of the Pioneer 2-DX, as seen in Figure 9, which will support a square frame which contains motor driven rods, one for moving amongst the mechanism's y-axis, and the other for the x-axis, seen in Figure 10. At the intersection of these two rods would be the

mechanism for holding the marker and, eventually, for selecting the marker, the latter of which would most likely be done with some type of spring-based compression so as to apply an appropriate amount of pressure onto the marker for varying surface levels, as is common with the concrete pours seen on construction sites.

Upon further research and design work, the aforementioned new design concept developed into the gantry design, shown in Figure 11. It

consisted primarily of two linear translation systems which were each comprised of a stepper motor to drive the system, a lead screw to take the rotational motion supplied by the shaft of the stepper motor and convert it to translational motion, a shaft coupler to connect the stepper motor to the lead screw, a mounting platform which moves via the lead screw, and support rods to help stabilize the system and guarantee linear movement. As can be seen in Figure 12, one of these systems would allow for linear motion on one axis while the second would be mounted to the first and provide linear motion on the other axis. The marker holder would then be mounted to the platform on the second linear translation system. The gantry would be mounted to the Pioneer 2-DX using square stock tubing and be positioned in a manner similar to the one shown in Figure 13; additional supports to the top of the robot were also considered for future work as seen fit.

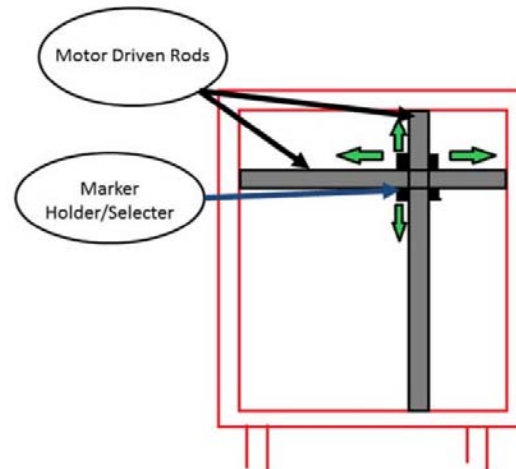


Figure 10: Gantry System Concept

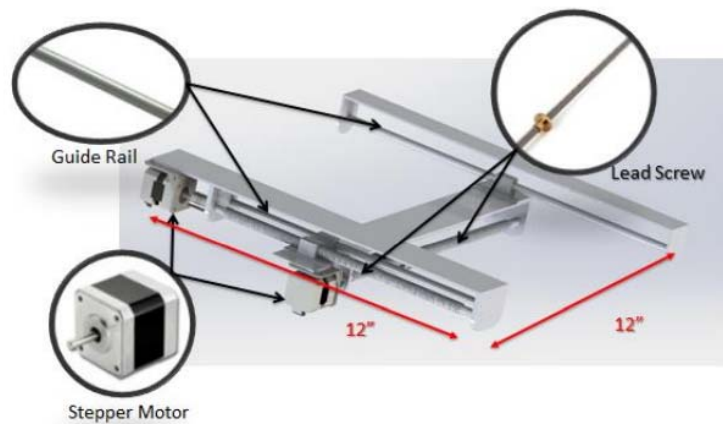


Figure 11: Initial Gantry Design

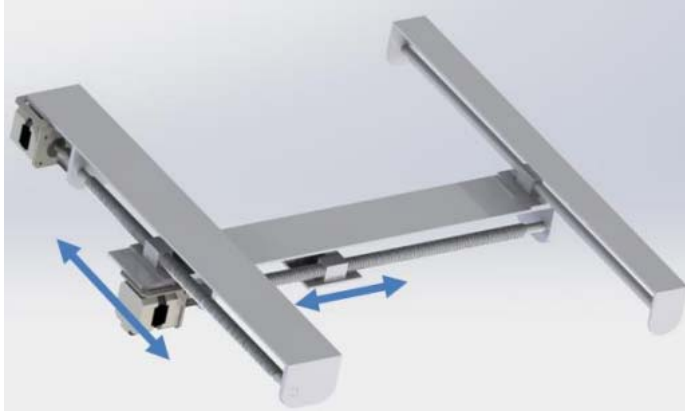


Figure 12: Linear Motion of Gantry

This design was a vast improvement from the previous one in quite a few ways. First of all, as discussed in previous sections, this design, with its platforms for mounting, was far more modular in the sense that altering the design for new marker holders was now effectively as simple as removing the previous marker holder and bolting the new one to the platform. In other words, unlike the

previous design, if the team decided to adapt the marker holder design whether it be for a new method of applying pressure to the marker, adding more potential for colors, or switching the marking medium all together (paint, chalk, etc.), this design now allowed for that without drastic revisions and virtually no need for changes to the rest of the mechanism. Also, the lead screw coupled to a stepper motor design would be far more accurate than the servo motors coupled to the lever arms as in the previous designs and the system would be more stable due to the nature of the gantry design with the support rods as opposed to the lever arms. As an additional feature, the gantry also allowed for the potential to draw shapes with much greater ease, which ended up being of use to the team in the later



Figure 13: Mounting of Gantry System

in the development process. With the previous design this was theoretically possible but would have been more difficult to implement whereas with the current design, the system could operate similarly to preexisting structures such as 3-D printers and laser cutters, moving the marker holder within the gantry to draw the shapes so long as they fall within the dimensional constraints of the gantry.

### 3.4 Sourcing the Linear Actuators

Now that the team had an improved design concept to work off of, the next major factor in continuing the design and fabrication of the marking mechanism would be sourcing parts while altering the design based on what was available as seen necessary. An initial issue the team ran into with this was budget constraints in that, while the team and sponsor agreed that purchasing the two linear actuators as preassembled systems would be ideal for ensuring the desired accuracy of the



Figure 14: Assembling Linear Actuator Bundles

marking mechanism, finding such assemblies that had the desired travel length while still being cost-effective was quite difficult at first, with the first assembly that was sourced being priced at around \$1000, which was nearly half the team's budget. However, upon further research, the team came across the site OpenBuilds, which specialized in smaller scale projects such as hobbyists building their own 3-D printers and CNC mills. Through this site, the team was able to purchase two linear actuator bundles for a little over \$200; a drastic improvement from previous attempts.

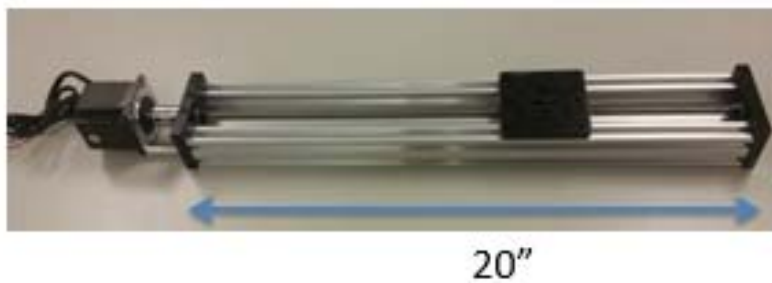


Figure 15: Linear Guide Rail

These linear actuator bundles, as seen in Figure 14 and Figure 15 proved to be an appropriate compromise between purchasing fully assembled actuators and sourcing all the parts separately in that the bundles came in pieces without

instructions, but guaranteed that everything down to the lock collars and shims were properly sized so that the system would operate properly. After a few hours of receiving the parts, the team

managed to have both linear actuators assembled and properly functioning, leaving a majority of the mechanical work for the marking mechanism now just to assembling the two actuators into a completed gantry and designing the marker holder that would be mounted to it.

### 3.5 Gantry Design Iterations and Fabrication

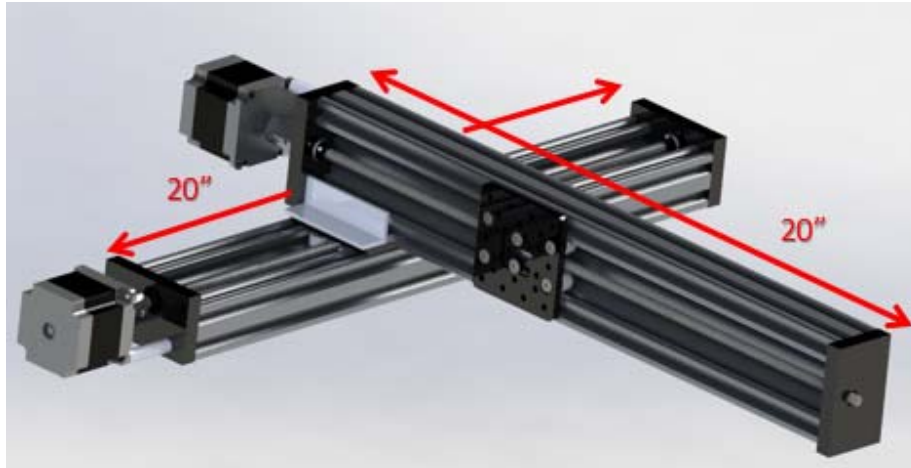


Figure 16: Connection of Linear Guide Rails

Up until this point, the gantry design had been fairly consistent with its initial design concept in that it would be two linear actuators surrounded by a support system that would be mounted to the robot at the sides. However,

upon consulting the machine shop and taking into account such factors as time remaining and the subsystems that relied on the gantry's completion, such as the marker holder's height and all of the programming for adding movement to the gantry, the design was altered to be more simplistic, while still being at least adequately functional so the project could move forward. As can be seen in Figure 16, this new design proposed an attachment system using brackets without any external supports. As represented by Figure 17, testing the design through Finite Element Analysis proved that, while the realistic gantry would probably have a slight tilt, the structure itself would only deform about  $6 \times 10^{-2}$  inches even when a load of 50 lbs was concentrated on the end of the cantilevered actuator. The idea behind this design was that the first actuator would be mounted directly to the top of the robot and second actuator would be mounted to the first in such a manner that a majority of its weight (being mainly as a result of the stepper motor) would be focused in the same area as the first actuator. In order to complete this design, two sets of brackets were fabricated from angle iron; one set for holding the actuators together, and one for mounting the system to the robot. These brackets were then fastened using bolts, lock nuts, and t-slot nuts where appropriate. Another key variance of this design from the originally designed gantry is the fact that the second linear actuator was rotated so that it sits on its side. This was done to increase simplicity

for designing the mounts for the marker holder and RTS prism in that the height of the marker holder was no longer limited and the prism could now easily be mounted to the platform in a position within view of the RTS. This alteration proved incredibly useful in that, when the team needed to make a makeshift marker holder for the sake of demonstrating the project, this design was easily mounted to the gantry within minutes with no issues whatsoever. Additionally, the fact that this variation of the gantry design was able to be constructed in three days with under \$30 of additional cost, it could very well be considered a success in terms of what it needed to accomplish. A picture of this assembly can be seen in Figure 18.

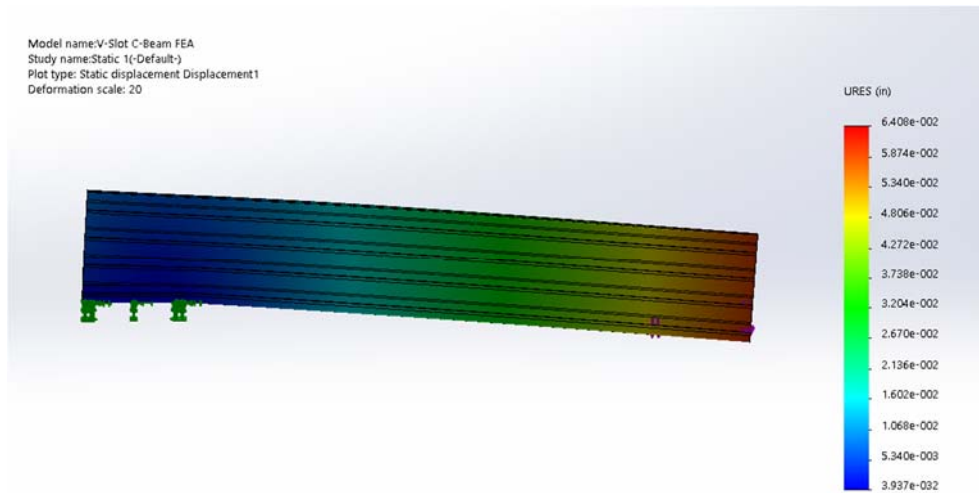


Figure 17: Finite Element Analysis on Linear Guide Rail



Figure 18: Entire System

### 3.6 Improvements to Fabricated Assembly

While the current gantry assembly functioned properly as desired, the team realized from repeated testing that its motion was not as smooth or quiet as it should be. Upon further inspection, as can be seen in Figure 19, the team discovered that one of the nuts from the wheels on the gantry platform were making contact with the guide rail, causing it to occasionally scrape as it moved, most likely as a result of the slight tilt caused by the cantilever mounting design. While this was minimal enough that it did not prevent the gantry from functioning, the team decided to have the machine shop grind down one of the unnecessary edges of the rail that the nut was scraping against. Though this would put a higher load on the wheels since there was no longer and contact support from the nut, doing this greatly increased how smooth and



Figure 19: Contact between Wheel and Guide Rail

quietly the gantry could travel now. Another minor issue discovered was similar in the sense that a nut was scraping against the guide rail, but this time it was in the cantilevered linear actuator. This scraping was far less apparent and hindering than the previous case, but did cause occasional displeasing noises, so the team decided to try to rectify this. Upon repeated testing, the team determined that this scraping was not due to tilt, like in the previous case, but was more likely due to a slight inconsistency in the surface of the linear rail when it was fabricated. As such, and since the scraping was so minimal, the team discovered that applying WD-40 to the guide rail at the point where the scraping was occurring was sufficient to solve the problem. Eventually this may



need another application, but so far after two, nearing three, weeks of repeated testing there have not been any issues.

### 3.7 Marker Holder

Building off of the team's advisor's suggestion about designing a marker holder which accounted for using more than one color, the team decided to create a marker holder which could use three colors as a starting point for the development of the robot. Eventually the team reached the design concept depicted in Figure 20 which was a revolver design. The basics behind this idea were that the marker would be held in the chamber, initially suspended by springs, then, as the stepper motor rotated the shaft for the marker holder, a wedge shape would push down the marker being used. The only further iterations that this design really underwent were as a result of changing the mounting location for the marker holder, be it underneath or to the side of the second linear actuator in the gantry.

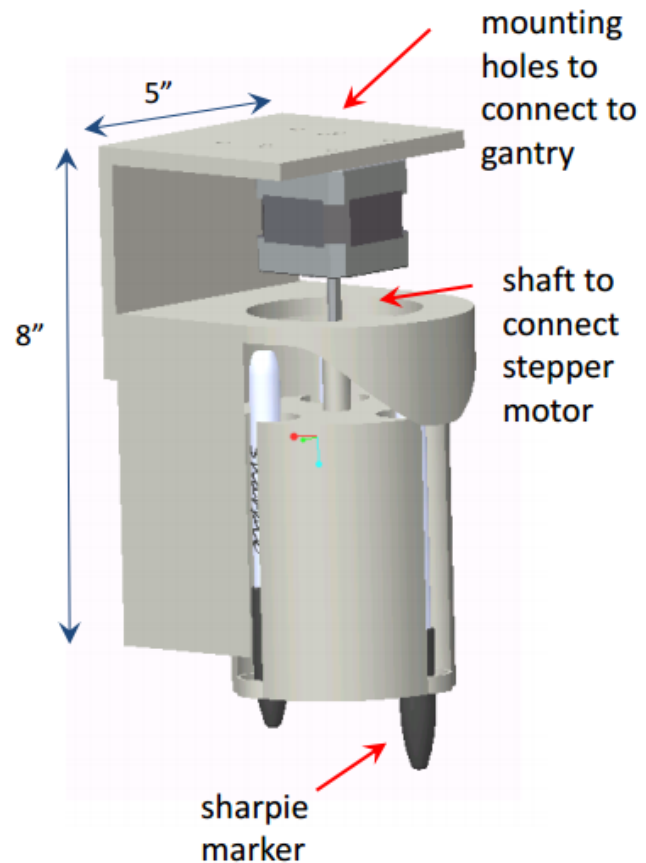


Figure 20: Marker Holder Design

the mounting location for the marker holder, be it underneath or to the side of the second linear actuator in the gantry.

### 3.8 Obstacle Detection and Avoidance

For obstacle detection, since the team was already planning on using the aforementioned Pioneer robot, the team decided to use the robot's onboard SICK LMS 200 LiDar, as can be seen in Figure 21. For more detail on this LiDar, please refer to the corresponding subsection in the final design section of this report. Additionally, this system was considered adequate for the team's purposes

since it can measure the distance of objects within a 180 degree field. However, as can be seen by the representation of typical obstacles the robot could encounter in a construction site in Figure 22, the team realized that it would be beneficial if the LiDar was angled downward. As can be seen in Figures 23 and 24, the following mount was designed to angle the LiDar at an incline of about 10 degrees so it could see slightly over 5 feet in front of the robot. In terms of an obstacle avoidance algorithm, the team came up with the basic idea,



Figure 21: SICK LMS 200 LiDar



Figure 22: Example Construction Site

depicted in Figure 25 that once the robot detects that an obstacle is within its radius of influence, the robot will veer from its desired path and continuously check for the obstacle, returning once it is clear of said obstacle. However, despite these design concepts, there was not enough time left to fully develop them, so the team ended up just using the LiDar as is so it can be employed for future use.

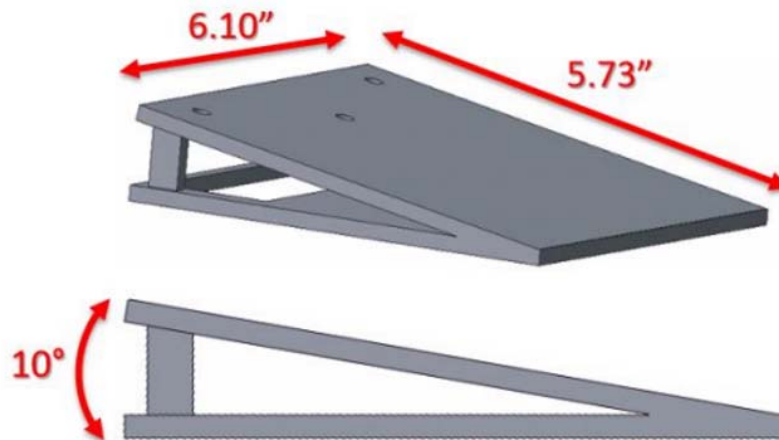


Figure 23: LiDar Mount Design

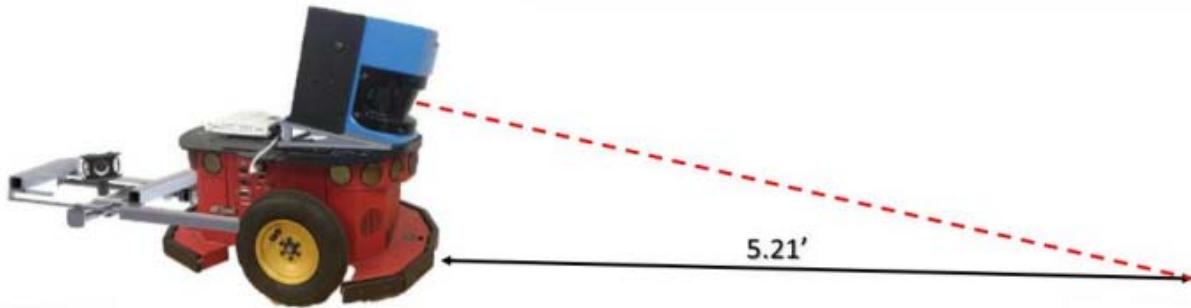


Figure 24: Distance of LiDar from Mount

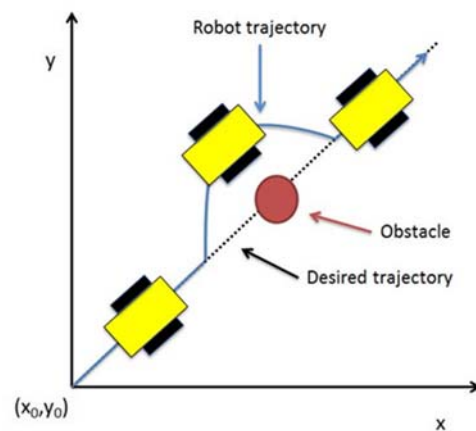


Figure 25: Obstacle Avoidance Algorithm

### 3.9 Electronics

Different components of our design when through changes throughout the year. The Ransen software, Pointor, which only allows for the user to upload a dxf file and returns a text file of the endpoint coordinates had to be improved upon. This endpoint to endpoint system wouldn't work for this project due to the robot having to eventually avoid objects; the robot would have to skip over the entire line if it ran into anything. So, the team developed a density propagation program which takes the two endpoints and creates intermediate points a half inch apart so that the robot would be able to return to the closest point after avoiding an obstacle.

The Arduino Uno is the first microcontroller used for the project. Everything worked fine on this microcontroller, but more pins were needed after having to insert three motor drivers. An Arduino Mega, which works from the same IDE as the UNO was purchased to solve this problem.

The Mega is almost identical to the UNO on the software side, but the Mega offers more memory and 54 pins.

Arduino products also have a driver that supports two stepper motors that the group started researching. It came with a fully library and functions that were supported by Arduino, and they had the ability to be stacked on one another if more motors need to be driven. The problem with the drivers is that they did not draw enough current to properly move the stepper motors. That improper current draw would cause the marker to mark inaccurately. Instead the group purchased SmartLynx stepper motor drivers which were inexpensive and had a current rating of up to 7 Amp.

The Raspberry Pi 2 comes with a default operating system called Raspbian. This OS currently cannot run executable files. The group reconfigured the Raspberry Pi so that a new OS, Windows IoT Core, that would be able to run the Pointor software which is an executable file. Upon running the new OS, the functionality of the Raspberry Pi was lost due to the fact that you could not code directly from the Raspberry Pi. With this hurdle, the group decided it would be best if the Raspberry Pi was just in its original OS and the Pointor software would run before or at going to the construction site.

A Laser Measurement System 200 LiDar is mounted to the top of the Pioneer and used to detect any obstacles in the way of the robot's path. Running into anything would cause the robot's coordinate system to be off. The LMS is a non-contact measurement system and would alert the robot if something came too close to touching the robot. After detecting an object, the path planning algorithm would begin and the robot would move around the object until the LiDar can't see the object with its 180° range of view.

## 4. Final Design

The team's final design for the construction marking robot consists of the Pioneer 2-DX as the robotic platform along with its SICK LMS200 LiDar for obstacle detection with the gantry system mounted to the back of the robot as the marking mechanism with the revolver marker holder mounted to the platform on the gantry. While communication between the RTS and the robot has not been completed due to technical issues with the Trimble equipment, there is a mount for the prism to go on top of the marker holder and the RTS can properly track said prism while the robot is in operation. In terms of the system's functionality, it is powered by a series of LiPo batteries and the functions for converting the 2-D CAD files of the layouts to useable coordinates and angle values as well as those for controlling the robot and gantry have also been fully developed. For more specifics on each individual subsystem, please refer to the subsections below.

### 4.1 Marking Mechanism

The final marking mechanism being implemented on the robot, as can be seen in Figure 26, is a gantry system consisting of two linear actuators bought as bundles from OpenBuilds and assembled by the

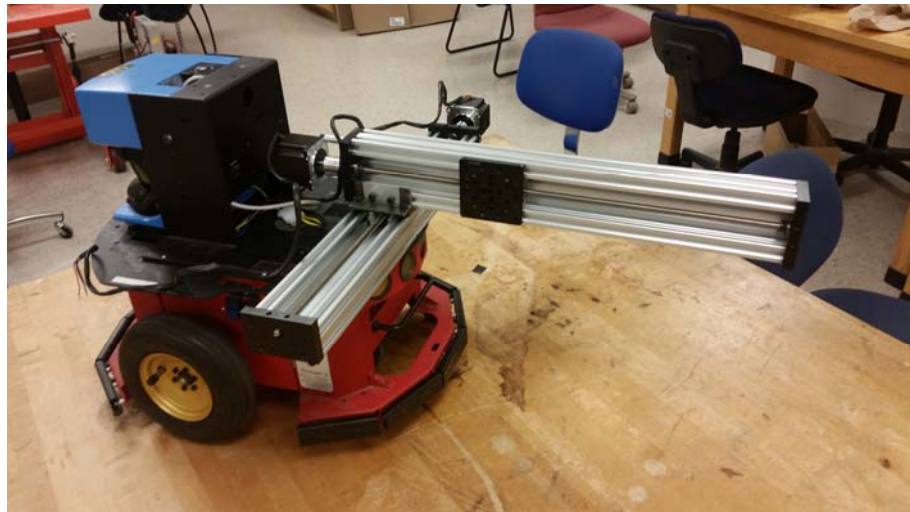


Figure 26: Marking Mechanism on Pioneer

team. Each actuator allows for a travel of roughly 20 inches which was reduced to about 17 after the brackets were added to complete the gantry; despite this reduction though, the design still exceeds the desired goal of a workable area of 12"x12" for the gantry. These actuators are each powered by a NEMA-23 stepper motor which drives a lead screw with a pitch of 2 mm. Originally, the gantry was supposed to be assembled with a series of support structures, similar to those in



Figure 27: Initial Layout of Support Structures

Figure 27, but for the sake of assembly time and weight reduction, the team ended up using a bracketed cantilever structure similar to the one in Figure 28 where the first actuator would be mounted directly to the top of the robot. To verify that this less supported structure would not fail, Finite Element Analysis was performed on the cantilevered actuator, as can be seen in Figure 29 (note that the deformation scale was increased to 25 times the actual case so the deformation could be visualized). Due to unresolved meshing errors, the study had to be simplified down to just the linear rails; the bracket was simulated by creating a fixture to the rail of the same size and location as the brackets and the system was tested by applying a worst case scenario where a 50 lb load was applied at the very end of the beam. Despite this force being far greater than any load the gantry would realistically experience, the beam only deformed about  $6 \times 10^{-2}$  inches. This was further proved by the fact that after repeated testing, the gantry never mechanically failed. A few specific features to note about this design

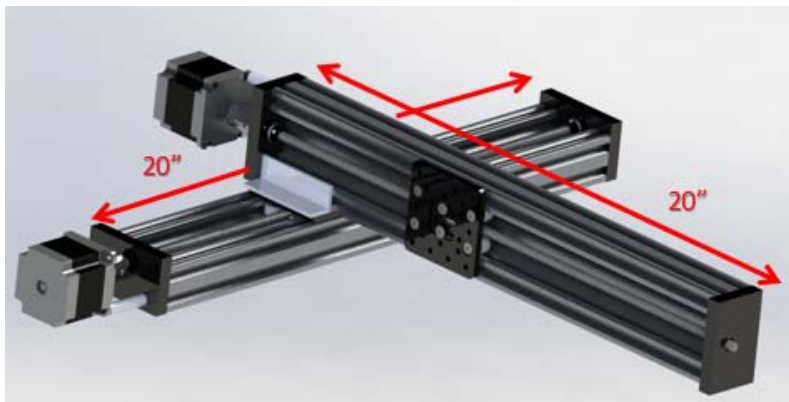


Figure 28: Mounting of Guide Rails Together

aside from the aforementioned dimensions are the fact that the marking mechanism can reach outside the range of the robot if needed, the mounting plate design allows for easily switching out marker holders, and the orientation of the cantilevered actuator being sideways allows for easy mounting designs of varying marker holders and mounts for the RTS prism.

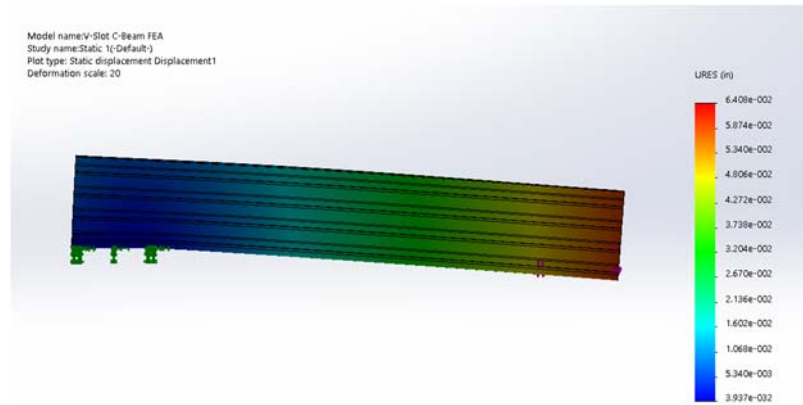


Figure 29: Finite Element Analysis on Guide Rail

## 4.2 Marker Holder - Revolver

The final design for the marker holder, as can be seen in Figure 30, is roughly 10 inches tall and 5 inches wide. To select a color, the NEMA-17 stepper motor will rotate a full 120 degrees to change colors, and 60 degrees to not have a single marker selected. It is currently being 3-D printed and will be mounted to the platform of the marking mechanism once it is completed.

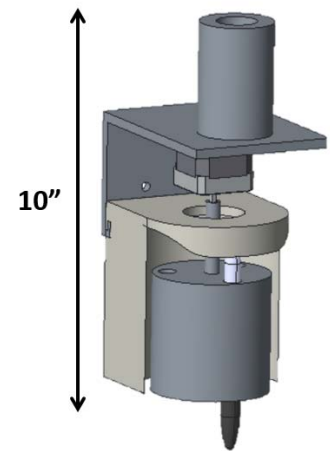


Figure 30: Final Marker Holder Design

## 4.3 Robotic Total Station

In order for the robot to plot the floor plans in the correct positions the robot will need to implement localization or know its exact location in the construction site. Some sensors that can be used for localization include GPS, visual sensors, and wheel encoders. These sensors do not have enough accuracy for our robot. The sponsor had an idea of implementing an existing technology in the construction industry into the project for localization.

Localization will be done with the use of a robotic total station. A robotic total station similar to the one being used by the team can be seen in Figure 31. It will be provided by Trimble and will include a reflective prism for tracking, a tablet for communication, a software suite along with an API for integration of the raspberry pi and robotic total station.

A robotic total station is a measurement system similar to the LIDAR system previously noted. The RTS uses optical sensors to measure the exact position of a reflective prism which can be seen in Figure 32. The RTS makes its measurements by knowing the angles at which it is oriented and the slope distance to the target.

The RTS uses triangulation to specify its location in the global frame. This is done by knowing 2 points of interest and their global coordinates to measure where it is in comparison to the floor plan. The RTS will be used to be sure that the marks the robot makes are in the correct position in relation to this floor plan as well. Localization via the RTS is necessary because real-life factors such as surface imperfections, wheel slippage, and sensor reading errors can lead to inaccuracies in the readings from the laser range finder and imbalance in the wheels will cause the internal coordinate system to be inaccurate.

The team was provided a formal class dealing with Trimble's technologies including the robotic total station and 3D point cloud analysis. It was administered by a sales representative and ex-employee of Trimble, Rob Miller. Rob has recently communicated with Trimble about our team and the project. Trimble is currently in the process of donating all of the technology the team will need to implement this system into our design. This is great because the typical package cost is in excess of \$40,000 which is well out of the team's budget. Rob also plans to meet with the team a few times next semester for guidance with the project since he did work with the Project Lion explained in the introduction.



Figure 31: Robotic Total Station



Figure 32: Prism for Tracking



## 4.4 Obstacle Detection

The Sick LMS 200 LiDar, as can be seen in Figure 33, came with the Pioneer and will be used for obstacle detection. It has 3 different settings that it can operate in which can be seen in Table 2 and are differentiated by the number of measurements made in one pass and the field of vision. The team has chosen the setting in which the LMS 200 has a full or 180 degree field of vision and the most number of measurements for that field of vision. It is labeled as having an angular resolution of 0.5 degrees. This was done to ensure that the LIDAR has the most amounts of data possible for the full range of vision. This was done because some of the obstacles encountered on the construction site are small or thin and the robot needs to be able to avoid them.



Figure 33: SICK LMS 200 LiDar

**Table 2: LiDar Settings**

<b>Angular resolution</b>	<b>0.25°</b>	<b>0.5°</b>	<b>1.0°</b>
<b>Max Scanning angle</b>	100°	180°	180°
<b>Max # of measure values</b>	401	361	181

## 4.5 Electronics

### 4.5.1 Density Propagation Program

After running the dxf file through Ransen software, the text file of coordinates are sent to the density propagation program, which is a program that could read in the coordinate pairs and generate intermediate coordinate points that are a half inch apart. By adding these intermediate points there is a greater point density, in turn making less error in the line when moving around an obstacle. This way instead of leaving the rest of the line unfinished, the robot will be able to get back onto the closest intermediate point after the obstacle and continue the line.

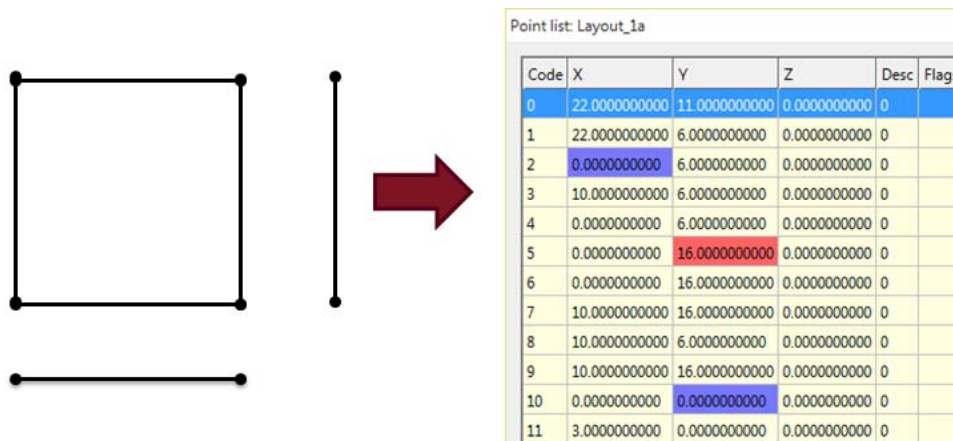


Figure 34: Point Propagation through Pointor

### 4.5.2 Stepper Motors

The gantry system and the marker holder is mechanically driven by stepper motors. The gantry system compose of two NEMA 23, a bipolar stepper motor moving the X and Y axis of the gantry. Stepper motors were selected over DC motors because of the precision the stepper motor gives the robot. The motors are placed at the ends of the linear guild rails, connected to the lead screw, and held together with a coupler. The NEMA 17 that is rotates the marker holder to different colors and non-marking locations, will be smaller than the motors for the gantry because the marker holder does not requires as much torque.



Figure 35: NEMA 17 Stepper Motor



Figure 36: NEMA 23 Stepper Motor

### 4.5.3 SmartLynx

The stepper motor drivers connects the stepper motors to our microcontroller. They also push enough current to run the stepper motors. The stepper motors that the group use require 2.8 Amps/phase, so motor drivers need to be powerfully enough to run the stepper motors. The SmartLynx bipolar stepper motor driver is a SPI based motor driver and fairly inexpensive compared to other motor driver with the current rating that the SmartLynx has.



Figure 37: SmartLynx Motor Driver

### 4.5.4 Arduino Mega

A microcontroller is used to actually program the stepper motors to move to specific locations with precision. The Arduino Mega was selected because of its many pins and simplicity. Arduino supports C/C++ programming language. There were libraries that supported the SmartLynx motor drivers. The Mega is connected to all stepper motor drivers to command them to move the stepper motors a certain distance or number of steps. The Arduino Mega currently

runs a program that drives the gantry to draw various shapes on the ground. Most of the marking that the robot is going to do will be straight lines and will only require the gantry to make small adjustments to compensate for error, but there are shapes that have to be drawn on the concrete. The program drives the motors to move a certain number of steps in an ordered list to make the shapes.



Figure 38: Arduino Mega

#### 4.5.5 Robotic Total Station

Another sponsor, Trimble, donated a robotic total station (RTS) with a radio for communication for receiving real-time information to error correction. The RTS comes in two different modes (tablet mode and OEM mode). Since the robot can't read the tablet, OEM mode is needed to receive the coordinates from the RTS. The radio uses serial connection, but a serial to USB cable is used to connect to the Raspberry Pi.



Figure 39: Robotic Total Station

### 4.5.6 Batteries

LiPo batteries were used to replace the older DC batteries that exist in the Pioneer. The previous batteries were heavy and would not meet our sponsor's requirements of having the robot being light enough to be carried by one person. The old batteries also would not run as long as the sponsor would've like as he eventually wants the robot to run overnight. Our advisor recommended we purchase these LiPo to power the Pioneer, Raspberry Pi, Arduino Mega, and the SmartLynx motor drivers.



Figure 40: LiPo Batteries

### 4.5.7 Raspberry Pi 2

The Raspberry Pi 2 allows all the components to communicate with one another. This microprocessor will connect to all subsystems through USB or I2C, allowing the Raspberry Pi to send commands out to have the components work in unison. The Arduino Mega has to receive values to instruct what direction and how far to move the marker holder. The Pioneer needs to be commanded on where to move, and the LiDar has to alert the Pioneer that something is in the way.



Figure 41: Raspberry Pi 2

## 4.5.8 Final Program Flow Chart

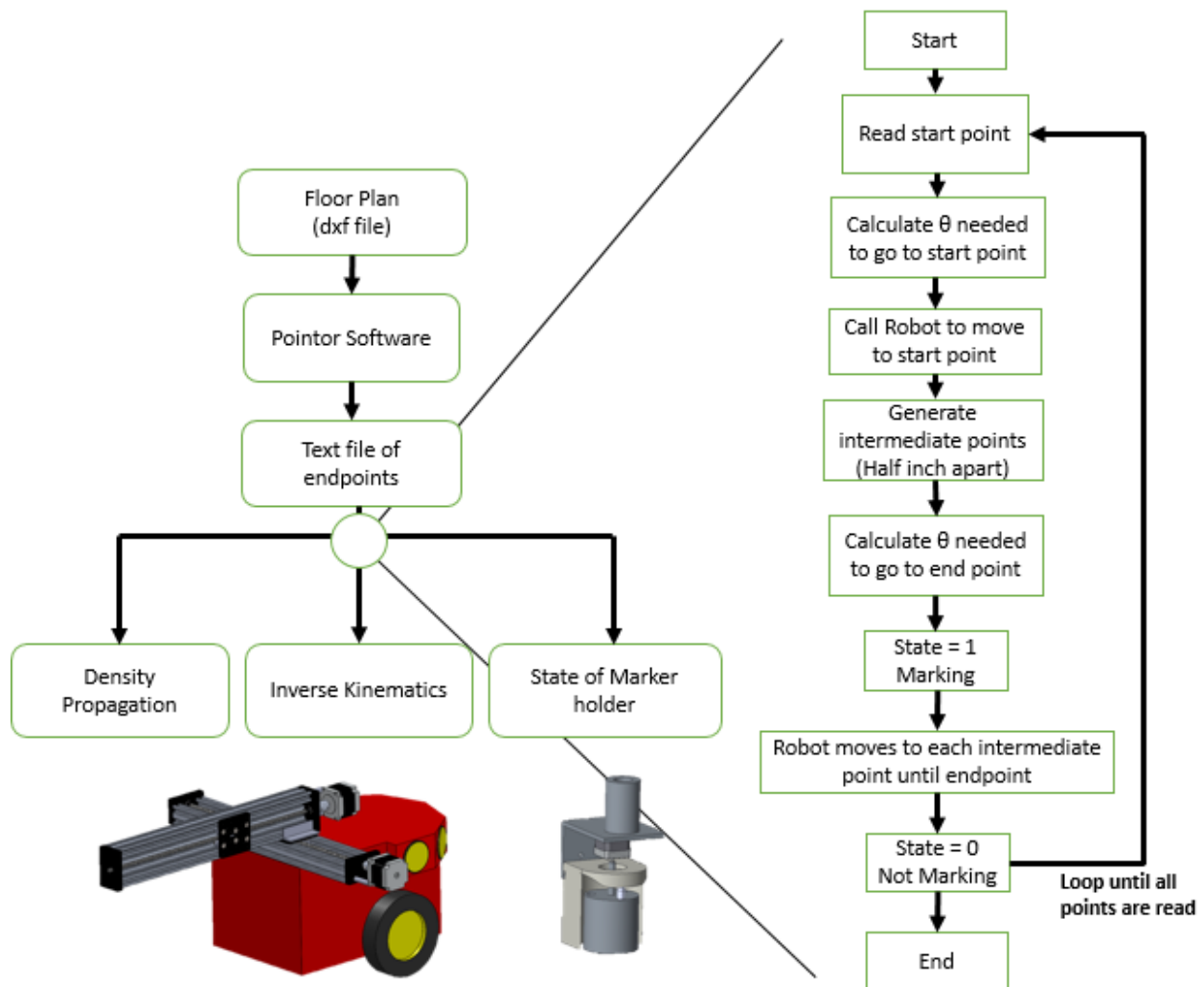


Figure 42: Final Program Flow Chart

This flowchart includes the programming flow that sends the proper information to the Pioneer 2 to call its movement function properly and controls when the marker holder pushes the marker to mark or not.

## 5. Design of Experiment

The group conducted a few experiments in order to test the robot's operation at different points in the project. The first goal was to determine how many steps of the NEMA 23 stepper motor was required in order to move the mounting plate on the gantry system by 1 cm. The current state of the gantry system at the time was a single linear actuator mounted on the robot and the group wanted to demonstrate control over the motor and to ensure that the linear guide rail was assembled correctly. Another goal for the gantry system was to be able to mark out different symbols. At this point the gantry system was completely assembled and the team wanted to demonstrate control over moving both linear guide rails at the same time. An important goal of the project was to be able to read the location data from the Robotic Total Station using the 2.4 GHz external radio and then send that data to the Raspberry Pi 2.

From these tests, it was determined that it took about 2600 steps to shift the mounting plate on the gantry system by 1cm. The team was able to fully control the single linear actuator, so with confidence the group moved forward in completing the gantry system. With regard to moving both linear guide rails at the same time, the team was able to control the gantry system in order to generate several basic shapes, such as a square, triangle, and diamond. During this testing, a squeaking sound was discovered as the top linear mounting plate moved toward the robot. In order to fix this, some WD-40 was applied on the rail. As for the radio, our team used an application provided by Trimble in order to control the Robotic Total Station, however we were unable to streamline the tracking data needed. An attempt was made at creating a new application to do so, however it has not yet been completed.

## 6. Considerations for Environment, Safety, and Ethics

When considering the environment and safety in the project, the location of use was the most important thing to base our decisions on. Since a construction site is full of hazards, the group wanted the final product to be both safe to use as well as not harm the environment it may come in contact since it will be used while the site is still open to the outdoors. The main concern with our project in terms of safety and the environment was the electronics and the LiPo batteries. In order to ensure the safety of the user, the group was careful to contain all open electrical wires and components. In regards to the LiPo batteries, one concern was the charging. It's important to use a LiPo compatible charger using Constant Current/Constant Voltage charging or damage to the battery may occur. The other issue to consider with LiPo batteries is that they can explode or catch fire if the battery is exposed to heat or it is punctured. To combat these problems the battery needs to be placed in a fireproof container, such as a LiPo bag, and also handled with care. On the construction site the user must wear a hardhat, have closed-toed shoes, and long pants. Finally, the weight of the robot might present a small risk, so it should be transported by two people to ensure safety to the person transporting.



## 7. Project Management

### 7.1 Schedule

Following the prescribed Gantt chart plan for completing the project was one of the greatest struggles for the group. One of the biggest problems encountered was determining how long individual tasks would take to complete. Because tasks always took longer than expected deadlines were continually pushed back causing a rush to complete the project at the end. There was also a severe lack of time management in the group when it came to balancing the project with other priorities, such as other classes and jobs. Many tasks were completed late because the group procrastinated on starting them.

Starting at about halfway through the project the Gantt chart started not to be followed and tasks were made for the week at the weekly group meetings on Sunday. Breaking up the tasks into what needed to be done each week greatly helped the group in accomplishing tasks. If the project was redone the deadlines from the Gantt chart would be more strictly followed and more time would have been put in at the beginning of the project. With stricter deadlines in the beginning, the team would have had time to complete the gantry with better results, had more time to troubleshoot communication between the subsystems, and had time to test the project as a whole on a larger scale.

### 7.2 Resources

The resources available to the group were a machine shop, machinists, a supportive sponsor, and a faculty advisor, Dr. Nikhil Gupta. The group used Dr. Gupta to the fullest extent through almost weekly meetings and routine communication to get input. Dr. Gupta was crucial in developing the final design of the gantry system, selecting components in the electrical side of the project, and working with the group weekly to help combine the subsystems of the project. The group's sponsor, Mark Winger, was also a great resource with educating the group on the needs of the construction industry and providing a tour of the construction site he is currently supervising. Due to the set-up of final design of the gantry system and the time it took to settle on the design the group didn't use the machine shop. It was determined to be easier to order the linear guide rails as

a whole to ensure they fit together perfectly instead of having the machine shop create the parts from scratch and hoping for a correct fit.

If the project was repeated, the group should have started meeting with Dr. Gupta earlier. With more constant contact in the beginning of the project there would have been less time wasted with designs that clearly wouldn't have worked, and more time to perfect the final design, as well as more time left over for the end stages. The group also could have benefited from the 3D printers the school had had they created the marker holder earlier in the project instead of having to spend budget money on outsourcing to ensure it was completed in time.

### 7.3 Procurement

The budget for the team was \$2,500. The entirety of the gantry system for the construction marking robot cost \$1,587.59, which comes out to be a little over 60% of the total budget. The majority of this cost is comprised in the linear guide rails and the batteries. The linear guide rails cost \$124.45 each, with a total of \$248.90 for the two necessary to construct the gantry system. The three LiPo batteries purchased cost \$199.99 each, with a total of \$599.97, and the charger for the batteries cost \$119.99. Figure 43 and Table 3 show the complete breakdown of the budget and the cost of individual parts in the prototype, and Table 4 is the bill of materials for the project with the price of each part included.

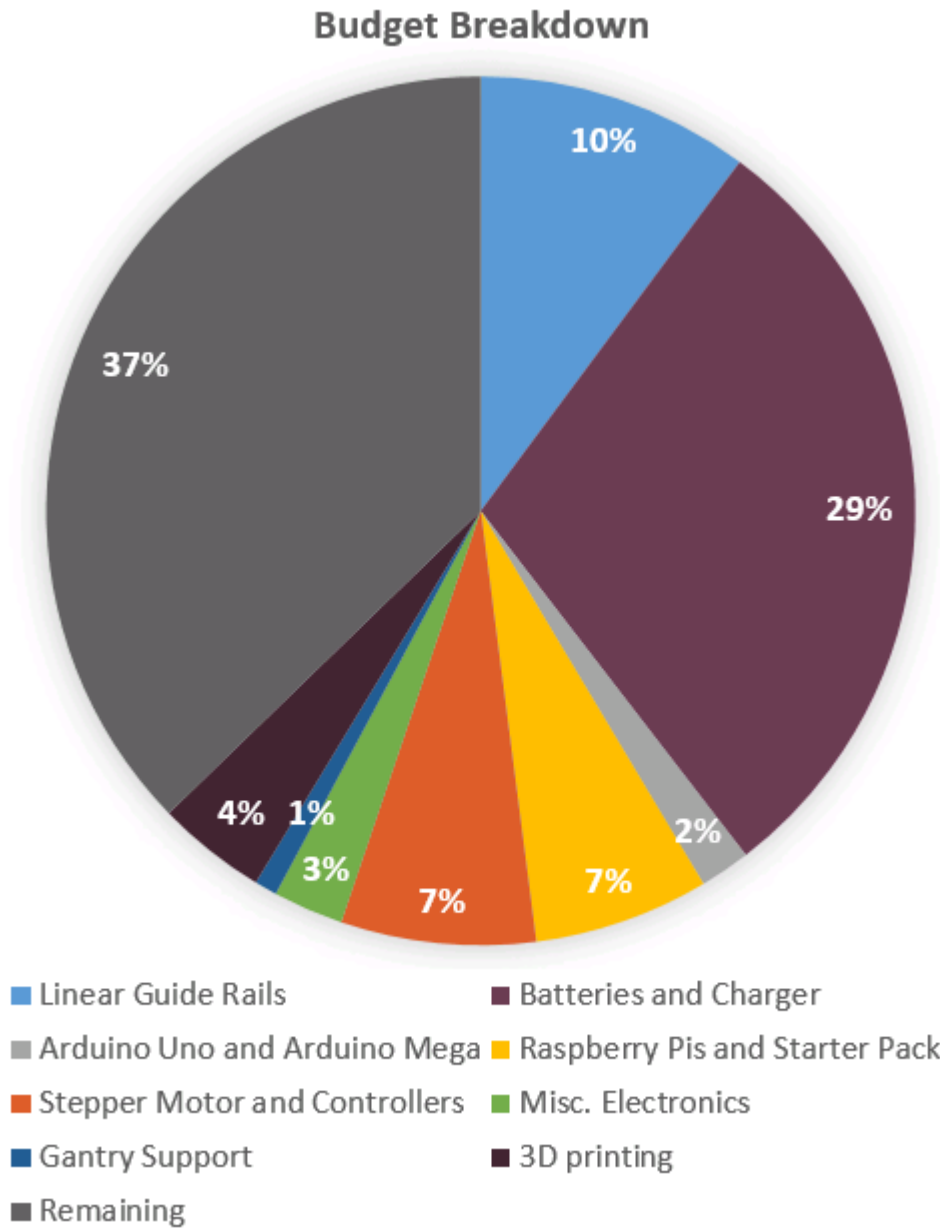


Figure 43: Budget allocation by subsystem

Table 4: Price of parts by subsystem

Part	Cost
Linear Guide Rails	\$248.90
Batteries and Charger	\$719.96
Arduino Uno and Arduino Mega	\$45.96
Raspberry Pis and Starter Pack	\$159.81
Stepper Motor and Controllers	\$177.61
Misc. Electronics	\$64.11
Gantry Support	\$20.59
3D printing	\$100.00
<b>Total</b>	<b>\$1,587.59</b>
<b>Remaining</b>	<b>\$912.41</b>

Table 3: Bill of Materials

Part	Quantity	Cost
Arduino Uno	1	\$24.51
Rasp Pi Starter Pack	1	\$59.95
Rasp Pi	2	\$99.86
Stepper Motor HAT	3	\$67.50
Linear Guide Rails	2	\$248.90
LiPo Charger	1	\$119.99
LiPo Batteries	3	\$599.97
Arduino Mega	1	\$21.45
Nema 17	1	\$50.05
Voltage Regulator	2	\$31.90
Stepper Motor Controller	2	\$60.06
3D Printing	1	\$100.00
T-Nut Aluminum Extrusions	1	\$11.99
Black Angle Corner Connector	4	\$24.95
Xtreme Mini V Wheel	1	\$3.15
Xtreme Solid V Wheel Kit	2	\$13.50
Low Profile Screws M5	8	\$6.45
Aluminum Spacers	2	\$0.40
Eccentric Spacer	1	\$2.00

The budget of \$2,500 was plenty for the gantry system created. However, two major components in the final prototype of the construction marking robot were donated to the project: the robotic total station, provided by Trimble, and the Pioneer 2dx robotic platform, provided by CISCOR. The robotic total station retails at around \$40,000 and the Pioneer 2dx has a base price of \$3,295.

When calculating the total price of the system as a whole these components must be included and the price of the system constructed would be around \$45,000.

## 7.4 Communication

Communication within the group was fine as a whole. Each member responded to Group Me messages and e-mails within a reasonable time and consistently made time for the project. The group chemistry was also good, and each member had a respectful attitude when communicating with the group. Sponsor meetings were arranged to be weekly with the entire group and the sponsor, Mark Winger. In these meetings Mr. Winger was updated on the status of the group and asked relevant questions for the future steps. Mr. Winger was also available at any point by phone or e-mail. Communication between the group and the advisor, Dr. Gupta, was slow at first and got increasingly better. In the beginning stages of the project the group rarely contacted Dr. Gupta. It was soon realized that he was going to be crucial to the completion of the project and soon there was regular e-mailing between the group and Dr. Gupta to set up meetings and ask questions. By the end of the project the group was meeting with Dr. Gupta at least once a week, but most weeks many times. As a whole there were very few communication problems due to attentive group members, a motivated sponsor, and an advisor willing to take the time to ensure the best product possible was created.

## 8. Conclusion

While not all of the original project goals were completely fulfilled, this project was certainly a good step in the right direction towards developing the construction marking robot. Despite issues with organization early on in the process, technical issues with the Pioneer robot, and communication issues with the RTS equipment, the team still managed to develop a fully functioning marking mechanism and holder and mount it to a robot that can take in a text file of coordinates and angles taken from a 2D CAD file of a layout. Additionally, while the obstacle avoidance algorithm was not developed and there were issues for establishing communication between the robot and RTS, the LiDar is already integrated onto the platform for obstacle detection and the prism can be mounted to the marker holder in a location that can be easily tracked by the RTS. For future work, the primary recommendation would be to eventually fully integrate the gantry and mobile robot into a single platform. In terms of improvements that can be made to the current design, using four cell LiPo batteries would be recommended, also, the gantry system could be improved to include a full support structure similar to the one shown in Figure 27; however, the design may need additional supports and possibly a caster wheel attached to the back. In terms of what the team could have done better, there were issues with organization early on in terms of project progression and team structure; because of this, the team lost time early on that could have allowed for more troubleshooting with the remaining parts of the project. Additionally, the electronics should have proper housing and the gantry should have had a more robust support structure to really refine the final product. Aside from the aforementioned improvements to the designs, a significant recommendation that this team would give to any future teams would be to make sure they remain organized and communicate well, both within the team and to the advisor and sponsor, right from the beginning. This team's greatest accomplishments and shortcomings could be ultimately derived this and as such it the most important recommendation to make for the success of continued work on this project by future teams.

## 9. References

1. "Project Lion - A DPR/Trimble Automated Layout Robot." *YouTube*. YouTube, 25 Apr. 2013. Web. 25 Sept. 2015.
2. Prouty, Joseph M. Robotic Construction Site Marking Apparatus. Joseph M. Prouty, assignee. Patent US 20130310971 A1. 21 Nov. 2013. Print.
3. "Construction Industry- Robotic Layout." *Totalmark Technologies*. N.p., n.d. Web. 25 Sept. 2015.
4. "Fire Sprinklers." *Carnation Construction*. N.p., n.d. Web. 05 Dec. 2015. <<http://www.carnationconstruction.com/Techniques/07-03-Techniques-InteriorInfrastructure-FireSprinklers.html>>.
5. "We've Been Framed!" Web log post. *Eightmangana*. N.p., n.d. Web. 05 Dec. 2015. <[http://eightmangana.blogspot.com/2011\\_01\\_01\\_archive.html](http://eightmangana.blogspot.com/2011_01_01_archive.html)>.

# Appendix A

## Importance of Customer Needs Survey

Importance ranking (1-10, with 10 being with the utmost importance)

Serviceability 5

All weather 3

Safety 10 (Can you clarify what you mean by safety? Whose safety? Operates safely? Safety is a big deal in construction. )

Error report 10

Tolerances on error report 10

Durability 7

How many SQFT 100 sq.ft.

Cost of manufacturing 1

Time of operation 10 (100 sq.ft. in 10 minutes)

User friendly 5

Level of autonomy 8

Use of robotic totaling station 10

Marking arm 1

Transportability 8

Ability for more colors 5 What do you mean? Like using an inkjet printhead over vs pen?

Accuracy 10 ½"

Line continuity 7

Different terrains 3

Other uses for robot 4

Sharpie for marking 2



## Appendix B – FMEA

No.	Name	Failure Mode	Cause	Effects	Method of Detection
1	Mounting Bolts	Bolt shears	Too large normal force	Structure can fall apart	If any of the parts fastened together come loose
		Threading strips	Too large axial force	Loosens hold on connected parts or can fall apart completely	Check if any bolts spin freely when a torque is applied by a hex driver
2	Platform Wheels	Cracking	Wheel kit overtightened	Gantry may catch during operation	If gantry is not moving smoothly, check the individual wheels
3	Lead Screw	Bending	Too large normal force applied	Platform cannot move properly	Check to make sure the lead screws are still straight
		Shearing	Even larger normal force applied	Gantry will fail	Check condition of lead screws
4	Stepper Motor	Burning out	Too much current supplied	Stepper motors will no longer function	Check if stepper motor feels overheated
5	Linear Rails	Bending	Too large normal force applied	Platform cannot move properly	Check to make sure linear rails are still straight

6	Motor Drivers	Burning out	Overloading with current	Can't drive stepper motors	Check for irregularities when running motors
---	---------------	-------------	--------------------------	----------------------------	--

# Appendix C - Design for Manufacturing and Reliability

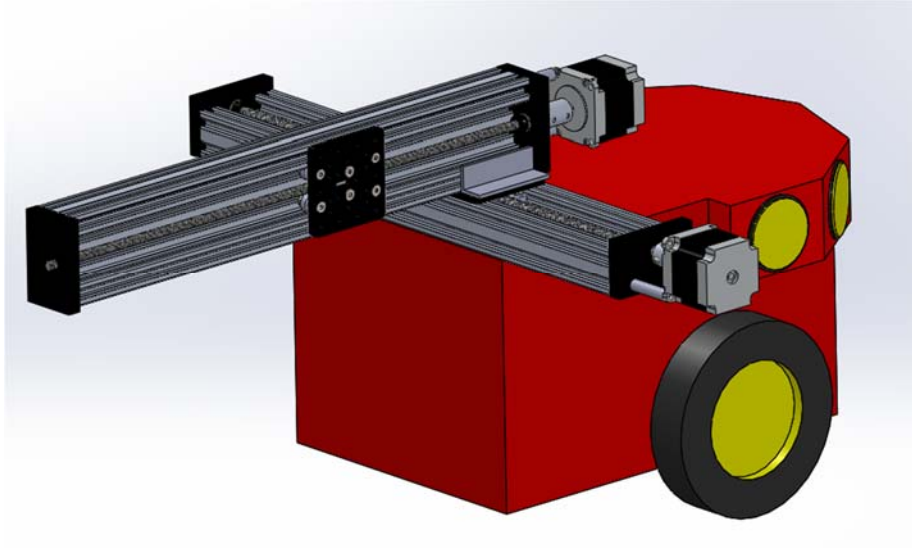
## 1.1 Assembly Time

The construction marking robot's assembly process first began with the Pioneer 2-DX robot which was provided by CISCOR. Since this robot was provided preassembled with the LiDar mounted, this portion of the assembly time can be considered virtually zero. For the gantry, the team order two C-Beam linear actuator bundles from Open Builds which included the extruded rails, lead screw, stepper motor, mounting platform, and all necessary spacers, bolts, and nuts. While the bundle included all of the parts necessary to construct the linear actuators, assembly instructions were not included, so it took about 4 hours to fully assemble both linear actuators after receiving the parts. Most of that time was due to having to figure out how to assemble the first actuator; most likely it would take about an hour at most to assemble each actuator for someone who already knows how to assemble it. In order to complete the gantry, the two linear actuators needed to be mounted together and then to the robot. In order to do this, brackets were first machined from angle iron and then attached to the platform of the first linear actuator and to the t-slot nuts inserted into the rails of the second actuator. After this, another set of larger brackets were fabricated for mounting the gantry to the robot; four holes were made in the robot's mounting platform for the mounting points of the bracket with the other end of the brackets being bolted to t-slot nuts in the side of the first linear actuator. This whole process took about six hours including the machining of the brackets. After testing the gantry, another two hours were given to slight modifications such as milling and filing down edges of the extruded rails that were causing parts to scrape and therefore not move as smoothly. This assembly was decently shorter than originally anticipated, but this is mainly due to design changes made with the intent to speed up the process and due to having direct access to the machine shop rather than having to wait for it to be manufactured.

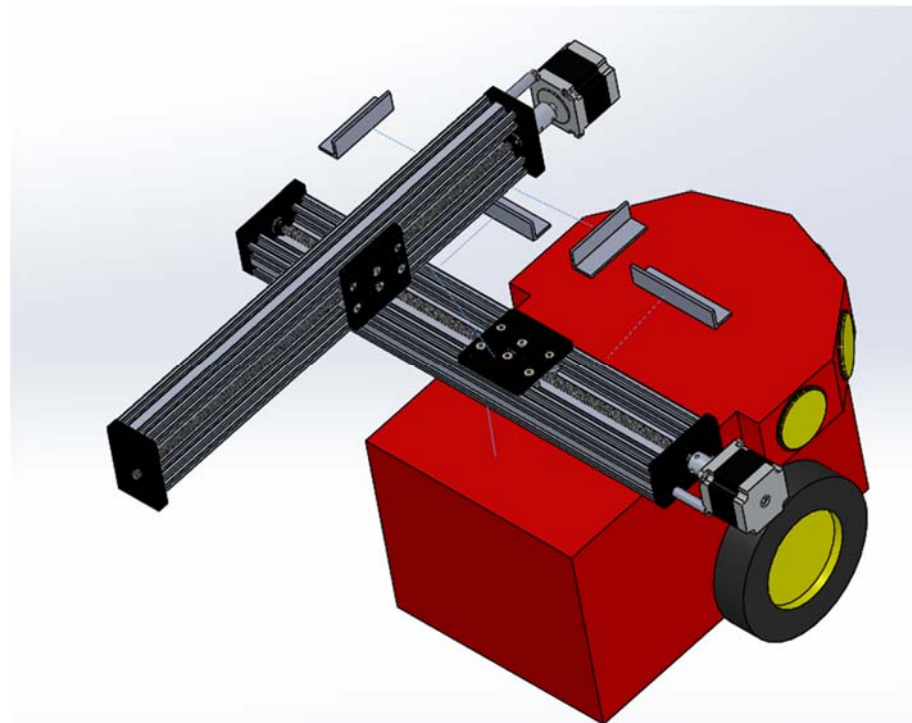
The marker holder took around 5 hours total to create on Creo Parametric. It then took a little over 30 hours to be printed from a 3D printer.

## 1.2 Components

Counting the Pioneer 2-DX with the LiDar as a single part, since it was provided and is therefore not itself assembled part of the assembly process, there are a total of 149 parts that make up the prototype, not including the robotic total station. Each linear actuator bundle contains 42 parts including every bolt, washer, and nut, a total of four pieces of angle iron along with 4 pieces of 1/16<sup>th</sup> inch aluminum sheet were used as mounting brackets for the gantry, and 16 bolts, 8 t-slot nuts, and 4 lock nuts were used to fasten the gantry assembly together and to the robot. This design cannot really be simplified into fewer components, but it could be modified so the linear actuators were lighter. However, for future work, the design's complexity should be increased to guarantee more accurate results. While the current version of the prototype can be assembled much faster which leads to more time for testing the other crucial components, additional error arises due to the fact that the second linear actuator is effectively hanging off the end of the robot. To rectify this, it would be appropriate and necessary to add a support system for the actuator, most likely out of extruded rails, a passive linear component such as a shaft with a bearing to support the moving end, and a caster wheel to support the overall structure. Done correctly, this would limit the tilt in the second actuator as well as the unnecessary movement the system experiences while moving. The marker holder is comprised of 6 parts: the outer casing, the revolver, the stepper motor, and three sharpie markers.



**Assembled View**



**Exploded View**

### 1.3 Design for Reliability

Reliability has become more important in the recent century than it ever has before. Now that mechanical, electrical, and computer systems are more prevalent, designing for reliability has been incorporated into the design process in order to save time and money, and increase safety. There

are many different types of reliability ranging from mechanical and software to human and life cycle reliability. All the different types of reliability have certain methods to follow to optimize the product or service in respect to reliability. A few of these methods were utilized in the design of the construction marking robot.

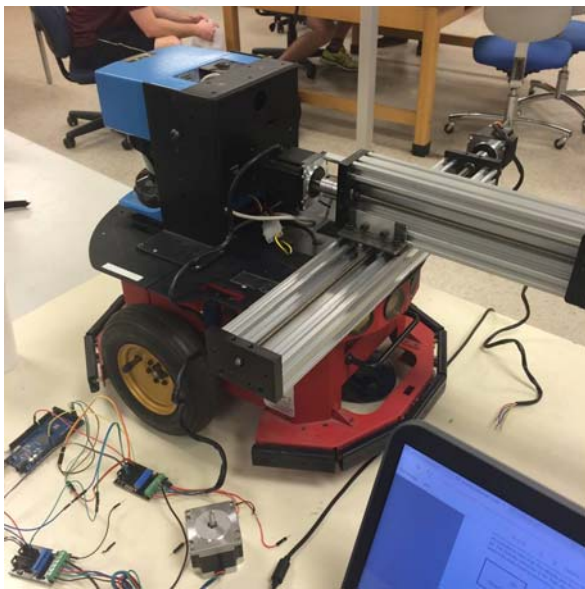
In order for a product to be worthy it must hit the right market at the right time with the right price. Even if all of these criteria are met, if the product or service is unreliable, it will fail. The process required to make a product or service reliable is very iterative in nature, first conceptually and then experimentally. The design process is the most important step of the product life cycle as it rules how the rest of the product life will live out. Throughout the design process the team held multiple meetings to ensure that everyone was on the same page and agreed on the movement of the project. There were also major meetings held with advisors and sponsors in order to ensure that the team was moving in a customer desired direction as well as are approaching problems in the correct manner to satisfy the customer and sponsor simultaneously.

The prototype developed by the team is nearly done and some testing has already taken place. The testing has already identified a few reliability concerns to the team. The construction marking robot is only a proof of concept and was designed to help identify some major conceptual flaws with the idea of marking construction lines using an autonomous robot. There have only been a few tests done thus far and provided promising results as to a successful final product or service. The more test that will be run, the more data and information can be processed by the team to ensure success.

One of the major components that would be affected by multiple uses would be the linear guide rails. Due to their nature of moving a platform using a screw driven system, the parts are expected to wear. The construction site is not a clean place and the dirty/hazardous environment will not help contribute to the longevity of the product. One solution to extending the life of the linear guide rails would be to add dust covers and covers over the bearings. This would help prevent dirt and debris from accelerating the wear process. The longer the life of the product, the more profitable the product or service is. The dust covers and bearings are more of an initial cost and it would be the design team's job to make the decision whether or not the longevity is worth the extra cost.

Due to the nature of the construction marking robot project being a very integrative project, many things needed to be customized. Customization is not cheap but allows room for the design team to implement their own level of reliability into the design of the product. One major thing focused on by the team was the ease of use. The easier the product is to use, the less training is needed to use the product or service. More complexity went into the design to allow for a seamless interaction with the customer and the robot. This seamless interaction would seem more appealing to the customer if a competitor would appear in the market.

One situation in which designing for reliability was considered greatly by the team was the



**Construction Marking Robot**

mounting of the gantry system onto the robotic platform. There were multiple ways in which the gantry could have been mounted. Some methods included castor wheels while other methods were more balanced and did not need the extra support/parts of the castor wheel. The team met with a fellow student to bring in his expertise of mounting to the team's design. The student decided that in this case, simpler is better. Instead of designing multiple mounting brackets and extra guide rails, the team directly mounted the gantry onto the robot as seen in Figure 1. This method of mounting the gantry to the robotic platform is not

the most reliable method for joining the two but with time and money constraints the team did what was necessary and correct in the situation at hand. FEA was done in order to ensure that the gantry could support the weight and forces that maybe experienced while on the job. The mounting style seen here will see a lot of vibrations which can speed up the wear process as well as corrupt electronics mounted on board. The vibrations could also cause some of the marks that are being made to be off target and therefore cause error in the marking which is what the entire project is aimed at minimizing.

Some other focus points for the team are things such as robot reliability, human error, and maintenance. The robot must be reliable because there could be the possibility of humans being

on the construction site during the operation. The robot weighs nearly 50 pounds and could easily hurt someone if it were to run into someone's ankle and or fall from a structure onto a hard hat. Human error could affect the way in which the lines are to be laid out. If the robot and robotic total station are not set up correctly, the lines which are drawn on the ground could be off and/ or not accurate. Without these accurate lines, the contractors would not be able to finish the construction project safely. Walls would not be in the correct places and some plumbing maybe hanging outside of walls or even ceilings. Maintenance would be key in many aspects ranging from the cleanliness of the construction site to the up keep and cleanliness of the robot itself. It would be within the training of the operator as to the cleanliness standards for the robot as well as the construction site.

Being that the construction marking robot project will be continued for another year, this year's prototype will give major insight into some problems that maybe encountered in the future but are unforeseen. By following the design process and keeping the reliability of the product in mind, the design team has the potential to create a great product that could revolutionize the construction industry and the way in which contractor communicate with each other as well as save time and money while doing it. This introduction of new technology into the construction industry will build a better relationship between it and the people in the field who will be using it.

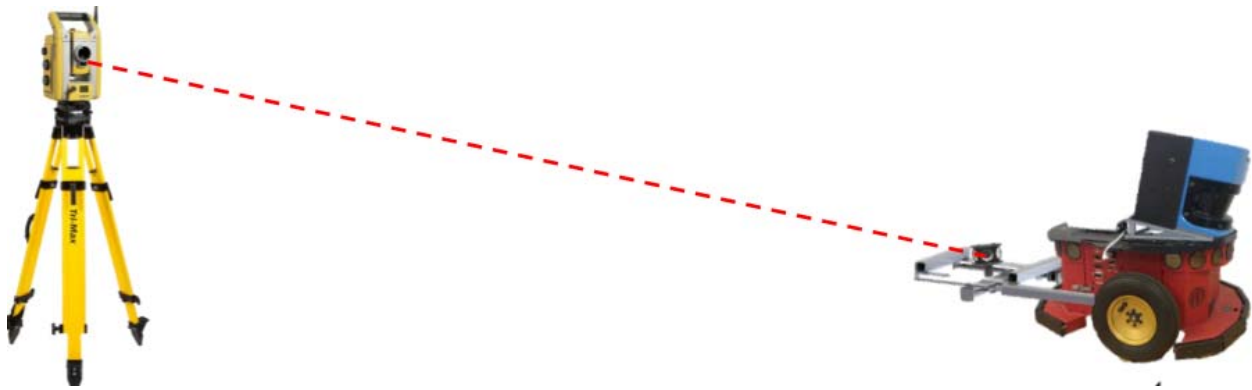


## Appendix D – Operations Manuel

### Functional Analysis

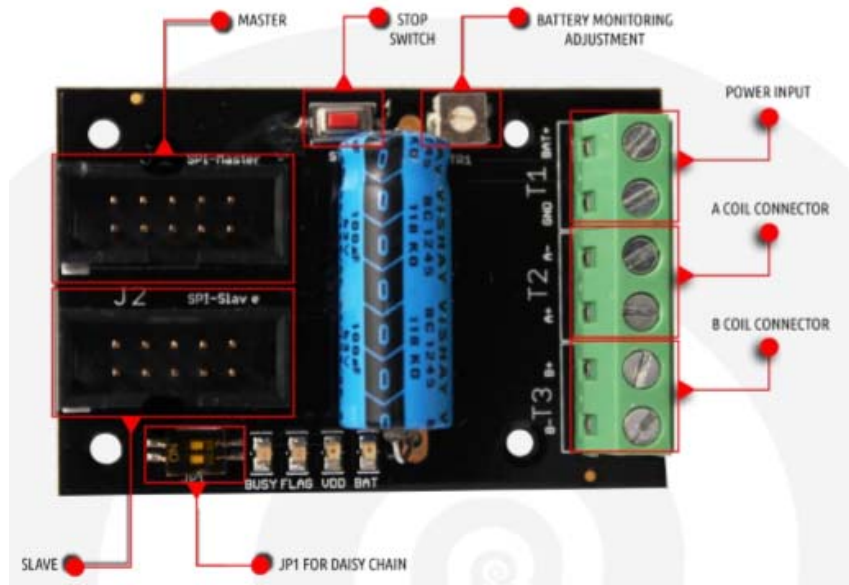
The function of this project is to design an accurate robotic marking system that will read in a floor plan and print it out on concrete at full scale. The project works by first taking the floor plan as a CAD file of type dxf and running this file through the Pointor program created by Ransen Software, this will be run on the Raspberry Pi 2. This software will analyze the floorplan file, and determine all of the individual lines that make up the floor plan. From there, the program will output a text file containing the endpoints of all the lines in the floorplan. This text file of endpoints will then be read into a density propagation program that will take the endpoints and add intermediate coordinate points along the line that are half inch apart. This program will also include a bit for the current marking status, as well as calculate the angle (theta) necessary for the robot to turn in order to move to the next point. The Pioneer 2-DX, a differentially steered robot the team is using was donated for use by the CISCOR lab. This robot will read in the updated text file and pass the velocity, linear velocity (which are both constant) and the theta as parameters for the Pioneer's movement function; this will be done by the Raspberry Pi 2 calling Pioneer's function stored on its own operating system and will be physically connected to it via an Ethernet cable. Attached to the Pioneer is the gantry system designed by the team. The gantry consists of two linear actuators run by two NEMA 23 stepper motors that are controlled by SmartLynx bipolar motor drivers. The code controlling the stepper motor drivers is written on the Arduino UNO microcontroller. The marker holder is connected to the gantry system and run by a NEMA 17 stepper motor; it will be able to switch between a few different markers in order to vary the marking color. The marking holder will also have a standby position which will be used when the robot is not marking. A tracking prism will be mounted on top of the linear guide rail and will be tracked by Trimble's robot total station. The total station will be controlled by Trimble's 2.4GHz external radio and will transmit the prism's location back to the external radio which is connected to the Raspberry Pi 2 on the robot. If the prism's location does not exactly match the text file of coordinates as the robot moves, the difference in distance will be sent from the Raspberry Pi 2 to the Arduino UNO board's linear guide rail movement function. This function will convert the distance to stepper motor steps so that the marker will shift in order to compensate for any accuracy

errors. As the robot moves from point to point, the position of the prism will constantly be sent to the Raspberry Pi 2, so that the marker placement can constantly be adjusting to ensure at least half inch accuracy of the printed floorplan. Instead of having to manually set the gantry's platforms to the center, the raspberry pi will send a soft-stop to the stepper motors to move the platforms on the X and Y axis to the center to prepare for marking. The marking robot will have a LiDar LMS200 (Laser Measurement System) used for object detection. The LiDar will have a function that will be called by the Raspberry Pi to check to see if any obstacles are range and close enough to knock the robot off its course. The LiDar will be mounted on an angled slope to ensure that the LiDar can see shorter objects as the LiDar only has a horizontal movement and not vertical. The mount give the LiDar a 5 foot range before it reaches the ground.

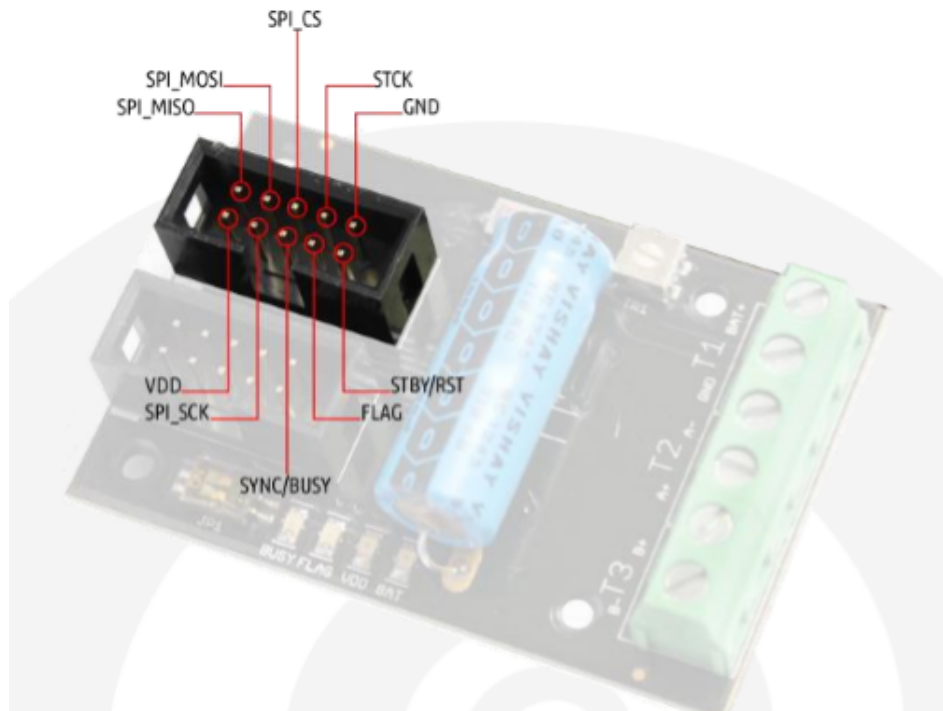


*RTS tracking Pioneer*

# Project Specification



*SmartLynx motor driver – Top view*

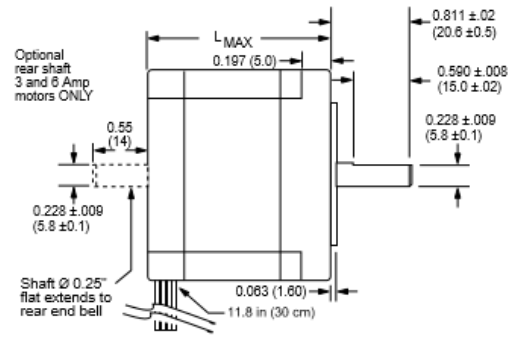


*SmartLynx J1 header pin layout*

**NEMA size 23 1.8°  
2-phase stepper motor**



**Mechanical Specifications**  
Dimensions in inches (mm)



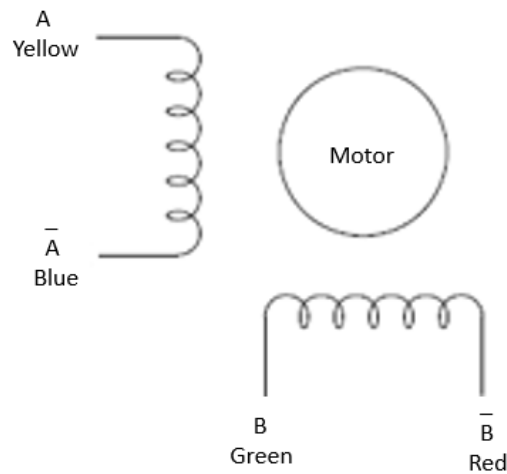
*NEMA 23 Mechanical Specifications*

**Specifications**

2.4 Amp motors		Single length	Double length	Triple length
Part number		M-2218-2.4 S (1)	M-2222-2.4 S (1)	M-2231-2.4 S (1)
Holding torque	oz-in	90	144	239
	N-cm	64	102	169
Detent torque	oz-in	3.9	5.6	9.7
	N-cm	2.7	3.9	6.9
Rotor inertia	oz-in-sec <sup>2</sup>	0.00255	0.00368	0.0065
	kg-cm <sup>2</sup>	0.18	0.26	0.468
Weight	oz	16.9	21.2	35.3
	grams	480	600	1000
Phase current	amps	2.4	2.4	2.4
Phase resistance	ohms	0.95	1.2	1.5
Phase inductance	mH	2.4	4.0	5.4

(1) Only available with single shaft.

*NEMA 23 Technical Specifications*

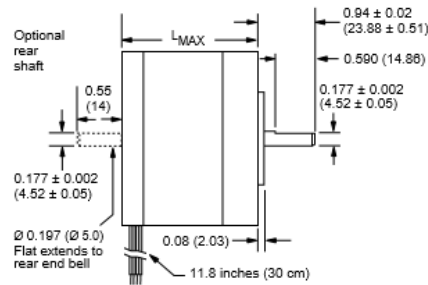


*NEMA 23 Wiring Diagram*

**NEMA size 17 1.8°  
2-phase stepper motor**



**Mechanical Specifications**  
Dimensions in inches (mm)



*NEMA 17 Mechanical Specifications*

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

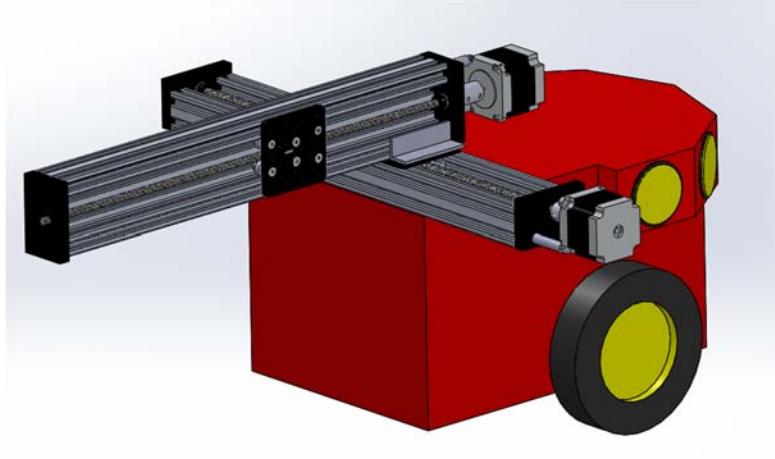
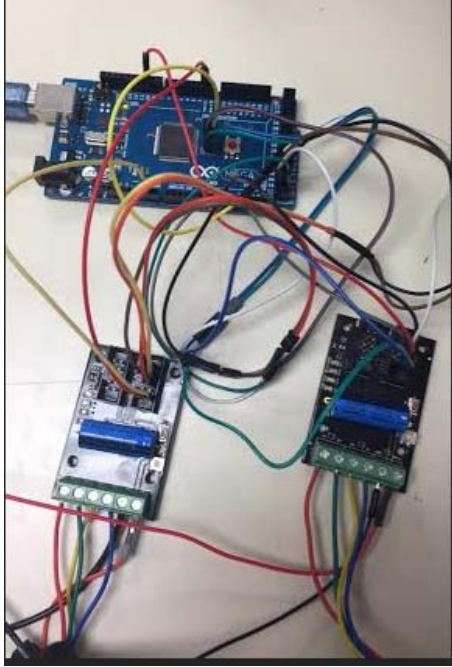
*Arduino Mega2560 Technical Specifications*

**Technical Specifications:**

- Broadcom BCM2836 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz
- 1GB RAM
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

*Raspberry Pi 2 Model B Technical Specifications*

## Project Assembly



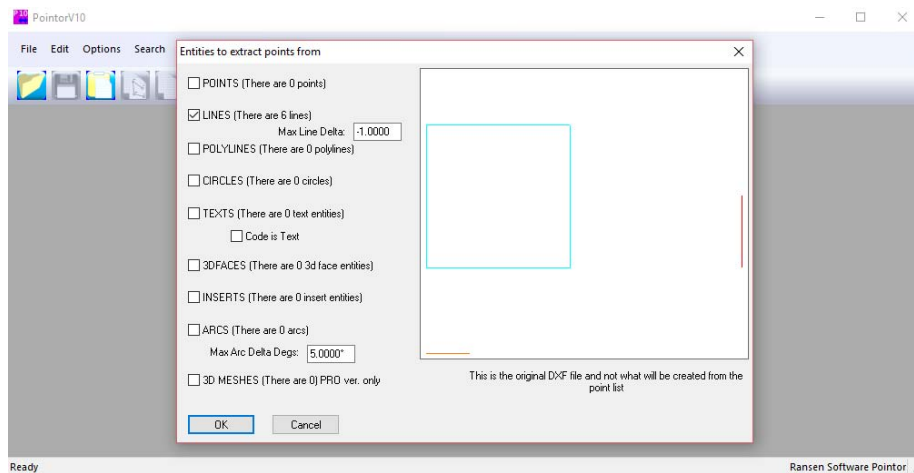
*Circuit for SmartLynx Motor drivers connected to Arduino Mega2560 with 3D gantry system  
SmartLynx Motor drivers to Arduino Mega2560 Pin Connections*

<b>SmartLynx pin</b>	MOSI	MISO	SCK	SS	Reset
<b>Arduino Mega pin</b>	51	50	52	53	1
<b>Arduino Mega pin (2<sup>nd</sup>driver)</b>	ICSP-4	ICSP-1	ICSP-3	10	ICSP-5

# Operation Instruction

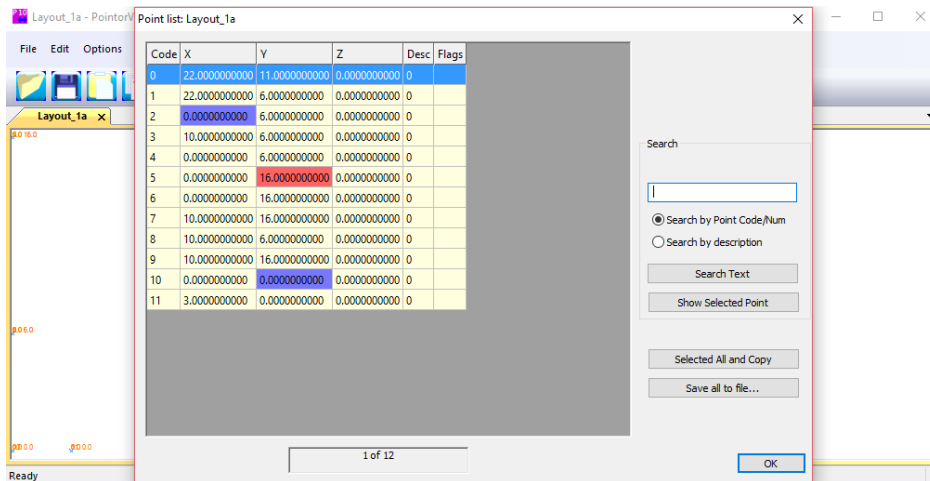
## Software Instructions

To convert a dxf file into a txt file of coordinates, you first run the Ransen Software program Pointor. You select file > Open DXF file... and then select your CAD file. Once you open the desired file, the program will analyze the line types and their amounts based on the file.



*Ransen Software Pointor's DXF analysis*

After clicking OK, you will see a screen of coordinate points where the line endpoints were. Next click on the View Pointlist icon to see the list of coordinate points relative to the dxf file.



*Ransen Software Pointor's Pointlist*

Click Save all to file... for the format select Code X Y, select comma <,> for Decimal Separator, and for the Delimiter select space. Type in 1 for “Start writing from row”, then select OK and choose a name for your file.

Next compile reader.cpp with the command “g++ reader.cpp” this will create an executable file a.out. We then pass the txt file of coordinates through the executable using the command “a.out < Layout\_1a.TXT > output.txt” this will create a new txt file output.txt while will be sent to the robot in order to properly call its movement function. (Note: Layout\_1a.TXT is what I called the coordinate file created by the Pointor software, you will enter in the name of your file instead).

In order to control the gantry system download the Arduino software and open the NEMA23 Arduino file created by the team. This code will run a function that takes in an integer to represent the distance and will turn the stepper motor appropriately in order to shift the marker holder as needed according to prism location.

To use the Trimble external radio, connect it to a USB to Serial converter and plug it in a USB port. Run the C# code on a .NET framework and change the Port on your device to match the com port in the code if needed. So far the code will turn on and off the radio. Sending a break for 100ms will turn the radio on, sending a break for 1500ms will turn the radio off. Refer to Trimble’s documentation on the 2.4 GHz external radio for more information.

## Connecting Raspberry Pi to Arduino via I2C

While using Raspbian, Go to Terminal and type following commands to download the proper libraries used to connect the two:

- `sudo apt-get install python-smbus`
- `sudo apt-get install i2c-tools`

Now you must install I2C support and kernel. Type:

- `Sudo raspi-config`

A screen should pop up. Go to 8 Advance Options, then go to A7 I2C. Click yes to enable the ARM I2C interface, and yes to the I2C kernel module to be loaded by default. Reboot Raspberry Pi. After reboot, go back to terminal and type:



- `sudo nano /etc/modules`

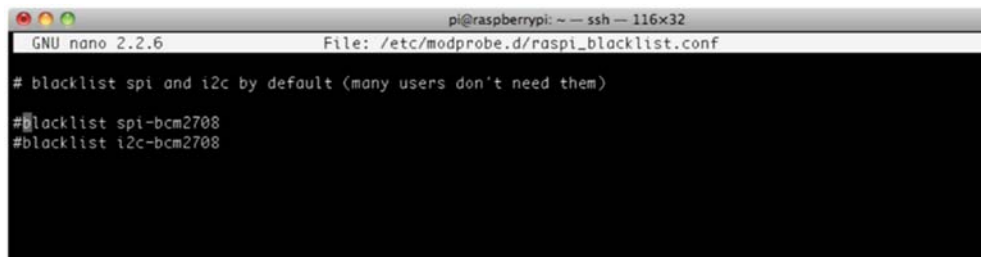
If file exists, go to the very end of the file and comment out these lines by putting # in front:

- `i2c-bcm2708`
- `i2c-dev`

Exit the file (CTRL-X and Y) then type:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Add the lines:



```
pi@raspberrypi: ~ -- ssh -- 116x32
GNU nano 2.2.6 File: /etc/modprobe.d/raspi_blacklist.conf
# blacklist spi and i2c by default (many users don't need them)
#blacklist spi-bcm2708
#blacklist i2c-bcm2708
```

*Figure 12: Raspberry Pi 2 Blacklist code*

Afterwards the configuration file needs to be updated. Add:

- `sudo nano /boot/config.txt`

Add the following lines to the bottom of the text file:

- `dtoverlay=i2c1=on`
- `dtoverlay=i2c_arm=on`

\*note that the '1' in 'i2c1' is a one not an L.\*

You are finished if completed correctly, not you have to reboot:

- `sudo reboot`

After reboot go to terminal and type:

- `i2cdetect -y 1`

This pulls up the address block of the Raspberry Pi. This is how you check to see if the Arduino is sending its address to the Raspberry.

## Hardware Instructions

In addition to the software component of the robotic total station, it will need to be physically set up and leveled.

In order to control the NEMA23 stepper motor you will need six male to female jumper wires, and two male to male jumper wires for each board. The connection is as follows:

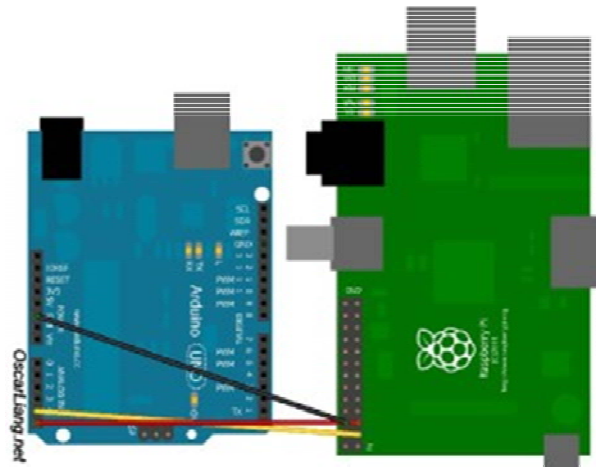
*SmartLynx Motor drivers to Arduino Mega2560 Pin Connections*

<b>SmartLynx pin</b>	MOSI	MISO	SCK	SS	Reset
<b>Arduino Mega pin</b>	51	50	52	53	1
<b>Arduino Mega pin (2<sup>nd</sup>driver)</b>	ICSP-4	ICSP-1	ICSP-3	10	ICSP-5

The stepper motor wires will be connected according to the NEMA23 wiring diagram. Be sure to connect the SmartLynx ground pin to another ground pin on the Arduino board. On the Smartlynx you will connect the power supply's positive to Battery, and the negative to the ground wire slot next to Battery.

The external radio is connected to a USB port via a Serial to USB converter. Ultimately this should be connected to the USB port on the Raspberry Pi 2. The Raspberry Pi 2 will be plugged into the Pioneer's operating system via an Ethernet cable, and will be connected to the Arduino board via I2C, which allows the Raspberry Pi to send values that will be used in functions on the Arduino's code.

The Raspberry has built in pull-up resistors on the GPIO pins, but they are only available on pins 3 and 5 or the GPIO0 and GPIO1, they are called the SDA and the SCL respectively. The Arduino Uno's I2C pins are A4(SDA) and A5(SCL); Mega's are 20(SDA) and 21(SCL)



*Raspberry Pi 2 to Arduino Connection*

## Troubleshooting

Some of the current issues with the project include being able to fully utilize Trimble's external radio in order to operate the robot total station so that it can track the prism. The documentation on the radio includes some pseudocode as well as the functions the radio contains, but actually accessing the functions with code is not given and is proprietary to Trimble. An application will have to be developed in order to use the radio, our group can currently turn the radio off and on but this will have to be added to. In addition to this, the radio's hardware seemed very inconsistent. For example, the radio would occasionally not respond to the same code being run under the same parameters.

Another issue we had is to utilize the Pointor software on the Raspberry Pi 2 because it is an exe file and the Pi operates on a Raspbian operating system. Our group downloaded Windows IoT operating system on the Pi in order to run the Pointor software that way, however we found it was very limiting and took away much of the Pi's functionality so Raspbian was redownloaded instead.

## Regular Maintenance

The LiPo batteries will have to regularly charged, as well as Trimble's tablet, Trimble's external 2.4 GHz radio, and the robotic total station's battery. The LiPo batteries must be at the same charge, preferably fully charged. The batteries try to balance out their voltage level, and if they are voltage levels are at least close, the batteries will short. If batteries begin to swell, discontinue using immediately.

## Spare Parts

- Arduino Mega – 1
- SmartLynx Stepper Motor Driver - 2
- Jumper Wires – MM MF FF – 10 each
- LiPo Battery – 1
- Markers – 2 of each color
- Raspberry Pi 2 – 1
- Flat-head Screwdriver – 1
- Stepper Motors - 2

## Appendix E – Specifications Sheets

### Spec for NEMA 23

#### Dimensions

Size: 56.4 mm square × 56 mm<sup>1</sup>

NEMA size: 23

Weight: 700 g

Shaft diameter: 6.35 mm

#### General specifications

Shaft type: 1/4 inch "D"

Steps per revolution: 200

Current rating: 2800 mA<sup>2</sup>

Voltage rating: 2.5 V

Holding torque: 180 oz·in

Resistance: 0.9 Ohm<sup>2</sup>

Inductance per phase: 2.5 mH

Number of leads: 4

Lead length: 30 cm

## Specs for SmartLynx

Supports up to 128 micro steps per full step

SPI interface for communication with microcontroller

On-board 32 MHz crystal oscillator

Programmable speed profile and positioning

Two level over temperature protection

5 bit ADC for battery voltage monitoring

Stall detection

### Specifications

L6470 package: POWERSO36

Motor supply voltage: 8V to 45V

Maximum output current: 7

## Specs for Arduino Mega

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

## **Raspberry Pi Specs**

- Broadcom BCM2836 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz
- 1GB RAM
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

## Appendix F – Operation Codes

```
#include <SPI.h>
#include <Wire.h>

#define SLAVE_ADDRESS 0x04
#define SPI_PORT PORTB //this is correct
#define MOSI_PIN 51 //I corrected this
#define SCK_PIN 52 //I corrected this
#define MISO_PIN 50 //I corrected this
#define CS_PIN1 53

#define SPI_PORT2 ICSP //this is correct
#define MOSI_PIN2 ICSP-4 //I corrected this
#define SCK_PIN2 ICSP-3 //I corrected this
#define MISO_PIN2 ICSP-1 //I corrected this

#define CS_PIN2 10

// SmartLynx Reset pin
#define RESET_PORT PORTE
#define RESET_PIN 1

#define RESET_PORT2 ICSP
#define RESET_PIN2 5
int counter = 0;
const int SPI_CS1 = 53; //SPI cable select pin
const int SPI_CS2 = 10;
const int STEP_PIN1 = 50; //pin for single stepping
const int STEP_PIN2 = 50;
int number = 0;
int state = 0;
// smartlynx register addresses
const byte NOP = 0b00000000;

const int SMARTLYNX_ABS_POS = 0x01;
const int SMARTLYNX_SPEED = 0x04;
const int SMARTLYNX_ACC = 0x05;
const int SMARTLYNX_DEC = 0x06;
const int SMARTLYNX_MAX_SPEED = 0x07;
const int SMARTLYNX_MIN_SPEED = 0x08;
const int SMARTLYNX_FS_SPD = 0x15;
const int SMARTLYNX_STEP_MODE = 0x16;
const int SMARTLYNX_STATUS = 0x19;
//smartlynx commands
```



```
const int SMARTLYNX_SET_PARAM = 0x00;
const int SMARTLYNX_GET_PARAM = 0x20;
const int SMARTLYNX_RUN = 0x50;
const int SMARTLYNX_MOVE = 0x40;
const int SMARTLYNX_SOFT_STOP = 0xB0;
const int SMARTLYNX_HARD_STOP = 0xB8;
const int SMARTLYNX_GET_STATUS = 0xD0;
// step size selection
const int SMARTLYNX_STEP_FULL = 1;
const int SMARTLYNX_STEP_HALF = 2;
const int SMARTLYNX_STEP_QUARTER = 3;
const int SMARTLYNX_STEP_ONE_EIGHT = 4;
const int SMARTLYNX_STEP_ONE_16TH = 5;
const int SMARTLYNX_STEP_ONE_32ND = 6;
const int SMARTLYNX_STEP_ONE_64TH = 7;
const int SMARTLYNX_STEP_ONE_128TH = 8;

void setup() {
  Serial.begin(9600);
  // start the SPI library:
  SPI.begin();
  //pinMode(27, INPUT);
  //pinMode(26, OUTPUT);
  pinMode(10, OUTPUT);
  smartlynxSetMaxSpeed(200);
  smartlynxSetMaxSpeed2(200);
  pinMode(13, OUTPUT);
}

void loop() {
  SQUARE();
  TRIANGLE();
  CIRCLE();
  //delay(100);
}

void CIRCLE(){
  while (counter < 100)
  {
    digitalWrite(10, LOW);
    digitalWrite(53, HIGH);
    smartlynxRun2(0, 0);
    delay (100);
    smartlynxRun(0, 150);
    counter = counter + 1;
  }
}
```

```
}
counter = 0;
while (counter < 100)
{
  digitalWrite(10, LOW);
  digitalWrite(53, HIGH);
  smartlynxRun2(0, 150);
  delay (100);
  smartlynxRun(1, 150);
  counter = counter + 1;
}
counter = 0;
while (counter < 100)
{
  digitalWrite(10, LOW);
  digitalWrite(53, HIGH);
  smartlynxRun2(0, 150);
  delay (100);
  smartlynxRun(0, 150);
  counter = counter + 1;
}
counter = 0;
while (counter < 100)
{
  digitalWrite(10, LOW);
  digitalWrite(53, HIGH);
  smartlynxRun2(1, 150);
  delay (100);
  smartlynxRun(0, 150);
  counter = counter + 1;
}
counter = 0;
while (counter < 100)
{
  digitalWrite(10, LOW);
  digitalWrite(53, HIGH);
  smartlynxRun2(1, 150);
  delay (100);
  smartlynxRun(1, 150);
  counter = counter + 1;
}
counter = 0;
while (counter < 100)
{
  digitalWrite(10, LOW);
  digitalWrite(53, HIGH);
```

```
    smartlynxRun2(0, 0);
    delay (100);
    smartlynxRun(1, 150);
    counter = counter + 1;
}
counter = 0;
}

void TRIANGLE(){
    while (counter < 100)
    {
        digitalWrite(10, LOW);
        digitalWrite(53, HIGH);
        smartlynxRun2(0, 300);
        delay (100);
        smartlynxRun(0, 150);
        counter = counter + 1;
    }
    counter = 0;
    while (counter < 100)
    {
        digitalWrite(10, LOW);
        digitalWrite(53, HIGH);
        smartlynxRun2(1, 300);
        delay (100);
        smartlynxRun(0, 150);
        counter = counter + 1;
    }
    counter = 0;
    while (counter < 100)
    {
        digitalWrite(10, LOW);
        digitalWrite(53, HIGH);
        smartlynxRun2(0, 0);
        delay (100);
        smartlynxRun(1, 300);
        counter = counter + 1;
    }
    counter = 0;
}

void SQUARE(){
    while(counter < 100)
    {
        digitalWrite(10, LOW);
        digitalWrite(53, HIGH);
```

```
    smartlynxRun2(0, 300);
    delay (100);
    smartlynxRun(0, 0);
    counter = counter + 1;
}
counter = 0;
while(counter < 100)
{
    digitalWrite(10, LOW);
    digitalWrite(53, HIGH);
    smartlynxRun(0, 300);
    delay(100);
    smartlynxRun2(0, 0);
    counter = counter + 1;
}
counter = 0;
while(counter < 100)
{
    digitalWrite(10, LOW);
    digitalWrite(53, HIGH);
    smartlynxRun2(1, 300);
    delay (100);
    smartlynxRun(0, 0);
    counter = counter + 1;
}
counter = 0;
while(counter < 100)
{
    digitalWrite(10, LOW);
    digitalWrite(53, HIGH);
    smartlynxRun(1, 300);
    delay (100);
    smartlynxRun2(0, 0);
    counter = counter + 1;
}
counter = 0;
}
```

//I made these two functions, though I'll probably just get rid of them in the future...

```
void gpioClear(int CS_Pin) {
    digitalWrite(CS_Pin, LOW);
    digitalWrite(CS_Pin, LOW);
}
void gpioSet(int CS_Pin) {
    digitalWrite(CS_Pin, HIGH);
    digitalWrite(CS_Pin, HIGH);
}
```

```

}

void spiWriteByte(byte data) {
  gpioClear(SPI_CS1);
  SPI.transfer(data);
  gpioSet(SPI_CS1);
}

void spiWriteByte2(byte data) {
  gpioClear(SPI_CS2);
  SPI.transfer(data);
  gpioSet(SPI_CS2);
}

static unsigned char spiExchangeByte(unsigned char data)
{
  unsigned char rdData = 0;
  gpioClear(CS_PIN1); // chip select low
  SPDR = data;        // start transmission
  while(!(SPSR & (1<<SPIF))); // wait till transmission completes
  rdData = SPDR;      // read data byte
  gpioSet(CS_PIN1);  // chip select high
  return rdData;
}

static unsigned char spiExchangeByte2(unsigned char data)
{
  unsigned char rdData = 0;
  gpioClear(CS_PIN2); // chip select low
  SPDR = data;        // start transmission
  while(!(SPSR & (1<<SPIF))); // wait till transmission completes
  rdData = SPDR;      // read data byte
  gpioSet(CS_PIN2);  // chip select high
  return rdData;
}

static unsigned long smartlynxDataTransfer(unsigned long value, unsigned char len)
{
  volatile unsigned char data=0;
  volatile unsigned long rdData=0;

  switch(len)
  {
    case 3:
      //transfer 3rd data byte
      data = value>>16;
      rdData = spiExchangeByte(data);
      rdData = rdData << 8;

    case 2:

```

```

    //transfer 2nd data byte
    data = value>>8;
    rdData |= spiExchangeByte(data);
    rdData = rdData << 8;

case 1:
    //transfer 1st data byte
    data = value;
    rdData |= spiExchangeByte(data);

}
return rdData;
}
static unsigned long smartlynxDataTransfer2(unsigned long value, unsigned char len)
{
    volatile unsigned char data=0;
    volatile unsigned long rdData=0;

    switch(len)
    {
    case 3:
        //transfer 3rd data byte
        data = value>>16;
        rdData = spiExchangeByte2(data);
        rdData = rdData << 8;

    case 2:
        //transfer 2nd data byte
        data = value>>8;
        rdData |= spiExchangeByte2(data);
        rdData = rdData << 8;

    case 1:
        //transfer 1st data byte
        data = value;
        rdData |= spiExchangeByte2(data);

    }
    return rdData;
}

void smartlynxWriteData(uint32_t data, uint8_t byteCount)
{
    for (int index = 3; index >= 1; index--) {
        // right shift data according to loop index
        gpioClear(SPI_CS1);

```

```
    gpioClear(SPI_CS2);
    SPI.transfer(data >> (8 * (index - 1)));
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
}
}

void smartlynxSetMaxSpeed(uint32_t maxSpeed)
{
    uint32_t data;
    // data conversion for smart lynx
    data = 0.065536 * maxSpeed;
    gpioClear(SPI_CS1); // assert chip select
    // set maximum speed command
    spiWriteByte(SMARTLYNX_SET_PARAM | SMARTLYNX_MAX_SPEED);
    gpioSet(SPI_CS1); // de-assert chip deselect
    // set maximum speed of rotation
    smartlynxDataTransfer(data, 2);
}

void smartlynxSetMaxSpeed2(uint32_t maxSpeed)
{
    uint32_t data;
    // data conversion for smart lynx
    data = 0.065536 * maxSpeed;
    gpioClear(SPI_CS2);
    // set maximum speed command
    spiWriteByte2(SMARTLYNX_SET_PARAM | SMARTLYNX_MAX_SPEED);
    gpioSet(SPI_CS2);
    // set maximum speed of rotation
    smartlynxDataTransfer2(data, 2);
}

void smartlynxSetMinSpeed(uint32_t minSpeed)
{
    uint32_t data;
    // data conversion for smart lynx
    data = 4.1943 * minSpeed; // minimum speed can be 0
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // set minimum speed command
    spiWriteByte(SMARTLYNX_SET_PARAM | SMARTLYNX_MIN_SPEED);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
    //set minimum speed of rotation
    smartlynxDataTransfer(data, 2);
}
```

```
void smartlynxSetAcceleration(uint32_t acc)
{
    uint32_t data;
    // data conversion for smart lynx
    data = 0.0687 * acc;
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // set acceleration command
    spiWriteByte(SMARTLYNX_SET_PARAM | SMARTLYNX_ACC);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
    // set acceleration value
    smartlynxDataTransfer(data, 2);
}

void smartlynxSetStepSize(uint8_t stepSize)
{
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // set step size command
    spiWriteByte(SMARTLYNX_SET_PARAM | SMARTLYNX_STEP_MODE);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // set step size
    spiWriteByte(stepSize);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
}

void smartlynxSetSpeed(uint32_t runspeed)
{
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // set speed command
    spiWriteByte(SMARTLYNX_SET_PARAM | SMARTLYNX_FS_SPD);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
    // set speed
    smartlynxDataTransfer(runSpeed, 2);
}

// this function will put motor in continuous running mode
void smartlynxRun(uint8_t dir, uint32_t runspeed)
{
```



```
uint32_t data;
// data conversion for smart lynx
data = 67.1 * runspeed;
gpioClear(SPI_CS1);
// run command
spiWriteByte(SMARTLYNX_RUN | dir);
gpioSet(SPI_CS1);
// set speed
smartlynxDataTransfer(data, 3);
}
void smartlynxRun2(uint8_t dir, uint32_t runspeed)
{
uint32_t data;
// data conversion for smart lynx
data = 67.1 * runspeed;
gpioClear(SPI_CS2);
// run command
spiWriteByte2(SMARTLYNX_RUN | dir);
gpioSet(SPI_CS2);
// set speed
smartlynxDataTransfer2(data, 3);
}
// this function will move motor by specified number of steps (stepCount)
void smartlynxMove(uint8_t dir, uint32_t stepCount)
{
gpioClear(SPI_CS1);
// move command
spiWriteByte(SMARTLYNX_MOVE | dir);
gpioSet(SPI_CS1);
// set number of steps to move
smartlynxDataTransfer(stepCount, 3);
}
void smartlynxMove2(uint8_t dir, uint32_t stepCount)
{
gpioClear(SPI_CS2);
// move command
spiWriteByte2(SMARTLYNX_MOVE | dir);
gpioSet(SPI_CS2);
// set number of steps to move
smartlynxDataTransfer2(stepCount, 3);
}
// single step driving for manual control
void smartlynxSingleStep(void)
{
// pulse to STEP pin
gpioClear(STEP_PIN1);
```

```
    gpioClear(STEP_PIN2);
    delayMicroseconds(1);
    gpioSet(STEP_PIN1);
    gpioSet(STEP_PIN2);
}

// this function will stop motor by decelerating it
// to minimum speed set
void smartlynxSoftStop(void)
{
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // soft stop command
    spiWriteByte(SMARTLYNX_SOFT_STOP);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
}

// this function will stop the motor immediately
void smartlynxHardStop(void)
{
    gpioClear(SPI_CS1);
    gpioClear(SPI_CS2);
    // hard stop command
    spiWriteByte(SMARTLYNX_HARD_STOP);
    gpioSet(SPI_CS1);
    gpioSet(SPI_CS2);
}
```

```
#include <iostream>
```

```

#include <cstring>
#include <string>
#include <sstream>
#include <iomanip>
#include <cmath>
using namespace std;

int main()
{
    cout << std::fixed;
    cout << std::setprecision(2);
    float x, y, a, b, tmp, theta, xdiff, ydiff;
    float PI = 3.14159265358979323846;
    int n, arm, counter, counter2;
    theta = 0.00;
    counter = 0;
    counter2 = 0;
    while(cin >> n)
    {
        cin >> x;
        cin >> y;
        arm = 1;
        //cout << "Start point ";
        // cout << x << ", " << y << ", " << arm << "0" << endl;
        //Find theta to turn the robot towards next point
        if(counter != 0)
        {
            xdiff = x - a;
            ydiff = y - b;
            theta = (atan2 (ydiff,xdiff) * (180 / PI)) - theta;
            cout << x << ", " << y << ", " << arm << ", " << theta << endl;
        }
        else
            cout << x << ", " << y << ", " << arm << ", " << "0" << endl;
        counter = 1;
        //Call robot to move to (x,y)
        //Put marking arm down
        cin >> n;
        cin >> a;
        cin >> b;
        counter2 = 0;
        xdiff = a - x;
        ydiff = b - y;
        theta = (atan2 (ydiff,xdiff) * (180 / PI)) - theta;
        if((x == a)&&(y == b))
            cout << "Repeat point" << endl;
        else if(x == a)
        {
            if(b > y)
            {
                tmp = y;
                while(b > (tmp + 0.5))
                {
                    tmp = tmp + 0.5;
                }
                if(counter2 == 0){
                    cout << x << ", " << tmp << ", " << arm << ", " << theta << endl;}
                else{

```

```

        cout << x << ", " << tmp << ", " << arm << ", " << "0" << endl; }
        //Call robot to move to (x,tmp) while marking
        counter2 = 1;
    }
}
if(y > b)
{
    tmp = y;
    while(b < (tmp - 0.5))
    {
        tmp = tmp - 0.5;
        if(counter2 == 0){
            cout << x << ", " << tmp << ", " << arm << ", " << theta << endl; }
            else{
                cout << x << ", " << tmp << ", " << arm << ", " << "0" << endl; }
                //Call robot to move to (x,tmp) while marking
                counter2 = 1;
            }
        }
    }
}
else if (y == b)
{
    if(a > x)
    {
        tmp = x;
        while(a > (tmp + 0.5))
        {
            tmp = tmp + 0.5;
            if(counter2 == 0){
                cout << tmp << ", " << y << ", " << arm << ", " << theta << endl; }
                else{
                    cout << tmp << ", " << y << ", " << arm << ", " << "0" << endl; }
                    //Call robot to move to (tmp,y) while marking
                    counter2 = 1;
                }
            }
        }
    }
    if(x > a)
    {
        tmp = x;
        while(a < (tmp - 0.5))
        {
            tmp = tmp - 0.5;
            if(counter2 == 0){
                cout << tmp << ", " << y << ", " << arm << ", " << theta << endl; }
                else{
                    cout << tmp << ", " << y << ", " << arm << ", " << "0" << endl; }
                    //Call robot to move to (tmp,y) while marking
                    counter2 = 1;
                }
            }
        }
    }
}
//cout << "End point ";
arm = 0;
cout << a << ", " << b << ", " << arm << ", " << "0" << endl << endl;
//Call robot to move to (a,b)
//Lift marking arm back up

```

```
    }  
    return 0;  
}
```

## Biography

Justin Gibbs – Justin is a Mechanical Engineering senior currently studying at Florida State University. Justin looks to expand his knowledge base and continue his education in hopes of receiving a Ph.D. and eventually being a professor.

Christian Baez - Christian is a Computer Engineering student at Florida State University. Christian seeks to start his career after graduation, becoming an asset for his company.

Derrick Portis - Derrick is a senior at Florida A&M majoring in Computer Engineering. Derrick looks to start working as a software engineer after graduation.

Kelsey Howard - Kelsey is a Mechanical Engineering senior at the FAMU-FSU College of Engineering. Upon graduation Kelsey looks to start working in industry with a focus in thermal fluids and thermal fluid design.

Brandon Roberts - Brandon is a senior in Mechanical Engineering at the FAMU-FSU College of Engineering currently seeking his Bachelors of Science with plans to move to seeking his Master's degree through the university's BS/MS program, then moving on to industry.