Contents

Week 1

B1: Write fundraising letters

Since establishing ARES Robotics, the two teams plan to fundraise and go to outreach events together. Since ACME has more experience with writing fundraising letters, Emma was tasked with writing them. She wrote one letter for existing sponsors and then modified it to explain the teams and FIRST for potential sponsors. There are four levels of sponsorship for ACME and ARES. All of them give sponsors benefits, such as their company name/logo on the team website, company name/logo on the side of the robot, and having a private presentation to the sponsoring company. ACME had these support levels last year as well and they worked very well because the sponsor could choose their support level based on what incentives they wanted. The ARES captain, Sean, looked over the letter and approved it. After, the teams developed a potential sponsor list. and wrote another fundraising letter targeted to new sponsors.

B2: Discuss the Engineering Notebook process with both teams

With all of the new team members on ACME (and with ARES being a rookie team) it made sense to discuss the Engineering Notebook as a group. Emma made a presentation on the general setup and format of an Engineering Notebook (EN) and presented it to both teams. Then they split up into teams and Kelly discussed the specifics of the ACME EN with members. He went over how ACME uses Overleaf to format the EN and how this year members will be writing their entries straight into Overleaf.

The Captains decided to write all of the EN entries in Overleaf because it is more simplistic process than what they did the previous year (i.e. writing entries in Google Sheets and then transferring them into LaTeX). Since ACME is a smaller team this year, it was fairly simple to teach everyone how to use the platform. Having the entries written directly into Overleaf also allows the EN to be ready for competition earlier because each week can be generated when members finish writing their entries.

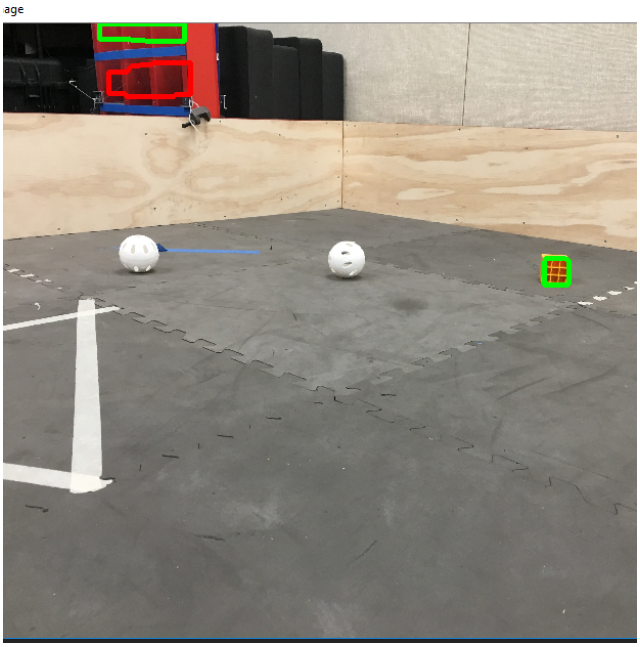


Figure 2.1: Original Position of the Camera

Week 2

S1: Test vision software

Finding the gold particle at the beginning of the autonomous posed many challenges for the software team. Modifying a python vision script from the Relic Recovery season, Emma and Kelly were able to successfully detect the gold particle. However, the problems started when they realized that with the robot starting above the ground, the camera would have full view of the crater full of gold particles behind the line of particles they were trying to detect, as seen in Figure 2.1.

After discussing possible solutions, including identifying the three particles first (both silver and gold), Emma decided to experiment with camera angles from approximately five inches above the field. Without having the field nor the mounting point on the robot, Emma tried her best to find the optimal position for the camera, although a final position will need to be determined once the robot is complete.

Although there are different ways to solve this problem using programming, for simplicity's sake, Emma is hoping to troubleshoot more with the camera when the field arrives sometime in the next week. As they already have functional code, it would be beneficial to only have to position the camera correctly.

S2: Experiment with new motor controller features

New motor functionality added with v4.0 of the SDK and the new Rev Expansion Hub firmware has the ability to have a PIDF control loop on the Rev Hub itself. This means that rather than commanding some vague 'power' to the motor, and then having to empir-

ically determine the relationship between this power and the velocity of the motor, you can directly command a velocity, and the control loop on board the hub will match that velocity regardless of voltage droop. After upgrading the firmware in the hubs and testing this functionality with a lone motor, Kelly re-wrote the MecanumDrive class to take advantage of it. After deploying this to the Relic Recovery robot, Kelly found that the robot was much easier to control. Without any feed-back control on the robot controller side, the robot was very stable, and it was much easier to move the robot slowly, because the hub would now do the work of closing a loop around the velocity of the robot.

H1: Develop intake prototypes

After the whole team watched the game video, several members began prototyping mechanisms to intake and sort cubes. Because the game requires the robot to pick up minerals that are inside the crater, the robot either needs to be able to climb over the crater to intake the minerals, or extend some sort of intake over the crater wall.

Kelly decided to try developing an intake mechanism that would allow the robot to stay outside of the crater, and reach over the wall, which would allow the team to continue using a mecanum drivetrain, for its increased maneuverability. The first concept they thought of was to extend a rake in front of the robot, on linear slides, that would pull the minerals back to the robot and up the crater wall. The rake would have tines spaced a little bit under 2.75 in apart, so the silver minerals would be pulled back, but not the gold minerals. This would allow the robot to cycle back and fourth between the crater and the silver depot, to score minerals as fast as possible. When this prototype was tested, Kelly found that the gold minerals could still be pulled back, when a silver mineral would get stuck in the tines of the rake, so the rake concept would not work to sort out the minerals.

The next concept Kelly tested was a spinning surgical-tubing intake, similar to ACME's Velocity Vortex robot. By spinning the two stages of the intake in opposite directions, the minerals would first be pulled into the robot, and then pushed against the rake, expelling the gold minerals and allowing the silver to pass into the robot. This design relied on the fact that if the intake was extended over the outer boarder of the crater, any minerals in it would be inside the crater, and thus not subject to the two mineral maximum. This worked well to pull in the minerals once they were brought to the top of the crater wall by the rake, but the transition between the two counter-rotating brushes caused the minerals to get stuck, or rejected prematurely.

After watching videos of robots from Res-Q, the last FTC game to involve the same game elements, Kelly decided the surgical tubing intake was worth pursuing, because it could quickly bring minerals to the top of the robot one at a time, which would be necessary for sorting them out. Kelly built a new prototype that had the surgical tubing whips mounted directly over one another and rotating the same direction. Ashlin built a acrylic back plate and funnel that would guide the minerals up the intake, and allow only one to pass through the top at once. Kelly built a simple sorting system that consisted of two parallel bars mounted far enough apart to allow the gold to pass through, but not the silver. When this was combined with the spinning intake, the minerals would be sucked up into the robot, and then divided into gold and silver.

After Jon finished the prototype for the intake he began to find the optimum distances the needed to be from the ground, crater edge, and edge of the robot. This is because these measurements are vital for CADing and fabrication. Jon found that the optimum distance from the robot to the roller was 7 inches. While the optimum distance from the floor to the

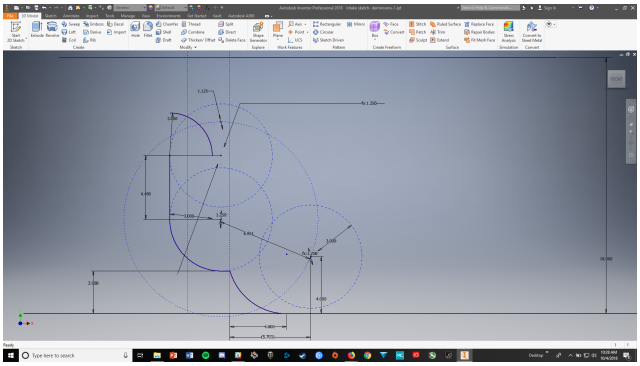


Figure 2.2: Deploy-able intake dimensions

roller was 4 inches.

H2: Investigate possible drivetrain designs

Given that a good drivetrain is vital for a successful robot, the team needed a solid design. The team decided there were two ways of going about creating a good drivetrain for the Rover Ruckus challenge. The first was to design a drivetrain capable of going inside the crater. The team thought this might be a good idea seeing as the intake design would be simpler and we could traverse all areas of play. This would need to be a drivetrain with very good traction, torque, and durability. For this purpose the team looked into a drop-center tank design (WCD) or a tracked design. Suspension was also considered. These were considered because both WCD and tracked designs are very effective at traversing rough or bumpy terrain. Like the crater rim. Jon did research in this area and found a simple suspension (bell crank) design and sketched a few ideas for a tracked drive train (fig 0.1). After discussing these together, and WCD, the team decided that going into the crater was probably not necessary. Thus, the team stopped development on these ideas and continued pursing the second way of going about designing a drive train.

H3: Develop Drivetrain Prototypes

Shawn and Ben worked on developing a rocker bogie drivetrain to test. The drivetrain used two different rotation points on both sides to be able to get over things, such as the crater wall. They used a picture from the Internet to assist in the construction of the drivetrain. After it was finished, the drivetrain worked as expected but wasn't quite within the size constraints. Sadly, the drivetrain had to be taken apart.

B1: Develop a schedule for the first part of the season

After the Build Weekend, the team got together to discuss the schedule for the first part of the season. The schedule was based around the robot goals that members want to accomplish for the first qualifier. There were several other factors to consider as well, such as leaving time to fabricate, assemble, practice auto and driving. With these thoughts in mind, here is the schedule the team developed.

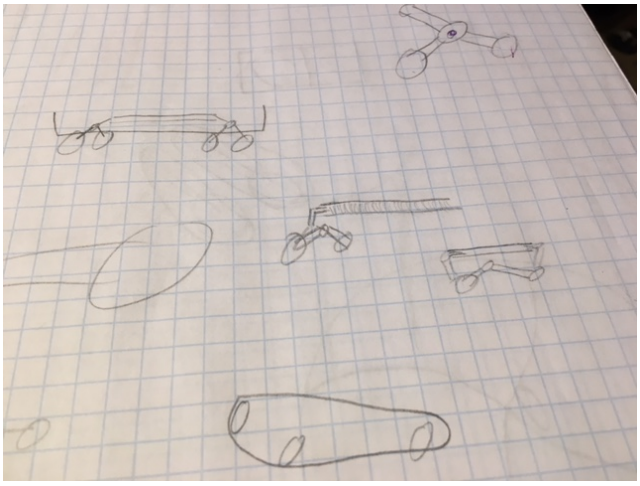


Figure 2.3: Bell crank suspension and tracked drivetrain sketches

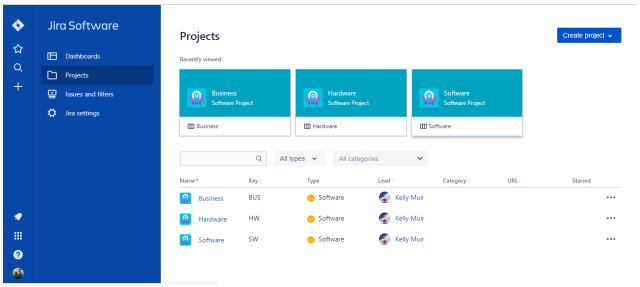


Figure 2.4: Jira Dashboard

B2: Setup a task tracking system

Last year, the team used a whiteboard in the robotics room to keep track of tasks that needed to be completed. This worked great, except, unless you took a picture of it after every meeting, you could not easily access it remotely. Which is why this year, ACME captains decided to use Jira to keep track of tasks. Jira is an online task tracking product developed by Atlassian. Team captains have the capability to assign tasks to members, as do mentors. The team has found that this software works very well for them and will continue to use it in the future. You can see the Jira dashboard in Figure 2.4.

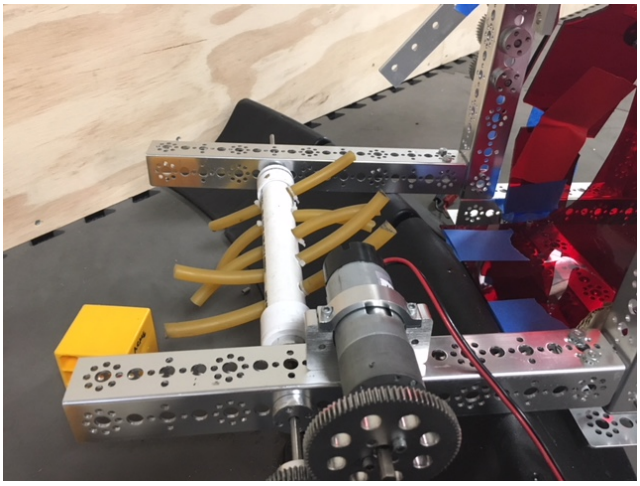


Figure 3.1: Deploy-able intake prototype

Week 3

H1: Work with intake prototypes

After looking further into the dual flywheel intake design, Jon and Kelly decided this was not a practical option because it would be difficult to fit onto the robot and create an efficient chain setup. Kelly and Jon then decided to go with a swing down roller design, as this would be easy to fit into the existing design and will likely work just as well. After deciding this, Jon built a prototype for this new intake.

H2: Develop a Team Marker

Shawn and Ben were tasked with developing a team marker. The team marker should fit within the size constraints and, ideally represent something about the team. Ben and Shawn decided to make a prototype anvil marker that could roll. Ben sketched out an anvil in his notebook and measured the height of cardboard. Based on this measurement, he divided the drawing into equal sections that were that height. Shawn the cut out the cardboard pieces and they pieced it together.

Week 4

S1: Integrate the hub feedforward functionality into trajectory followers

Before, with each update of the control system, the trajectory would return a target velocity at that given moment, which would be multiplied by the empirically determined coefficient that would convert it to a power to command to the motor. The target position of the trajectory would be compared to the actual position of the robot, and a PID loop would use this error to correct the power commanded to account for the error. Now, the velocity returned by the trajectory is directly commanded to the motor, and the rev hub does the work of converting that to a voltage that is applied to the motor. The positional error still is accounted for, but now by correcting the commanded velocity, not the commanded power. Even with very low gains on the Positional PID loop, this yielded markedly better results than the previous trajectory follower, because of the new functionality in the Rev Hubs. The second necessary change to the trajectory follower was the way error was calculated. Previously, positional error was split into two components, axial and lateral, but the axial error was always parallel to the robot's heading, and the lateral error was always perpendicular to it. Because the new pathing system treated the drive completely holonomically, this no longer made sense, because when the robot was strafing, the axial error would be normal to its direction of travel, but when the robot was driving forward, the axial error would be tangent to its direction of travel. If the robot was moving through a curve with constant heading, always on the path but 1 inch behind where it wanted to be, the error would first be primarily axial, and then primarily lateral, which would cause strange behaviours when these errors are fed into separate PID loops. To counteract this, Kelly changed it so that the axial error was always tangent to the robot's direction of travel, and the lateral error was always normal to the robot's direction of travel. This improved the usefulness of the error measurements.

```
public synchronized Pose2d update(double t, Pose2d pose, TelemetryPacket packet) {
       if (t >= duration) complete = true;

       Pose2d targetPose = path.get(axialProfile.get(t).getX());
       double theta = path.deriv(axialProfile.get(t).getX()).pos().angle();
       Pose2d targetVelocity =
           path.deriv(axialProfile.get(t).getX()).times(axialProfile.get(t).getV());

       Vector2d trackingError = pose.pos().minus(targetPose.pos()).rotated(-theta);
       Vector2d trackingCorrection = new Vector2d(
              axialController.update(trackingError.getX()),
              lateralController.update(trackingError.getY())
       );

       double headingError = pose.getHeading() - targetPose.getHeading();
       double headingCorrection = headingController.update(headingError);

       Pose2d correction = new Pose2d(trackingCorrection, headingCorrection);
       return targetVelocity.plus(correction);
   }
```

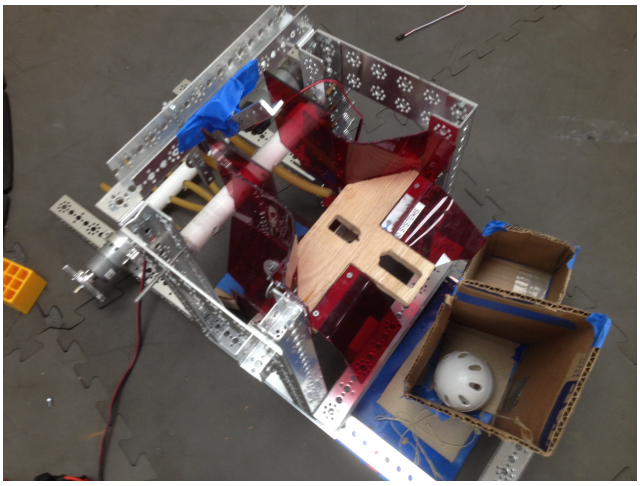


Figure 4.1: Sorter Version 2

H1: Sorting Mechanism

Ashlin, Aidan and Kelly worked on the process for sorting the cubes from the balls. The first idea was a mechanical sorting system that would function like a coin sorter, allowing the cubes to fall through while the balls rolled past. This design was discarded because it took too much space and was not consistent. A new sorting system involved a color sensor and a U shaped cup. In figure 4.1 the new system is shown connected to a prototype robot. The cube or ball would roll into the U-cup and then be read by the color sensor. The U-cup would then rotate about 135 degrees clockwise or counterclockwise to deposit the cube or ball in the corresponding cartridge. This idea did not work because there were not available servos that could rotate the required 270 and the system required too much space. This failed sorting system led to a third sorting system that was based around version two but took up less space and did not have to rotate as much. The final design, figure 4.2, allowed the ball or cube to go into the rotating circle part where it would be read by the color sensor and then a servo would rotate the circle part about 45 degrees clockwise or counterclockwise to deposit it in the corresponding cartridge. This system worked significantly better because it allowed there to be only 90 degrees of rotation needed by the servo, the process would be significantly shorter, and the whole system could be condensed to take up no more than a 6 inches.

H2: Investigate potential lift designs

The lift is going to be one of the most important mechanisms on the robot this year, as it is required for both scoring and latching. It will also need to be capable of moving very quickly, when carrying minerals to the top of the lander, and powerful enough to lift the entire robot off the ground at the end of the match. Because the team had only used Rev linear motion kits and drawer slides before, Kelly investigated the other options available for linear motion. Any potential solution needed to be capable of extending to the top of the lander, which is 32 inches off of the surface of the field, which would require two 16 inch stages, or three shorter stages. Because the lander bracket is only about four inches

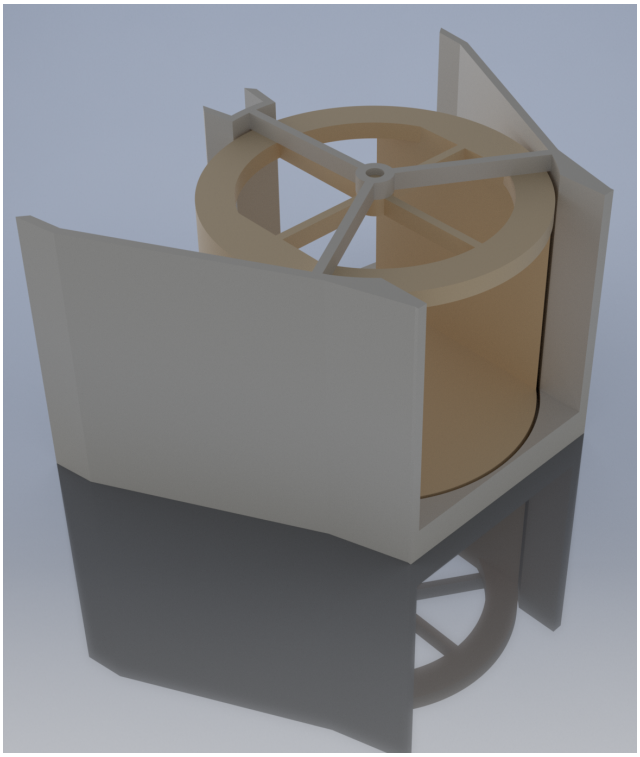


Figure 4.2: Final Sorting Mechanism

above the top of the robot, the mechanism that will attach to the lander only needs to be mounted to the first stage, meaning that any subsequent stages do not need to be as robust, as they will only ever bear the weight of the scoring system, not the robot itself. These requirements are quite similar to the FRC challenge last year, Power Up, which required teams to lift blocks above their robot to score, and then hang from a bar. Team 254, in particular, had a design that is super promising. It consisted of two fixed rails, a box-like first stage than ran between the fixed rails, and a smaller second stage carriage that ran within the first stage frame. Since this overall form factor fit all of the requirements, the team decided to settle on it for now, and investigate hardware that it could be constructed from. Team 254 constructed their lift out of square aluminum tubing and bearings that would roll against the surface, making it super fast and smooth. While this provides the best performance, the main drawback to this approach is the difficulty in manufacturing the brackets that mount the bearings to the tubing to the tolerances necessary to make everything run smoothly. If any parts are out of alignment or not parallel, then it gets a lot harder to move the lift. Another pre-fabricated option that many teams used last year to extend their relic arms over the field wall is Actobototics x-rail. The x-rail has a v-shaped groove on each side of its cross-section, allowing a v-bearing to run in the track. This would cut down on the number of bearings needed for the lift, because each pair of bearings can constrain the lift in two dimensions, not just one. If the bearings are aligned so that the majority of the forces go radially into the bearings, rather than axially, than it could be quite robust. Kelly configured the lift using these parts, and then put together a list of parts necessary to construct the lift. Because the brackets are pre-fabricated, the tolerances are much tighter than could be achieved if the team manufactured them.

H3: Design lift gearbox

Another goal for the lift was finding a suitable gearbox that would fit all of the requirements for the lift. During this process, Kelly decided against a shifting gearbox because of the necessary complexity, which meant that at least two motors would need to be used to get sufficient torque to lift the robot at the gearing necessary to make the lift fast enough. A ratchet would also need to be involved so that the robot could hang from the lander without powering the motors, making it easier to start and end the match. After researching options for a dual-motor gearbox, Kelly settled on a Versa Planetary from VexPro, because it could use a dual motor input to drive one gearbox from two motors, removing the need to join the two motors after the reduction and was modular meaning any custom gear ratio could be created; it could also be integrated with a ratchet kit to allow for one-way rotation when lifting off the ground. After assuming that the robot would weigh the full 20 kg, and requiring the motor to be able to accelerate it at 2g, a 25:1 reduction would be necessary. This should be enough to lift the scoring mechanism to the top of the robot in 1.5s, so the gearbox should be sufficient. Kelly added the necessary parts to the order sheet.

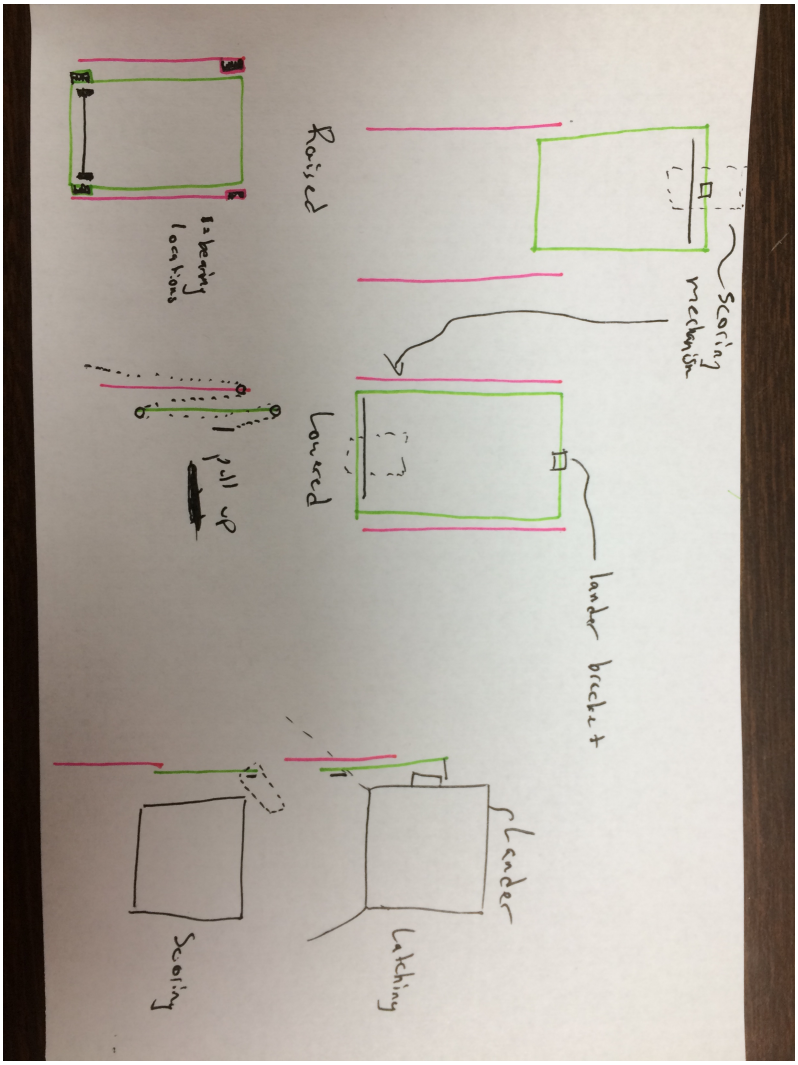


Figure 4.3: Lift concept sketches

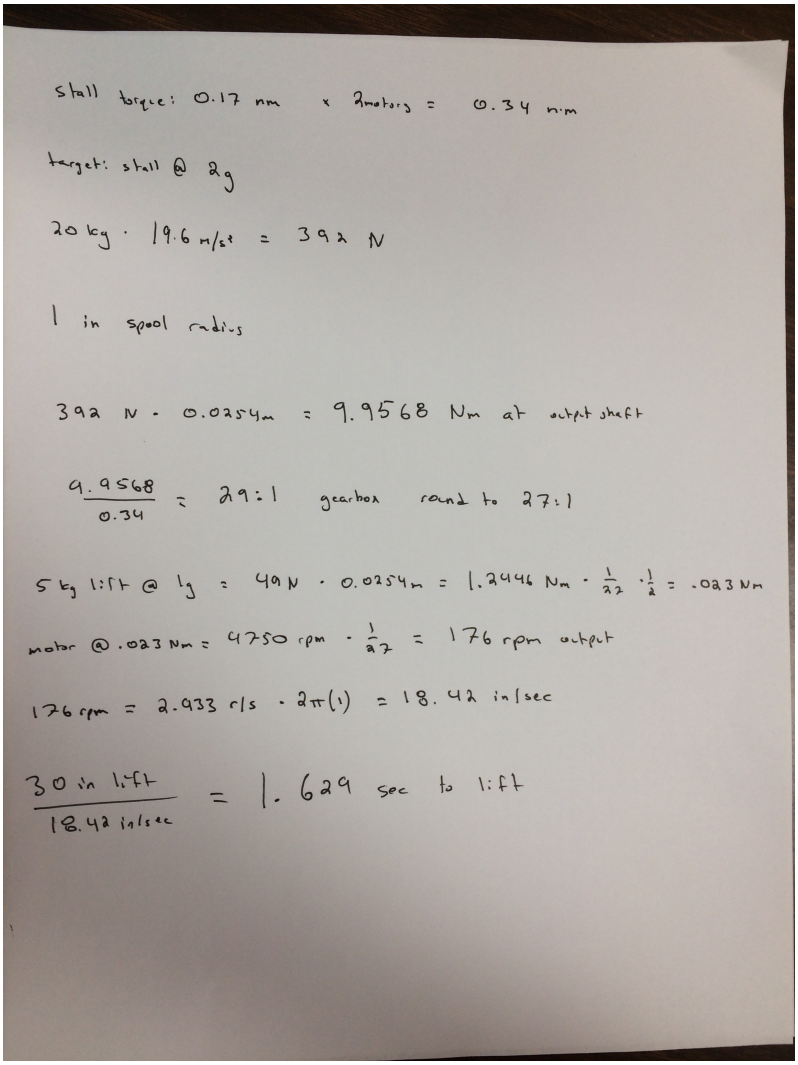


Figure 4.4: Lift gearbox math

Week 5

S1: Start Working on Lift Kinematics

This week, Emma began to work on lift code for the robot. There were several things that the team needed the lift to do. First there needed to be a driver controlled portion of the code so that drivers could manually lift and lower the lift. The other modes needed were run to position; where the lift could go to a certain position and hold position; so that the lift will stay at the position it is currently at. Emma used an enum for this. Enums are excellent for storing data that doesn't present as textual or numerical data, which is why they are ideal for this situation. Emma also learned how we use motion profiling on the robot. It is important to integrate motion profiling into the lift so that movements are as smooth and precise as possible. Here is the code that Emma wrote this week.

```
private enum LiftMode{
    DRIVER_CONTROLLED,
    HOLD_POSITION,
    RUN_TO_POSITION;

}
       switch (liftMode){

public void goToPosition(double position){
    liftProfile = MotionProfileGenerator.generateSimpleMotionProfile(
           new MotionState(0, 0, 0, 0),
           new MotionState(position, 0, 0, 0),
           1, 1, 1 //find real values eventually
    );
    liftMode = LiftMode.RUN_TO_POSITION;
    startTime = System.currentTimeMillis();

}
```

A motion profile is generated every time goToPosition is called. Emma is hoping to work on the other modes next week.

H1: Continue Developing The Team Marker

Shawn and Ben hadn't quite finished the anvil when they realized that the marker had no deployment mechanism and it looked super bad. They then decided to go with just a cube with a loop that could be dropped by a servo over the side. Shawn was tasked with making the block and he made it out of acrylic. Then Ben used basic Tetrix to make a servo mechanism that would unhook the cube and make it fall. Ben built the mechanism and found that a lot of servos didn't work.

H2: Drivetrain decision

The entire team made a matrix to help weight the pro and cons of the prototype drive-trains and already proven styles. The matrix took a lot in to consideration, such as agility, size, strength, traction, and maneuverability. The results of the matrix was a mecanum style

Figure 5.1: Acrylic Cube

drive-train driven by orbital 20 gearbox and belts. We choose this style because of its high speed, agility, and maneuverability, the downside to this style drive-train is the traction, which in turn can make playing defense a lot hard, compared to a none mecanum drive-train.

H3: Develop X-rail based lift

After downloading all the required parts from Actobotics website and importing them into Inventor. He began assembling them per the design previously created. The first issue that had to be resolved was the mounting of the bearing bracket onto the x-rail. Because the maximum amount of space should be left on the inside of the lift, a mounting scheme other than the one suggested by Actobotics would have to be used. By swapping out the standoffs that came with the kit for shorter ones, the bearings could be brought closer to the x-rail they were attached to, reducing the spacing between stages to about a quarter inch. This reduced clearance between the stages necessitated another change. The default mounting hardware would fit, but as soon as bolts were added to the model the bolts would collide with the opposite rail. To resolve this, separate mounting hardware had to be found on the Actobotics website, allowing the bolts to clear each other. It would still be necessary to use lower profile 6-32 bolts than the standard Tetrix ones, but they should be able to be found at a hardware store. The inner carriage turned out to a be a little different than planned. There was no way to run more bearings along the front and back of the first stage box, because those grooves were already occupied by the bearings fixed to the top of the fixed stage. This meant that for the second stage to mount, a second piece of x-rail would have to be attached to the inside of the first piece, allowing the carriage to attach to it, or the carriage would have to run along the inside of the first stage. This would be possible with x-rail, because if the bearings pressed against the inside, the v-groves would

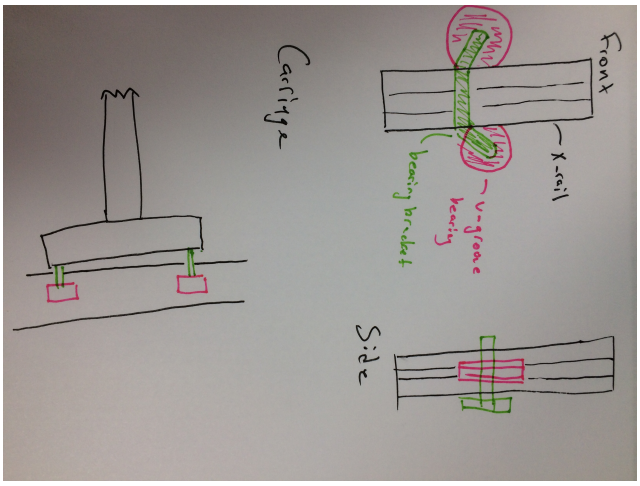


Figure 5.2: X-rail detail view

prevent it from rotating forward and back, provided the top and bottom sets of bearings were spaced far enough apart to reduce the force applied laterally. This, however, would necessitate a different mounting strategy, because the brackets would have to attach to the end of the x-rail, rather than the sides. Unfortunately, the hole pattern on the end of the x-rail did not nicely match up with the hole pattern on the bracket, so Kelly decided to put two smaller vertical pieces of x-rail on either side of the cross-brace that would support the scoring mechanism, and then mount the bearings to that.

B1: Discuss Gracious Professionalism

Although ACME members from last year were well versed in the ways of Gracious Professionalism, the new members - and ARES members - were not. As Gracious Professionalism is the underlining value of FIRST, Emma and Kelly decided to give a presentation on the subject. Pulling up a Google Slides presentation in the Drive, Emma was able to write a fairly comprehensive guide on how to be a Gracious Professional. After having Kelly read it over, and spending some time developing the Flowers Dialectic, Kelly presented it to both teams. The teams had a great discussion about how and how not to behave at competition, as well as how to be a Gracious Professional in their everyday life. The only thing this presentation could not totally explain was how rewarding it can be to practice these values. The team especially saw this last year at the Northern California Regionals, when they helped their friends, team 5214: Tech Support, by buying them an emergency servo. The team ended up being on the winning alliance because of it.

Week 6

S1: Continue working on lift kinematics

This week Emma continued to work on software for the lift. She wrote the code for the other modes that the lift needs to execute. These being the "hold position" and "driver controlled" modes. As stated in her last entry, Emma used a enum to switch between modes. Emma also experimented with motor encoders for the first time. Especially how you can use encoder ticks to find out how far up or down the lift has moved. Below is the code that she wrote.

```
private enum LiftMode{
    DRIVER_CONTROLLED,
    HOLD_POSITION,
    RUN_TO_POSITION;
}

private int inchesToTicks(double inches) {
    double ticksPerRev = liftMotor1.getMotorType().getTicksPerRev();
    double circumference = 2 * Math.PI * RADIUS;
    return (int) Math.round(inches * ticksPerRev / circumference);
}

private double ticksToInches(int ticks) {
    double ticksPerRev = liftMotor1.getMotorType().getTicksPerRev();
    double revs = ticks / ticksPerRev;
    return 2 * Math.PI * RADIUS * revs;
}

private double getLiftHeight(){
    return ticksToInches(getEncoderPosition());
}

private void setLiftHeight(double height){
    setEncoderPosition(inchesToTicks(height));
}
public void update(){

    double liftPower;
    switch (liftMode){
        case DRIVER_CONTROLLED:
            double start = getStartingPosition();
            double max = getMaxLiftPosition();
            double min = getMinLiftPosition();
            int currentPos = getEncoderPosition();
            setStartingPosition(start);
            setMaxLiftPosition(max);
            setMinLiftPosition(min);
            setEncoderPosition(currentPos);

            if(currentPos > start){
                liftPower = this.liftPower;
```

```
            }else if(currentPos == start){
                liftPower = this.liftPower;

            } else {
                liftMode = LiftMode.HOLD_POSITION;
            };
            update();
            break;

        case HOLD_POSITION:
            double liftHeight = getLiftHeight();
            double error = pidController.getError(liftHeight);
            liftPower = pidController.update(error);

            break;

        case RUN_TO_POSITION:
            MotionState currrentState =
                liftProfile.get(System.currentTimeMillis() - startTime);
            break;
    }
}
public void goToPosition(double position){
    liftProfile = MotionProfileGenerator.generateSimpleMotionProfile(
            new MotionState(0, 0, 0, 0),
            new MotionState(position, 0, 0, 0),
            1, 1, 1 //find real values eventually
    );
    liftMode = LiftMode.RUN_TO_POSITION;
    startTime = System.currentTimeMillis();

}
}
```

Of course there is more, as far as the variables and setters and getters, but this should give you the general idea. Through this Emma also realized that relying on getting the motor position to account for where the lift is at all times probably isn't very reliable. So she plans to use a Hall Effect sensor on the lift to narrow the field of possible positions.

H1: Sorter CAD

Ashlin and Aidan continued to finalize the sorting system for the robot and completed the CAD model of the sorter. Aidan realized a flaw with the system, if a ball then a cube went into the sorter right after each other then the cube could jam the sorter. This was because the interior length of the rotating piece was almost 4 inches long, so if a ball then a cube went in the sensor would read the ball and then start rotate but the cube would block the rotating and jam the servo. Kelly, Ashlin, Aidan, and Jon all thought of ideas to solve this problem including shortening the sorter's dimensions, creating a servo powered gate, or just changing the whole system. The team finally decided to solve the problem once the sorter was fabricated. The problem could only occur if the two items went into the sorter

immediately after each other. The team believed that the intake would probably space out the items sufficiently so this problem wouldn't occur, and if it did occur small adjustments could be made to increase the space between items.

H2: Develop custom lift

The x-rail lift design would work, but Kelly still wanted to investigate another type of lift, with square tubing, bearings, and custom bearing blocks. While more labor-intensive and harder to pull off, the end result would be better than with x-rail if it turned out correctly. Where the x-rail only needed bearings on one side, because the v-bearings would be able to resolve both radial and lateral loads, normal bearings can only resolve radial loads, so bearings will have to be placed on at least three sides to constrain the lift in both dimensions. Since the lift is composed of two mirrored parts, connected by the frame of the first stage, Kelly could get away with only running bearings along three sides of each tube, excluding the outer one, because the bearings in the y-axis would push the two sides against each other, preventing the lift from shifting side-to-side. Each bearing assembly would consist of four bearings, two on either side of the tube pinching it, and two next to each other running along the inside of the tube. Kelly began the CAD by making 1 in by $\frac{1}{2}$ in by $\frac{1}{16}$ in wall aluminum tubing in 16 inch lengths. One of these would be mounted to the drivetrain, and the other would make up the vertical portion of the first stage, rolling along the fixed one. A bearing assembly would go at the top of the fixed one and the bottom of the moving one, keeping the bearings as far apart from each other as possible, distributing the load and stabilizing the lift. The inner carriage would be made out of a H-shaped arrangement of more tubes, with a bearing assembly mounted at the top and bottom of each side of the H. After considering mounting the bearings by sandwiching the tube between two machined plates, Kelly realized this would not support the inner bearings, so decided on machined blocks that would be fixed to the tube with rivets. The blocks would have a quarter inch hole on the side for attaching the pinching bearings, and hole at the top where a bolt would connect the two blocks and support the inner bearings. After assembling this, Kelly found there was not enough room on the vertical portion of the first stage for both sets of bearings to roll against it, so it was neccecary for them to be swapped out for one inch by one inch tubes instead.

H3: CAD Lander Clamp

To lower and raise the robot from the lander, the team decided to attach a latch to their lift. The team decided to do this because by attaching it to the scoring lift the team would not have to create a separate lift or arm to hang from. Kelly made the original prototype for this mechanism - which used two pillow blocks that when placed around the hook on the lander had a bar slid through to have the robot hang. Jon then used this basic prototype to CAD a more finalized design.

H4: Finalize robot design

The team got together and went over the robot design, making sure that everything had been considered and accounted for before the final stages of CAD were completed and parts were ordered. Everybody presented prototypes and CAD that they had been working on, and the team decided whether or not to stick with it, and discussed any problems that

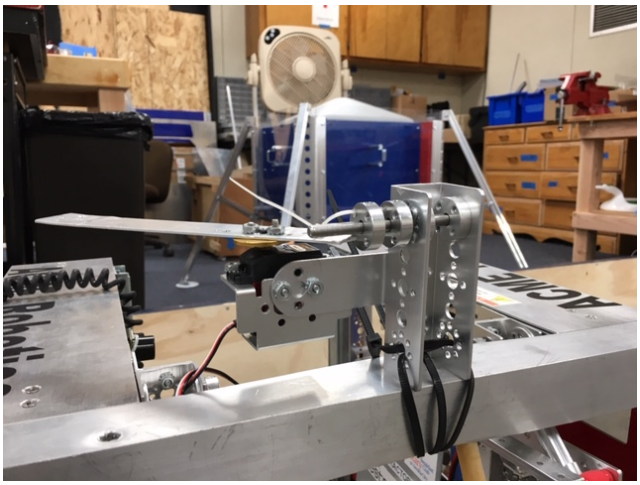


Figure 6.1: Latch Prototype

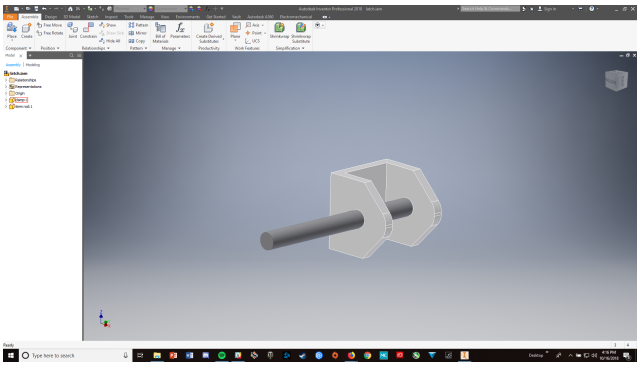


Figure 6.2: Latch CAD

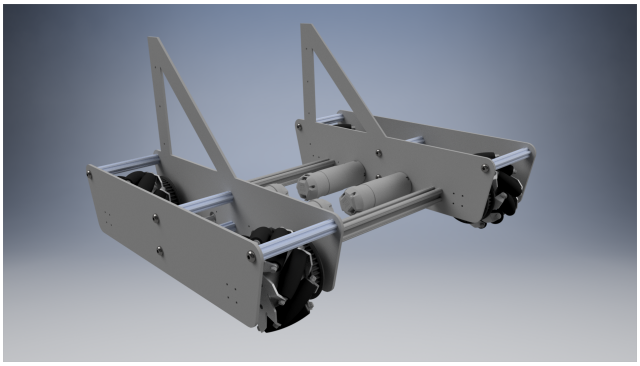


Figure 6.3: Drive train CAD

might arise as the parts went into production. The first thing to decide was the type of drive base. The team had used mecanum drives the previous two years, and was confident they could build it correctly, but did not want to dismiss a tank drive. The tank drive that was being considered was a chain driven 6-wheel drop center west coast drive. The mecanum drive being considered was the same thing that the team had used to success last year, a belt-driven mecanum drive, with dead axles constrained between drive plates giving it extra rigidity. Oren had made some improvements over last year's design, including using smaller belts and swapping out parts to increase maintainability. The team decided they should use mecanum drive for its increased maneuverability unless a definite need for tank drive was needed. Because the defensive advantage given by the advantage of tank was countered by the maneuverability of mecanum and the ability to get out of tight situations, and the settled upon intake had no need for the robot to be able to go inside the crater, the team decided to go with a mecanum drive. The team decided to stick with the intake design consisting of a rake that would extend over the crater wall and pull minerals toward the robot and a series of surgical-tubing whips to suck the minerals in from there. The active sorter using a color sensor was chosen because it would be more reliable when the robot was in motion, and the double cartridge design was chosen because it would allow the two mineral types to be scored independently. Finally, the team decided to use the custom lift rather than the x-rail based one because Oren was confident that he would have the time to manufacture it to the neccecary specifications, and Dan felt it was something that would be able to be made on the manual mill at GSS.

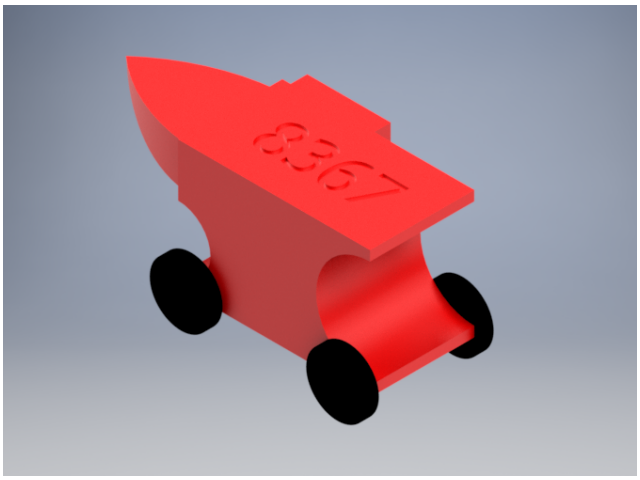


Figure 7.1: Anvil Marker

Week 7

H1: Finish Designing the team marker

Shawn and Ben worked on developing a new team marker that was an anvil. Ben took the old anvil hat CAD from last year and derived the file and re-scaled it to fit in the proper dimensions. Ben then added a hook and for the mechanism and engraved the top of the anvil with the team marker. They then rendered the model. At the next meeting, they further developed the mechanism.

H2: Complete Sorter and Intake CAD

Ashlin and Aidan worked on finalizing the attachment of the sorter to the intake and the rest of the robot as shown in 11.1. They put the sorter at a 45 degree angle to so that the objects would naturally go into the sorter and out into the cartridges easily because of the steep angle. Because of this they needed to edit the size of the cartridges so that the cartridges would have enough room to be raised up. They also attached a servo to the bottom of the sorter that it would be what rotates the sorter in either direction depending on if the item is a ball or cube.

H3: Testing Rev Linear Slide Kit

Recently, the team received the Rev linear motion kits they ordered. To see how exactly the kit went together and see it would be mounted on the robot; Jon put some of the kit together. Jon made several segments, measured, and calculated the exact amount of 15 mm extrusion would be needed for the robot. This gave the team a better idea of how it would be assembled on the robot and a basis for CAD.

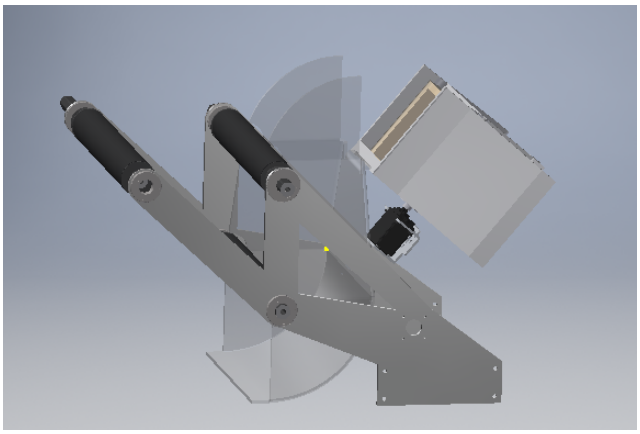


Figure 7.2: Sorter and Intake CAD

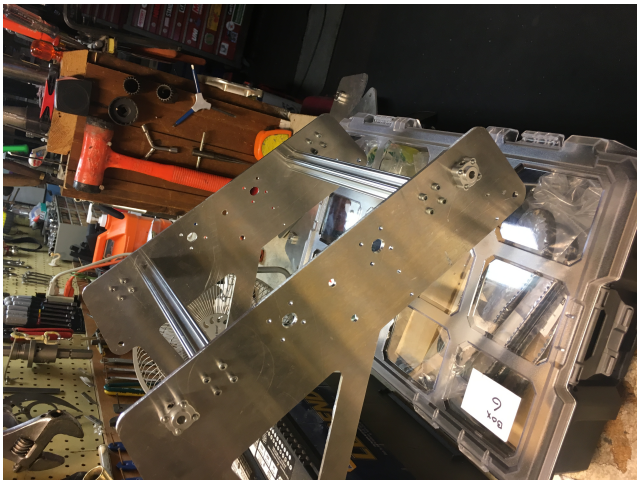


Figure 7.3: Drivetrain plates (assembled)

H4: CNC Milled Drive-Train

Oren, Ashlin, and Kelly worked with the Nu wood-shop teacher, to use their new CNC Laguna to cut out the 1/8 inch aluminum plates that make up the entirety of the drive-train. They had learn how to do auto tool changes and setting the X,Y and Z axis on the new tool as to allow them the most accurate product in the end.

B1: Decide what business tasks need to be completed before the Burlingame competition

With the team's first competition confirmed, Emma thought it might be good to start figuring out what business tasks needed to be completed before hand. Using the business Jira board, Emma wrote up tasks that covered all areas of the business team. Such as, finishing fundraising letters, writing up the goals and action plan in the EN, and writing up

ACME's outreach events in the EN. Having a list of tasks written out, whether on paper or on a screen, is better way to keep track of tasks rather than just having a running list in your head. Most of the tasks she created were specific to the EN and need to be completed before their first tournament. She will be working on several of these tasks in the weeks to come.

B2: Finalize the budget with the team captains

Finalizing the budget for the season was important to complete sooner rather than later because so many other tasks (especially business tasks) depended on it. Using the list of parts needed for the robot, the team was able to estimate how much all of the parts would cost ($2500), including how many extras they were going to need to build spare parts just in case something broke on the robot. The team also left enough room in the budget to allow for iteration, if need be ($1500). That total came out to be $2,200. As an extra precaution, the team decided to add about 30% of wiggle room onto that total, bringing the total number to $6000. The team thinks that this is a very reasonable number and is planning to reach it by sending out fundraising letters to the community. As the team is not supported by any school, it is important to decide the budget as early as possible in order to send them out quickly.

B3: Print, sign, address, stamp and send the fundraising letters

Although the fundraising letters were already written, the team still needed to print them out, address them, and mail them. Emma printed out over 50 individually addressed letters. Then, she, Kelly, and Sean and Eli from ARES signed their names at the bottom of the letters. ACME believes that this looks professional and adds a bit of a personal touch to each letter. Eli and Sean addressed each envelope and the mentors sent them out later that day, as you can see in figure 7.5. Both teams are expecting funds to come in the coming weeks.

B4: Start interviewing team members for bios

As always, the most distracting task of the season has come again. This is of course, interviewing members for their bios in the Business Notebook. Why is it so distracting to people? Because listening to Emma and Kelly give an interview to each member is for some reason very entertaining. Besides that they were able to gather all of the information that they needed. Some of the questions asked were, "What is your wildest idea for a FTC challenge?" and "What is your favorite gracious professionalism story?" Emma plans to type these bios up in LaTeXalong with a photo of each team member, as shirts and jackets arrive.

3.6.1 Expenditures

Name	Projected	Actual
Team registration	$275	$275
Competition registrations	$775	$525
Field	$266	$266
Total parts	$2,500	$1,334
Buffer	$1,500	$0
Total	$6,000	$2,400

3.6.2 Income

Name	Amount
Team Membership Fees	$2,400
Corporate sponsorships	$3,700
Private sponsorships	$1,550
Other fundraisers	$200
Total	$7,850

Figure 7.4: Final Budget

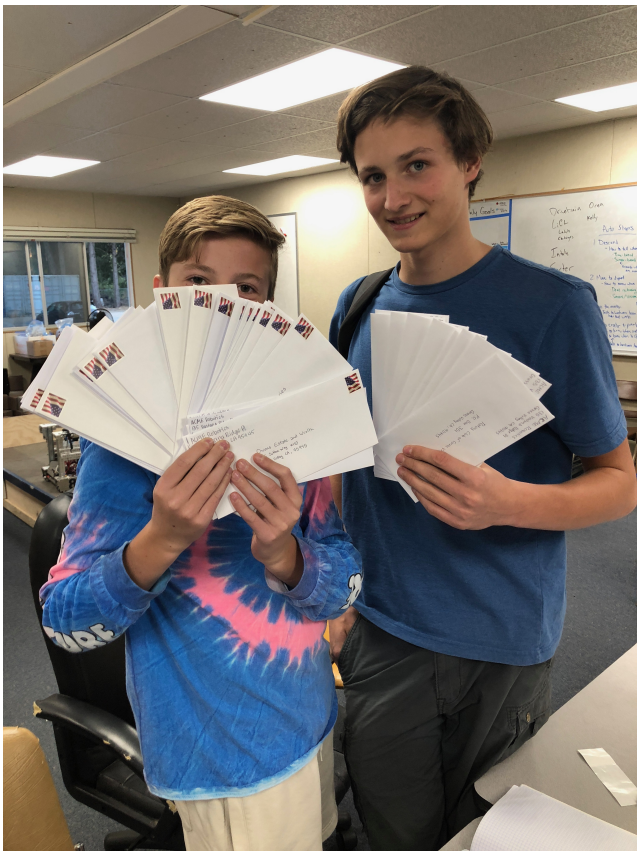


Figure 7.5: Eli and Sean with the final letters

Figure 8.1: Oren checking part measurements

Week 8

H1: Design of The Deployment Mechanism

Shawn and Ben worked on revising and iterating the marker deployment mechanism. They discussed the best way to easily drop the marker and came up with a bar and hook mechanism. This mechanism works by setting a hook on a bar and then rotating the bar so the hook slides of the end and into the depot.

H2: Turn wheel inserts for the drivetrain

On Monday, Oren and Jon went to Datum Precision Machining to manufacture four wheel inserts which had been previously CADed. The team used Datum because they had a lathe that they were willing to let ACME use. Oren and Jon were not familiar with using a lathe so the owner of Datum, Jon, showed them the basics of operating a lathe. This was important because not only did critical parts for the robot made, but ACME created better ties with Datum and picked up useful skills. Now the team can hopefully continue to work with Datum when it needs parts manufactured. Oren and Jon knowing the basics of turning adds another tool in ACME's belt as well as bolstering Oren's and Jon's individual know how.

B1: Work on the Business Notebook

The team realized that there was a lot of work that needed to be done on the Business Notebook, so Emma decided to take on that task. The BN includes all of the team's outreach events, the sustainability plan, our goals and action plans, team bios, fundraising information, current sponsor list, etc. With so much to, Emma decided to start with the team bios as those just needed to be typed in LaTeX. Although there were plenty of answers to questions she and Kelly had asked team members the week before, she only chose 4 or 5 for each bio so that they only filled one page and would leave room for a picture. After the

bios were completed, Emma started to write up the outreach events ACME did at the end of last season and over the summer. ACME had been writing about four paragraphs about each outreach event, which Emma thought was a bit excessive. So, she came up with a new format. A picture from the event is at the top of the page followed by the title for the event. A short summary sentence is after that, then bullet points highlighting parts of the event that explain what the team did, how it effected the team and how it benefited the people who the team helped or presented to. She found this method to be much more concise and easy to read. Next week, Emma plans to continue work on the BN by finalizing the budget and writing up the teams goals and action plans.

Week 9

S1: Change waypoint representations

Roadrunner (ACME's motion planning and control library), is designed to be usable by any team, regardless of their drivetrain type. For this reason, the default behaviour of the path generator is to keep the headnig of the robot consistant with the direction of travel, so that the paths are executable by a tank drive robot. Spline paths are created by calling `PathBuilder.splineTo (Pose2d pose)`, where the positional component of the pose represents the location of the endpoint of the spline, and the heading component represents the direction of travel on entering the point, which in a tank drive is the same as the heading of the robot. To facilitate holonomic drives, where the direction of travel is not neccecarily, and in fact idealy is not the same as the heading, an optional `HeadingInterpolator` can be passed, which could be constant, linear, or some other sort of more sophisticated interpolator. In most cases, a linear interpolator should be used, so that a single profile can be used for the entire path, and heading will vary linearly with displacement along the path. The problem then becomes an issue of code readabilty and path creation. To go to a point, `builder.splineTo (new Pose2d (x, y, entranceDirection), new LinearInterpolator (startHeading, endHeading))` must be called. If the start heading of one endpoint does not mach the endpoint of the last, a heading discontinuity results, making things tougher. Points with different entrance and exit directions are also more dificult to create.

To resolve this, Kelly created two new classes, a `Waypoint` class, and a `TrajectoryBuilder` class. A `Waypoint` contains a Pose2d to store the position of the robot at the Waypoint, as well as the entrance and exit angles of the waypoint. This allows a waypoint to be declared as such:

```
Waypoint SAMPLE_RIGHT_DEPOT = new Waypoint(new Pose2d(48, 24, -PI / 4), PI / 4);
```

The entrance and exit poses required by Roadrunner's path generation can be accesed by `waypoint.getEnter()` and `waypoint.getExit()`. To construct a set of paths, a TrajectoryBuilder object is constructed, and then passed a series of waypoints. The headings are automaticaly taken care of, and multiple paths are created if the waypoints require the robot to come to a complete stop on its way to traverse them.

```
public TrajectoryBuilder to(Waypoint waypoint) {
    currentPath.splineTo(waypoint.getEnter(), new
        LinearInterpolator(lastWaypoint.getHeading(), waypoint.getHeading()));
    lastWaypoint = waypoint;
    if (waypoint.getStop()) {
        trajectories.add(new SplineTrajectory(currentPath.build()));
        currentPath = new PathBuilder(waypoint.getExit());
    }
    return this;
}
```

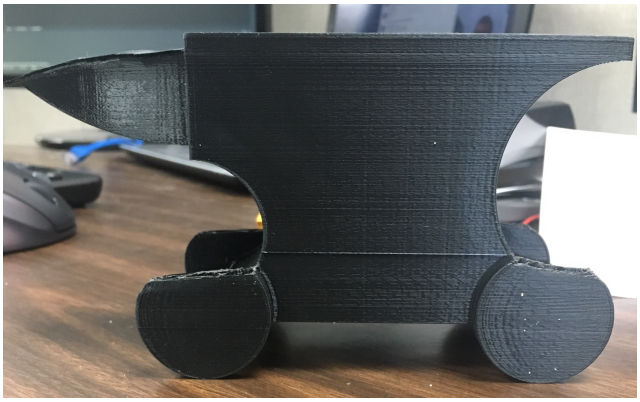


Figure 9.1: Anvil Marker

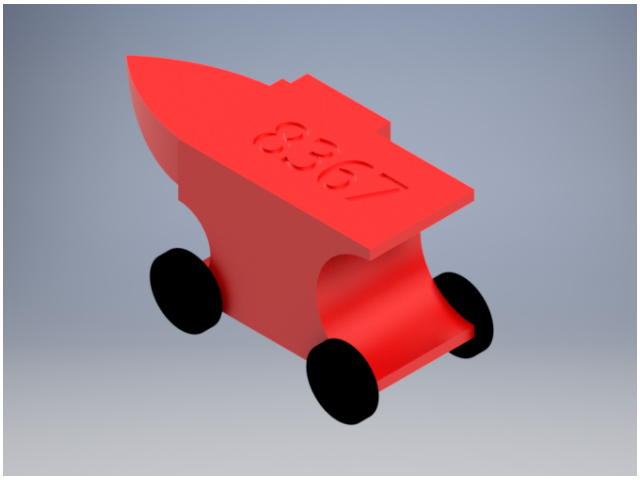


Figure 9.2: Anvil Marker

H1: Print the Marker

This week Ben finished revising the model of the team marker in Inventor Pro 2018. He then converted the .iam file to an .stl file to be printed. Then, he gave Kelly the file to be printed. Towards the end of the meeting, Ben tapped about 10 churros to be used in the drive train. Figure 9.2 shows the final marker.

H2: Fabrication

Now that the team had most of the parts of the robot, they could begin fabricating the robot. To assemble the intake and drivetrain, the metal cross beams and Vex rollers needed to be cut down to size. Jon had a chop saw and band saw at his house so he took the Vex rollers and cross beams home to be cut. To complete the rollers for the intake, holes had to be drilled in the rollers for the surgical tubing. After drilling a test piece, Jon had to rearrange the way the rollers were clamped in the vice. He then drilled the rest of the Vex

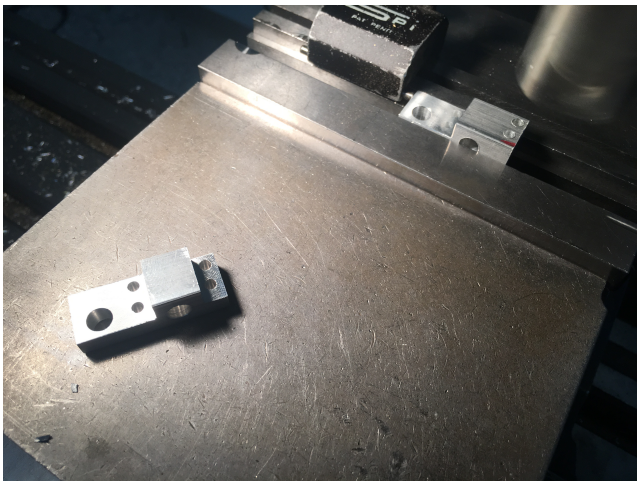


Figure 9.3: Bearing Block

rollers.

H3: Lift Fabrication

Oren went to GSS a sponsoring company to work with a one of there mechanical engineer, on milling custom bearing blocks for the lift. The team this year wanted to custom fabricated a lift that would be robust enough to hold and lift the in both the the beginning and end of the match.

B1: Continue work on the Business Notebook

It was very important to the team to print the Business Notebook at least 24 hours before the team left for the tournament. This meant that in order to meet that goal most of the big stuff that needed to be put into the Business Notebook (budget, goals and action plan, etc.) needed to be done quickly. Emma worked on writing up these things in the BN this week. First, she finalized the budget with the other team captains and mentors and added a few additional expenses such as the cost of the field. Then she put all of this information into a table that could be easily read and understood. For the goals and action plan Emma worked with Stephanie to come up with a new way to display the team's plans for the future. Basically, each goal is listed and given a short summary, then the plans to meet that goal are written in bullet form and then several paragraphs below that illustrate how the team is working to complete the goal or the results of that plan. Emma also added a section to the outreach events about future events that the team is planning. Such events include a presentation to the local Girls Who Code club, a meet up with a VEX robotics team in the county and a workshop with the Girl Scouts. (For more information on these events and other outreach events, please peruse the Business Notebook).

Week 10

S1: Investigate problem with motion profile generation

While testing pathing on the robot, Kelly found that under certain circumstances a trapezoidal profile would be returned instead of a s-curve profile. This increased error significantly by attempting to instantaneously change the acceleration of the robot, and made the motors sound weird when there were discontinuities in their pitch caused by the acceleration discontinuities. After testing Kelly determined that the problem occurred when the profile was not long enough for the robot to achieve maximum acceleration.

```
        var upperBound = maximumVelocity
        var lowerBound = 0.0
        var iterations = 0
        while (iterations < 1000) {
            val peakVel = (upperBound + lowerBound) / 2

            val searchAccelProfile = generateAccelProfile(start,
                maximumVelocity, maximumAcceleration, maximumJerk)
            val searchDecelProfile = generateAccelProfile(goal,
                maximumVelocity, maximumAcceleration, maximumJerk)
                    .reversed()

            val searchProfile = searchAccelProfile + searchDecelProfile

            val error = goal.x - searchProfile.end().x

            if (abs(error) < 1e-10) {
                return searchProfile
            }

            if (error > 0.0) {
                // we undershot so shift the lower bound up
                lowerBound = peakVel
            } else {
                // we overshot so shift the upper bound down
                upperBound = peakVel
            }

            iterations++
        }
```

In this code snippit, a binary search is used to compute the max velocity that will be reached in the case that the robot does not have enough time to reach its actual max velocity. This algorithm had been working last year, but when it was re-written in Kotlin over the summer in preparation for the public release of Roadrunner, a mistake was made in these two lines:

```
            val searchAccelProfile = generateAccelProfile(start,
```

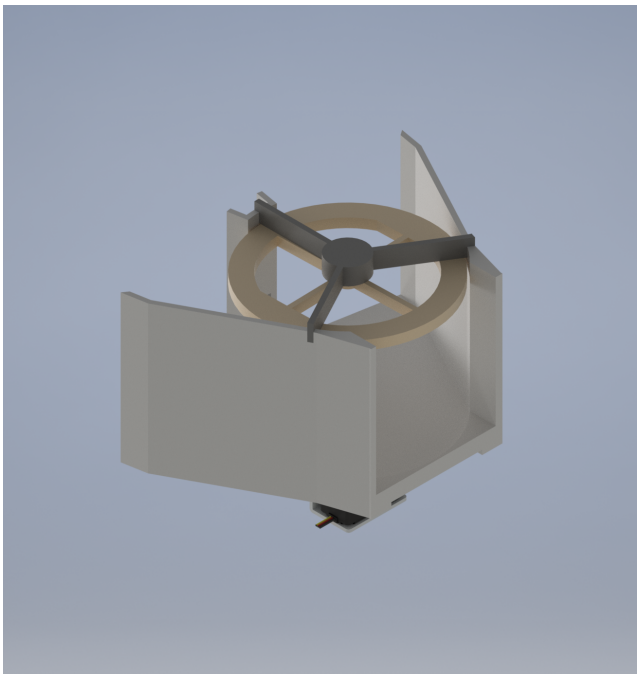


Figure 10.1: Sorter CAD

```
    maximumVelocity, maximumAcceleration, maximumJerk)
val searchDecelProfile = generateAccelProfile(goal,
    maximumVelocity, maximumAcceleration, maximumJerk)
       .reversed()
```

Rather than generating the search profiles with `peakVel`, the variable whose correct value is being searched for, the profiles were generated with `maximumAcceleration`, the constant constraint used to generate all the profiles. This meant that the search profile never actually changed, so the thousand iterations of the search would be preformed to no avail, and then a backup trapezoidal profile returned. By changing those two lines to generate the search profile using `peakVel` instead, the search would converge to within 10^{-10} in around thirty iterations, the same performance as last year.

H1: Sorter Finalization

The sorter needed to be printed as soon as possible so Ashlin and Aidan made some finalizing touches to the sorter so that it could print. First they split it the base design into two parts making the top be its own piece. They did this so that the top could be removed to allow for the swiveler to be put in the sorter and also so that the top piece could be printed separately to allow for the print to be easier. They also edited the top piece so that a bearing could fit into it tightly which would allow for the overall rotation of the sorter to be that much smoother. They printed the sorter on one of their sponsor's printers (see Figure **??**).

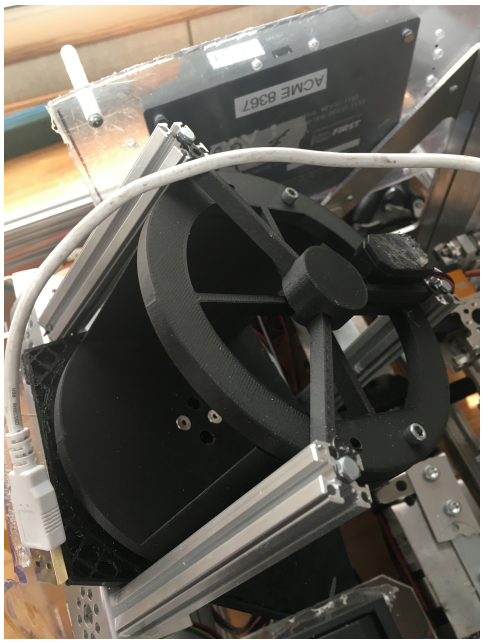


Figure 10.2: Sorter and Intake CAD

H2: Intake shield

After the intake shield was cut Ashlin and Aidan heat bent the shield so that it would be shaped exactly as the 3D model. After the shield was bent they tested it to make sure it successfully in-took balls and cubes. The testing was successful and the shield worked spectacularly.

H3: Drivetrain Assembly

This week Oren assembled all of the parts for the drive-train, there was some errors and misshapes. That include needed motor mounts, new more accurately cut cross members and some needed screw holes for mounting things. The drive-train when completed functioned actually like we had planned and was very silent when running.

B1: Add finishing touches to the Business Notebook

The final part of the Business Notebook was writing the Appendices. The Appendices are very important because they include all of the extra information that doesn't fit other sections in the notebook. For example the fundraising letters are put in to the appendix because they are very long and don't fit perfectly in the section where fundraising and sending the letters is mentioned. Therefore, they are referenced in those places but the real letters are at the back of the notebook. This is the same story for the budget details. There is a general budget in the budget section but the individual purchases are in the appendix. The final thing to do with the BN is to read through it once more, fix typos, add a few more photos and print the notebook. Emma took team and mentor photos for the bios and then a

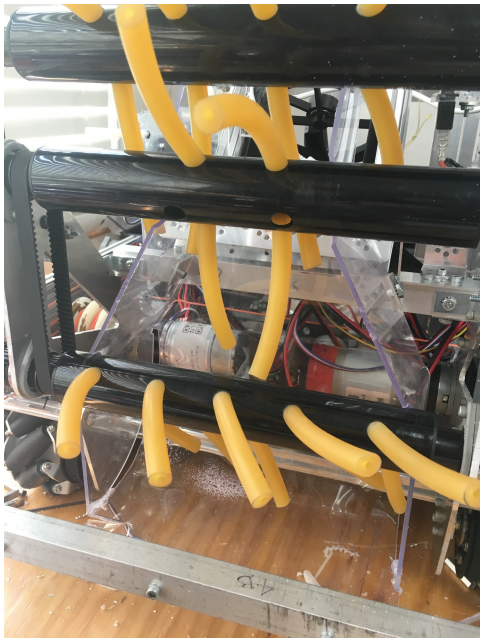


Figure 10.3: Intake Shield

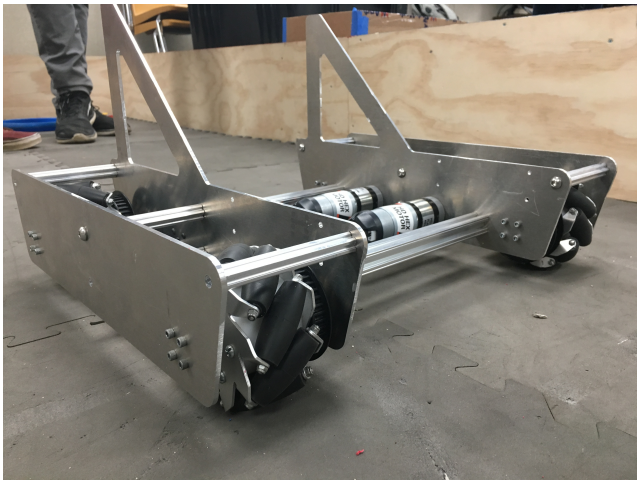


Figure 10.4: Drivetrain

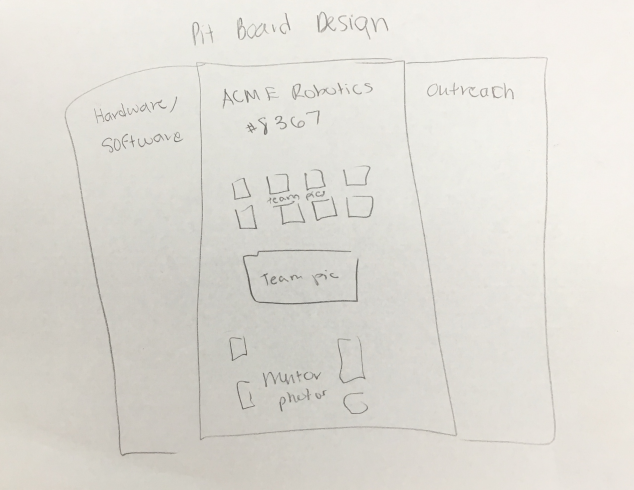


Figure 10.5: Pit Board Sketch

team photo for the "team" section. After that was done, she downloaded the BN PDF and printed the notebook. There are still a few things to add in order for the notebook to be completely done, such as write the summary page and space out the sections with dividers, but the goal to print the notebook at least a day before the tournament was met!

B2: ACME Board

This week, Ben was tasked with making the ACME board. To do this he typed up multiple short paragraphs about subsystems, outreach, and software. Then he printed them out along with photos that show a visual. In Figure 10.5 you can see the sketch that Ben made of the layout of the board.

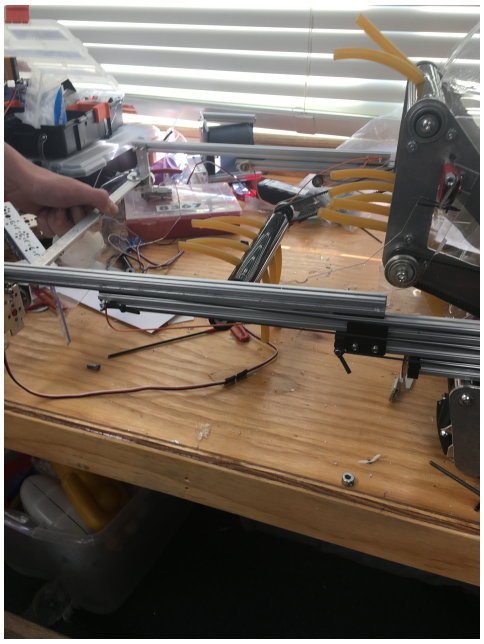


Figure 11.1: Rake and Slides

Week 11

H1: Assembling Rev Linear Motion Slides for the Rake

Continuing the manufacturing process, Jon and Aidan assembled the linear slides for the intake rake. The team wanted to create a four stage slide so that no matter where the robot was around the crater it could reach every cube or ball in the crater. However the team had only bought two linear slide kits which would only allow for a two stage slide. John and Aidan found extra parts from the previous years linear slide system. This allowed for there to be a three stage system which would give them enough length to reach to almost the back of the crater and get plenty of minerals. After constructing the slides, Jon and Aidan had to fine tune them to get them both to run smoothly and with little resistance because there was only going to be one motor powering both slides and if there was too much resistance the motor wouldn't be able to power them. After these were constructed, Aidan went on to build the rake portion of the slide and Jon attached the slides to the robot. Jon did this by using Tetrix L brackets and screwing them into the outside plate of the drive train. This made sure the slides were inside the 18 inches and were firmly attached to the robot in case they got bumped by another robot during competition. To create the rake Aidan needed to find a way to minimize the length it added to the robot. The rake had to sit low enough so that it could tuck under the intake system but it needed to be high enough so that it would pass over the crater wall. Aidan solved this by having the axle of the rake sit exactly 4 inches above the ground.

Figure 11.2: Rake Shield

H2: Shield Attachment

After the shield was built it needed to be attached to the robot. The attachment piece would attach to the side plates of the intake But couldn't attach to the back of the sorter because the screws on the attachment might catch cubes when the went up the intake. To solve this Aidan created brackets that would allow the attachment piece to screw into the shield on the sides which would decrease the likely-hood of cubes getting caught.

B1: Write the summary and cover pages for the Business and Engineering Notebooks

As the tournament draws closer, finishing the Business Notebook becomes ever more important. Even though the notebook itself was already printed, a cover page and summary page was still needed to complete the task. Emma worked on the summary page this year. The summary page is not only an introduction to the team's mission, it is where specific parts of the notebook worth noting can be found, as well as specific events. Emma adapted the summary page the team used last year by rewriting the team's story and mission for this year, altering the season highlights, and choosing specific topics of interest in the notebook. The season highlights are notable events that are exciting or great accomplishments of the team. The team's overlying goal for this year is to maintain the FIRST program in our community and train members on our team so that ACME can keep going in the future. The summary also mentioned goals on how the team wants to progress to Worlds again this year. With these things complete, the BN is now complete.

Week 12

B1: Attend the Burlingame Qualifier

This weekend, ACME Robotics attended the Burlingame qualifier. This was their first qualifier of the season and they were looking forward to seeing other robots teams had come up with, seeing how their robot performed and whether or not they would need to rethink their strategy for the next tournament.

The day began bright and early at 8 o'clock. The team arrived at the venue and set up their pit. Their pit had a poster board that Ben and Shawn had made, a huge collection of ACME wristbands and, of course, RedVines. ARES, ACME's sister team was also attending this tournament. This was their very first tournament and they were looking forward to seeing what competition was like.

Next, the team prepared for their judging session. This was the first time Jon, Ashlin and Aidan had ever been to judging. Each team member had a talking point: Ben gave the intro, Jon and Aidan talked about outreach, Oren, Shawn and Ashlin discussed hardware and Emma and Kelly explained software. Judging went really well for them. The judges were engaged in their process and acknowledged their original and custom robot design, even if it wasn't entirely functional. They had really good questions and even gave the team time to talk about what they thought was unique about the team.

Judging had put everyone in an excellent mood and Oren and Emma prepared for inspection. ACME notoriously never passes inspection for the first time, mainly due to not fitting inside of the 18. The team made it a goal for this tournament to pass both robot and field inspection the first time. Both went off without a hitch and qualifying matches began after opening ceremonies.

As the team had such a complex robot design, they did not get it all done before the tournament. They had to rethink their strategy in order to even have a robot that could potentially score. Because of this, software didn't get the robot until that morning and they were rushing to have software to compete with. With matches about to start, the team had to make a quick decision: sit out the first match to have more time to work on the robot or put the robot on the field to see what would happen. The captains decided to allow more time to work on the robot and Emma went to the match as the team representative. This was a slight downer for the team, as it cost them a match, but they agree now that it was the right thing to do.

After several more matches, that the team did put the robot on the field for, judges started to come around. Three groups of judges talked to the team: control, design and connect. Each of the captains had a chance to talk to the judges more about their subteam and all three of the interviews went off without a hitch.

At the end of qualification matches, ACME placed 20th and ARES 18th. ACME was super proud of ARES and their performance. A few team members had to go home to work on homework, but a few stayed to watch finals and for the awards ceremony. The team was nominated for two awards, Control and Connect, which they were very proud of. However

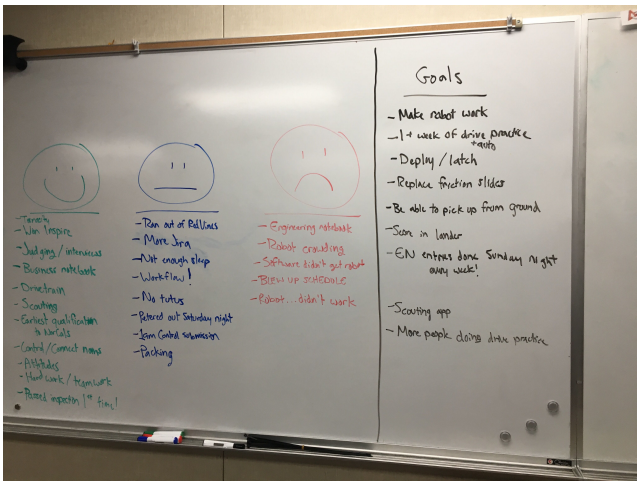


Figure 12.1: Burlingame Retrospective

the best surprise came when the final awards winner was announced. ACME had won the Inspire Award! The team had never qualified for the Northern California Regionals at the first tournament and were very honored and relieved. This meant that the team only had to worry about making the robot as perfect as possible and not have to stress about qualifying. Everyone was super excited to win this award and could not have been happier. The team is looking forward to the next competitions to come!

B2: Post Match Retrospective

After such an eventful tournament for ACME, the team had a retrospective to assess their performance. This gives ACME a good sense of what they did wrong, what they did right, and how they can improve for next time. Helping ACME to decide exactly where they want to go next with the robot and the team as a whole, and how to get there. Some of the things the team thought they did well was judging, the business notebook, and the drivetrain was very solid. Some things they thought didn't go well was the robot wasn't nearly as functional as it needed to be, keeping up with EN entries, and, perhaps most importantly, they didn't where tutus! After writing down and sorting all the reflections of the various team members; the team discussed them further and created goals for the upcoming Intel tournament. Goals are essential because they give the team something to work towards as well as keeping them focused on the important tasks. Some of the goals the team is now working towards are to deploy from, and re-latch to, the lander, create a hard deadline for EN entries, and finish the scouting app to streamline scouting. Overall the team was happy with how the retrospective went and excited to move forward and improve.

Week 13

S1: Pre-Tournament Scouting Application

Emma has begun work on a new scouting tool that will aid the team before the tournament even begins. The main function of this app will be to get information about the teams they will be competing with, especially their match history, awards, rankings, etc. Having this information will be good going into a tournament because it gives the team a good idea of what they are up against and who they should be making connections with. Emma is using the Orange Alliance API to find this information. The Orange Alliance is a website that centralizes FTC team data so that all FTC teams can have access to it for scouting purposes. Their API can be used to get all of this data. The other uses for this pre-scouting app include having a match schedule with projected and actual scores, the projected best division based on OPR scores, and potentially simulated matches that would see how ACME would fare against other teams with ACME's top score and their top score. Majority of these things would happen in the future, as Emma's main goal right now is to get a simple console app that would get the data when requested. Eventually, Emma is hoping to integrate this into Kelly's scouting app so that all of the scouting fun would be in one place.

S2: Joystick ramping

In previous years, ACME had used a linear profile for the teleop controls. This meant that the commanded speed of the robot would scale linearly with the distance the joystick was pushed, i.e. if the joystick was pushed half way then the robot would move at half its maximum velocity. The issue with this scheme is that it is difficult for drivers to precisely align, because they have to make such small movements with the joystick for the robot to move small distances slowly and accurately. To resolve this, there were multiple modes that drivers could enter that would scale the velocity of the robot by $\frac{1}{2}$ or $\frac{1}{4}$. This meant that divers had to remember to enter the correct mode before they made a precise movement, and the exit it before they traveled quickly from one point to another.

An alternative to this approach is to use a nonlinear relationship between the joystick and the robot speed. Kelly experimented in Desmos, an online graphing calculator, and came up with two options. The first was a dual-zone approach. This is two linear functions combined into one, so that the lower speeds of the robot take up a greater proportion of the joystick range, and the higher speeds are compressed into the extremes. This function is determined by the threshold between the two linear functions, and the slope of the first function.

$$dualZone(r) = \begin{cases} s_{lope}x & 0 \leq r \leq t_{hreshold} \\ \frac{(1-s_{lope}t_{hreshold})(r-1)}{1-t_{hreshold}} + 1 & t_{hreshold} \leq r \leq 1 \end{cases} \tag{1}$$

The second option is an exponential function. It accomplishes the same thing as the first, but is smooth, and only has one parameter, the exponent.

$$exponential(r) = r^{exponent} \tag{2}$$

After playing around with these functions, Kelly implemented them in a subclass of `Gamepad`. To transform the vector representing the position of the joystick into a vector representing the speed of the robot, it is separated into a heading and magnitude, the magnitude is scaled by the appropriate function (either linear, dual zone, or exponential), and then multiplied by the max speed of the robot.

The next teleop improvement Kelly worked on was power ramping. This would limit the acceleration of the robot and smooth out any sudden commands the driver applies. Each time the power was updated, a vector calculating the distance between the last commanded power and the desired command is calculated. If the magnitude of the vector is greater than the maximum change in velocity over the period of time since the last update occurred, it is clamped to the maximum allowed change, and then added to the last commanded velocity to get the new velocity to command.

H1: Hardware Discussion

After the first competition the team's goals were reevaluated. The sorting mechanism was not functional and untested. The lift motor was not correctly mounted and its spool was causing issues. The rake mechanism for the intake used a pulley system that had a tendency to unspool and the REV friction slides were not working correctly, which resulted in each side retracting at different speeds. The team had a meeting to discuss what the next steps were. They listed priorities, such as the lift, intake and the sorting mechanism. These were the things that the team felt were vital to successfully compete in the next competition. The week was spent testing several concepts that the team didn't have time to test before the Burlingame tournament.

The outcome of the discussion lead to a large change in the overall placement of specific subsystems, resulting in a great deal of CAD redesign. The team ended of changing the placement of the drivetrain's cross members to allow for the correct placement of the lift-motor. The team also realized that the design could be more streamlined and compact, by adding the intake's super-structure to the drivetrain's aluminum side-plates. Along with adding mounting holes for the subsystems that was not previously foreseen.

H2: Sorter Testing

Aidan took on the project of testing the sorter. The team didn't have time to incorporate the sorter into the robot before the Burlingame tournament and still needed to be fully assembled and tested. Aidan and Ashlin tried to assemble the sorter specifically so that it would take up as little space as possible. They did this because there was limited space on the inside of the robot and the sorter, the cartridges, and the lift motor were all competing for this room. After fully assembling the sorter and testing the the team decided it took to much room and they wouldn't pursue it any further.

H3: Latch Design and Development

Ben and Shawn designed the latch to make the robot hang. They used a design where, if the robot was fully turned off, the robot would still latch. They designed it so that the servo pulls the axle back so they robot isn't latched anymore. They then prototyped this design with Tetrix and found that there were problems. Then they solved these problems and are going to try to find a way to manufacture a more professional looking latch.

H4: Testing of Rev linear slides

After the team's first qualifier where the rake did not function properly, ACME wanted to do some testing and see what the problem was. The rake system is crucial to ACME's current design so it was important that this was done ASAP. After Jon did some testing with the rake, he found that the problem lay with the Rev linear slides. There was too much friction in the slides and the string was spooled wrong and was causing the slides to move unevenly. He also found that the string had a tendency to become un-spooled. Jon and the rest of the team decided that removing one of the slides might remedy the problem. After Jon tested this some, he found that while somewhat better, the results where still far from satisfactory. This lead Jon to the conclusion that the Rev linear slides would have to be scrapped for some other form of linear motion. Jon then commenced brainstorming, researching, and testing for several other forms of linear motion. Nearly all the options Jon looked into did not fit for what ACME had in mind. For example, drawer slides were too heavy could not reach far to far crater wall. One that did look promising however was X-Rail. Jon thought X-Rail looked promising because it was similar to the Rev linear slides in function but could carry more weight, were sturdier, and used running bearings (making them smoother and removing friction). This made X-Rail the most practical and simple option. After presenting his findings to the team, ACME decided to give X-Rail a try.

Week 14

S1: Make a GET Request from Orange Alliance API

When you need to obtain data from an API, you can use a GET request. In order to request something from an API you need to have an API key and use specific routing commands. Since Emma is going to use Java for this she made a Java application using the IDE Intellij. She had never done a GET request before, let alone in Java, so this was an interesting experience for her.

Emma used a website called Mkyong.com that provided an example of a GET request that would return the version of the API. Although Emma could have just control C and control V-ed this into her application but, it was important that she knew how this code operated so that she could understand it for the future. Therefore, Emma spent a long time learning what each line of code did and how it was vital to getting a request. Finally, after fixing a couple of mistakes, the code did what it was supposed to do (yay!) and Emma, through lots of Googling, was able to thoroughly understand how it works.

H1: Diverter

After the Burlingame tournament the team realized how complicated their approach to sorting and depositing the minerals in the lander were. This was because the system had too many steps and required too many different parts and space. After the tournament, various members of the team started hypothesizing of alternative solutions to sorting. One solution that was thought up was a single cartridge for the balls and cubes to go into where they would be read by color sensors to determine the arrangement of balls or cubes. When dumped in the lander a diverter would be used to push the balls or cubes in the one direction or the other to allow for easily scoring ten points. Ashlin and Aidan took on the task of prototyping this diverter. A problem they noticed is that the if the combination of minerals had both a ball and a cube then both items would roll out so fast that the diverter wouldn't be fast enough to sort the minerals. They fixed this problem by implementing a gate that would separate the two minerals and cause a delay between them so the diverter would have time to re-position. Aidan and Kelly tested this the diverter and found it worked fairly accurately.

H2: Latching Hook

After Ben and Shawn prototyped the servo and pin latch mechanism, the team realized that a simpler design would be easier to control and manufacture. Ben started CADing the part and Kelly suggested they use math to find how thick they should make the aluminum. So Kelly, Ben, and Aidan started discussing it, they used a combination of physics and trigonometry to determine the stress as shown in figure 14.3 . Then they used the stress analysis tool in Inventor to see where the most stress was and were the aluminum would, in theory, bend or break as shown in fig 14.4. They found that the red was only around 1.5 ksi, so they looked up the max ksi before elastic deformation and found that it was around 40 ksi. So the aluminum was perfectly fine.

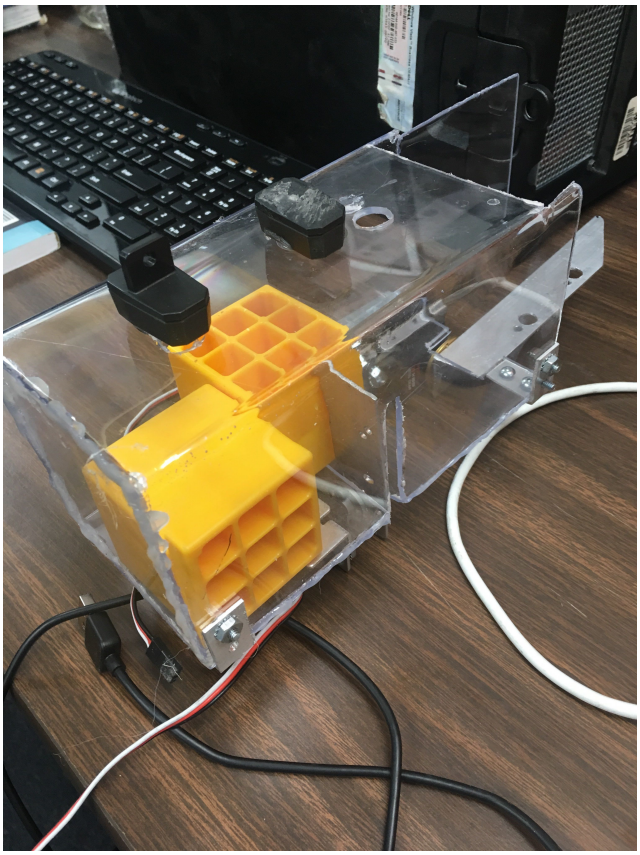


Figure 14.1: Cartridge prototype

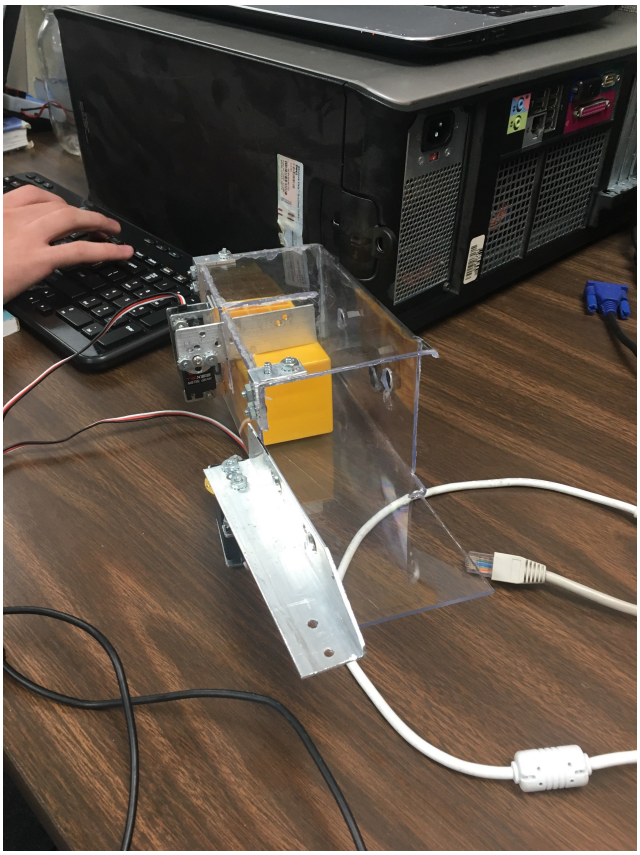


Figure 14.2: Cartridge prototype

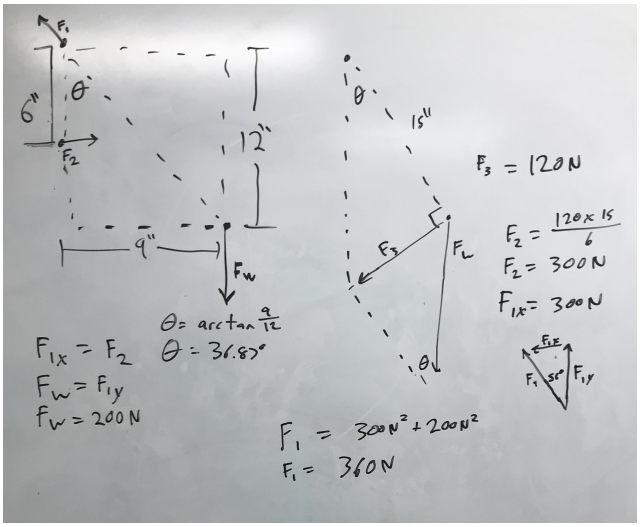


Figure 14.3: Stress Math

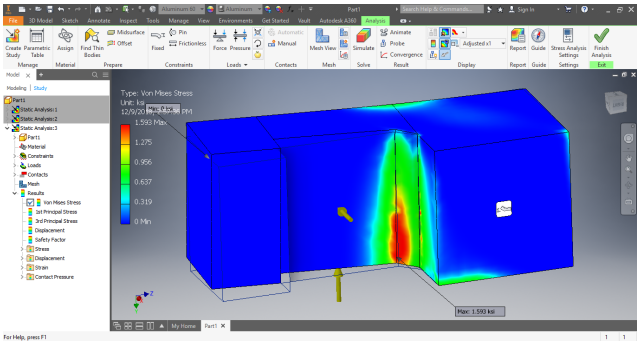


Figure 14.4: Stress Analysis

H3: Drive-Train Update

After a rush finish for the drive-train, due to the Burlingame tournament. The team felt it was necessary to simplify the drive-train to allow for more room and more precise mounting of other mechanism to the drive base. Currently on the robot their is a separate aluminum plate that holds the intake to the drive-train, which is structurally weak and allows for a lot of excess vibrations when the intake is running. As a result it was re-CADed so that the added on part was cut out with the inside plate as a whole this allows for more room between each drive half, along with being more structurally sound. Another very important change is the placement of the rear cross brace closest to the linear lift. It was moved closer to the motors to allow for more room, so that the lift motor and spool can be mounted directly under the lift for the pull down string.

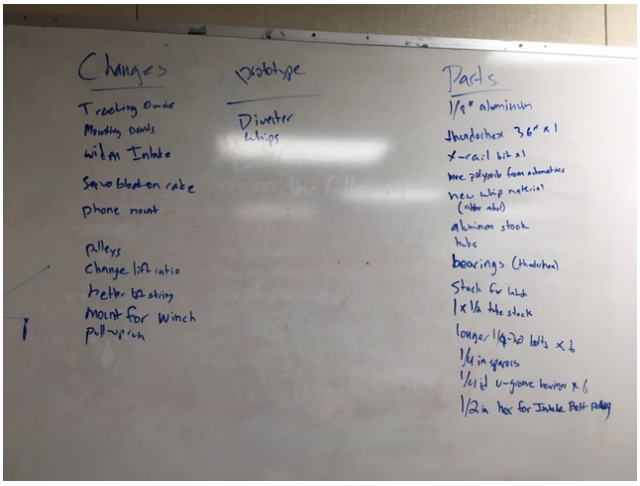


Figure 14.5

B1: Parts List

Since the team was planning a lot of revisions for the robot, Jon and Oren put together a parts list for all the parts they would need (figure 14.5). It is important that the team stay organized and keep a clear record of all the parts they buy. This makes sure that ACME keeps track of their budget (both how much money they have and how much they have spent), which parts have been ordered and which have not, and which parts they need in the first place. It also helps whoever is ordering the parts to know exactly which parts to buy and where to get them. without doing things like this to keep organized, almost nothing would be done in a timely fashion. To create this list in the first place, ACME had a hardware meeting similar to the previous one to confirm and list all the changes that they wanted to make to the robot. As you can see in image . This ensured that everyone on the team was on the same page about what was changing on the robot, what needed to be bought and built to do so, and to give a basis for a parts list. It is important that all team members are on the same page so that time and resources are not wasted. If someone on the team is spending time and material building a mechanism that the team has decided does not need to be there, then that time and resources have been wasted. Keeping everyone on the same page about what is happening to the robot keeps everything running more efficiently.

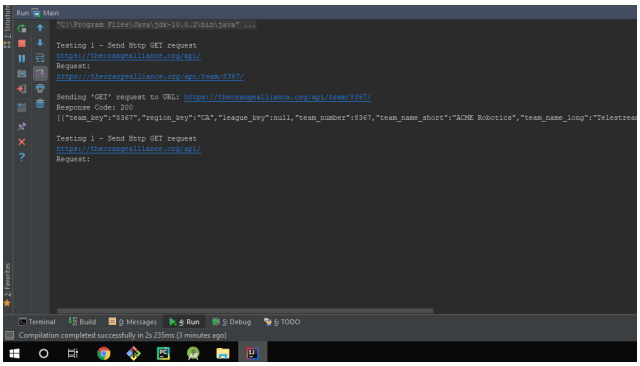


Figure 15.1: The Console for the Prescouter

Week 15

S1: Set up a console for the Prescouter

The next step for the prescouter was to make it usable. This was accomplished by using a console. Emma chose to make a simple console application first, rather than tackling a full UI because it is easier to work out all the kinks of the code before adding the additional stress of a making a functional UI. So for now, having a console will satisfy the needs of the app. Since there are different commands for the Orange Alliance API, Emma needed to make it so that what the user typed in would be the URL that was sent to the API. She accomplished this by using a scanner. This was the first time Emma used a scanner and at first she thought it would be rather difficult but it turned out to be fairly simple.

Emma realized that it might be useful to keep the program running in a loop so that the user didn't have to play the project again and again. She used a while loop with a constant variable set to true, so that the program could run indefinitely. With this little addition done, the console was fully functional as seen in figure 15.1. The next step for the prescouter is parsing the JSON response from the API so that it is easier to read.

H1: Diverter

After the success of the prototype diverter the team decided to make it their primary sorting and depositing system. Ashlin and Aidan began to make a hopefully final version out of poly carbonate. They used poly carbonate because they knew that when they were depositing minerals they were only going to be able to tip the diverter at a small angle and so they needed a slippery material so the cubes wouldn't become stuck. The completed diverter is show in figure 15.2.

H2: Testing X-Rail Slides

Now that the X-rail had arrived from order, the team saw fit to construct a set of X-rail for testing. Jon, Ben, and Shawn took on this task. Given that the X-rail was as yet unproven, it was important to run it through its paces to ensure it was a good option. While on paper the X-rail was perhaps the best option for the horizontal linear motion the team was looking for, in practice things could be different. After Jon, Ben, and Shawn completed the

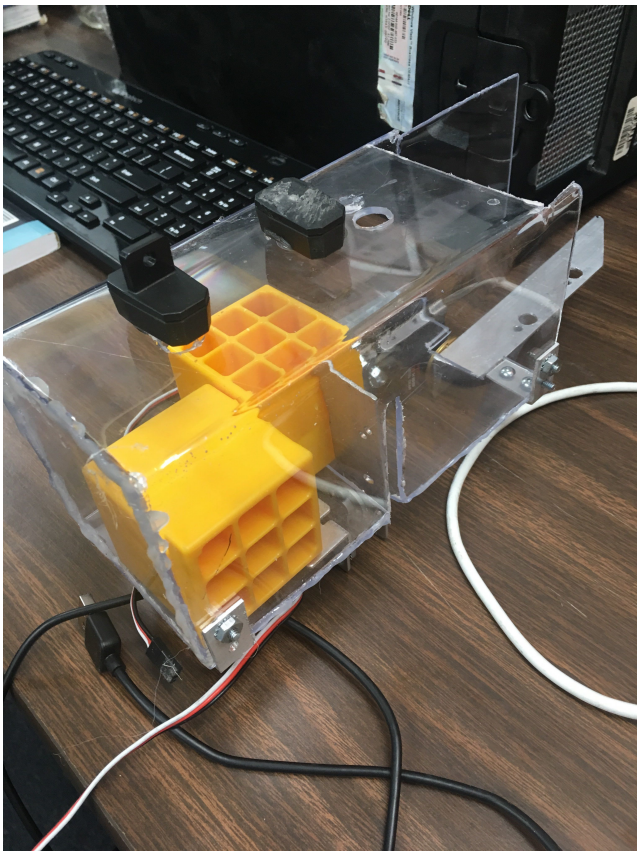


Figure 15.2: Diverter Design

Figure 15.3: X-rail installation

X-rail kit, Jon began testing the X-rail slide. He tested that the surgical tubing was taught enough to pull blocks and balls back to the intake, he tested how and where it would be mounted, he tested the rigging, and he tested where the servo for the rake would mount. After testing all these qualities, Jon determined that the X-rail would be a good replacement for the Rev Linear slides. While the X-rail would function in almost the same way as the Rev linear slides; A couple things would have to be changed. The rake servo would have to be mounted lower, and the rake would have to rotate from side to side rather than up and down. Being that the X-rail is larger than the Rev slides, the rake servo would need to be mounted lower so the rake could be as close to the ground as possible. Even with the servo being mounted lower, it was still high enough off the ground that the rake would have to deploy in a different way. The team decided that having the rake swing down from the X-rail would not only be a more efficient use of space, reduce wobble and strain, but would also more easily allow the use of X-rail. This would require the design of a new rake which set to work on.

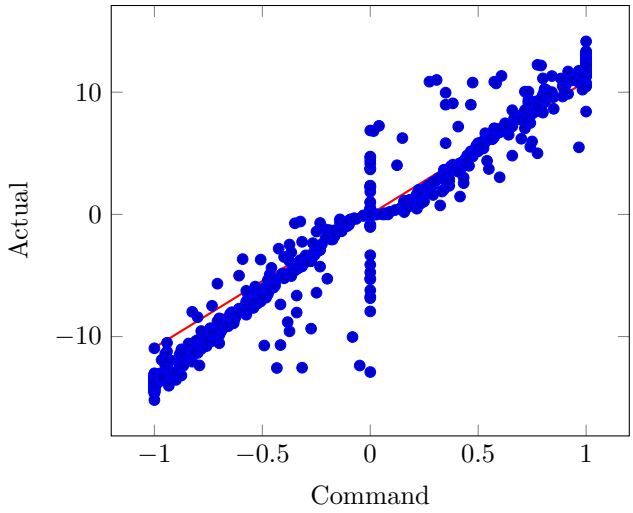


Figure 16.1: Actual vs. Commanded velocity

Week 16

S1: Tune lift kinematics

The lift needed to be motion profiled in order to run during auto (to get off of the lander, and then lower the lift back down), as well as to run it automatically during teleop, so the drivers only need to press a single button for the lift to automatically go to a set position. Because the two motors of the lift drive the same gearbox, it would not work to use the on board PIDF velocity control. If the Rev Hub commanded different voltages to each motor, they could end up fighting each other, and potentially burn out, or at least reduce the efficiency of the lift. To avoid this, Kelly decided to run the motors in an open loop mode, where the voltage applied to the motors will be commanded to the rev hub in a number between -1 and 1. This means that a feed forward constant will have to be used within the subsystem code, rather than on board the rev hub. This constant relates the speed commanded, on the interval of -1 to 1, to the actual velocity of the lift, in inches per second. If this were the drivetrain, a opmode could run the lift to collect the neccecary data, but since running the lift too far in any direction could potentially break the robot, Kelly did not feel comfortable running the lift autonomously until the control loop was tuned. Kelly instead ran the lift manually, moving it up and down with the joystick at various velocities and positions, while logging the data.

By preforming a regression on the data, Kelly determined kF to be 0.0917. Kelly also determined 14 m/s, 30 m/s/s, and 30 m/s/s/s to be reasonable values for $maxV$, $maxA$, and $maxJ$ respectively.

Kelly then tuned the lift PID loop. Because the lift only has one degree of freedom error is not the end of the world, and smooth motion is the priority, so Kelly omitted D from the loop, and kept P and I low. After the loop was tuned the lift was able to run to a desired position at the press of a button during teleop.

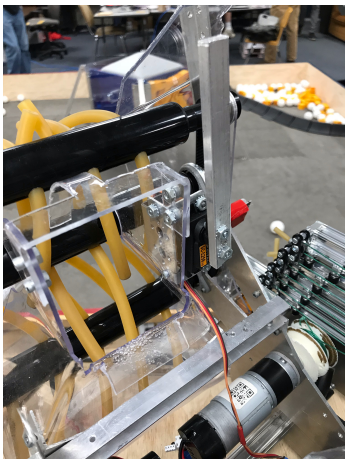


Figure 16.2: Intake Gate

H1: Intake Gate

After completing the attachment of the diverter to the robot the entire intake was almost complete. Aidan realized that to meet the two materials maximum rule the intake was going to need another gate that would stop any more materials from going into the diverter. Previously the team had thought that this problem was already taken care of with the cartridge itself. The cartridge was supposed to be able to only hold two and then the gate at the top of the cartridge would close and stop any excess minerals. This was not going to work because the way that gate closed would actually pull in more minerals and get them caught in the robot. To fix this problem a gate needed to be created at the top of the intake. Shawn created this gate as shown in figure 16.2. This fixed the problem because it stopped any extra minerals from going into the diverter and the extra minerals could easily be repelled with the intake running reverse.

H2: Attaching the rake and testing the intake

Now that all systems of the intake at least had one prototype, the team wanted to attach them all to the robot for testing. The team quickly ran into complications with the rake however. Originally, ACME planned to mount the rake in a similar fashion the the Rev linear slides by bolting the bottom to the outside drive plate. However, the X-rail was much taller and longer then the Rev slides; meaning the servo the rake mounted to would be to high off the ground and outside the 18'. The team first tried to mount the servo lower with a long bracket. This turned out to be too flimsy and was quickly scrapped. Next the team tried an elongated arm from the servo to the rake. This produced similar problems of stability and got in the way of deployment. The team then discovered that they could mount the X-rail upside down by mounting it from outside drive plate. This solved the issue of the servo being too high up and far out. It would also streamline the whole design and was stronger than any previous solutions. After some brainstorming about how exactly to mount it the team decided on cutting a new outside drive plate with an extended top that would curve over and mount to the top of the X-rail via bolts. (Fig 0.2). To test this idea, the team created a prototype and mounted it to the robot. This way they could test if it

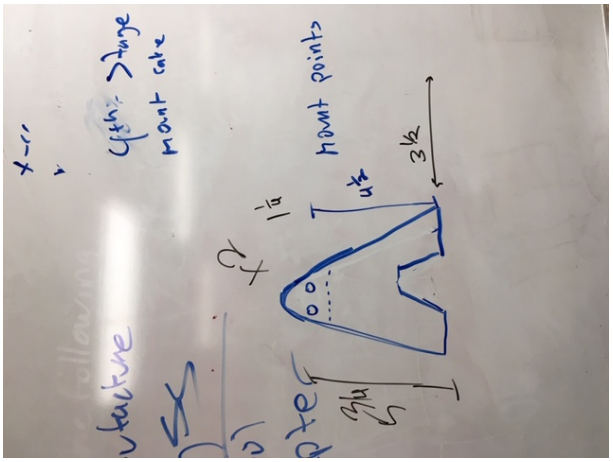


Figure 16.3: Sketch of X-rail mount on the outside drive plate

was feasible before cutting a new drive plate. Which would use valuable material and time. The tests were successful and the team is going ahead with the design.

H3: Building the pulley system for the lift

This week Ben worked on developing the pulley system for the lift. He started thinking that he might use pulleys and tetrix L Brackets that already had holes so that he didn't have to cut new ones. Ben then cut down the side on on of them to try to get the bracket as close as possible to the side of the lift. He then used some v bearings to be able to string it. Then, after he adjusted the design, he made two more to put at the bottom and one to go at the top.

H4: Deconstructing Robot

After all the sub-systems had been tested Oren, Ben, Shawn, and Adian took apart the main portion of the robot, as to prepare for the coming week. They made sure to organize and keep all the necessary parts to make a smooth re-construction of version 2.0. There was a considerable amount of modifications made to fully built subsystems to make sure they would fit on the new drive-train. One of the changes that had to be made was to the intakes rollers length, due to the inner dimension of the aluminum plates increasing, which in turn increased the length of the rollers.

Figure 16.4: Prototype outside drive plate mount

Week 17

S1: Characterize and tune the drive train

Before the robot can be able to follow paths during auto, the control loops must be tuned. There are two cascading control loops that control the robot's position. The first is the PIDF velocity loop on bard the Rev Hub. Changes to the rev hub firmware this year mean that rather than commanding velocity in some unit-less number between -1 and 1, velocity can actually be commanded, in radians per second. This takes some of the complication out of the control system, because the constant that relates that unitless number to an actual velocity does not need to be found, but the on board loop still has to be turned.

The first step is to turn off the feedback components of the controller, by setting kP, kI, and kD all to zero. To tune the feedforward constant, kF, Kelly first set it to 1. Then he wrote a opMode that gradually ramped the commanded velocity up while the robot hurtled down a long strip of field tiles, shepherded by Emma. The opMode logged both the commanded velocity and the actual velocity reported by the Rev Hub. The velocities were recorded in radians per second rather than inches per second, because the velocity loop works with individual motors rather than the velocity of the robot its self. The measured data is shown in figure 20.1. While it is obvious that the real line of best fit should be of the form $y = ax + b$, that would not translate to a usable kF for the velocity control loop, as the Rev Hub has no option to set a $kStatic$. Because of this, static friction will have to be ignored, and the regression calculated in the form $y = ax$, which would make $kF = \frac{1}{a}$. This form of the regression still leaves the coefficient of determination sufficiently high, and even if there is some leftover error, the feedback portion of the loop will take care of that. After preforming the regression, kF is found to be 12.579. If the mass of the robot significantly changes, this process will have to be preformed again to find the new coefficients.

The next step in the tuning process is to determine the feedback components of the velocity control loop, but this is made easier by a reasonably tuned feedforward component. It is important to tune the feedback coefficients, because the default tunings are meant for an unloaded motor. The motors respond well then just a bare shaft is spinning, but the coefficients are much too low once they are moving around the mass of a 18kg robot. First kP is tuned, by increasing it until the loop becomes unstable, and then backing off a bit. Then kD is added to dampen the rest of the oscillations, and kI to take up any remaining constant error. To tune these coefficients the robot follows a series of motion profiles to travel in a square in the middle of the field while Kelly watches on the dashboard, adjusting the coefficients as needed. Because changing the coefficients on the Rev Hubs requires stopping the motors and re-setting their run mode, the coefficients can only be updated between paths.

After finding a reasonable tuning for the velocity, Kelly tuned the positional PID. The same process was repeated, driving the robot in a square around the middle of the field, but this time the positional coefficients were tuned. The corrections determined by the three positional PID loops, axial, lateral, and heading, are added to the velocity calculated from the trajectory to determine the velocity commanded to the drivetrain, which is turned into the velocity commanded to each motor. Because the velocity loops reduced the error quite a bit to begin with, only P terms were needed on the positional loops.

With the control loops tuned, the robot was now ready to begin executing paths.

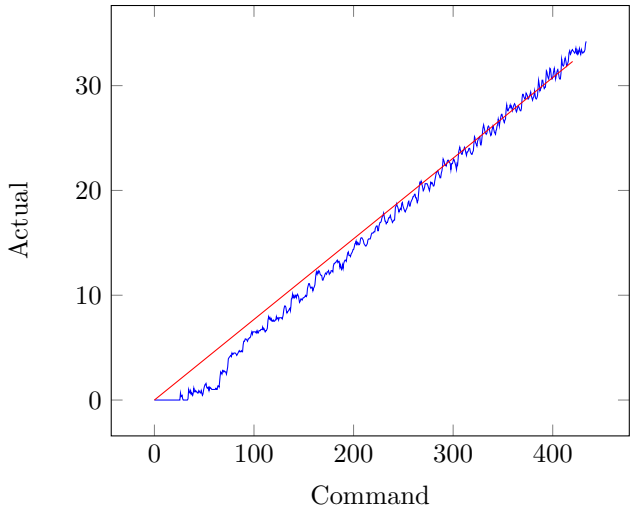


Figure 17.1: Actual vs. Commanded velocity, with line of best fit

H1: X-Rail Mounting

This week the inner drive plates were cut. However, since the outer plates were not cut, the X-rail could not be mounted. As such, a stop-gap method was used to mount the X-Rail. While technically the X-rail was mounted before on Tetrix, it hardly worked. This meant that if the team wanted to have a functioning robot, they would need to create one that worked better. ACME wanted to meet a deadline for having the robot able functioning so it was important to make this stop-gap solution. Jon was assigned to create this. The easiest way to do this seemed to be by drilling mounting holes into two sheets of aluminum and folding them. It was important that the mounting solution be simple and quick to make because the deadline that the team needed to have a functioning robot by was fast approaching.

H2: Cutting Drive plates

For the cutting processed we decided to use a CNC laguna, for its precision and accuracy rather than by hand or CNC plasma cutter.The job set up was pretty simple since we had been walked through it the previous time. The set up entailed us, to cut out two pockets two feet by 1 foot so as to allow our aluminum when screwed down, to rest flush with the top of the spoil board. We then orientated the board so that the bottom left corner of the board was at the X & Y zeros, to allow for an easy tool path set up in Aspire. As an extra precaution in Aspire, tabs were added to the material that would be completely cut out, so that the router wouldn't throw the loose aluminum pieces. Since this plate was modeled to have precise mounting holes, it required us to make two tool changes, not hard to do in the program, but required us to learn a new step, Tool Touch offs. When we had finished the program portion of the set up we moved back to the Laguna, were we started with setting our X,Y & Z zero's. The Z zero had to be set to the bottom of the pocket because the Laguna cuts from the bed surface not from the material. Those zeros are very important to get as accurately as possible, because the tool switches and touch offs are based on the

first bit and the said zeros. After we made sure everything was set up to the best of our ability, we began cutting, about half through the first pass on the profile the 1/4 O-flute bit snapped due to depth and side impact on the bit. We quickly switched the bit and reset all our zeros and tool touch offs, then resumed the cut. We later talked to a professional, who told us that the bit we were using was for 1/16th inch aluminum not 1/8th, so we ended up buying a new one to replace theirs.

Week 18

S1: Parse the JSON Response

With the prescouter, the response from the API is in JSON, which is very difficult to read as it is all strung together. Since this app is specifically made to make prescouting easier, it was important to parse the information. As Emma had no experience parsing before, she spent some time researching the basics before attempting to write the code. She decided to use a JSONArray to parse the data since the response was an array. This array consists of the JSON response from the GET. Then, Emma made JSONObjects for each key then used the array to get the value. Finally she printed the result to the console. Here is the JSONArray that she used to parse the data.

```
String jsonResponse = response.toString();
JSONArray array = new JSONArray(jsonResponse);
JSONObject teamKey = (JSONObject) array.get(0);
System.out.println("Team Key: " + teamKey.get("team_key"));
```

Each key in the array has a separate JSONObject in order to parse it. As this is a lot of code to have in the Main class of her application, next Emma has decided to make each command she will use into a different class and then call it in Main when she needs that specific data parsed.

H1: X-rail mounting work around

With the new outside plates now cut, the team could begin the final stages of fabricating the robot. There was a problem however, the part that was supposed to be bent over to mount the X-rail from the outside drive plate had broken off during the bending process. This caused the team to learn the hard way that aluminum is not suited for 90 degree bends. To overcome this problem, Jon bolted two Tetrix brackets to the inside of the outside drive plate. This acted almost exactly as the originally mounts would have. Jon did it this way because it was quick, simple, and functional. With a "robot done" deadline fast approaching, this was likely the best solution. While this was neither an elegant or permanent solution, it was at least functional.

H2: Re-Constructing Robot

After the majority of the newly fabricated parts were completed, we began building version 2.0. The build process the second time around was much faster and went very smooth. We completed the reconstruction with in two meetings, and then began the long and tedious process of wiring everything up. As a team we have learned from past experience that wiring and organization is very important for a functional robot. Organization allows for easy fixes when necessary and creates no confusion when working through software and hardware issues, this is an area that in the past has been neglected. But we have dedicated time and effort to make sure that wiring is cosmetically pleasing and concise.

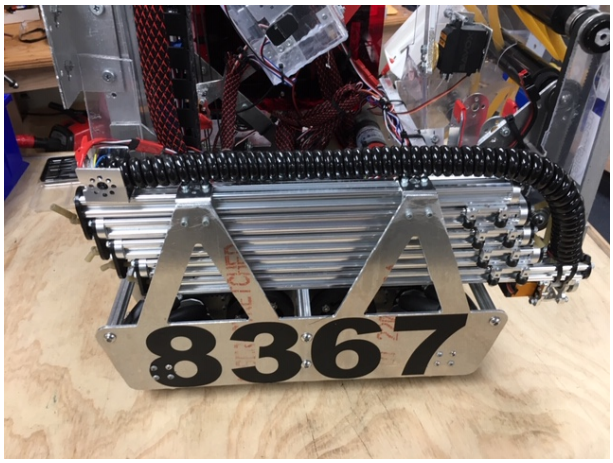
Mount.JPG 

Figure 18.1: X-Rail Bracket Mounts

B1: Writing Tournament Presentation

The writing tournament is an event in Nevada county for 6th and 7th graders. There are three parts to the tournament, the response to a presentation, the creative writing piece, and letter writing. They always have quest speakers and this year, ACME was asked to come.

Emma and Kelly prepared a half an hour presentation about ACME, FIRST, and their 2017-18 season. The presentation basically told the story of their journey to Worlds with important information like how their design process and how software works mixed in. The students participating in the event were given a prompt to respond to. The prompt this year was "What makes a robotics team successful?" ACME made sure to include multiple ways for this question to be answered. For example, a robotics team can be successful by being Gracious Professionals, but also by doing well during competition.

The presentation went off without a hitch and the kids seemed to enjoy it as well. The participants were given a chance to ask Kelly and Emma questions after they were done with the presentation, and had several in depth questions that needed answering. After that, the students wrote their essays, which were later graded by a group of teacher judges. ACME got a chance to read the essays and include some of the winning ones in **Appendix D of the Business Notebook**.

This presentation was a great way to connect with more students in the community, especially those that enjoy writing and could be potential recruits to facilitate the engineering and business notebooks in the future.

Week 19

S1: Introduce OOP to the Prescouter

In order to simplify the code in her Main class, Emma decided to use Object-Oriented Programming methods to call the different commands she intends to use while prescouting. She knew what object-oriented programming was, but decided to refresh her memory via the internet to make absolutely certain she understood the concept.

After doing that, Emma picked out which commands from the API she wanted to use. She decided that she wanted to be able to get information about the team, their match performance, the awards they had won, the results of their tournaments, and the events that they attended. Emma thought that these commands would yield the most information about the team she would be prescouting. She made a class for each command at set to work. Emma used the exact same method to parsing the JSON response in each class and also set it up so that they could be easily called in the Main. After a lot of tweaking, Emma got it working. It is now very easy to call commands in Main and her code is a lot cleaner. It is very clear as to why OOP is very useful and an important tool to use while programming.

H1: Lift Cross brace and LED side plates

Most of the lift had been completed and one of the only things left was to create the cross brace at the top of the lift. The cross brace was necessary because without it the entire lift would wobble five to ten degrees when at max height. While Aidan created the brackets on the side of the lift to hold the cross brace, Aidan, Kelly and Oren discussed what they wanted to make the cross brace to be made out of. They decided that instead of just the metal plate running across they wanted to make the cross brace out of cast acrylic so that they could engrave patterns into the acrylic and light it up with LED's.They decided to do this with the acrylic instead of poly-carbonate because the poly-carbonate would melt a little when engraved. They then realized that they could also create similar illuminated plates on the side of the robot. However before these plates could be cut the designs needed to be created. Since this might take a few days, Aidan created a temporary cross brace that would serve until then.

H2: Adjusting the angle of the diverter cartridge

When most of the intake had been completed, Kelly, Aidan and Oren decided to fully test the diverter and lift. One thing they realized is that if there was a cube in the back of the diverter, the cube would not gain enough speed to roll out. They decided that the angle of the diverter needed to be increased to allow for this. Aidan fixed this problem by slightly cutting into the arm from the diverter to the servo which allowed the diverter to tilt farther as shown in figure 19.1. Aidan tested this and the cube gained sufficient speed with the new angle.

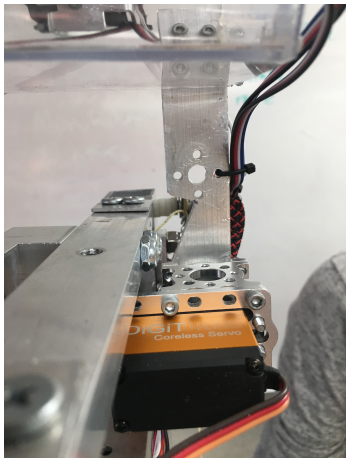


Figure 19.1: The cut to increase diverter angle

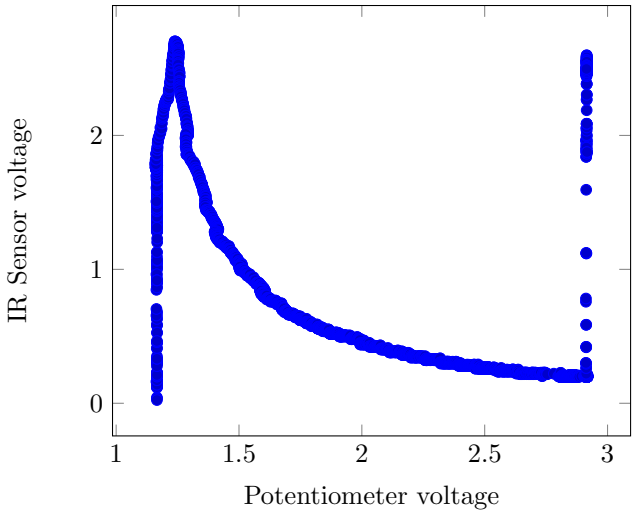


Figure 20.1: Distance reported by sensor vs actual distance

Week 20

S1: Deploy during auto

Kelly worked on deploying from the lander during auto. Because the exact position at at which the robot is initialized may vary, using the encoders would be an inconsistent way to deploy. To consistently determine the robot's distance from the ground, Kelly decided to use a IR proximity sensor mounted at the bottom of the robot. After mounting the sensor and verifying that the robot's full range of travel would be encompassed by the sensor's usable range, Kelly took the sensor off to linearize. Because it is an IR sensor, it measures distance by the intensity of reflected IR light. This means that there is a nonlinear relationship between the voltage returned by the sensor and the distance between the sensor and the ground. Before a loop can sucessfully be closed around this sensor, the reading must be linearized. Attempting to use a PID loop with a non-linear sensor yields different responses at different locations within the sensor's range. and can lead to either instability or inaccuracy. To determine this relationship Kelly and Aidan built a test rig that contained both the IR sensor and a linear potentiometer. The linear potentiometer would return a voltage proportional to the displacement of the stylus along its length. By logging the voltage returned by both the sensors, and offsetting the voltage returned by the potentiometer so that 0 corresponded to the IR sensor being in contact with the ground, Kelly found the relationship between voltage and distance. This distance is still unit-less, but the important part is that it is linear.

After removing the unusable data at either end and preforming a regression, Kelly found the relationship to be $d = 0.3928v^{-1.66868}$.

Kelly then used this sensor to close a PID loop, consistently lowering the robot to the ground each time.

H1: New Team Marker placement

After some testing, it was determined that the current placement for the team marker was unreliable. The team marker had previously been placed high on the rear of the robot which caused a great deal of deviation in where the marker landed. Ashlin and Jon were set to this task. They figured that since the problem was that the marker was too high, they would simply put it lower on the robot. This proved to be a little tricky as there was not much space on the lower portion of the robot. The entire right side was taken up by the X-Rail and the left side had both rev hubs on it. The only place that there was space was behind the rev hubs and in front of the lift. Now that they had a place to put it, Ashlin and Jon had to figure out how it would deploy. The original method of deployment (a vertical arm the marker would fall off of when it was swung down) would not work here as there was not enough space. They then came up with the idea of a kicker that push the marker off the side of the robot. They liked this idea because it allowed the servo to sit above the marker, saving space, and it would be more reliable than the previous method. The only problem with this method was that there was nothing stopping the marker from simply falling off the robot. To mediate this problem, Ashlin cut a V shaped fork into the arm that pushed the marker off. The fork could then slide over a section of the marker, holding it in place. This solved the problem, and after some testing the new deployment method worked reliably.

H2: Wiring and Rev Hubs

After most of the subsystems were built the team could then find permanent positions for the rev hubs, battery and phone mount. The team decided to put the rev hubs over the left side wheels because that is where there was the most room and the lowest chance of them getting damaged. To do this Shawn created a poly carbonate sheet that attached to the aluminum side plates and served as a firm attachment place for the rev hubs. Oren then realized that the danger with the rev hubs being there is the chance that wires could get caught in the drive train. To prevent this he created an aluminum plate to attach over the left side drive train to prevent important wires from falling in. Once the rev hubs were mounted the next important thing was to wire all the servos, motors and sensors. The difficult part of this was how to get the wires from the diverter servos because the diverter moved around so much. Kelly fixed this problem by using a wire track that could hold the wires safely inside while matching the up and down motion of the lift. The wires still needed to be run to the other side of the robot and the intake shield attachment bar served as an ideal wire trail from the right side to the left. Once all the wiring was complete the battery and power button were mounted in easily accessible locations and a temporary phone mount was created.

Figure 20.2: Marker Placer

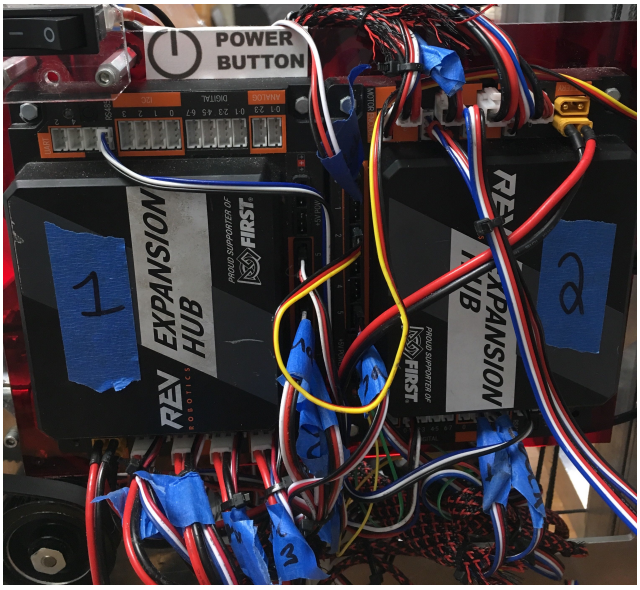


Figure 20.3: Wiring

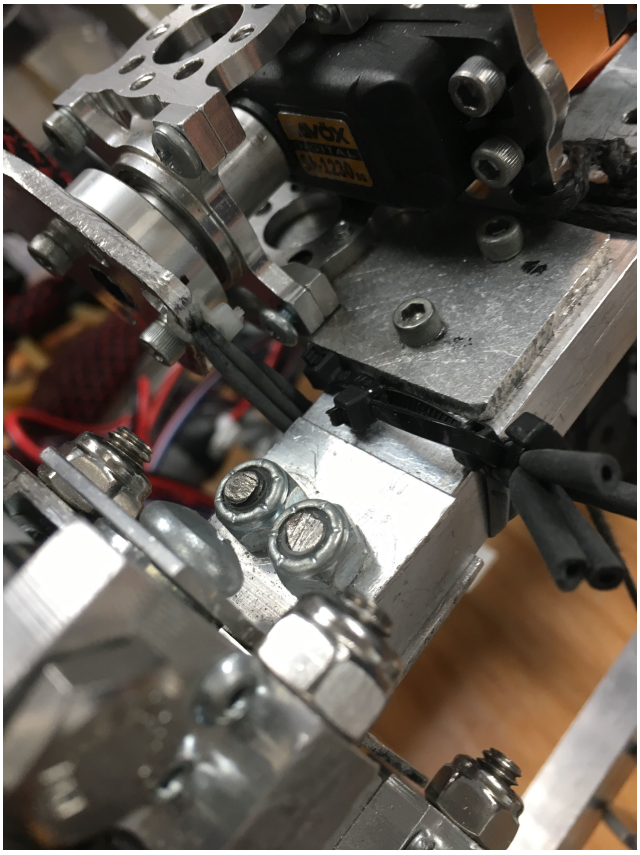


Figure 21.1: Surgical Tubing

Week 21

H1: Cartridge Servo

While Aidan and Kelly were testing the subsystems they saw that the cartridge servo was having trouble doing its full rotation. The servo especially had a problem when there was two cubes in the diverter because that is when it was carrying the most weight. The servo seemed to have trouble with the up part of its dump rotation as well as the up part of its reverse rotation. This caused a problem because it meant that whatever was used to help the servo needed to help it in both directions. Kelly and Aidan solved this by attaching a quadrupled up piece of surgical tubing to a stationary piece of the carriage and the other side to the servo horn. They connected it so that when the carriage was at its balancing point the surgical tubing had the least amount of tension. They did this so that when rotated farther clockwise then its balancing point the surgical tubing would help the servo in its reverse rotation and when rotated farther counterclockwise then its balancing point the tubing would help in the dump rotation. This helped the servo and allowed it to rotate in both directions with little difficulty.

H2: Lift Skids

With the lift now in working order, the team conducted some tests with it to see if it was up to snuff. The team quickly found that the lift didn't quite have the robot hanging the full four inches off the floor. The team put Aidan, Ashlin, and Jon on the job to fix this issue. They found that the robot was angled in such a way that the front was lower than the back. This meant that even though the back of the robot was four inches off the ground, the front was not. This was because the top of the lift stuck out farther (because of the hook) than the bottom. The trio thought they could add a plate to the back of the robot to alleviate this problem. They thought if they put a plate that stuck out as far as the hook, the robot would be flush with the lander along the entire robot. This would also make sure none of the lift or back of the robot would get caught on the lander and would just overall smooth the lifting and lowering process. They realized however that there was no where to put the plate that would interfere with the lift. All the available attachment points either moved with the lift or were too close to the 18 inch limit. This is when the team got the idea for skids. These would essentially act as skis for the robot to slide up the lander with. These were also smaller and less obtrusive than the plate. They decided to construct them out of poly-carb because of their flexibility (meaning it would be hard to break them) and because they would be easier to shape.

B1: Re-Organize the Outreach Section

ACME does a lot of outreach and the outreach section of the BN was looking a little bit disorganized. So, Kelly, Emma and Stephanie talked about how it could be better. They decided they needed clearer categories of outreach and came up with the following sections:

- Community Events
- FIRST Connections
- Team Building

They also realized that it would be good to explain what types of outreach events fell under each category. Also, instead of listing a bunch of future events they were planning at the end of the outreach section, those events could be split up by category and put under the explanations.

Emma took on the job of reorganizing. While in the process, she realized that Team Building needed to be split into two different section in order for it to make more sense. Before doing this, this section encompassed all of the events that helped build up the program in the county as well as events that the team does for fun. Thus, Emma created another section called Program Building, which housed all of the events that helped create a sustainable program in the community. Emma also added a "Contents" section to the front page describing each category. She thought that it would be good to list what was in the category so that the judges could find things that seemed interesting to them. She made sure to order the Contents in the exact same order in which the individual events were listed for added easy accessibility.

When finished, the outreach section was much more organized than it was before. It was easy to find things and the categories made sense for the events that were in them.

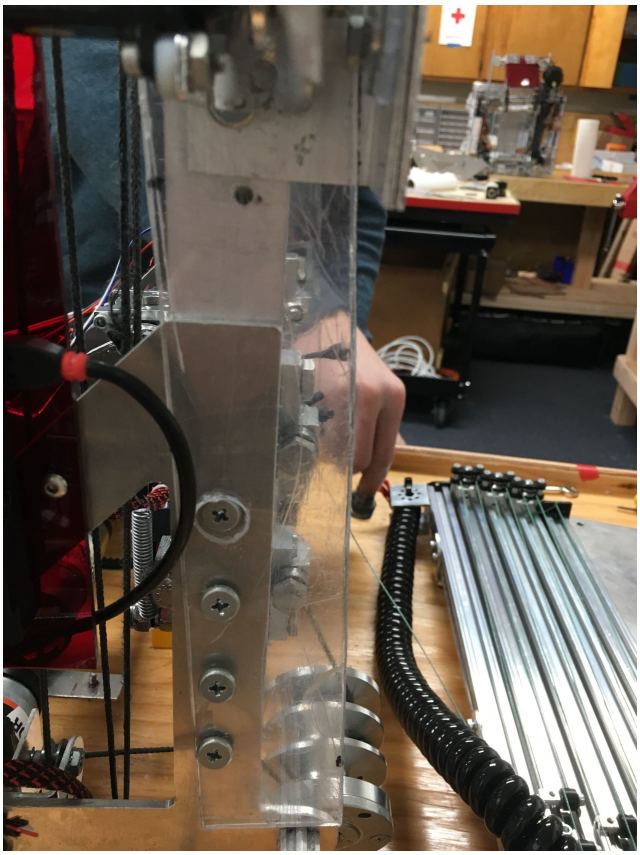


Figure 21.2: Lift Skids

Week 22

S1: Finish the console version of the PreScouter

This week, Emma finished the console version of the PreScouter. After calling each request class in Main and making sure that they were all working, Emma set about writing the code that would allow the user to make specific requests with ease. She decided to use a series of if else statements to accomplish this. First, the console asks the user what specific request they would like to make. There are only five requests a user can make because that is the only data that they need from the API. After making a request, Emma uses an if else function to prompt the user for the rest of the information needed in order to complete the URL to send to the API. Then, with the information from the user, Emma pieces together the URL and makes the GET request.

At the end of the sendGet() method, Emma used another if else function to print out the data. She decided to use another if else statement because it was easy to use the request from the beginning of the program in order to call the specific class that parsed the data. After she got that all working, the console version of the PreScouter was complete.

S2: Design double sampling paths

While supporting ARES at one of their qualifiers, Kelly saw that many teams either had no auto, or had an auto that only deployed from the lander. Kelly decided that this would make a auto that could disturb both of the gold minerals would be really good to have. In the scenario where an alliance partner did not leave the lander during auto, the location they started at would not matter, so these paths would only have to be designed for ACME's prefered start location, next to the crater. After experimenting by pushing the robot around, Kelly found that there was not enough room between the parked robot and the sampling field to approach the sampling field from the lander side, so the robot would have to first enter the depot, and then push the gold mineral toward the parked robot. This also happened to be an easier modification to make, because it only required duplicating the depot waypoint where the robot dropped off its marker, and adding a waypoint between them to disturb the gold mineral. Kelly tested the paths and found that they worked!

H1: Lift String

After most of the robot was completed Kelly and Aidan ran a few practice matches to make sure that everything was working properly. One thing that kept happening was the pull down string on the lift kept breaking right in the middle. This showed that the string evidently was not strong enough. The string had a 200 lb rating but evidently the impulse of the robot when going up or down was much higher than this. Kelly decided that the team should order some higher rated string to ensure that this wouldn't happen again. The team ordered a 900 lb rated string from amazon that was 3 mm thick. The team didn't know if the change in thickness from 1 mm to 3 mm was going to cause problems so Aidan used some left over rev slides string, that was also 3 mm thick with unknown rating, to test for adverse effects because of the change in thickness. Because of the thickness of the string more slack developed when the lift raised which caused the problem of the string jumping off the spool a few times. Aidan fixed this by wrapping extra coils around the spool for the

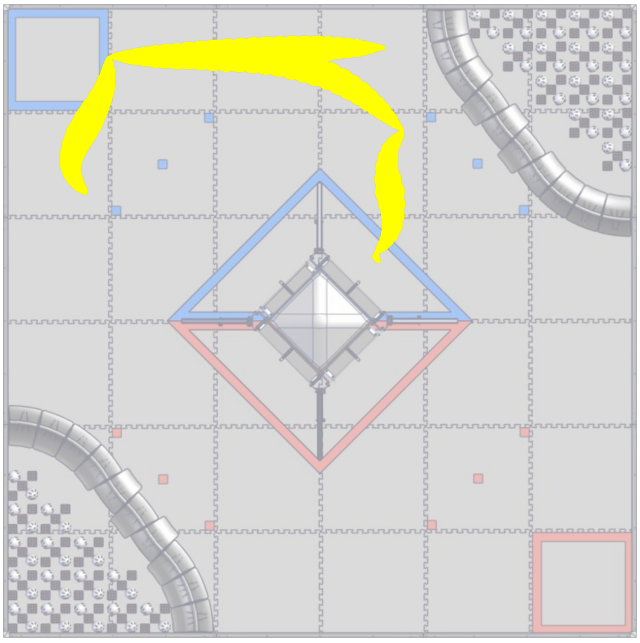


Figure 22.1: Double sampling path (line thickness corresponds to robot velocity)

pull up run which made it so less string was released per rotation for the pull down than the pull up which fixed the problem. When the new string arrived the team now knew how to use it without problem.

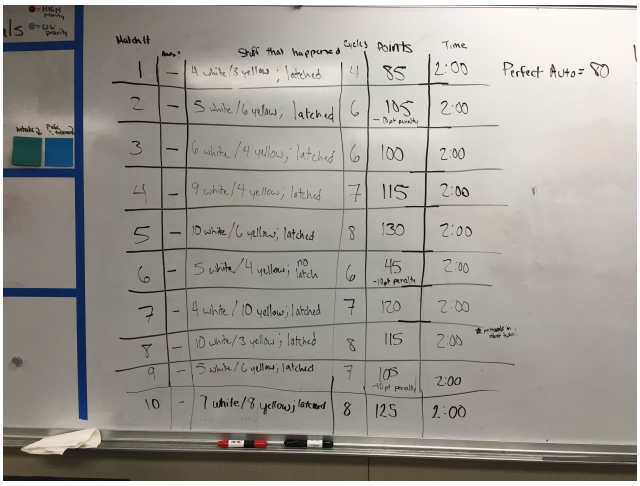


Figure 23.1: Score Keeping Chart

Week 23

H1: Drive Practice

Over the week, the drive team ran practice matches to practice their driving skills. The drive team consists of Shawn and Aidan as the drivers and Emma as the drive coach. At the beginning of the week, they just ran the teleop part of a match, using a real FTC timer that they found on YouTube. They kept track of their scores and performance on the whiteboard, as seen in fig 23.1. For their first round of drive practice, they were averaging about 120 points during just teleop.

H2: Dumper Arm Replication

During the week, Ben and Shawn were tasked with replicated the dump arm due to it bending during matches. This was a problem so Ben and Shawn decided to make more. They ended up tracing the original profile and then cut out the holes with a drill press. Then, they used a hacksaw to cut out the shape and then filed it.

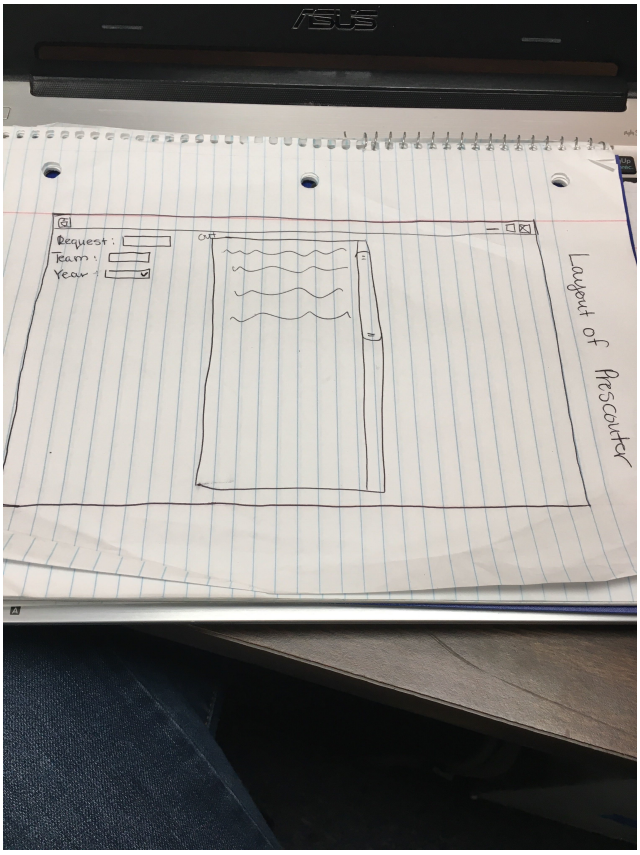


Figure 24.1: Sketch of what UI will look like

Week 24

S1: Set up the UI for the pre-scouting app

Emma started working on the next part of the prescouting app which is to create a UI. Emma had some experience with creating UIs before as she worked on the scouting app last year. First, she sketched out the design of the UI and everything she would need to display the data, as you can see in Fig 24.1.

Next, Emma set to work identifying which components she needed to use to complete the UI. She decided for the request and the team number, she would use TextFields. For the year, she would use a Choice component that would allow the user to select a preset year. She used a Button for the submit button. Finally, she decided to make a Panel on top of her Frame in order to section off the TextArea were the data would be printed from the rest of the screen.

Now, Emma needed to go about making this UI a reality and actually function properly. She began by setting up the aesthetic parts of the UI (Frame, TextFields, etc). Then she

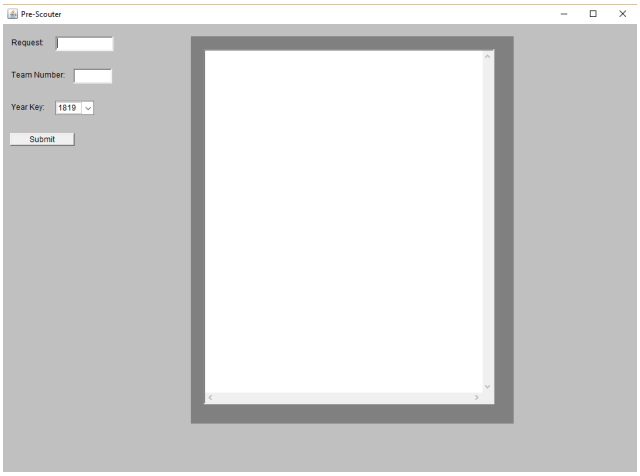


Figure 24.2: Design of UI

started to figure out how to get the submit button to actually do something. She decided to use an ActionListener for this. An ActionListener is how you tell in Java f a button has been pressed or not. If it has, an action occurs after that. That action that takes place after Emma's submit button is pressed is an if else statement figures out what is in the request TextField and what is in the team number field in order to piece together the URL. She managed to get that working, and is now trying to figure out how to send the GET request once the button has been pressed. Above is the current design the UI, as seen in Fig 24.2.

S2: Write tracking omni code

To use the omnidirectional tracking wheels installed on the robot Kelly needed to write software to translate their movements into robot movements. The kinematics to transform a relative robot pose update to a field pose up date could be reused from the existing localization code that used the drive motors, so the task was to translate tracking wheel deltas and a heading delta reported by the imu into a robot pose delta.

Because the drivetrain is holonomic, software assumes that the robot moves with constant positional and heading velocity each update cycle, rather than the traditional twists used by differential drives. After drawing a diagram of the paths taken by two tracking wheels during a robot update, Kelly decided that the best course of action would be to first determine what part of each tracking omni delta was accounted for by the heading delta, and then transform the resulting vector from tracking omni space into robot space.

Accounting for the portion of a tracking omni delta caused by a heading delta is relatively simple. Given a heading delta θ, the linear distance reported by a tracking omni will be $r\theta \sin \alpha$, where r is the distance from the tracking omni to the center of rotation of the robot and α is the angle between the radius and the axis of the tracking omni. While this value could be theoretically determined, Kelly doubted it would be accurate enough to base all robot localization off of, so they decided to determine it empirically. Because the value would be determined empirically, a new coefficient, k, can be defined to equal $r \sin \alpha$, eliminating the need to empirically determine more than one constant.

Kelly wrote an opmode to turn the robot a set angle, and then measure the distance

reported by each tracking omni. By dividing the distance reported by the heading delta reported by the imu, they empirically determined k. They found that k was equal to 3500 ticks per radian for the axial tracking wheel, and zero for the lateral wheel.

The next step was to convert the two distances traveled by the tracking wheels into the robot coordinate system. This is a very simple problem if the tracking wheels are orthogonal to the robot, one of them is the x component and one of them is the y, and is still very simple if they are orthogonal to each other but rotated relative to the robot, because the vector representing their movement can just be rotated by their angle relative to the robot. However, the general case, for two tracking wheels oriented in arbitrary directions, proved to be much more difficult.

After a couple naive approaches that involved rotating vectors representing the vectors representing each wheel individually, and constructing equations that passed and were tangent to the vectors and then solving the system, Kelly came to the realization that the same general concepts used in all holonomic wheels could be applied. Most importantly, the dot product of a vector representing robot movement and a unit vector pointing in the direction of the rollers of a wheel is equal to the distance traveled by the rollers. If $\vec{r}$ is the robot position delta, and $\hat{a}$ and $\hat{b}$ are the unit vectors in the direction of the a and b rollers respectively, then $\vec{r} \cdot \hat{a} = |\vec{a}|$ and $\vec{r} \cdot \hat{b} = |\vec{b}|$. By rewriting this as matrix multiplication the following equation results:

$$\begin{bmatrix} |\vec{a}| & |\vec{b}| \end{bmatrix}^\top = \begin{bmatrix} \hat{a_x} & \hat{a_y} \\ \hat{bx} & \hat{b_y} \end{bmatrix} \vec{r}^\top$$

and after solving for $\vec{r}$:

$$\vec{r}^\top = \begin{bmatrix} \hat{a_x} & \hat{a_y} \\ \hat{bx} & \hat{b_y} \end{bmatrix}^{-1} \begin{bmatrix} |\vec{a}| & |\vec{b}| \end{bmatrix}^\top$$

To actually compute this, the inverse of the unit vectors are derived from angles representing the direction the rollers point, and the inverse matrix is pre-computed, and then each row represented as a vector, so the resulting vector can be represented as two dot products. This small utility class encapsulates this math:

```
public class TrackingWheelKinematics {

    private Vector2d r0, r1;

    public TrackingWheelKinematics (double a0, double a1) {
        if (a0 == a1) throw new IllegalArgumentException("you can't point them in
            the same direction silly!!!");
        double a = Math.cos(a0);
        double b = Math.sin(a0);
        double c = Math.cos(a1);
        double d = Math.sin(a1);
        double determinant = a*d - b*c;
        r0 = new Vector2d(d, -b).div(determinant);
        r1 = new Vector2d(-c, a).div(determinant);
    }

    public Vector2d transform (Vector2d dist) {
```

```
        return new Vector2d(r0.dot(dist), r1.dot(dist));
    }

}
```

H1: Investigate Intake Whips

After the Napa tournament one evident problem was that the intake was to slow. One way the team thought that this could be fixed is if the stiffness of the whips was increased. Aidan and Ashlin were designated the task of investigating different whip materials. They wanted a material that was stiff enough that it could easily hit the balls and cubes up and through the intake, but they didn't want it to be too stiff that it would cause damage to the poly carbonate shield. Aidan went to the hardware store and bought a couple different types of gas tubing of varying stiffness. The team then tested the whips and determined that to maximize the intake the thicker whips should be put with three in the middle roller as shown in figure 24.3 and on the outer four whips for the crater roller. Aidan also determined that if he placed the most outside whips 30 degrees ahead of the rest this would help directing the balls and cubes into the center of the whips.

H2: Lift Version 2.0

After the Napa tournament, it was evident that there was some issues and restrictions with the current design of the lift. The most notable ones being the inconstant lift height through out a match, and the over all lack of tolerances due to the bearing blocks assembling process. With both of these issues so closely affecting the over all performance, the hardware team set out with two plans to fix the problems. Oren set out re-CADing, and ordering new parts for a second lift, that would be able to easily be switched out for the current one when the time came. Meanwhile the other hardware members made parts and improvements to the current lift as a precaution in case the new lift wasn't made in time for NorCals.

H3: Change the Lift String Standoffs

One recurring problem that was noticed prior and during the Napa tournament was that the pull up run length was very inconsistent. This was because the standoffs that held the pull up run bearings continued to bend which changed the overall length. Aidan was designated the task of creating more reliable standoffs. He decided that the best way to do this would be to manufacture custom pieces for each bearing location. He completed CAD drawing of each desired piece as shown in figure 24.4 and 24.5 and planned to cut the parts out the next week.

B1: Napa Tournament

ACME attended the Napa Qualifier. This was their last qualifying tournament, and since they had already qualified for the Northern California Regionals, this tournament was sort of a practice tournament and they didn't feel completely stressed out about matches.

They practiced judging bright and early and set up our pit area. This tournament they wanted to test out a new pre-scouting technique where a few scouters went around and asked their alliance partners to draw out their autonomous paths on a field diagram they

Figure 24.3: Intake Whips

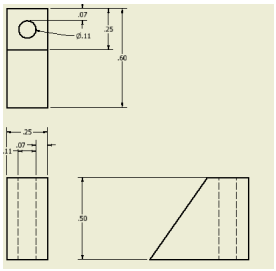


Figure 24.4: Pull Up Block 1

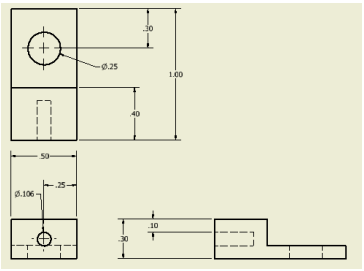


Figure 24.5: Pull Up Block 2

had printed out. This worked quite well, as it allowed ACME to run different auto paths based on their partner's capabilities and paths.

The team's qualifying matches went very well! They won every one and placed third overall. Since they didn't need to qualify, they weren't interested in being alliance captain, so they decided to let the first place team pick them for their alliance. Although ACME went out in the semi-finals, they had a great time competing.

Another big accomplishment for this tournament was that every single group of judges came and talked to the team. This meant that they made the shortlist for every award. This was a first and they were very proud that their judging presentation made such a good impression on the judges.

ACME ended the day by winning the Control award for their software. They were really happy and had a fun and exciting day!

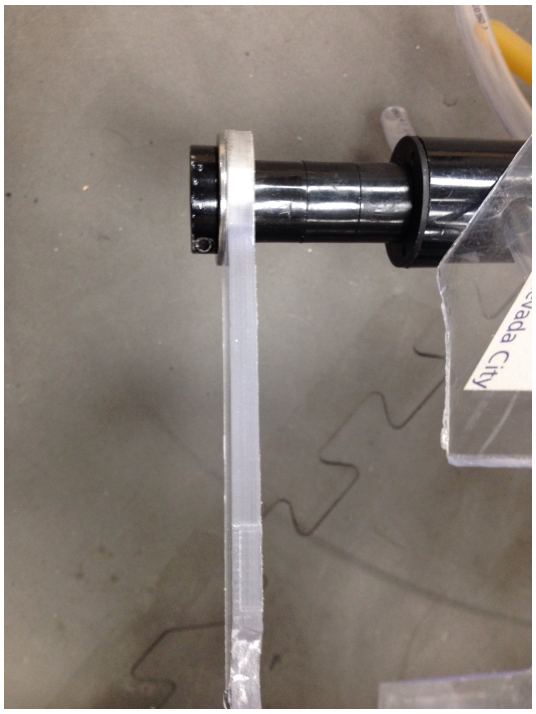


Figure 25.1: New Arm

Week 25

H1: Increasing the Intake arm Length and Thickness

Ashlin and Aidan realized that another way to increase the speed and effectiveness of the intake was to increase the reach of the third intake roller (the one that flops down into the crater). They decided that if the whips were an inch longer this would significantly increase the reach of the whips while not slowing it down. In order for this to work the roller arm need to be increase an inch in length and positioned an inch higher so that the whips did not hit the ground or crater to much that they might cause unnecessary friction. New arms were cut out however a belt for the new length did not exist. So the team decided to not increase the arm or whip length, but did increase the arm thickness as shown in figure 25.1 They did this because they knew from experience that the arms often had to absorb significant impact from collisions and previous arms had snapped because of this. The team believed that increasing the thickness would result in more durability. The new arms were cut and put on the robot, where they were tested and found to be very strong and durable.

H2: Change lift gear ratio

After the Napa tournament, the team brainstormed ways to decrease the time it took to score one pair of minerals. Because there is such a short distance between the crater and the lander, it takes barely any time to travel between them. This meant that the drive team would often be waiting for the lift to reach the top before they could score. The lift can be

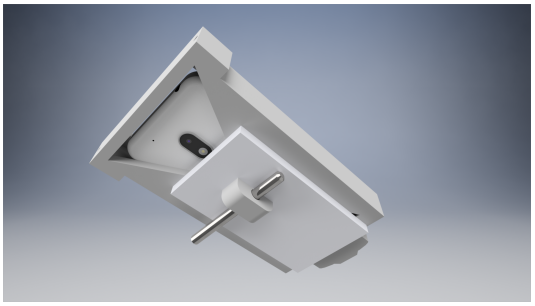


Figure 25.2: Phone Case Assembled

made faster by changing the gear ratio of the lift gearbox, however this comes at the expense of torque and could potentially effect the ability of the robot to hang from the lander. The initial gear ratio had been picked with the conservative safety factors of a 20kg robot under an acceleration of 2g, and was more than enough to lift the robot. Kelly re-calculated the ratio with reduced safety factors of a 20kg robot and a 1g acceleration. This results in a force of 196 N, and when applied to the 1 in diameter spool this results in 2.483 Nm of torque at the gearbox output shaft. With the current reduction of 25:1, this resulted in 0.0975 Nm of torque at the motor output shaft Kelly found that by going to a 18:1 ratio, the torque at the motor output shaft would be .135Nm, which was under the stall torque of .15Nm. By assuming a new maximum velocity of 20 m/s, instead of the old 14 m/s, and recalculating the motion profiles for the lift, Kelly found that the lift would reach the top in 1.8s, rather than 2.6. The team ordered a 3:1 and 6:1 stage for the lift gearbox, and swapped it them out for the current reductions. After the modifications, they verified that the lift indeed did run much faster, while still being able to raise the robot off the ground.

H3: CAD and Fabricate Lift Version 2.0

Oren finished the CAD over the weekend to allow enough time for the fabricating process, the Largest improvement and change came in the form of the bearing blocks. The blocks were redesigned to be machined out of one bar of aluminum to increase the overall tolerances by .01 and allow for a more precise assembly. Due to the form factor and the assembly of the lift systems inner box frame; it required 2 different styles of bearing blocks (PIC1) (PIC2). Another change that was sought after was the ability to find someone that could weld the inner frame to again strengthen and increase the tolerances. After asking around Oren ended up finding two potential welders, unfortunately the first welder Oren contacted was unable to help due to having a very badly broken leg. But the second contact Chris Kelly (Local Bike Welder) was able to help us, but was not available for a week. Oren then set out to machine all the parts at a sponsoring company GSS, with the teams mentor Dan Philips. The machining process took 8 hours over a stint of 2 days to produce all the parts and cut the aluminum tubing, to prep for the welding process in the following week.

H4: New Phone case and Mounting System

The old phone case suffered from constant DC's and the Micro USB port on the phones being worn out, leading to more DC's. Ashlin was tasked with making a new phone case

Figure 25.3: Oren machining parts at GSS with Dan

that would house the phone permanently and provide a secure connection to the rev hubs. As well as be easily removable from the robot. To do this the case was designed to have a pin system that would allow for removal of the phone in seconds while keeping it secure and positioned correctly for sampling. In Figure 25.2 the top of the case has a bar that is secured with two bolts so that the phone can be removed if needed.

Week 26

S1: Figure out how to send the GET request after a button press

Emma spent a lot of time trying to figure out how to get the GET request to wait until the user had pressed the submit button. This turned out to be a very frustrating problem, and one that took a lot of time to solve. At first, she tried to use an if statement to check whether or not the button was pressed. However, this sent her down a long rabbit hole of making the button static, because the if statement would only work in the main if the argument was static. Then she switched from a regular button to a JButton, and then to a JToggleButton, to try to use the same if statment solution. Then, after days of agony of trying to get it to work, Emma finally realized that she was being silly and could just use another ActionListener to tell when the button was pressed so that it would send the GET request. It didn't just work like magic, however, she still ran into a problem where the http.sendGet() was throwing and exception. To combat this, Emma used a try statement that caught the exception. Luckily, this worked and Emma could move on to the next challenge which is making the GET request print out to the TextArea.

S2: Finish the Pre-Scouting App

Emma needed to add the finishing touches to the prescouting app in order to get it operational. In order to do this, Emma needed to figure out how to get the GET request to print the data it gathers into the TextArea. This meant that she had to edit the individual request classes in order for it to properly parse the data. Emma decided to use a StringBuilder to accomplish this as it would give her the ability to string all of the data together in one place. After doing this for each class, she switched her focus to getting the data to print to the TextArea. She did this by checking the "request" TextField for the request and then depending on what it is (using and if else statement), she prints the data into the TextArea. Below, is the code that accomplishes this:

```
if(requestField.getText().equals(team)) {
    Team team = new Team();
    team.setTeamArray(responseArray);
    dataArea.setText(team.printTeamText());
 }
```

With the prescouting app complete, Emma is now looking forward to testing it out before a tournament.

H1: Lift 2.0 Fabrication

Early in this week, Oren went to meet with the local welder to begin the process of setting up and welding both the box frame and the carriage for the new lift. He was hoping to complete the new lift by Tuesday, as to give the drivers enough time to practice, but unfortunately hit some complications and the lift wasn't completed in time. So Oren focused the rest of his time making sure the current lift was running as well as possible. Which required him to deconstruct the lift, sand down rough spots on the runners and change out broken bearing

to allow for the lift to run. After which he assembled everything and bolted it back onto the robot ready for Norcals.

H2: Attaching a Second Pull Up Run

H3: Manufacturing and Installing the New Pull Up Run Standoffs

B1: Refine the Business Notebook

There were a few things that needed improvement in the Business notebook, so Emma and Kelly set about fixing those things. Both of them needed to write up a few outreach events, including the Telestream Tech Talk and meeting up with Animatronics. Emma also needed to update a couple of goals and action plans with the progress and results that have taken place over the past couple of weeks. After fixing a couple of pictures, the BN was ready to go. ACME revisits their BN from time to time in order to keep it up to date. It is important to do this so that all data is relevant when they present their notebooks at a tournament.

Week 27

S1: Switch to Camera 2

Until now the vision processing code for detecting the location of the gold mineral in the sampling field had used an outdated API for accessing the camera on the phone. As a result, camera code was unstable, and was prone to occasional random crashes, and would sometimes freeze and stop returning frames for processing, meaning that the drive team would need to restart the app before auto could successfully be run. Kelly decided to make a few changes to resolve this. First, the old way that vision pipelines could access the camera was through a singleton class that was tied to the life cycle of the entire robot controller app. When the app started, it would begin capturing frames from the camera, and these would be sent to a `FrameGrabber` class, which would then return the most recent frame to vision pipelines when needed. Because frames were always being captured using the old API, the system had some mysterious bugs, so Kelly decided to switch to a system that only captured frames when needed. Kelly also wanted to switch to the newer Camera 2 API, as well as provide a better system for multiple vision pipelines to be run at once, which would be necessary for a crater vision pipeline to aid in intaking during auto.

The layout of the activity stayed the same, but Kelly updated the preview window to use the newer API. However, rather than the preview window being initialized when the activity started, it is initially hidden. Kelly then wrote a class `VisionCamera` that when created would retrieve the preview window from the activity layout, enable it, and attach itself a frame listener for the preview window. The `VisionCamera` also attached itself as a listener to the `OpModeManagerNotifier`, which would allow it to automatically shut itself down when the opmode that created it stopped, even in the event of a crash within the opmode that prevented normal shutdown. The `VisionCamera` also handles asynchronous loading of the opencv libraries from a package seperately installed on the robot controller phone. This reduces the time it takes to deploy to the robot controller phone, because the opencv libraries never change, and does not initialize the libraries until they are actualy needed, reducing the startup time for the main robot controller activity.

To allow vision pipelines to access the camera frames, Kelly created a `VisionPipeline` interface that allows the `VisionCamera` to send captured frames to a set of pipelines. This also allows the actual vision processing to take place in a separate thread, so it does not slow down any other opmode activities. Kelly refactored the existing pipeline that detected the gold location to use this new process.

The following listing shows the process for initializing the `VisionCamera`

```
public VisionCamera () {
    activity = AppUtil.getInstance().getActivity();
    opModeManager = OpModeManagerImpl.getOpModeManagerOfActivity(activity);
    opModeManager.registerListener(this);
    trackers = new ArrayList<>();

    final CountDownLatch countDownLatch = new CountDownLatch(1);

    final BaseLoaderCallback baseLoaderCallback = new
        BaseLoaderCallback(activity) {
        @Override
        public void onManagerConnected(int status) {
```

```
            if (status == LoaderCallbackInterface.SUCCESS) {
                Log.i(TAG, "opencv load successful");
                countDownLatch.countDown();
            } else {
                Log.e(TAG, "error loading opencv");
            }

        }
    };

    AppUtil.getInstance().runOnUiThread(() ->
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0, activity,
                baseLoaderCallback);
    });

    try {
        countDownLatch.await();
    } catch (InterruptedException e) {
        Log.e(TAG, "interrupted while loading opencv");
    }

    view = (JavaCamera2View)
         activity.findViewById(com.qualcomm.ftcrobotcontroller.R.id.cameraViewId);
    view.setCvCameraViewListener(this);
    AppUtil.getInstance().runOnUiThread(() -> {
            view.enableView();
            view.setVisibility(View.VISIBLE);
        }
    });

}
```

H1: Cutting a New Latch Piece

H2: Lift Assembly 2.0

H3: New Team Marker Placement

B1: Attend the Northern California Regional Championships

B2: Northern California Regional Championships retrospective

B3: Plan outreach events

Week 28

S1: Refactor Auto Actions

Kelly decided to refactor how actions were handled in autonomous to make it easier to have actions be triggered when certain conditions were met, or to have actions triggered by passing waypoints in trajectories. Because of the way the robot state machine works, all mutation functions within subsystems are non-blocking, they update the state of the subsystem but they do not actualy update the hardware. For the hardware to be updated, the `Robot#update()` function must be called. This allows for easy code reuse between auto and teleop, subsystem mutator functions are called to trigger actions, either when a button is pressed during teleop or sequentialy during auto, and then the update function is repeatedly called. In auto, execution is controlled by having subsystems return whether they should currently be blocking new commands, and then waiting for all subsystems to stop blocking before starting the next operations. This provides an easy way to start several actions and then wait for them all to complete, or to preform one action after the next. For example, in the beginning of auto the first step is to call `Lift#lower()`, which puts the lift into lowering mode. Then `Robot#waitForAllSubsystems()` is called, which continually updates the robot until all subsystems cease blocking. Once the robot reaches the ground as detected by the sensor on the bottom of the robot, the lift switches into the finding latch mode, which uses a hall effect sensor to move the frame into a position that allows the robot to disengage from the latch. Once the lift has disengaged from the latch, the lift stops blocking, which allows the next action in the autonomous routine to execute, which is a trajectory that rotates the robot out from the latch. After this trajectory completes, the next trajectory is started, and the lift is told to lower and calibrate, which is not a blocking action and will happen asynchronously while the rest of auto runs.

This system breaks down then a action needs to be triggered partway through another action. If it is something within a subsystem, e.g. moving a servo to a position to clear an obstacle when the lift reaches a particular height, it is easy to implement within that subsystem, but for an action to be triggered partway through a trajectory, for example to begin extending the rake before reaching the end, a larger restructuring was necessary. Kelly added an `AutoAction` interface to facilitate triggering actions. All auto actions are defined in the `AutoOpMode` class, which is an abstract class that handles things that are common to all auto routines, and are declared like such:

```
public AutoAction extendRake = () ->
    setRakePosition(robot.config.getStartLocation() == StartLocation.CRATER
           ? AutoPaths.RAKE_POSITION_CRATER
           : AutoPaths.RAKE_POSITION_DEPOT);

public AutoAction deployMarker = () -> robot.intake.deployMarker();

public AutoAction retractRake = () -> {
    robot.intake.setRakeRetractBlockingDistance(10);
    robot.intake.retractRake();
};

public AutoAction startIntake = () -> robot.intake.setIntakePower(1);
```

```
public AutoAction reverseIntake = () -> robot.intake.setIntakePower(-1);

public AutoAction stopIntake = () -> {
    robot.intake.setIntakePower(0);
    robot.intake.groundIntakererIn();
};

public AutoAction groundIntake = () -> {
    robot.intake.setIntakePower(.75);
    robot.intake.groundIntakererOut();
};

public AutoAction blockingIntake = () -> robot.intake.intake();

public AutoAction raiseLift = () -> {
    robot.lift.liftTop();
    robot.lift.setAsynch(false);
};
```

If a particular auto routine needs to change the behavior of an action, it can by overwriting the variable. When an opmode creates an instance of `AutoPathsBuilder` (the class that dynamically builds paths based on a configuration), it passes `this`, allowing the builder to add a the opmode's actions to the trajectories it builds. For example, this function constructs a set of trajectories to move the robot to its parking position. The intake is stopped after reaching the first waypoint, and intake is started when the end is reached.

```
 public ArrayList<Trajectory> toPark () {
    TrajectoryBuilder builder = getBuilder();
    if (location != GoldLocation.LEFT) builder.to(PARK_CLEAR_ONE);
    builder.to(PARK_CLEAR_TWO)
        .addActionOnCompletion(opMode.stopIntake);
        .turnTo(PI);
        .to(park())
        .addActionOnCompletion(opMode.blockingIntake);
    lastPosition = park();
    return builder.build();
}
```

This function constructs a set of paths that sample the second mineral in a double sampling routine. It demonstrates how multiple actions can be added to a single segment, and how actions can be added part way through a segment. In particular, passing a negative number to `addAction` will trigger the action the specified number of seconds before reaching the end of the trajectory. To ensure that actions are not triggered if the robot is horribly out of positoin, a trajectory will not trigger an action until both the time requirement is met, and the total tracking error is less than a specified threshold, currently at one inch.

```
public SuperArrayList<Trajectory> toSampleSecond () {
    TrajectoryBuilder builder = getBuilder();
    if (location != GoldLocation.RIGHT) builder.to(SAMPLE_SECOND_CLEAR);
    builder.to(sampleSecond());
```

```
            .addAction(-2.5, opMode.retractRake)
            .addAction(-2, opMode.groundIntake);
            .addActionOnCompletion(opMode.extendRake);
        lastPosition = sampleSecond();
        return builder.buld();
    }
```

This system allows autonomous routines to be declared very easily:

```
@Autonomous(name="doubleSample")
public class DoubleSampleAuto extends AutoOpMode {

    @Override
    protected void run() {
        this.extendRake = () -> robot.intake.goToPosition(30);

        AutoPaths paths = new AutoPaths(this, goldLocation, startLocation);

        robot.drive.setCurrentEstimatedPose(paths.start().pos());
        robot.drive.followTrajectories(paths.startToRelease());

        robot.drive.followTrajectories(paths.toSampleBoth());

        robot.intake.deployMarker();
        robot.waitForAllSubsystems();

        robot.drive.followTrajectories(paths.toSampleBothReturn());
        robot.pause(1000);
        robot.drive.followTrajectories(paths.toIntake());
    }
}
```

H1: Attaching New Lift

H2: New cross plate

B1: Marketing the Hackathon

Week 29

H1: Re-wiring Robot

H2: Changing the Marker Deployment Mechanism

Week 30

S1: Re-tune lift

Now that the new lift was on the robot, Kelly decided it was time to re-tune the lift. Because there is only one encoder between the two lift motors, it is not safe to use `RUN_USING_ENCODERS`, so the control loop needs a normal tuning, with k_v, k_a, and k_{static}. Kelly began by writing an opmode to gradualy ramp up the voltage applied to the lift motors until the lift reached either the top or the bottom, then stop the lift and revers. The opmode logged both the applied power and the actual velocity of the lift in a csv file, and because the lift was accelerating so gradualy it was safe to assume that the only acceleration on the lift was due to gravity. After running the opmode for a while and collecting a lot of data, Kelly pulled the file off the phones, and seperated them using a python script based on the direction the lift was moving, and the state of the sensor that detected the frame being at the bottom of its range of motion. The data from the lift moving down while the frame was at the bottom was super noisy, so Kelly threw it out and only considered the other three sets. The graph in figure 30.1 shows the collected data. As you can see, the slope of all three groups of data is the same, but the intercept is different.

Because the neccecary voltage to achieve a particular motion state in a linear mechanism is

$$V = \omega k_v + \alpha k_a + k_s tatic$$

and the acceleration across all the runs is constant (gravitational acceleration, $386 in/sec^2$, the slope of all the graps is $\frac{1}{k_v}$, and the intercept is a combination of k_a and k_{static}. Because k_a is dependent on the mass of the thing that is moving (and in fact can be written as $k_a = mk_f$), it is important to compare lines with constant mass when calculating $k_s tatic$. By comparing both states where both the carriage and the frame are moving (when the mass of the parts of the lift that are in motion are greatest), (shown in purple and blue in figure 30.1 k_{static} and k_a (for that mass, and by extension k_f) can be calculated. Because k_a is going to be the same for both, and k_{static} is going to have the same magnitude but opposite signs (because its sign is the same as the direction of travel), k_a was found to be the midpoint between the two intercepts, and $k_s tatic$ is one half of the distance between them. By dividing the calculated k_a by the mass of the entire lift, Kelly found k_f.

The following listing shows the way a feedforward command is actualy calculated from a desired motion state and known mass.

```
public static double K_V = 0.0643202688898;
public static double K_STATIC = 0.118777073737;
public static double K_A = 0.000127390384279;
public static double MASS_ROBOT = -20;
public static double MASS_CARTRIGE = .75;
public static double MASS_FRAME = .75;
public static double G = 386;

private double getFeedForward (MotionState state, double mass) {
    double ff = K_V * state.getV() + K_A * mass * (state.getA() - G);
    if (Math.abs(ff) > 1e-4) ff += Math.copySign(K_STATIC, ff);
    return ff;
}
```

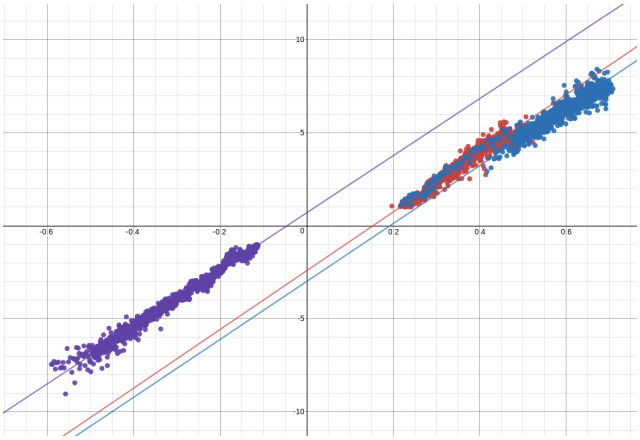


Figure 30.1: Collected lift data

This listing shows the way this function is used within the profile following code.

```
MotionState target = profile.get(t);
double error = getPosition() - target.getX();
double correction = pidController.update(error);
double feedForward = getFeedForward(target, getMass());
internalSetVelocity(feedForward - correction);
```

H1: Constructing the Pit

H2:

B1: Planning the Nevada County Student Hackathon

Week 31

H1: Changing the Intake Arm Length and Whip Stiffness

B1: Work on the Business Notebook

Figure 32.1: Ashlin Modifying the Latch

Week 32

H1: Latch Tuning

Once the team started drive practice for worlds a problem that Aidan noticed was that latching with the new latch was proving to be difficult. With the new modifications to the latch that raised it higher off the ground, there was less room to latch, making it harder. To fix this, the solution was to mill the inside face down to increase the gap used for latching. Oren ran several stress analyses in Inventor to ensure that the latch could still support the weight of the robot. With data from the stress analysis it was determined that removing 3/16 of an inch would be optimal for ease of use when latching and still strong enough to hold the full weight of the robot. Ashlin then went with Dan to GSS and did the desired modifications. Latching then became significantly easier for the drivers.