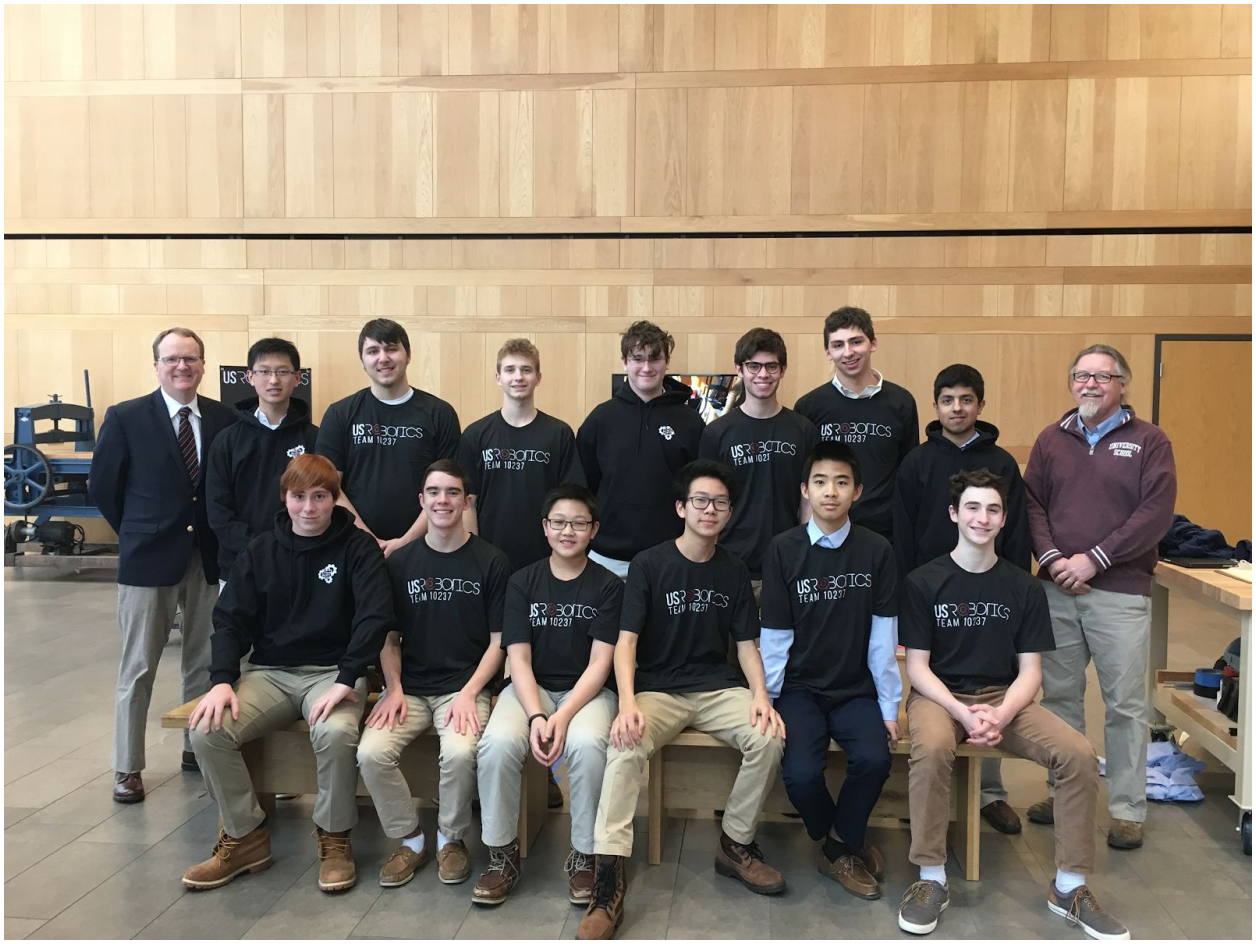
Table of Contents

About the Team



University School started the US Robotics team during the Res-Q season. During our rookie season we made an excellent first showing and managed to impress the judges. We were one of two rookie teams who made it to states. During Velocity Vortex we returned to states and finished in a higher ranking than in the previous year. We also won the PTC Design award. Since then we have improved our engineering, building, and programming skills. The team put more hours in this year and worked hard to make a functional and beautiful robot. Everyone on the team is eager to excel in the FIRST community and take what they have learned on to college. The team is excited and proud of the work that they have accomplished this season.

Michael Letterio (Senior): Michael co-founded the robotics club in 2014. He is a lead builder and a part of the drive team. He played a crucial role in the design and build of the robot. As a senior, Michael hopes to have a successful last season. When Michael is not in robotics, he enjoys building and flying R/C planes and drones, skiing, and video editing.

Andrew Heater (Senior): Andrew joined the team last year. He helped with troubleshooting and ideating design elements on the robot and taking photos. When he is not in robotics, Andrew enjoys playing video games.

Dylan Siegler (Junior): Dylan joined 2014 and is now a lead programmer on the team and one of the drivers. Dylan has worked tirelessly on his autonomous and has big plans for the future of our robot. When he is not in robotics, Dylan enjoys programming, playing squash, looking devilishly handsome, and gangling.

Max Borsch (Junior): Max joined the team in 2014 as one of the programmers. During meetings, Max helps come up with ideas as well as working with the drivers to program controls. When he is not in robotics, Max enjoys programming personal projects as well as playing tennis and soccer.

Reed Chen (Junior): Reed joined the team in 2014. He played a crucial role in the design and construction of the robot this year. Reed has improved in CAD since last year and has taken on the huge responsibility of updating the CAD model of our robot and making animations. When he is

not in robotics, Reed likes to play video games and tennis.

Daniel Pichkar (Junior): Daniel joined the team in 2014. Daniel has taken on a larger role this season prototyping and building the robot. He continues to do research and outreach on various platforms. Outside of robotics, Daniel flies his drone and works on personal projects.

Maheep Brar (Junior): Maheep joined the team in 2014. He assists with the ideating and building of the robot and continues to learn and become proficient in CAD to do so. He also enjoys science and video games.

Benjamin Wyant (Sophomore) Ben joined this year and has helped with building both robots. He plays baseball competitively and likes to ski. He is a programmer and is in AP comp sci this year. Next year he will play a large role in the building of the robot.

Tyler Nettis (Sophomore): Tyler joined the team in 2016. He helped to build the ProtoBot and will help build the main robot next year. In his free time, he enjoys cars and is a big Formula 1 fan.

Gary Huang (Freshman): Gary joined the team this year from Beijing. He has contributed many important contraptions to our robot, including a relic mechanism and will continue to test ideas next year. While not in robotics, he enjoys coding and Chinese.

Noah Mitchell (Freshman): Noah joined the team this year. He has spent a lot of time working on our robot, and will continue helping the

team next year. While not in robotics, he enjoys playing soccer.

Zihao Qing (Freshman): Zihao joined the team this year. Throughout the year, he has submitted many important ideas, and will become a key team member next year, just like this year. While not in robotics, he enjoys doing his math homework.

Vincent Chen (Freshman): Vincent joined the team this year, as a programmer for the ProtoBot, and will program the main robot next year. While not in robotics, Vincent likes to play Ravenfield and do PenOhio.

Jay Chiang (Freshman): Jay joined the Robotics team this year, as a crucial contributor to the team. Next year, he will still contribute his skills to the team.

Outreach

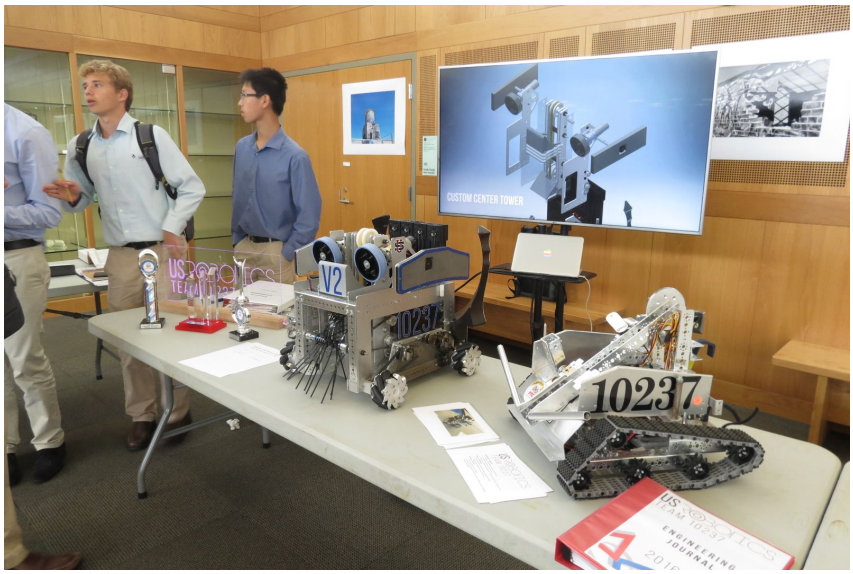
This year we wanted to impact all of the communities we are part of, beyond just our FIRST communities. We accomplished this through a variety of events and activities. We worked hard to promote STEM and FIRST throughout University School, Northeast Ohio, and abroad.

School Club Fair (9/1/17): We showcased the team for students, parents, and faculty, introducing them to FIRST and FIRST Tech Challenge. People were able to drive around the Res-Q robot and ask questions about what the team does. The fair brought in many new recruits to our ranks, who are eager to get started.

Team members: Reed, Max, Dylan, Michael, Daniel, Maheep

People impacted: 50-75

Hours: 4



FTC Kickoff Event (9/9/17): We attended the kickoff event at Rockwell Automation in Cleveland at the beginning of the season. In the morning, Dylan and Mr. Sweeney taught an introduction to programming class. During this class, we taught about 20 teams (50 team members) about basic programming concepts and how to get basic teleops and autonomous running using OnBot Block Code. The other team members assisted other teams with other issues such as updating their phones, updating their apps, and getting their electronics working.

Team members: Reed, Max, Dylan, Michael, Daniel, Maheep, Andrew, Tyler
People impacted: 100
Hours: 6

Solon Scrimmage (10/18/17): The Solon scrimmage is a yearly event where Northeast Ohio teams come to show their early designs and discuss plans for the season. This year many rookie teams showed up to the scrimmage. We helped these teams set up their electronics and showed off our gripper robot.

Team members: Everyone
People impacted: 40
Hours: 8

Kenston Jr. FLL Expo (1/20/18): This was a first for our team. Volunteering with the kids in Jr. Fll really was a really eye-opening experience. Seeing the next generation of engineers was amazing. The creativity, ingenuity, and problem solving skills are really promising. Michael worked as a team reviewer and Dylan and Michael led the core values challenge.

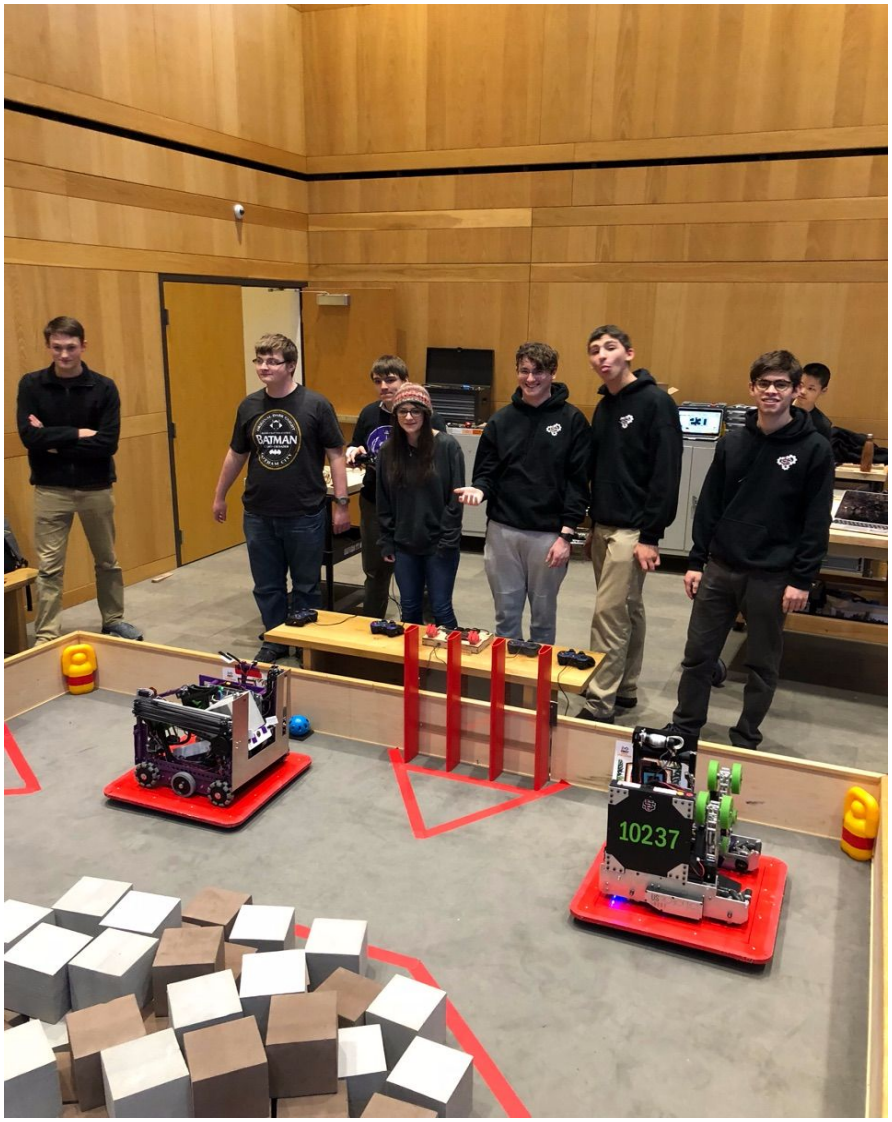
Team members: Dylan, Michael, Daniel
People impacted: 75
Hours: 3

Hosted Practice with Team 8581 (1/27/18): A key part of the FIRST experience is interacting with other teams and sharing ideas. We invited over Team 8581ÆDIFICATORES to run some matches and talk strategy, engineering notebook, and have some fun.

Team members: Dylan, Michael, Daniel, Max, Vincent, Ben

People impacted: 12

Hours: 3



Mentoring FLL Teams (Ongoing): The middle school students at University School were really excited to hear that we would be giving them a helping hand this season. Team 5416 SWARM and Team 5413 NEWTS are putting in maximum effort and time to make sure their FLL season is a great one. This is something we hope we can continue to do because the boys on the teams are really excited to keep doing FLL and hop onto the FTC team , when they come to the high school.

Team members: Dylan, Michael, Daniel, Reed, Maheep

People impacted: 12

Hours: 6

YouTube, Discord, and Reddit: Cooperating and sharing ideas with teams throughout the nation and globally has been really fun and educational for our team. Our team has decided to use multiple platforms to do this. On YouTube we have posted videos that share our design and CAD for the season. On Discord, members of the team have been very engaged discussing upcoming events, helping teams solve problems, and having good laughs. We used Reddit this year to create a survey for teams to use to evaluate the pros and cons of different phones.

Team members: Dylan, Michael, Daniel, Reed, Maheep

People impacted: ~5000 (views on YouTube, Reddit submissions, and Discord conversations)

Hours: 255

Strategic Plan

US Robotics has several goals which they plan to accomplish over the course of the season.

In competition, we hope to accomplish two sets of goals: one on the field and one with the team. Our goals on the field include qualifying for States, succeeding as a final seed alliance, and moving on to the next level of competition, North Super Regionals.

We plan to create a detailed Engineering Notebook that documents our entire season. In addition to the first team goal, we want to put emphasis on design and out of the box thinking. The team seeks to achieve awards such as the Design Award, Rockwell Collins Innovate award, Control Award, and Think award.

Outside of the competitive sphere, we also hope to increase our community outreach by being more active in a variety of communities. This helps us become role models who raise awareness of the STEM and FIRST programs everywhere. We have set out these goals not only to better our performance, but also to improve our relationships with one another and the communities we are part of.

Business Strategy

US Robotics is funded and supported by University School. Because we are a non-profit, we are unable to get sponsors. Although we don't have sponsors, we have to assess each of the purchases that we make.

Here are the questions we ask when using our budget to make purchases:

1. What problem will we solve by purchasing this item?
2. What other options do we have to solve this problem? If so, why is this specific item better than other items?
3. Can we make a prototype to ensure that the item will actually work?
4. Have we created a purchase order that clearly states the item, reasoning, cost, and place of purchase?

Once all of the above questions have been answered then the team proceeds to purchase the item. This process makes sure that parts that are purchased are more likely to actually be put to use. This process also ensures that more than one person is a part of the decision process to purchase items.

Design Strategy

This is the design process our team uses when working on mechanisms for the robot. This ensures that ideas are thoroughly evolved before they are finalized and implemented on the robot. The team will use materials and tools to effectively bring the ideas to life.

Step 1 – **Understand** – Define the problem. What parts of the problem are important or not?

Step 2 – **Define** – Determine solution specifications. What restrictions or constraints do we have? What time constraints do we have?

Step 3 – **Explore** – Do background research. Are there similar challenges to what we are doing? Are there any existing mechanisms or designs that might be adapted to our challenges? How do they differ from what we want.

Step 4 – **Ideate** – Generate concept solutions. Using inspiration from exploration and considering the constraints, come up with feasible ideas.

Step 5 – **Prototype** – Learn how our concepts work. Quickly build the idea and test it to see how well it performs.

Step 6 – **Refine** – Do detailed design. Working off the prototype, identify the issues with the prototype and try to fix them.

Step 7 – **Choose** – Determine a final concept. Which idea performs the best and meets all constraints?

Step 8 – **Implement** – Implement the detailed solution. Try to fix issues. Look into more permanent materials and manufacturing process.

Step 9 – **Test** – Does the solution work? How well does it work? It what ways does it excel and in what ways does it fail?

Step 10 – **Present** – Get feedback and approval. How well does the design work? What things can we still improve? What do other people say about our design.

Step 11 – **Iterate** -- Using the results from testing, identify issues, redesign, prototype and test further, and repeat until a we have designed something that meets the initial design goals.

Game Strategy

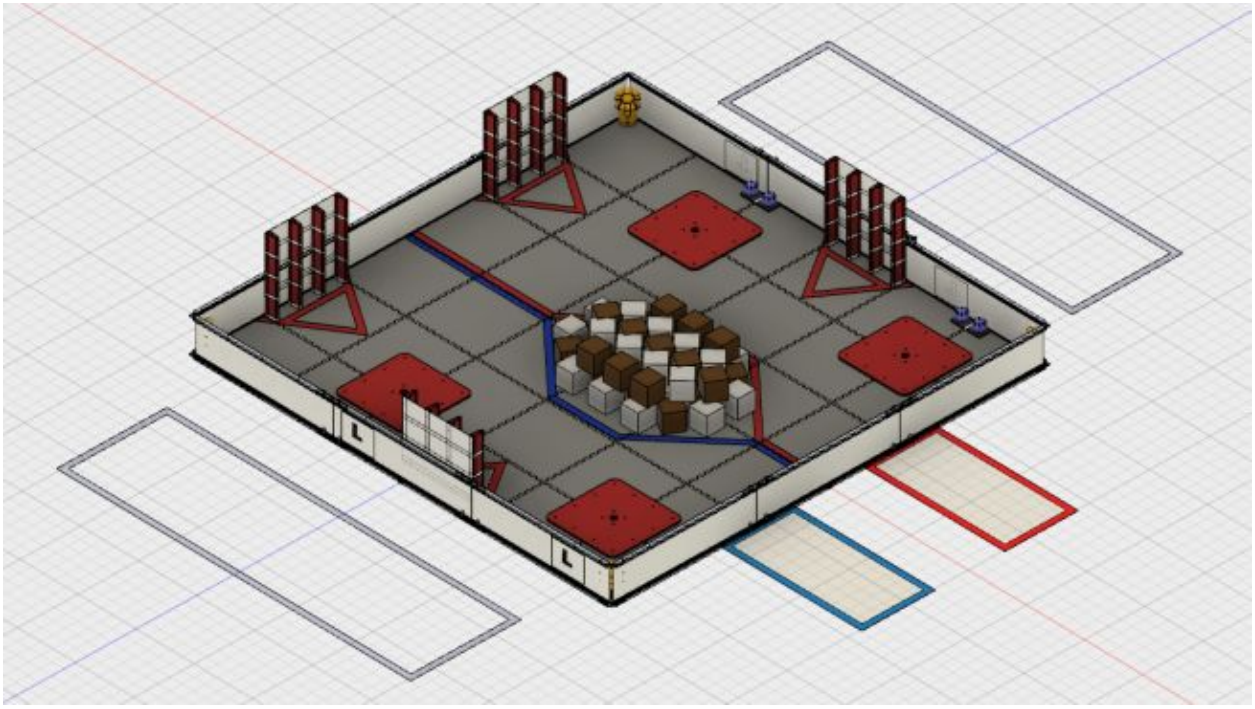
Our team's plan has evolved throughout the season as the robot became more developed. Below is our current strategy that we are using at competition. This section will be updated as mechanisms and capabilities of the robot evolve. It will be job of the Drive Team coach to assess the situation during each match and make scoring decisions accordingly.

Autonomous (85-115 points): A consistent autonomous that includes knocking the appropriate jewel, placing a glyph in the correct cryptobox according to the key, and parking in the safe zone. Additionally the robot will attempt to enter the glyph pit and collect 1 or 2 more glyphs to place into the cryptobox.

Tele-Op (124-154+): Consistently scores a cryptobox and attempts a cipher. By completing a cipher we hope to offer our partners the ability to score a relic early.

End Game (40-130): We score 1-2 relics upright in zone 3. If this is done in a timely manner, we will then park on the balancing stone.

Early Season: Relic Recovery Begins



Game Reveal

We started the season by attending the game kickoff event at Rockwell Automation. This was a great opportunity to network with other teams and talk about some of the challenges we faced last year and even help rookie teams with questions about the competition or the design process we went through last year.

This is a picture of us showing our Velocity Vortex robot to other teams and explaining how it works

This year's game, Relic Recovery is unique in a few different ways from previous

games we have competed in. As soon as the second part of the game manual was released we began to immediately ideate and layout the aspects of the competition we wanted to focus on.

We had ultimately decided on a strategy that focused directly on being highly efficient at stacking glyphs. We realized that a large sum of points could be gathered based on the patterns and column and row bonuses. We believe that if we are quick enough and can collect the glyphs in an organized manner, completing a full cipher, we could accumulate a score greater than what would be achievable if we focused primarily on the relic and balance stone.

These were our preliminary thoughts about the robot:

- The robot should focus on doing the glyphs well
- The robot should be fast and not too heavy
- Glyphs earn more points than the relics
- Robot should be built to be easily programmable for autonomous

Defining The Problem

We discussed our first thoughts on the game. We first examined the point values of each of the things we could do in the game and tried to devise a general strategy on what to go for:

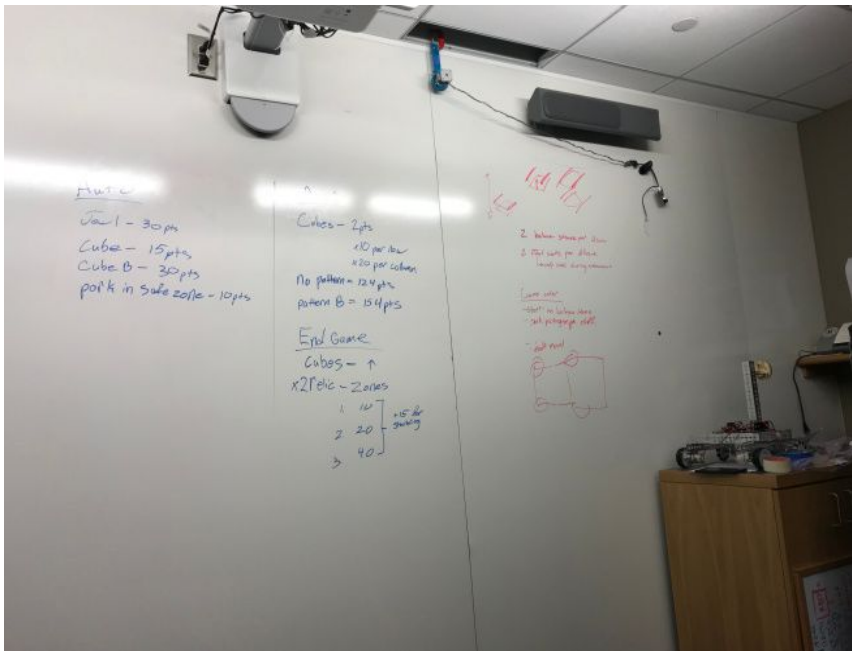
- We determined that stacking glyph would be the key part of our strategy
- After realizing this, we then thought that we could use the same arm to both lift the glyphs and score some points with the relic

We also realized that autonomous mode would be fairly easy if we made sure of a few things:

- A dedicated stick of some sort in the right place would make knocking off the jewel trivial
- Camera placement would be critically important in order to read the key
- Since the robot has to balance at the start of the game, this may pose an issue because as the robot is built the orientation it needs to start in will be the same

We then brainstormed potential ways of harvesting glyphs. One idea (blue in the picture) was to have a linear actuator controlling an arm that harvests the glyphs. Another idea (in red and green) was to have 2 belts harvesting the glyphs and

one glyph stored inside the robot.



This was one of our initial brainstorming sessions. We discussed and wrote what we thought would be the most valuable and achievable sources of points.

Drivetrain

We first looked into drive trains to use on our robot. We considered the pros and the cons of the following drive trains:

Modified West Coast Drive

- Two omniwheels on the front and the back and the middle 2 wheels being power
- All wheels would be on the same plane
- Pros
    - Solid and proven drive train
    - Could be easily and quickly built
    - Would allow for quick tank movement
- Cons
    - Didn't allow movement along the x axis in order to line up with the cryptobox
- We also considered mecanum wheels or some way to move along the x-axis on the lift so that we could be quicker putting the glyphs in

Mecanum Drive

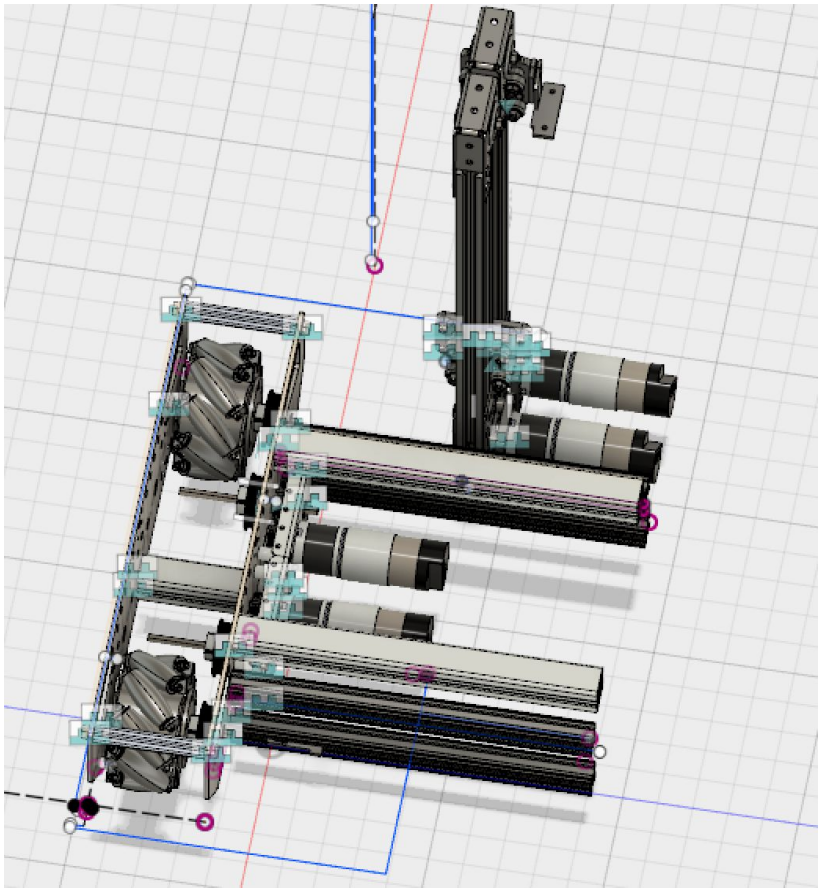
- Having four mecanum wheels
- Pros
    - Would allow us to strafe to line up with the cryptobox
    - Four wheel drive would mean we could get considerable speed
- Cons
    - All four wheels would have to be powered
    - This would mean we would need room for 4 motors somewhere
    - We have had issues in the past with mecanum wheels being less accurate in autonomous due to the lower traction and an unstable chassis

We decided to but the TileRunner chassis by AndyMark to see if it or a similar modified West Coast drivetrain would be viable. We installed a simple arm with a gripper to test out the drivetrain and the arm idea. We found that the lack of maneuverability was a big issue and thus we decided to go with a mecanum drivetrain. However, we liked the structure of having the wheels and motors guarded and driven by belts and inward facing motors.

We set out to design and build a drivetrain inspired by the TileRunner.

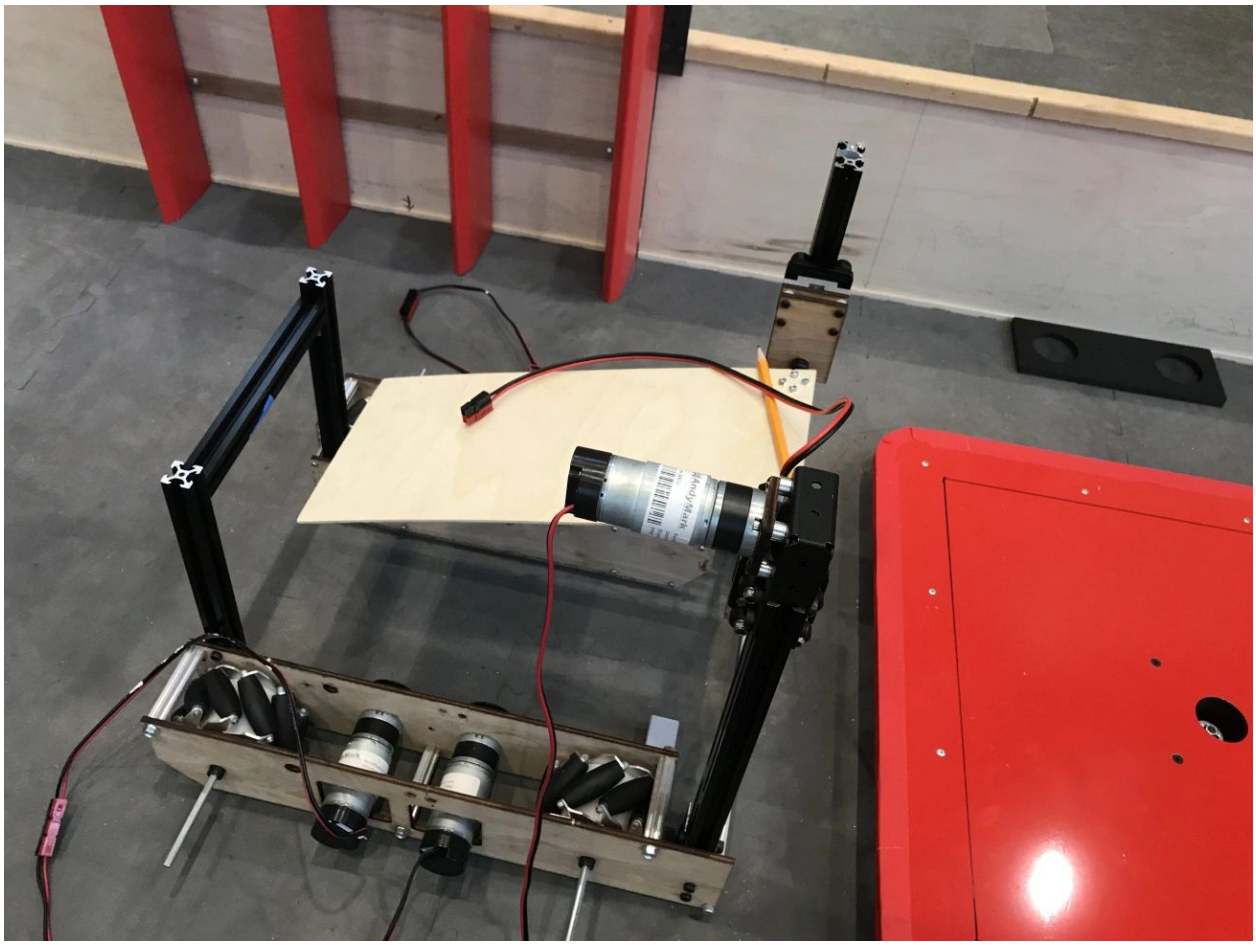
- We used commonly available extrusions to connect the two sides
- We first designed the drivetrain in CAD so we could get the right belt sizes

and measurements



One half of our drivetrain.

We then prototyped the chassis out of wood



We used the laser cutter to prototype our whole drivetrain out of wood.

We then looked into using our CNC shopbot to cut aluminum. Although intended to cut wood, by changing the bit and the settings we were able to get the shopbot to cut aluminum. We cut out the drivetrain in aluminum and assembled it.

After using the drivetrain extensively, we have found the following:

- The drivetrain is very solid and has taken many hits
- The double reinforced motor and wheel mounts have proved to be very solid
- Some of the belts have loosened a little bit, which may indicate that something is shifting a little bit
- The profile of the plates has made it exceptionally hard to park on the balancing stone

Starting Prototypes

When we sat down to plan out the general form of our robot, we came to the following conclusions:

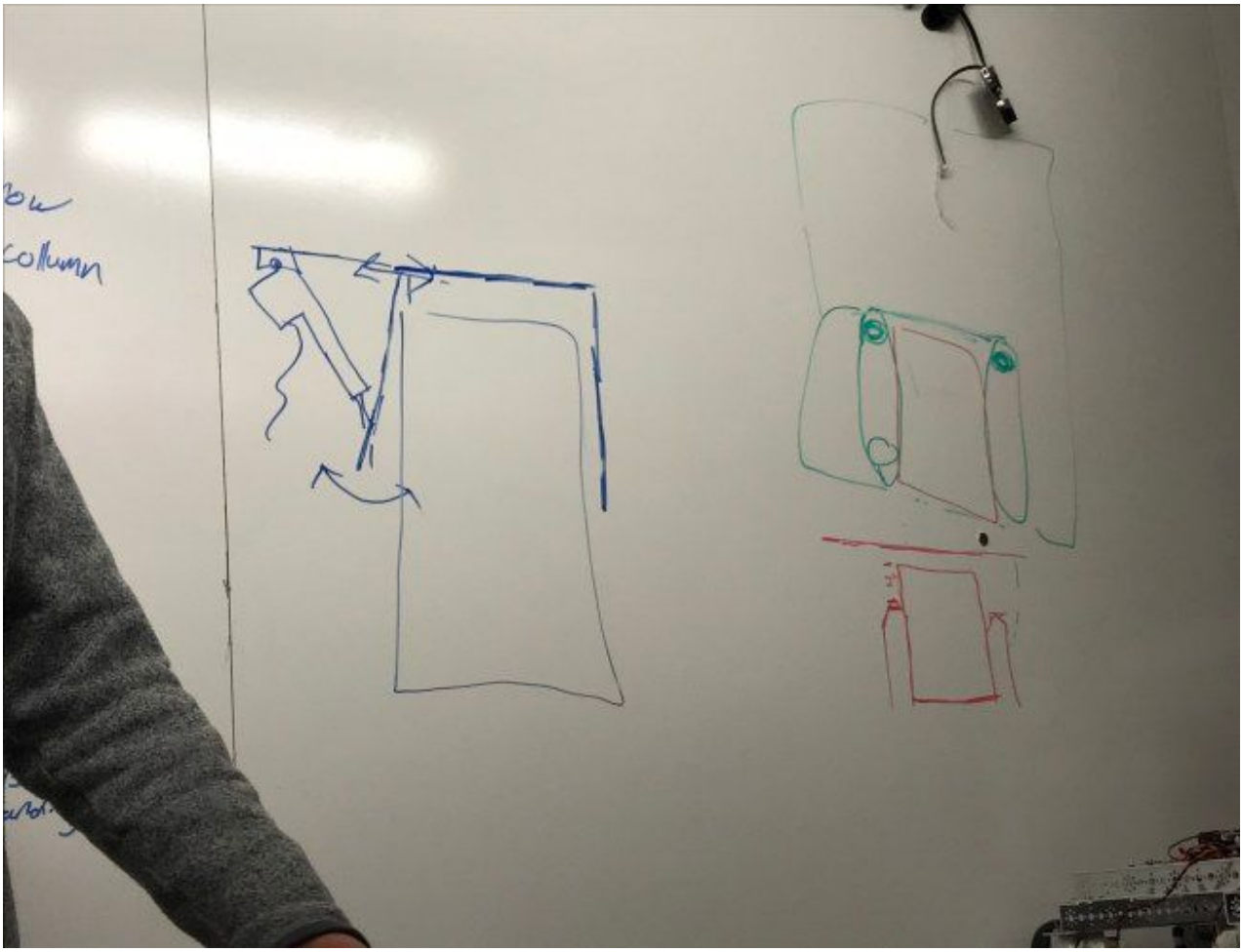
- We wanted to have the phone and the jewel mechanism on the same side of the robot for autonomous
- We wanted a glyph grabbing mechanism on the front of the robot
    - We thought a lot about how to actually acquire the glyphs and we came up with three main systems
        - A grabber would grab the glyphs and the grabber would go up and down on a lift mechanism
            - Pros
                - Simple design
                - We knew that it would work
            - Cons
                - Torque and grip might be an issue (the glyphs might fall out)
                - How would we score two glyphs?
                - This would require us to turn around every time we would need to score glyphs

        - If we were only scoring one at a time, how would we overcome the 18” limit?
        - We would have to line up with the glyphs fairly accurately to grab them
    - Wheels or a conveyor belt would grab the glyphs and they would be lifted up by a lift
        - Pros
            - Potentially much faster to get the glyphs
        - Cons
            - Would have to lift the whole harvesting assembly
            - Still wouldn’t overcome the height restriction
    - A one sided grabber would grab the glyphs
        - Pros
            - Simpler
            - Might help to line up the glyphs better if not at the perfect angle
        - Cons
            - Even less torque
            - Same issues as single gripper
- We considered doubling (stacking two high) each of these designs,

but we ran into issues with staying in the sizing cube while still being able to get the glyphs up high enough

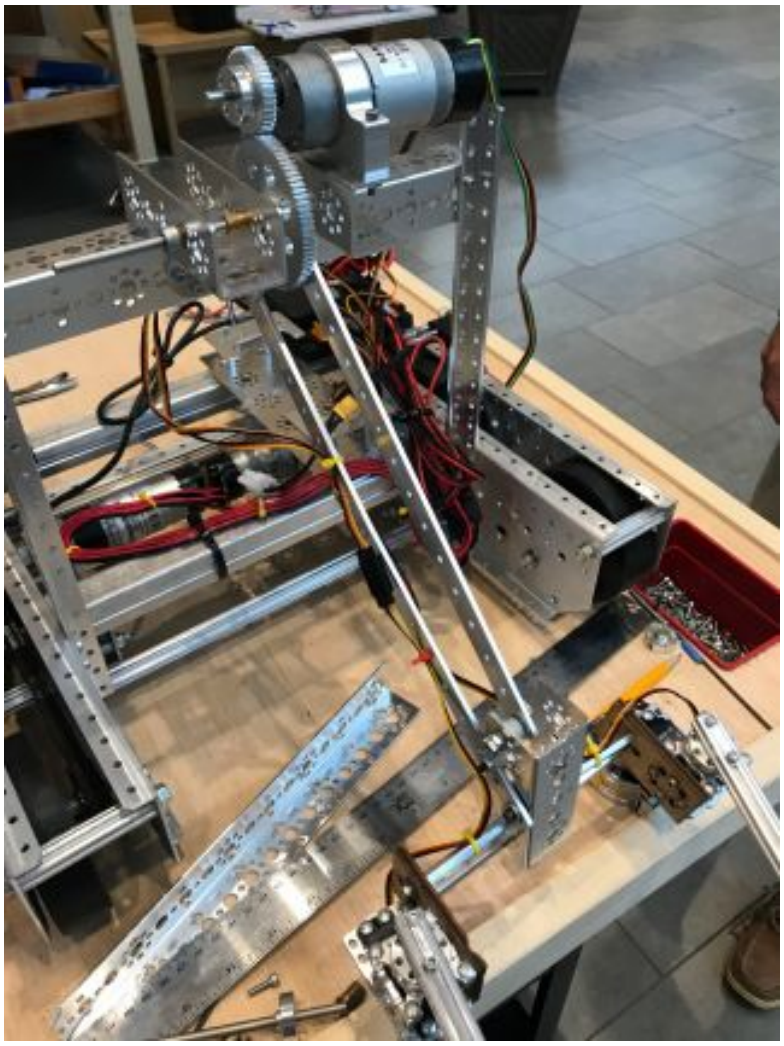
- We initially thought that a grabber and lift design would be the best path to pursue

- Additionally, we considered what our relic mechanism would go
  - We thought we would have some type of drawer slide similar to last year's robot, but on its side
  - We knew that if we left a bit of vertical space empty on the side of the robot, we could put a relic mechanism in there later

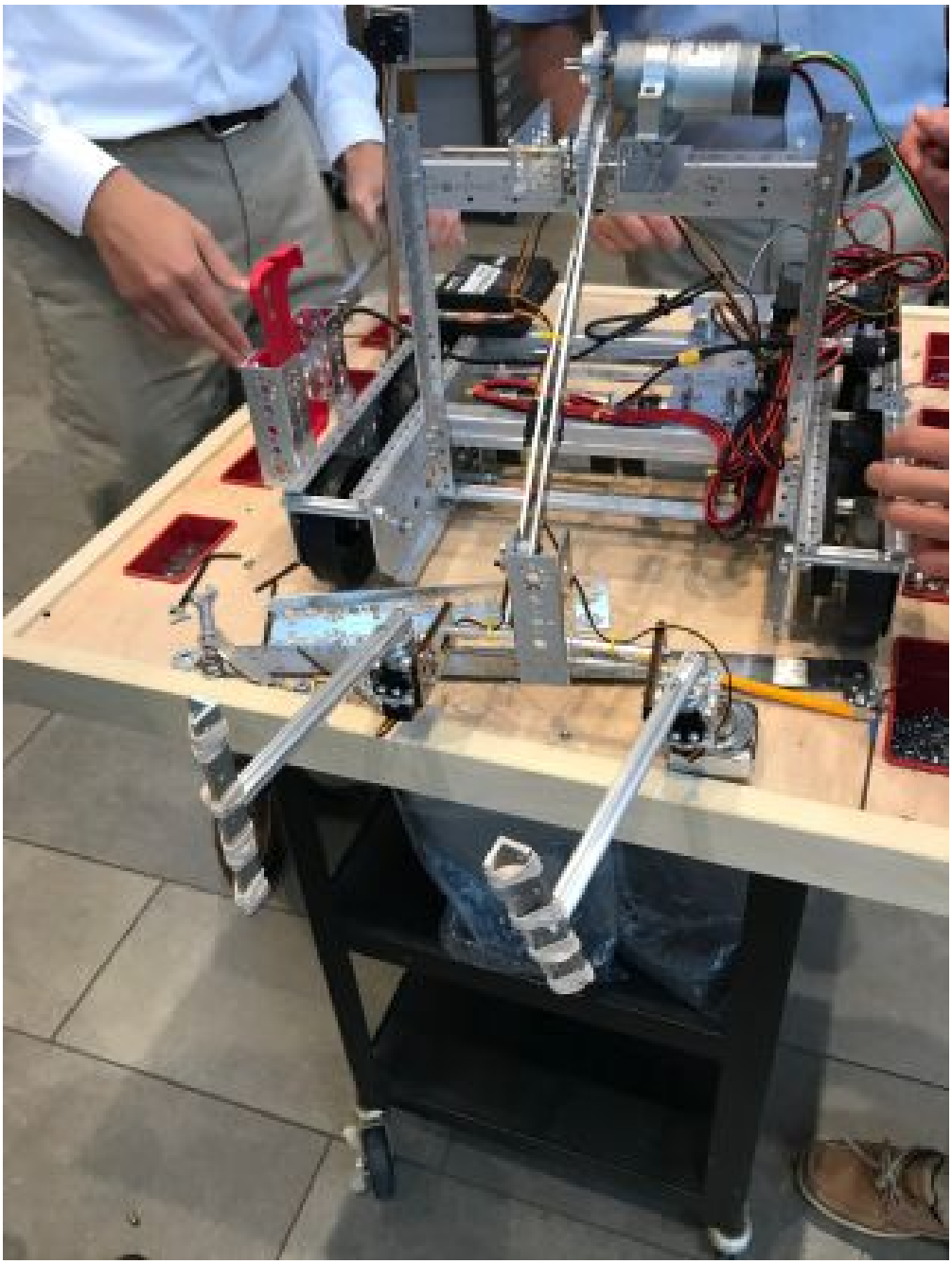


This was one of the designs for the glyphs' harvester. In blue, one linear actuator grasps the glyphs while another side plate rigidly holds it.

For our first prototype, we made a gripper design with an arm-like lift system.



This was our lift arm prototype, which was built on the TileRunner chassis.



This is a front view of the gripper and arm lift.

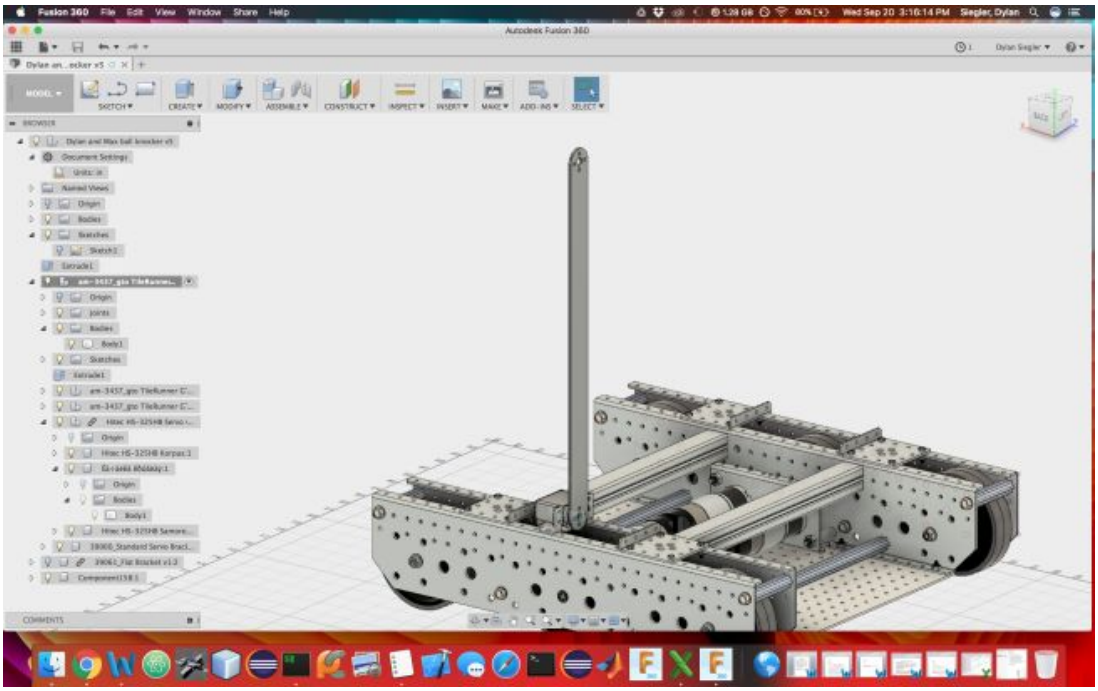
After building this prototype, we reached the following conclusions:

- The arm was not stable for for three main reasons

    - It was not supported well so it was wobble
    - It was very hard to control
    - The distance away from the robot changed as it went up, making it very unpredictable
- We also found a few issues with the gripper
    - The gripper didn't have enough torque or grip to keep the cubes gripped

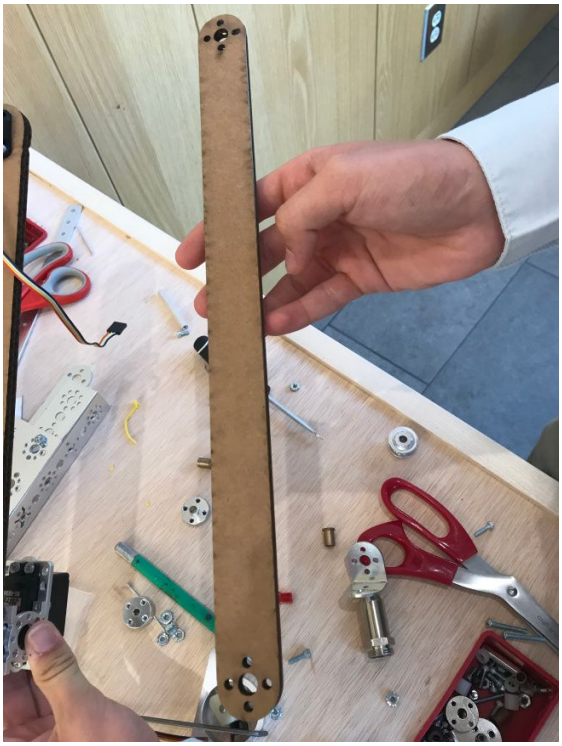
Jewel Knocker Design

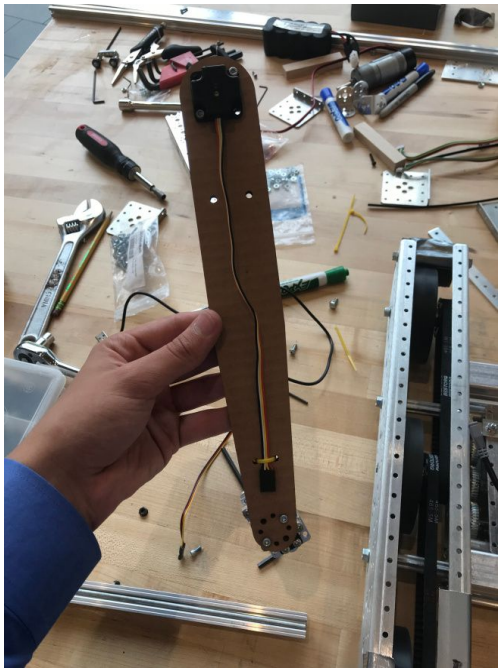
We started making a design for the jewel knocker in CAD. We wanted to have a stick with a color sensor at the end that would pivot down from the side of our robot. By moving the robot, we could knock the correct jewel off.



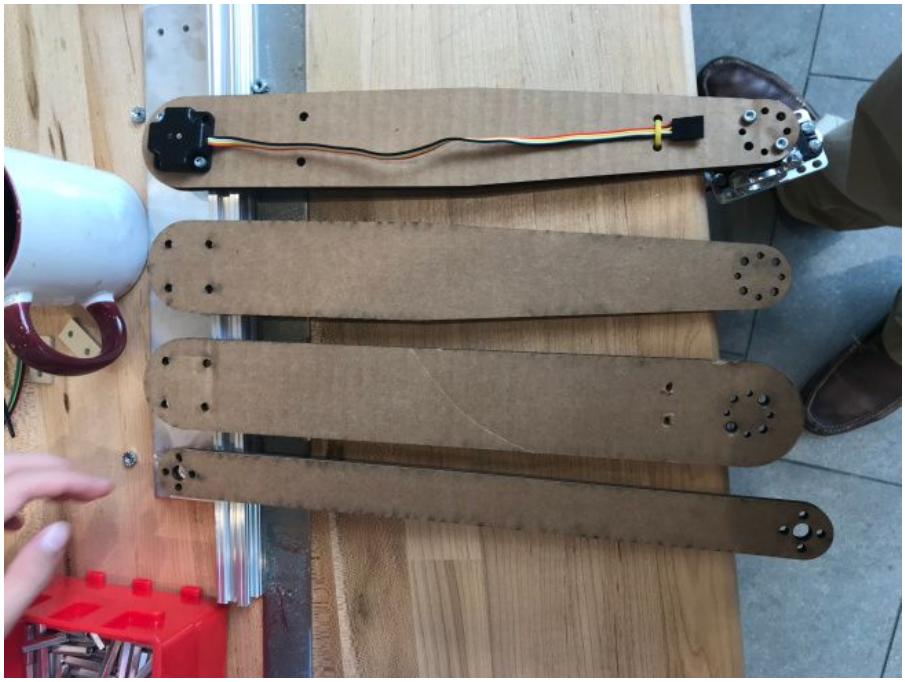
We positioned the servo and designed the first iteration of the jewel knocker in Fusion.

We imported the TileRunner model into Fusion and positioned the servo on the TileRunner as a test to see if it fit. We found that we might need a bit of clearance inside the robot to house the jewel knocking stick. From there we made many iterations of a jewel knocking stick that would accommodate a Modern Robotics color sensor and fit onto the servo.





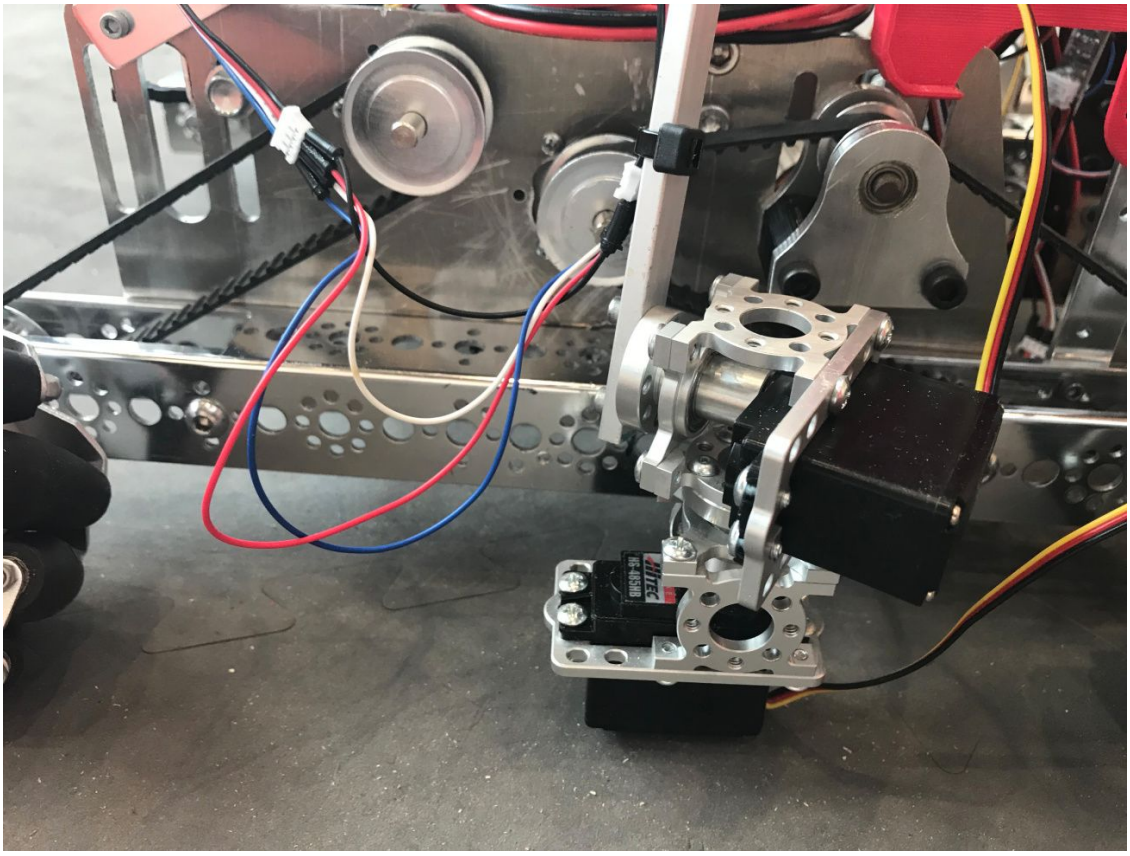
These are our four jewel knocker prototypes (in order, top left, right, bottom left, right).



All four iterations of our jewel knocker.

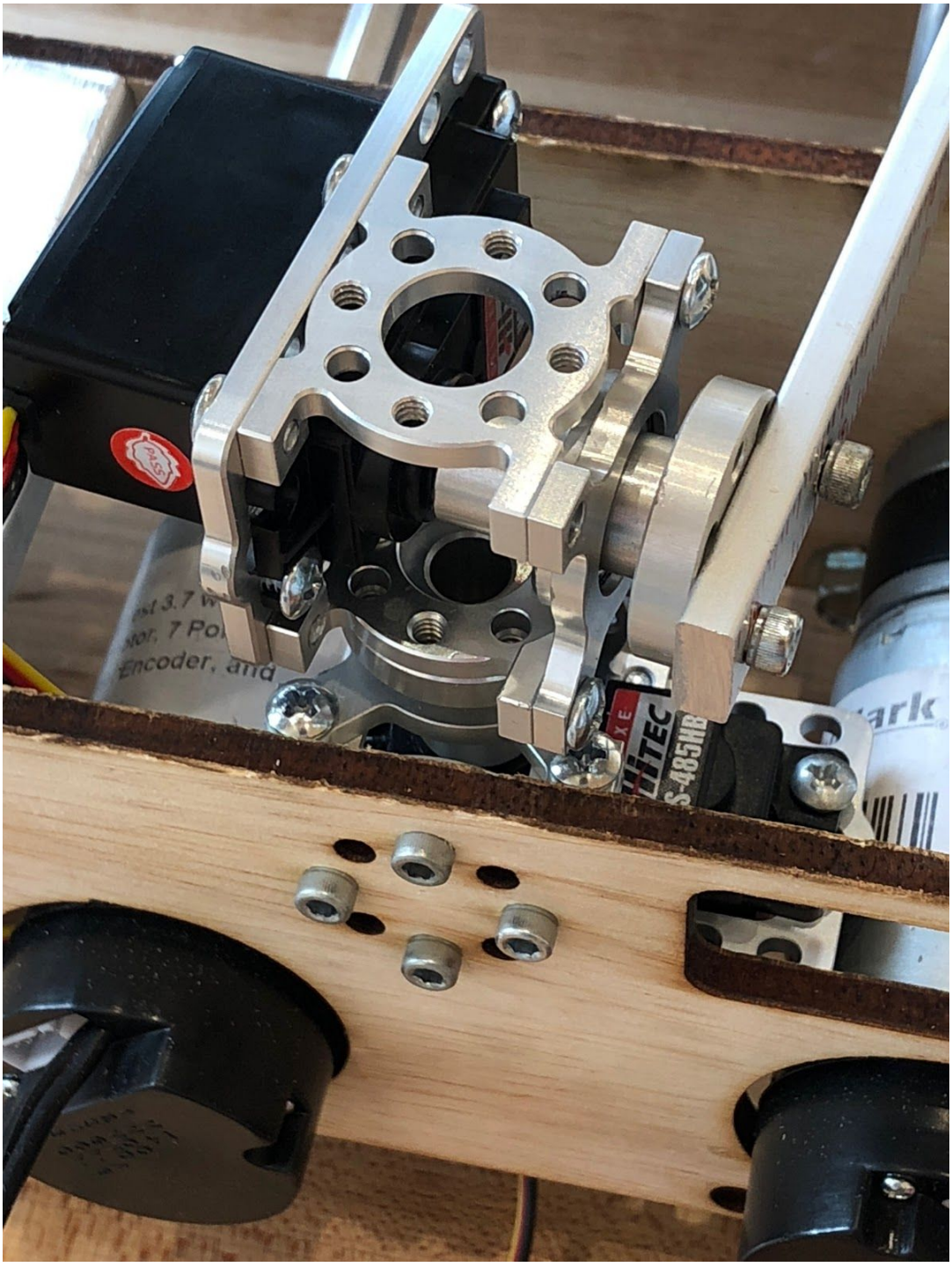
We finally decided on attaching a metal ruler to the servo by drilling holes in it. This solution was lightweight and has proved to be long lasting.

However, we realized that moving the robot to knock the jewel would be very inconsistent. Thus, we simply attached another servo to the first servo. We had to make a bit more room on the side of the robot for this, but it made the autonomous much more consistent. Here is our first prototype:



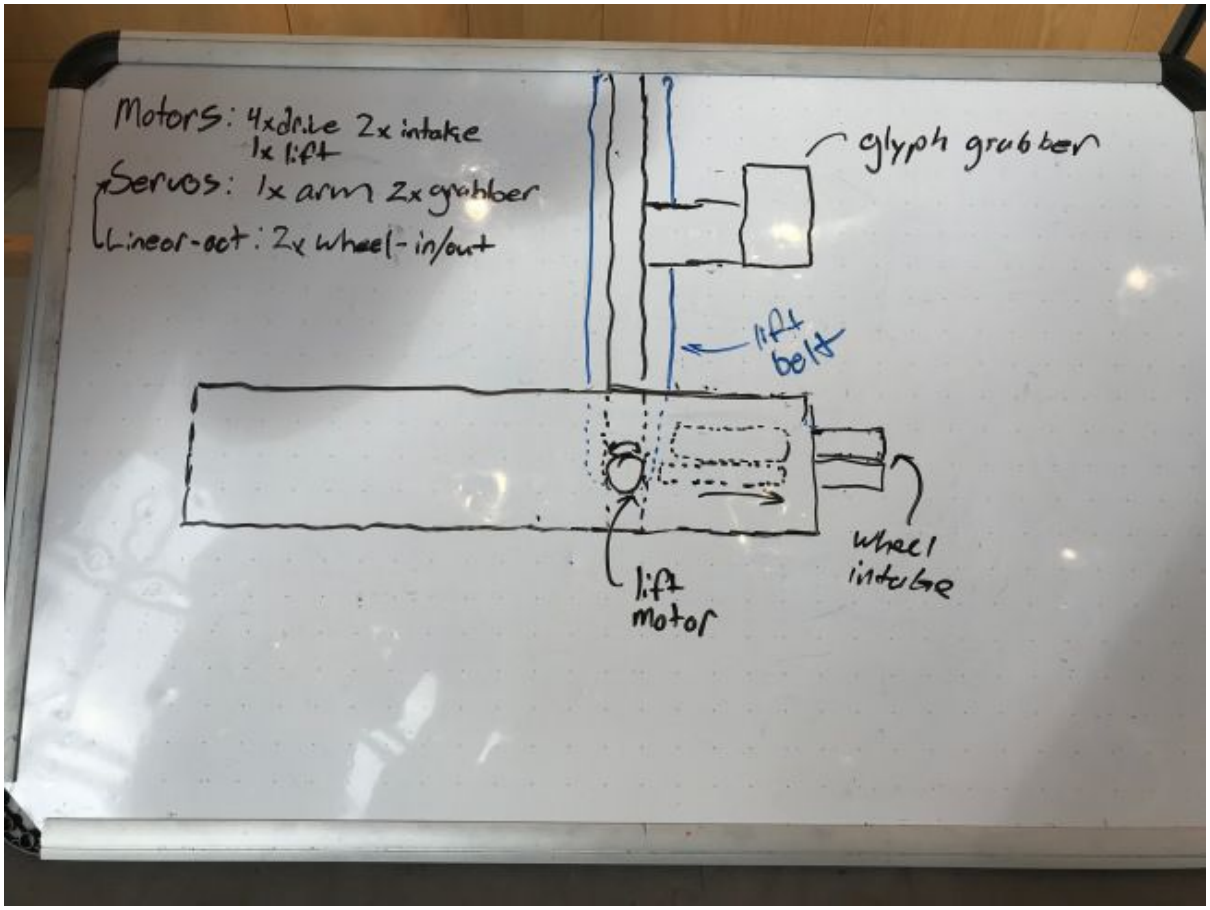
This is the first prototype of our jewel knocker.

We attached the jewel knocker to the new chassis. It fits perfectly between the two drivetrain motors and is sufficiently long to reach the jewels from the balancing stone.



Jewel knocker attached to the full robot prototype.

Lift Design

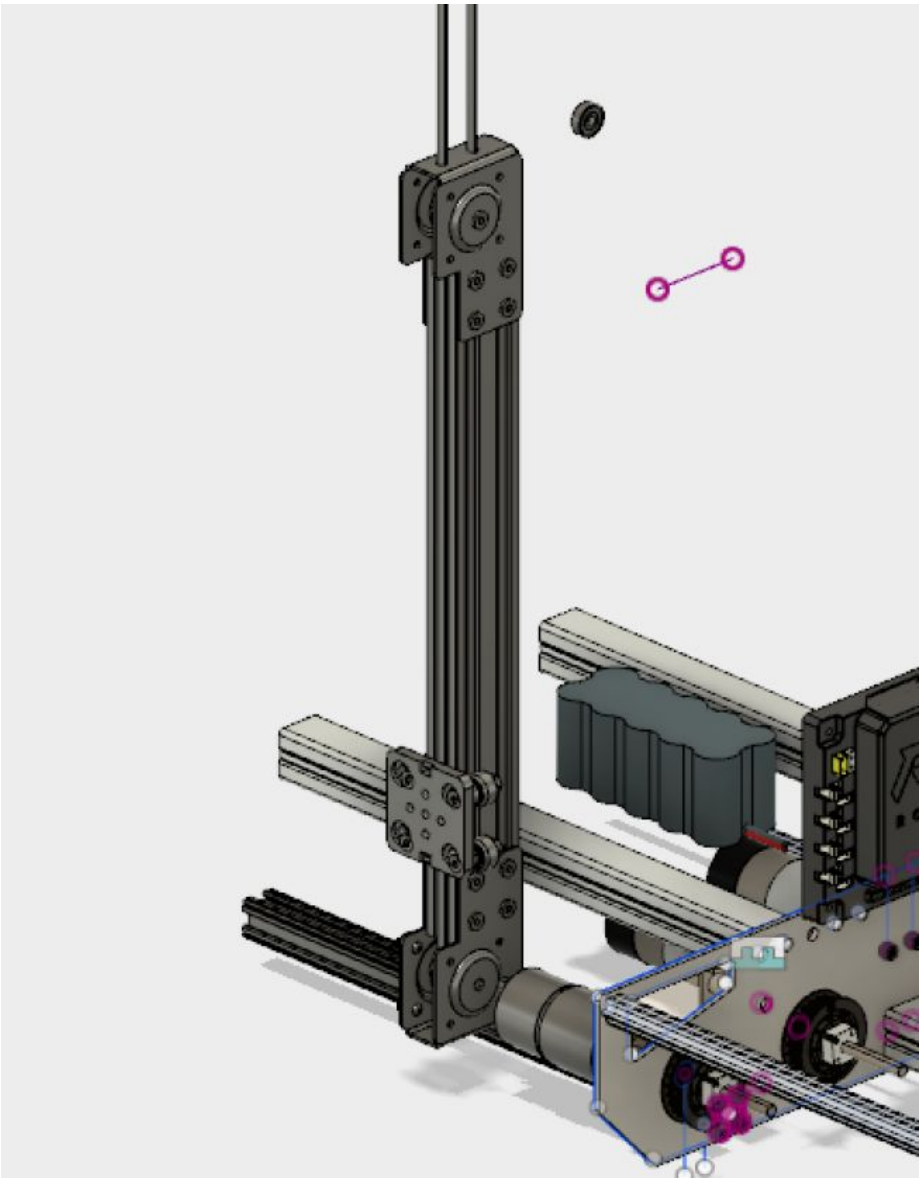


These are the sketches for our first lift idea.

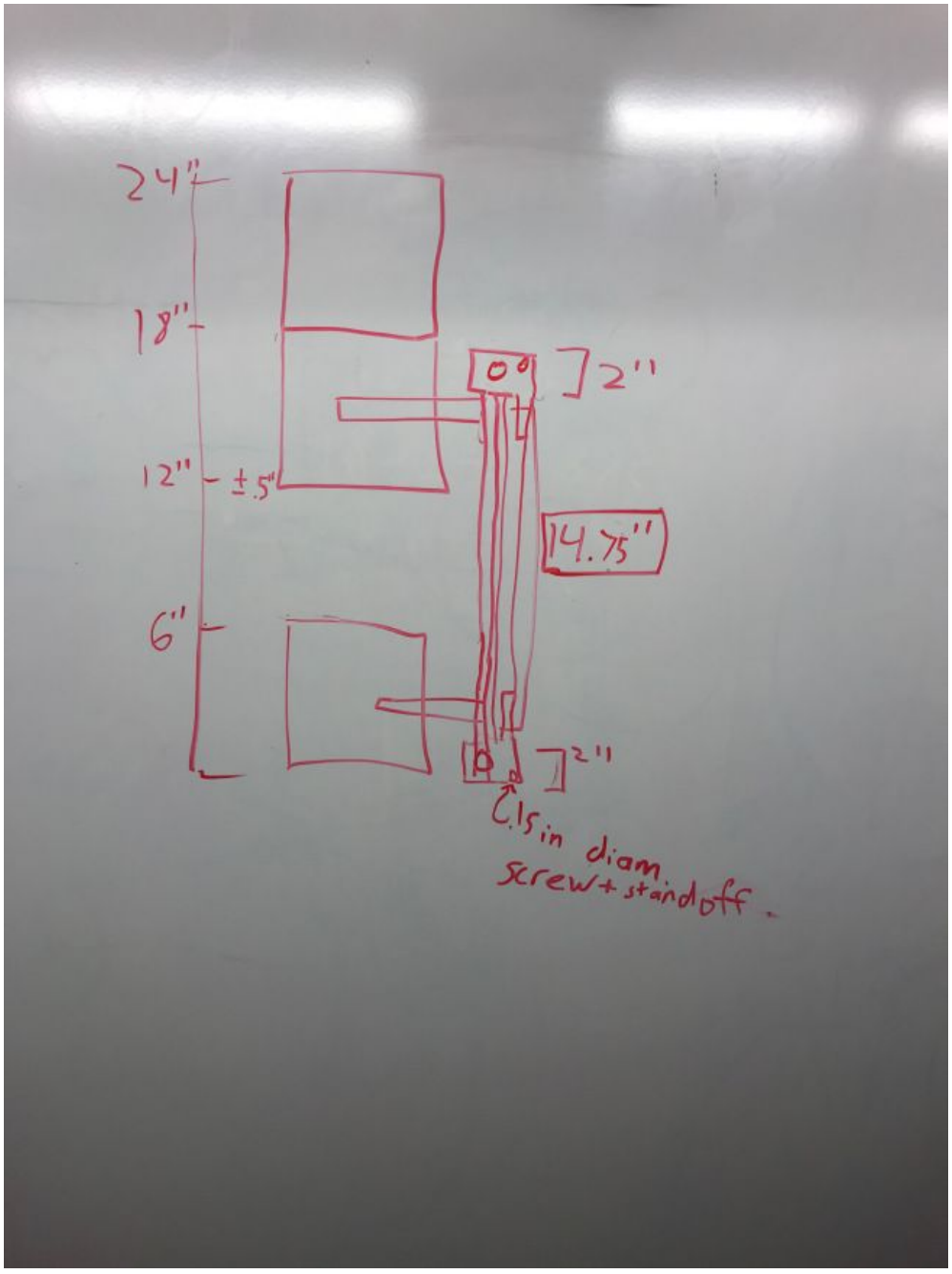
The team gathered around a whiteboard and discussed ideas and feasibility of different systems. The above diagram shows one of the designs that we believe would be most effective in terms of engineering and design. This design is our idea of a fast, effective, and sturdy lift system. This system should allow us to lift two glyphs at a time and lift them up with ease.

We decided to use a linear actuator system powered by a motor and a belt. A pair of grabbers would ride on a rail that was moved up or down via a belt attached to

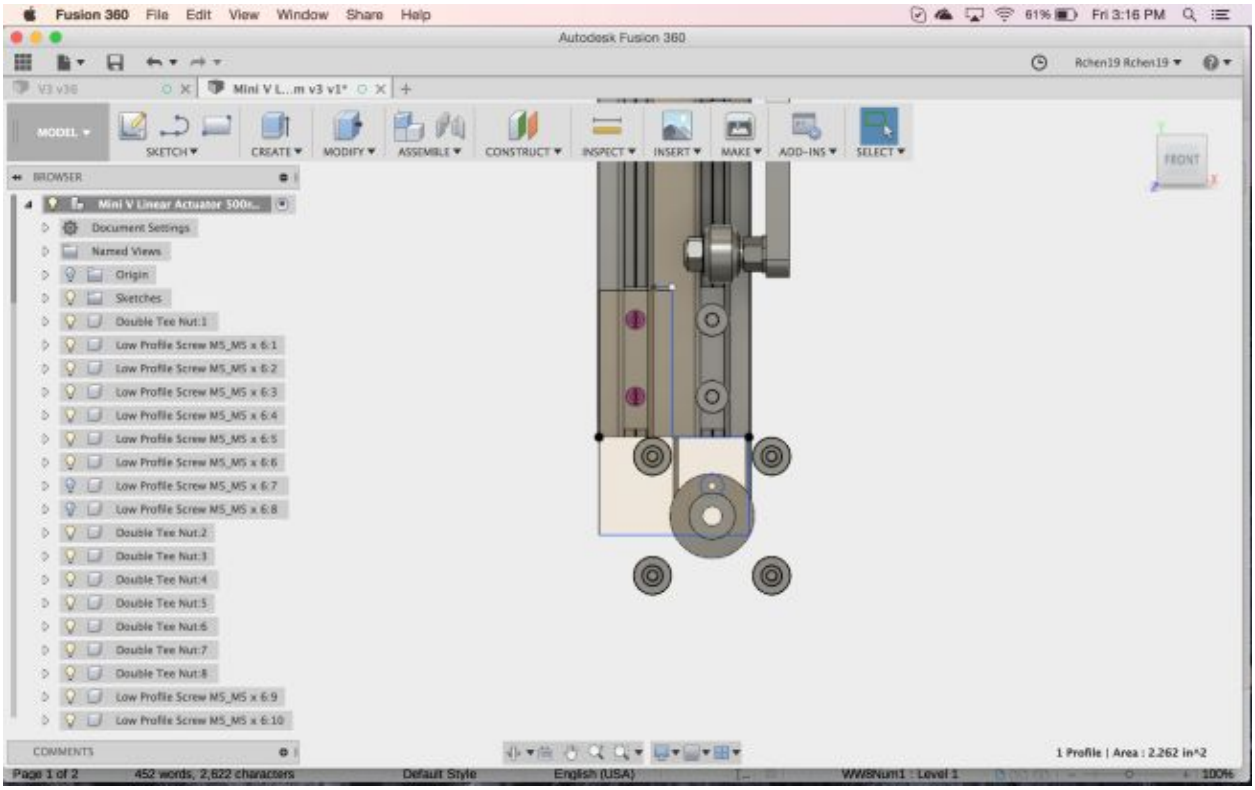
a motor. We cut down the piece of X-rail to fit with our defined dimensions. Because of the way the “cart” rolls on the x-rail, we only need one rail, saving space, weight, and mounting spaces.



This is a CAD model of the lift and the bogie in Fusion (only the near side of the robot is shown).

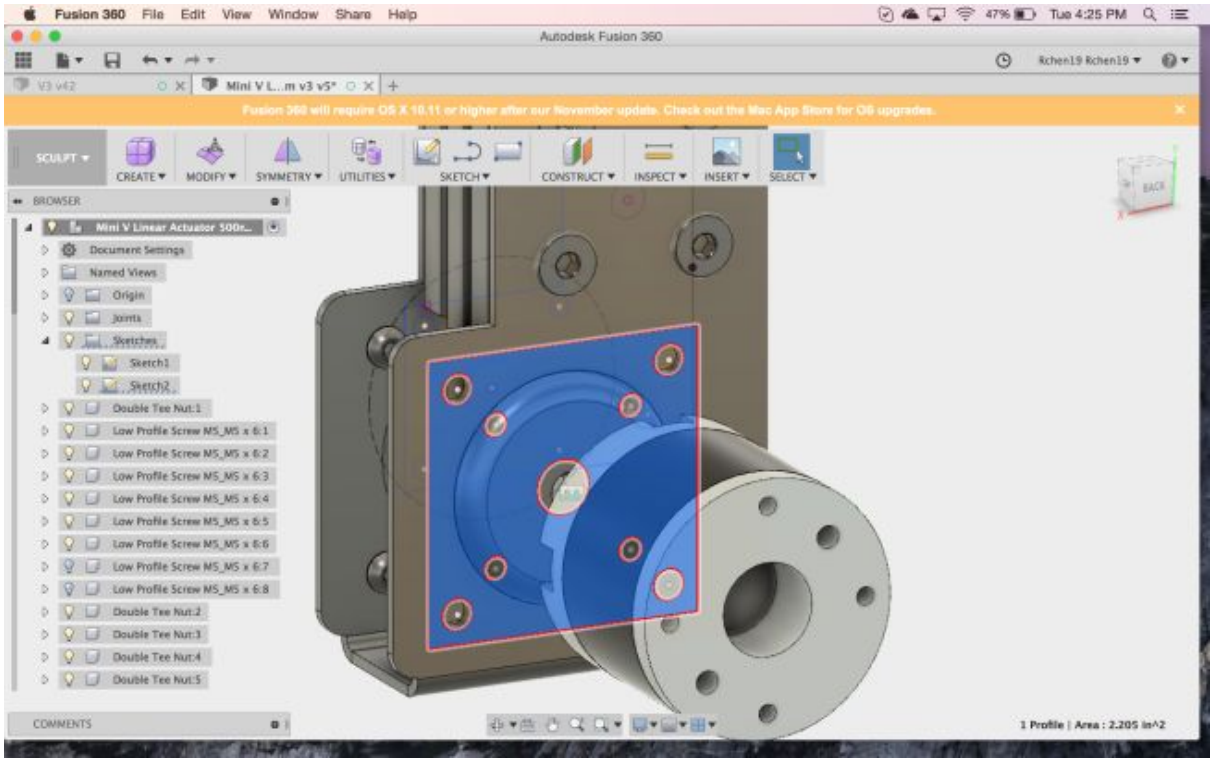


We drew out the length of each part of the lift to see how much we would have to take off in order to complete a column.

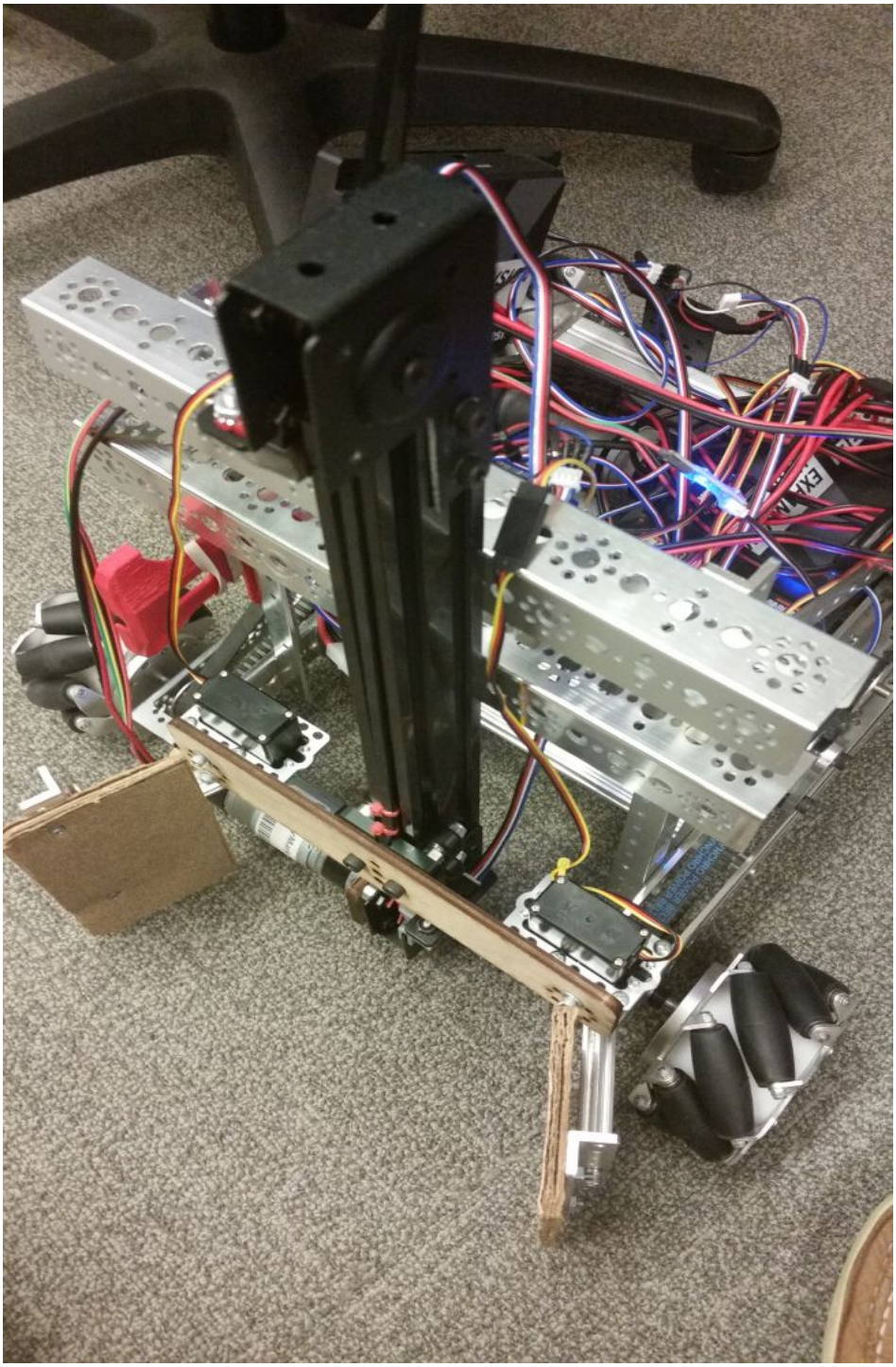


This custom plate helped us get enough room to stack higher.

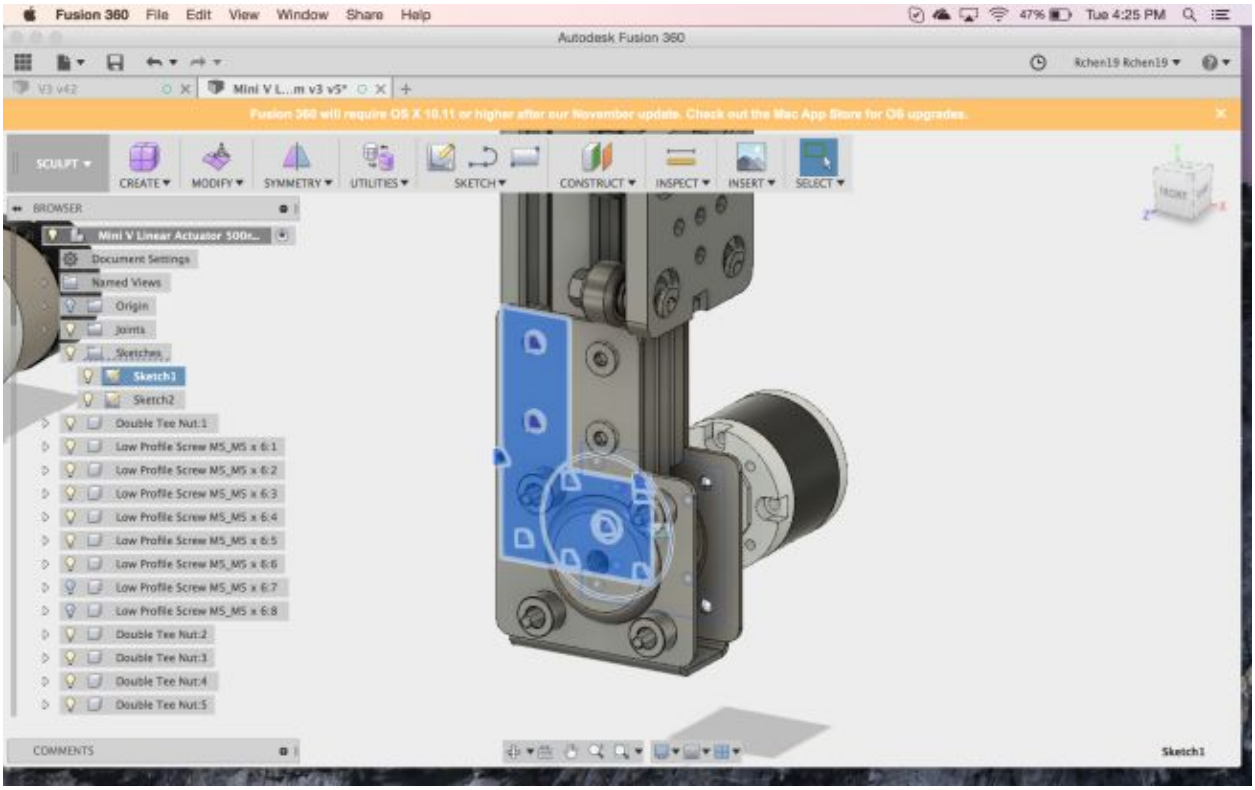
We designed custom plates to give us 14.875 inches of travel to fix the problem encountered. We also had to design a custom motor mount because the X-rail was designed for a camera slider and a stepper motor.



This is the custom motor mount plate along with a few slight modifications to the top of the channel to get a bit of extra height.



This is the first prototype of the lift with the gripper on it.



We measured in CAD and determined that we wouldn't have enough room to use our current single lift.

We want a lift that reaches the top level of the cryptobox without having to pick up a stack of two glyphs due to the difficulty and time consumption of lining up. Our linear slider does not reach this height. Even if we tilted the lift to the diagonal of the 18 inch cubed limit, we still wouldn't reach the required height, not to mention the difficulty in creating such a pivot point.

We brainstormed three ideas in order to solve this problem:

Put a linear slider on the linear slider

- Pros
    - The linear sliders are extremely stable individually

    - Reaches the required height (barely)
- Cons
    - Requires two motors
    - The additional weight of the second linear slider decreases the stability of the lift
    - Possibility of belts slipping
    - Complicated to program

Pivoting arm

- Pros
    - Easy to achieve the required height
    - Not a complicated design
- Cons
    - Previous tests show questionable stability
    - Takes up a large amount of space
    - The distance from the grip to the cryptobox changes as the arm moves up and down

Use the lift on last year's robot

- Pros
    - Proven system
    - Easily reaches the required height, can go above the required height
- Cons
    - Has a tendency to break
    - Up and down motion is not smooth

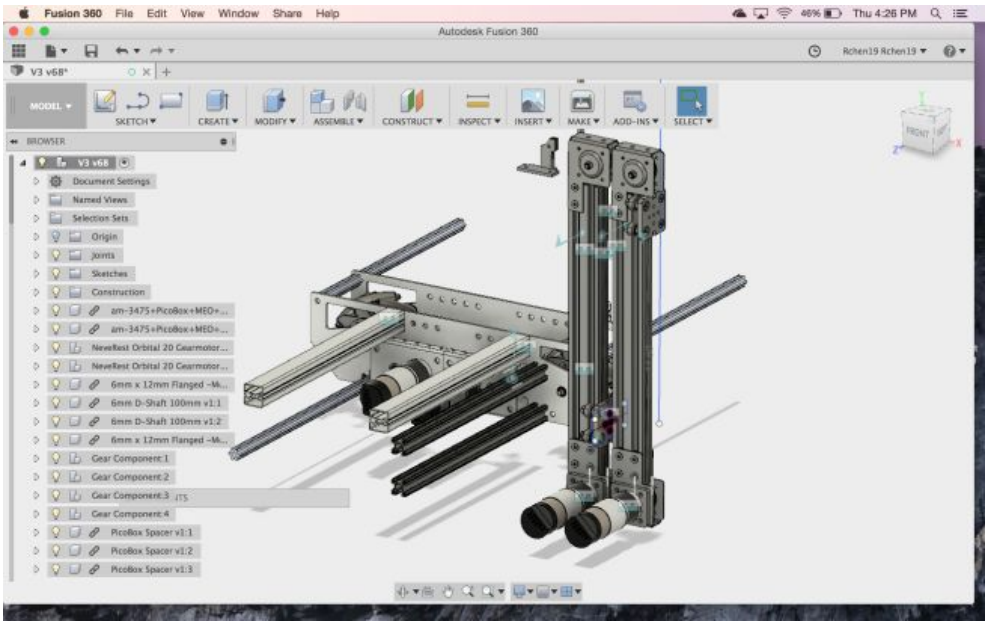
We decided that we wanted the lift to be able to reach the fourth row of the cryptobox because we discovered that stacking two glyphs on top of another

was time-consuming. To reach the fourth level, we needed a lift with a travel distance of 19 inches.

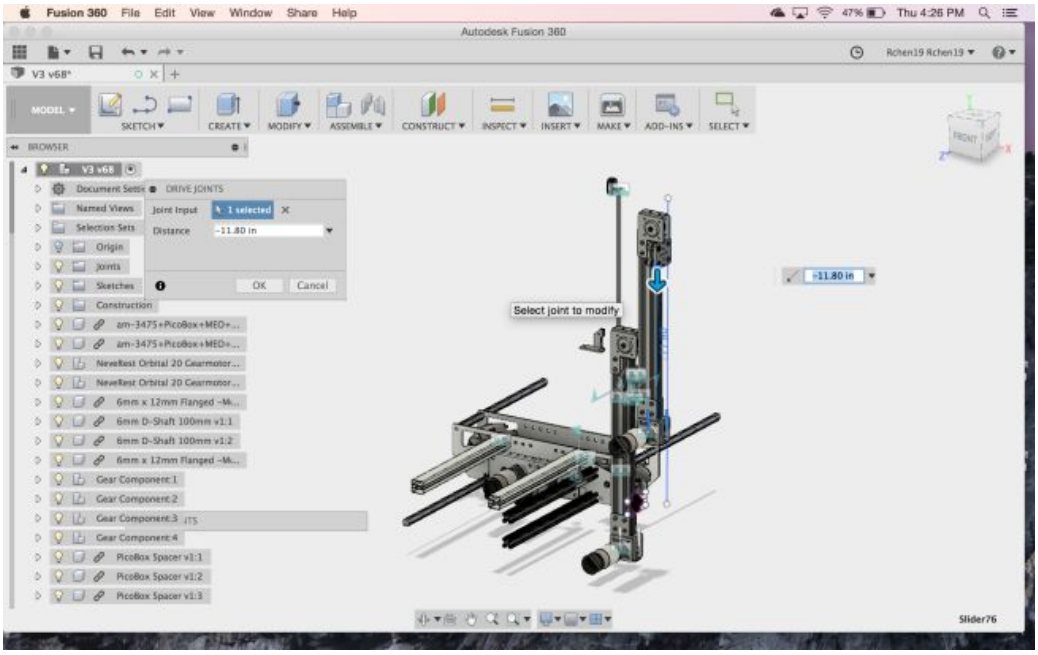
We decided to use the lift attached to the lift approach. After tightening the belts and performing some load tests, we determined that the lift can handle the weight of another lift attached to it. We extended the travel distance of the lift by cutting the plates that held the pulleys for the belt by 1.5 inches, giving us 19.5 inches of travel. The programmers said that it would be “easy” to program.

We disassembled the lift, cut the plates down, reduced the height of the lift to fit within the 18-inch box, and reassembled the lift.

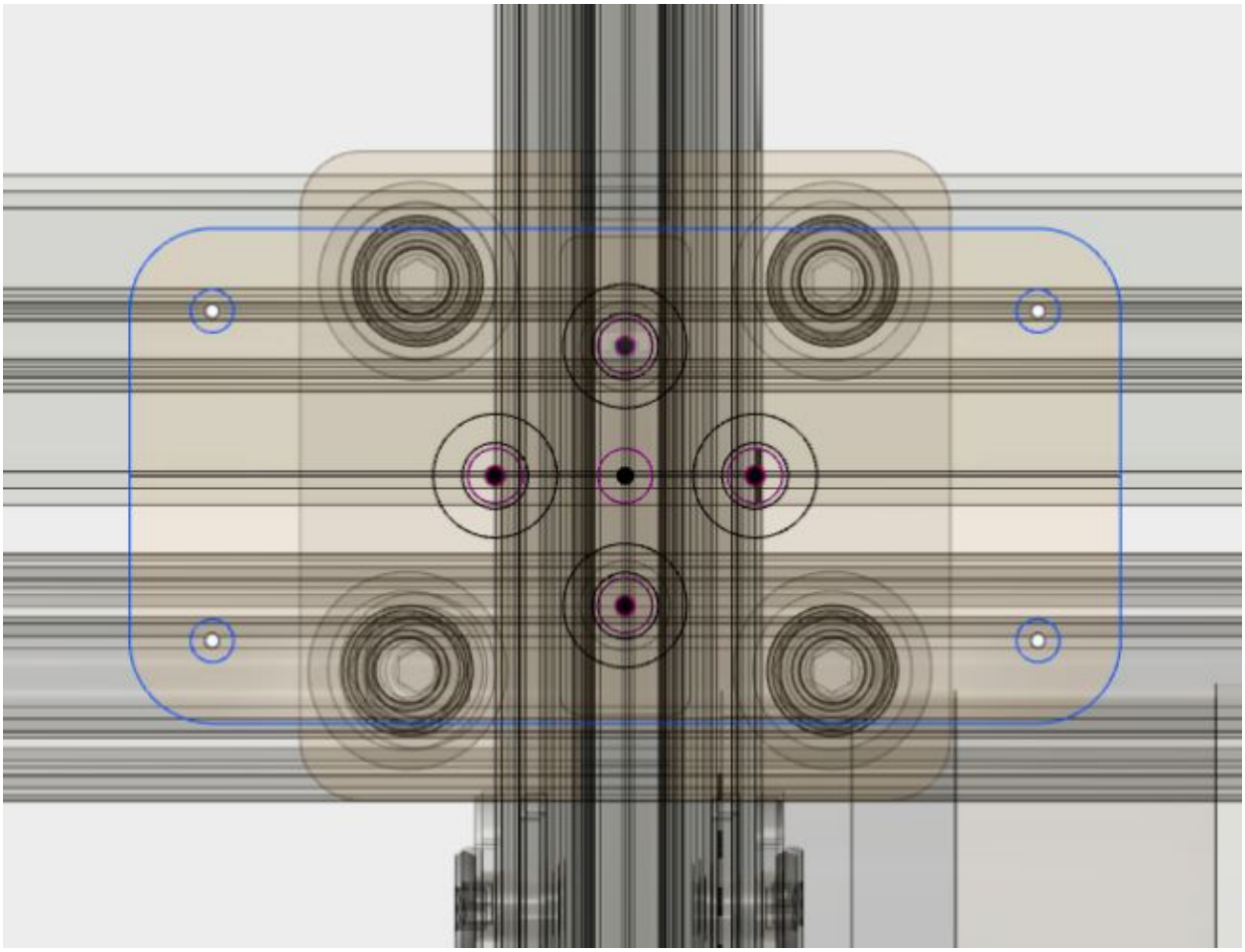
We started modeling the lift in Fusion to make sure our design would work out, before we buy the second linear slider.



This is the design of the double lift concept in CAD.



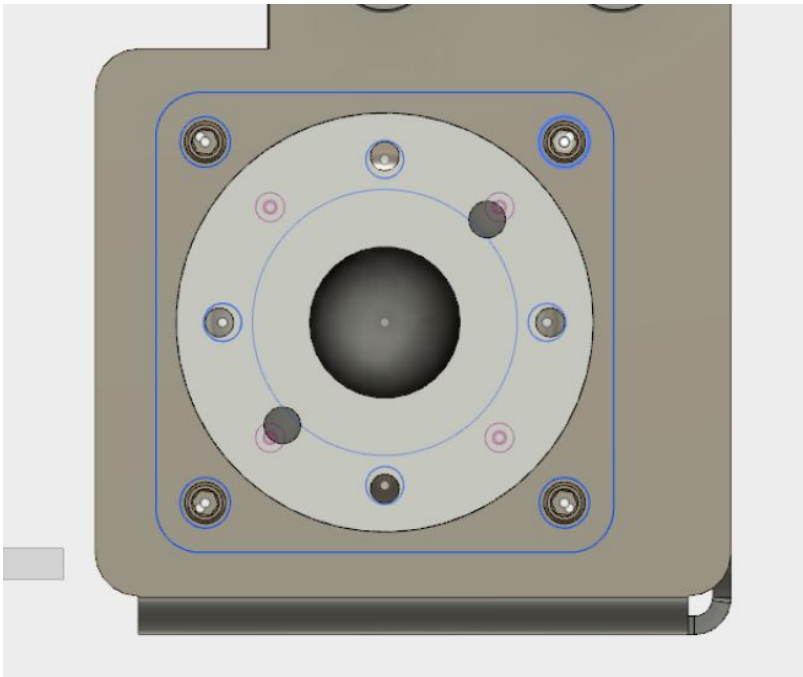
We added joints to the system to make sure nothing hit anything else while moving up and down. Everything cleared so we ordered the second lift.



This is a design of the plate that connects the two lifts in CAD.

This is a wooden prototype of the connecting plate.

We found out that although we cut the lift down to 17.5 inches, we had to raise the top plate and the pulley attached to it to tighten the belt, pushing us outside of the 18 inch box. We had to disassemble and cut the x-rail down again.



This is the motor mount for the second lift designed in CAD.

Made a new motor plate for the second lift because the holes on the original CAD model were not the correct size.

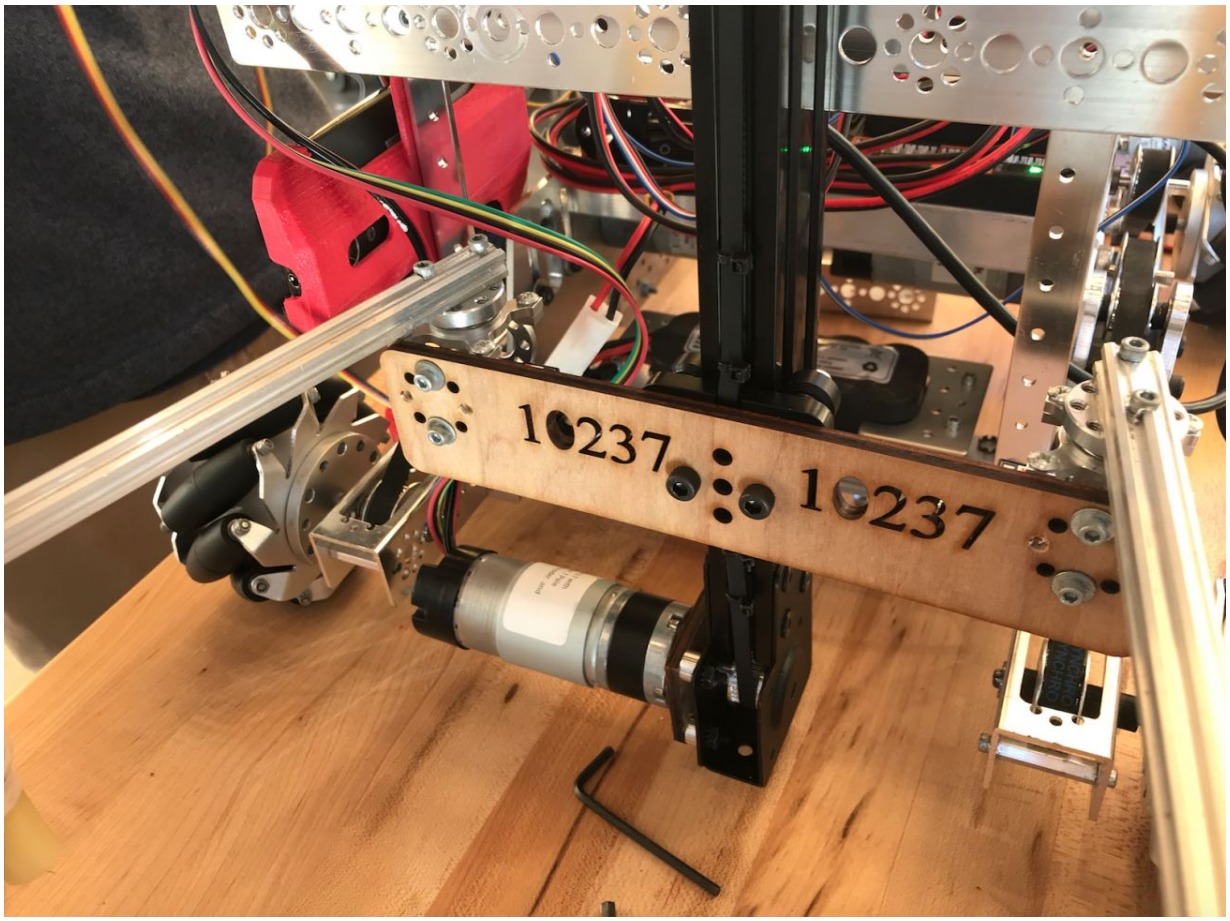
Once the second lift arrived and we built it, we identified a few problems with the design:

- The belt needs to be tighter because it kept on slipping
- The holes in the plate connecting the lifts together are too large, resulting in a wobbly second linear slider
- The wooden plate is also bowing in the center

Thus we decided to design a new plate with tighter tolerances and less space between the holes to help to reduce bowing and wobbling.

Gripper Design

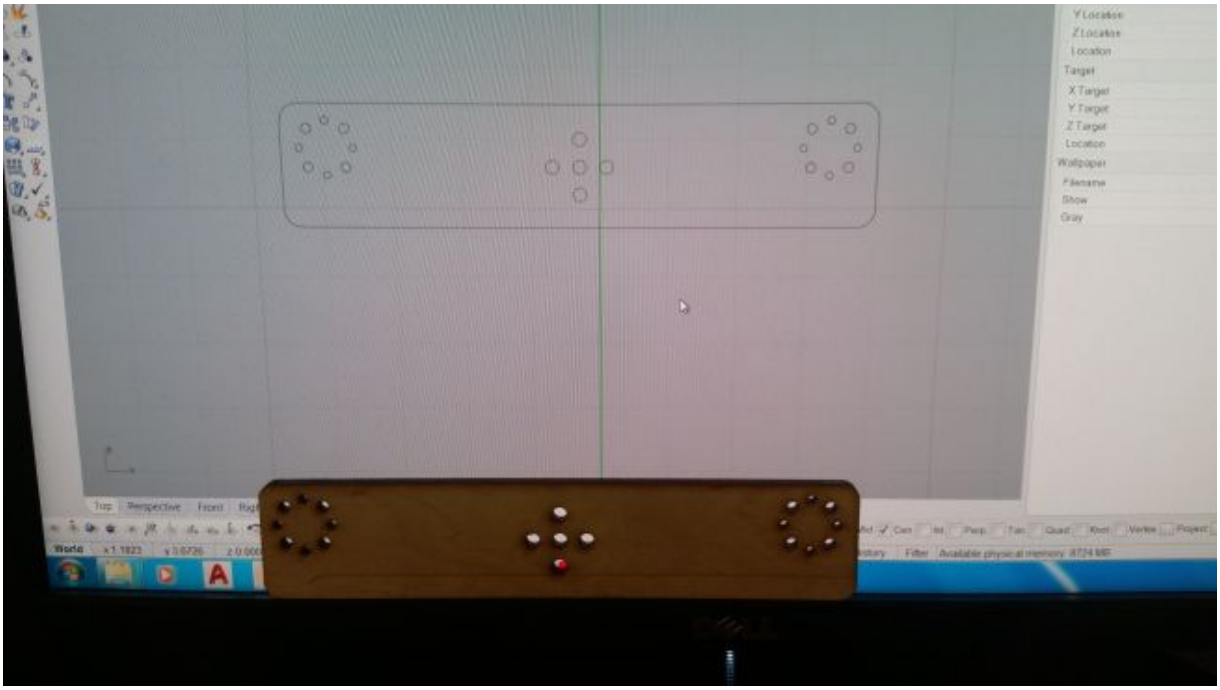
We decided on having a gripper attached to a lift. We designed a plate that would have two servos, each of which would have an arm on them. We designed this plate in CAD and printed it out of wood.



This is the first gripper prototype. We attached another metal rod with surgical tubing to act as the actual gripping surface.

We attached metal rods to the servos and metal rods perpendicular to those with surgical tubing to grip. This worked decently, but the servos were too close

together so it wouldn't grip the glyphs properly. To solve this, We lengthened the bar connecting the two servos.



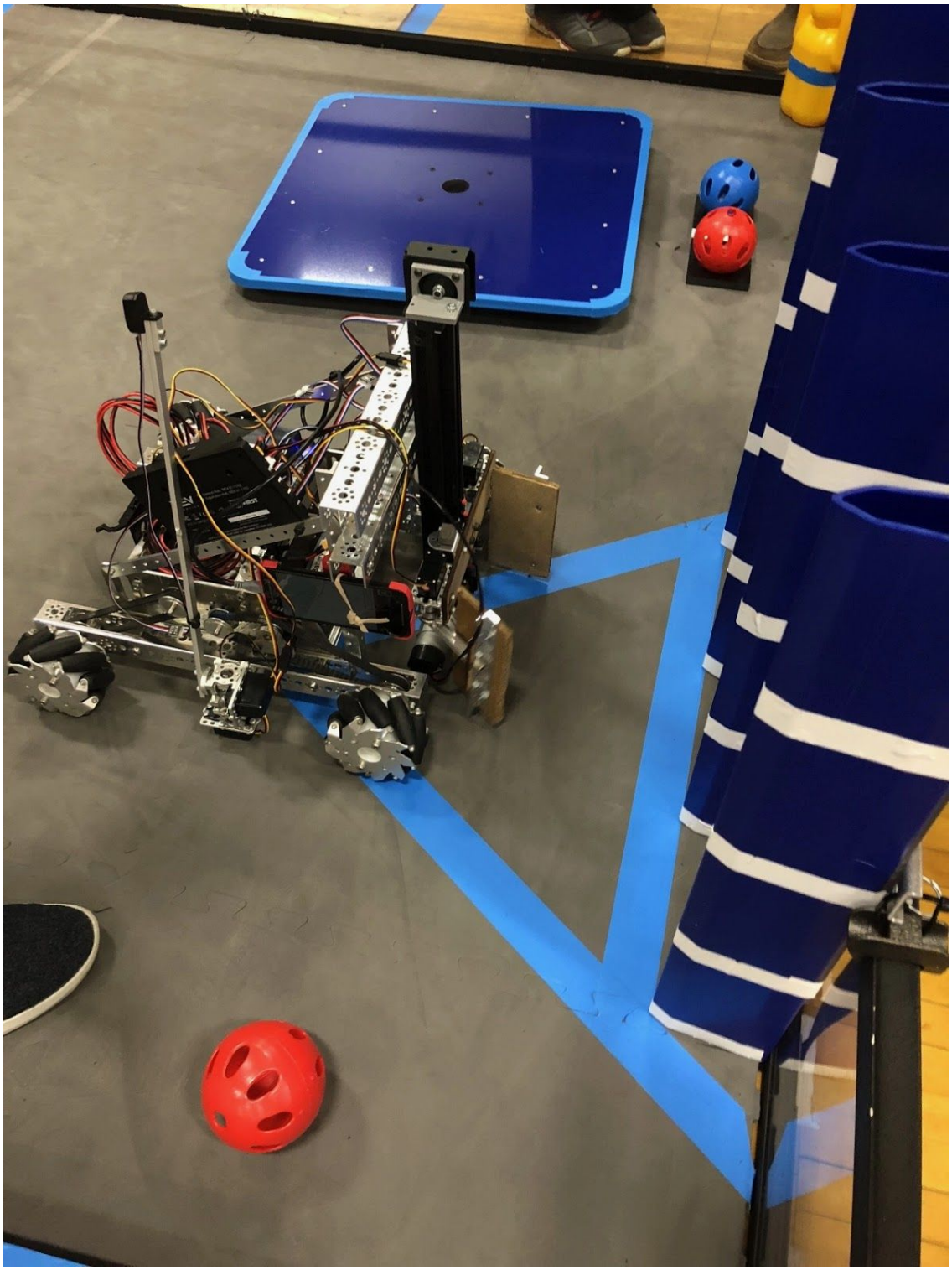
This is the redesigned plate.

We attached the wooden plate for the gripper to the lift. We proceeded to test out the design on the field. The lift worked smoothly, however we are unable to reach the top row of the cryptobox. Also the metal rods and surgical tubing didn't grip the glyphs well.

We addressed the problem we were having with gripping the glyphs. We removed the surgical tube we had attached. We then added cardboard plates to increase our surface area. We lined the cardboard with sandpaper to see if a rougher surface would help keep the glyphs in place. We later found out that sandpaper is illegal.

The tests were successful, however we needed to replace sandpaper with a competition legal material. We tested the gripper in further detail at the Solon Scrimmage and during practice.

We continued to utilize the chassis from our mecanum robot. This allows us to quickly iterate lift and gripper concepts.

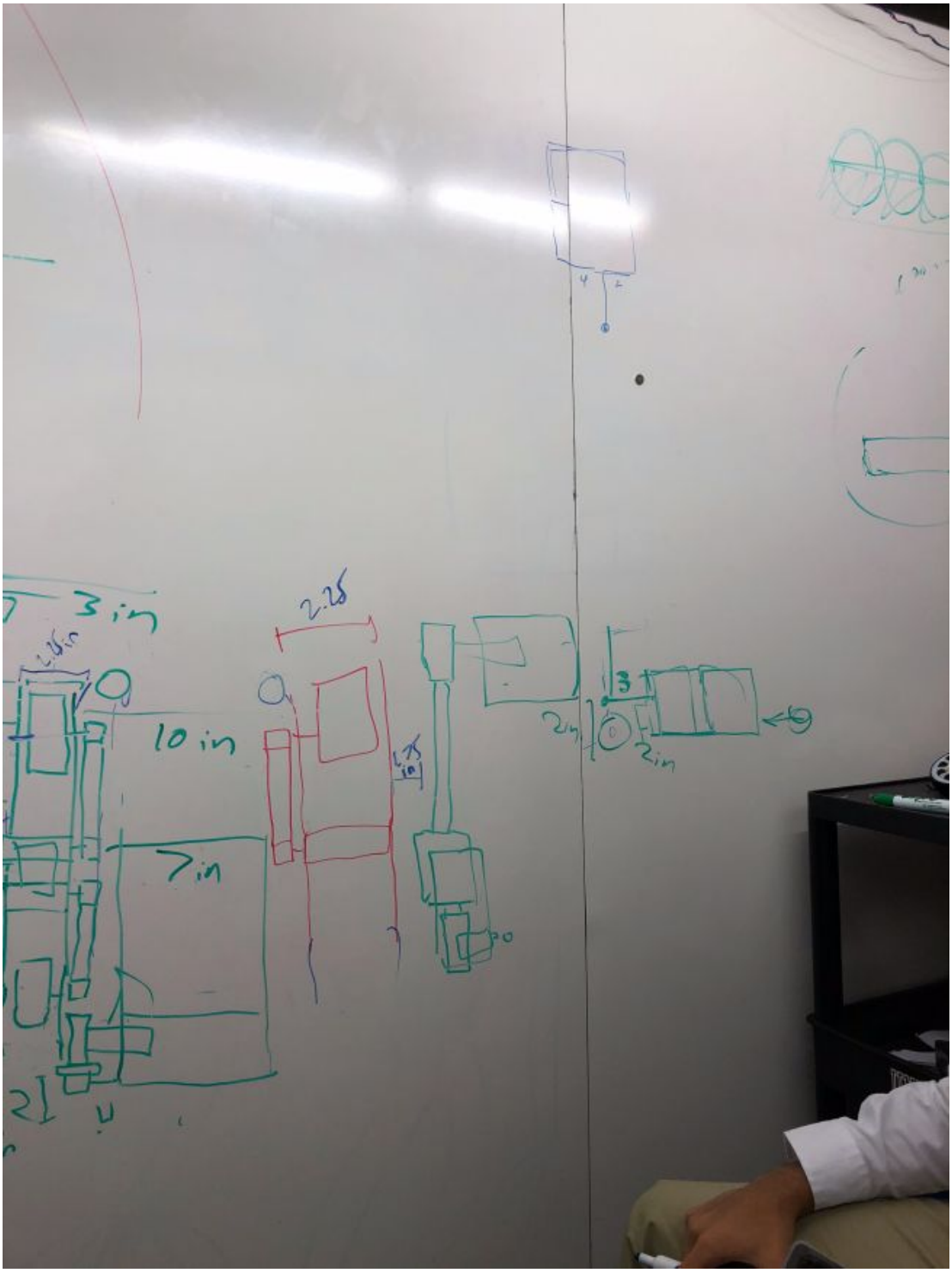


The final gripper prototype at the Solon scrimmage.

Mid Season: Introducing Parallel

Why Change

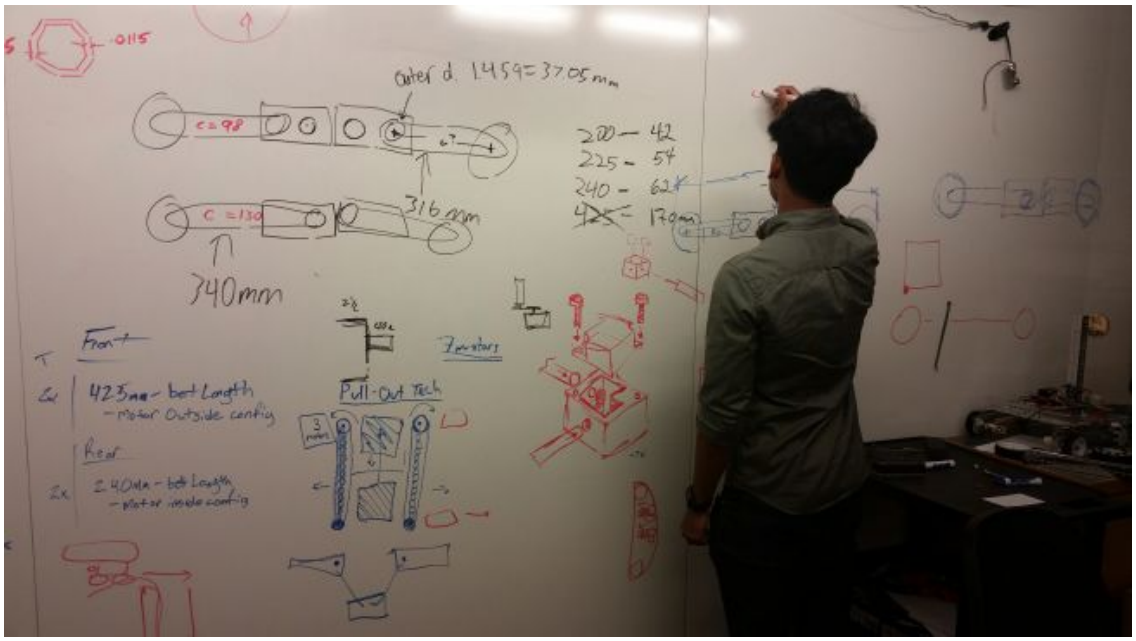
The whole team gathered to discuss what we plan to do moving forward. We discussed our current robot's capabilities and what we'd like to accomplish. We decided that it would be difficult to use our current platform and design to complete a full cryptobox with cipher quickly. We decided to create a paddle that would rotate 90 degrees and flip 2 glyphs into the cryptobox. This, in theory, will help us reduce the amount of time spent harvesting and moving the glyphs. We decided to keep the mecanum wheels because of the benefits they give us with lining up to the cryptobox.

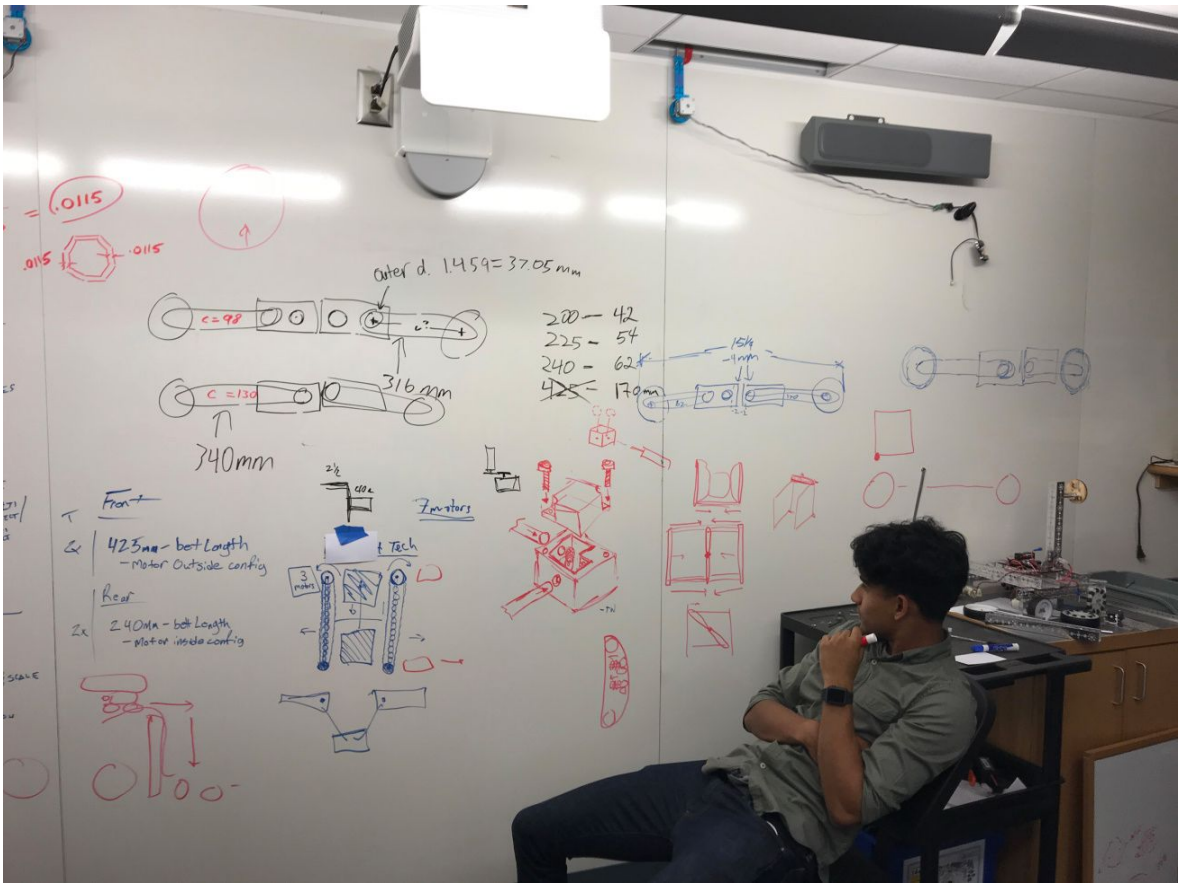


These were the initial sketches of the new robot.

The Mecanum Chassis

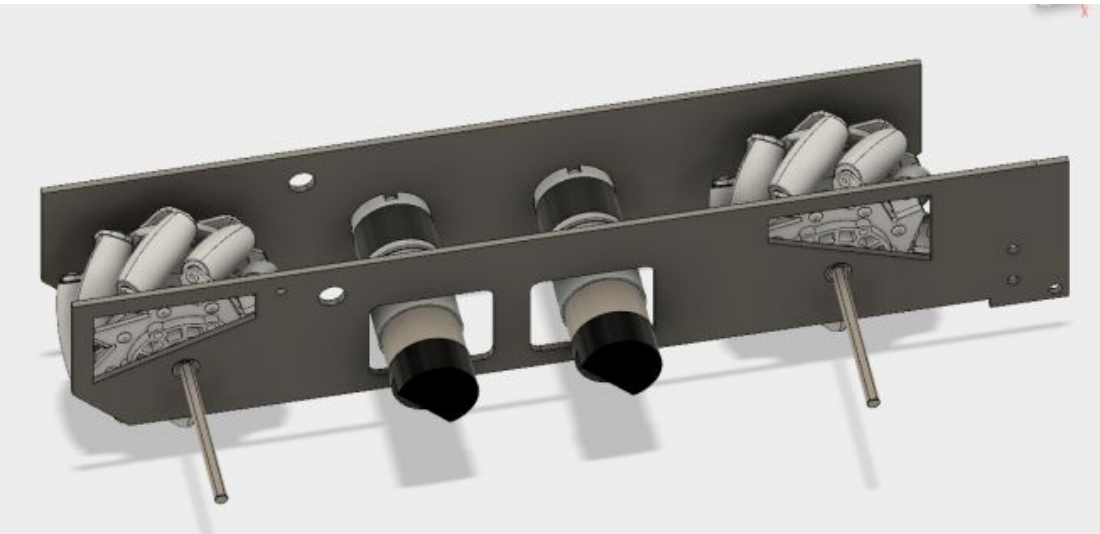
We decided to use the general form of the TileRunner. We would have four parallel plates with the mecanum wheels sandwiched between the two plates on each side of the robot. We would mount motors facing inward in the middle of the robot and we would use belts to transmit the power to the mecanum wheels. Because Andymark sold belts with lengths of 200, 225, 240, and 425 mm, we had to revise our design and do some math to determine which belts we need and the spacing for our gearboxes.





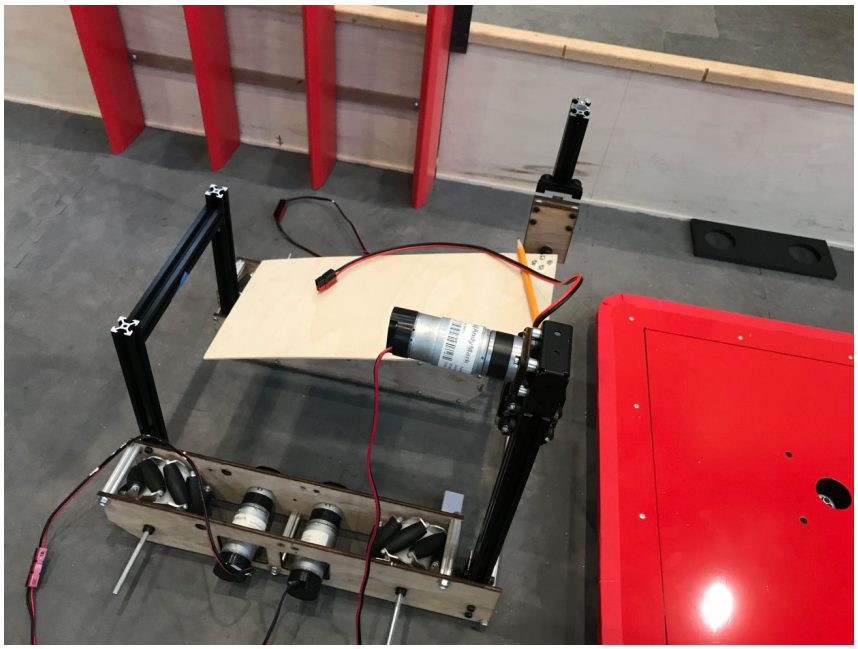
Our mentor, Arjun, helped us with getting the correct spacing for our chassis so that the belts would be taught.

We then started designing the side plates in CAD. We designed one side and then mirrored it.



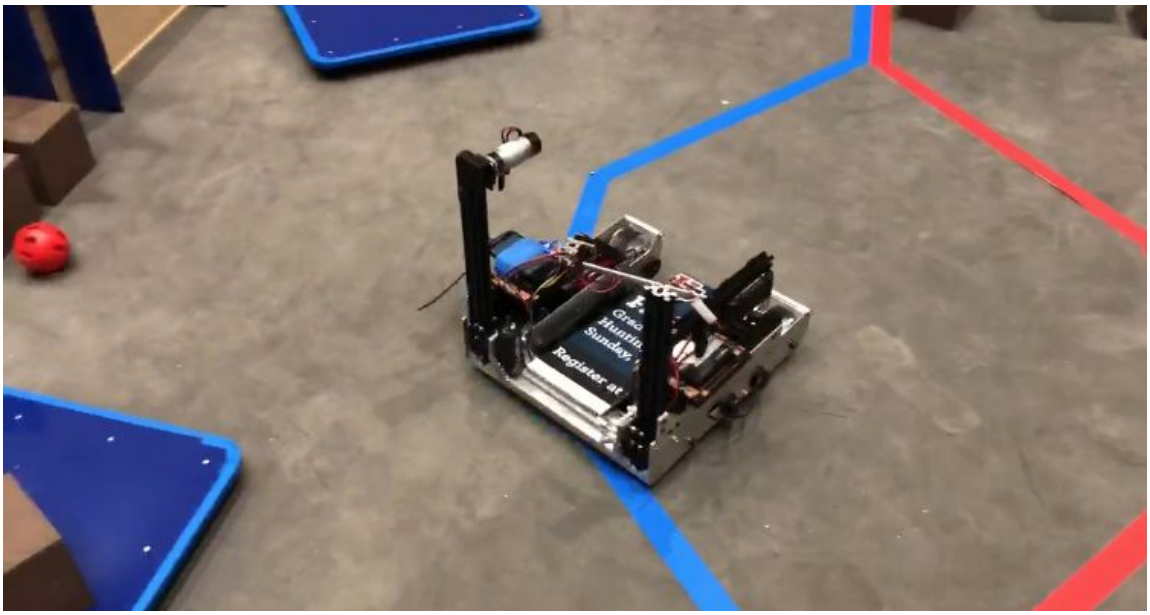
This is the CAD design for one half of the chassis.

We prototyped this chassis out of wood. The belts took a long time to come in, so we continued to design the flipper and lift design on this chassis. We found that we would need peanut extrusions running between the two sides because the two churros that originally connected the two sides of the robot made it very flimsy.



This is the wooden prototype of Parallel with the flipper attached.

Once the belts finally arrived, we tested the robot with the wooden frame. We found that using the AndyMark 3.7's made the robot very fast and uncontrollable. However, all of the spacing worked and we incorporated holes for support to connect the two sides so we made the chassis out of metal.

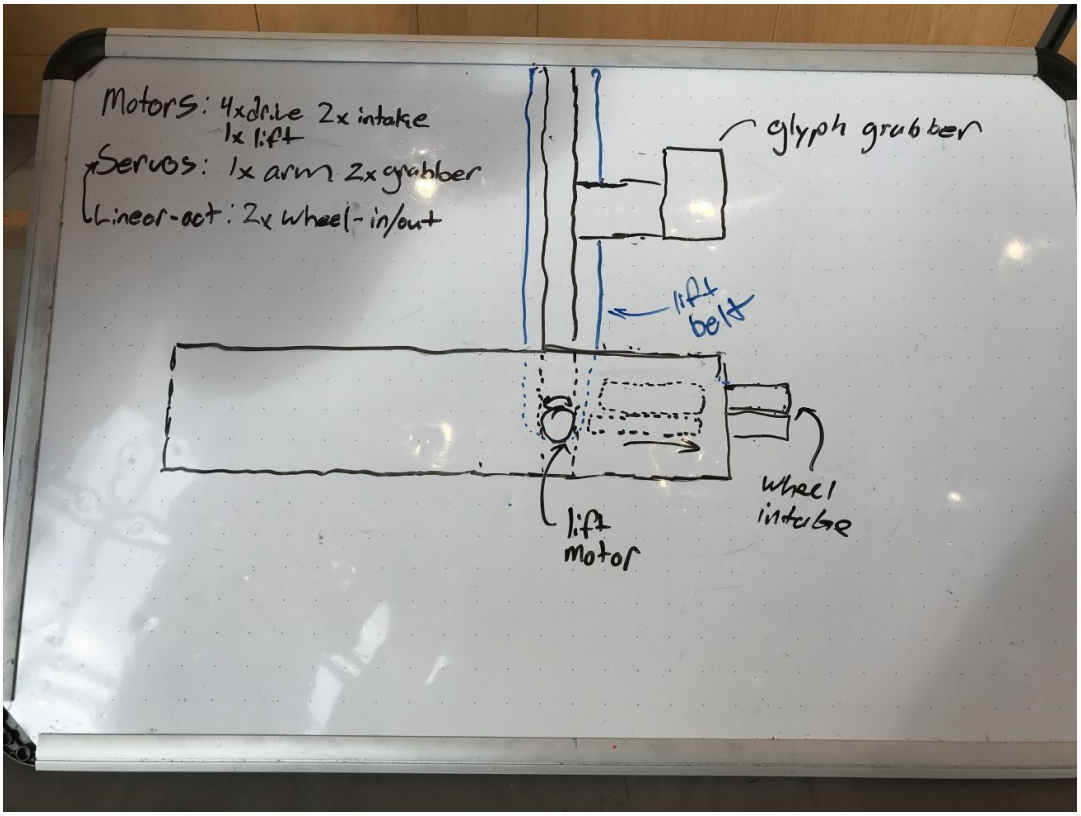


This is the metal version of Parallel driving around the field with the flipper and lift attached.

We decided to switch to the AndyMark 20 motors for the drive motors. Although slower, these motors still provided ample speed and were much more controllable.

Harvester Design

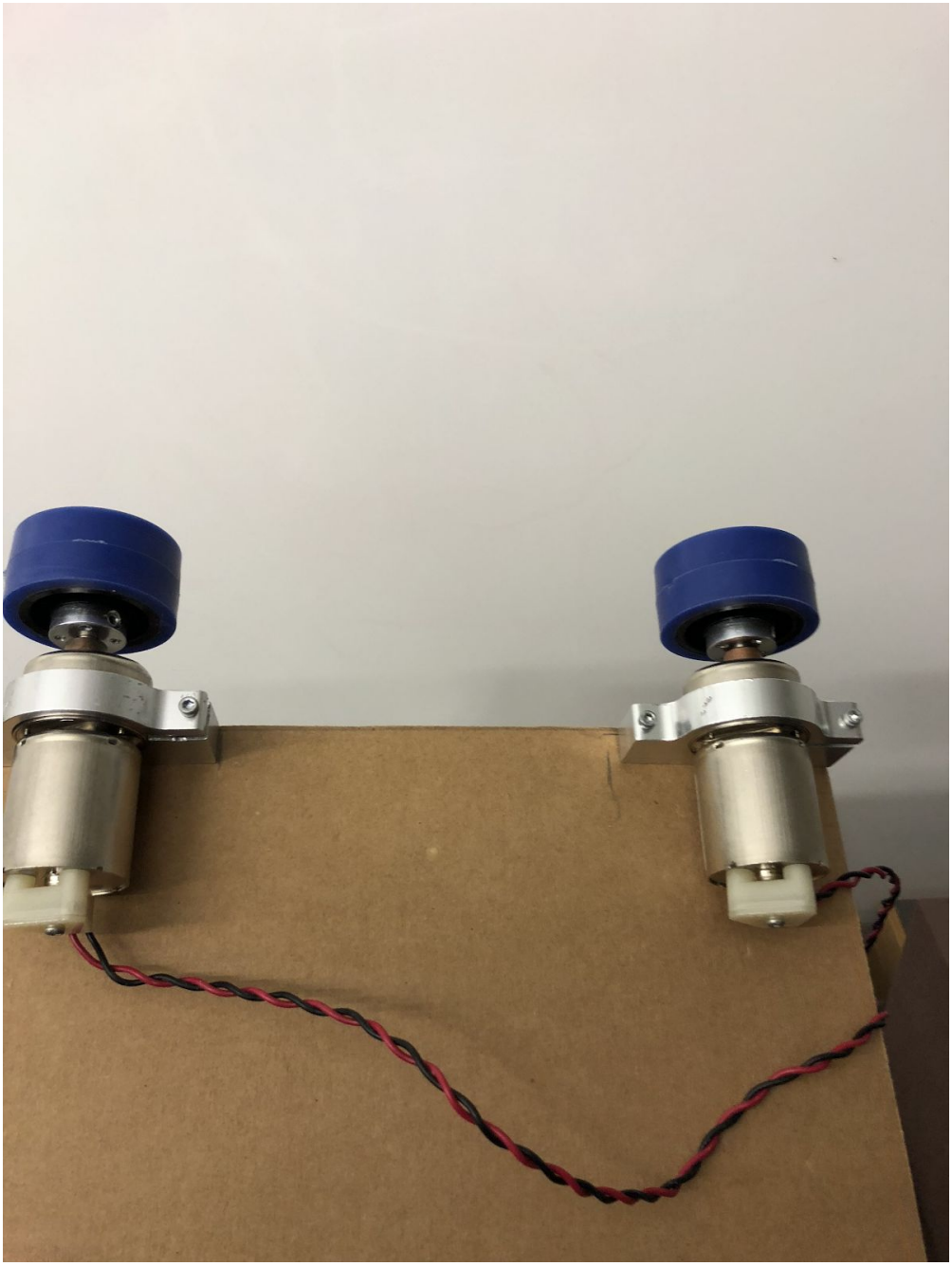
We decided on a harvester design for the glyphs because it would be much faster than having to align and grip each glyphs individually. Initially, the two wheels would extend out of the robot to harvest the glyphs. These two wheels would put the glyphsinto a lift. We planned to have this harvester feed into the gripper design. Thus, the harvester would be on the same side as the gripper. This meant that the harvester would have to retract.



This was the initial design for the harvester. The wheels would harvest the glyphs and the grabber would grab it from there. The wheels would also retract into the robot so we could place the glyph in the cryptobox.

We decided to use the Rev hex motors to independently spin two sets of wheels so that we could effectively harvest glyphs at an angle.

Shortly after this design we found the idea for a flipper. We realized that with the flipper idea, we could have the harvester put the glyphs directly onto the flipper. This meant that we didn't need to linearly retract or extend the harvester wheels. However, in order to fit within the sizing cube we did need to fold the harvester up and let it fall down by gravity at the start of the match.

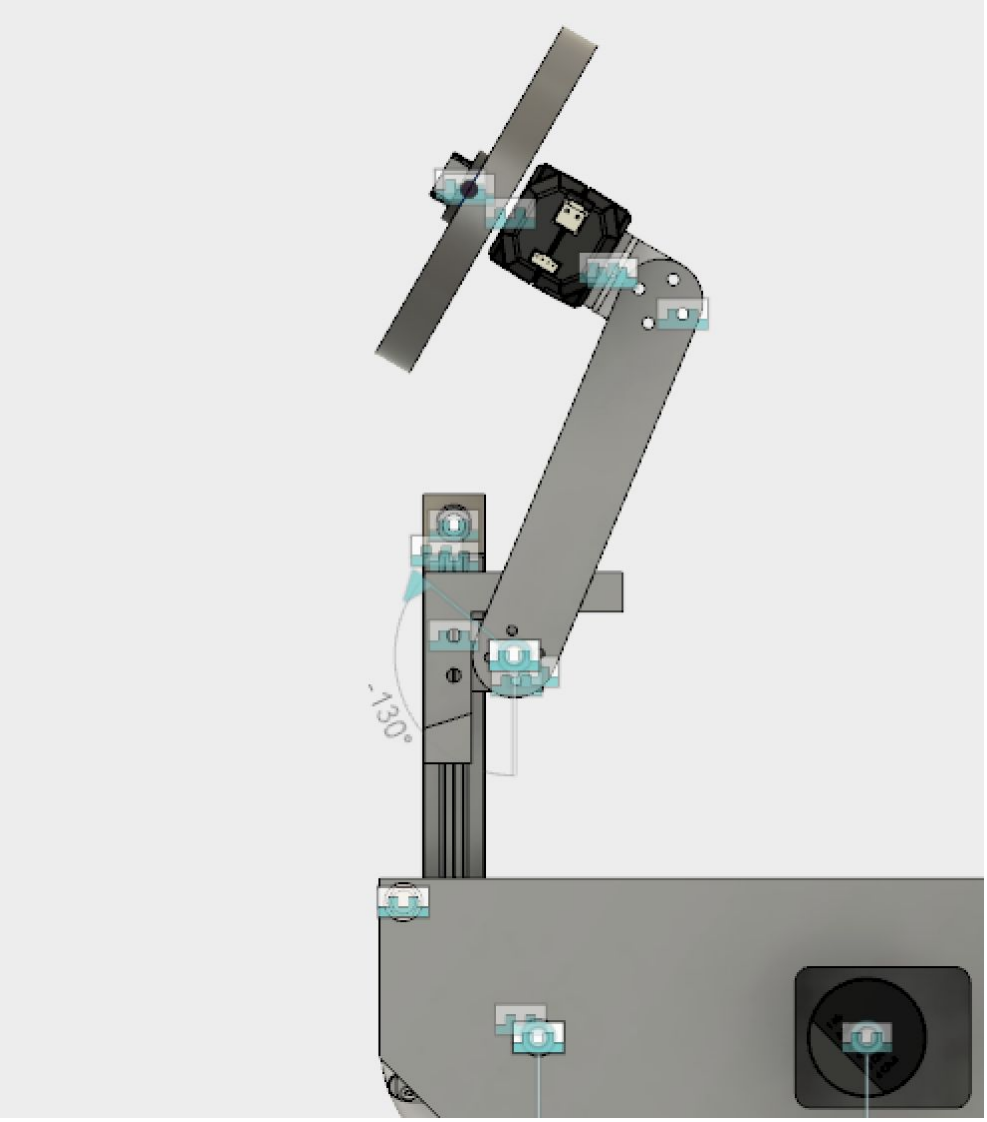


A first rough prototype testing the feasibility of a wheel intake.

We also wanted to mockup a harvester to see if wheels could be used to collect the glyphs. We used motors and wheels we had from previous years attached to cardboard. The design was nowhere near perfect because the wheels were near

the very top of the glyphs and the wheels were very hard.

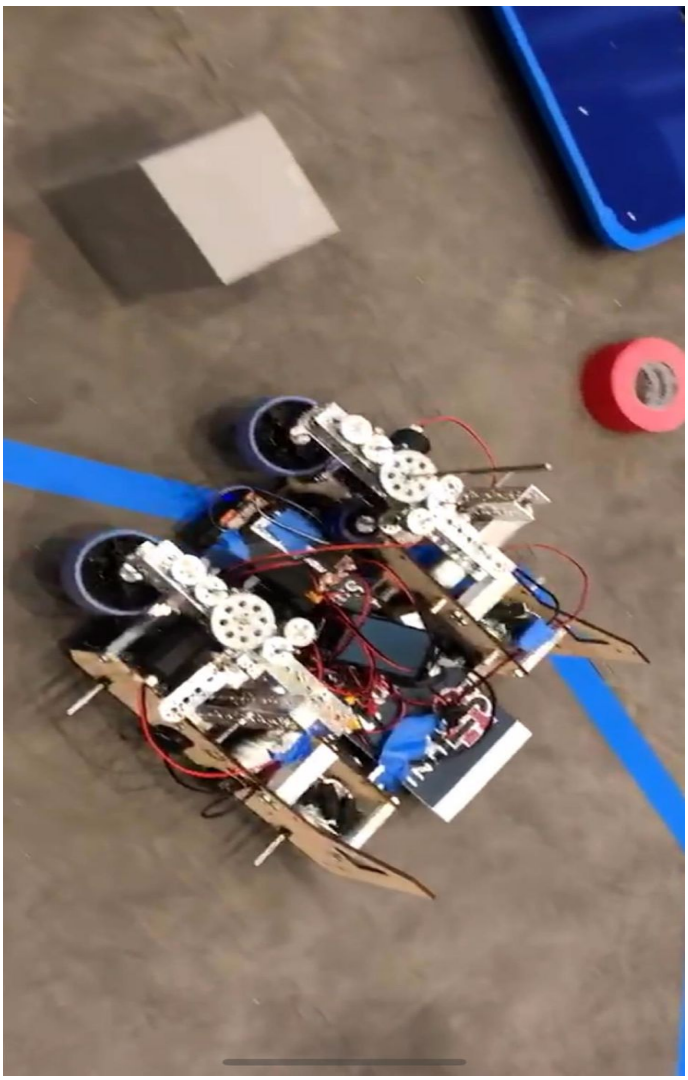
The model did show us, however, that with a proper design wheels could be used to intake glyphs onto the flipper.



This is a CAD model of the harvester design. The whole assembly falls down at the beginning of the match due to gravity.

These are the star wheels that we tested.

Although the idea of non-rigid intake wheels was promising, these wheels were too squishy and didn't grip the glyphs at all. We then tried AndyMark's green compliant wheels. We used both the small and large compliant wheels in the intake.



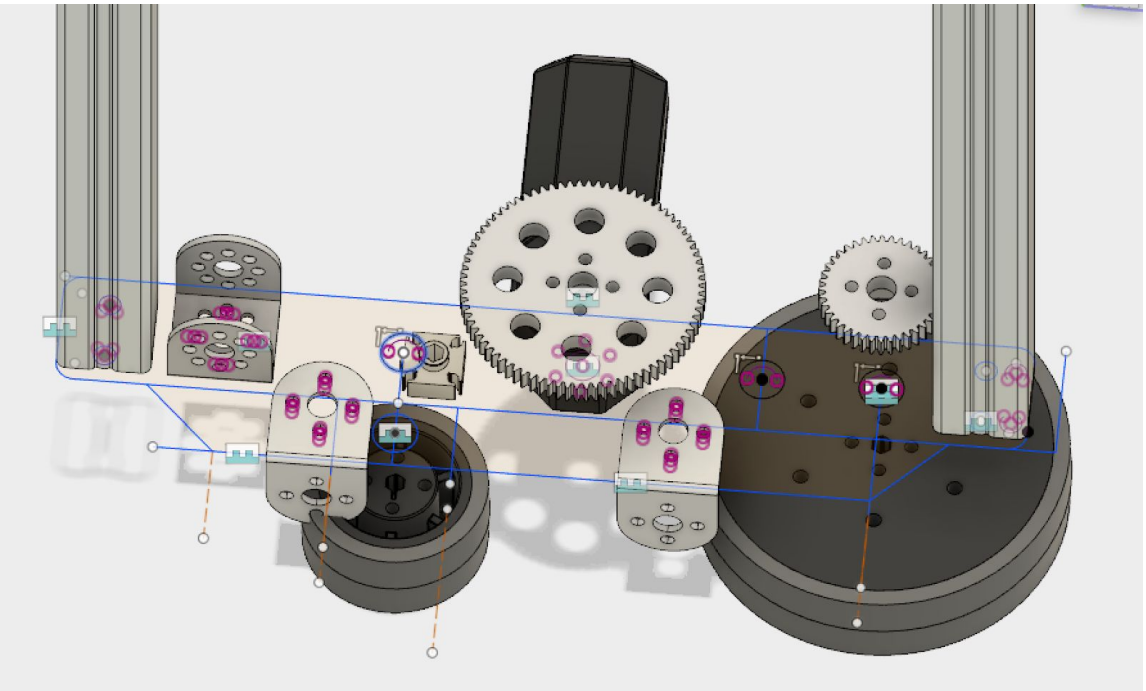
Prototype harvester using Andymark flywheels and tetrix parts.

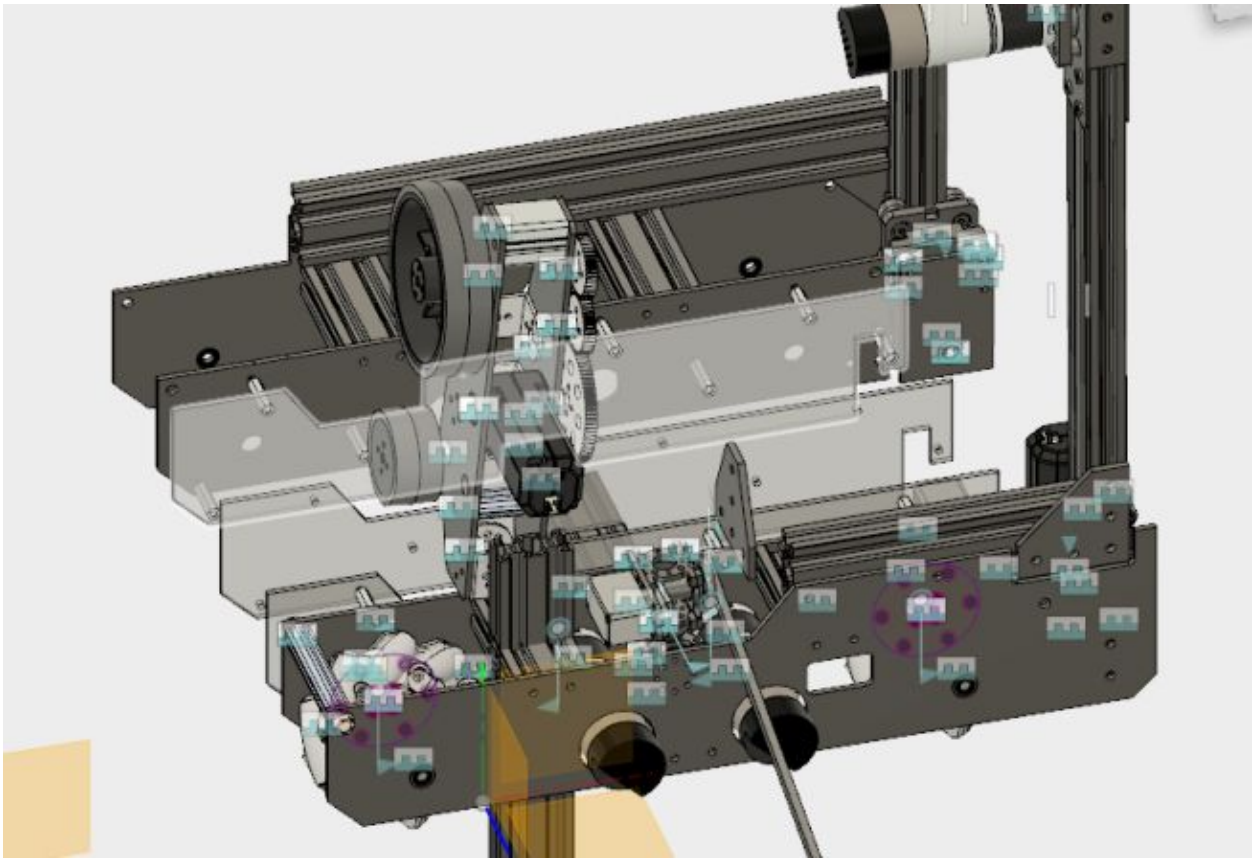
We decided that only using one pair of wheels was not good enough to quickly harvest two jewels. Instead, we want 2 pairs of wheels connected by a belt.

Mike created a prototype using tetrix pieces. It uses 2 wheels on each side, a 4 inch wheel and a 2 inch wheel. They are connected using gears. The second pair

of wheels gives the robot better harvesting capabilities. The gears are in a 2-1 ratio, increasing the speed of the wheels (the Rev Core Hex motors have low rpm but high torque).

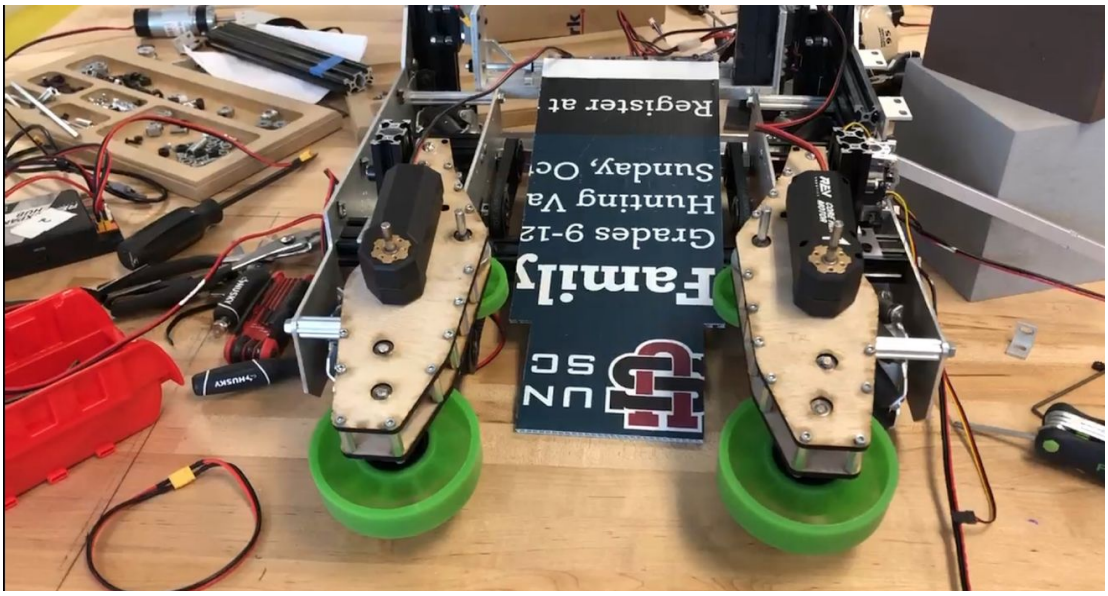
After building a physical prototype we thought that using wheels is the best way to collect glyphs into our flipper. Because of the tolerances on the glyphs, we decided that softer wheels would be better for harvesting the glyphs.





We first designed the harvester in CAD and added it to the chassis.

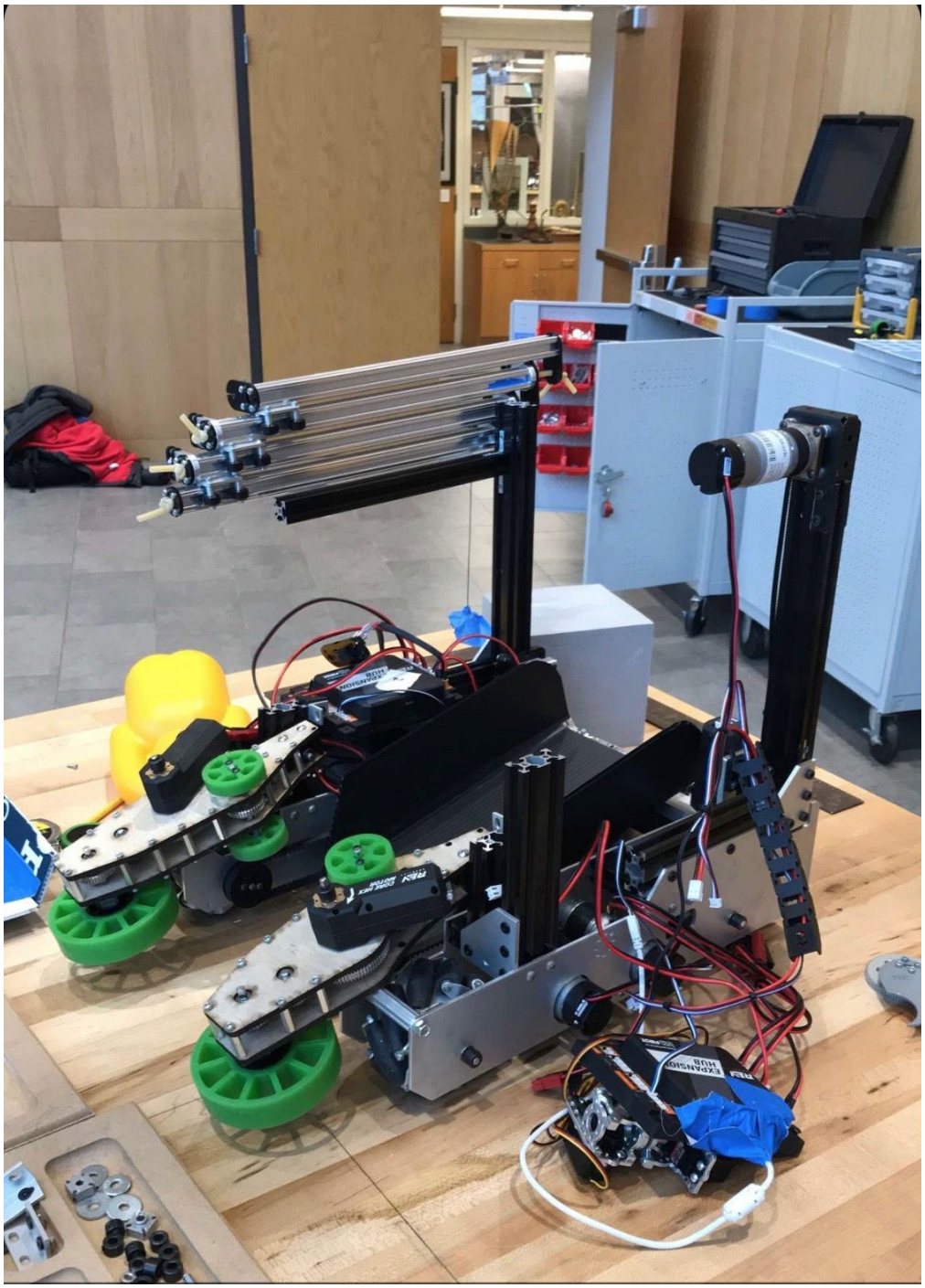
We tested out this new harvester design with hard wheels. It performed much better than the single wheel design.



This is the compliant wheel harvester prototyped out of wood.

This design was very effective. It enabled us to harvest the glyphs at an angle. However, since the wheels were not harvesting at the center of mass of the glyphs it often caused the glyphs to flip as it was going onto the lift.

To fix this, we changed the plates and their positioning so that the wheels harvested more at the center of mass of the glyphs. We also attached another set of small compliant wheels to the same axle as the first set of compliant wheels.



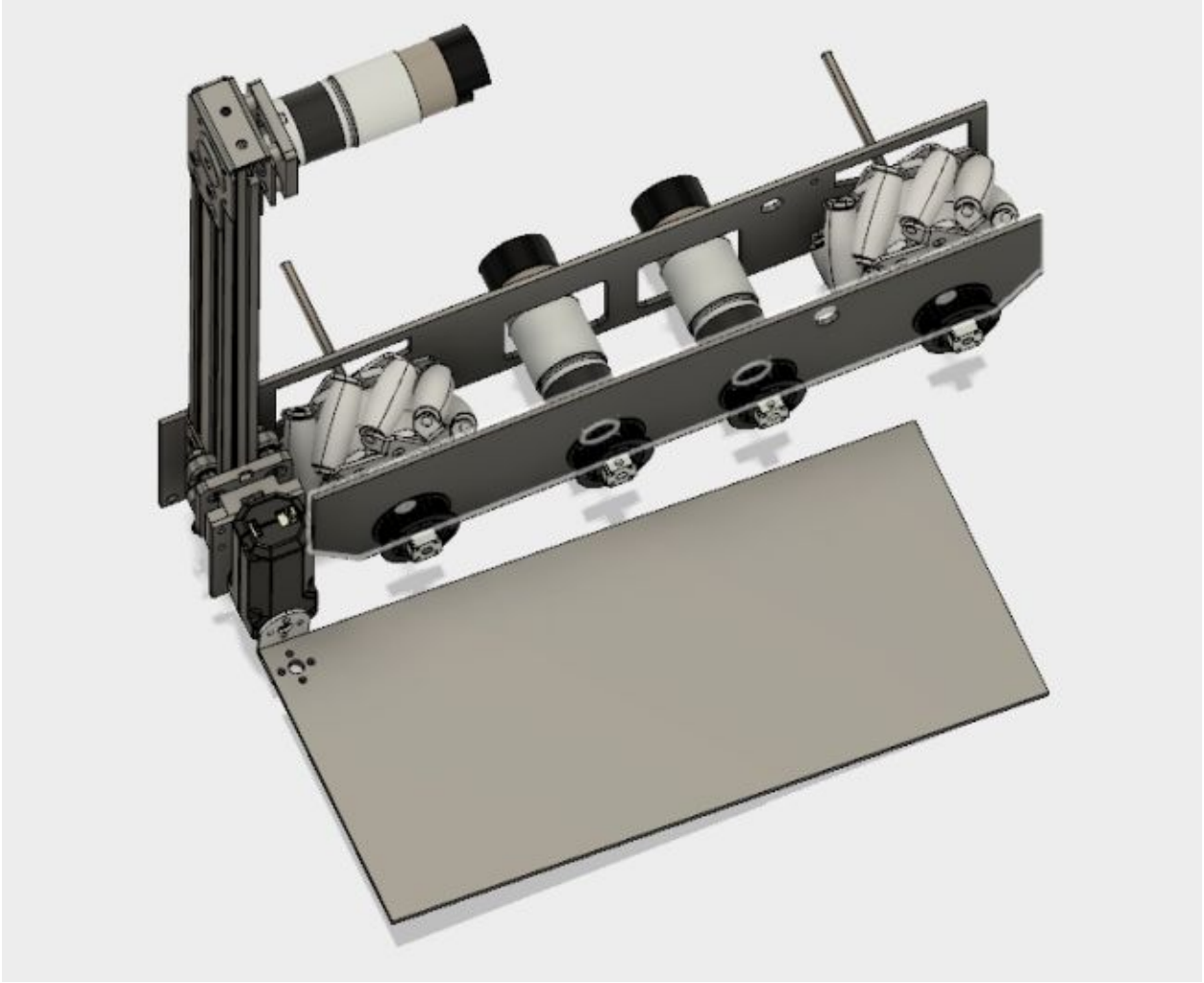
This is the final wood prototype of the harvester.

This design for the harvester was very effective and fast. By controlling the

speeds of the two motors, we could harvest at an angle and straight-on. Finally, we made the plates out of metal.

This is the final iteration of the harvester before Cleveland.

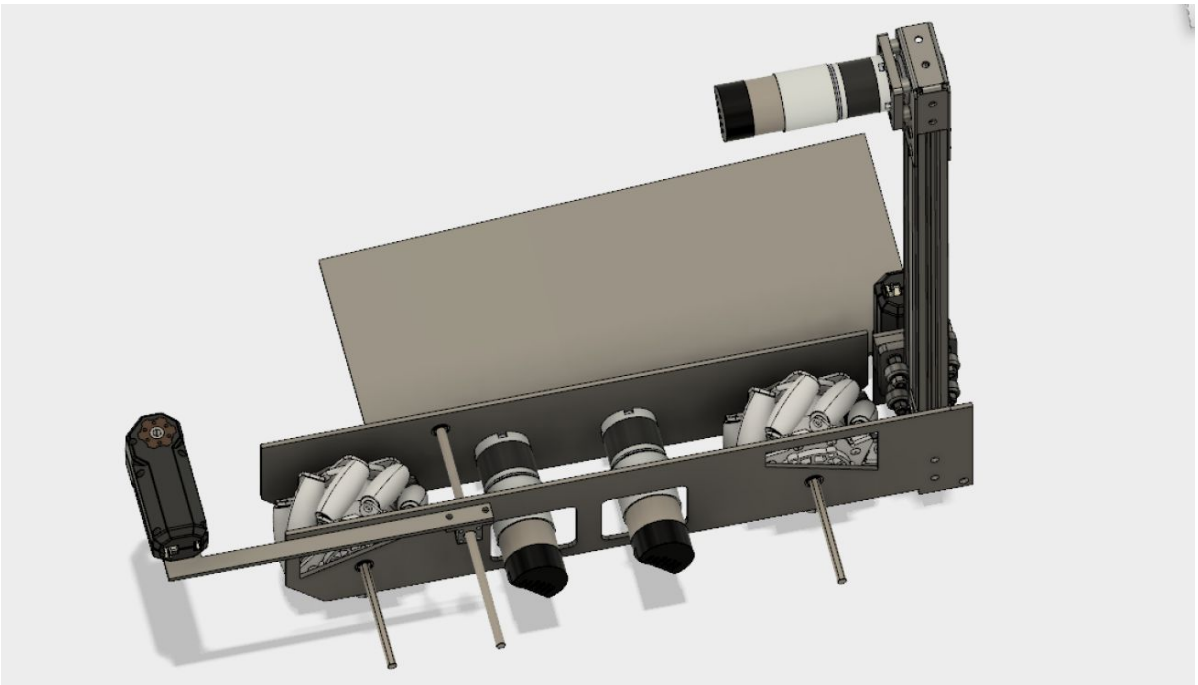
Flipper Platform



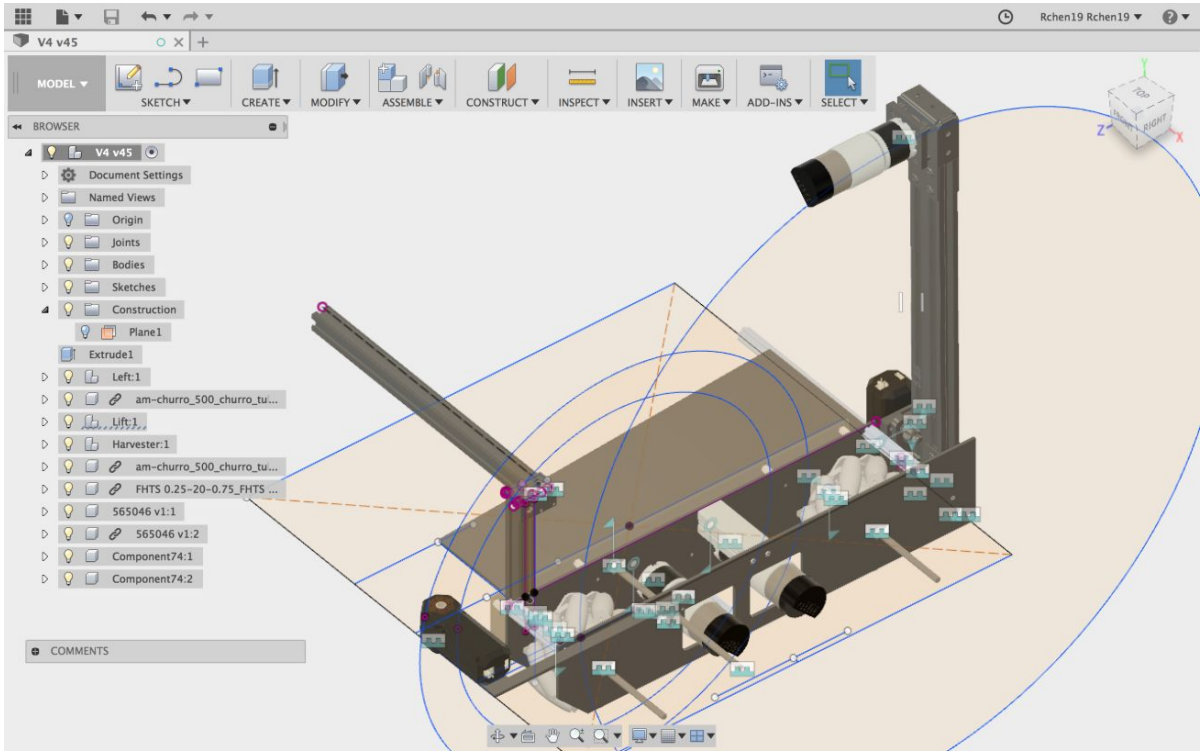
First CAD concept of a flipper platform utilizing the previous lift.

We decided to use what we learned from our early prototypes for the lift however implement it with a flipper. We created a custom mounting plate that allows us to lift the flipper plate and motor so that we could maximize the number of glyphs

we score.



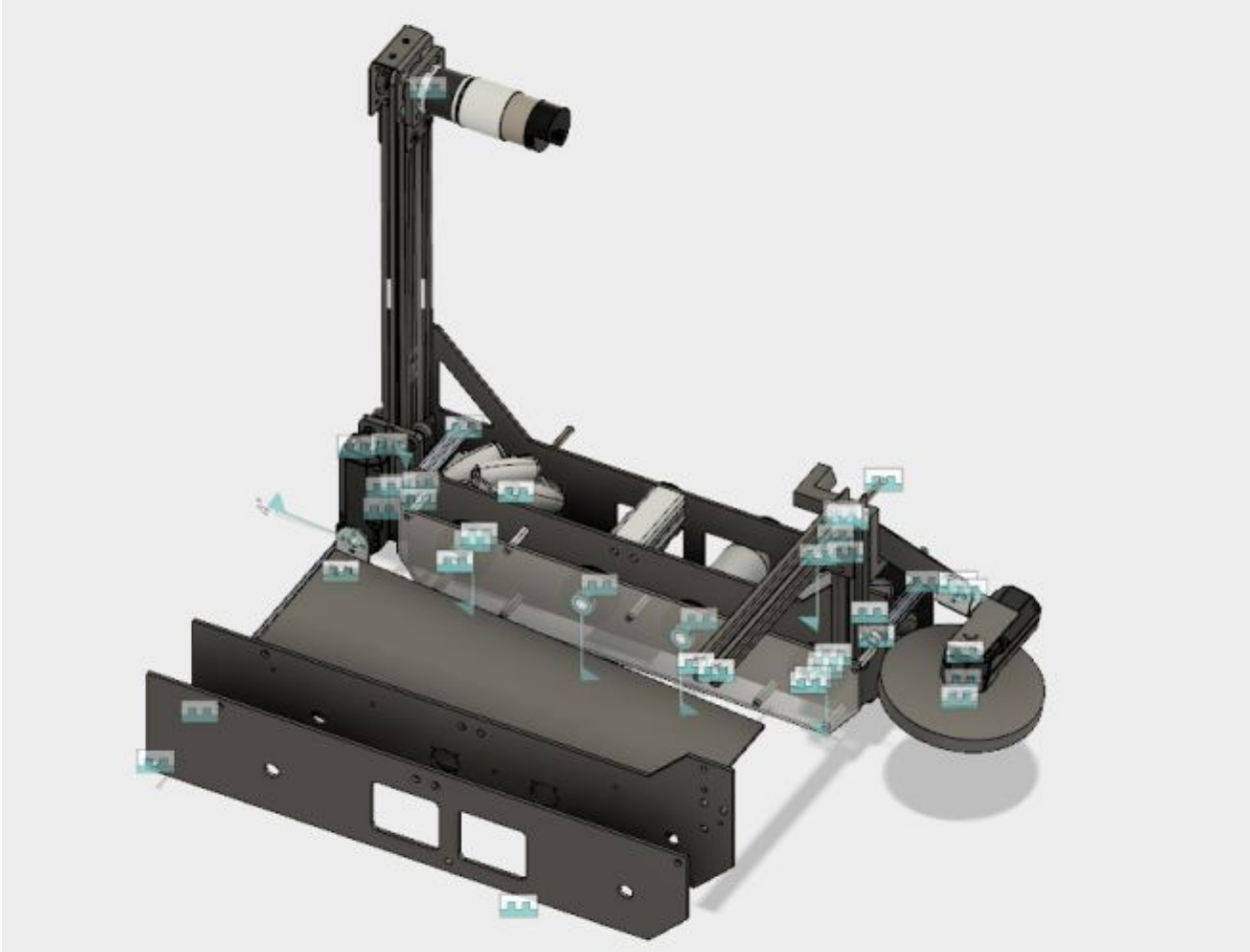
Second view of initial lift platform with concept of intake flipper.



Sketches that represent the radius of the articulating flipper.

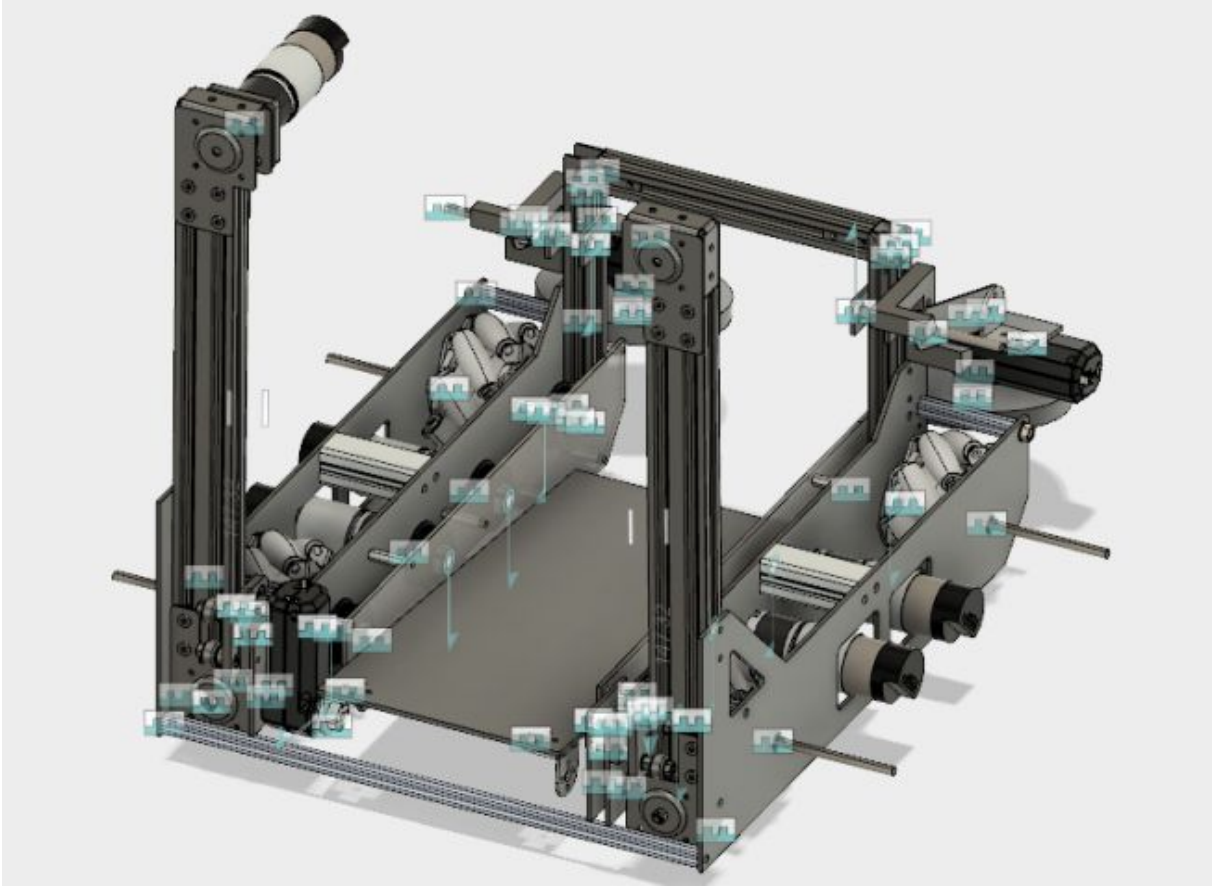
We started working on how we plan on reinforcing the robot and adding structural rigidity. With the current design we are limited in the number of places we can join the two sides of the robot. We decided to run a bar up front and another bar back behind the flipper.

Replaced the x-rail with v-rail from OpenBuilds because of the better connectors Open Build provides. We also made a new harvester that is attached to the v-rail instead of the side plate. This allows the wheel to be close to the robot but still clear the v-rail when it swings down.



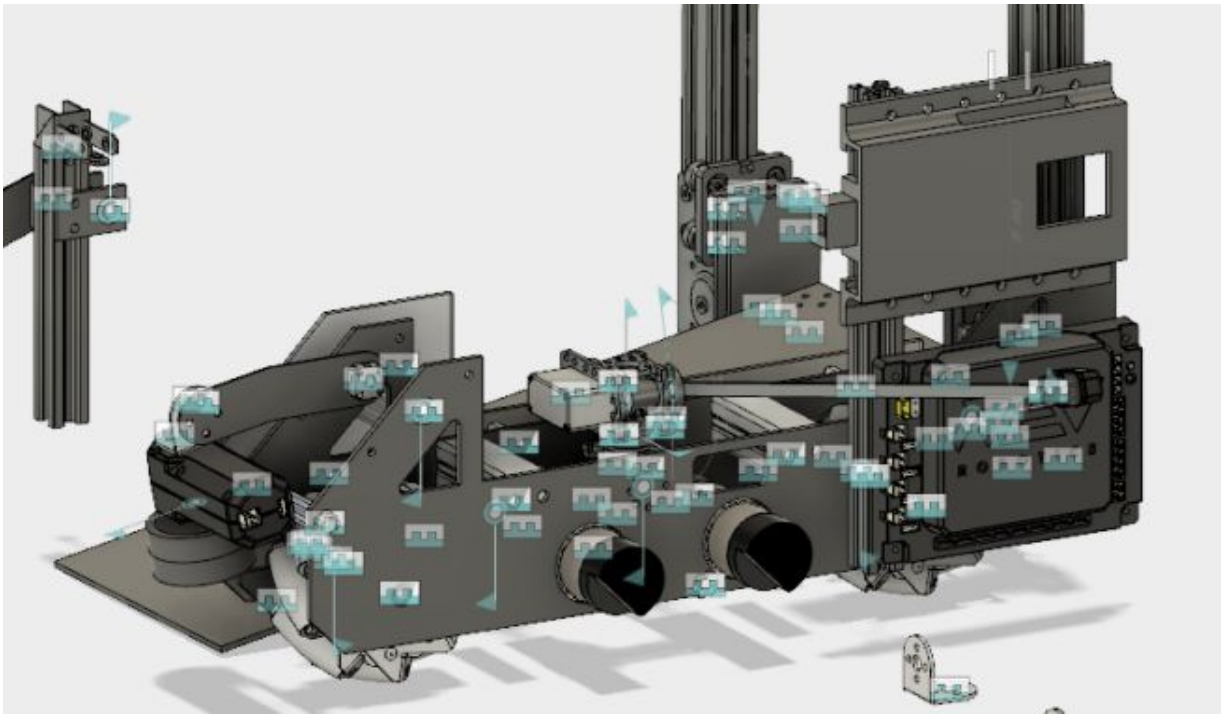
This is the first mock-up of the flipper on the lift in CAD.

Improved mounting points on the harvester and plates. Modified the plates to provide better support for the lift and V-rail. Cut down the V-rail and churro. Made the right-side plates.



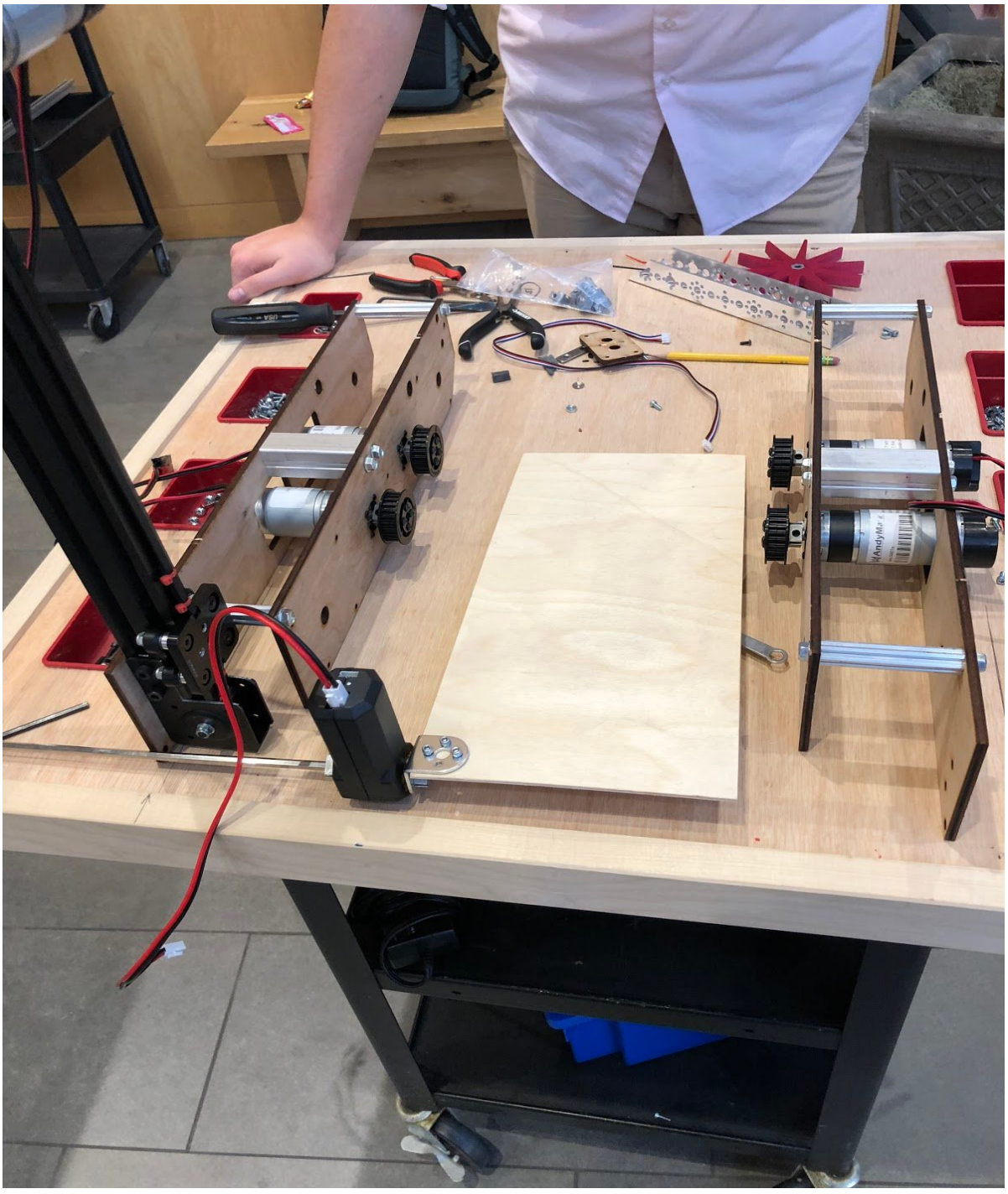
What the robot looks like when we mirrored the two sides.

After finding additional mounting points below the paddle for the peanut extrusions, we decided that the v-rails were not needed. This meant a redesign of the harvester, again. With our wooden prototype, we also discovered that the plates often got stuck on the balancing stones. To remedy this, we increased the clearance from .5 inches to an inch. We also added mounting holes for the jewel knocker.

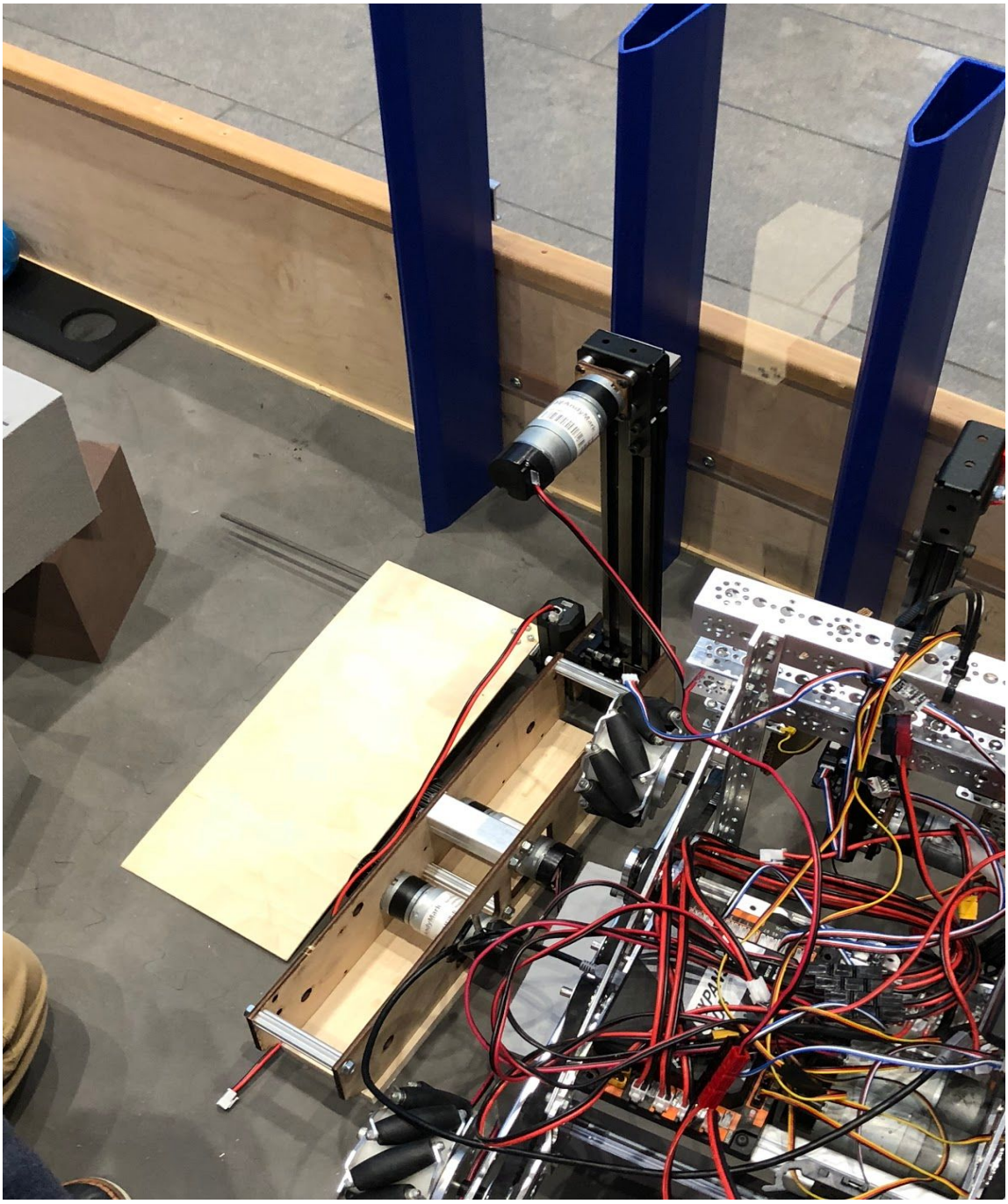


Testing the placement of the electronics and phone mount.

We assembled the mockup of the robot. We used the wood plates we cut out the previous day and 1 of the linear slides from the gripper robot.



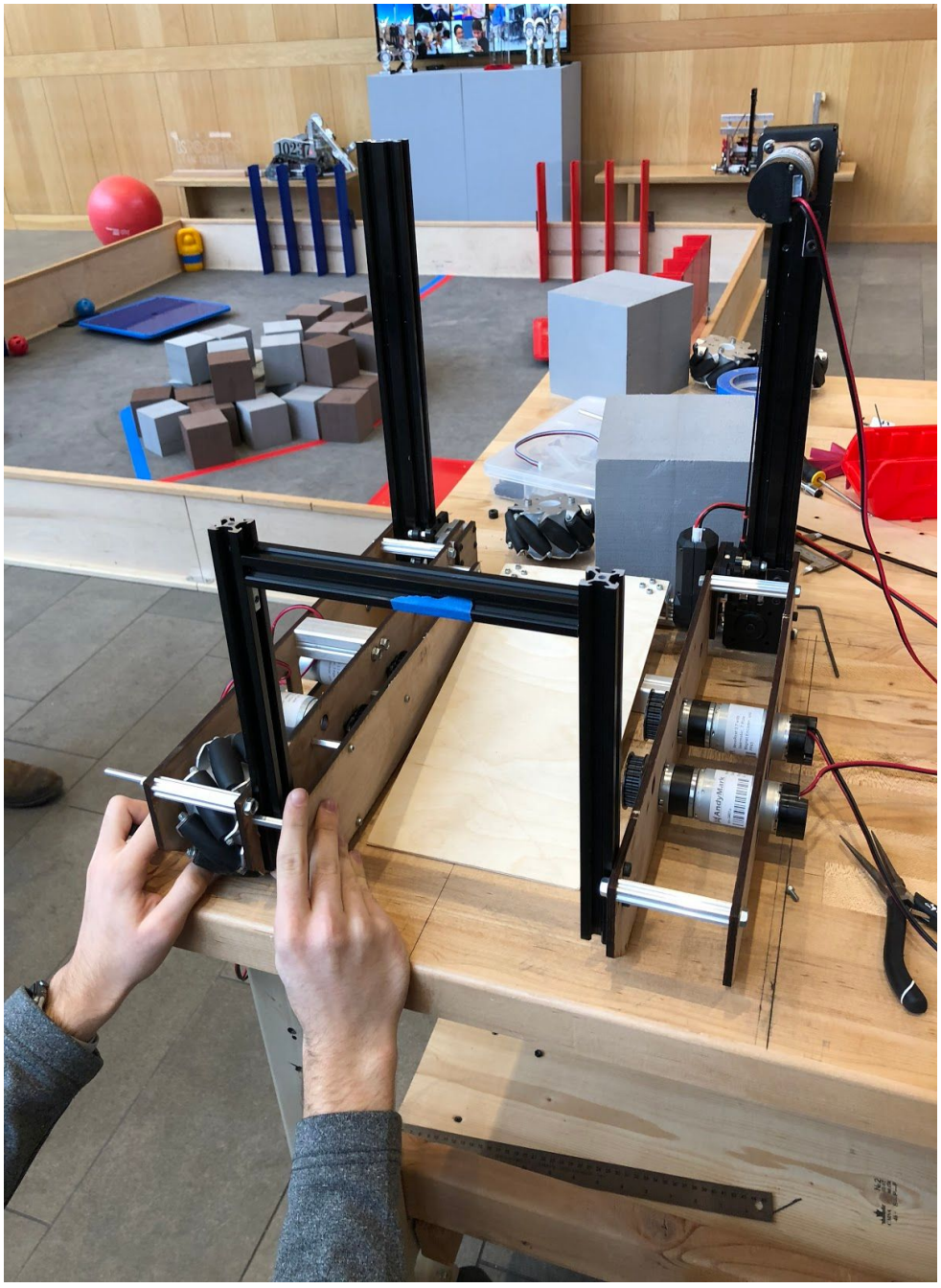
Front view of our first wood protype.



Wood flipper being tested, using the gipper bot at electronics support.

We came into school on a Sunday because we were eager to continue working

on the new design. We took off the second linear slide on the gripper robot and began joining the two sides of the robot together.



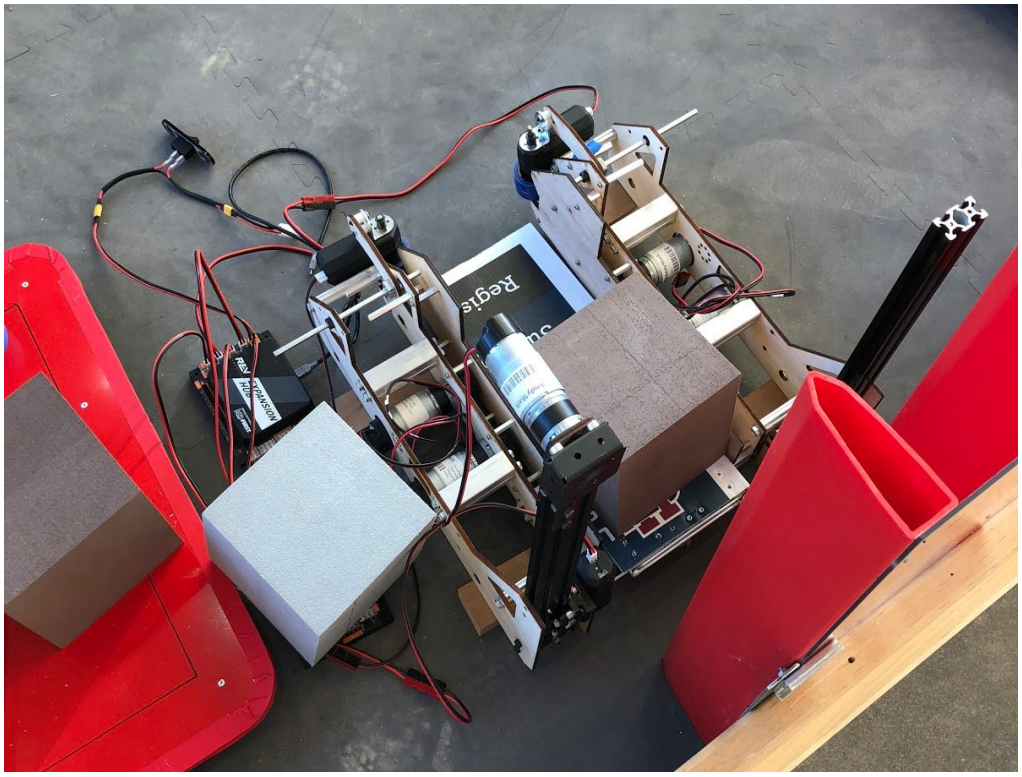
First Prototype utilizing OpenBuilds rail as support for intake testing.

We inserted the wheels into their proper positions on the new chassis to see if the tolerances were ok. We also added the flipper that we made out of wood. Finally we could see our robot lift and flip the glyphs into the cryptobox.

After being away for Thanksgiving break, we decided to meet on a Sunday again to get more work done on the robot. We implemented our first design for our harvester using Rev Core Hex motors. The wheels struggled to intake the jewels and more work needs to be done to improve upon the intake.

We replaced the wood with corrugated plastic with metal inserts so that the flipper wouldn’t sag as much.

The wooden plate being used for the flipper was replaced by corrugated plastic. The corrugated plastic was less flexible and we could reinforce it by running metal rods through the inside.

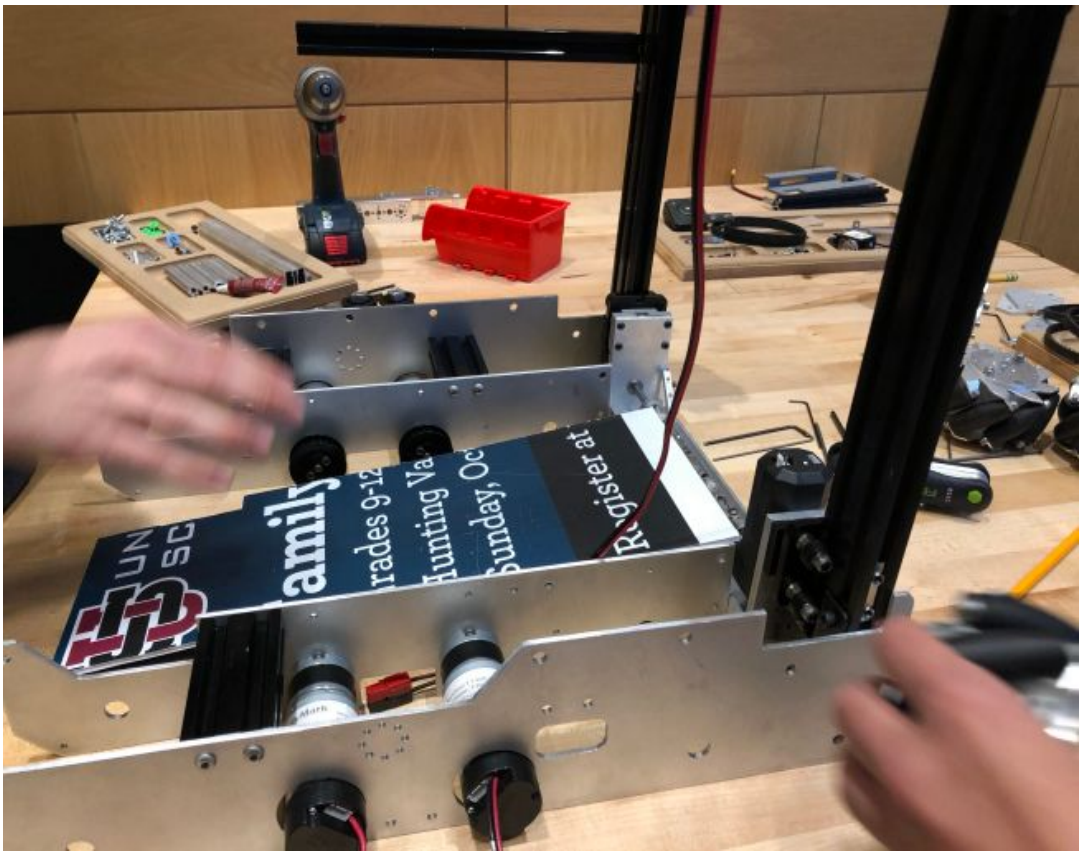


Wood prototype testing the full assembly of the robot.

The following days we used to figure out what modifications needed to be made for this robot to be functional. Firstly we needed to create a more reliable harvesting mechanism. We also would like to get the drivetrain up and running so that we could see the robot move.

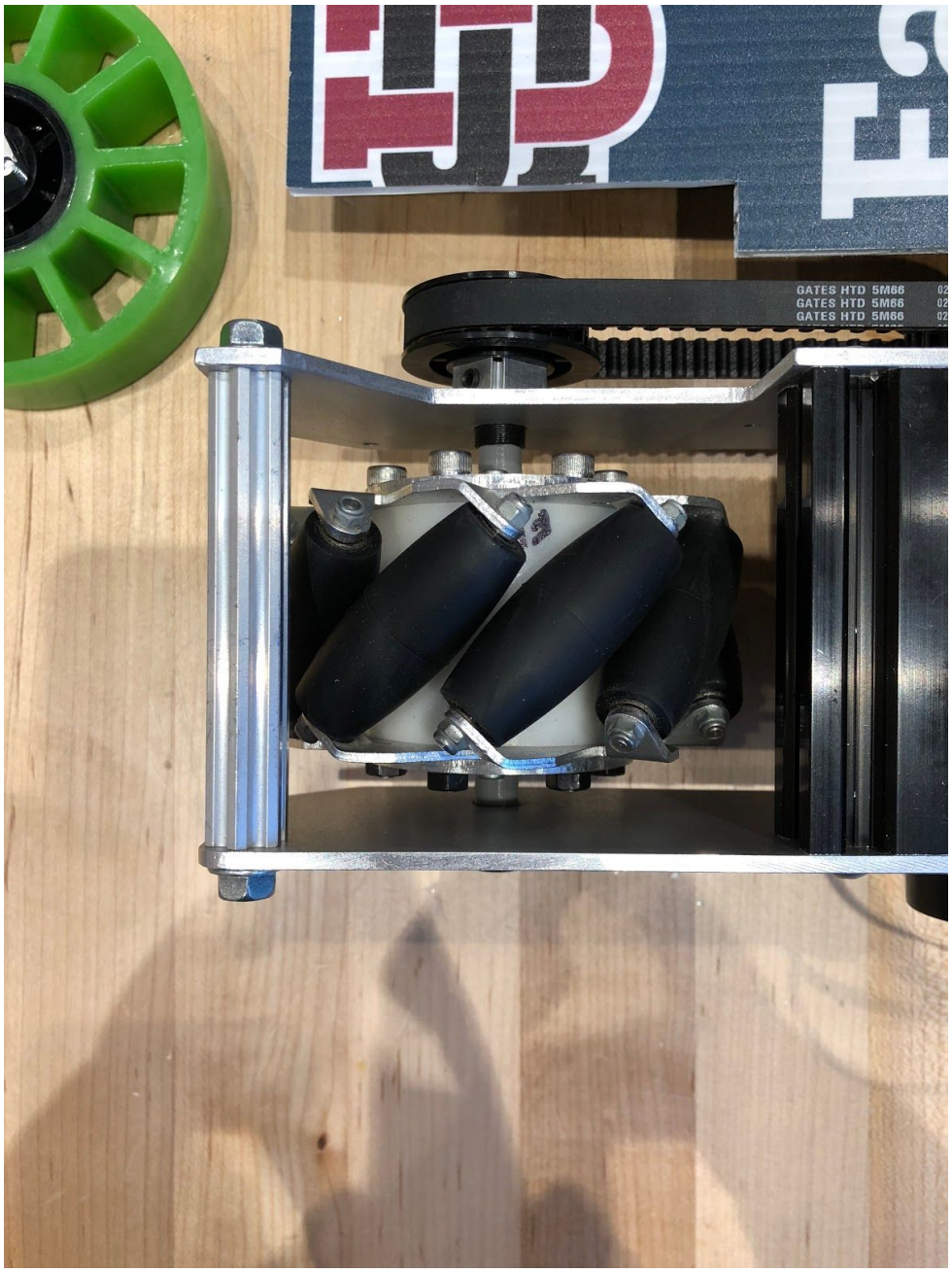
The team implemented a mockup of a new harvester that uses 2 wheels on each side. 4 inch wheels out front and 2 inch wheels on the inside of the robot. This new system was far more successful in intaking the glyphs onto the flipper and lining them up so that they are square with the cryptobox. We also received our belts for our drivetrain so we decided to test how our robot drives. Reed will proceed to design the mockup of the harvester in CAD.

We cut out the newest version of the plates out of metal because we feel that we finally have a version of the chassis that will work. The flipper lift was also reinforced to be more durable. We reassembled the chassis, lift, flipper, and wheels onto the new plates.

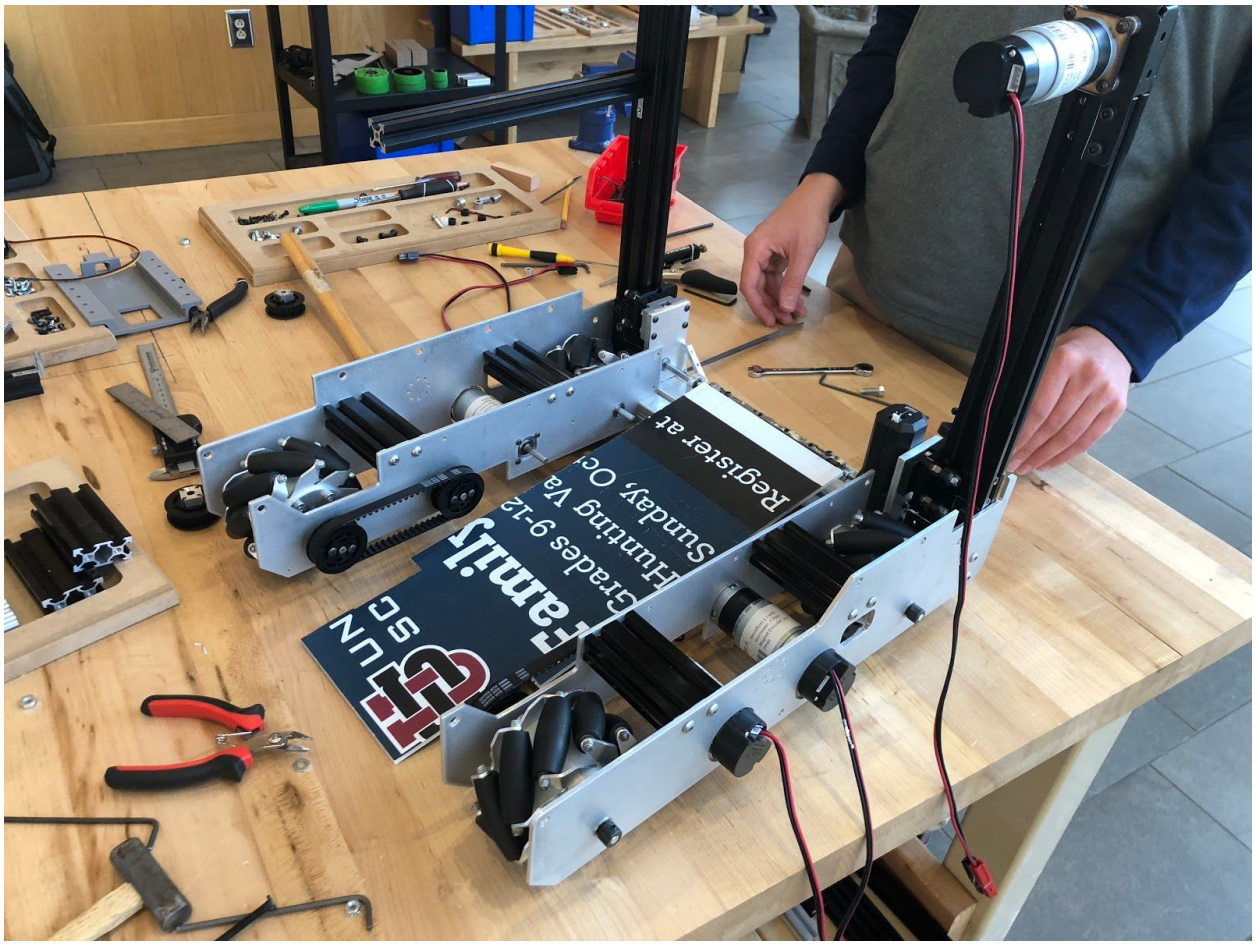


Final metal assembly being mocked up.

On Friday, we finished assembly and tested out driving the robot on the field. We found that the drivetrain was really fast but it was really slippery and hard to control. We believe this is because we are using the NeveRest 3.7 Orbital motors. We think it would be beneficial to switch to the NeveRest 20 Orbital motors because they have more torque and would decrease slipping.



Top down view of the wheel assembly.



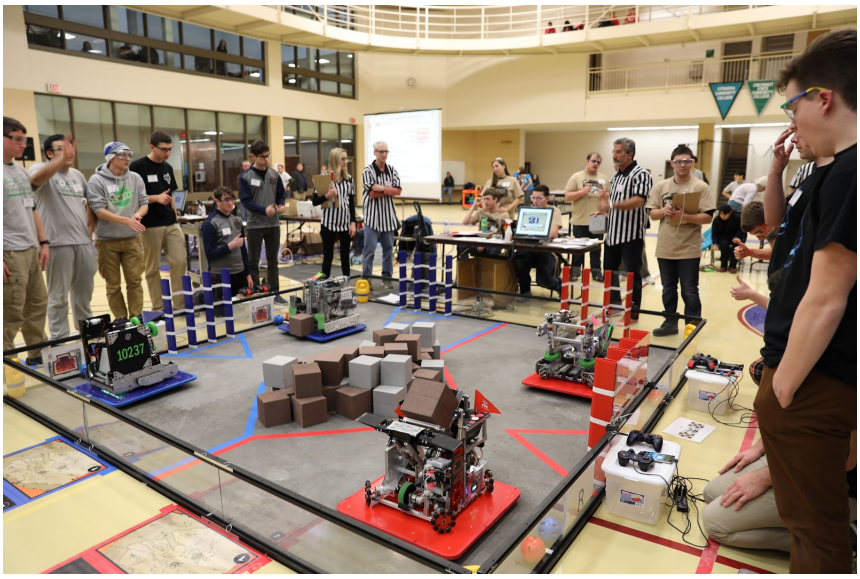
Pictured is the robot partially assembled.

Cleveland Qualifier

At the Cleveland qualifier, we performed well. In the qualifying matches we finished with a 3-2 record losing to the rank 1 and rank 2 teams . However, we were selected as the first pick of the 3rd ranked team, Hal (8497). We won the semi-finals matches, posting a score of 380 points. In the finals, we unfortunately lost to the alliance of Nuts and Bolts (5040), 7SigmaRobotics (10030), and NextNova(11302). We won the Design Award for our use of laser cutting wood and 3D printed to rapidly prototype nearly everything on our robot. We were the alternate at Cleveland with a top qualifying score of 305.

Our team won the Design Award and were the 2nd pick of the Finalist Alliance.

Despite not qualifying, we learned a lot from this competition. We learned a lot from the performance of 7Sigma and Nuts and Bolts, who together posted the 3rd highest score in the world at the time. The relic turns out to be a crucial part of this years game, especially at higher levels of play. Thus, we decided that we should continue to develop and improve our relic mechanism and try to have it solid and reliable for states.



Our robot (closer side blue) competing in a Finals match.

Additionally, we found that we could improve our stacking speed. We identified two main issues that were slowing us down: harvesting and lining up. We found that a lot of cubes would get stuck in the harvester or not be placed properly on the flipper. Additionally, we found that it would take a while to line our robot up with the cryptoboxes, especially in the corner/farther cryptobox.

Consequently, after the Cleveland qualifier we came up with a list of goals and things to work on.

For the autonomous, we wanted to look into using computer vision besides Vuforia in order to line up with the cryptoboxes more consistently and harvest second and third cubes during autonomous more consistently. Besides autonomous, we wanted to work on some improvements to make driving easier for both the drive train operator and the auxiliary controls operator. We wanted to make it so that the lift and flipper would keep its position even when the driver isn't apply power in order to make flipping easier and faster, we wanted to add an auto-reset feature so that we can quickly lower the flipper and lift to be ready for the next harvest, and we wanted to add an auto modulation feature to the harvester so that we can harvest quicker and faster (modulating the harvester motors helps to harvest). Additionally, we wanted to slow down the overall speed of the robot but still have control full over its movements.

For the robot, our main plans were to improve the harvester and work on the relic recovery mechanism. We already had a few iterations of the mechanism and we identified that it would be critical that we get a polished and working relic mechanism.

Kent Qualifier

Our team won the Control Award, 3rd Inspire and were the Captain of the Winning Alliance.

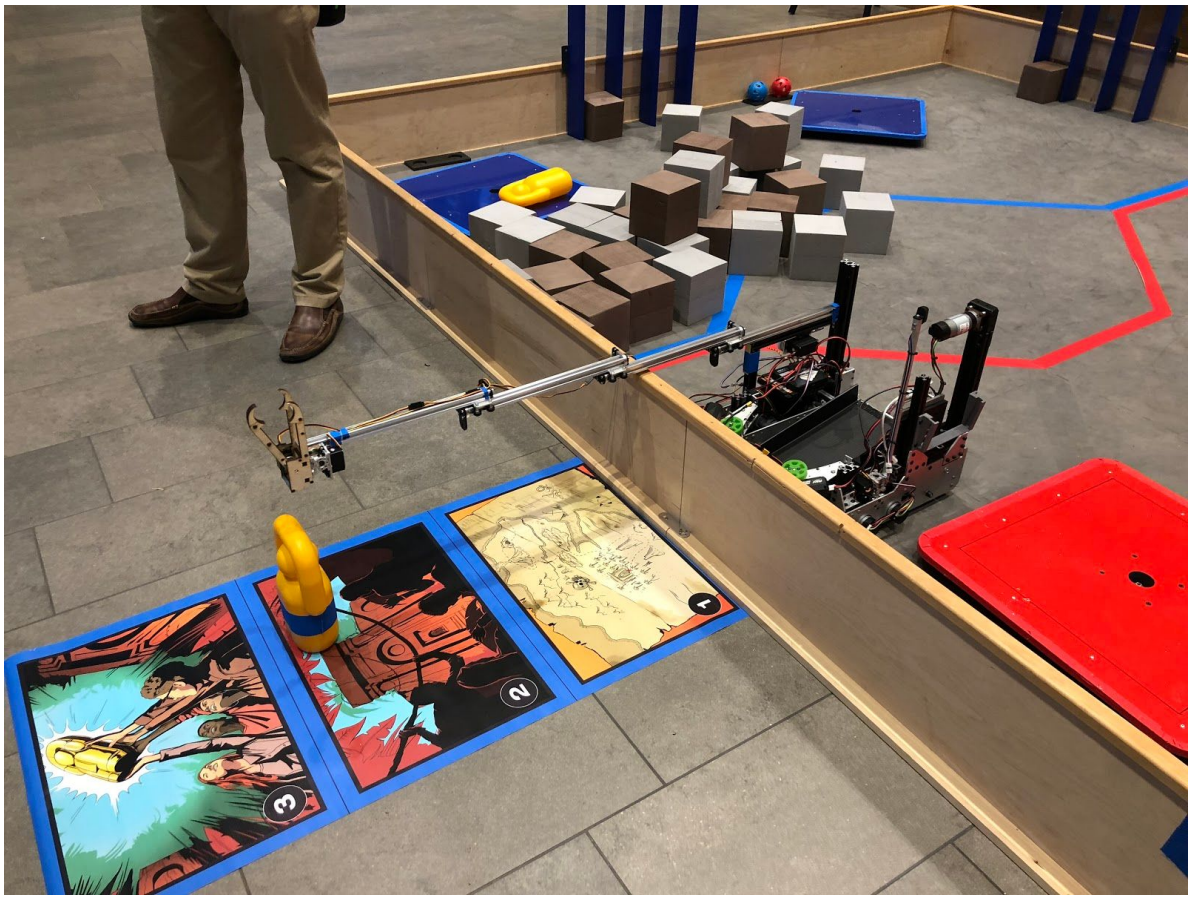
At the Kent qualifier, we had an amazing showing. In the qualifying matches we finished with a 5-0 record, going undefeated! As captains, we invited Hacksaw (1001) and Trial and Error (10144) to our alliance. We won the semi-finals matches, posting the 7th highest in the world at the time of 504 points. We advanced to the finals and finished the day as the winning alliance. We won the

Control Award for our innovative programming and multi-glyph autonomous. We also received recognition for 3rd Inspire and were a Design award finalist. We want to continue and tune our glyph scoring mechanism as well as integrate a relic mechanism and a solution for parking on the balancing stone.

Relic Mechanism

During our initial ideation process, we decided that our robot should have the flexibility to allow us to add a relic-recovery mechanism at a later date. This allowed us to focus our efforts on building a reliable platform flipper and glyph intake.

Now that we are further on in the season, we have begun to approach the task of adding relic recovery capability to our robot. We started to approach this task by utilizing the Actobotics Cascade Lift Kit. In addition to assembling that kit, we needed to design a subassembly that could grip the relic as well as rotate it up above the field wall.



This is our initial prototype of the relic mechanism attached to our robot, which was just short of the third zone.

As depicted above, our initial relic recovery prototype utilized an unmodified Actobotics Cascade Lift Kit, as well as a custom designed relic grabber. To create this grabber we utilized similar techniques to prototyping the grabber. We were able to iterate quickly using light weight wood. Unfortunately this first try at the relic recovery mechanism did not meet our goal of reaching zone 3. We then paused development of the relic mechanism, in favor of improving reliability of the flipper, intake, and autonomous.

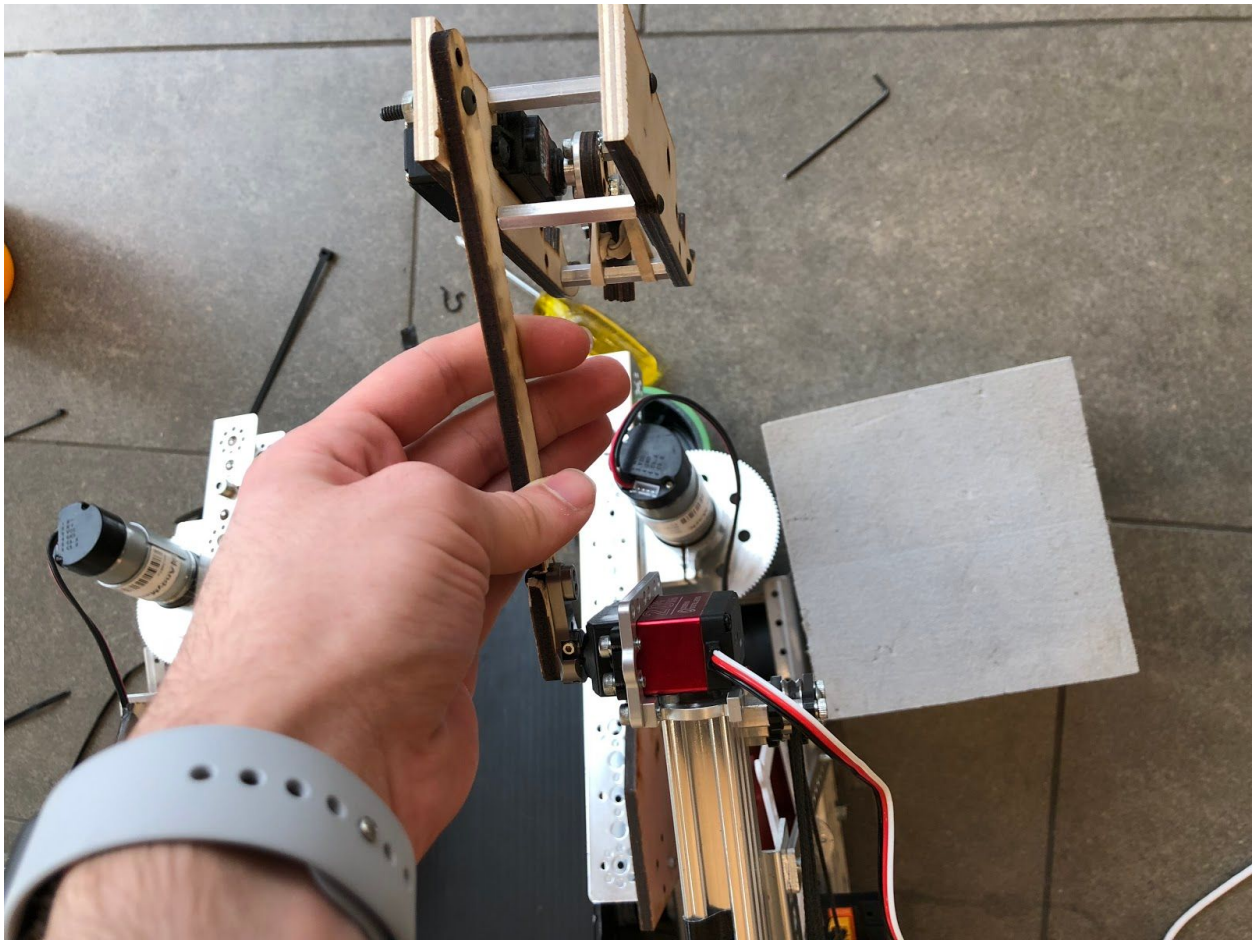
After the Cleveland qualifier we returned to developing the relic extension mechanism. In addition to reaching zone 3, we also challenged ourselves to create a mechanism that was reliable and would not inhibit other subassemblies on the robot. Since we were unable to reach zone 3 with our current implementation of the Actobotics kit, we decided to go for an articulating arm design. This provided the amount of extension need to reach zone 3. However there was a downside to using this type of design, it was a lot of mass that was being moved with a low degree of precision. This made it difficult to implement a gripper that could easily grip the relic.

T

Depicted is the articulating relic arm that used direct gearing and a belt drive, the wood while easy to prototype, introduce excess flex into the system.

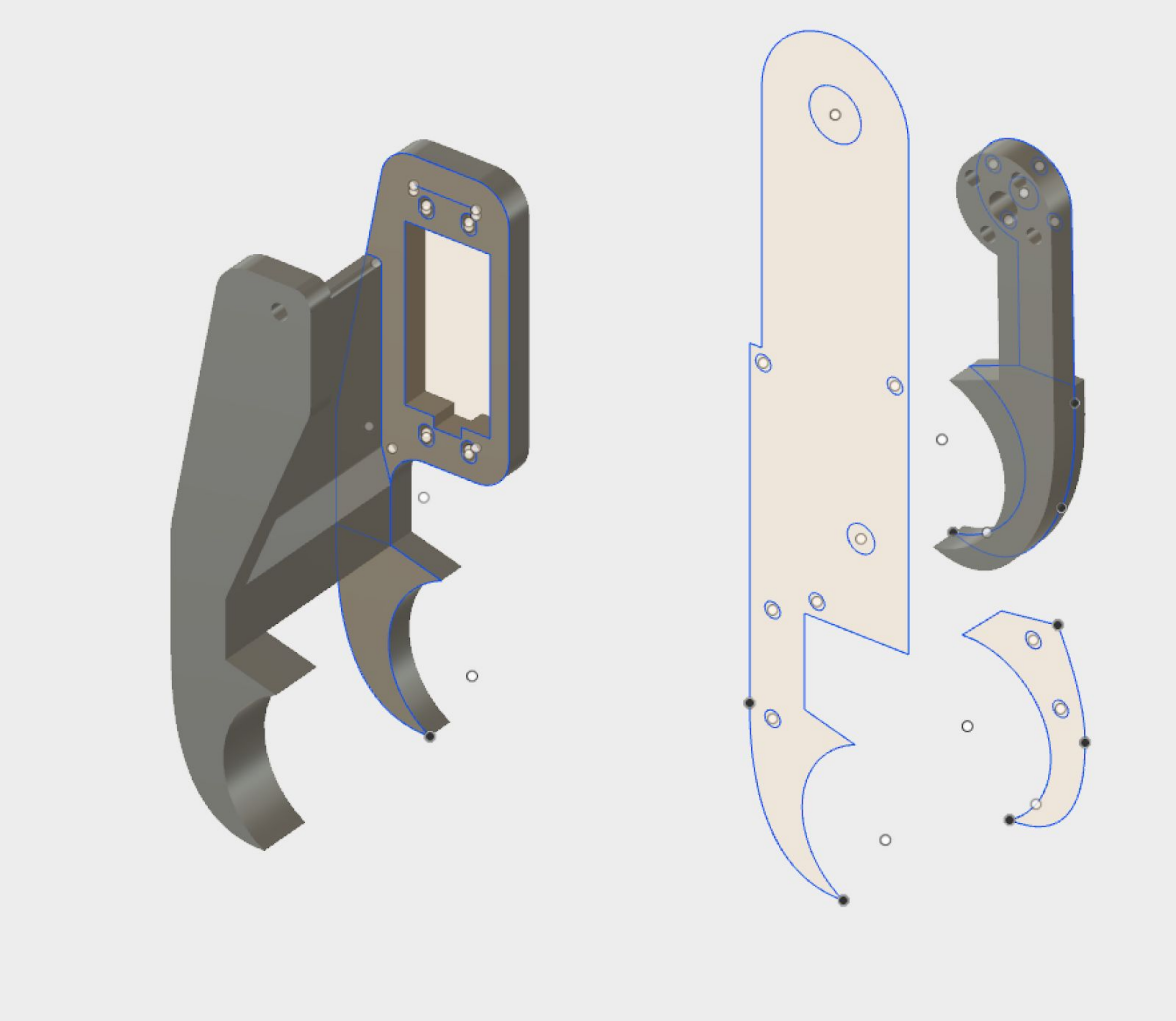
After having experimented with a pivoting arm prototype we decided to return to the Actobotics kit. We again explored different possibilities for maneuvering the relic above the game field wall. This next step was to create a lever attached to a high-torque servo. At the end of this lever would be an articulating claw that we repurposed from our previous attempts. This claw design worked well and could easily grab the relic. Our initial attempts to build a lever included the use of laser cut wood. However we quickly observed that this was incredibly unstable and prone to snapping.

Depicted the the reused relic grabber prototype. Attached is a snapped lever prototype.



Shown is the wooden prototype of the relic arm with the repurposed grabber.

As seen above, we used the Actobotics servo block to mount the high-torque servo to the end of the top cascade extension X-Rail. We then made a custom arm to attach to the servo, creating a lever that could move the relic above the field wall. We then used these wood prototypes to validate the system kinematics. After we validated the kinematics, we proceeded to fabricate a metal lever and 3D printed gripper assembly.



Shown is the CAD model of the 3D printed gripper assembly.

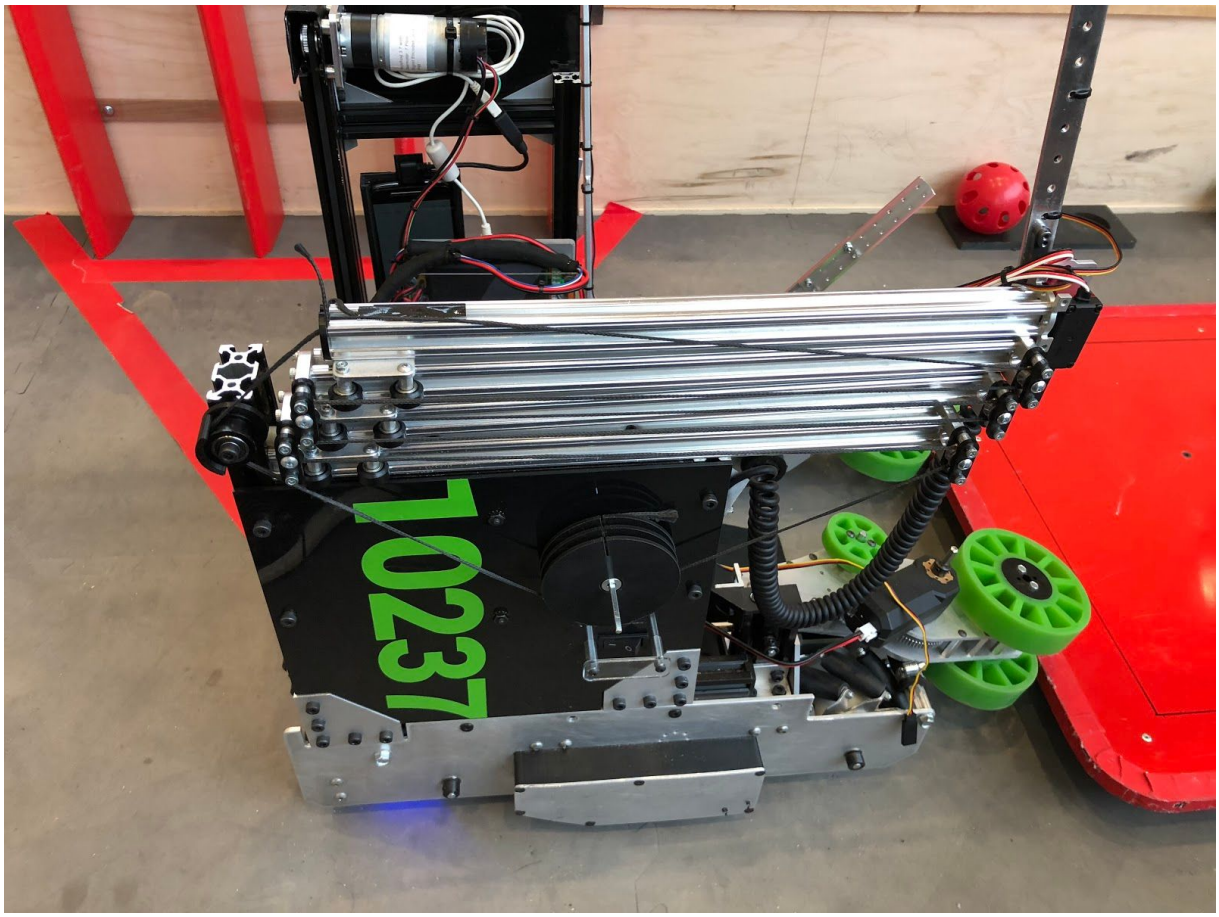
The metal and 3D printed relic lever and gripper assembly.

After the Kent State Qualifier we decided to continue development of the relic mechanism. Our ultimate goal was to have a polished system implemented roughly a week before the the state tournament. This would afford us the time to practice driving and relic recovery techniques in a controlled environment.

Building from our previous progress, we decided to begin the transition from our prototyping chassis to the main competition bot. This process was made much easier because we had initially considered adding a relic mechanism earlier in the design process. In order to attach the relic mechanism we decided to utilize the existing OpenBuilds rail on the robot frame. We then used T-nuts, to secure the Actobotics X-Rail onto the OpenBuilds rail.

After attaching the modified actobotics kit, we proceeded to find a clean and effective solution to protecting the new relic electronics and mounting the motor and pulley assembly. We started with simple and smaller plates that could easily be attached to the OpenBuilds rail. We then quickly began to explore the possibility of making a flush mount plate on the outside of the robot. This plate would allow us to have flexible mount points but also protect the cables the cableign and electronics from damage.

Our initial plate prototypes were only intended to mount a single Vex 393. We soon added functionality to mount the Rev Servo Power Module. While this design enabled us to test the Actobotics extension, it failed to protect exposed electronics on the robot.



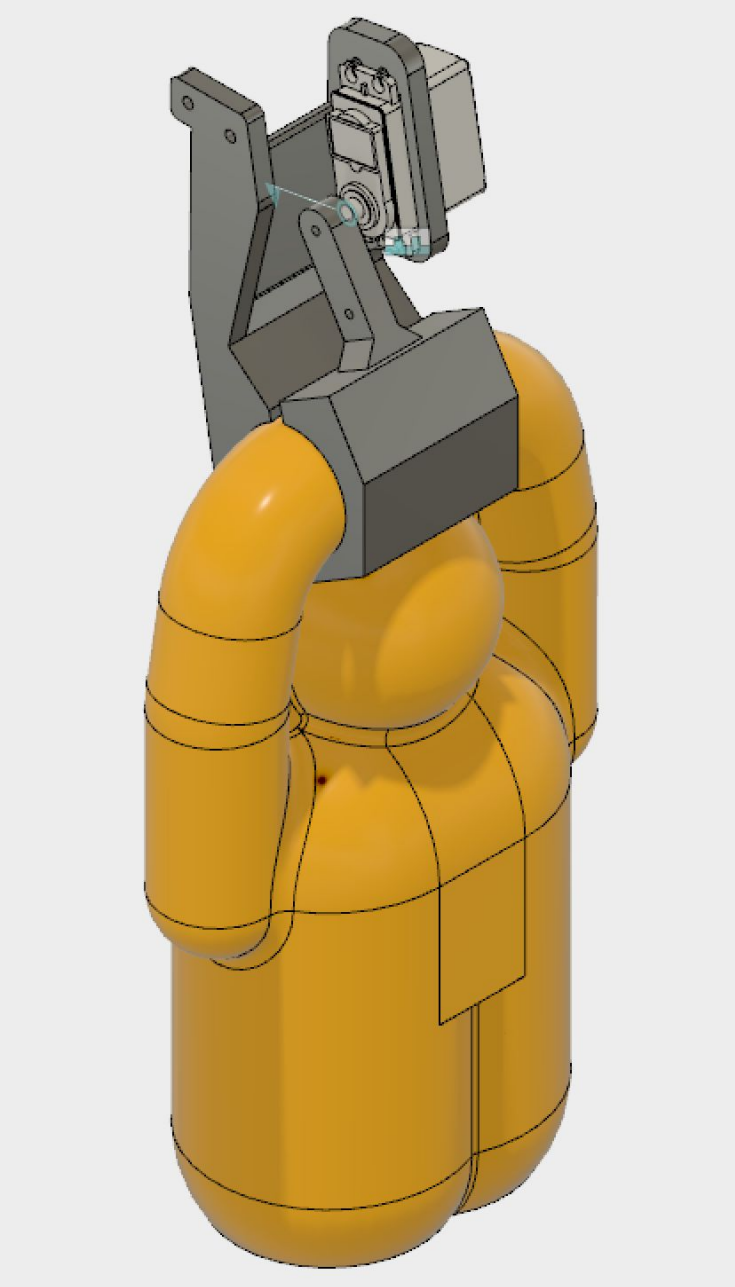
Side profile of relic extension with integrated Delrin plate.

After having completed multiple iterations of the relic extension, we began to refine the gripper for the relic. Having learned from our previous relic gripper designs, we started to define crucial improvements that need to be made. We determined that we needed to decrease the weight and size, increase grip strength, reliability, and accuracy. In order to achieve this, we decided to prototype utilizing a 3D printer. This was because in this instance, we felt limited by the capabilities of the laser cutter. By utilizing a 3D printer, we could create a highly integrated and singular mechanism. We decided to create a gripper

assembly that used only two seperate printed parts that reduced mechanical complexity, in turn increasing reliability.

Major iterations of the gripper design that showcase our transition between prototype methodologies.

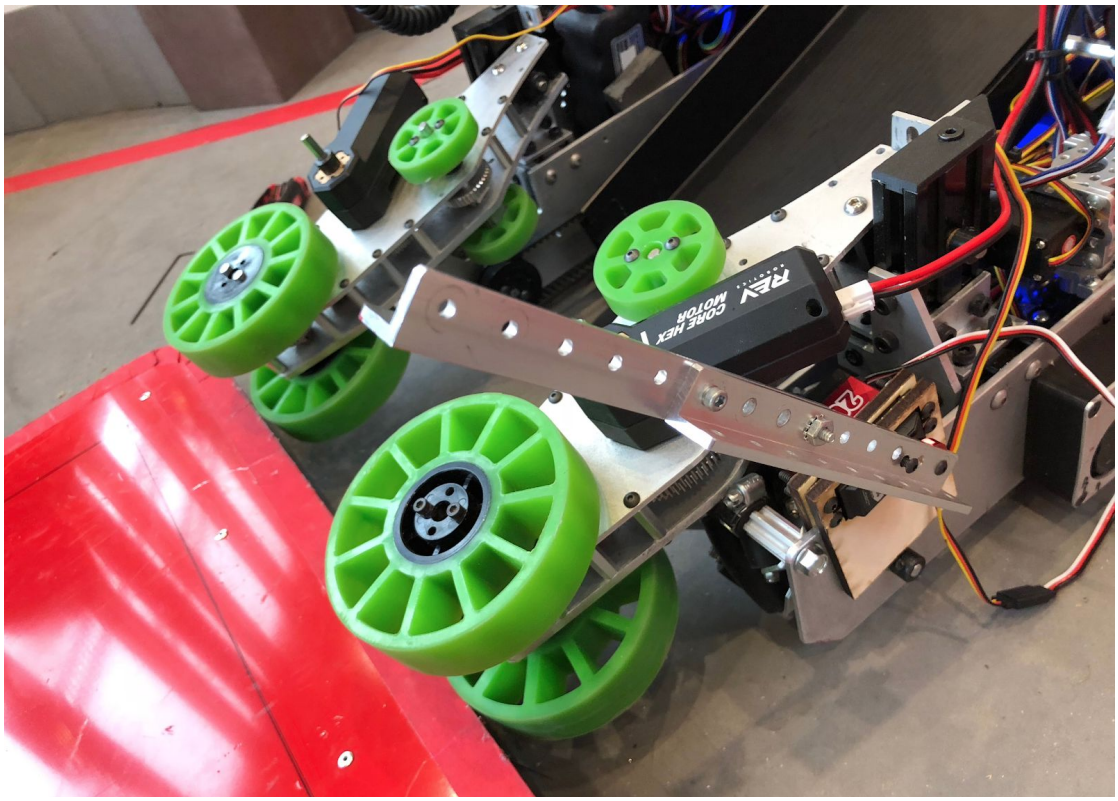
The near final iteration of the relic gripper. This design precisely matches the three dimensional profile of the relic.



Depicted is the CAD model of the near finalized iteration of the relic gripper. This model utilizes a technique that can be referred to as “boolean subtraction”. This allows us to create a 3D form that precisely matches the profile of the relic, decreasing the amount of precision required to grip the relic.

Balancing Stone Mechanism

Up until after Kent, the robot had difficulties getting back up onto the balancing stone. We wanted to solve this issue before the State tournament because we felt that it would be beneficial to be able to more easily park if time allows. The ground clearance of the robot and intake position make it both beneficial and detrimental for parking on the balancing stone. If we manage to get the intake over the balancing stone we found that it would push down on the plate and allow us to more easily climb back on. However, getting the intake over the stone was very difficult.



Our first prototype that would push down on the balancing stone.

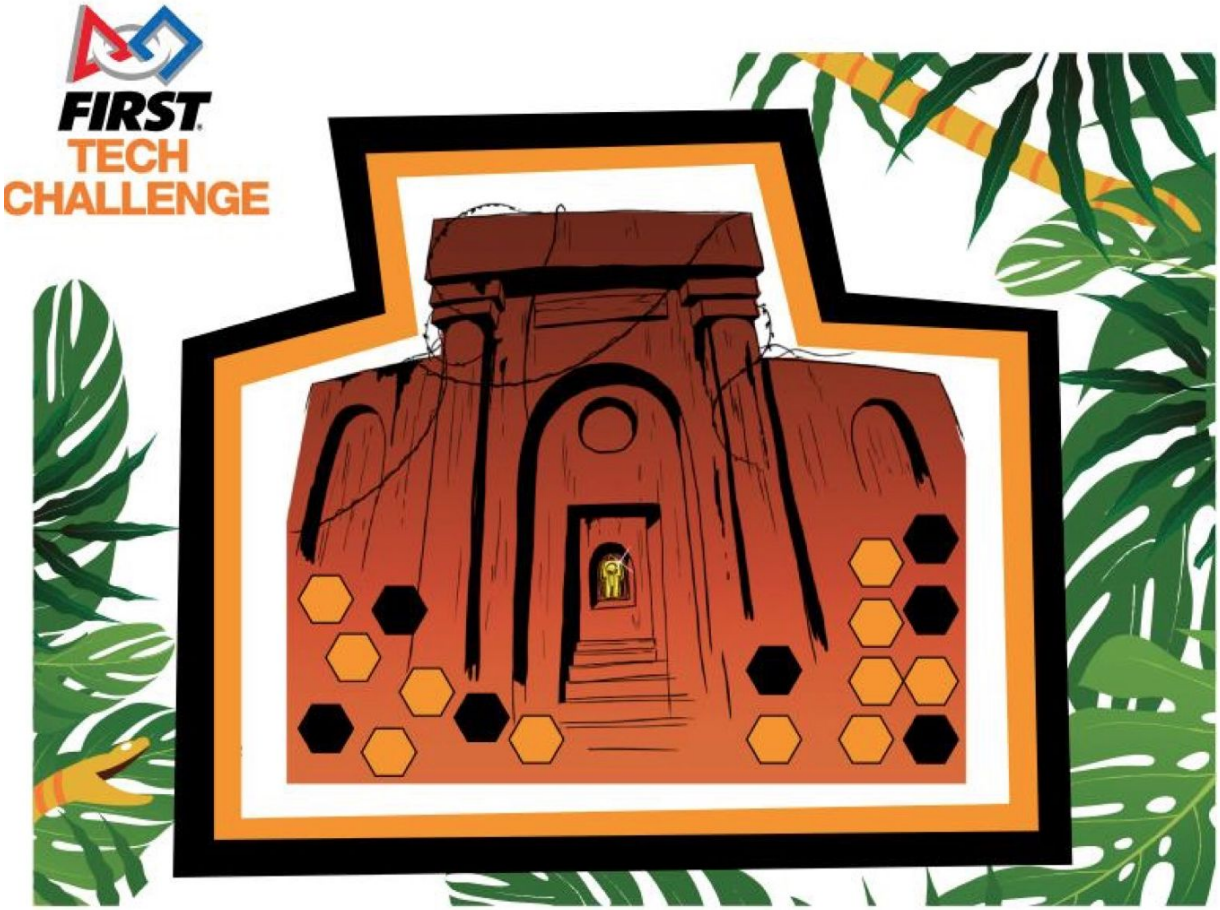
Our first prototype involved tetrix pieces and our strongest servos to see if that would be enough force to push down the balancing stone. Through our testing we concluded that pushing down the tile with the servos would be extremely difficult and nearly impossible.

We looked back at our problem and realized that we should use what we already have on our robot to our advantage. We realized that if we could create a mechanism to lift the intake slightly we could bring it over the balancing stone and then drop them. This would provide us with the force needed to push down the plate.

We mounted the servos under the intake to allow for it to move up and down.

This proved to be a fairly simple solution and it now allows us to park on the balancing stone much better than before.With the combination of the high-torque servos in order to lift up the heavy intake pods and drivers practice we finally can do every task on the field. In the future, we plan to enhance our parking ability and make it even easier for the drive team.

Programming: The Evolution of Autonomous



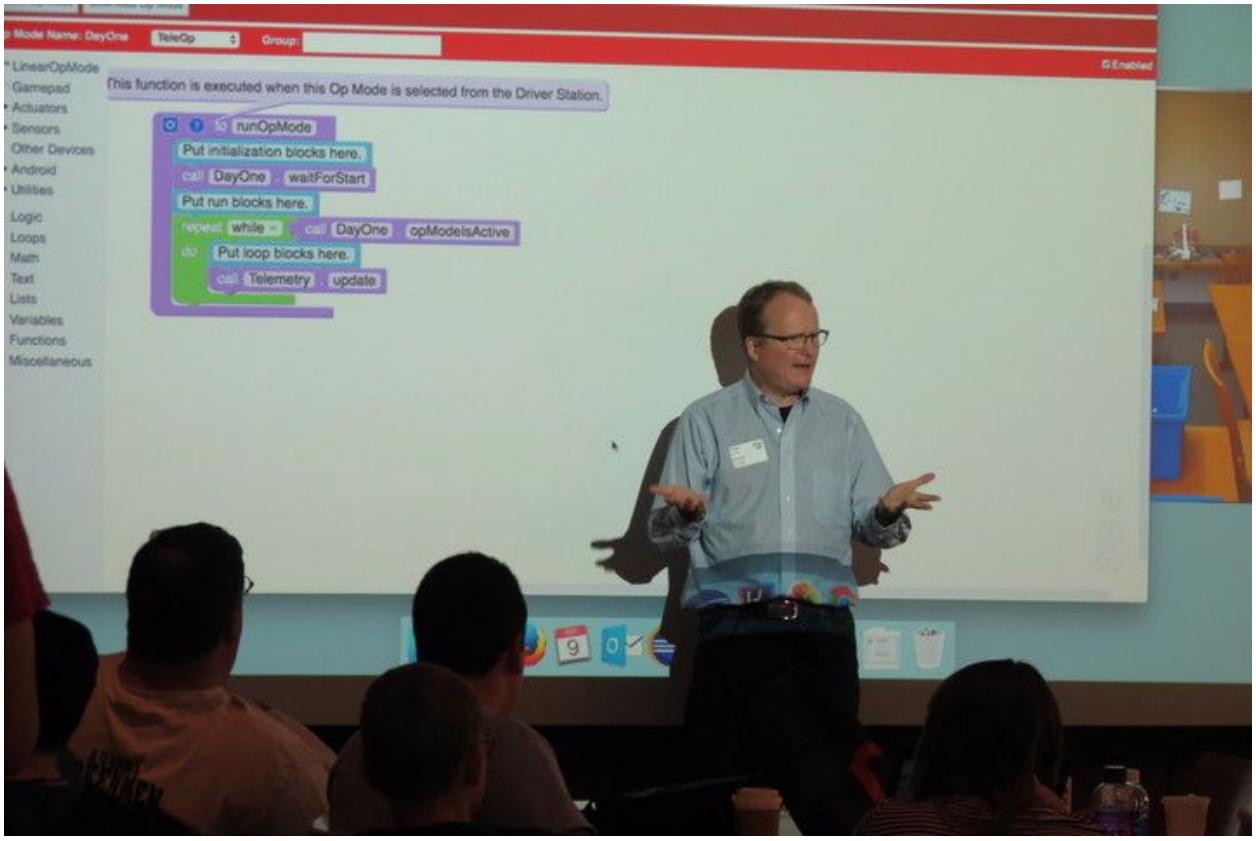
We set up each of our bots using a hardware, teleop, auto setup. For each robot we have a hardware class and a teleop class that has an instance of the hardware class. As well, we have one abstract for the autonomous that contains the code for the red and blue sides. Boilerplate code is then used as a concrete implementation of the autonomous for the red and blue sides and those files are registered for autonomous. For example:

```
public class MecanumBotHardware {

    public HardwareMap hardwareMap;

    public DcMotor frontRight;
    ...
    public BNO055IMU imu;

    public boolean frCorrectDirection = true;
    ...

    public void init(HardwareMap hm){

        hardwareMap = hm;

        getDevices();

    }
    public void getDevices() {

        frontRight = hardwareMap.dcMotor.get ("fr");
        ...

    }
    public void setPower(float p){

        frontRight.setPower(p);
        frontLeft.setPower(p);
        backRight.setPower(p);
        backLeft.setPower(p);

    }

    ...
```

```
    public void bringUpBallKnocker(){

        ballKnocker.setPosition(0.8);

    }

    ...

}

public class MecanumTele extends OpMode {

    private MecanumBotHardware robot = new MecanumBotHardware();

    ...

}

public abstract class MecanumAuto extends LinearOpMode {

    public MecanumAuto(boolean color){

        isBlue = color;

    }

    @Override
    public void runOpMode(){

       ...

    }

}
public class MecanumAutoBlue extends MecanumAuto {

    public MecanumAutoBlue(){

        super(true);

    }

}
```

Kickoff Event

We attended the kickoff at Rockwell Automation in Cleveland at the beginning of the season. In the morning, Dylan and Mr. Sweeney taught an introduction to programming class. During this class, we taught about 20 teams about basic programming concepts and how to get basic teleops and autonomous running using OnBot Block Code. As well, helped a number of teams with other issues such as updating their phones, updating their apps, and getting their electronics working.



Mr. Sweeney and Dylan teaching the introduction to programming class at the kickoff event.

First Autonomous

Our first programming goal was to make a simple autonomous that would knock the correct jewel off of the platform. We decided to drive forward or backward depending on the color sensor reading as this is an easy solution and requires no physical modifications to the jewel knocker (i.e. a second axis). We had to resolve the physical balance issue. Firstly, depending on the center of mass of the robot, the positioning of the jewel knocker may have to be moved. Secondly, because we start on the balancing stone, depending on if we move forward or backward to knock the jewel we will be in a very different location. In order to solve this, we will have to make sure that if we drive backward we can re-center ourselves back in the same position as if we drove forward. Here is the code for the jewel knocking:

```
if(isBlue){
    if(robot.readingBlue()){
        motorPower = -0.5f;
    }
    else {
        motorPower = 0.5f;
    }
}
else{
    if(robot.readingBlue()){
```

```
        motorPower = 0.5f;

    }

    else{

        motorPower = -0.5f;

    }

}

robot.setPower(motorPower);
sleep(200);
robot.setPower(0f);
robot.bringUpBallKnocker();
```

Control Scheme

For our first robot, we tried to develop a way to control the arm and keep it steady. We programmed in basic controls for the arm. Everything worked, however, the lift would violently fall back down from its own weight. We tried modifying the code so that it would apply a constant power whenever it wasn't being lifted or lowered but this worked questionably. We should just add ~10% to the power at all times. This should keep it from falling down and should cause the arm to raise and lower at about the same speed.

If that doesn't work, we will have to implement some sort of proportional (and maybe integral and derivative) position hold system using the encoders. Hopefully, this is the final solution because it should be able to keep the arm at a constant height and move it there quickly and be able to withstand significant perturbations without getting into a feedback oscillation. We never implemented a PID control loop because we abandoned the arm idea. This is the code for the lift:

```
float gripperLiftPower = -gamepad2.right_stick_y * 0.5f;

if(gripperLiftPower < 0.01f && gripperLiftPower > -0.01f){

   gripperLiftPower = 0.1f;

}
```

Once we switched to a lift and gripper design, we worked on a way of having "steps" for the lift. This would enable the drivers to quickly go to one of the positions (i.e. glyph 2, glyph 3, and glyph 4). We used the encoder to achieve this.

Although the code worked, it was very buggy and would sometimes artificially stop the movement of the lift. Thus, we decided to remove it.

After this we added a second lift to the gripper bot. We added the second lift to code and tried to implement a quick proportional control system. Although this was opperable, it still was very jerky. However, we decided not to change it because we were scrapping the idea the next day.

For the control scheme on our mecanum chassis, we used the same control scheme as last year. We control the robot like a drone with one stick controlling forward and back movement and strafing and the other stick controlling turning.

Once we started practicing more, we found that we needed a more precise control scheme. We added sneaking functionality to slow down the robot. This gave us the ability to precisely control the robot and move quickly. Furthermore, we found that doing the cryptobox that is on the side wall (opposite to the relic recovery zone) was very confusing with this control scheme because most of the controls were reversed. Thus, we also added a reverse button to reverse all of the controls. This made it much easier and intuitive when going for the far cryptobox.

While one driver controls the robot, the second driver controls the lift, harvester, and flipper. We wanted to harness the fact that our two harvester wheels so each harvester motor is independently controllable and reversible. We mapped the lift and flipper to the two sticks on the second gamepad.

After Cleveland, the first thing we worked on was improvements for teleop. For the drive controller, we added a toggleable brake mode. Additionally we slowed down the default speed of the robot in order to make it more controllable. We

also added a full power mode and a super slow mode. This gave the driver an easily accessible way to move the robot at every speed.

For the auxiliary operator, the primary aim was to get the lift to hold its position when no input is given. We found that using the default brake in the lift motor. This was unable to counteract the force of gravity. We then looked into using a PID controller to keep the lift motor's position. We tuned the built-in PID controller in the AndyMark motors so that it would keep its position with almost no oscillations.

We then added a magnetic limit switch on the bottom of the lift so that when the lift hits the bottom, it will stop the motor in order to help to prevent from stripping the belts. After that, we worked on an auto-home feature for the lift and flipper. This would allow the driver to quickly bring down the lift and flipper with one press of the button. Finally, while controller the harvester we found that modulating the harvester motors on and off helped a lot when trying to harvest stuck cubes. Thus, we added functionality to automatically modulate the harvester motors.

After Kent, we added code to manually control the jewel knocker so we could get the jewel out of the cryptobox if it got stuck in there for some reason. We also added a further toggleable slow mode used for getting the relic as well as a button on the driver's controller that automatically prepared the robot for parking (engaged the brake, brought up the flipper, and returned it to normal mode).

Since we added a relic recovery mechanism to the robot, we also added a toggleable relic mode to the auxiliary controller that allows the driver to control

the extension, gripper, and relic flipper precisely while at the same time being able to harvest cubes.

Detecting the Pictograms

We then started investigating Vuforia. We started to use Vuforia last year to detect the pictures on the field. Thus, we went to Vuforia's website and created a new zip file with the three new images in it. We used this to detect which column to move to. However, when we switched to OnBotJava we tried to get the zip file working with this but we realized that there was already a built-in Vuforia class for detecting the images - VuMark. This was a much simpler solution. Additionally, we decided that we needed to take multiple readings of the image in order to get a consistent reading, especially before we designed new phone mounts to get the picture in the frame better. We then picked the most seen image as the image that the autonomous thinks is there. This solution seems to work well even in our varied lighting conditions on our practice field. Here is the code we use to detect the images:

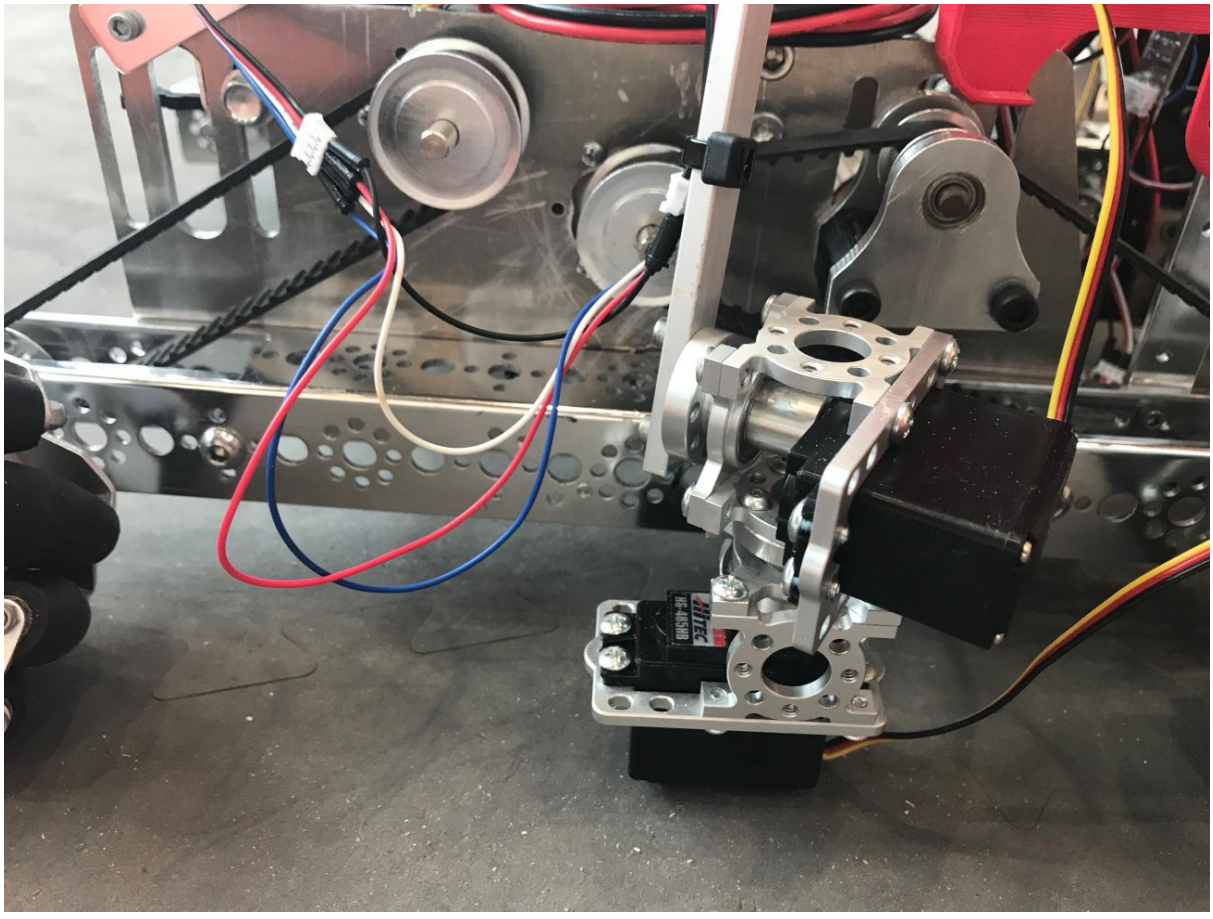
```
private int[] numTimesPicsSeen = {0, 0, 0};  // Left, right, center

int cameraMonitorViewId =
hardwareMap.appContext.getResources().getIdentifier("cameraMonitorViewId",
"id", hardwareMap.appContext.getPackageName());
VuforiaLocalizer.Parameters parameters = new
VuforiaLocalizer.Parameters(cameraMonitorViewId);
parameters.vuforiaLicenseKey = "...";
parameters.cameraDirection = VuforiaLocalizer.CameraDirection.BACK;
vuforia = ClassFactory.createVuforiaLocalizer(parameters);

VuforiaTrackables relicTrackables =
vuforia.loadTrackablesFromAsset("RelicVuMark");
relicTemplate = relicTrackables.get(0);
relicTemplate.setName("relicVuMarkTemplate");

private void countVisibleImages(){
```

```
   RelicRecoveryVuMark vuMark = RelicRecoveryVuMark.from(relicTemplate);

   numTimesPicsSeen[0] += (vuMark.equals(RelicRecoveryVuMark.LEFT) ? 1 :
0);
   numTimesPicsSeen[1] += (vuMark.equals(RelicRecoveryVuMark.RIGHT) ? 1 :
0);
   numTimesPicsSeen[2] += (vuMark.equals(RelicRecoveryVuMark.CENTER) ? 1 :
0);

}

private void resetNumTimesPicSeen(){

   numTimesPicsSeen[0] = 0;
   numTimesPicsSeen[1] = 0;
   numTimesPicsSeen[2] = 0;

}

private void sleepWhileCountingImages(int millis){

   float startingTime = System.currentTimeMillis();

   while(System.currentTimeMillis() - startingTime < millis){

       countVisibleImages();

   }

}

private int calculateVisibleImage(){

   float n = (float ) (numTimesPicsSeen[0] + numTimesPicsSeen[1] +
numTimesPicsSeen[2]);

   float probabilityLeft = ((float) numTimesPicsSeen[0]) / n;
   float probabilityRight = ((float) numTimesPicsSeen[1]) / n;
   float probabilityCenter = ((float) numTimesPicsSeen[2]) / n;
```

```
   if(probabilityLeft > probabilityRight && probabilityLeft >
probabilityCenter){

        return 0;

   }

   else if(probabilityRight > probabilityLeft && probabilityRight >
probabilityCenter){

        return 1;

   }

   else {

        return 2;

   }

}
```

Knocking the Jewels

After attempting to put a glyph in the cryptobox, we realized that moving the robot in order to knock the jewel would be very inconsistent. Thus, we simply attached another servo to the first servo and we found the correct values so that it would knock the correct jewel off. This has made our autonomous much more consistent. This is our first prototype:



The first prototype of our jewel knocker.

At first, we used Neverest 3.7’s for the drivetrain. These made the robot very fast, but also very slippery, which made autonomous very inconsistent. We decided to switch to Neverest 20s. Although still slippery, using the encoders and the gyro and driving slowly we can move consistently.

Over winter break we developed the bulk of the autonomous for our robot. For the cryptobox that is nearest to the drivers, we read the image, knock the jewel, drive forward, turn using the gyro sensor, then strafed according to picture it saw. From there, it drives back and forth in an attempt to hit the glyph in if it missed a bit during its first attempt. It then drives into the pit and harvests glyphs and then drives back and attempts to deposit it.

Harvesting Multiple Glyphs

We found that using a simple harvesting strategy, the glyphs often got jammed. Thus, we tried a number of strategies to unjam the glyphs while harvesting.

We tried multiple iterations of stuff to do when we detected the harvester was jammed using the encoders on the harvester motors.

```
while(opModeIsActive() && time.milliseconds() < 10000 &&
consecutiveTimesSeenGlyph < 4){

    if(robot.glyphSensor.getLightDetected() > 0.65f){

        consecutiveTimesSeenGlyph++;

    }

    else{

        consecutiveTimesSeenGlyph = 0;

    }

    boolean jammed =
rightHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200 &&
leftHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200;
    boolean enoughTimeToSpoolUp = time.milliseconds() > 200 &&
backupCooldown.milliseconds() > 600 && state != 3 &&
collectingTimer.milliseconds() > 300;

    if(jammed && enoughTimeToSpoolUp){
```

```
        state = 3;
        rightHarvesterMotorEx.setVelocity(-100,
AngleUnit.DEGREES);
        leftHarvesterMotorEx.setVelocity(-100,
AngleUnit.DEGREES);
        collectingTimer.reset();
        robot.setPower(0f);

    }

    if(state == 3){

        // first do a few quick modulations then back up
        if(collectingTimer.milliseconds() < 1000){

            robot.setPower(-0.1f);

            int modulatingState = (int)
(collectingTimer.milliseconds() / 250);

            if(modulatingState % 2 == 0){

                rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

            }

            else{

                rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
```

```
                leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

            }

        }

        else{

            if(collectingTimer.milliseconds() < 1600){

                robot.setPower(0.1f);

                rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

            }

            else{

                robot.setPower(-0.1f);
                rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

            }

            if(collectingTimer.milliseconds() > 2000){

                rightHarvesterMotorEx.setVelocity(0,
AngleUnit.DEGREES);
```

```
                leftHarvesterMotorEx.setVelocity(0,
AngleUnit.DEGREES);
                breakingSleep(500);

                state = 0;
                robot.setPower(-0.2f);
                backupCooldown.reset();

            }

        }

    }

    else{

        rightHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);
        leftHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);

    }

}
```

This is another strategy we tried:

```
while(opModeIsActive() && time.milliseconds() < 10000 &&
consecutiveTimesSeenGlyph < 4){

    if(robot.glyphSensor.getLightDetected() > 0.65f){

        consecutiveTimesSeenGlyph++;

    }

    else{

        consecutiveTimesSeenGlyph = 0;

    }

    boolean jammed =
rightHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200 &&
leftHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200;
    boolean enoughTimeToSpoolUp = time.milliseconds() > 200 &&
backupCooldown.milliseconds() > 600 && state != 3 &&
collectingTimer.milliseconds() > 300;

    if(jammed && enoughTimeToSpoolUp){

        state = 3;
        rightHarvesterMotorEx.setVelocity(-100,
AngleUnit.DEGREES);
        leftHarvesterMotorEx.setVelocity(-100,
AngleUnit.DEGREES);
        collectingTimer.reset();
        robot.setPower(0f);
```

```
    }

    if(state == 3){

        // first do a few quick modulations then back up
        if(collectingTimer.milliseconds() < 1000){

            robot.setPower(-0.1f);

            int modulatingState = (int)
(collectingTimer.milliseconds() / 250);

            if(modulatingState % 2 == 0){

                rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

            }

            else{

                rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

            }

        }

        else{
```

```
            if(collectingTimer.milliseconds() < 1600){

                robot.setPower(0.1f);

                rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

            }

            else{

                robot.setPower(-0.1f);
                rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

            }

            if(collectingTimer.milliseconds() > 2000){

                rightHarvesterMotorEx.setVelocity(0,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(0,
AngleUnit.DEGREES);
                breakingSleep(500);

                state = 0;
                robot.setPower(-0.2f);
                backupCooldown.reset();

            }
```

```
        }

    }

    else{

        rightHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);
        leftHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);

    }

}
```

And a final strategy we tried:

```
while(opModeIsActive() && time.milliseconds() < 20000 &&
consecutiveTimesSeenGlyph < 5){

    if(robot.glyphSensor.getLightDetected() > 0.65f){

        consecutiveTimesSeenGlyph++;

    }

    else{

        consecutiveTimesSeenGlyph = 0;

    }
```

```
    boolean jammed =
rightHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200 &&
leftHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200;
    boolean enoughTimeToSpoolUp = time.milliseconds() > 200 &&
backupCooldown.milliseconds() > 600 && state != 3 && state == 0
&& collectingTimer.milliseconds() > 300;

    if(jammed && enoughTimeToSpoolUp){

        state = 3;
        collectingTimer.reset();
        robot.setPower(0.2f);

    }

    if(state == 3){

        // first do 5 quick modulations then back up

        if(collectingTimer.milliseconds() < 500){

            int modulatingState = (int)
(collectingTimer.milliseconds() / 100);

            if(modulatingState % 2 == 0){

                rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

            }

            else{
```

```
                rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
                leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

            }

        }

        else{

            rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

            if(collectingTimer.milliseconds() > 600){

                state = 0;
                robot.setPower(-0.2f);
                backupCooldown.reset();

            }

        }

    }

    else{

        if((state == 0 && collectingTimer.milliseconds() > 300)
|| (state != 0 && collectingTimer.milliseconds() > 150)){
```

```
            state = state % 2;
            collectingTimer.reset();

        }

        if(state == 0){

            rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

        }

        else if(state == 1){

            rightHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);

        }

        else if(state == 2){

            rightHarvesterMotorEx.setVelocity(700,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(-700,
AngleUnit.DEGREES);

        }

    }
```

```
    telemetry.addData("right",
rightHarvesterMotorEx.getVelocity(AngleUnit.DEGREES));
    telemetry.addData("left",
leftHarvesterMotorEx.getVelocity(AngleUnit.DEGREES));
    telemetry.addData("time", time.milliseconds());
    telemetry.addData("collecting timer",
collectingTimer.milliseconds());
    telemetry.addData("state", state);
    telemetry.addData("FR encoder",
robot.frontRight.getCurrentPosition());
    telemetry.addData("Visibility matrix (L, R, C)",
numTimesPicsSeen[0] + " " + numTimesPicsSeen[1] + " " +
numTimesPicsSeen[2]);
    telemetry.update();

}
```

We found that by driving into the glyph pit to separate the glyphs along with the following harvesting strategy, we could consistently get 2 extra glyphs.

```
while(opModeIsActive() && time.milliseconds() < 6010 &&
consecutiveTimesSeenGlyph < 2){

    telemetry.addData("pos",
robot.frontRight.getCurrentPosition());
    telemetry.update();

    if(robot.glyphSensor.getLightDetected() > 0.55f){

        consecutiveTimesSeenGlyph++;

    }
```

```
    else{

        consecutiveTimesSeenGlyph = 0;

    }

    boolean jammed =
rightHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200 &&
leftHarvesterMotorEx.getVelocity(AngleUnit.DEGREES) < 200;
    boolean cooldownConditionsMet = time.milliseconds() > 200 &&
state != 3 && collectingTimer.milliseconds() > 800;

    if(jammed && cooldownConditionsMet){

        state = 3;
        collectingTimer.reset();
        robot.setPower(0.3f);
        unjammingMode = (unjammingMode + 1) % 3;

    }

    if(state == 3){

        if(unjammingMode == 0){

            rightHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(0,
AngleUnit.DEGREES);

            if(collectingTimer.milliseconds() > 500){

                collectingTimer.reset();
                state = 0;
```

```
                robot.setPower(-0.15f);

            }

        }

        else if(unjammingMode == 1){

            rightHarvesterMotorEx.setVelocity(0,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);

            if(collectingTimer.milliseconds() > 500){

                collectingTimer.reset();
                state = 0;
                robot.setPower(-0.15f);

            }

        }

        else if(unjammingMode == 2){

            rightHarvesterMotorEx.setVelocity(-900,
AngleUnit.DEGREES);
            leftHarvesterMotorEx.setVelocity(-900,
AngleUnit.DEGREES);

            if(collectingTimer.milliseconds() > 550){

                collectingTimer.reset();
                state = 0;
```

```
                robot.setPower(-0.15f);

            }

        }

    }

    else{

        rightHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);
        leftHarvesterMotorEx.setVelocity(800,
AngleUnit.DEGREES);

    }

}
```

Reliability

We use multiple sensors on our robot to make sure that our autonomous is consistent. Besides testing vision, we average all four encoders in order to get an accurate measure of distance. Furthermore, we use PID turning in order to have a fast, smooth, and accurate turn. We use the gyro in order to measure the error. Furthermore, we square our robot up against the cryptobox and record that angle. We later use that angle to help to try to get back to the cryptobox after harvesting more cubes so we can potentially put them into the cryptobox and so that we ensure a park. Finally, we maximize the area of our robot so that we have a high chance of parking if anything goes wrong.

Vision

After Cleveland we wanted to work on incorporating vision into our autonomous besides just the Vuforia VuMark detector. We looked into DogeCV, a vision library using OpenCV made by GTHS Robotics. We wanted to use the library for two main purposes: lining up with the cryptobox during autonomous (and later teleop) and consistently harvesting second and third cubes during autonomous. Our first challenge was seeing the cryptobox in the first place. Our phone was mounted on the wrong side of the robot for it to be able to see the cryptobox and the glyph pit.

We investigated using vision to line up then turning the robot or using mirrors, but we designed a solid and simple phone mount that can be pivoted by a servo. This allows us to see the cryptogram during the first seconds of autonomous then flip the phone out so we can see both the cryptobox with the front and back cameras on the phone respectively.

We used the DogeCV cryptobox detector to detect the cryptoboxes. We found three big issues with this solution. First, the robot had to be fairly far away from the cryptobox in order to see it. Thus, we had to back up before using vision. Second, the algorithm was not accurate enough to be repeatable. Finally, sometimes the algorithm would not see one of the columns so instead of going for center or right it would go for left or center respectively. Furthermore, the DogeCV glyph detector was inaccurate under most lighting conditions.

This is an example of the DogeCV detecting the cryptoboxes.

We talked extensively with Alex, the lead programmer on GTHS robotics, about ways to improve the OpenCV algorithms. However, GTHS robotics and their lead programmer, Alex, announced that they would not be further developing DogeCV.

Thus, we looked into modifying the DogeCV algorithms. Instead of simply trying to create a mask of where the cryptobox dividers are, we tried further trying to extract where the center of the edges were using a contrast sensitive filter on the region DogeCV identified. We also reduced the blurring operation that DogeCV performs on the image. We used the OpenCV histogram methods to calculate this high contrast mask. Once we calculated this high contrast mask, finding the ridge in the cryptobox divider was much easier.

This solution worked fairly well, but we still had to back up fairly far in order to see the cryptoboxes. We also added a tracking history so we wouldn't confuse the center column for the left one or the right one for the center one. With the

competition fast approaching, we decided to stick with our current encoder and gyro based autonomous because at this point it is more consistent, especially considering that we haven't fully tested our vision algorithm under different lighting conditions. This is the main filter (besides the DogeCV that is calling it):

```
Imgproc.cvtColor(input, input, Imgproc.COLOR_RGB2Lab);
Imgproc.GaussianBlur(input,input,new Size(3,3), 0);
Core.split(input, channels);
Imgproc.threshold(channels.get(1), mask, threshold, 255,
Imgproc.THRESH_BINARY);
Imgproc.equalizeHist(threshold, q);
Imgproc.dilate(q, output, contrastKernel);
```

This is an example of DogeCV detecting the glyphs.

Despite further investigation into using vision, because of the time constraints we decided to stick with just using sensors in order to get multiple glyphs in autonomous.

Cleveland Qualifier

At the Cleveland qualifier our autonomous performed very well. We scored an average of 87 points, getting an 85 point autonomous every round and one match we got an extra glyph. However, we noticed that it would have been much more helpful to get the glyph into the column perfectly as this would have made teleop much easier.

Kent Qualifier

Our autonomous performed very well at the Kent qualifier, earning an average of 1.3 glyphs per round. Additionally, we won the control award for our multi-glyph autonomous and teleop driver enhancements. However, we ran in to one major issue: we were given major penalties for controlling three glyphs and scoring while controlling three glyphs. Thus, after Kent we worked primarily on ensuring that this would never happen again by driving less into the glyph pit. We also added a multi-glyph autonomous for the corner cryptobox.

Ohio States

At the state tournament, we had a 1.5 glyph average. However, on the near box, we had a 3 glyph average. We knew that we needed to improve the reliability of getting extra glyphs. The main issue was the harvester jamming.

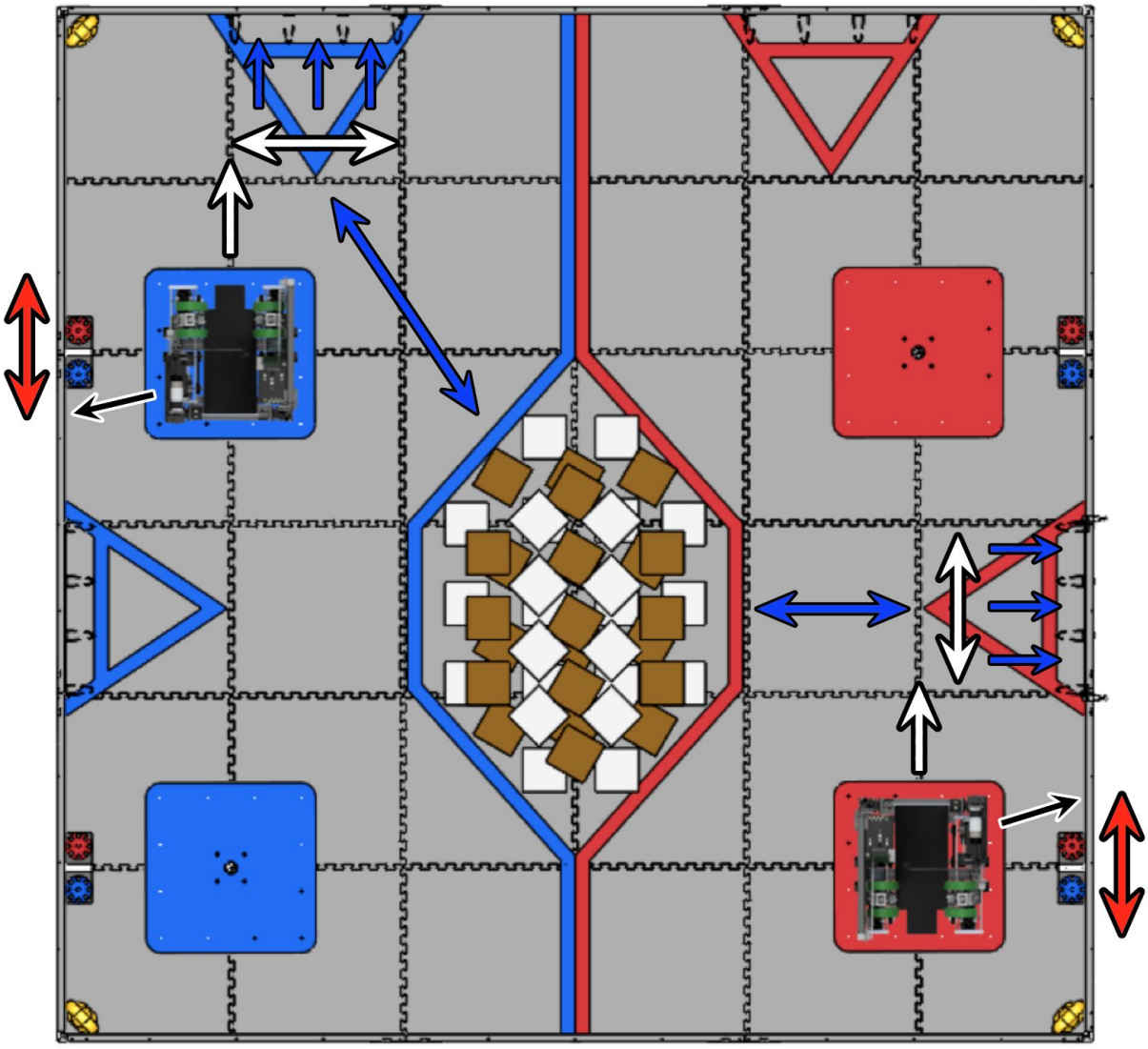
Autonomous Strategy

We have four autonomous programs for the four possible starting positions. The blue and red side autonomous programs are mirrors of each other, with small changes to accommodate the asymmetry.

For our near side position (starting on the balancing stone closer to the bottom of the field, when referencing the Control Award field diagram), we first extend our jewel knocking mechanism, read the color of the jewel, and knock off the appropriate jewel. We then read the pictogram image using Vuforia and drive forward. For consistency, while driving forward we tilt the phone so that it is looking backwards a bit and take two more readings of the pictograms. We then use the gyro to turn 90˚. After that, we strafe in order to line up with the appropriate cryptobox column. We hit against the cryptobox and record the gyro angle at that point for later use. We deposit the glyph and hit the cryptobox again in order to make sure that the glyph is deposited well into the cryptobox. We realign with the earlier recorded squared angle. We then enter the glyph pit while harvesting and modulating the harvester motors (a strategy that helps us harvest better). We again square up with the earlier recorded angle. Finally, we return to the cryptobox, deposit any extra glyphs, and extend the jewel knocker in order to maximize our chances of parking in the safe zone.

In the far side position (the upper balancing stones with reference to the Control Award field diagram), we again knock the jewel and read the image three times using Vuforia while driving off the stone. We then strafe to the appropriate cryptobox column and deposit the glyph. We use the gyro in order

to turn so that we will enter the glyph pit, but not incur a penalty by crossing the centerline. We enter the pit, attempt to harvest multiple glyphs, and return, again using the gyro for consistency and deposit any extra glyphs and park in the safe zone.

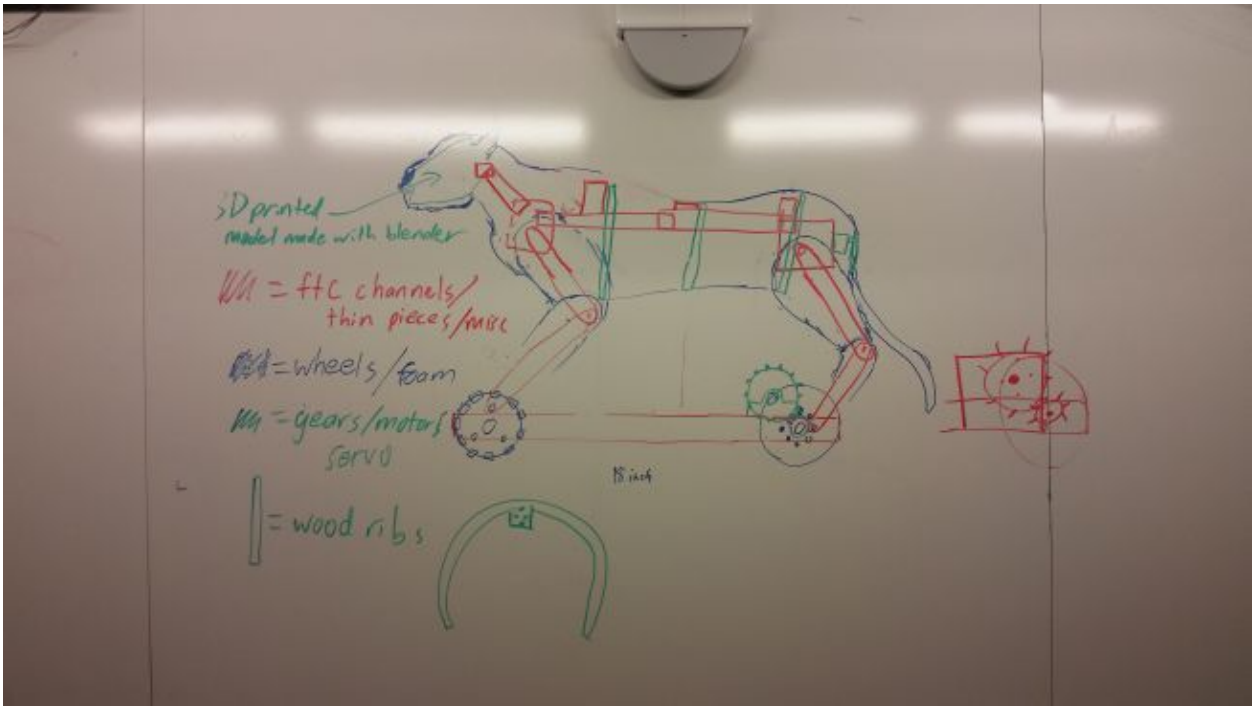


This is a diagram of our autonomous strategy, where the two starting positions not depicted are mirrors of the depicted starting positions.

Weekly Design Logs

8/21/17

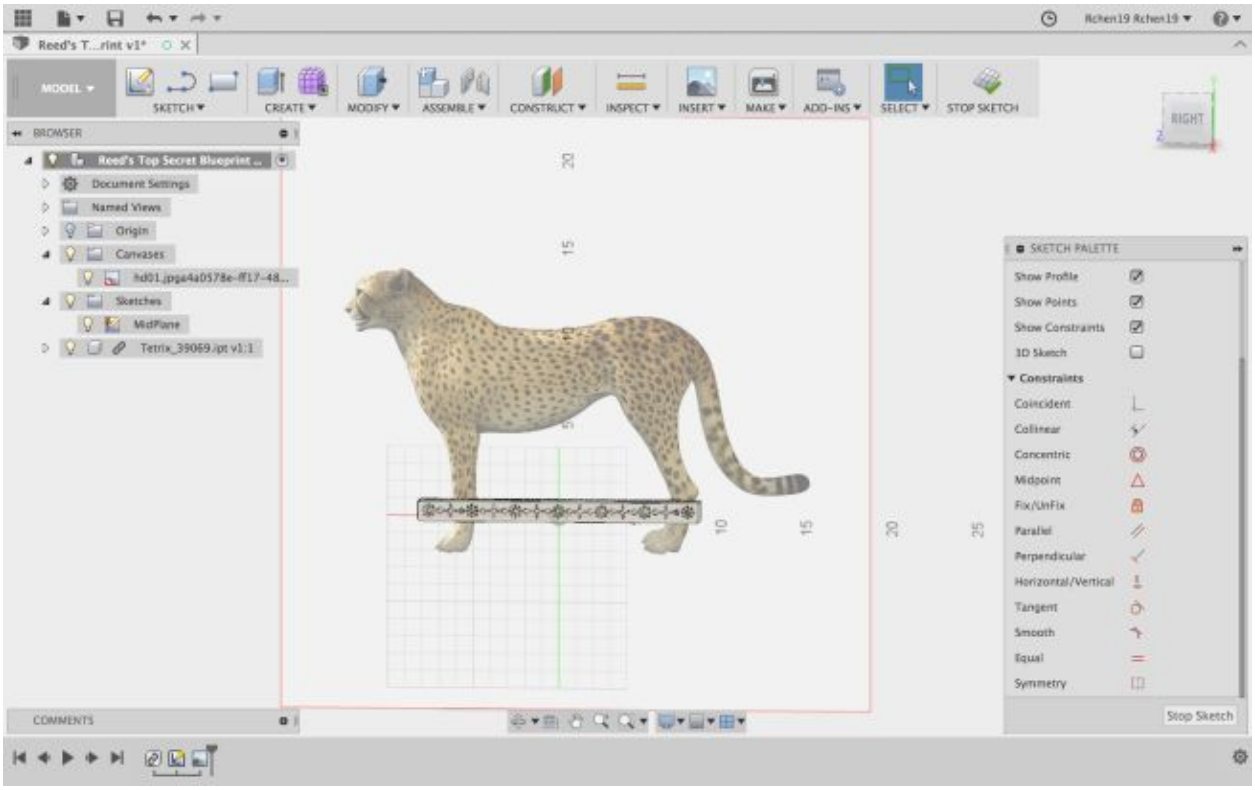
Because the teaser was jungle themed, we decided to do a jungle themed robot that would be fast and would have actuating legs.



Some goals:

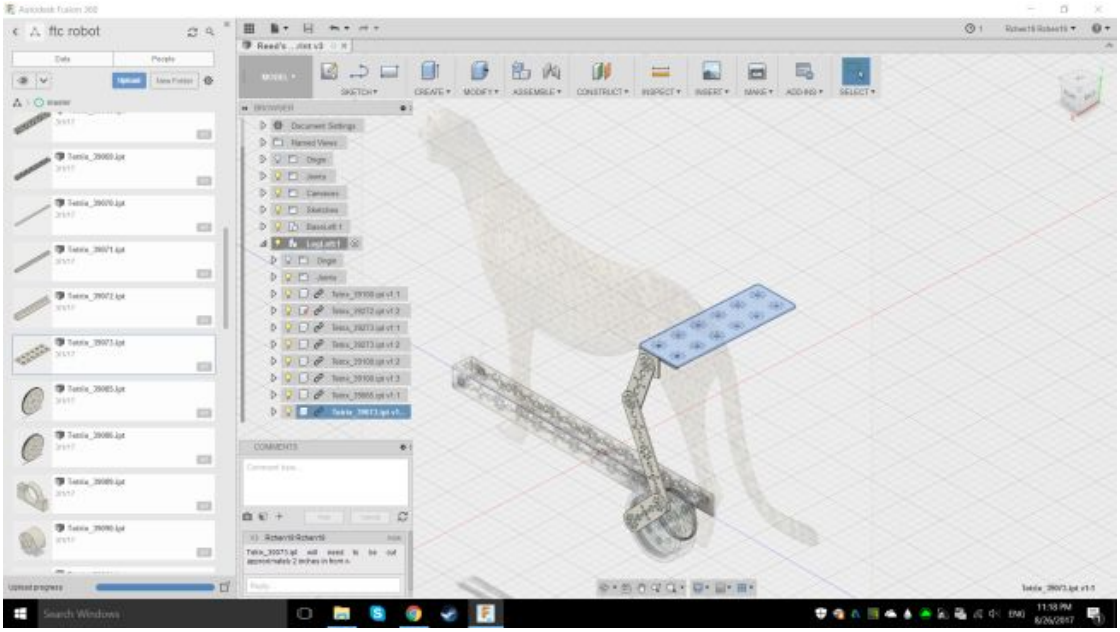
- Rear wheel drive
- Geared
- Omniwheels in the front
- Skeleton and chassis made from Tetrix kit
- Body made from custom wood, foam, and 3d-printed pieces
- Legs move with wheels
- Tail attached to servo

Reference image



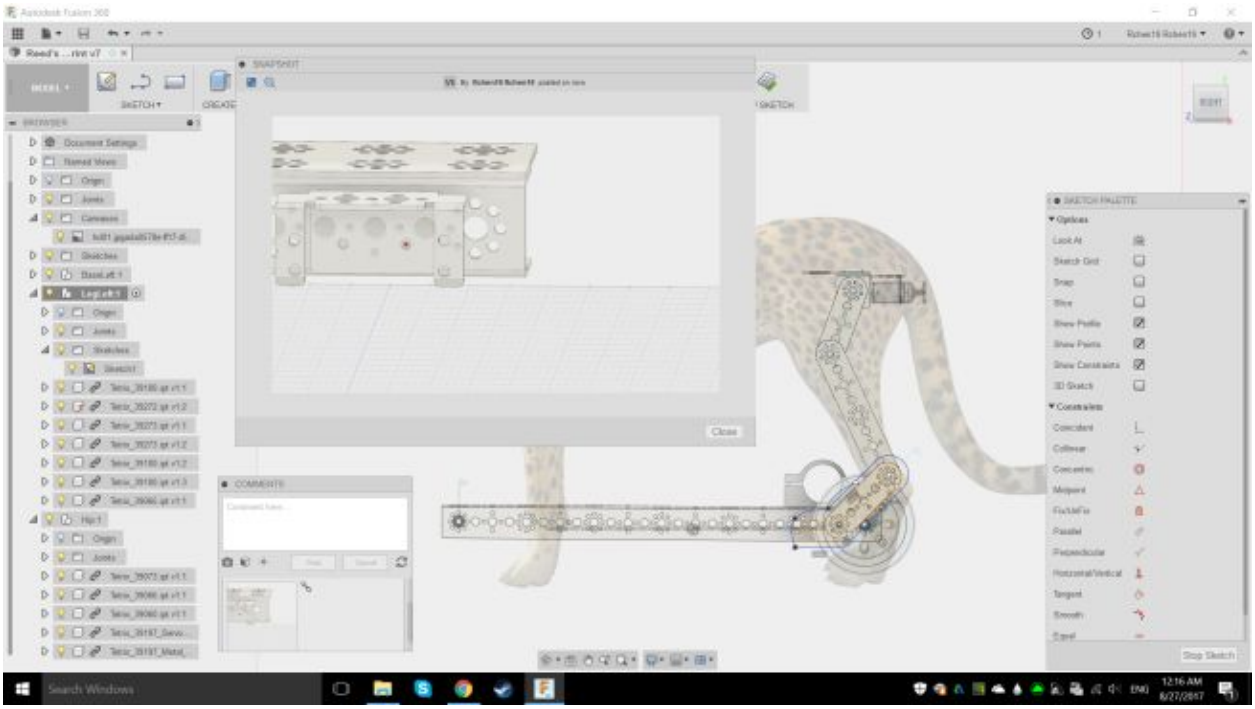
Getting set up

- Change units to inches
- Insert C-Channel for reference
- Insert reference picture as canvas



To create the hind leg:

- Inserted more Tetrix components
- Used joints to assemble
- Removed history to organize hierarchy

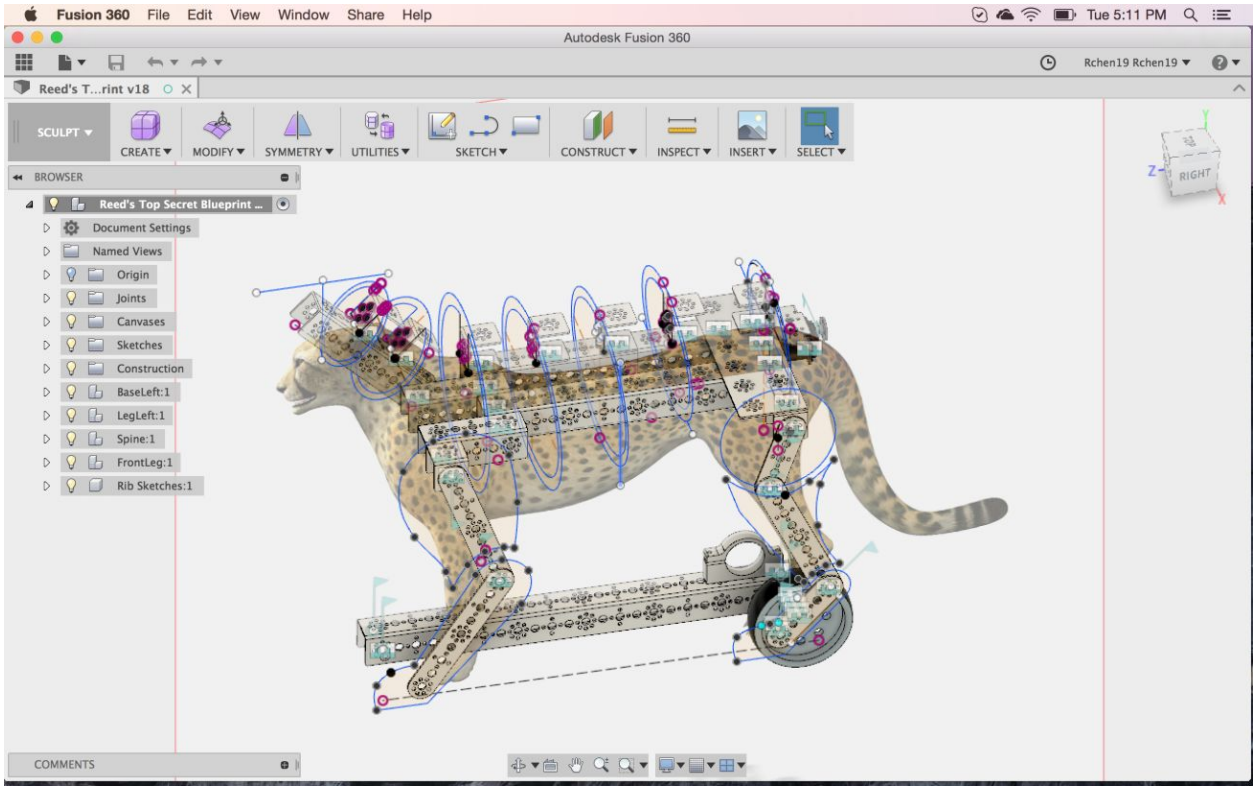


To create the hip:

- Added servo for tail
- Added comments with locations for modifying Tetrix pieces for the Cheetah Robot
- Began sketch on hind leg for foam body

8/28/17

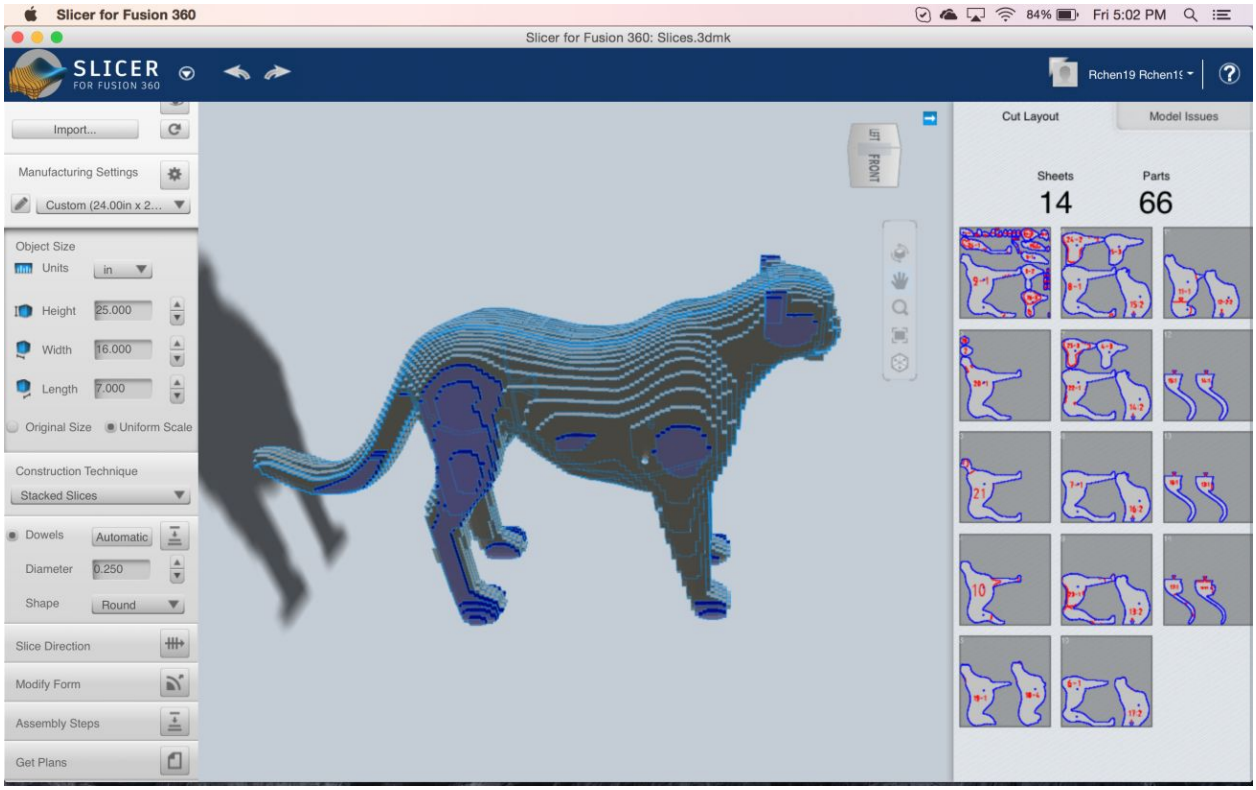
We finished designing the cheetah robot in CAD.



In order to make it look good, we wanted to cut cross sections out of foam. To create the ribs:

- The ring sketches will be laser cut from wood to provide support for the foam.
- They are mounted to 1-inch C-Channels that are mounted to a modified spine.

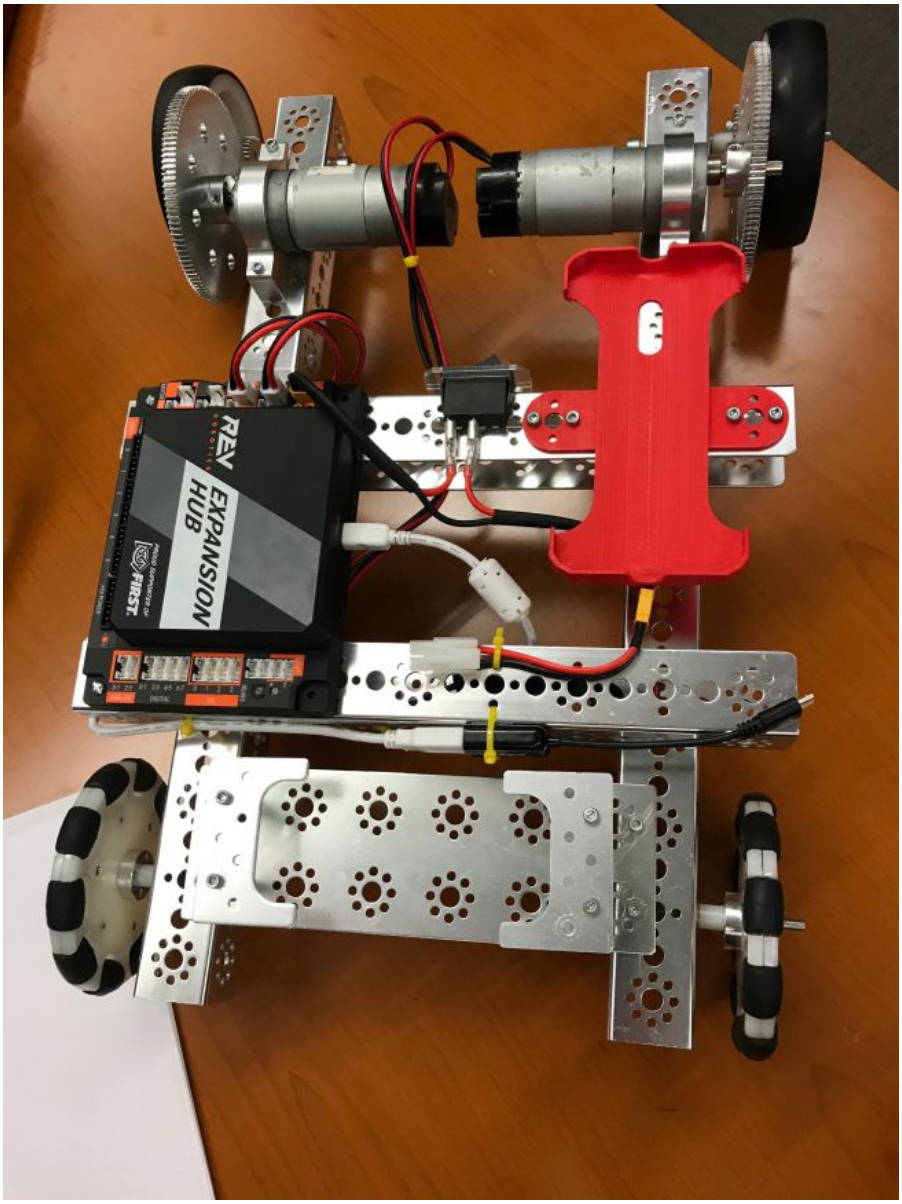
- The head will be made from a separate model. It will be composed of layered foam. The T sketch is a measurement of the dimensions required.



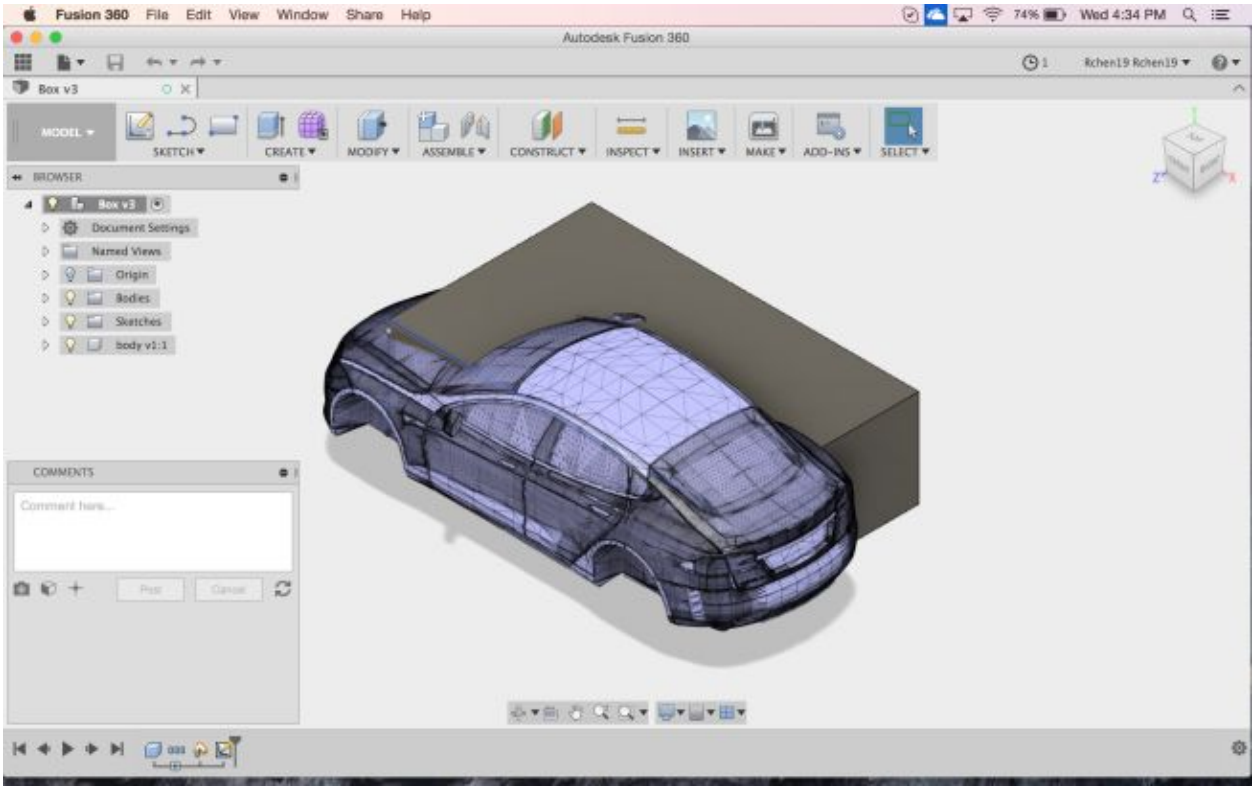
Slicer for Fusion 360 is a program that slices models into layers, along you to create those models through methods such as laser cutting wood or CNC foam.

9/4/17

We finished building the chassis.

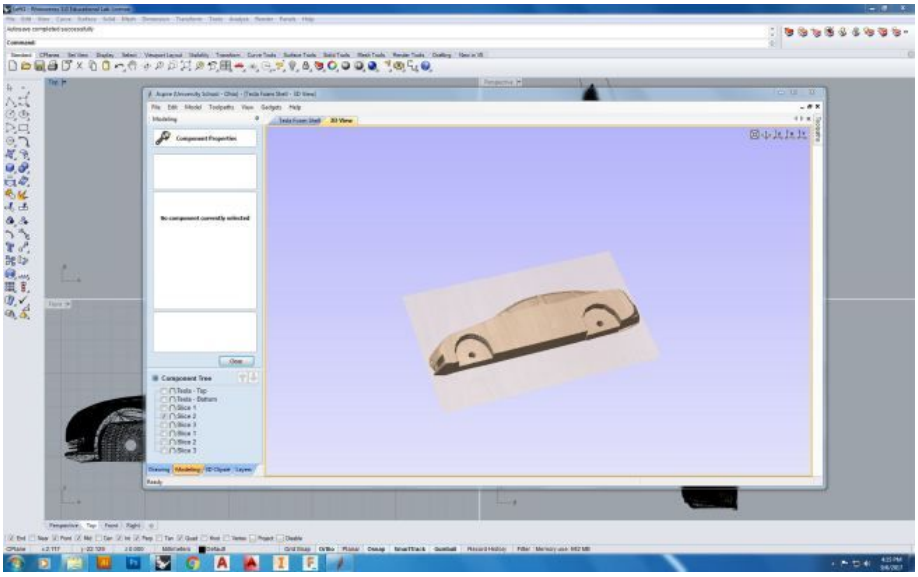


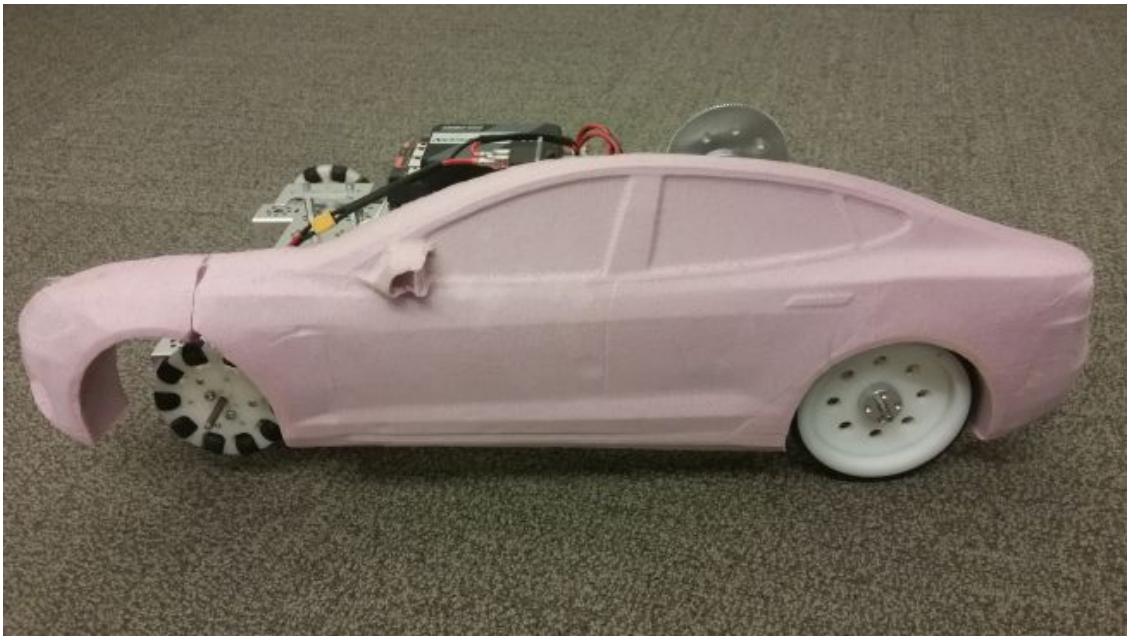
For the foam shell we decided to switch to a Tesla model.



Gray box is the approximate size of our chassis.

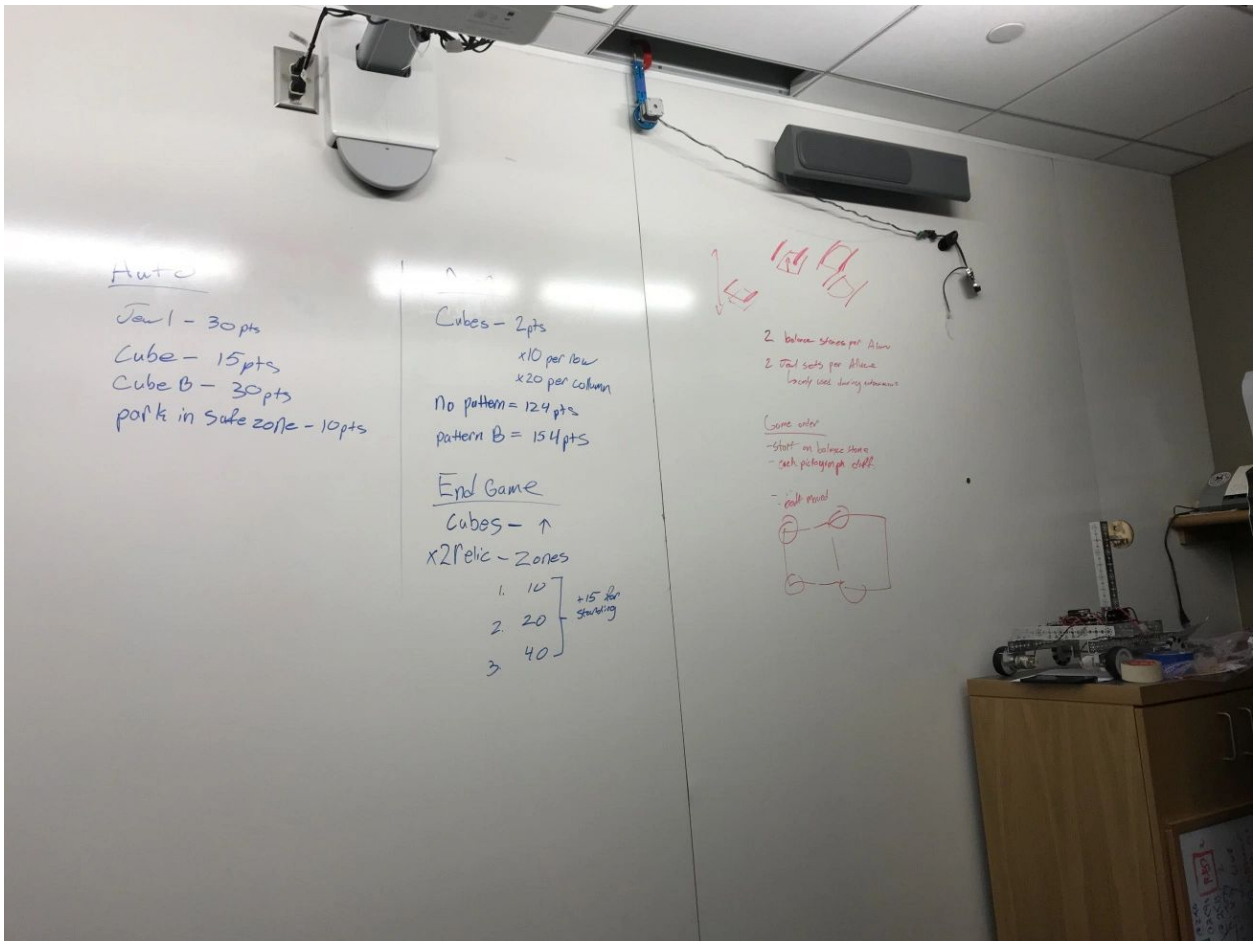
After exporting the mesh as an OBJ, we imported the model into Aspire, a CNC program, where we sliced the model into 2-inch thick slices, the thickness of the foam we will be using.





9/11/17

We did a lot of brainstorming about some ideas for the robot following the game reveal. We first looked at what was important in the game:



We then built the field.

Finally, we came up with some initial ideas for a robot:

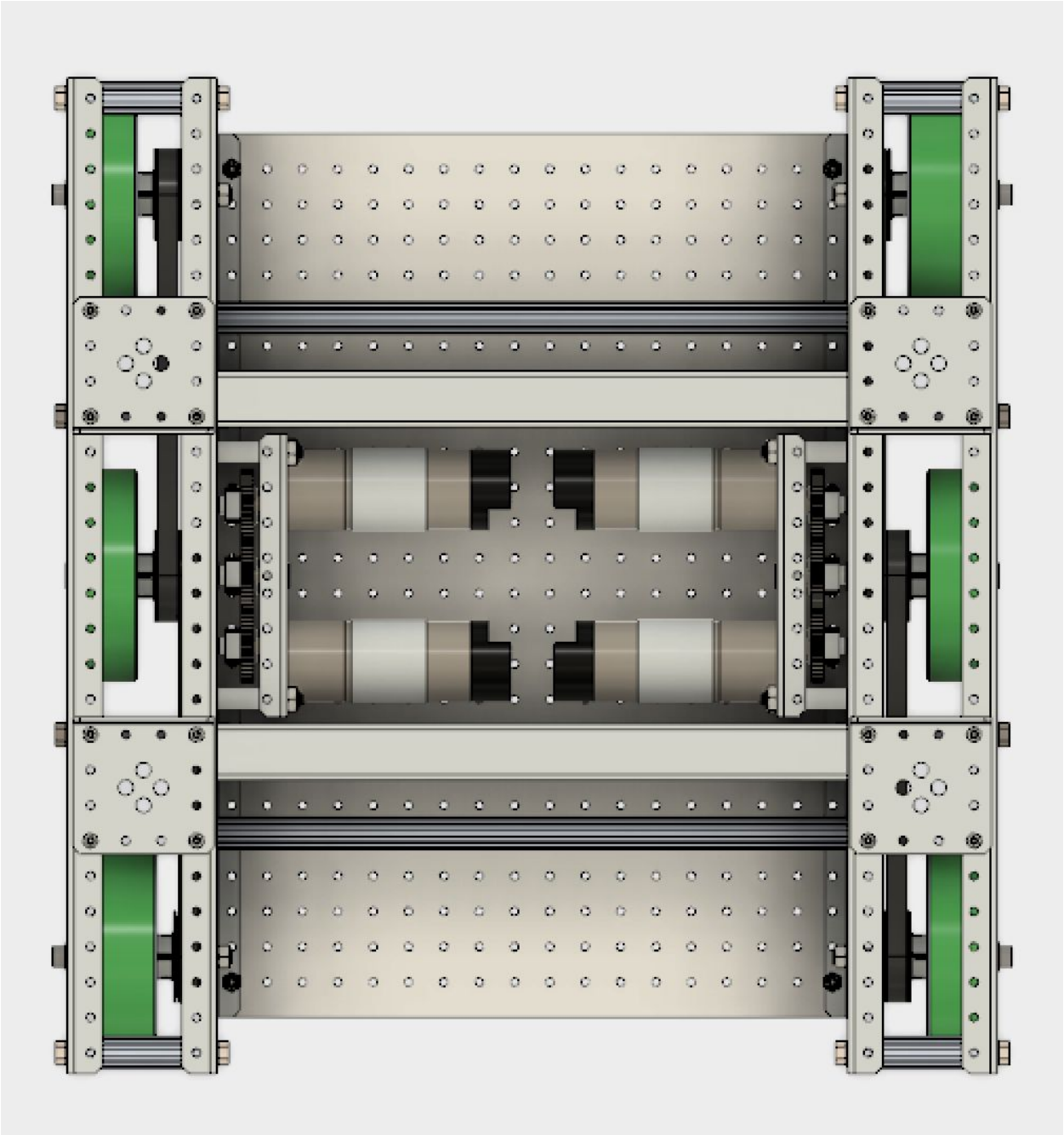
- Six wheel chassis
    - The two middle wheels will be stealth wheels, while the rest on the front and back are omni-wheels
    - They will be lined up on the same plan, unlike the West Coast drive-train

        - The way the robot rocks back and forth with a West Coast design may cause instability, hampering the autonomous mode
    - With this design, however, it will be stable and have a small turn radius similar to the West Coast design due to the use of omni-wheels
    - This design will need to be tested against the West Coast design to determine which is better.
- 3-D printer style lift
    - The mechanism for lifting the ball will be similar to a 3-D printer, capable of moving in the Y and X axes using rods, circular bearings, and belts
        - The ability to move along the X axes will allow us to insert the glyphs even if we are not perfectly aligned, saving precious time. If we are not perpendicular to the wall
        - We can rotate to become perpendicular, then move along the X axes to realign the glyph to be inserted
        - It may allow us to insert the grabber between the jewels and move along the X axes to push them off

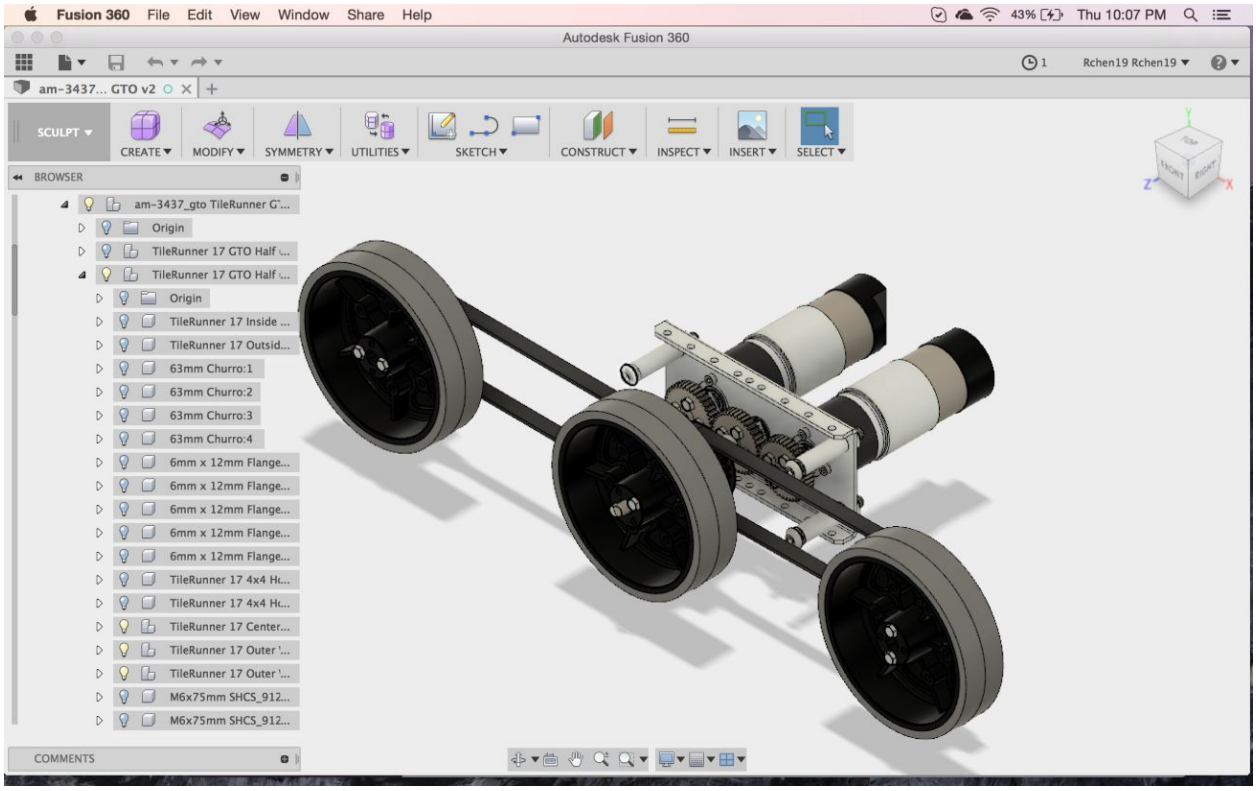
        - However, this is risky due to the limited space between the jewels. With the large number of points involved, it is probably better to make a separate mechanism for this
- The grabber mechanism may be two servos that cause the grabber mechanism to clamp onto the glyph
    - It will have a depth of around two inches, how much the glyph protrudes from the sides of the cryptobox
    - Because the cryptobox is taller than the robot, we believe the grabber should be able to pick up two glyphs at a time by first stacking one upon another, then picking both up
        - This will allow us to place the top row of the cryptobox without an additional mechanism, simplifying the design and making it lighter, faster, and more efficient
        - But this may cause issues if three out of the four spaces of a column are filled up, which may require us to remove the third block before completing the column.

9/18/17

We tried to create a drivetrain similar to AndyMark's TileRunner.

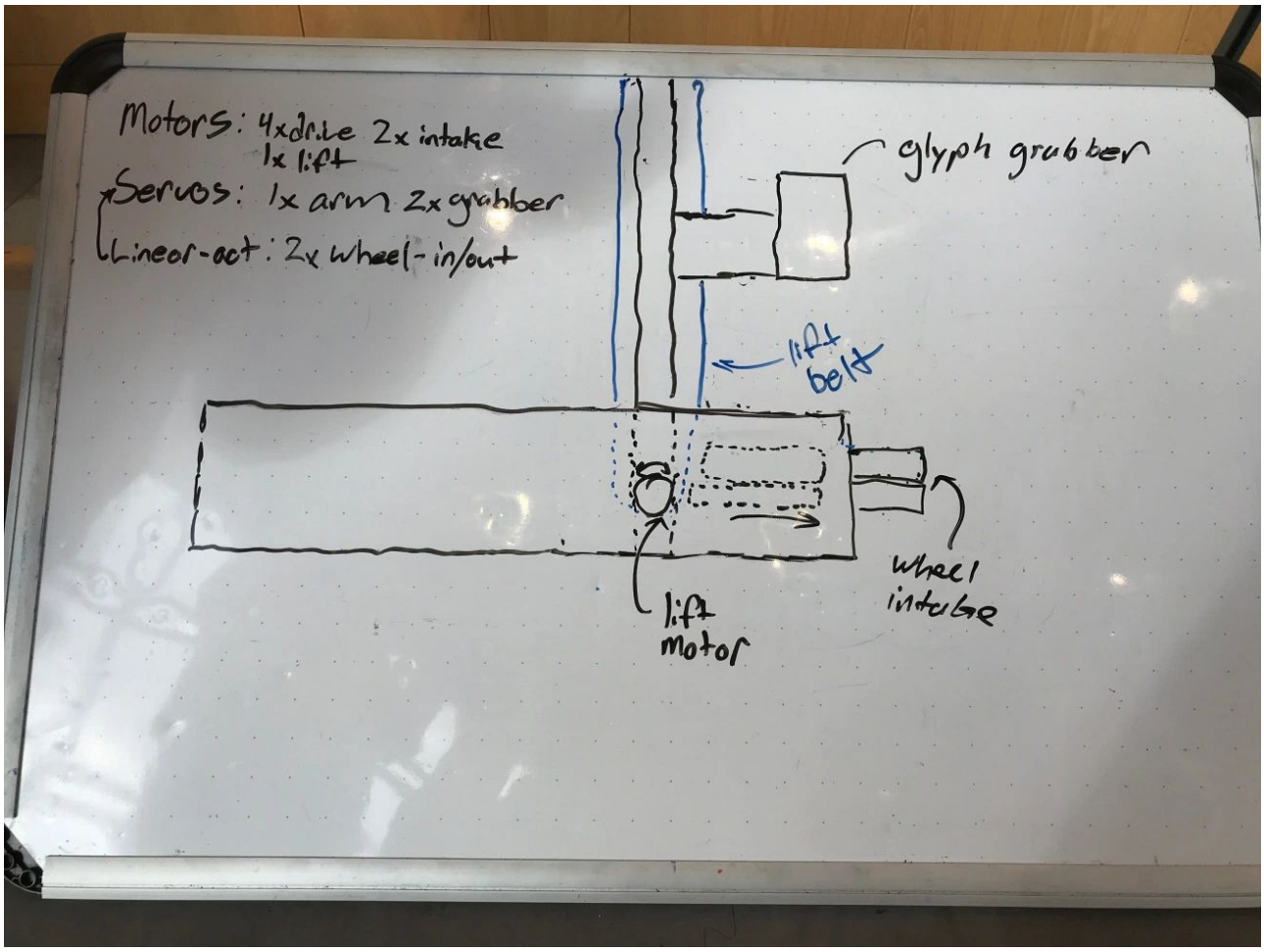


We determined which parts were used in the TileRunner to see if it would be more cost effective to buy the kit or just individual parts. We figured out what we could make by ourselves:



We determined it would be more cost-effective to buy the TileRunner kit.

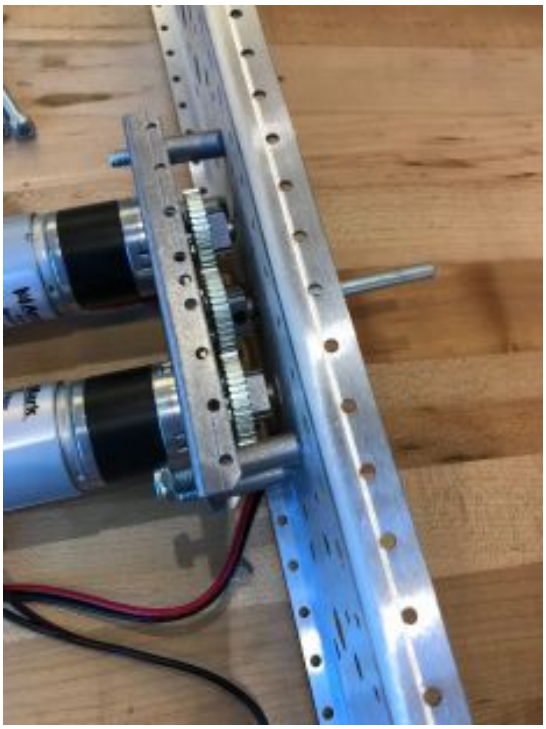
We also worked on the glyph mechanism, coming up with the following idea:

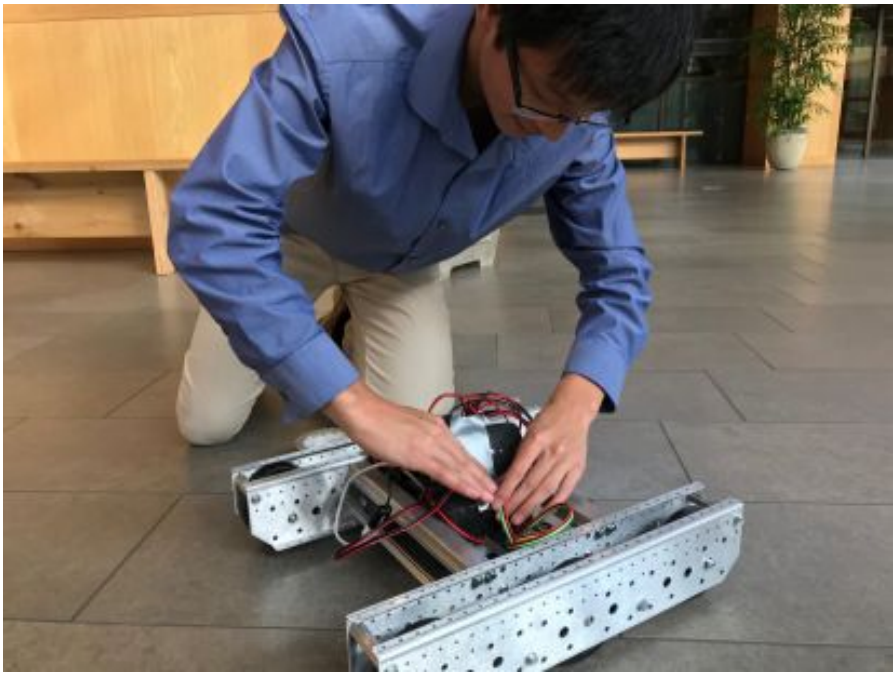


Having both a harvester and lift means we will be able to hold 2 blocks at a time. The belt lift is relatively simple. Having the harvester in the front instead of the back means we free up space in the back and avoid issues with transporting the glyph to the lift. However, we will have to rotate each time to get a glyph, slightly reducing speed.

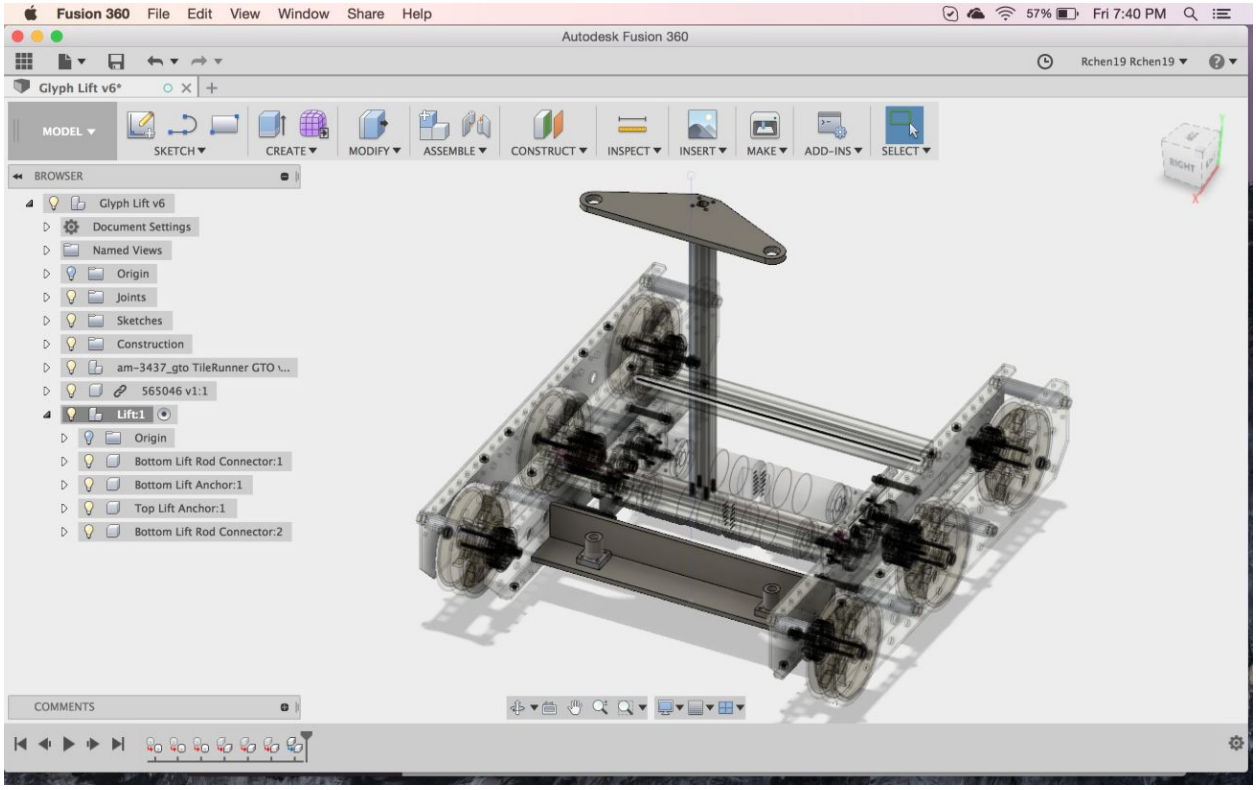
9/25/17

This week we built the TileRunner chassis.



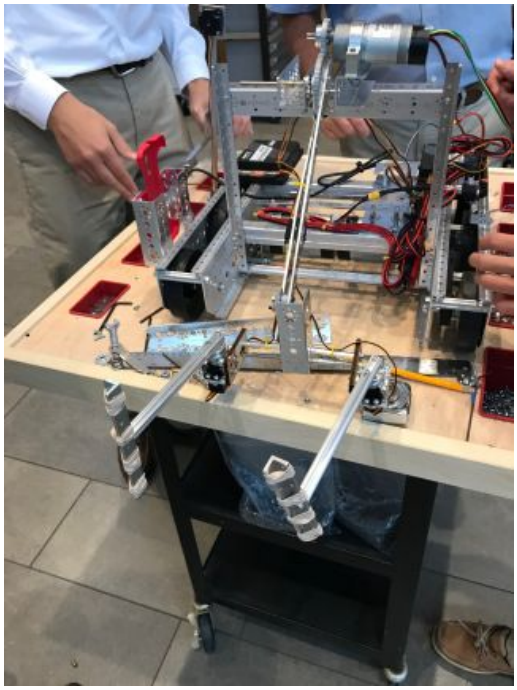


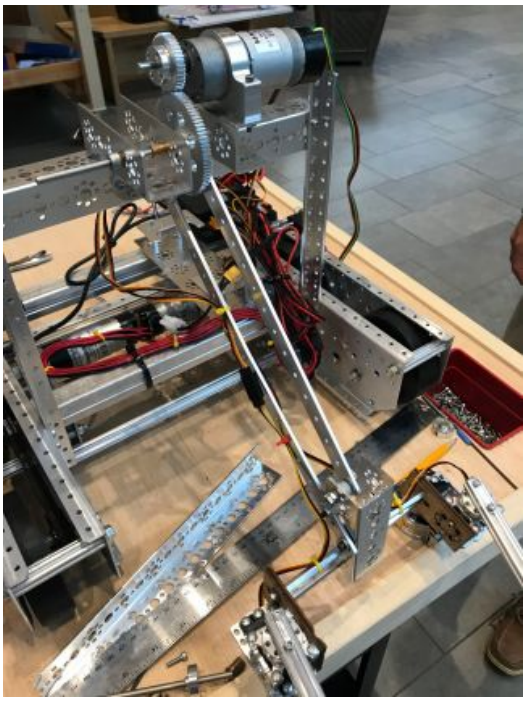
We also started to mock up the glyph lift in Fusion.



10/2/17

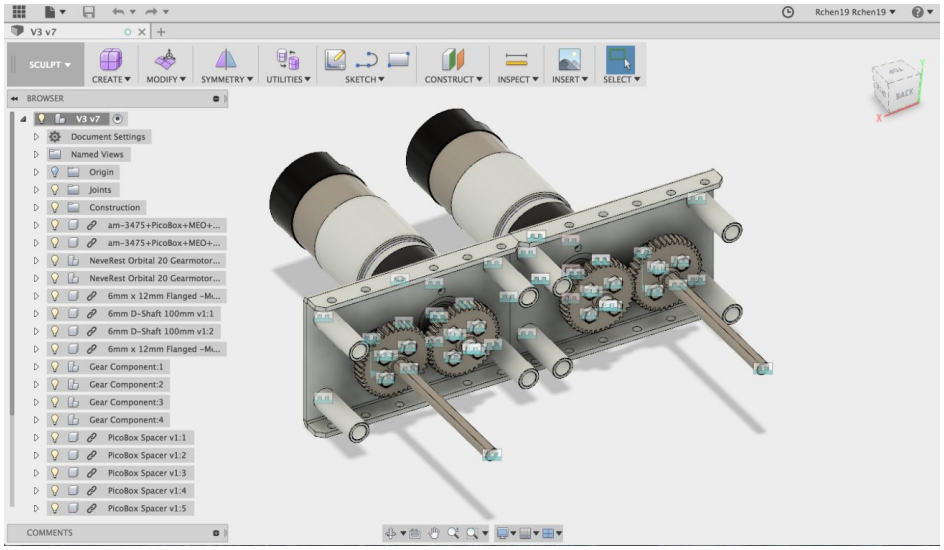
We used the TileRunner chassis to make a simple gripper mechanism with a lift.



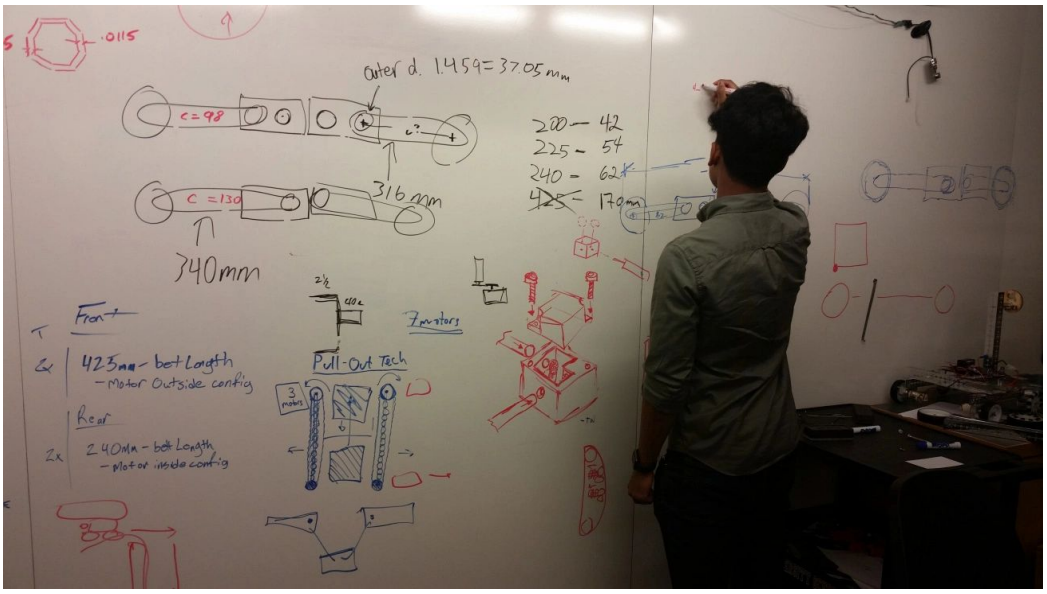


After attaching a makeshift arm to the Tilerunner chassis, we decided to use mecanum wheels and a linear lift. This was because the arm was not stable and wobbled a lot, the distance changed as you lifted the arm, and it took up a large amount of space. The mecanum wheels were much more agile than stealth wheels, especially when placing glyphs in the cryptobox.

We then began designing a gearbox to keep the motors for the Mecanum chassis centralized.

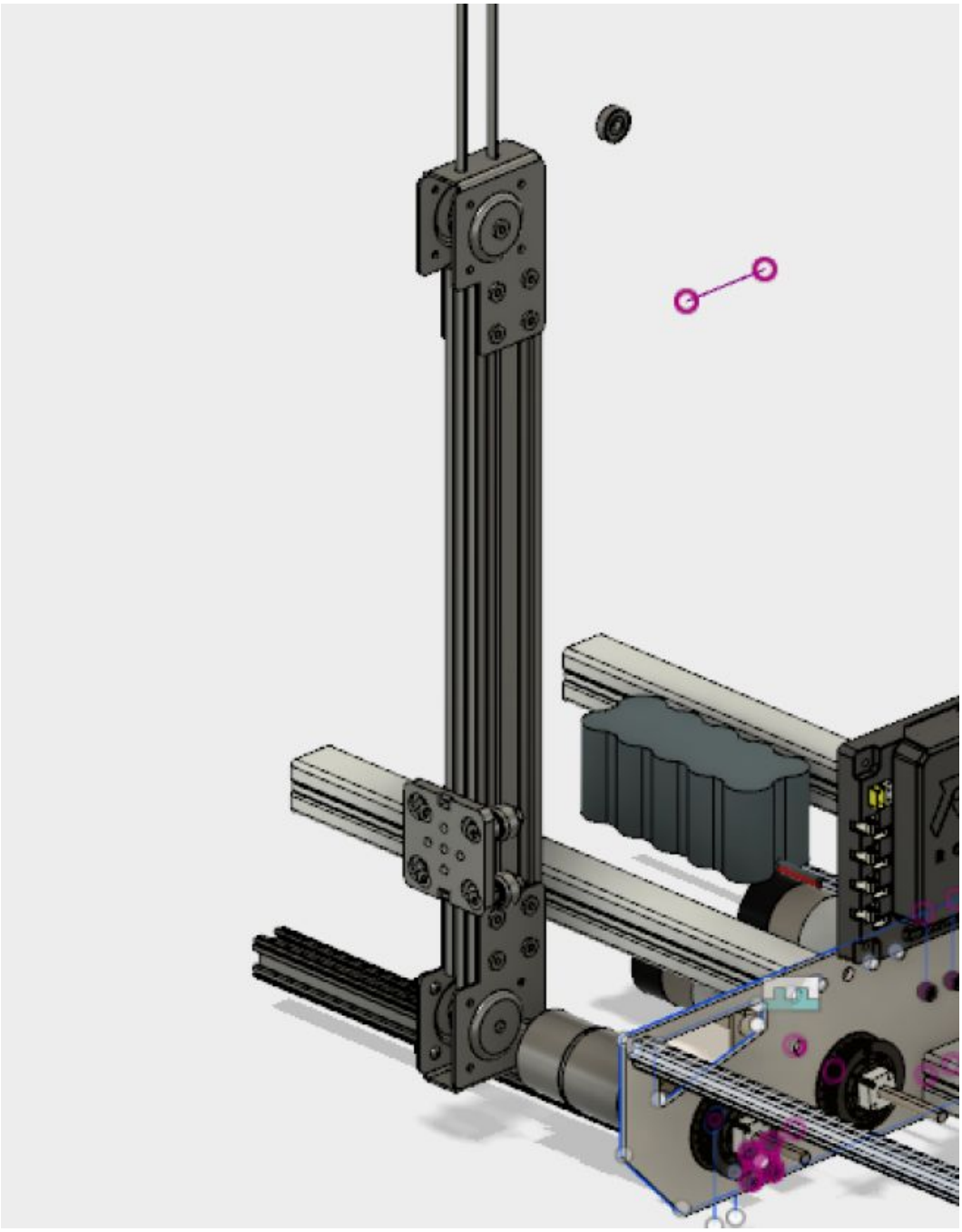


After some more brainstorming, we realized that we could use belts to transmit the power. We sketched out the design and used CAD to determine the length of belts needed.

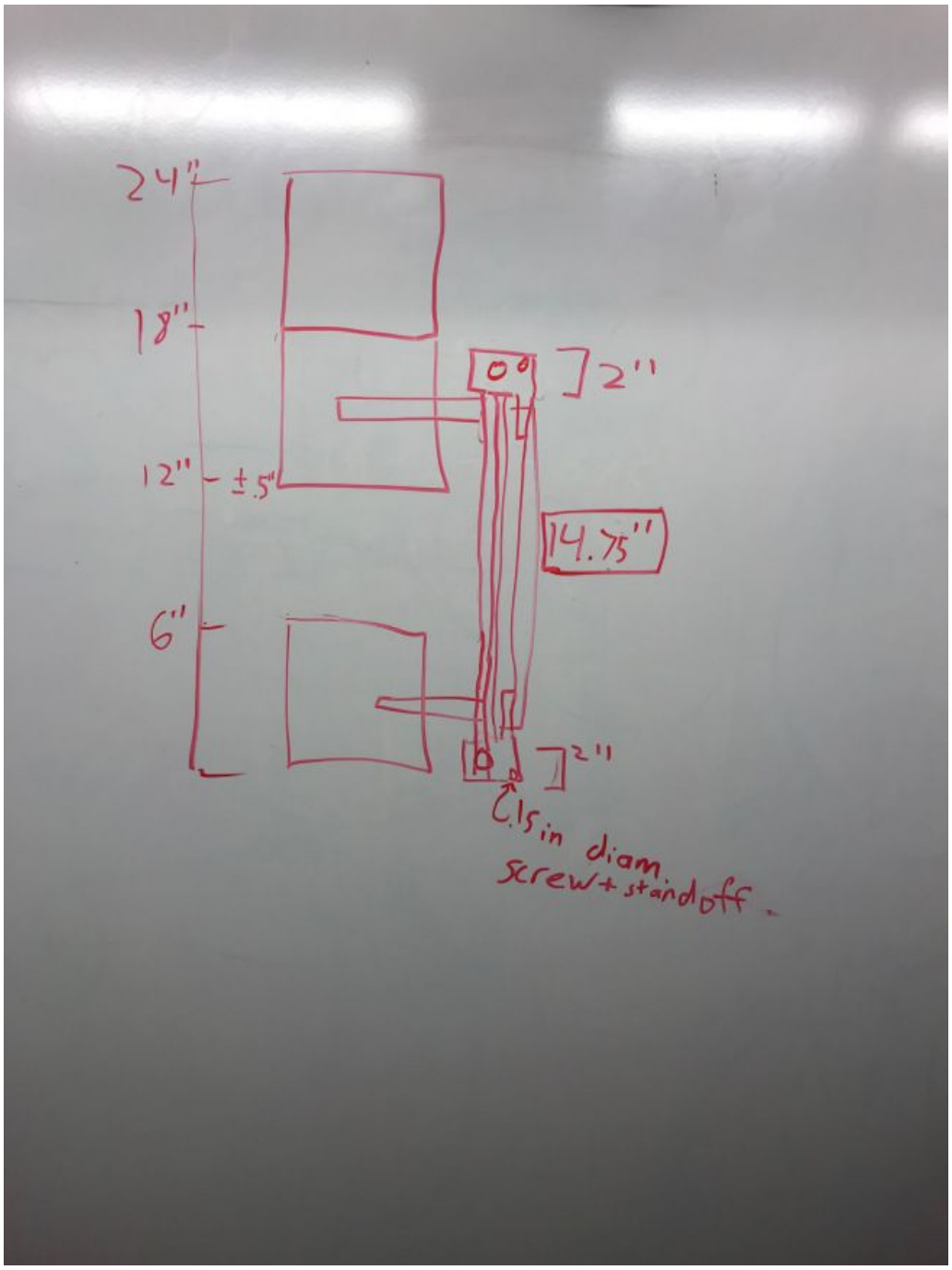


10/9/17

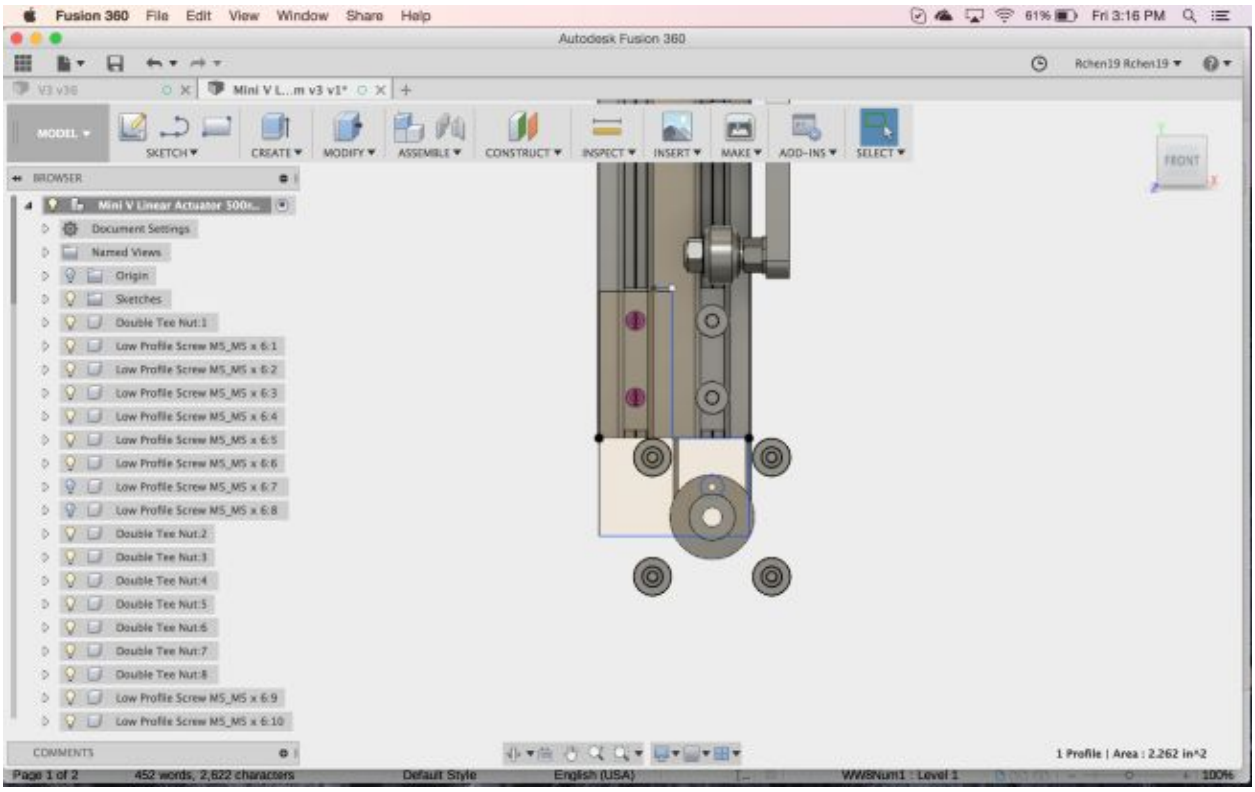
For the glyph mechanism, we began designing a linear slide rail lift powered by a motor. This way we would only need one bogie to roll on the x-rail. We designed the mechanism in CAD.



However, we found out that the “cart” can only move up and down 10.3 inches. Even if we stack a block on top of another before lifting, we still need at least 14.5 inches of travel to get to the top of the cryptobox.

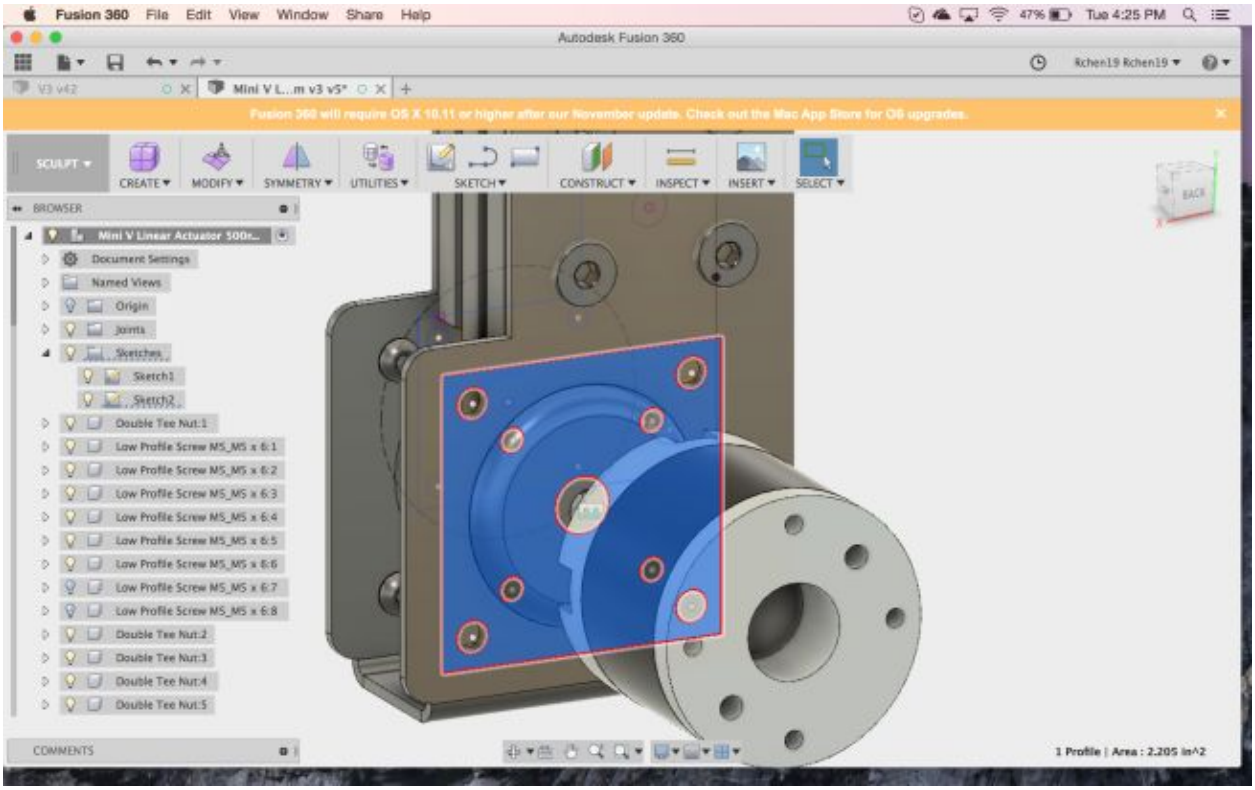


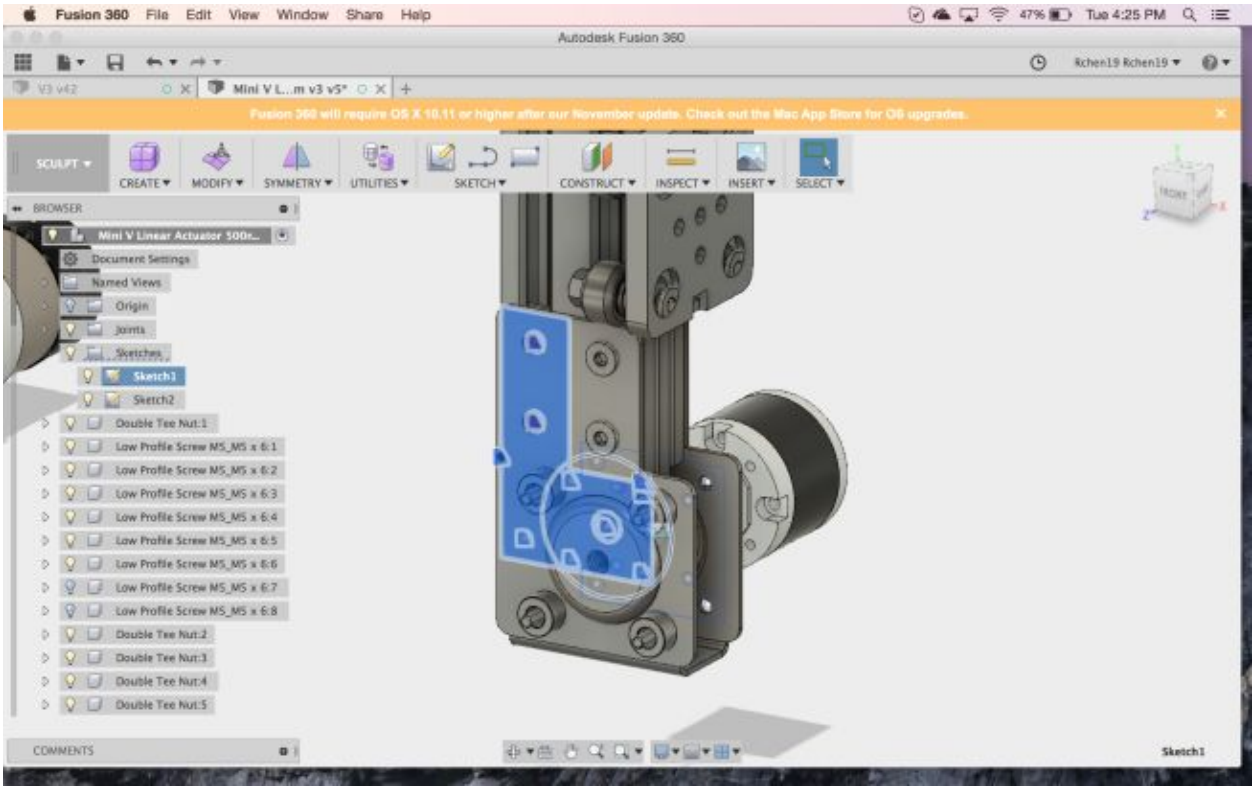
To overcome this issue, we designed custom plates to give us 14.875 inches of travel.


Fusion 360 File Edit View Window Share Help
Autodesk Fusion 360
MODEL
SKETCH
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
MAKE
ADD-INS
SELECT
BROWSER
Mini V Linear Actuator 500...
Document Settings
Named Views
Origin
Sketches
Double Tee Nut:1
Low Profile Screw M5_MS x 6:1
Low Profile Screw M5_MS x 6:2
Low Profile Screw M5_MS x 6:3
Low Profile Screw M5_MS x 6:4
Low Profile Screw M5_MS x 6:5
Low Profile Screw M5_MS x 6:6
Low Profile Screw M5_MS x 6:7
Low Profile Screw M5_MS x 6:8
Double Tee Nut:2
Double Tee Nut:3
Double Tee Nut:4
Double Tee Nut:5
Double Tee Nut:6
Double Tee Nut:7
Double Tee Nut:8
Low Profile Screw M5_MS x 6:9
Low Profile Screw M5_MS x 6:10
COMMENTS
FRONT
1 Profile | Area : 2.262 in^2
Page 1 of 2
452 words, 2,622 characters
Default Style
English (USA)

10/16/17

We wanted to test the lift that came in today. Because the lift was designed for a camera slider, the mounting hole patterns were for standard stepper motors. Thus, we had to create custom mounting plates.





We found that this lift design does not reach the top of the cryptobox. We want a lift that reaches the top level of the cryptobox without stacking cubes due to the difficulty and time consumption of lining up. Our linear slider does not meet reach this height. Even if we tilted the lift to the diagonal of the 18 inch cubed limit, we still wouldn't reach the required height, not to mention the difficulty in creating such a pivot point. We came up with some other ideas for the lift:

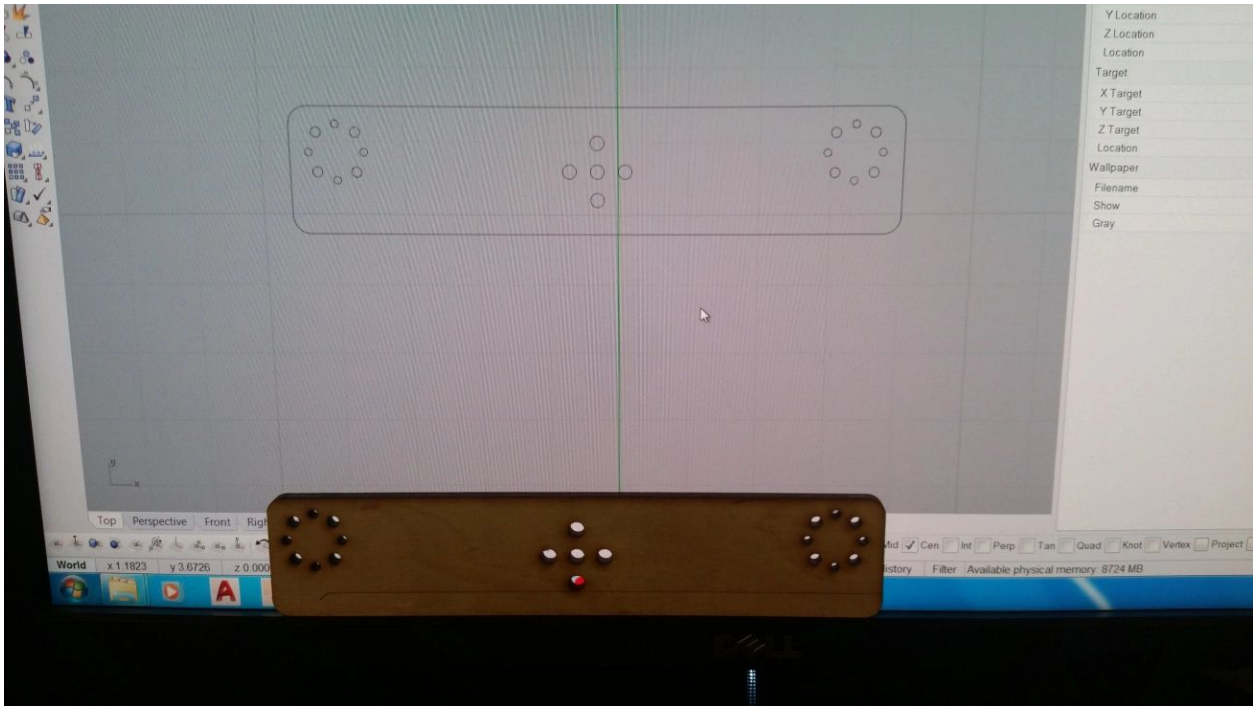
- Put a linear slider on the linear slider
    - Pros
        - The linear sliders are extremely stable individually
        - Reaches the required height (barely)
    - Cons

        - Requires two motors
        - The additional weight of the second linear slider decreases the stability of the lift
        - Possibility of belts slipping
        - Complicated to program
- Arm
    - Pros
        - Easy to achieve the required height
        - Not a complicated design
    - Cons
        - Previous tests show questionable stability
        - Takes up a large amount of space
        - The distance from the grip to the cryptobox changes as the arm moves up and down
- Use the lift on last year's robot
    - Pros
        - Proven system
        - Easily reaches the required height, can go above the required height
    - Cons

- Has a tendency to break
- Up and down motion is not smooth

10/23/17

We remade the gripper plate so that the grippers actually grip the cubes properly.

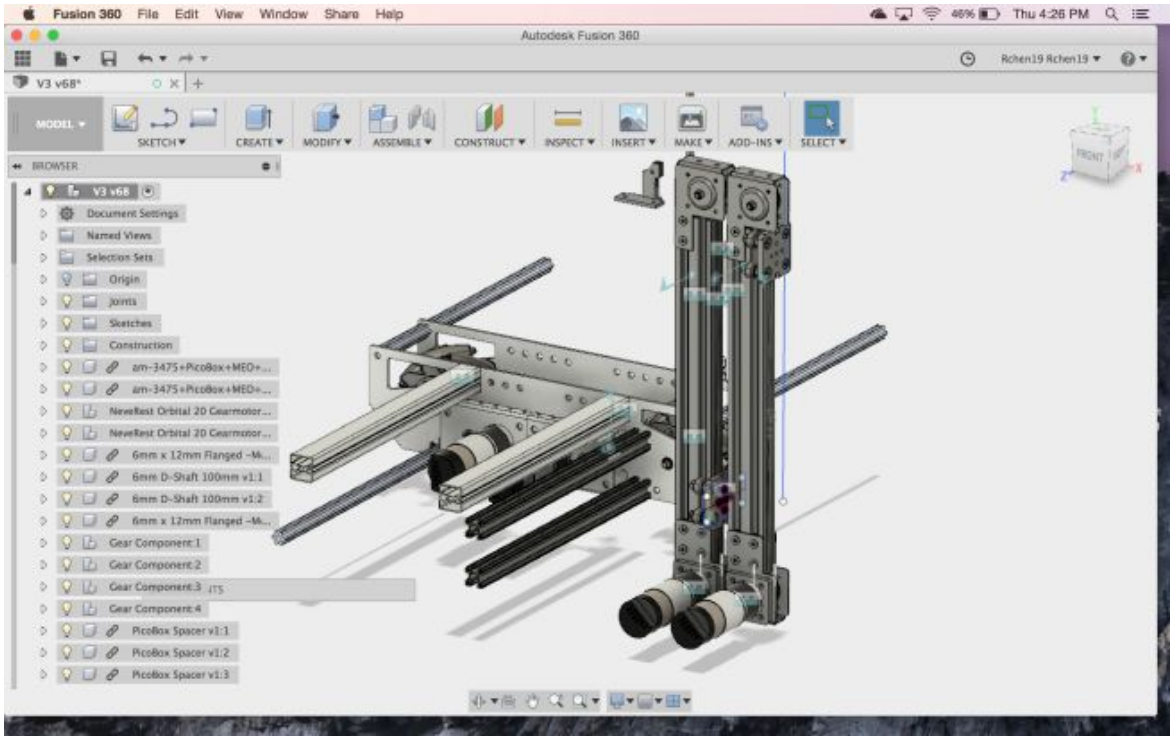


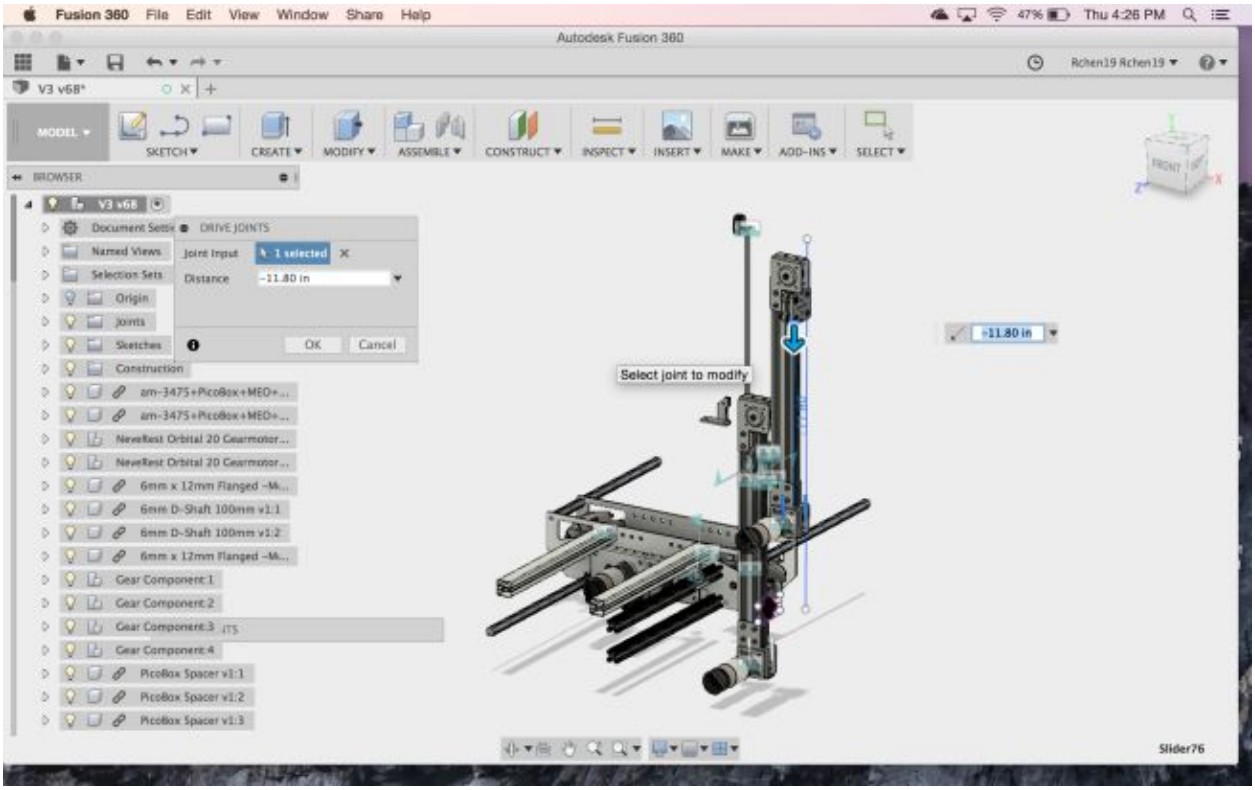
We decided that we wanted the lift to be able to reach the fourth row of the cryptobox because we discovered that stacking two blocks on top of another was time-consuming. To reach the fourth level, we needed a lift with a travel distance of 19 inches (6*3 +/- 1).

We decided to use the lift attached to the lift approach. After tightening the belts and performing some load tests, we determined that the lift can handle the weight of another lift attached to it. We extended the travel distance of the lift by cutting the plates that held the pulleys for the belt by 1.5 inches, giving us 19.5 inches of travel.

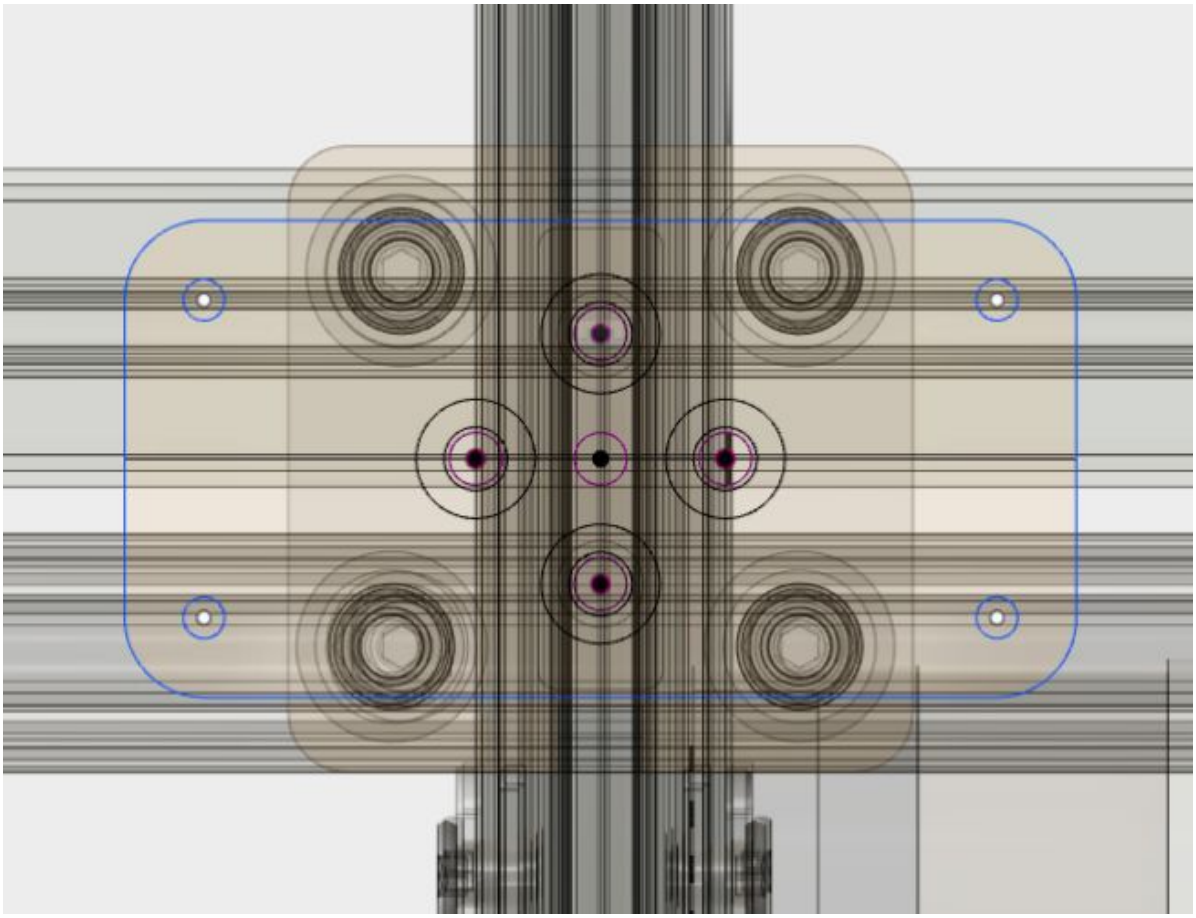
We disassembled the lift, cut the plates down, reduced the height of the lift to fit within the 18-inch box, and reassembled the lift.

We designed the double lift in CAD.





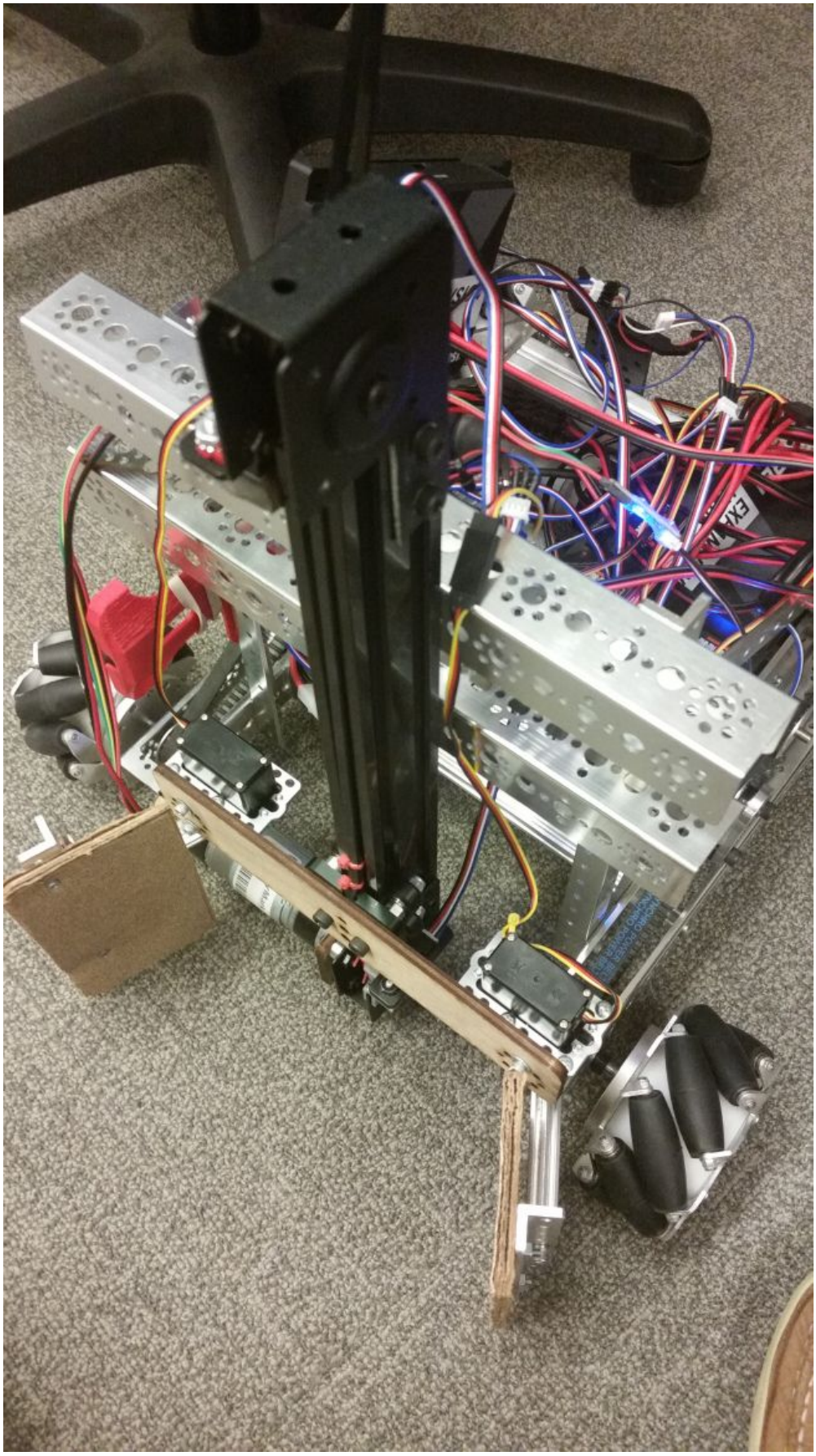
Using slider joints, we can see how high the lift extends accurately. Everything seemed to work out, so we decided to buy the second linear slider.



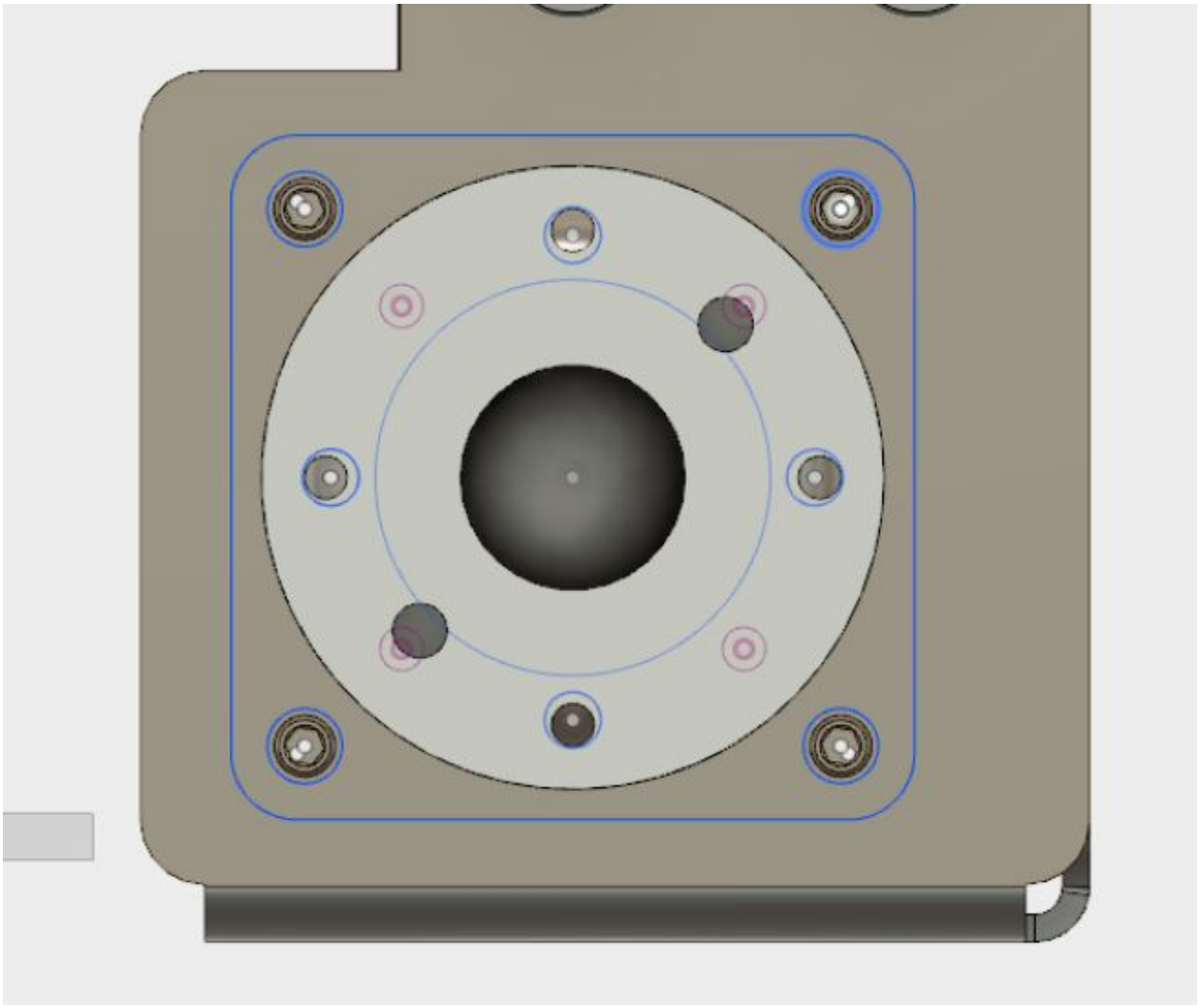
We made plates to connect the two lifts together.

10/30/17

We found out that although we cut the lift down to 17.5 inches, we had to raise the top plate and the pulley attached to it to tighten the belt, pushing us outside of the 18 inch box. We had to disassemble and cut the x-rail down again.



It now fits in the 18” box.



We made a new motor plate for the second lift because the holes on the original CAD model were not the correct size.

We also started to build the second lift.

After finishing building the lift, we identified a few problems:

- The belt needs to be tighter, keeps slipping.

- The holes in the plate connecting the lifts together are too large, resulting in a wobbly second linear slider.
- The wooden plate is also bowing in the center.

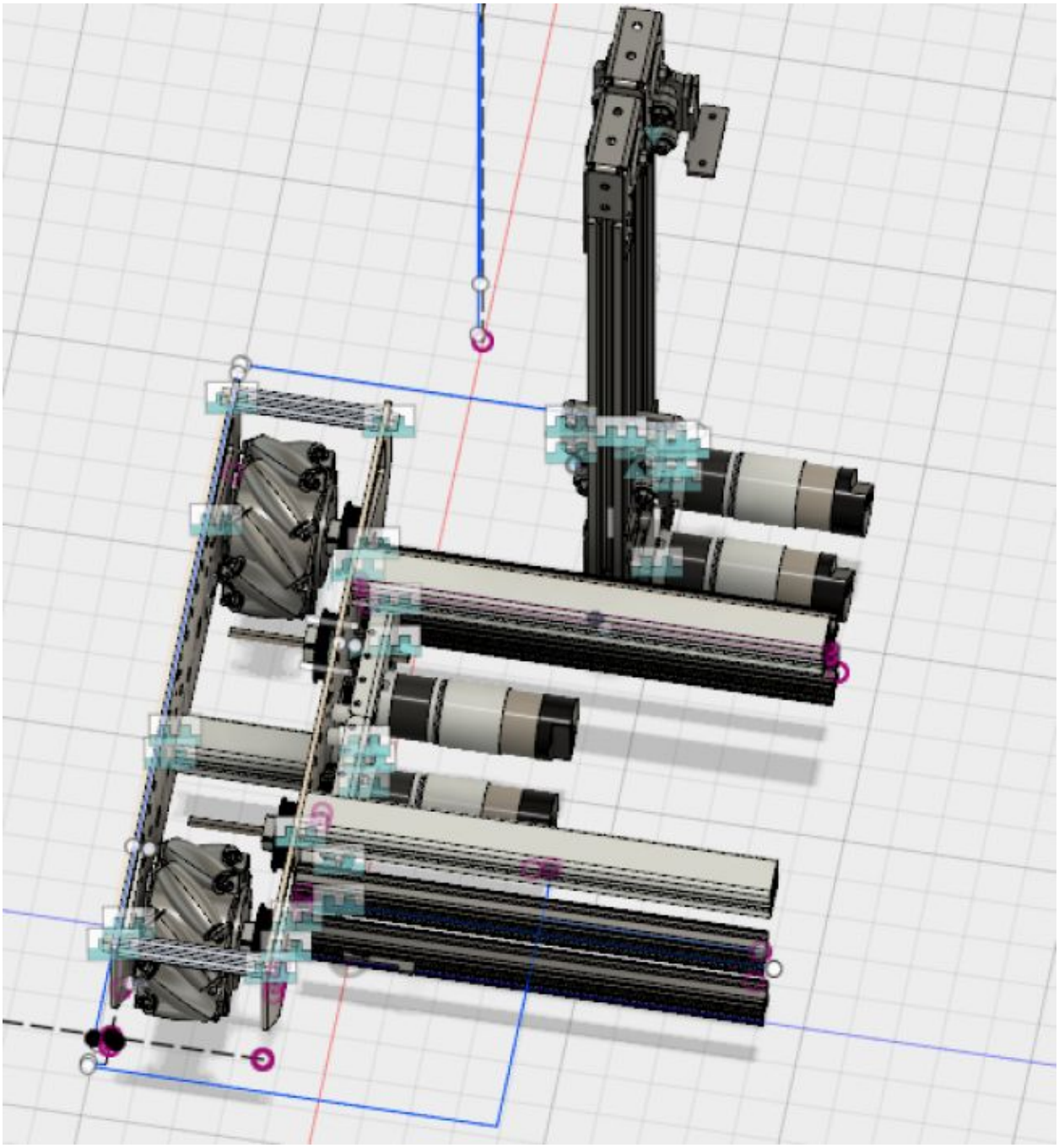
We designed a new plate with tighter tolerances and less distance between the holes to reduce bowing.

We discovered that if we mirrored our chassis (currently only have one side modeled), the motors would be overlapping. If we increased the width of our current chassis design to prevent the overlapping, our chassis would be one inch too wide.

We figured out how to keep everything in the 18” by:

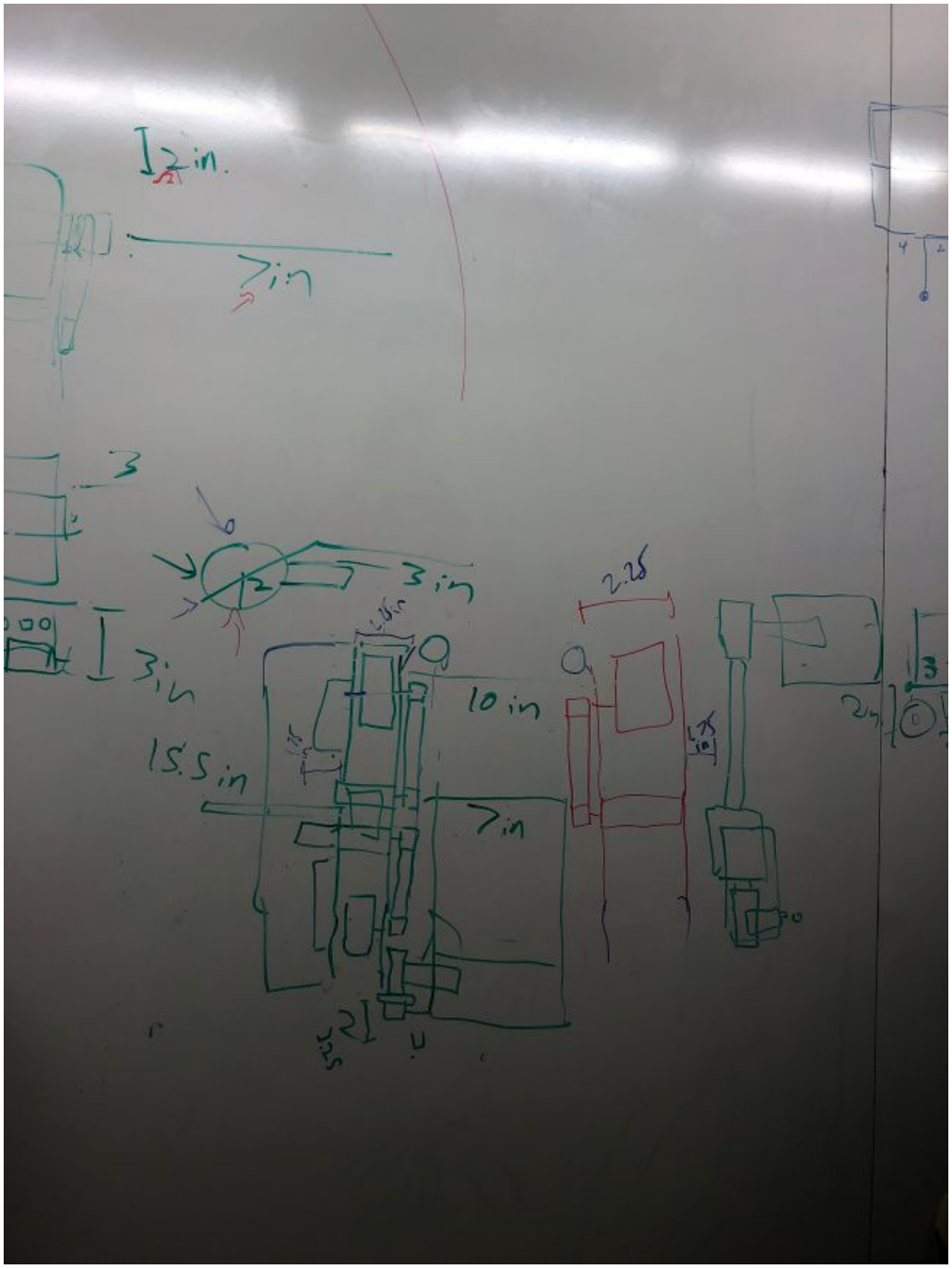
- Flipping a few bearings
- Cutting out the spacers for the gearboxes
- Cutting holes in the plates because the motors were now protruding through the plates after the spacers were cut, and
- Eliminated any spaces we could find between components.

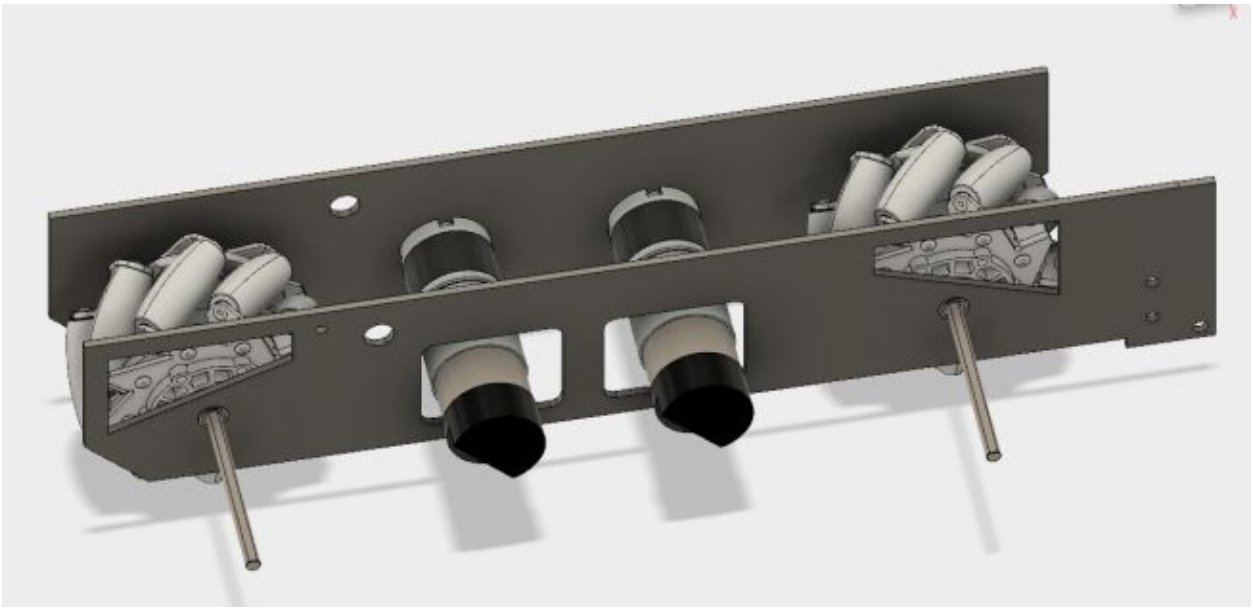
This gives us .122 inches on either side of the chassis, and .1 inches between the motors once we mirror the left side.

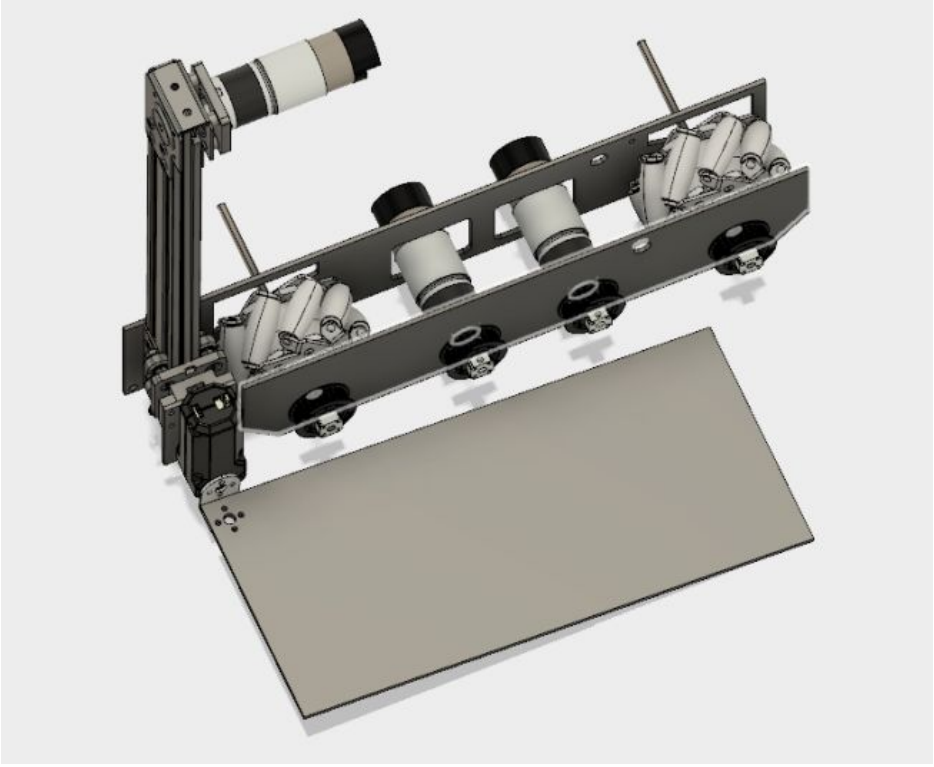


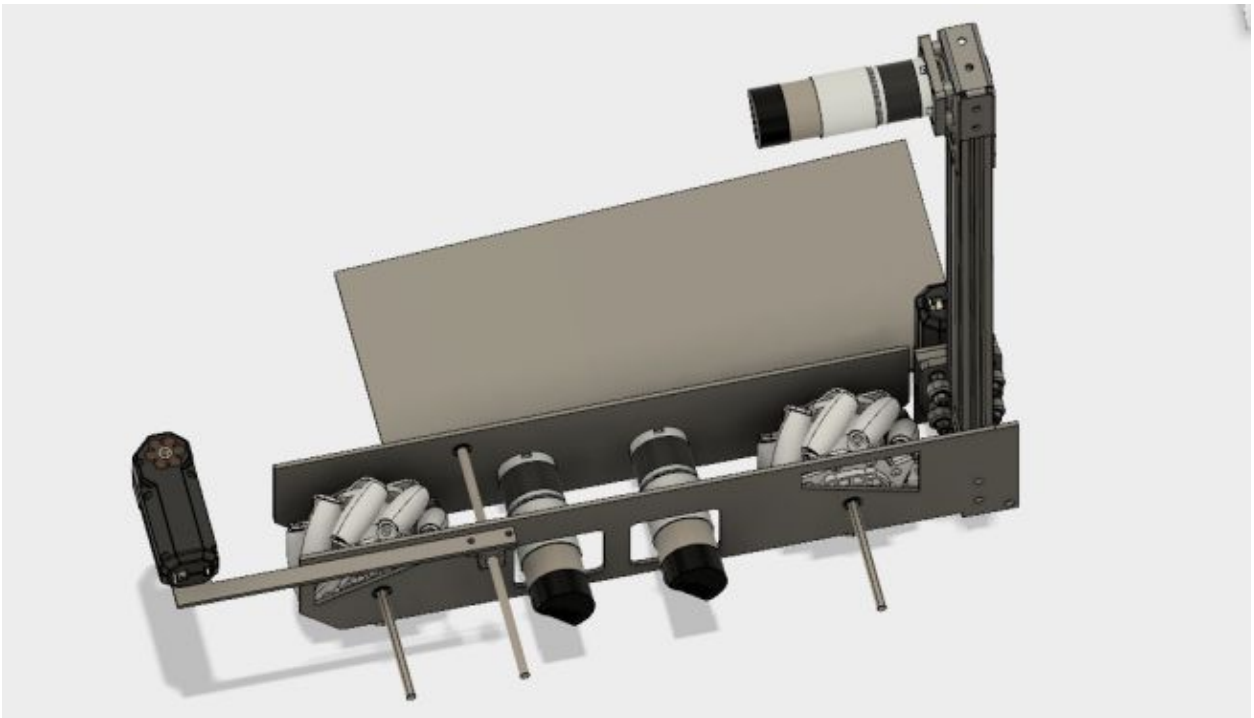
11/6/17

Over the weekend we saw a revolutionary design by NC Gears that consisted of a wheel-based harvester intake and a flipper. We decided that this would be much faster than our current design, so we began prototyping.


12 in.
7 in
3
3 in
3 in
2.25
10 in
15.5 in
7 in
2 1/4
3

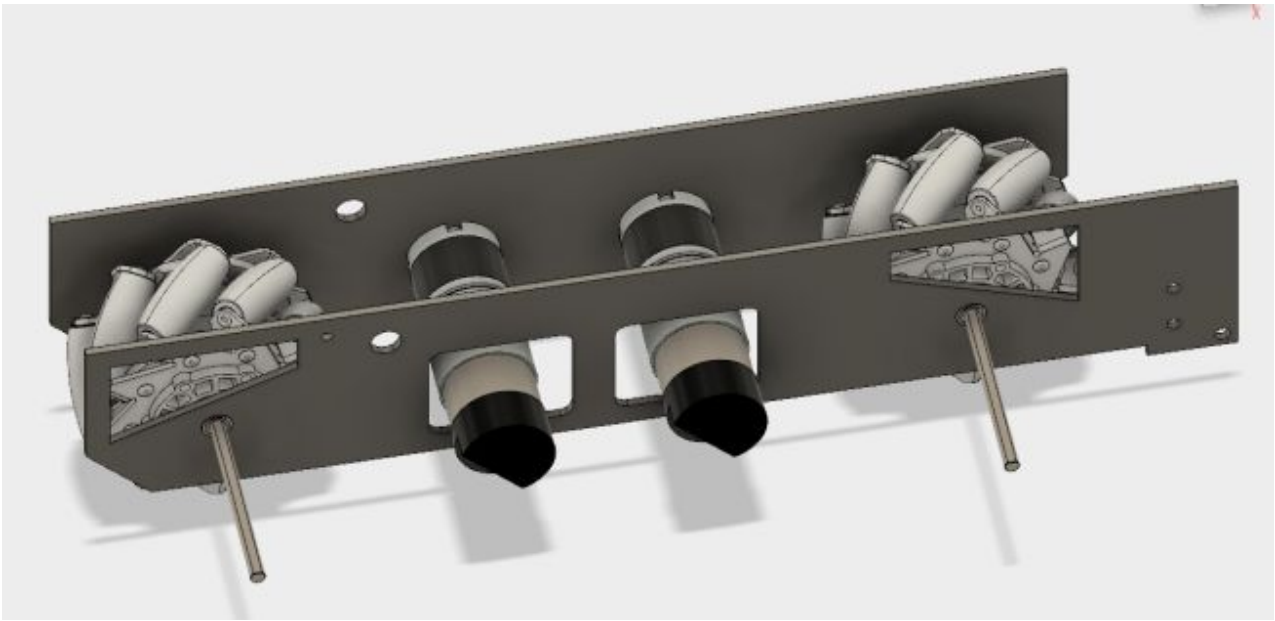




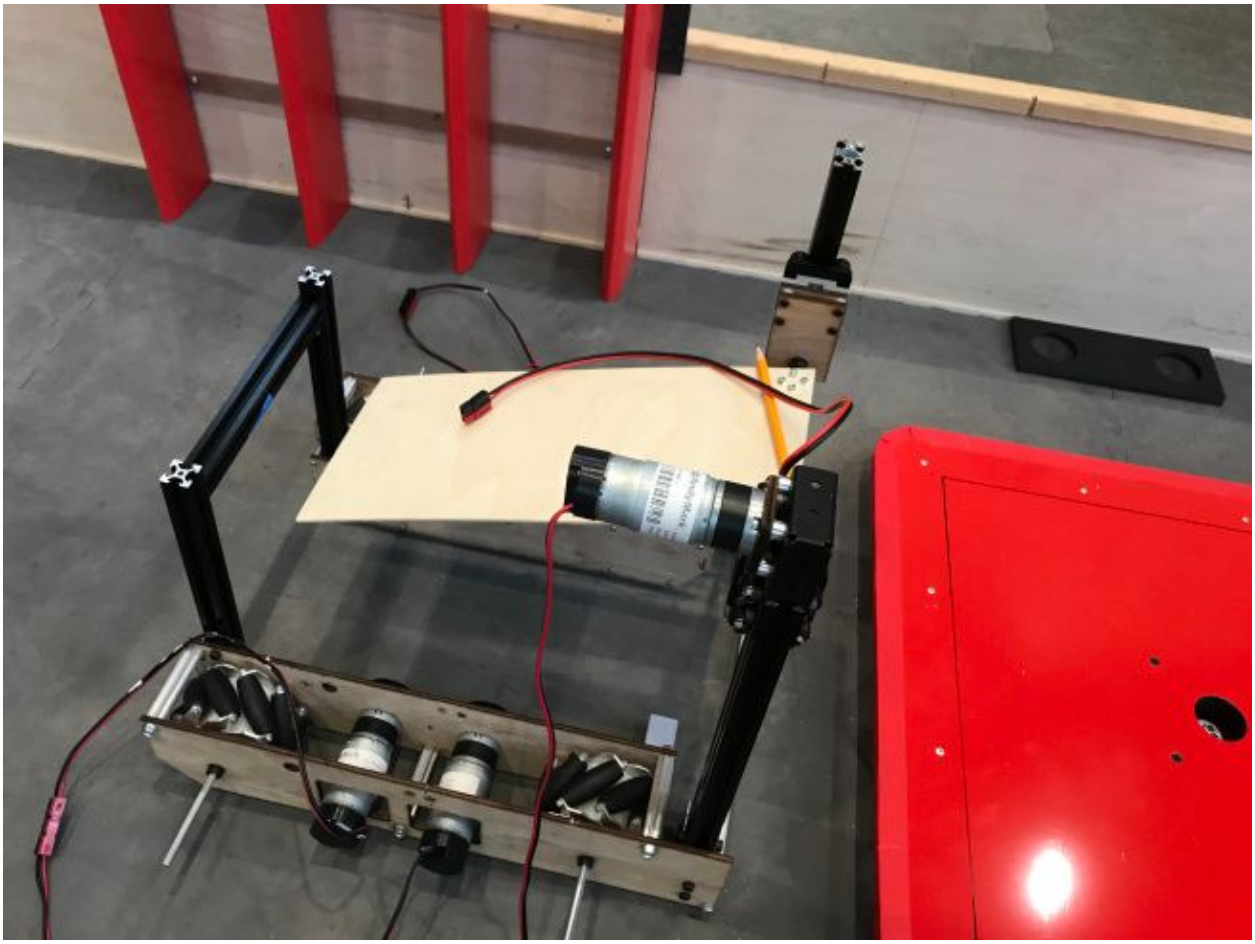


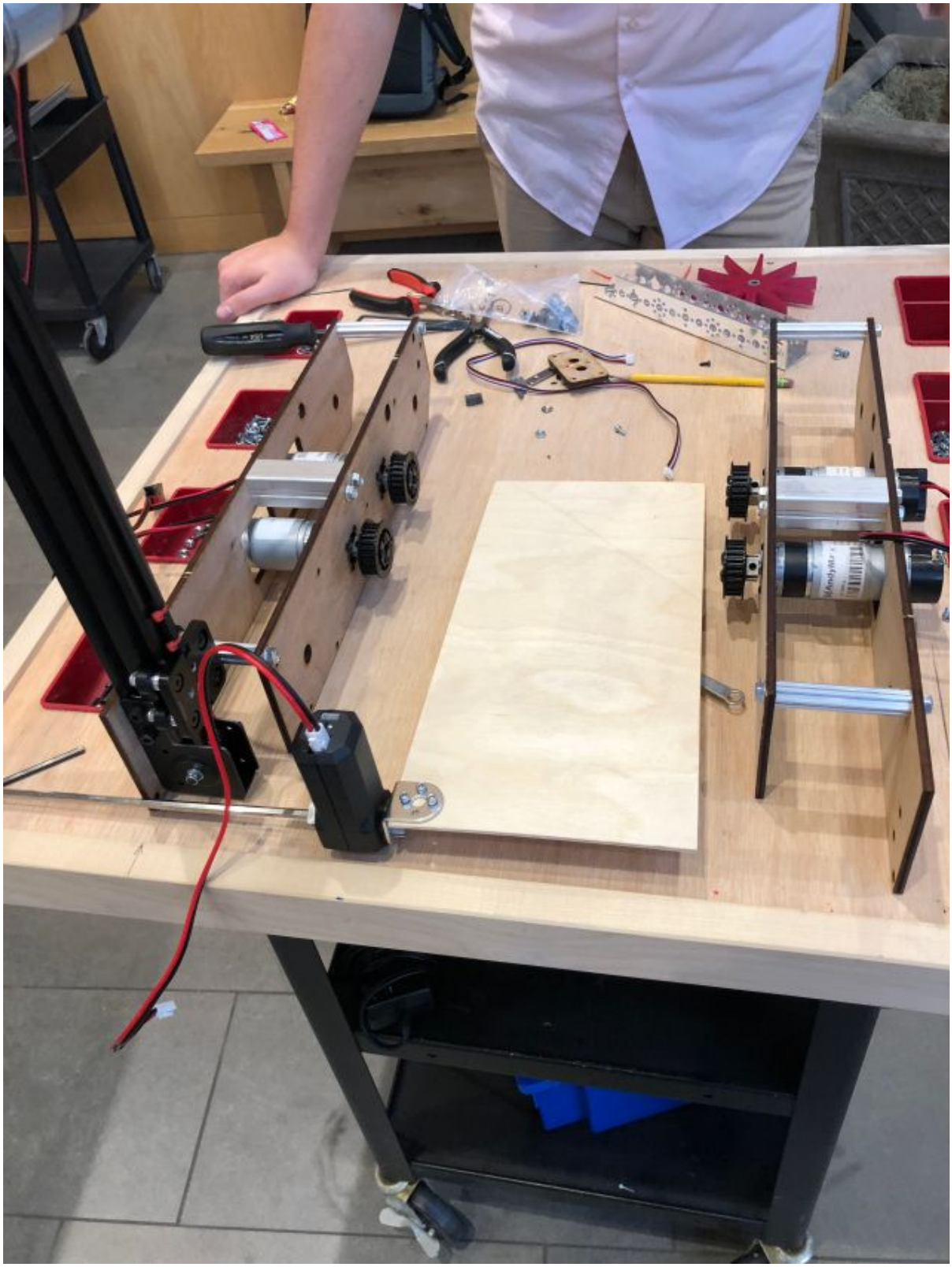
11/13/17

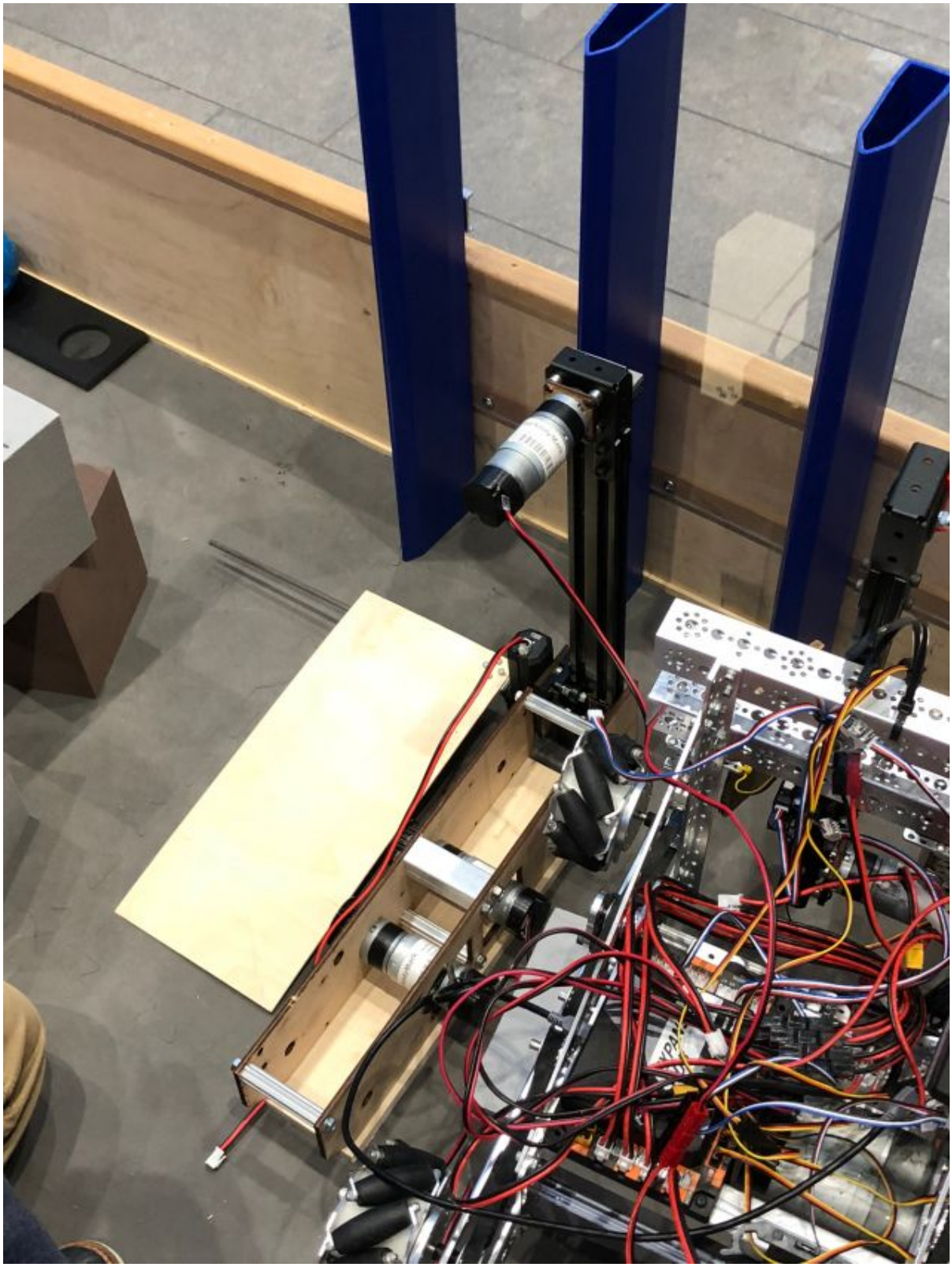
We started to design the plates for the new drivetrain this week.

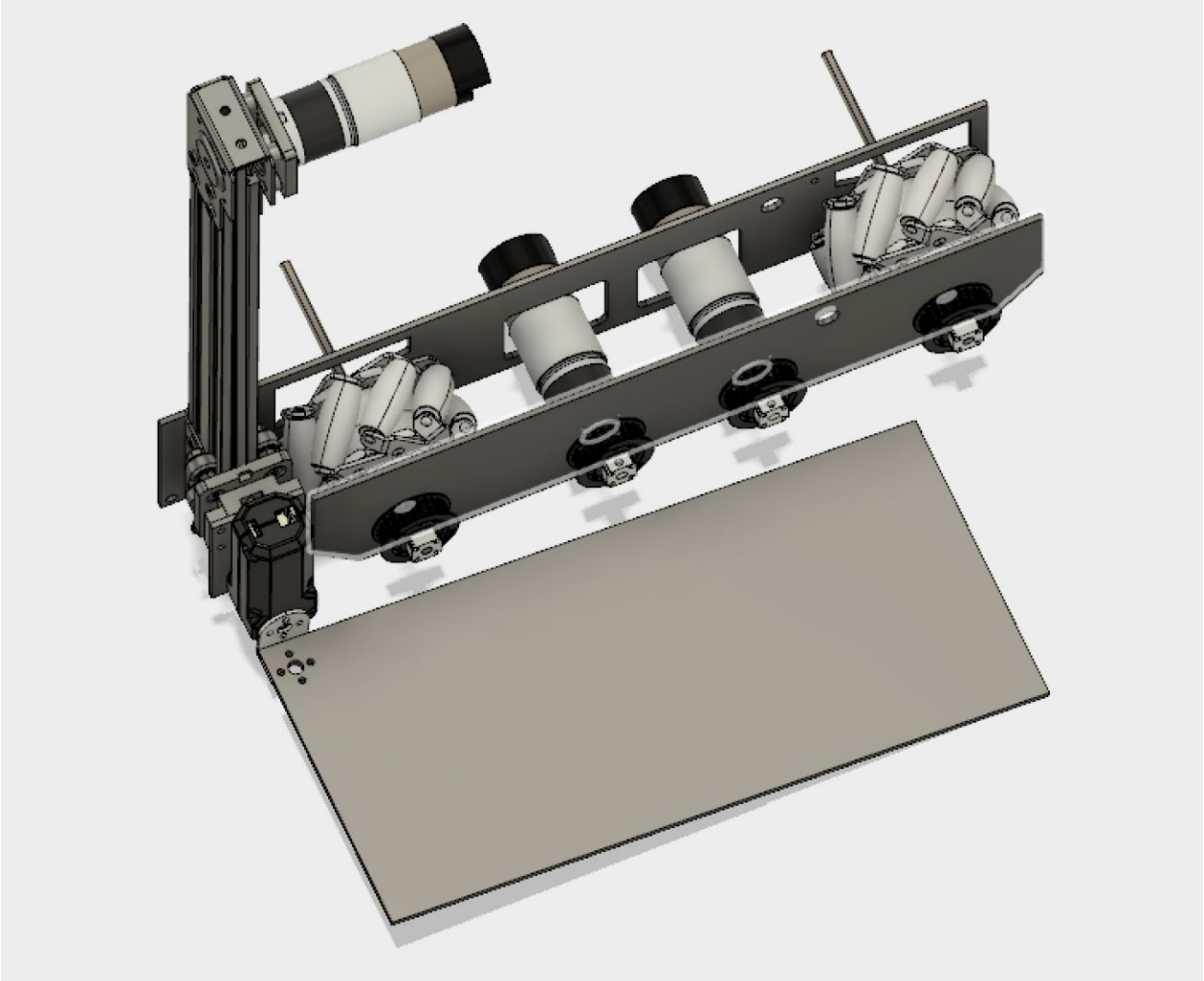


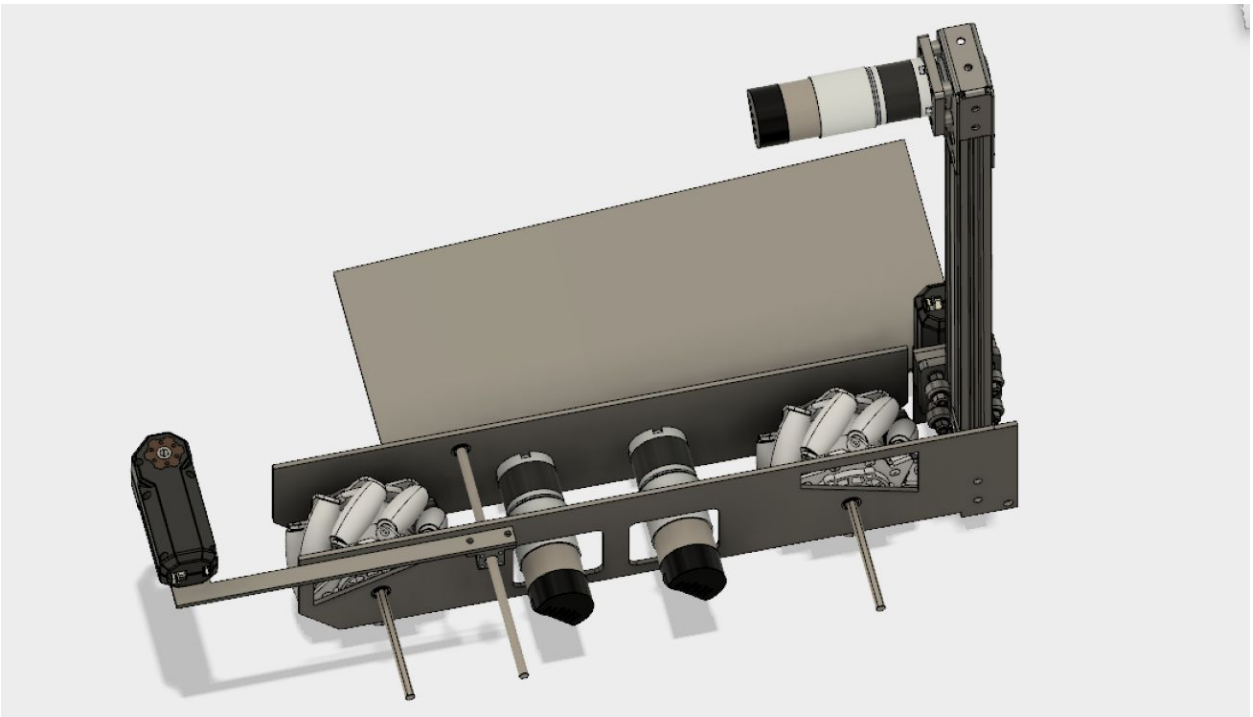
We assembled a wood prototype of the chassis. We discovered that the churros and the wheels are too low to the ground and will hit the ground when coming off of the balancing stone. We also discovered that we are going to need more supports under the flipper to help prevent shearing

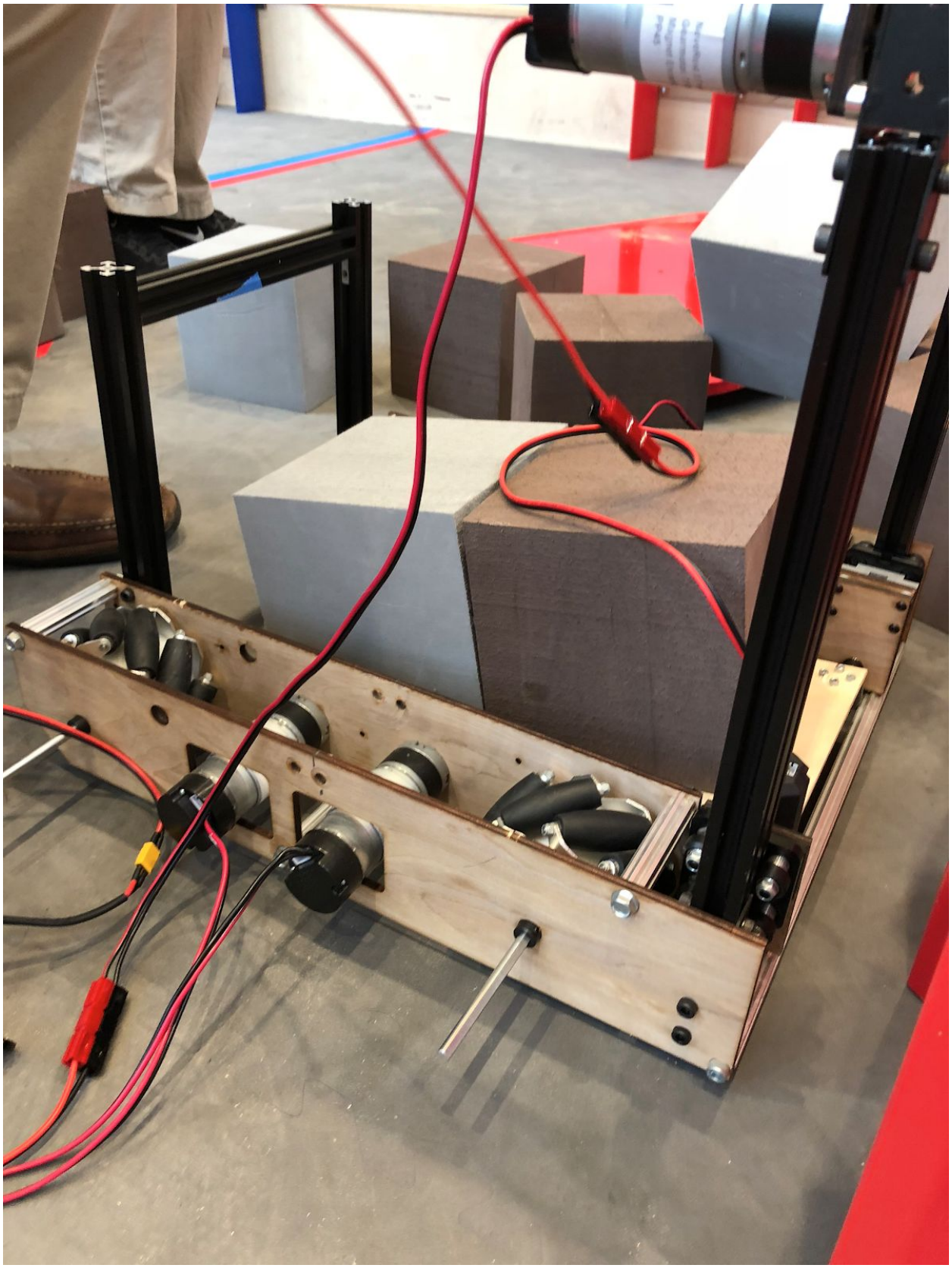




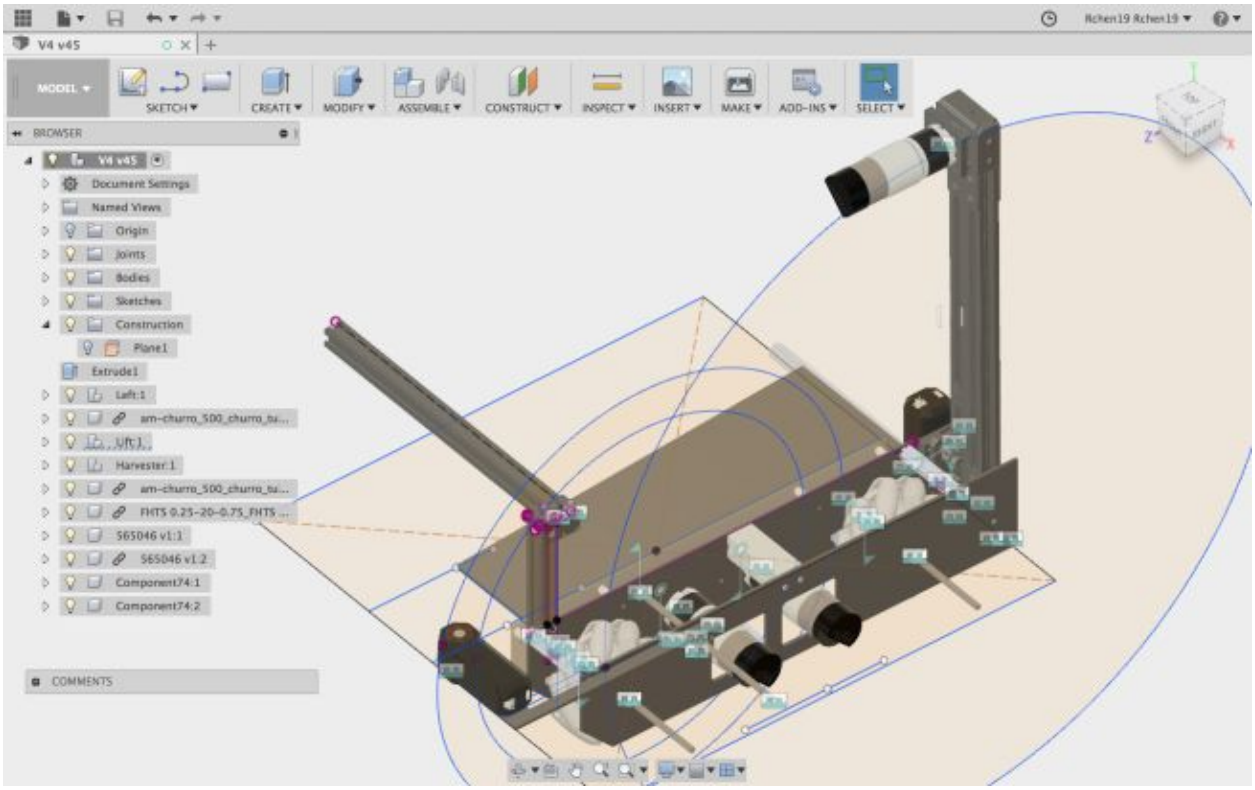




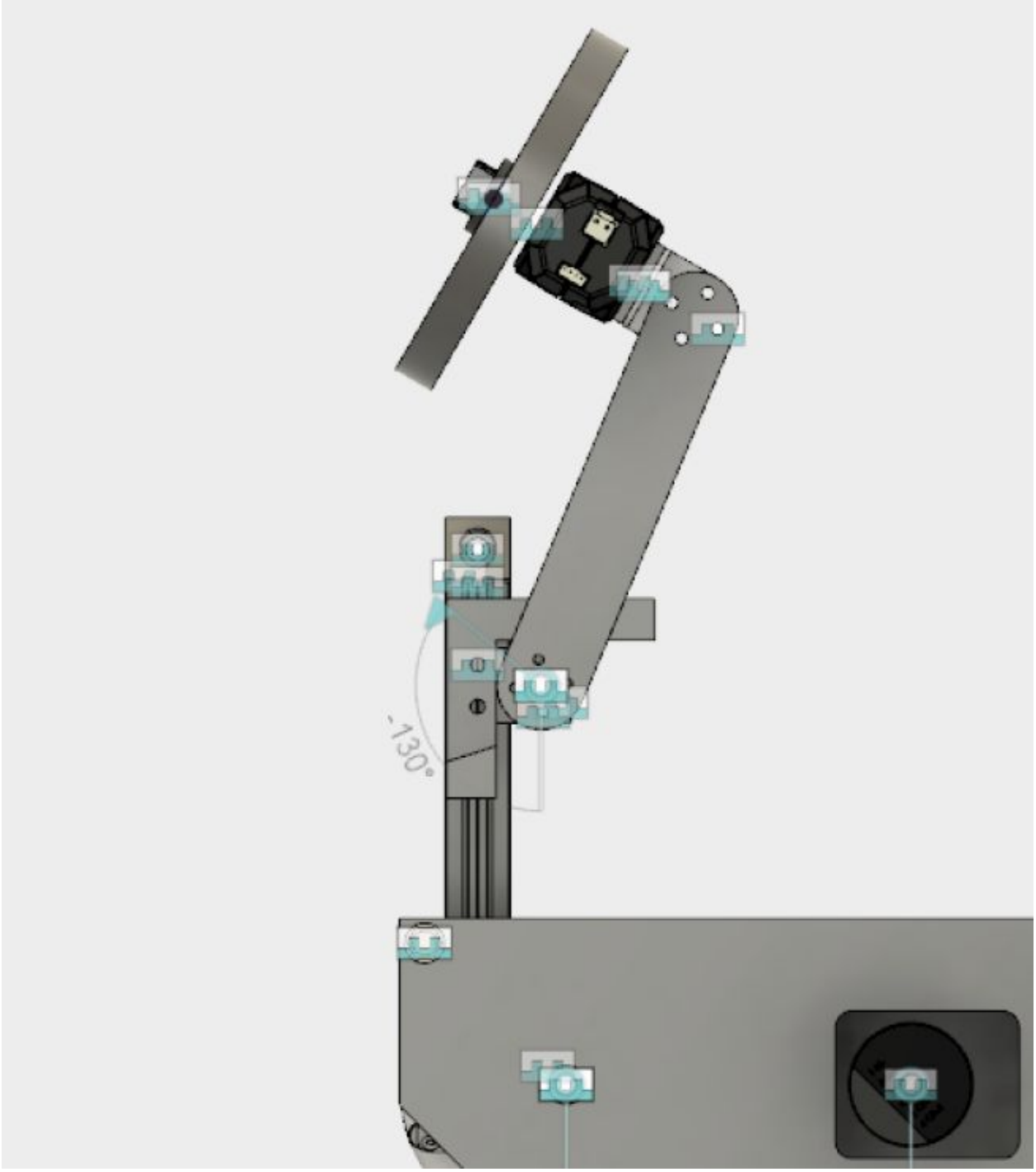




11/20/17

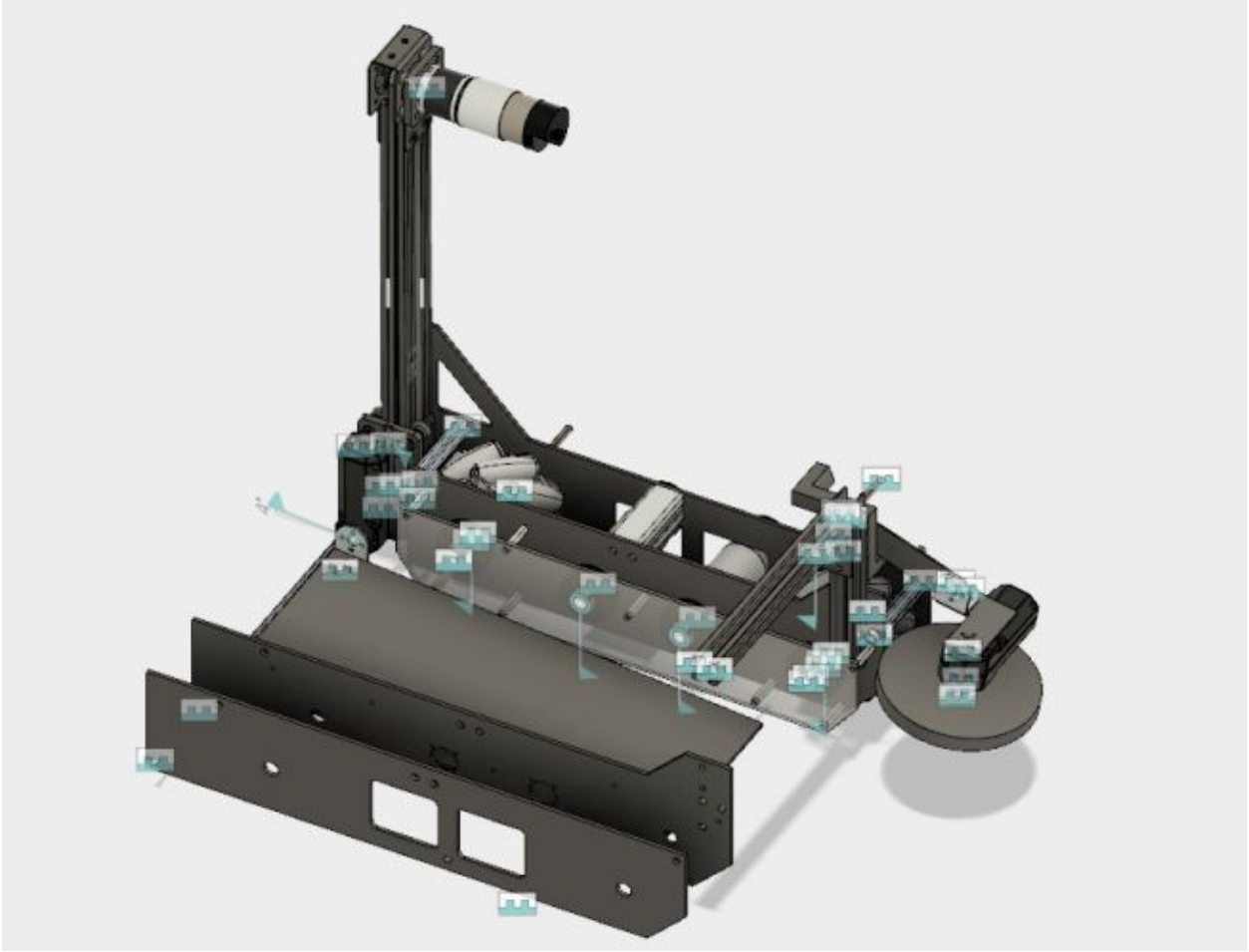


We continued designing the new design in CAD.

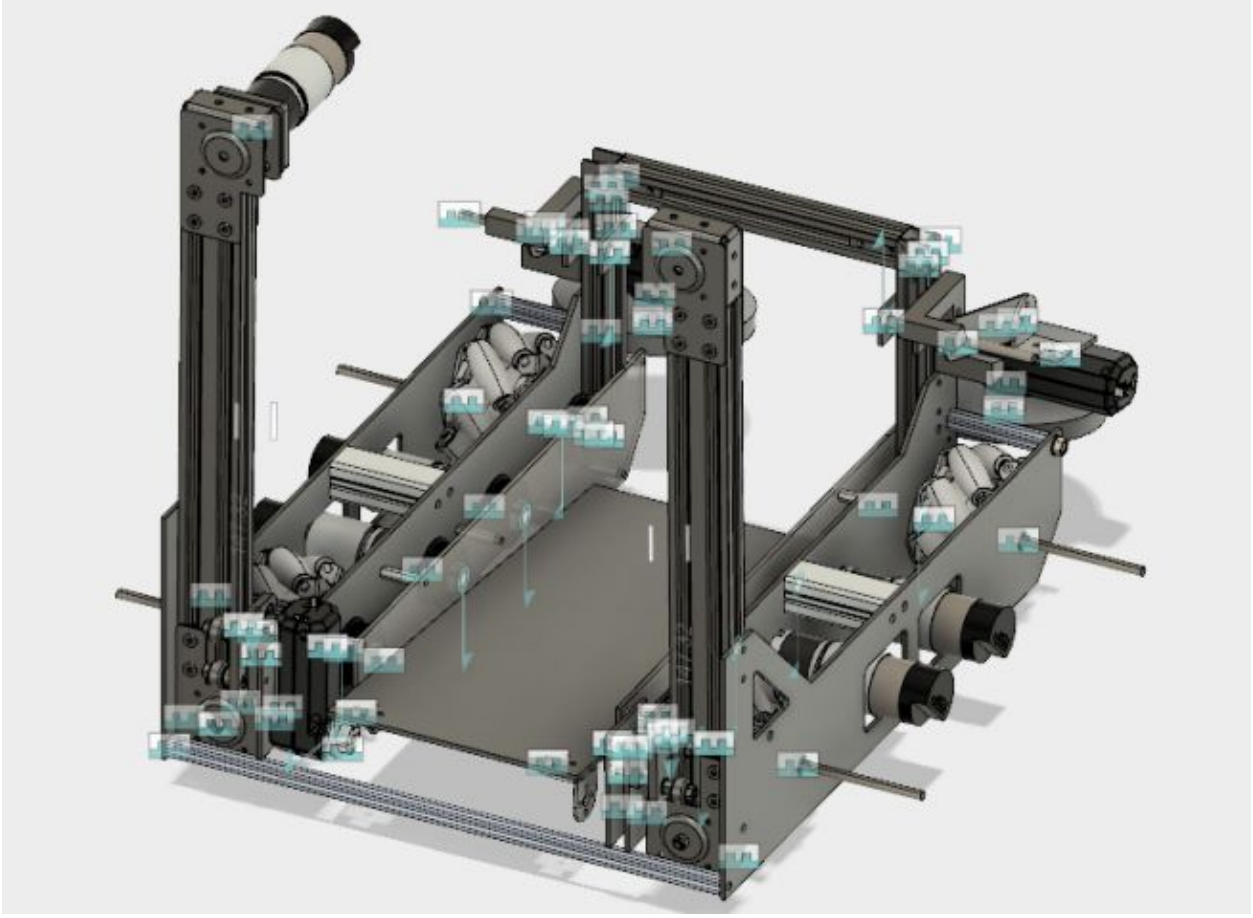


We replaced the x-rail with v-rail from Open Builds because of the better connectors Open Build provides. We also made a new harvester that is attached

to the v-rail instead of the side plate. This allows the wheel to be close to the robot but still clear the v-rail when it swings down.

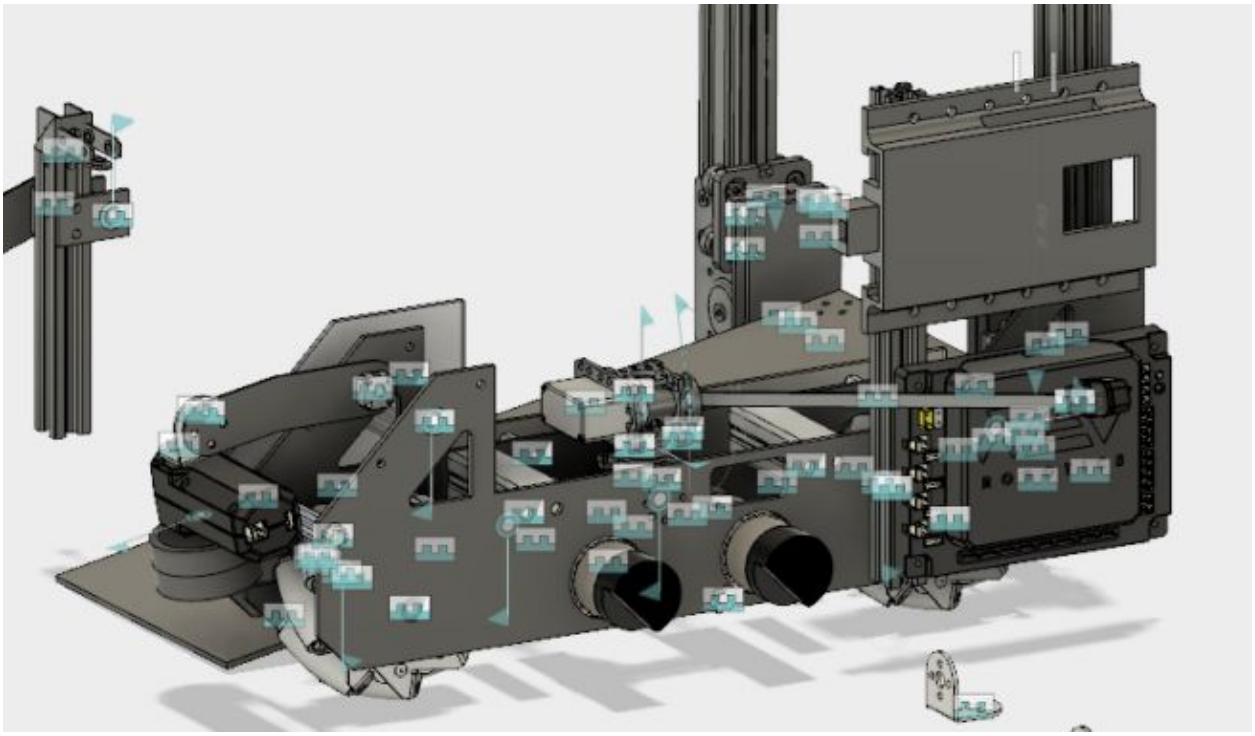


We improved mounting points on the harvester and plates. Modified the plates to provide better support for the lift and V-rail. Cut down the V-rail and churro. Made the right-side plates.



We finally mirrored the two sides of the design.

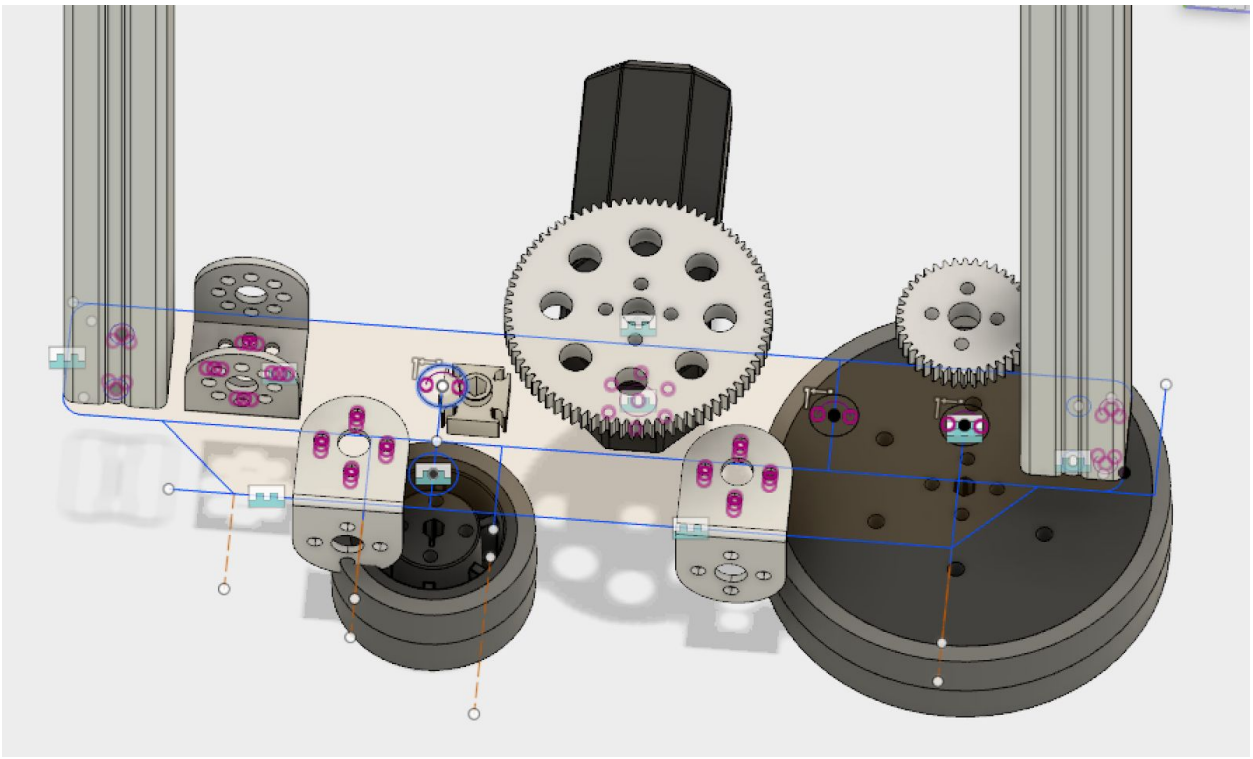
After finding additional mounting points below the paddle for the peanut extrusions, we decided that the v-rails were not needed. This meant a redesign of the harvester, again. With our wooden prototype, we also discovered that the plates often got stuck on the balancing stones. To remedy this, we increased the clearance from .5 inches to an inch. We also added mounting holes for the ball knocker.

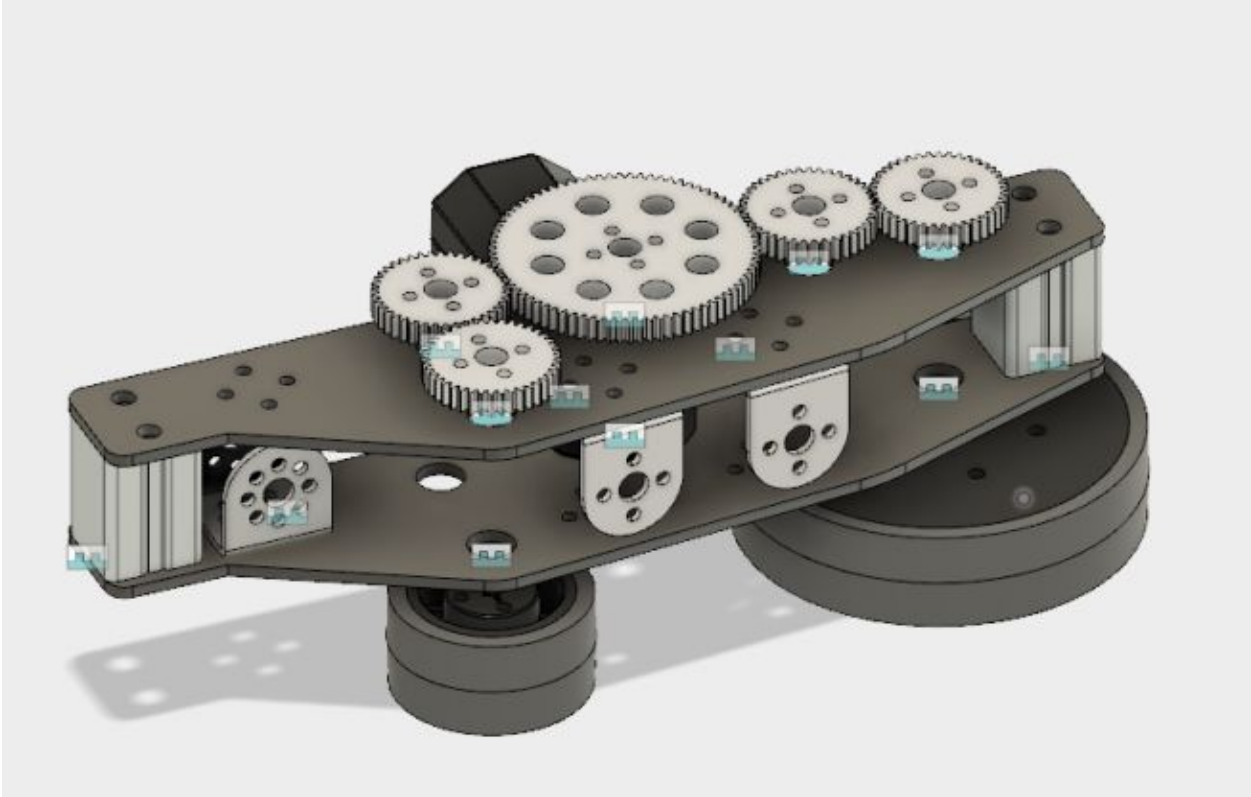


We decided that only using one pair of wheels was not good enough to quickly harvest two balls. Instead, we want 2 pairs of wheels connected by a belt.

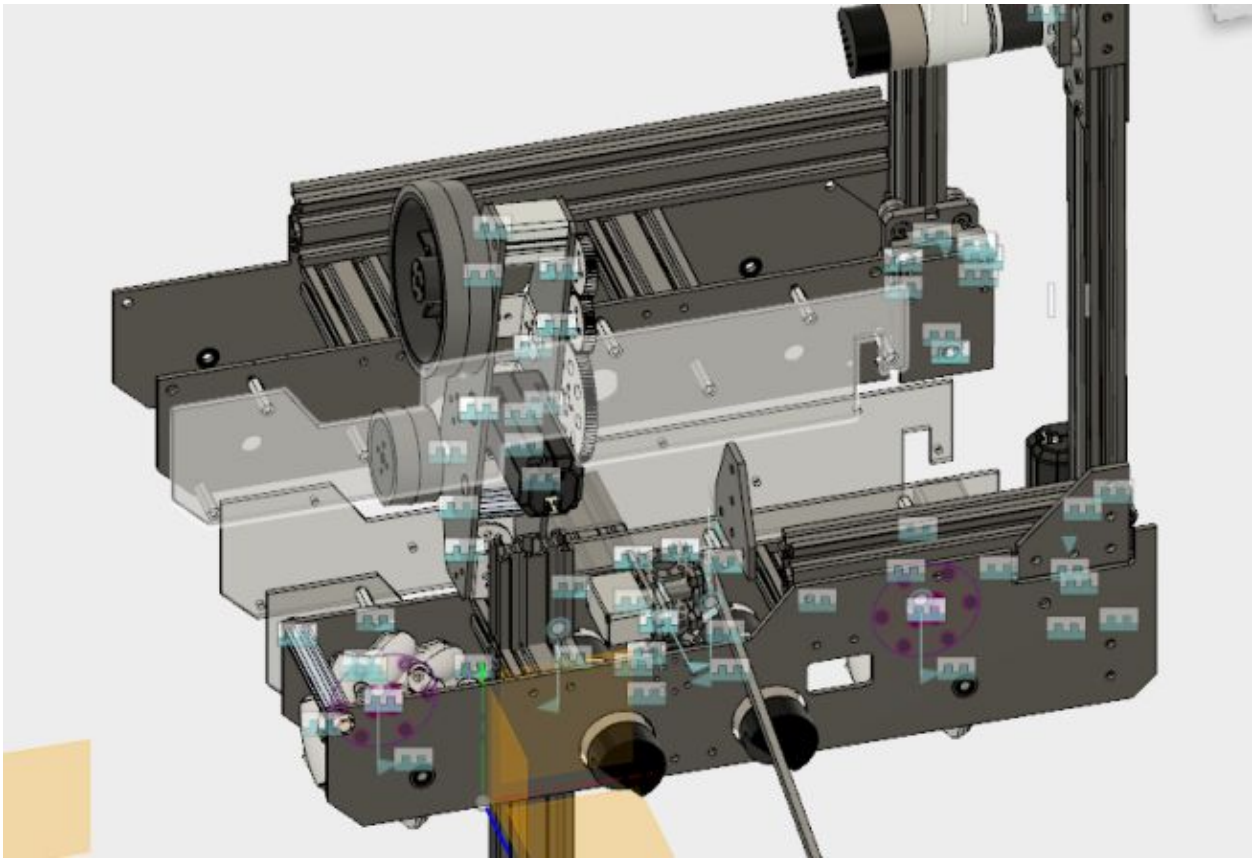
12/4/17

We began designing the intake “pods” that would drop down.

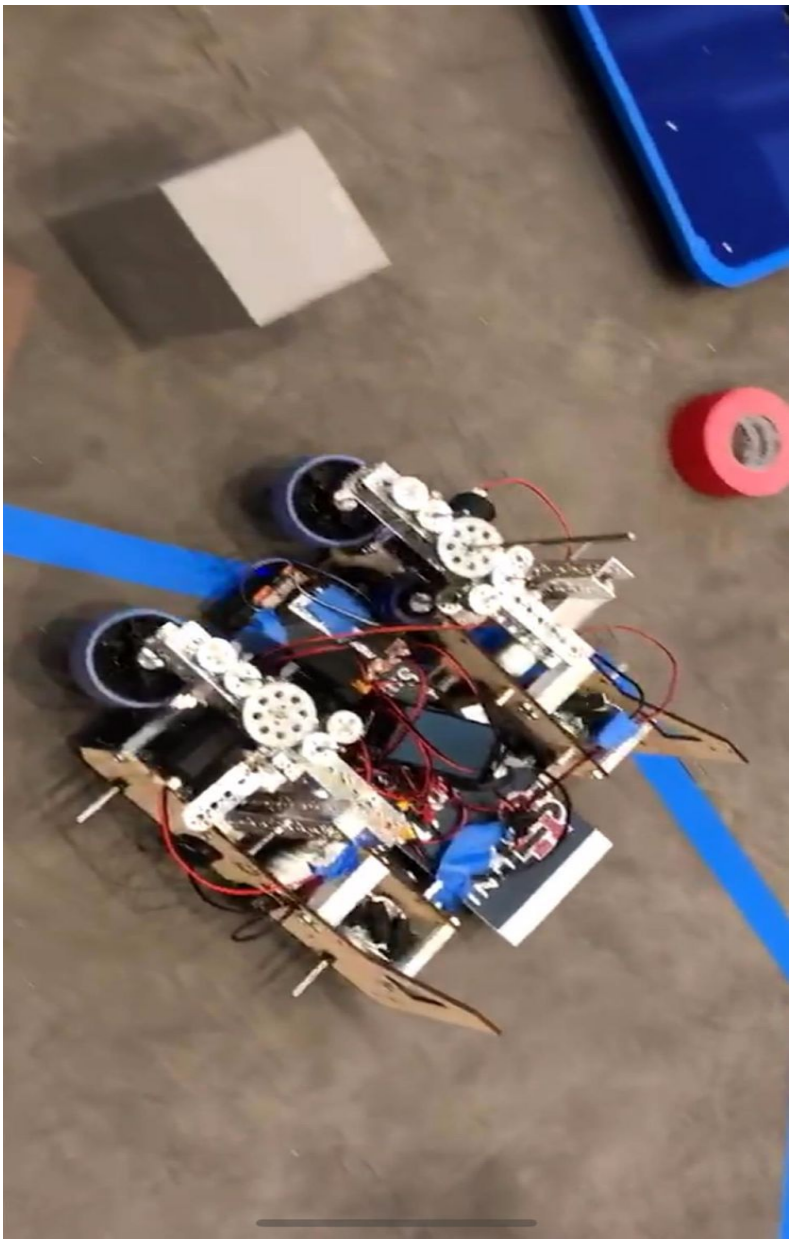




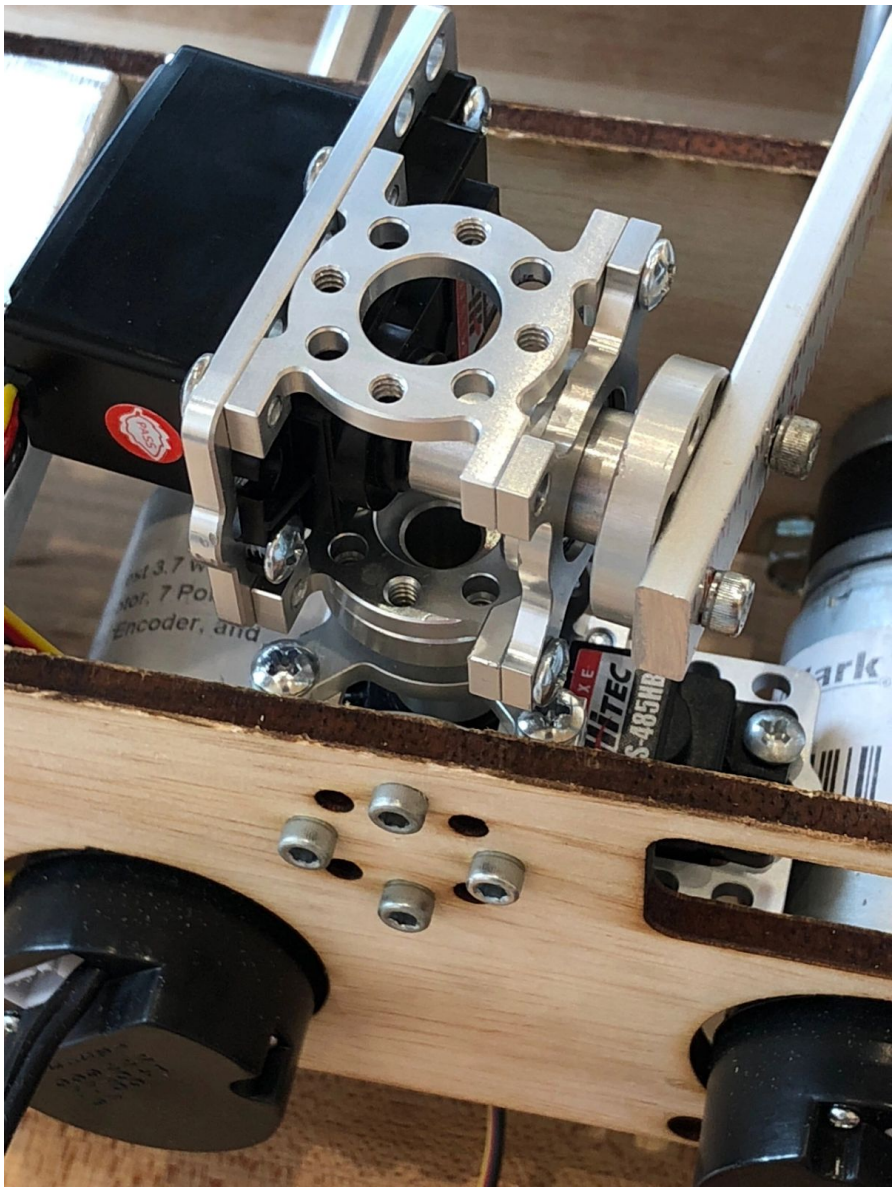
We brought our current robot together into one CAD file.



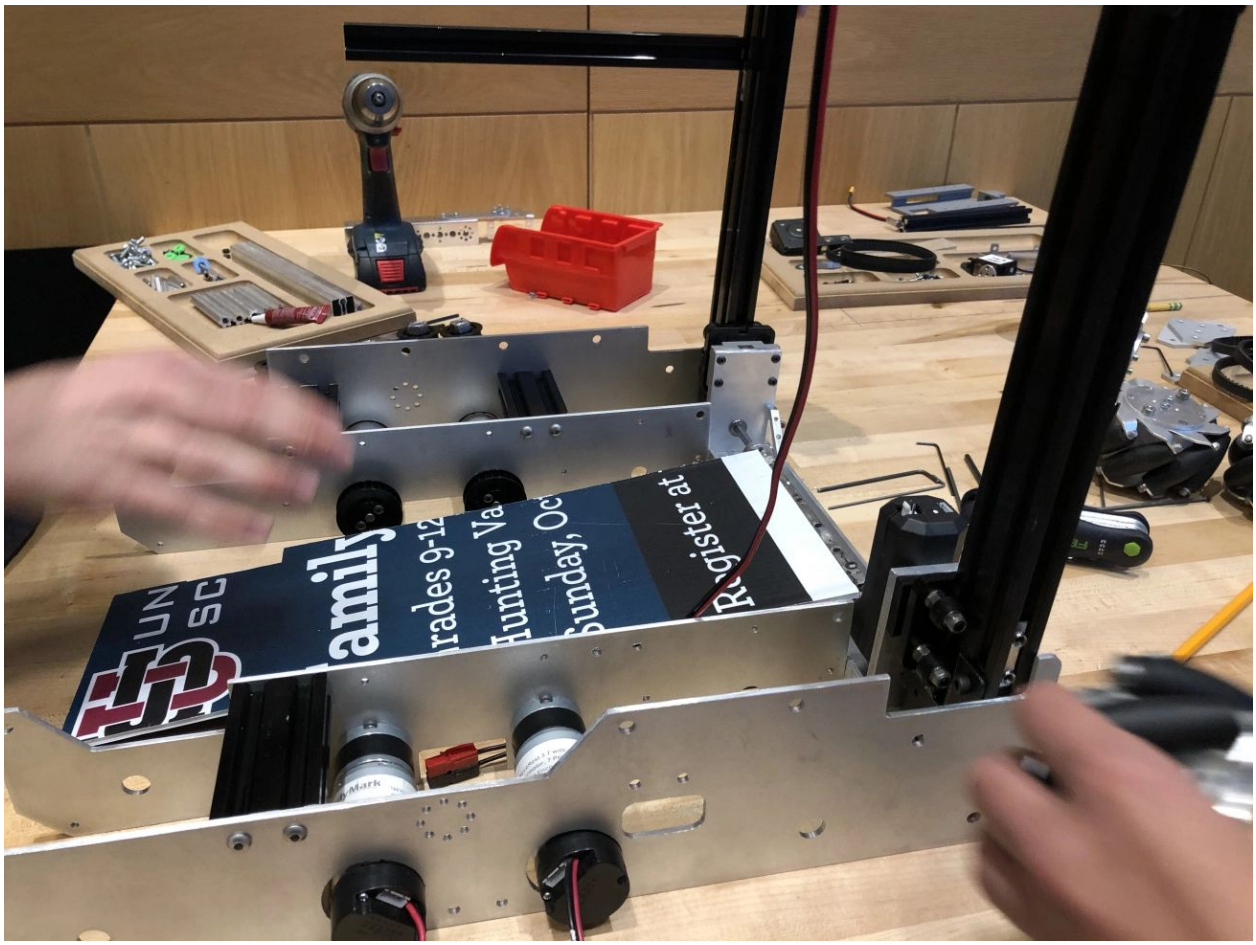
We prototyped a harvester pod.



We attached the jewel knocker to the robot.

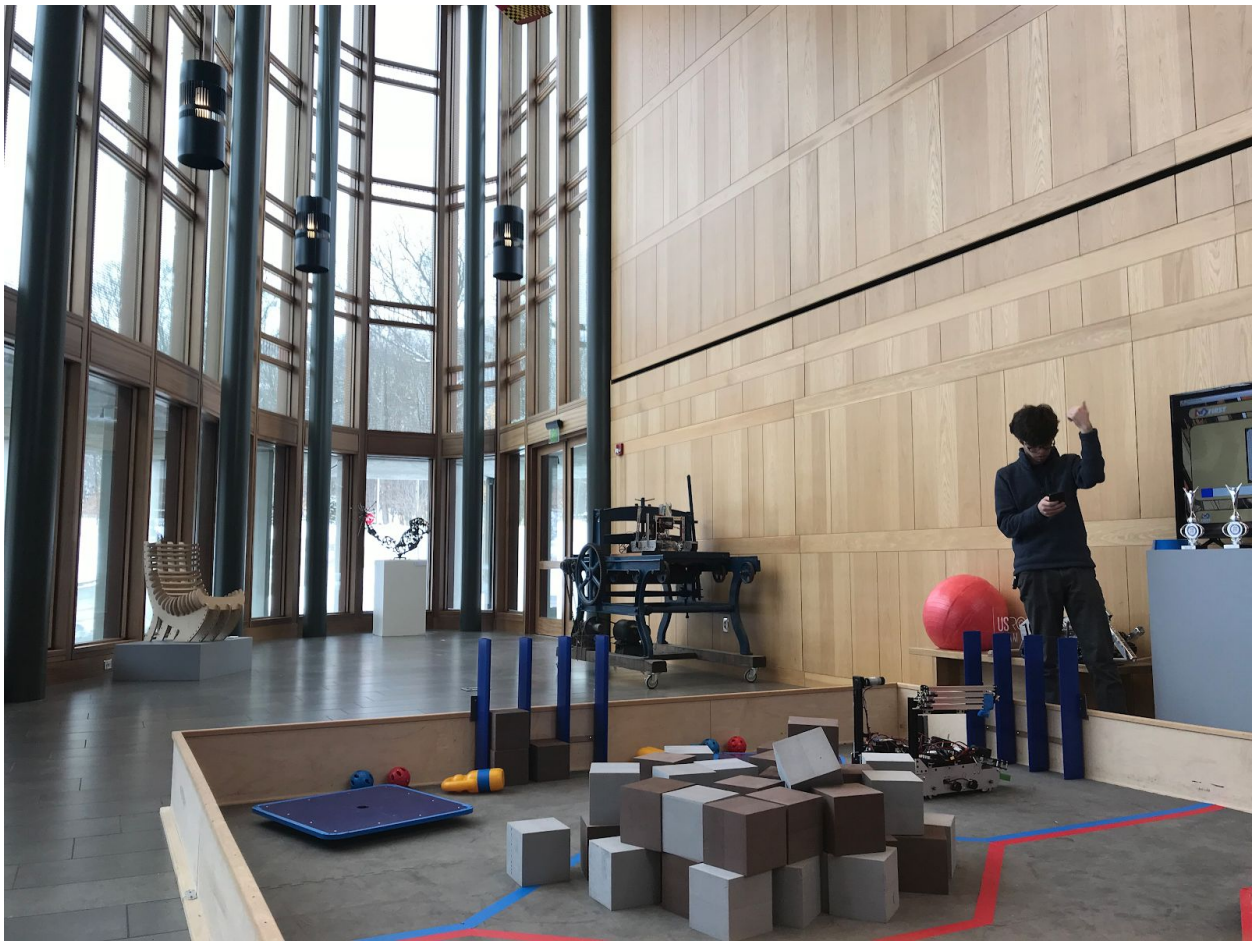


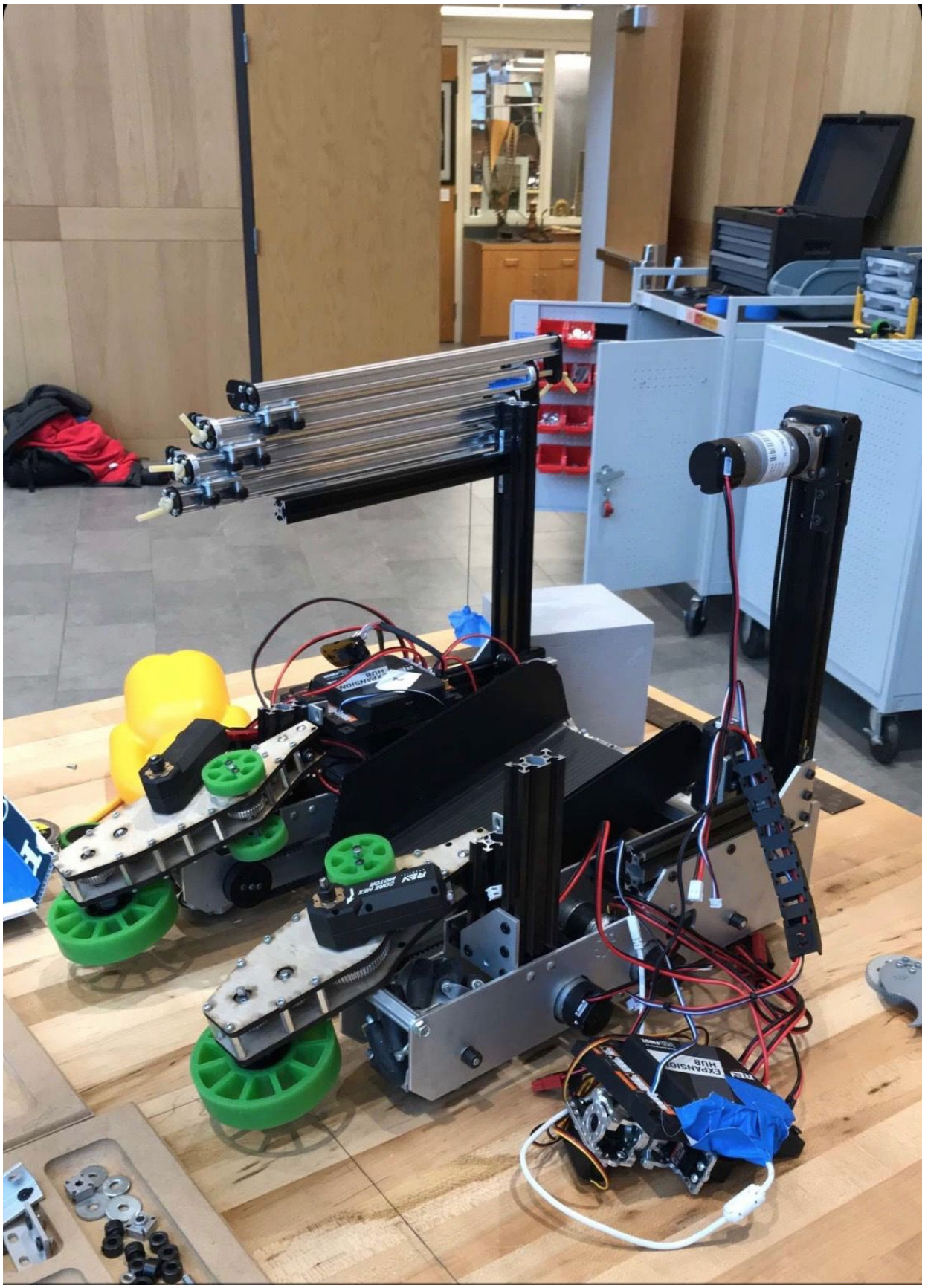
We cut the side plates out of metal and began to assemble the robot.


UN
SC
amily
rades 9-12
Hunting Va
Sunday, Oc
Register at

12/11/17

We prototyped the harvester pods out of wood. Additionally, we looked into using the Actobotics linear slide kit for our relic mechanism. We constructed the kit and attached it to our robot to start prototyping relic extension mechanisms.




EXPANSION HUB

12/18/17

We looked into perfecting our flipper and harvester designs. Additionally, we started to test relic mechanisms.

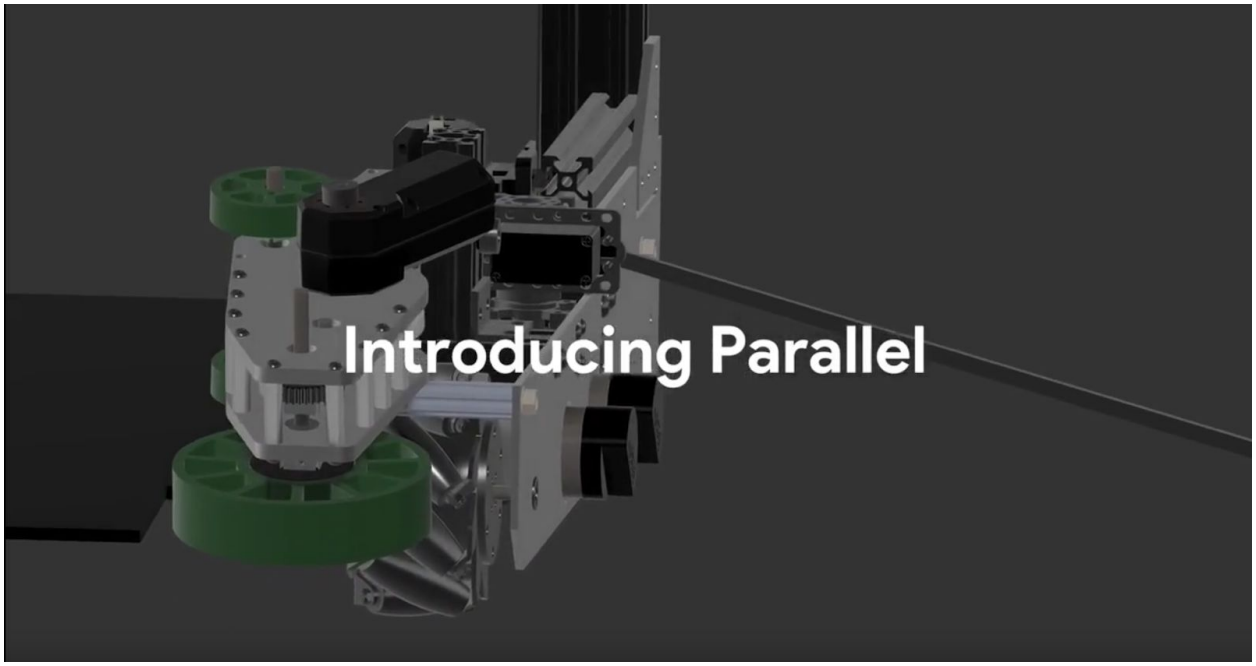
We tested the Actobotics kit without the surgical tubing in a cascading configuration. This didn't quite get us far enough.

12/25/17

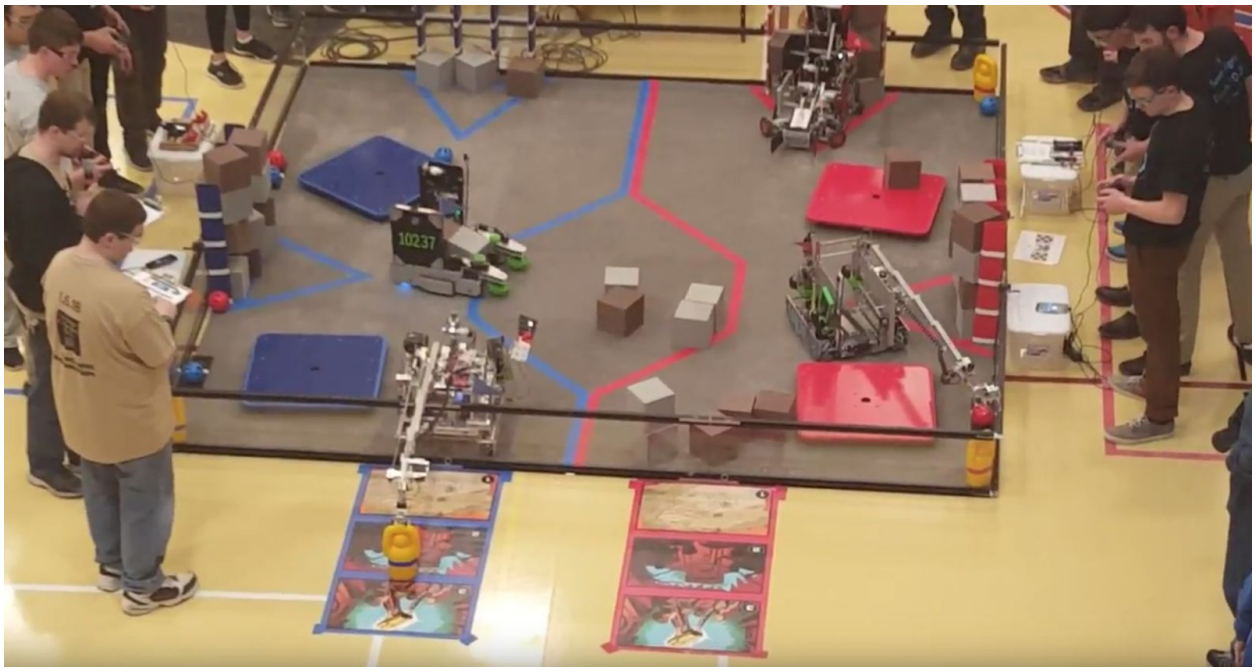
We worked on refining our robot in preparation for the Cleveland qualifying tournament. We added corrugated plastic side plates and finalized some of our wooden mounting plates to metal. We also removed the relic mechanism so we could look into its development further.

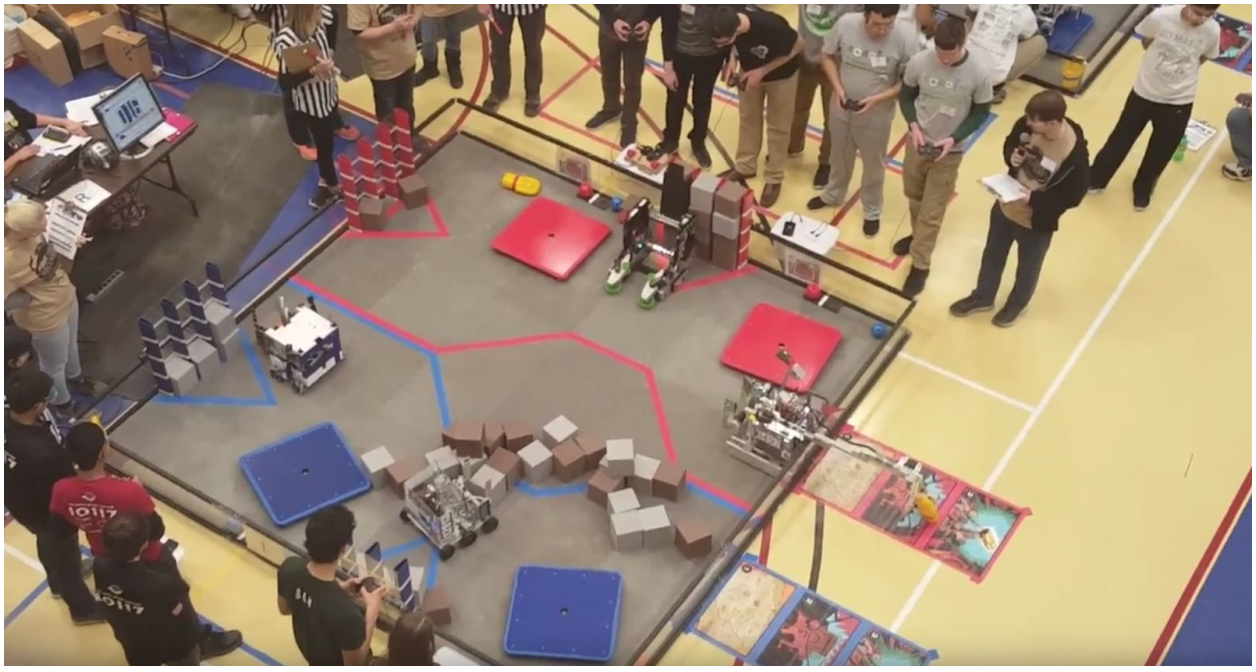
1/1/18

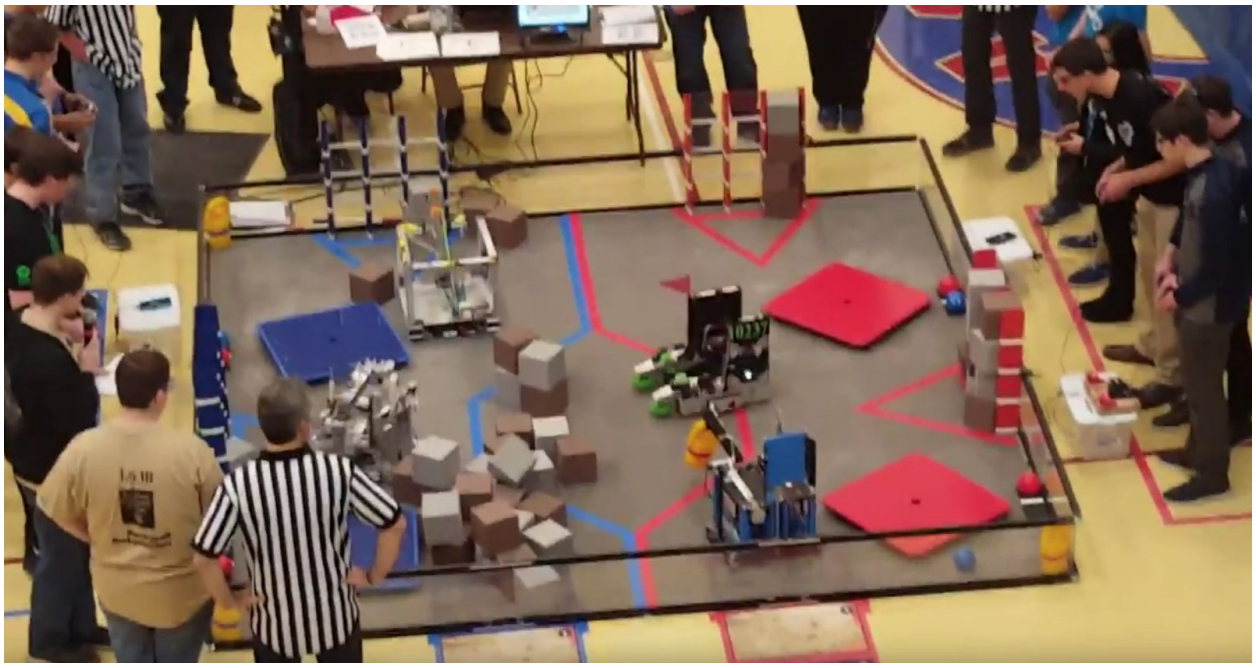
We spent the week practicing in preparation for the Cleveland tournament. We also made a CAD reveal video for our robot.



We competed at the Cleveland qualifying tournament this weekend. We won the Design award and were the first pick of the finalist alliance. We consistently completed a cipher.


10237



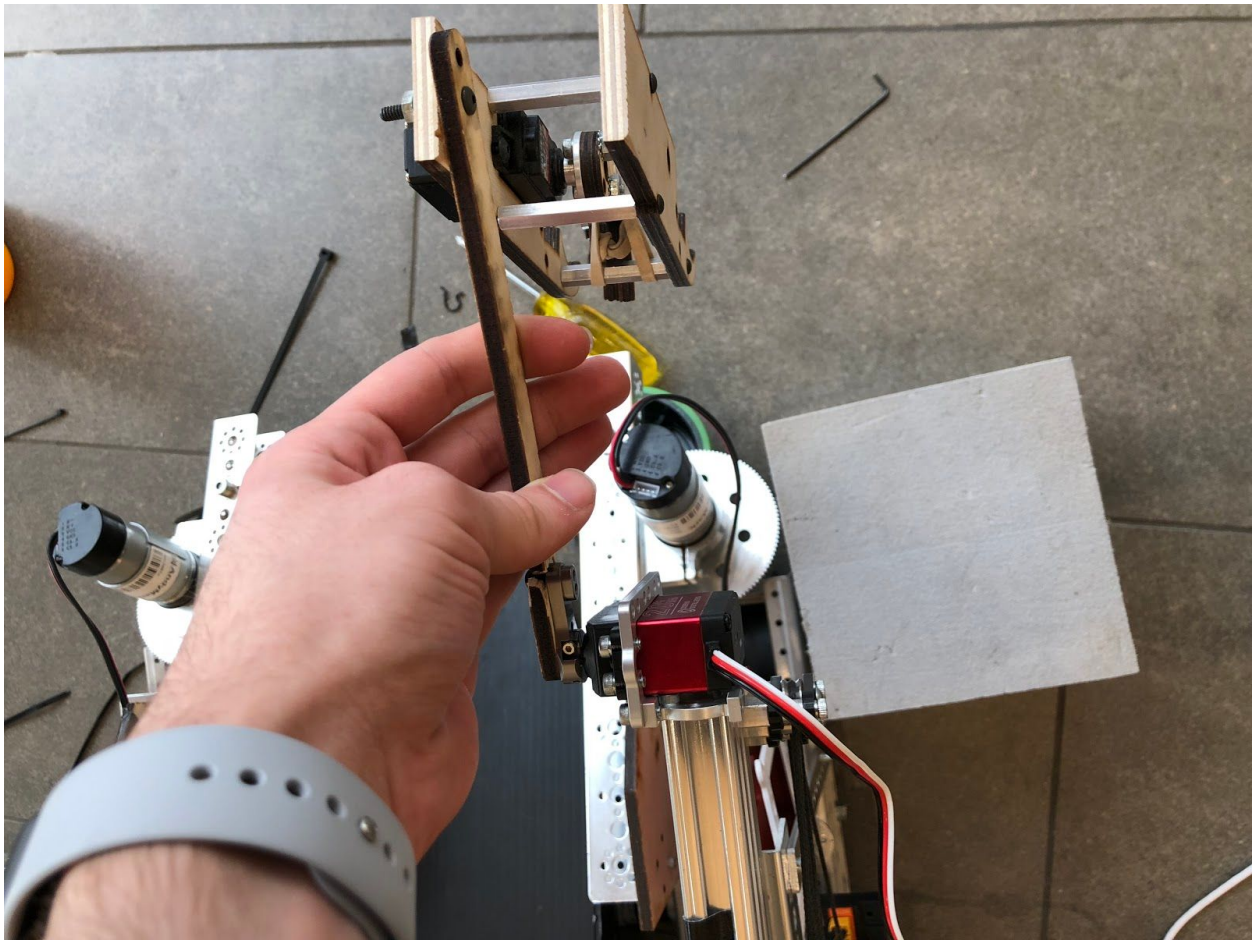


1/8/18

After the Cleveland tournament, we set out to reduce our cipher time and be able to relic by the Kent qualifier. We focused primarily on the relic mechanism. We switched to a belt based extending system.

1/15/18

We found this extension mechanism to require a lot of torque and be inconsistent due to belt slipping. Thus, we went back to the extension mechanism and looked into how to make it extend longer.

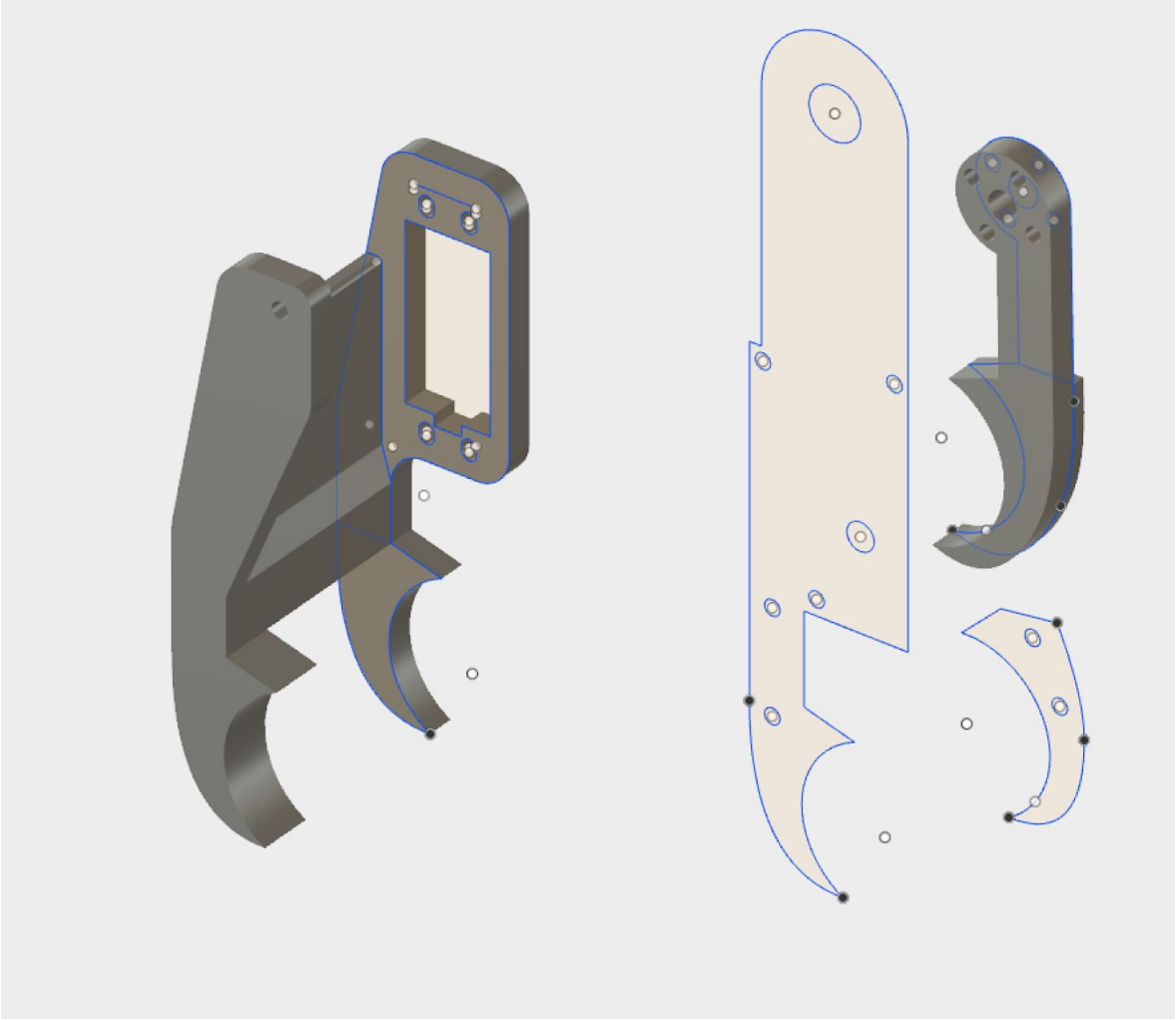


We found that a final flipping last stage attached to a servo may get us out there.

We also added a third set of wheels to our harvester, which greatly improved it.

1/22/18

We focused on a grabbing mechanism for the relic.



We designed this gripper based off of the profile of the relic. We determined that we would not be able to finish the relic mechanism by the Kent qualifier, so we focused primarily on practicing.

1/29/18

We focused solely on practicing this week. During the weekend, we competed in the Kent Qualifier. We were the winning alliance captain and we were nominated for Design award, won 3rd place Inspire award, and won the Control award.


Final Match 1


USROBOTICS
TEAM 10237

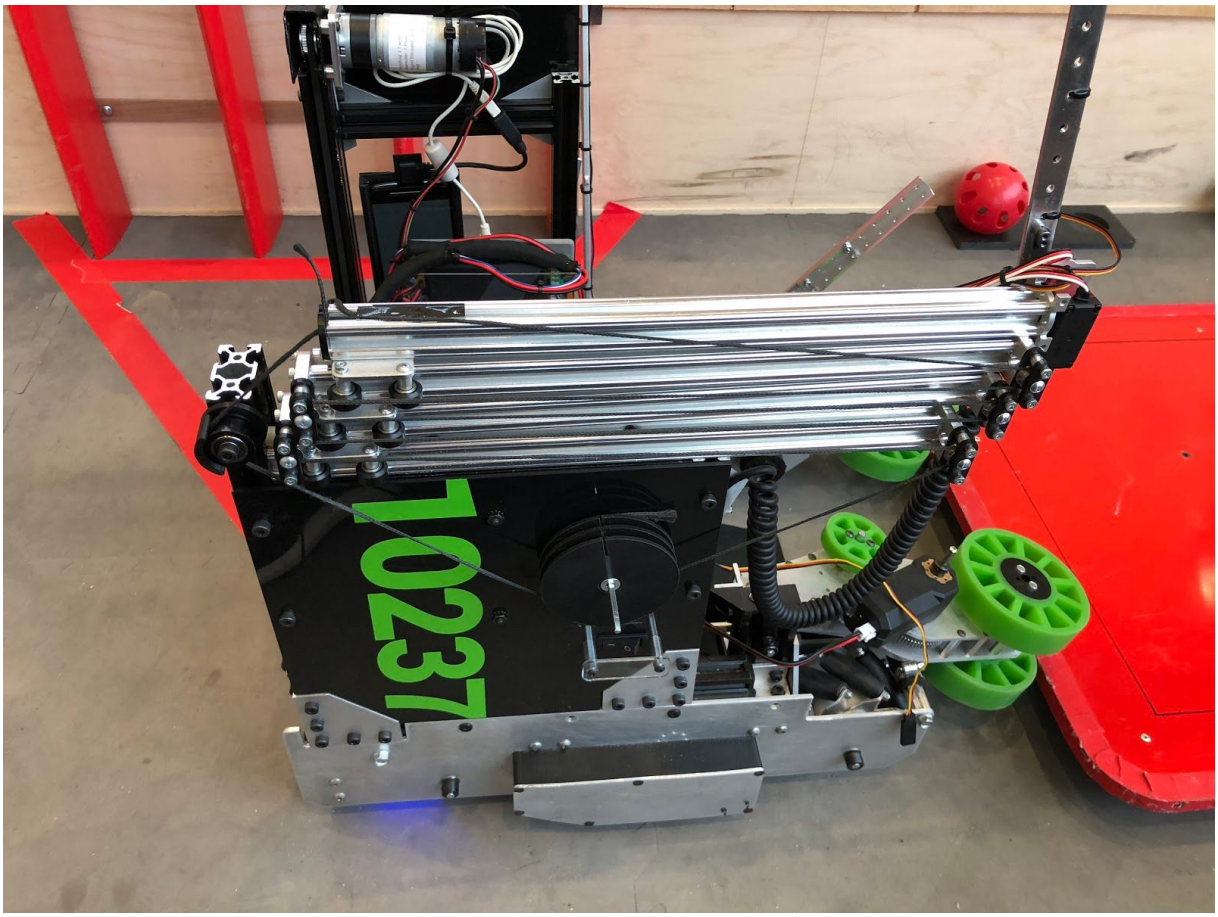
2/5/18

After qualifying for the state tournament, we focused on finishing our relic mechanism and driver practice. We added a longer final flipper extension to our relic mechanism.

Additionally, we started looking in to how we could retract the relic mechanism. We found that by attaching another wire in the inverse way of our extension wiring, we could retract the relic mechanism. We tried using one spool with two wires on it, but we found that they would get tangled and jam up.

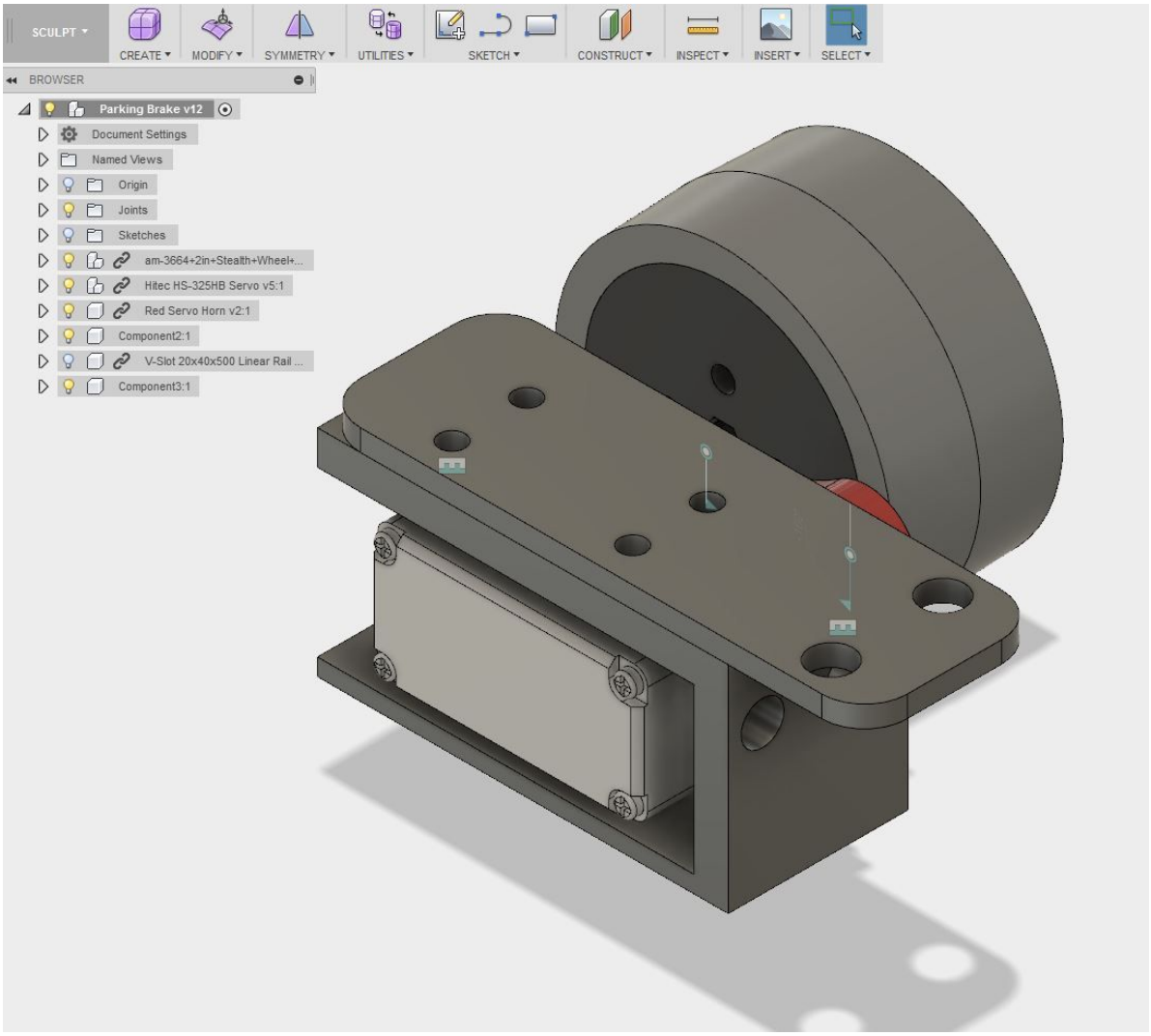
2/12/18

We solved the retraction problem by printing a spool with two channels on it to seperate the two wires. Additionally, we replaced the corrugated plastic side plates with delrin ones.



2/19/18

This week we added a parking brake to our robot. We have found that the combination of our Nexus mecanum wheels and our heavy robot, we quickly slide off of the balancing stone if we aren't perfectly centered. Thus, we designed a mount for a high torque servo. We then mounted a wheel to the servo horn. This meant that we could have a drop down wheel in the middle of the robot that provides friction so the robot doesn't slide off.



We then 3D printed the mount for the servo and attached it to the robot. We found this brake to help a lot and make parking much more consistent.

2/26/18

This week we practiced and worked on autonomous.

3/5/18

We again practiced and worked on autonomous.