RBS2

Engineering Portfolio

25324

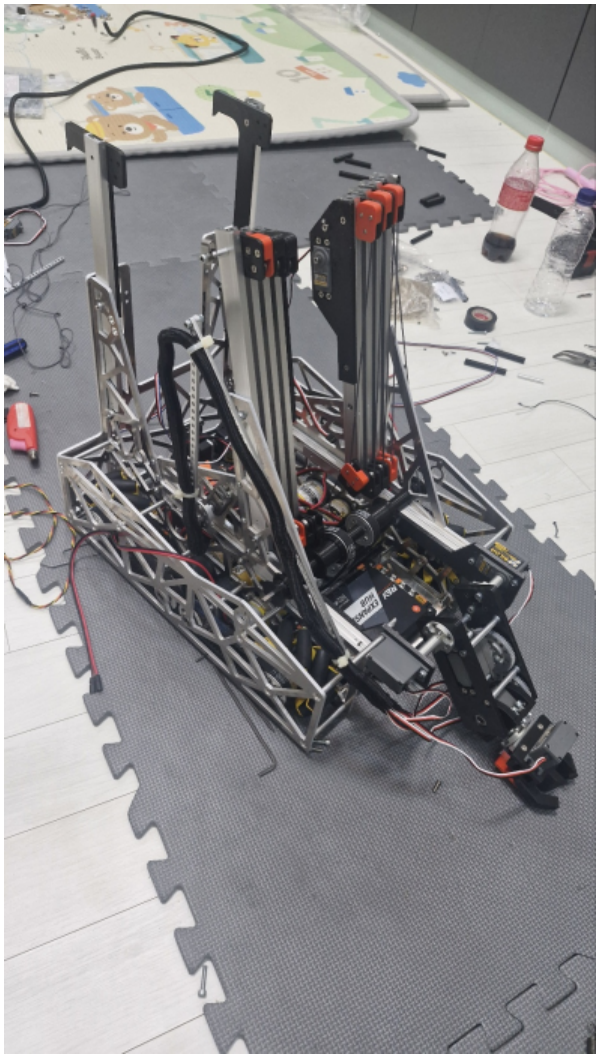
Team organization

-Captain:
Heewon_yeah

-Designer:
Mingae_kim
Mingyu_seo

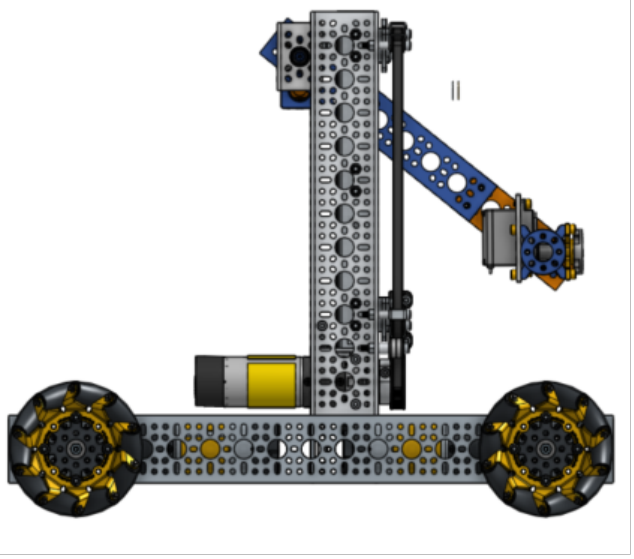
-Hardware:
Yuchan_song
Hyunmin_choi

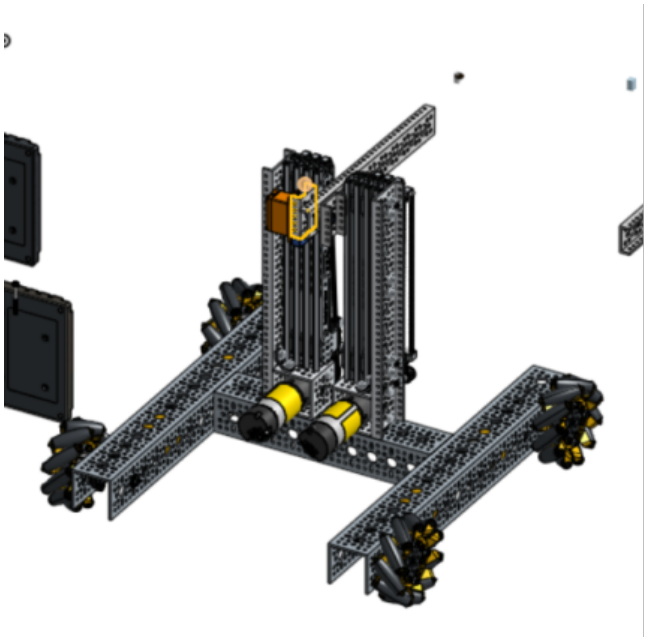
-Programmer:
Dahoon_lee
Sunghyeon_choi



Design

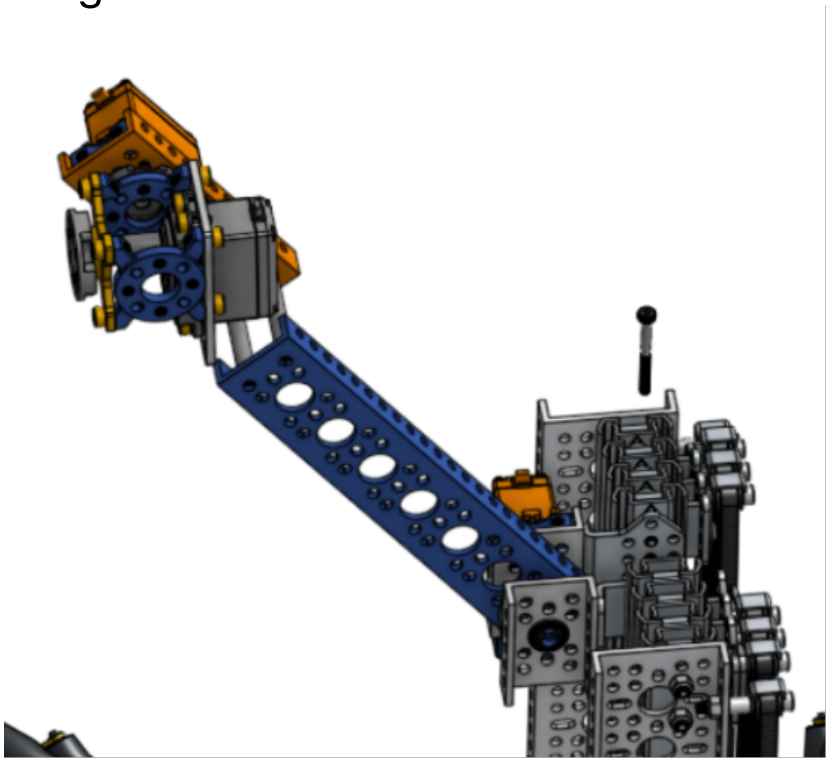
- **Robot Design Goal**
  - I had to design a robot to compete in the KRC competition.
  - When I first designed it, I thought about making a robot that could both chamber and basket.
- **Production Process**
  - I made it with the high slider at a right angle.
  - I attached one servo to the high arm and turned the arm to obtain samples from the human play zone and then turned the arm again to hang it in the chamber so that the robot's path would be as efficient as possible.
  - I attached a low arm and low slider for collection, and I used this to obtain samples inside the submersible and transfer them directly from the low arm to the high arm.
- **Efficiency Issues**
  - I think that by transferring the low arm directly to the high arm, collection and chambering can be done at the same time, which is efficient, and the basket can also be made efficient.
  - However, several problems were discovered when doing this.





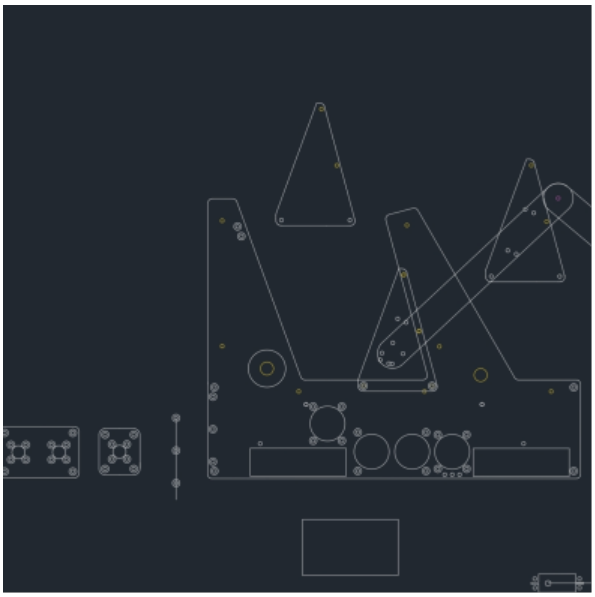
Design

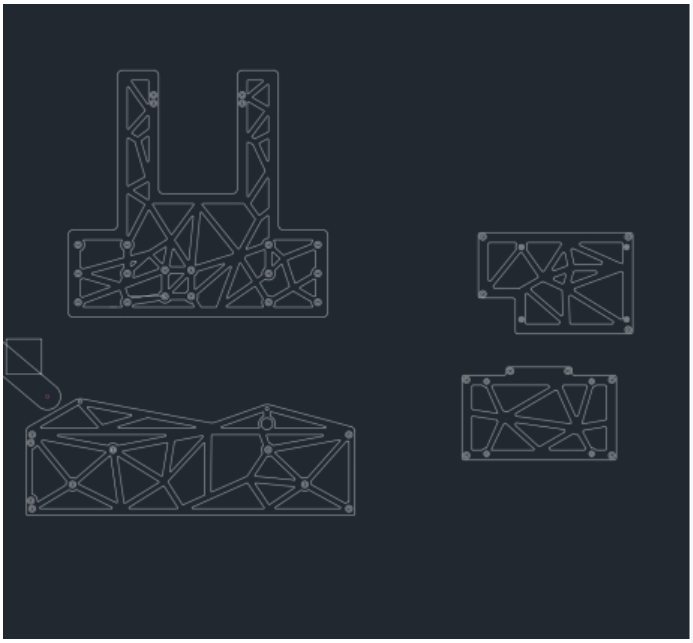
- **Problems**
  - First, we used one servo motor to rotate the high arm, but the speed of rotating the arm was too slow and it seemed to lack power.
  - So we changed the servo motor to a model with a higher torque, but it was still insufficient.
  - Sometimes, the servo motor would die.
  - When the high arm was perpendicular, it was easy to perform the chamber and basket missions, but it was a problem because it did not hang.
  - We also had a big problem with how to hang the first stage.
  - The sliders used a Gobilda slider and belt mechanism, but this was too heavy, did not move smoothly, and was difficult to do quickly.
- **Participation in the KRC competition**
  - We participated in the KRC competition with these problems, and were lucky enough to receive good results and participate in the FTC competition.
  - So now we have designed a robot that has fixed these problems.



Design

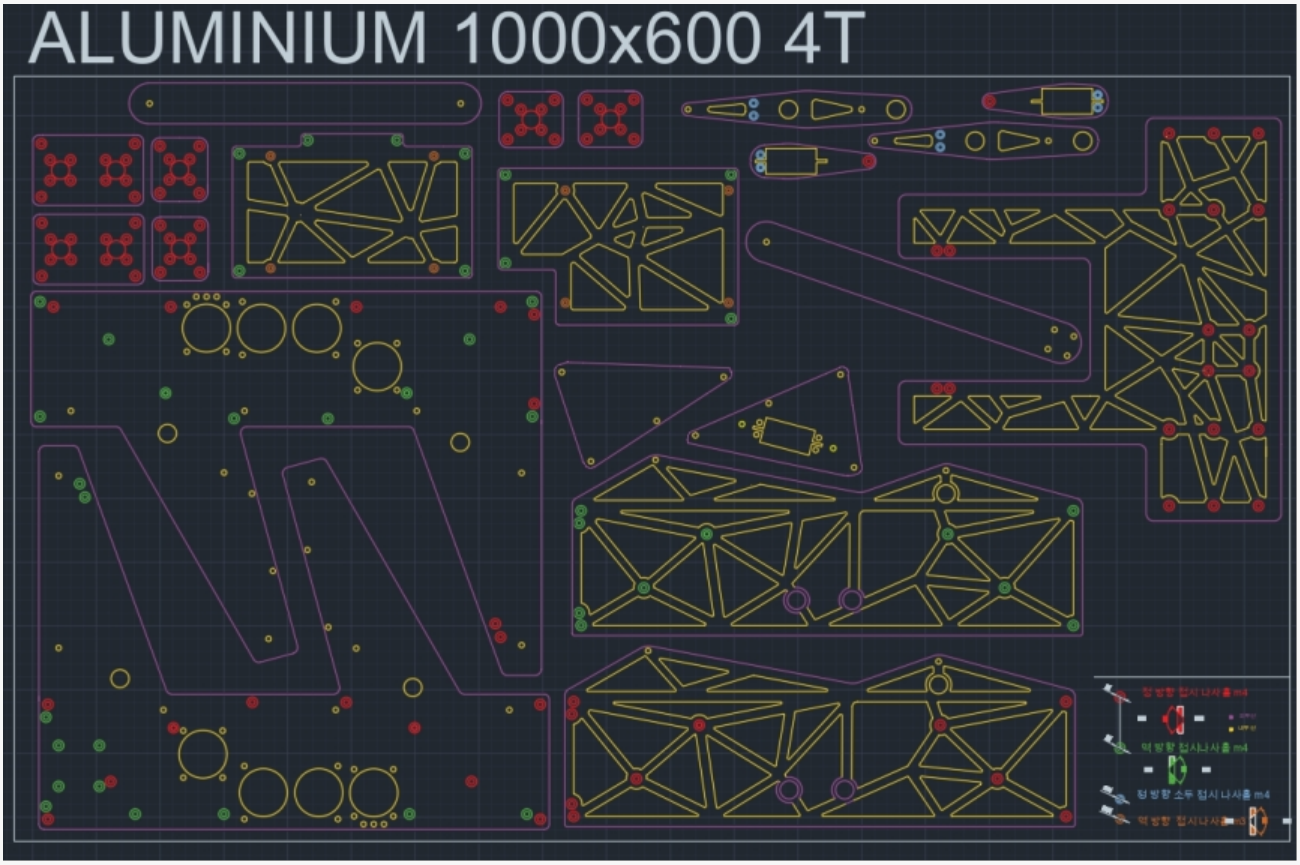
- **Problem**
  - One servo motor is not powerful enough to turn the arm.
  - When the high arm is perpendicular, the chamber and basket missions are performed smoothly, but there is a problem that hanging is not possible.
  - The slider movement is not smooth.
- **Solution**
  - The robot mechanism was similar, but the robot was rebuilt.
  - To lighten the weight of the robot, the frame was made of aluminum.
  - AutoCad was used to make the custom frame.
  - While making the custom frame, the drive mechanism was changed to a belt type.
  - To make the custom frame, the existing problems were solved while designing the frame.
  - The high slider was given an angle to enable hanging, and one more slider was added at the very back to enable one-step hanging.
  - The frame frame was designed while considering the slider position, and the shape of the frame was almost set.



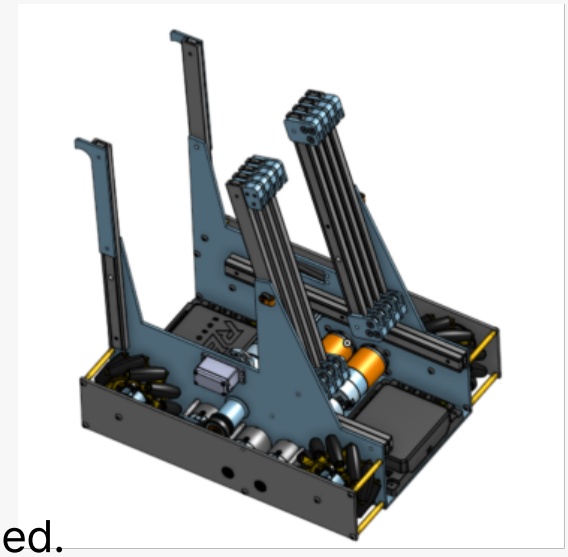


Design

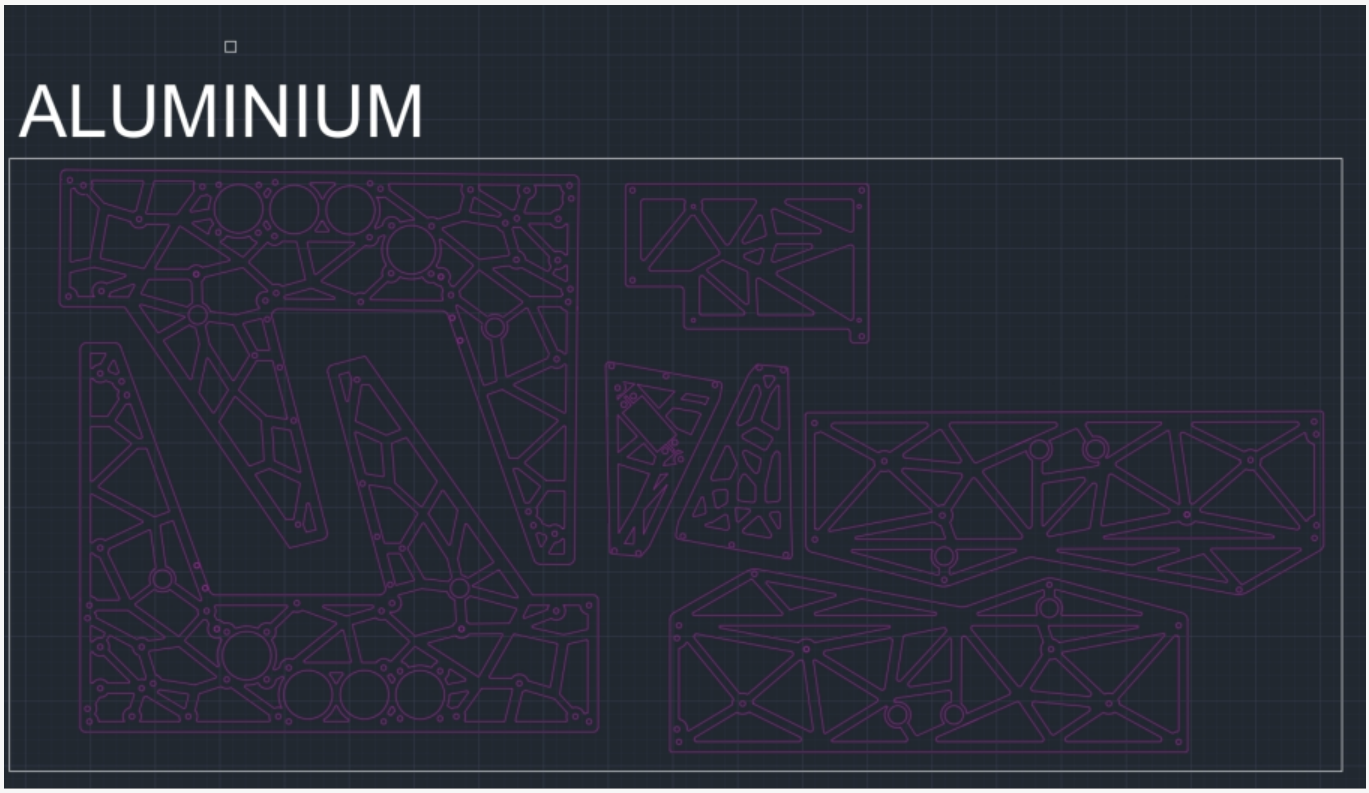
- **Adjusting the belt length**
  - I took care of the motor position while paying attention to the belt length.
  - I left some space because I thought it might be difficult to install the belt due to the belt tension.
  - However, since this was my first time using the belt mechanism, I didn't know that I shouldn't leave space.
- **Modifications**
  - The belt length of the first wooden frame I ordered didn't match, so I had to modify it again.
  - There were other minor problems besides the size.
- **Problem solving process**
  - I modified the CAD and checked the problems again through 3D CAD work, and then outsourced it to aluminum.



Design

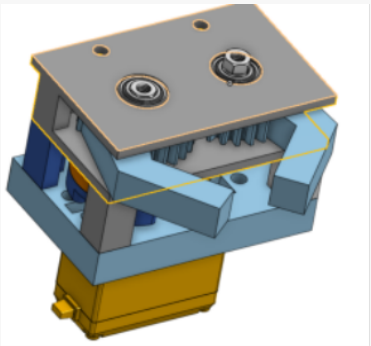


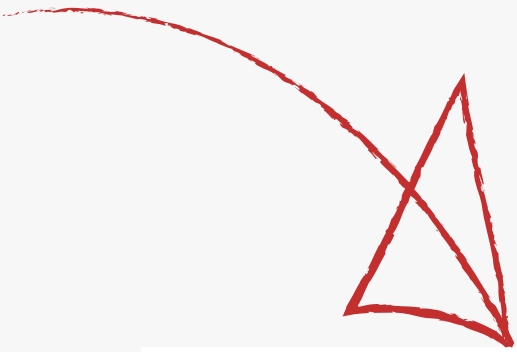
- **Assembly Process**
  - I started assembling the frames that arrived.
- **Problem Encountered**
  - A problem occurred.
  - But this time, the inner frame was too heavy because the holes were not drilled.
  - Also, due to a design error, the position of the bracket that mounts the control hub interfered with the odometry.
  - This caused the frame to be manufactured.
- **Solution Implemented**
  - I modified the CAD again due to the design error and drilled holes to reduce the weight.

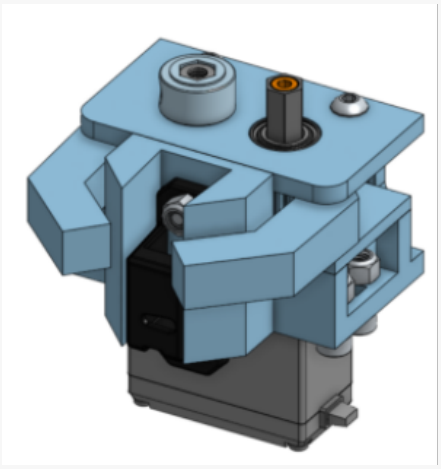


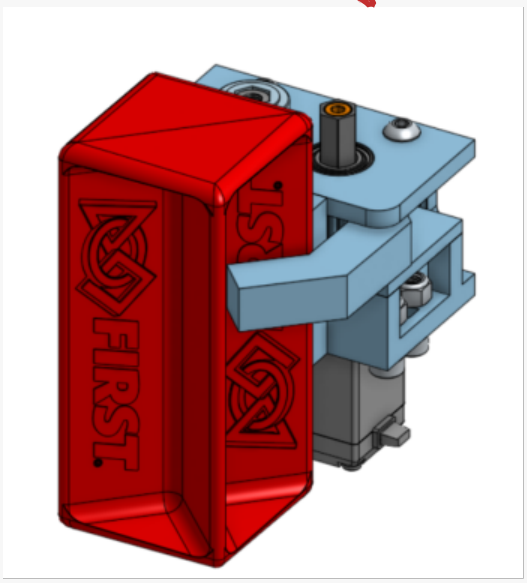
Design

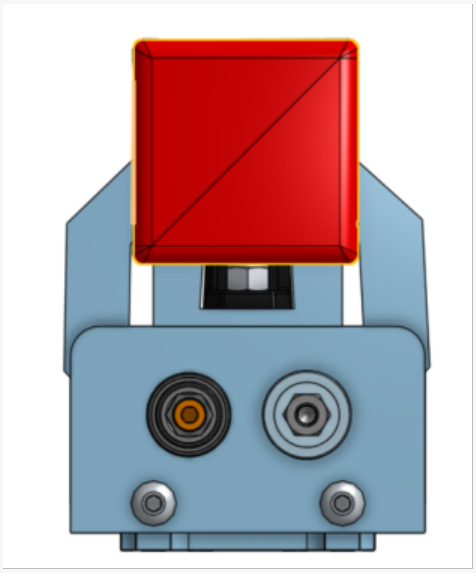
- **Gripper Design**
  - The Gripper design was made smaller than the original design and the design was slightly changed to accommodate the color sensor.
  - And a protruding part was added to the front to make it easier to hold the sample.
- **Effect of the Changes**
  - These changes will greatly help to improve the functionality of the Gripper and improve user convenience.
  - We plan to continue to develop the design through continuous feedback.





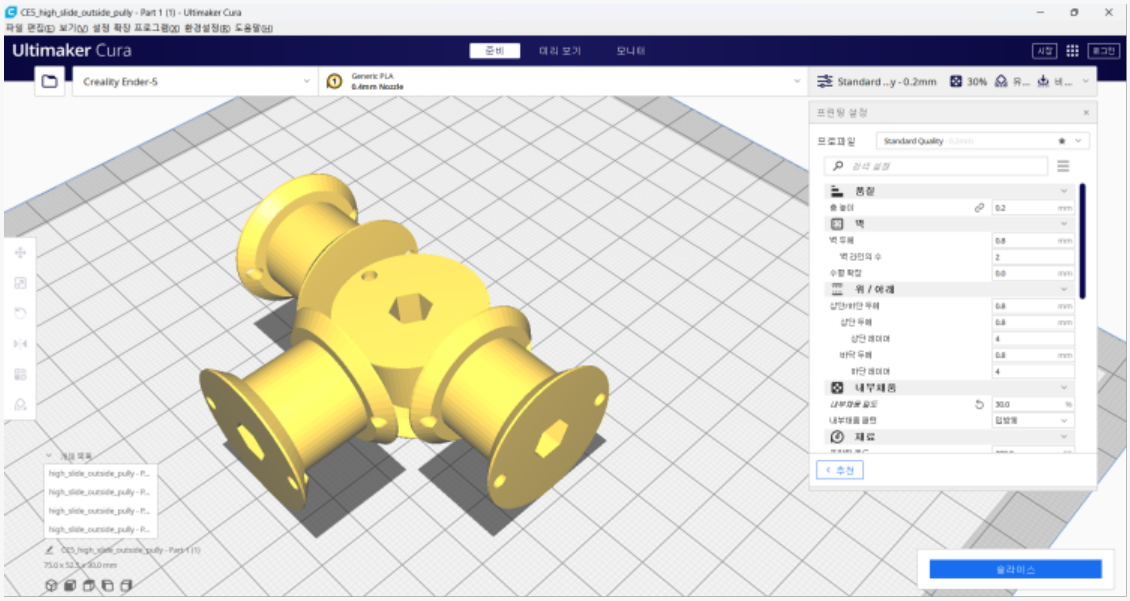


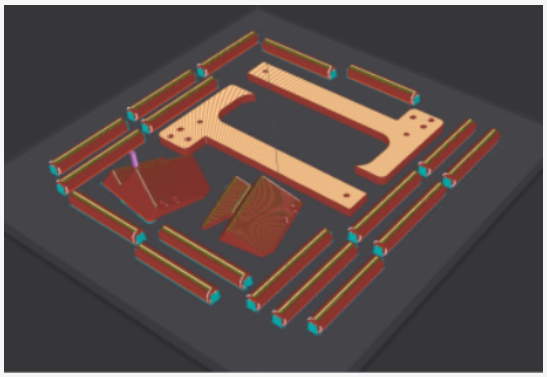




Hardware

- **Problem**
  - Almost all the parts were printed with a 3D printer.
  - However, a very serious problem occurred.
  - The rigidity and quality of the parts printed with a 3D printer were very poor in very important factors.
  - So we secured the rigidity by increasing the density of the parts printed with a 3D printer.
  - We improved the quality of the parts by slowing down the printing speed.



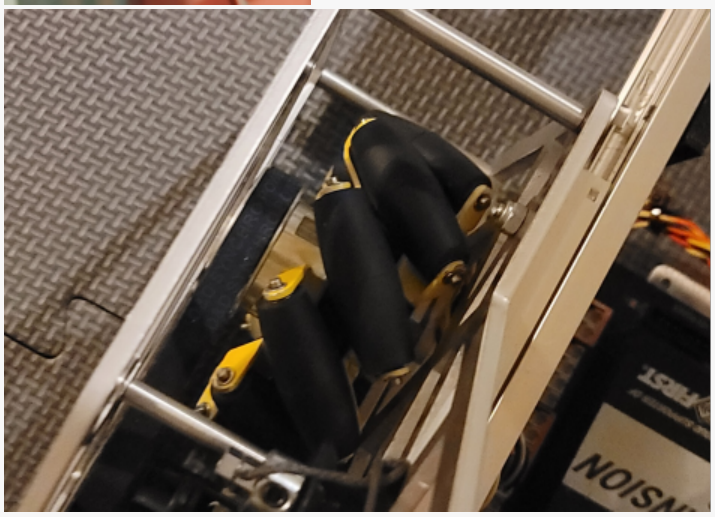


Hardware

- Servo Motor Information
  - I bought a servo motor, but it was different from what I expected.
  - The first connector of the two 3-pin connectors was full, and the remaining connectors only had 1 pin.
- Used Servo Motor
  - Since the control hub we use only supports 3-pin servo motors, we built the robot using the Gobilda servo motor.
- Axon Servo Motor Research
  - Without giving up, we researched the Axon servo motor and found out that the 1st pin is an encoder.
  - So we decided to use the Axon servo motor in case the Gobilda servo motor breaks down.

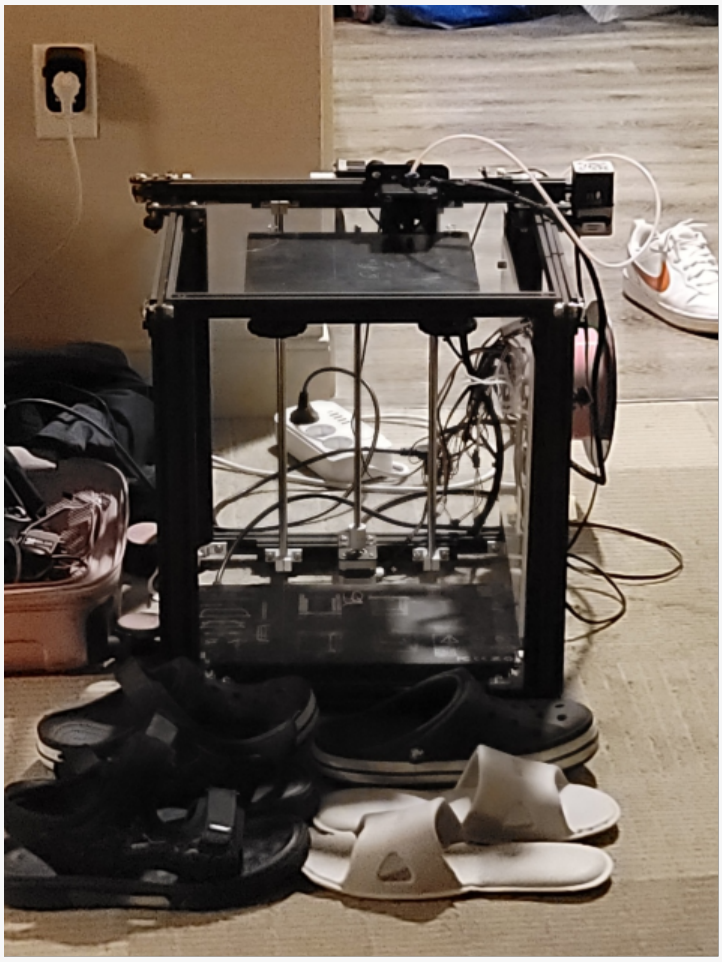
Hardware

- **Servo motor problem**
  - While installing the servo motor on the robot, an unknown problem occurred with the servo motor.
- **Mecanum wheel replacement**
  - While installing the Mecanum wheel and controlling the robot, the left rear Mecanum wheel was bent, perhaps due to an impact.
  - So, by replacing the size of the Mecanum wheel from 104mm to 96mm old wheel and installing a redesigned frame to fit the overall smaller wheel, we were able to successfully deal with the unexpected variable.
- **Wire management problem**
  - While organizing the wires, the wires got tangled and also fell off.
  - By tracking each of these and marking the beginning and end of the wires, we were able to drastically reduce the time it took to deal with the same problem again the next time.



Hardware

- 3D Printer
  - We disassembled the 3D printer in Korea and brought it to the US in carriers.
  - Korea uses 220V and the US uses 110V, so we had a power shortage problem.
  - This problem could have been solved by buying a transformer.
  - We decided to ask for help from other participants who were attending a nearby school because we thought it would be a waste of money.
  - This way, we were able to secure a lot of spare plastic parts in case our robot broke down.



Software

- Driving System
  - We have based the driving system used in the KRC competition.
  - This system allows the robot to move according to the driver's standard.
  - It makes driving a little more convenient and faster.

- PID and Odometry
  - We also wanted to secure accuracy in autonomous driving by using PID after purchasing odometry.
  - Since it was our team's first time using odometry and PID, there was quite a bit of trial and error from simple usage and setup methods.
  - So we wrote tuning and driving codes that fit our robot based on the road-runner_quickstart video uploaded by the FTC 6282 team on YouTube.

```
Actions.runBlocking(
        drive.actionBuilder(new Pose2d( positionX: 0,  positionY: -64.05,  heading: Math.PI / 2))
              /*  .stopAndAdd(this::Chamber_score1) */
                .lineToY( posY: -31.95)
              /*  .stopAndAdd(this::Chamber_score2)*/
             /*   .stopAndAdd(this::Chamber_score3)*/
                .waitSeconds( t: 0.5)
             /*   .stopAndAdd(this::Chamber_ready1)*/
                .waitSeconds( t: 0.5)
             /*   .stopAndAdd(this::Chamber_ready2)
                .stopAndAdd(this::Chamber_ready3)*/
                .lineToY( posY: -40)
                .splineToConstantHeading(new Vector2d( x: 47.5, y: -49.5), tangent: Math.PI/2)
                .lineToY( posY: -50)
                .setTangent(0)
                .lineToX( posX: 49)
                .lineToY( posY: -62)
                .lineToY( posY: -50)
                .setTangent(0)
                .lineToX( posX: 58.5)
                .lineToY( posY: -62)
                .lineToY( posY: -50)
                .setTangent(0)
                .lineToX( posX: 65.78)
```

Software

- **Tuning Process**
  - I tuned pid auto.
  - The problem that occurred while tuning was that an error screen kept popping up when running laterpushtest.
  - Afterwards, I found out that I didn't need to do lateralpushtest if it was 2deadwheel instead of 3deadwheel.
  - Also, when I first tuned, the angular graph on the dashboard was not consistent and there were a lot of outliers.
  - However, after tuning it twice more, I was able to catch the values more accurately and solved the problem of many outliers.
- **Manual Feedback Issues**
  - Also, when doing manualfeedback, the robot's path kept going wrong, so it took a lot of time to catch this value.
- **Code Creation**
  - After completing the tuning, I created an auto code using lineToY or lineToX.
  - However, an error kept popping up while using lineToX.
  - To solve this, I referred to several videos on YouTube and found out that I had to use the settangent statement before using lineToX.
  - I also created a dashboard using the MeepMeep code.
  - I made the autonomous driving code run on a computer without a robot in a simulator.
- **Gear Issues**
  - Also, there was a problem with the gears that were stuck between the shafts in the highslider, so I removed the shafts and replaced the gears.
- **Position Adjustment**
  - When I was making the autonomous driving code, I helped readjust the robot's position and re-place the blocks.

Software