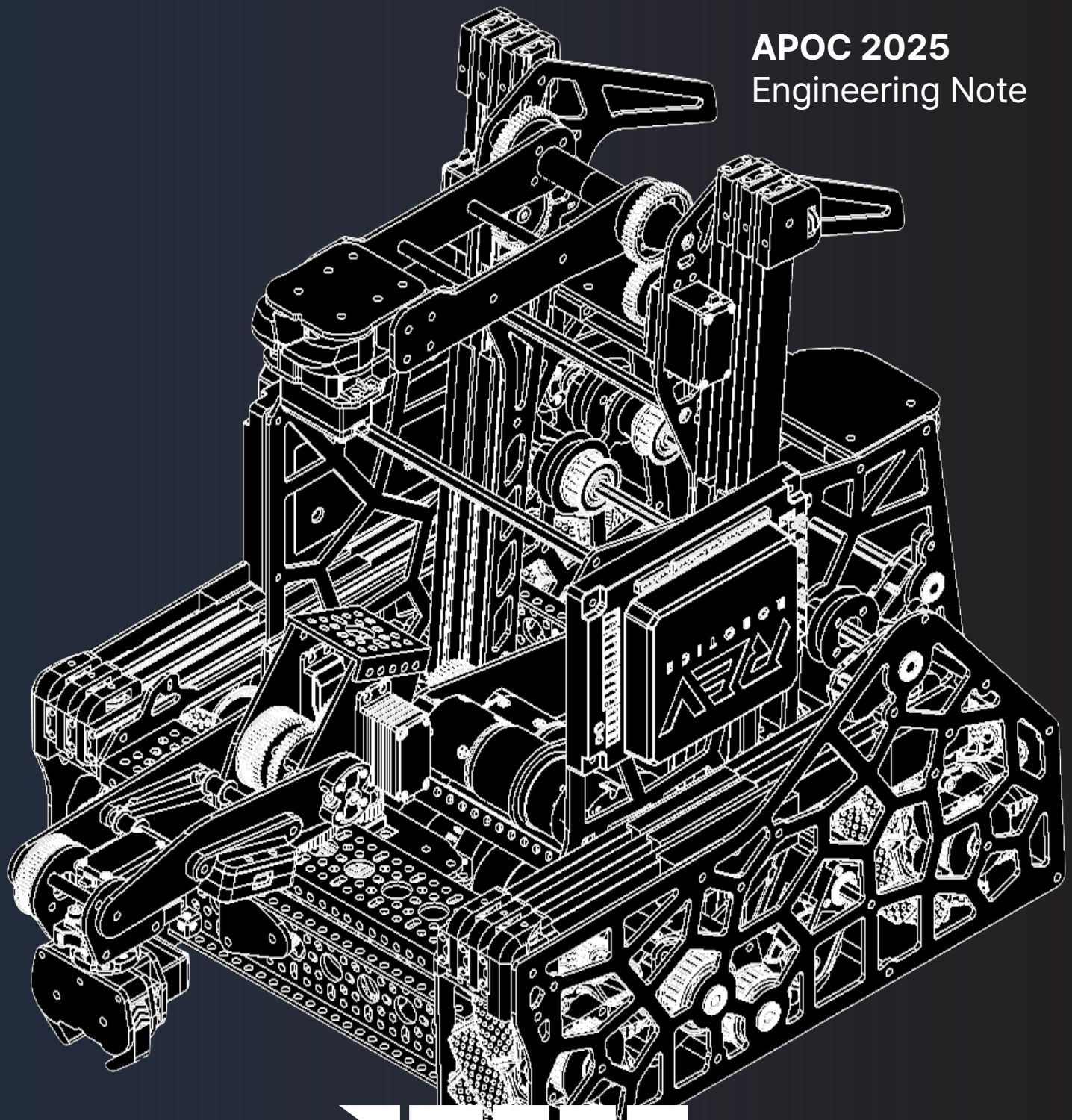
TALOS

APOC 2025
Engineering Note



2509

APOC TALOS Engineering Notebook

Editor | 23 Kim Joonsung (xsorexia@gmail.com)
Season | 2024-25 FIRST DIVE : Into the DEEP APOC
Date | 2024/12/17 - 2025/07/10

Team TALOS is made up of research members from the robotics club KROS, which operates with support from the Korea Science Academy and the Ministry of Science and ICT.

TEAM MANAGEMENT

TALOS 25309

This section is about the general information of Team TALOS.

APOC 2025
Engineering Note

I. Team and Members

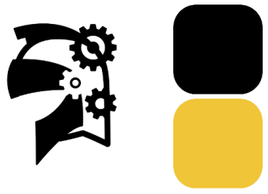
1. Team TALOS

1) Team Introduction

TALOS (#25309) is a FIRST Tech Challenge (FTC) team. Comprised of 14 research members from the robotics club KROS, TALOS has been participating in the Korea Robot Championship (KRC) since the 2018-19 Rover Ruckus season. Last year, during the 2024-25 CENTERSTAGE season, they achieved the remarkable feat of winning the Inspire Award and took part in the FTC World Championship held in Houston as the representative team from Korea.

📌 **Origin of our team name TALOS**

In ancient Greek mythology, it is said that the blacksmith Hephaestus, at the request of Zeus, created Talos, the first robot, to defend Crete. Our team name embodies our determination to build impressive robots, just like Talos, which Hephaestus created through reason and craftsmanship.



TALOS's official colors are yellow, black, and white, and its emblem is shaped like a helmet. Previously, the team used the Korean team number #5073, but since last season has been using #25309. They have produced team shirts and badges featuring this logo.

2) Team History

TALOS has consistently participated in the KRC since the 2018-19 season, and the 24-25 Into the DEEP season marks their sixth season in total. Alumni who have graduated continue to support the team as mentors, helping them grow further.

ROVER RUCKUS
Presented By Qualcomm

2018-19 ROVER RUCKUS

🏆 N / A

SKYSTONE
PRESENTED BY Qualcomm

2019-20 SKYSTONE

🏆 N / A

FREIGHT FRENZY
PRESENTED BY Raytheon Technologies

2021-22 FREIGHT FRENZY

🏆 Think Award (1st EN)

POWERPLAY
PRESENTED BY Raytheon Technologies

2022-23 POWERPLAY

🏆 Connect Award (2nd EN)

CENTERSTAGE
PRESENTED BY RTX

2023-24 CENTERSTAGE

🏆 Inspire Award

INTO THE DEEP
PRESENTED BY RTX

2024-25 INTO THE DEEP

🏆 **2023-24 FIRST World Championships**

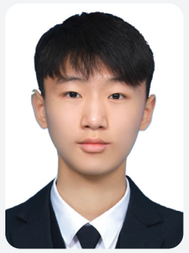
In the CENTERSTAGE season, we won the Inspire Award at KRC and represented Korea at the FIRST Championships in Houston.

Out of 10 matches, we won 4, and by interacting with overseas teams, we learned various techniques and workflows to apply to our robot design and build. This experience greatly helped us design our robot for the 24-25 season.

2. Members

1) Builder

Builders handle designing and building the robot. They test and decide which mechanisms to apply to each part, then build the robot based on these choices. They also discuss game strategies with programmers, adjusting plans to fit the robot's structure.



22 김수기 | Sugi Kim
Builder

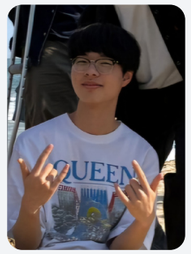
As a builder from the class of '22, I take part in building the robot.

22 최성빈 | Sungbin Choi
Builder

As a builder from the class of '22, I take part in building the robot.

22 강현빈 | Hyeonbeen Kang
Team Captain / Builder

As a builder from the class of '22, I take part in building the robot. I am also the team captain for APOC 2025.



25 전민수 | Minsu Jeon
Builder

As a builder from the class of '25, I take part in building the robot.

25 심우경 | Wookyung Shim
Builder

As a builder from the class of '25, I take part in building the robot.

2) Programmer

Programmers write the code that runs the robot. They join in planning and designing the robot, and during the build, provide code to test each part. Using Onbot Java, they ensure the finished robot works for both TeleOp and AutoOp.

22 유태우 | Taewoo Yu
Programmer

As a '22 programmer, I take part in coding.

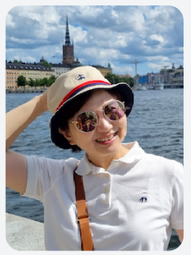
23 김준성 | Joonsung Kim
Outreach

As a '23 outreach member, I handle external tasks such as connecting with overseas teams, managing the website, and organizing the engineering notebook.

24 이재빈 | Jaebin Yi
Programmer

As a '24 programmer, I take part in coding.

3) Coach / Mentor

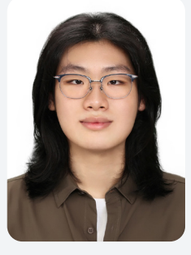


김호숙 선생님 | Dr. Hosook Kim
Coach

She is the head of the Department of Mathematical & Information Sciences at KSA and serves as the advisor for Team TALOS.

19 이승찬 | Seungchan Lee
Mentor / Alumni

As a mentor from the class of '19 at KSA and former team captain, he continues to support the team.



21 민지홍 | Jihong Min
Mentor / Alumni

As a mentor from the class of '21 at KSA and former team captain, he continues to support the team.

II. Team management

1. Team Management

1) Sponsor

TALOS operates with support from the Ministry of Science and ICT. Through the school, the team receives a budget used to buy parts, tools, and accessories needed to build the robot. They also work in a maker space called the Dream Design Center.

과학기술정보통신부
Ministry of Science and ICT

2) Workflow

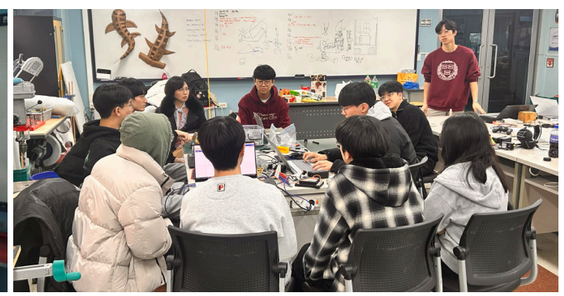
As introduced earlier, TALOS divides roles into builders, programmers, and outreach, with 11, 3, and 1 students handling each, respectively. As shown in the diagram below, tasks are scheduled by period, and each step needs to be completed for the next to proceed smoothly. Also, for design, planning, and idea meetings, all members participate regardless of role, allowing for broader perspectives.

Because students come from all over the country and attend a boarding school, it's hard to work together during vacations. So they hold multiple meetings during the semester and stay on campus from two weeks before competitions to build the robot. The photo below shows one of these meetings.

△ **Sep 8 |** Mission reveal

△ **Sep 9 |** Meeting with coach

△ **Sep 30 |** Discussing intake method



△ **Jan 12 |** Discussion at school

OUTREACH

OUTREACH

This section is about this season’s outreach activities at Team TALOS.

APOC 2025
Engineering Note

I. KSASF Booth

1. KSASF Preparation

1) What is KSASF?

KSASF stands for Korea Science Academy Science Fair, an annual science festival held at KSA. In odd years, it's run as an international event for gifted education institutions abroad, while in even years, it targets elementary and middle school students in Korea. In 2024, it was held as a domestic event with 140 students participating.

We thought running a booth where a robot completes missions would be a great experience for young students interested in math and science. Seeing it as a chance to introduce robotics and the FTC to them, we prepared and ran the booth over the summer break.

2) Process of preparing

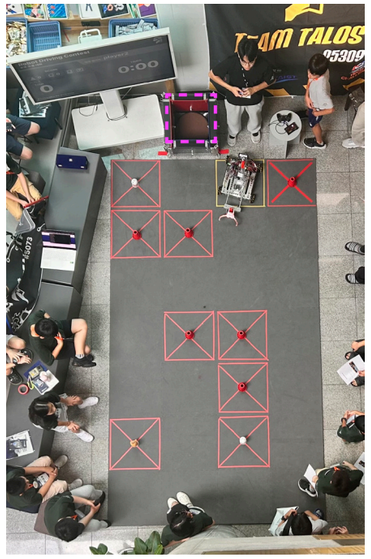
From July 29 to August 5, 2024, we stayed on campus to prepare the booth. Five builders and three programmers took part, designing the mission themselves and building the robot, then running a booth where students could control it and compete for scores.

⚒ **Builders | '22 J.H., '22 K.H., '24 L.S., '24 L.S., '24 H.J.**
Programmers | '22 Y.T., '23 K.J., '24 Y.J.

It was a great chance for the class of '24 to gain hands-on robot building experience before the competition, and they also prepared a separate engineering notebook to document the process in advance.

2. Booth

1) Mission



KSA-RC (Korea Science Academy Robotics Competition)는 로봇공학 연구회 KROS에서 진행하는 팝업 부스입니다.

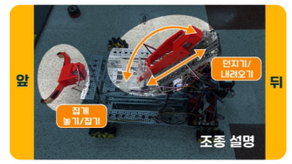
행사 타임라인

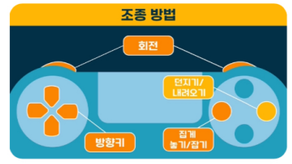
KSA-RC에 참가해보세요!
행사 타임라인은 아래와 같으니 참고하세요.

1. 행사 등록하기
직접 경기를 즐기고
다른 친구들과 경쟁하세요!

2. 규칙 익히기
직접 경기를 즐기고
다른 친구들과 경쟁하세요!

3. 경기 즐기기 / 경쟁하기
직접 경기를 즐기고
다른 친구들과 경쟁하세요!





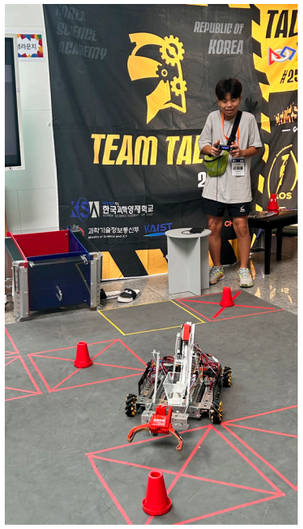
As shown in the top left photo, the robot starts in the yellow box. It can move anywhere except the red grid, where objects to pick up are placed. These objects can be collected and put into the pink basket, with points varying by type. The time limit was set to 1 minute.

We placed 6 red cones, 2 balls, and 1 doll as objects. Instructions (shown top right) were posted at the booth, and students waiting were taught how to use the controller.

Also, as seen in the photo below, we created a program (by '22 Yoo Taewoo) to count scores during matches, allowing students to see the remaining time and current score. Before running the booth, we tested the game with KSA students and gathered feedback.

2) Running the booth

We ran the booth on August 6 on the first floor of the school. Out of 60 students who signed up, 37 actually participated. Many showed interest, and we also had time to answer questions from younger students.



We filmed and edited the event as a vlog and uploaded it to YouTube. (YouTube 링크)

Students we met

60

Students who participated

37

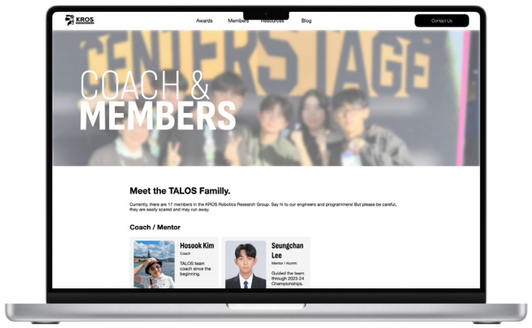
II. Website and resources

1. Creating the website

1) TALOS Website

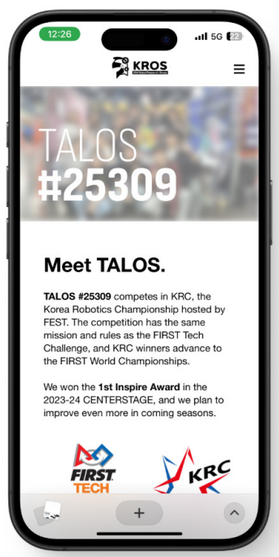
We created a website to introduce Team TALOS, aiming to share our engineering notebook, posters, world championship guides, and more.

Using the domain talosftc.com, the site sees about 50 visitors each week. We plan to update it annually with new content and resources.



Total visitors

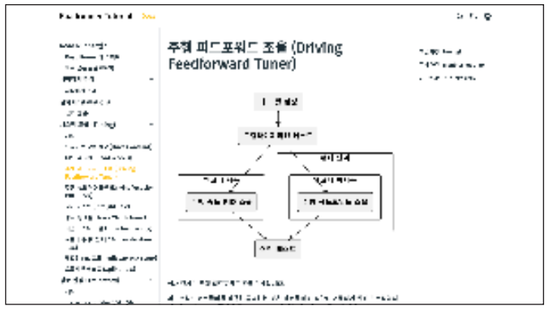
1083



2) RoadRunner Guide Website

Road Runner is a motion planning library created by ACME Robotics and others that makes it easy to use odometry and an IMU for autonomous driving. Since scoring high during the Autonomous Period is critical for overall FTC rankings, most top teams now use Road Runner for precise autonomous routines. However, in Korea, few teams actively use Road Runner to achieve high autonomous performance, due to various reasons.

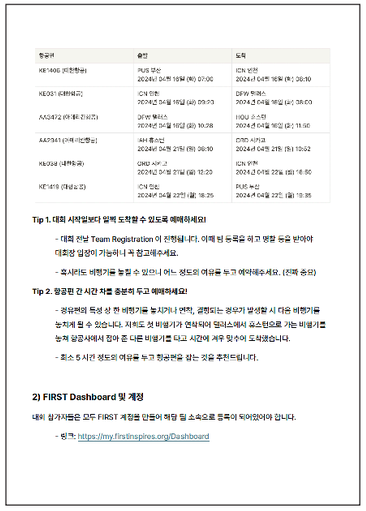
To help Korean FTC teams adopt it more easily, we built a Korean-language website (guide.talosftc.com) explaining how to use Road Runner. Designed to be simple and intuitive even for beginners, it also offers features like simulations and constant generators, and we expect it to be a valuable resource for many Korean FTC teams.



2. Sharing resources

1) FIRST Tech Challenge World Championship Guide (for Kor teams)

While preparing for the 2023-24 FIRST Tech Challenge Championship, Team TALOS faced many trial-and-error moments with booking flights, reserving hotels, and adjusting schedules, due to a lack of experience. It was our first time attending the world championship, and we had no teams to consult. After the event, we expected other Korean teams would face similar issues, so we created a guidebook to help them prepare.



We shared this guide on our website, talosftc.com.

III. Exchange with FTC teams

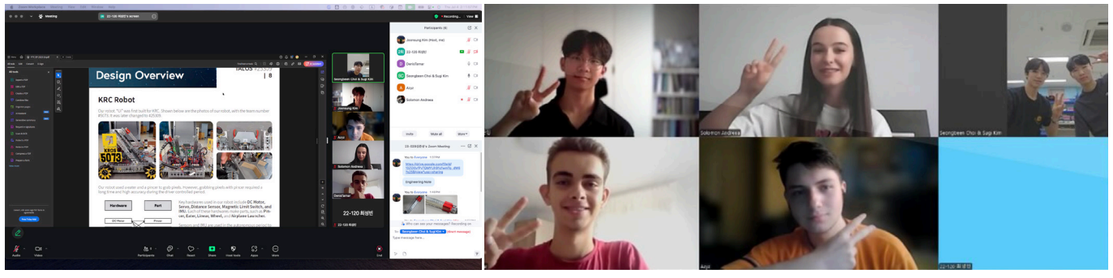
1. Exchange with overseas FTC teams

At the 2024 FIRST World Championships, we saw many overseas teams connecting through video meetings and other platforms. We thought it would be valuable for TALOS to also ask them about their robot design and build processes, show our work, and learn from them.

From summer 2024 to winter 2025, Team TALOS had opportunities to engage with teams from various countries. By sharing experiences on FTC competitions and prep, team operations, robot design and programming, and competition insights, we gained useful ideas for preparing for the KRC and also shared our own. These exchanges took place by reaching out via Instagram DMs and emails, or by accepting meeting requests.

1) 22114 Teoretika (Romania FTC)

Date | July 4, 2024
Participants | 3 from Teoretika, 3 from TALOS



Teoretika, now in their second season, suggested a Zoom meeting via Instagram, which was held on July 4. From TALOS, ‘22 K.S., ‘22 C.S., and ‘23 K.J. participated.

Both teams shared their Engineering Notes and Portfolios, discussing robot design, build, and programming for the 2023-24 CENTERSTAGE season.

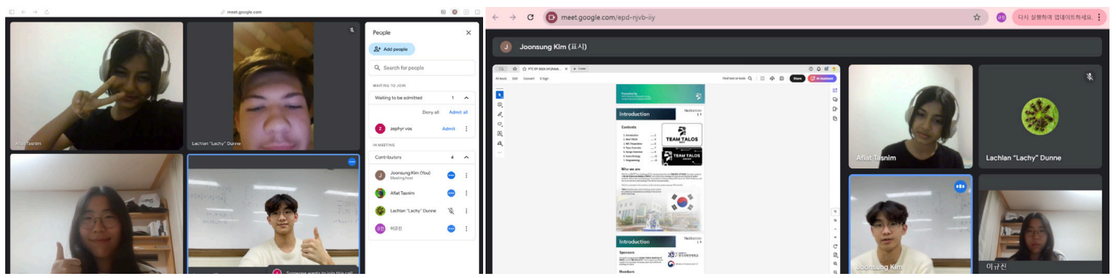
They also exchanged ideas on team selection and design processes. Teoretika said they let all interested students join the build process, then observe and select some to compete (due to the 15-member limit).

Teoretika used a webcam to detect AprilTags and Randomization Objects in the CENTERSTAGE season. We asked about their implementation and code, and received shared materials.

We also talked about how the FTC season and competitions run in Romania and Korea, and TALOS shared experiences from this year’s World Championship.

2) 18439 RoboKings Thorium (Austrailia FTC)

Date | October 25, 2024
Participants | 2 from RoboKings Thorium, 2 from TALOS



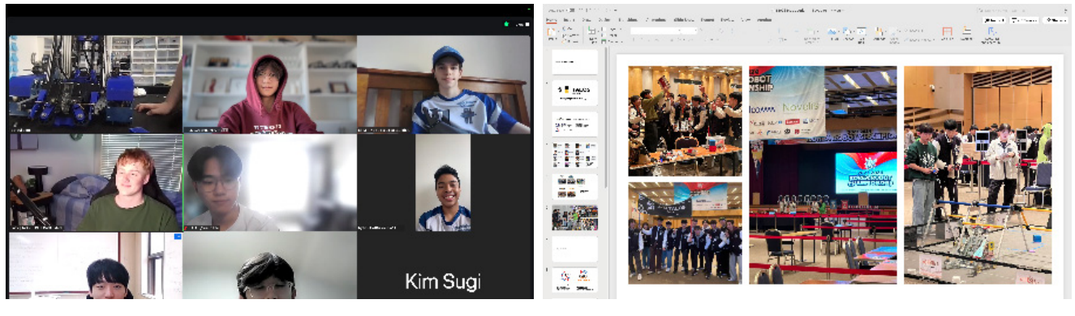
Through a student from 12993 RoboKings Aurum, whom we met at the 2024 FTC World Championship, we arranged a Zoom meeting with 18439 RoboKings Thorium. From TALOS, ‘22 L.K. and ‘23 K.J. participated.

Most members of RoboKings Thorium (hereafter RoboKings) were in their first FTC season, so they asked how TALOS prepared last season and how we’re preparing this one. We shared our Engineering Portfolio from last year’s Worlds, explaining our robot design and build process. We also discussed how focusing on the high basket could be key for this season’s robot strategy.

Both teams shared what they’ve learned personally through FTC, and we explained Korea’s season schedule and prep timeline.

3) 14380 Blue BotBuilders (Austrailia FTC)

Date | January 4, 2025
Participants | 4 from Blue BotBuilders, 5 from TALOS



Met with 14380 Blue BotBuilders, winners of the Inspire Award and Winning Alliance Captain at Australia's FTC Nationals, on January 4. From TALOS, '22 K.S., '22 C.S., '23 K.J., '24 L.S., and '24 Y.J. joined.

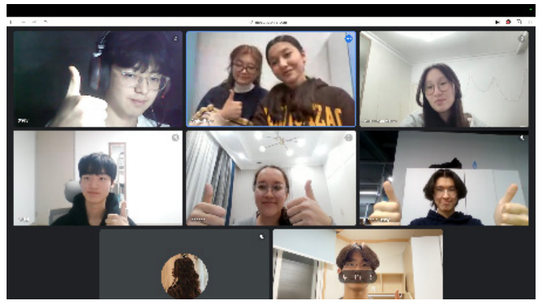
BlueBots shared their CAD and videos beforehand, so we prepared questions. They gave a detailed explanation of their active intake design.

They showed each robot part and explained their choices. TALOS, not yet started on building, showed our 3D model and detailed our intake plan.

We asked what penalties to watch for, and they advised being careful not to touch other robots during ascent.

4) 24893 Meow Meow (Kazakhstan FTC)

Date | January 7, 2025
Participants | 6 from Meow Meow, 4 from TALOS



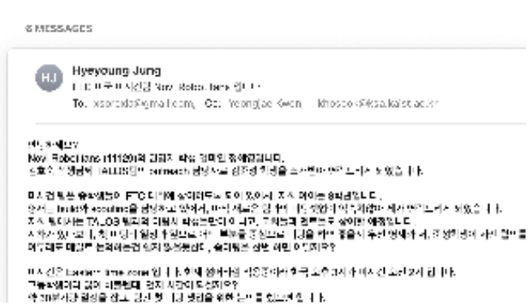
The Meow Meow team from Kazakhstan reached out, suggesting a meeting to discuss FTC in different countries, so we scheduled it for January 7. From TALOS, '23 K.J., '23 H.J., '24 L.S., and '24 Y.J. joined.

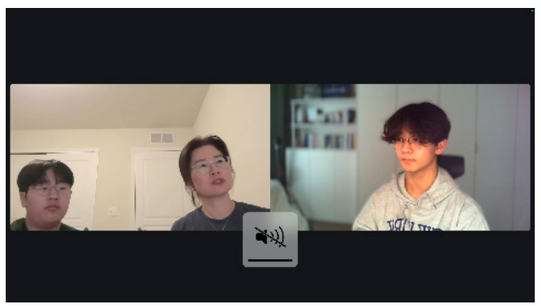
Meow Meow explained their X-shaped wheelbase (shown left), which we hadn't seen at KRC or Worlds, so we asked about its performance and control. They compared it to our tank drive, and we discussed pros and cons.

They shared that Kazakhstan holds eight Regional Competitions, open for teams to enter freely, giving them many chances. TALOS talked about our outreach efforts, described the KRC, and shared our 2024 Worlds experience.

5) 11129 Novi RoboTitans (US FTC)

Date | October 20, 2024
Participants | 2 from Novi RoboTitans, 1 from TALOS

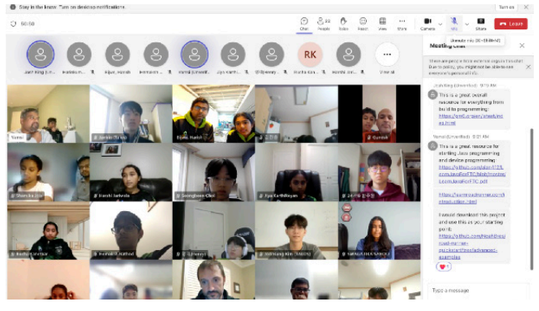




A mentor from Novi RoboTitans, whom we met at the 2024 FTC Worlds, suggested a Zoom meeting to help grow FTC in Korea and mentor Team TALOS. '23 K.J. joined from TALOS for the first meeting on October 20 to discuss future plans and schedules.

We planned a second meeting for the morning of October 26, where TALOS talked about areas where we needed mentoring. We asked for resources on FTC Dashboard, RoadRunner, and building processes. We also shared differences in judging between Korea and the U.S.

Date | October 26, 2024
Participants | 15 from Novi RoboTitans, 7 from TALOS



From TALOS, '22 K.S., '22 C.S., '23 K.J., '23 H.J., '24 Y.J., '24 H.J., and '24 L.S. joined. Novi RoboTitans had 4 coaches, 1 mentor, and 10 students. The meeting lasted about 1.5 hours.

Each team gave brief self-introductions and team overviews. Novi RoboTitans prepared a short PPT, with students taking turns presenting.

We received resources on FTC Dashboard and RoadRunner that we'd requested in the previous meeting. We asked many questions about robot building and got answers from their coaches. They also explained the PUGH Matrix method for choosing among different options.

6) 10001 RoboCavs Silver, 8479 RoboCavs Gold (US FTC)

Date | January 8, 2025
Participants | 5 from RoboCavs, 4 from TALOS

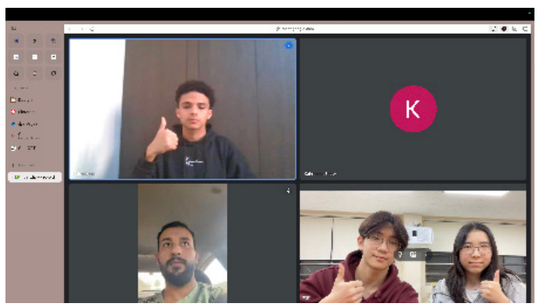

The RoboCavs team from Maryland saw TALOS's website and proposed a session to discuss Korea's FTC program and outreach. From TALOS, '22 L.K., '22 Y.T., '23 K.J., and '23 H.J. joined.

TALOS gave a PPT introducing the team and KRC, and briefly explained this season's robot design. RoboCavs also showed CAD files of their robot, introduced their team, and shared experiences with mentoring and outreach.

They explained that in Maryland, local comps are split into two divisions. We mainly discussed differences in competition formats, team operations, and mentoring, noting how RoboCavs parents serve as team mentors.

7) 18422 Wizards Robotics (리비아 FTC팀)

Date | January 8, 2025
Participants | 4 from Wizard Robotics, 2 from TALOS




Wizards Robotics from Libya reached out via Instagram to propose a session to discuss this season's robot design, outreach, and FTC programs in each country. From TALOS, '22 L.K. and '23 K.J. joined.

TALOS gave a PPT introducing the team and briefly showed this season's robot design, also explaining Korea's FTC event, the KRC. Wizards Robotics shared their outreach efforts and how local competitions work in Libya. Both teams asked questions about each other's robot designs and team operations.

In Libya, there are regionals, so teams like Wizards Robotics can compete multiple times. They said their team roles are evenly split with about four each in programming, building, and outreach. Wizards Robotics also won an Inspire Award and went to Worlds last year, though in a different division from us.

8) Conclusion

In the 2023-24 CENTERSTAGE season, we had no exchanges with other teams, but this season we actively connected and realized it's a great way to get help not just with building robots but also with team operations—and to share our own experiences in return.

People met
40+

Hours spent
8+

We've met and exchanged with over 40 people from 7 overseas teams. Hearing from Libya's Wizard Robotics that they connect with 50 teams each season, we plan to be even more proactive with meetings next season.

2. Korean FTC Teams

1) 25317 O.R.B.I.T

Date | August 25, 2024
Participants | 5 from O.R.B.I.T, 6 from TALOS

From TALOS, ‘22 K.S., ‘22 K.H., ‘22 J.H., ‘22 Y.T., ‘23 K.J., and ‘23 H.J. joined, while about 5 members from O.R.B.I.T took part.

O.R.B.I.T, now in their second season, asked how to better prepare since last year they competed with minimal prep. TALOS shared insights from last season and their Worlds experience.

2) 25360 R.O.G. United

25360 R.O.G. United, whom we met at the 2023-24 KRC, reached out with questions on preparing for the season.

Our club president, ‘23 H.J., responded by sharing info on parts, sensors we use, and our team’s GitHub with code.

3) Future Plans and Conclusion

During the 2024-25 season, we didn’t have many chances to connect with other Korean teams. However, we were able to share a lot based on last year’s KRC and Championship experience, and plan to reach out proactively to more teams for meetings next season.

ENGINEERING NOTE

BUILDING APOC 2025

This section is about the APOC 2025 season.

APOC 2025
Engineering Note

2025.06.16. (Mon)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. 3D print turret

Stopped by the 3D printer room and started printing parts for the turret.

2025.06.17. (Tue)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Building turret

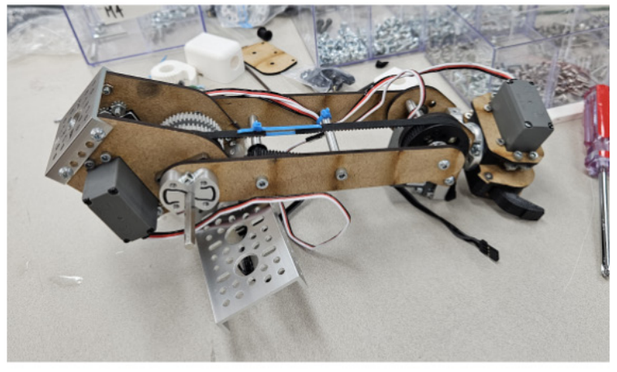
Problems during the build process:

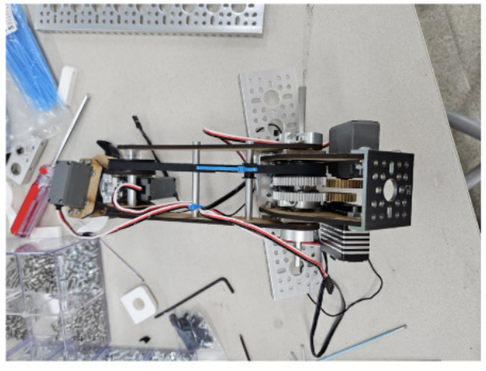
1. Gear and turret holder screw clearance
There was very little clearance between the screw and gear fixing the turret holder, causing the screw and gear to grind together. We tried changing the screw position, but the issue persisted, so we switched to flat-head screws.

2. Incorrect hole count
The design required 8 holes, but we mistakenly made it with 7. We rebuilt it with 8 holes, excluding the servo horn section, to match the original plan.

3. Timing belt interference and tensioning
There was a risk of the timing belt catching on the standoffs, and we needed a way to maintain belt tension. To solve this, we mounted a small black pulley on the standoff for the belt to run over smoothly.

4. Turret initial position alignment
Since it wouldn't be clear where the turret's starting position was when powered on later, we used a servo programmer in advance to set the initial position. We then built the turret to align precisely with that preset angle.





2. Reserving dormitory

Checked with the info desk to see if more dates were available. Then booked it for: tomorrow (18th) to the 19th, the 24th to July 3rd, and the 12th to 14th.

2025.06.18. (Wed)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Got Baekyang Hall key

Went to the info desk and picked up the key.

2. Assembled vertical/horizontal linear

Disassembled the Misumi linear rail, inserted a pulley part printed on the 3D printer, and built the vertical linear. Since the pulley wasn't aligned perfectly, re-printed and reassembled it.

Built the horizontal linear the same way as the vertical.

3. Purchased materials

Looked on Coupang for 8mm heat-shrink tubing, cable ties, and 40cm male-female jumper wires, trying to keep costs low.
Also asked the Intl. Research team if we could get super glue in return for helping them with the laser cutter before; after they confirmed, we'll pick it up Thursday morning.

2025.06.19. (Thu)

구분 | SUMMER_EN_BUILDING
책임자 | 22 C.S.

1. Future consumables purchase

Went to a nearby Daiso and bought various lengths of black cable ties and masking tape. Also placed large orders on Coupang for a spring set, 8mm heat-shrink tubing, and 20cm/40cm male-female 40-pin jumper cables:

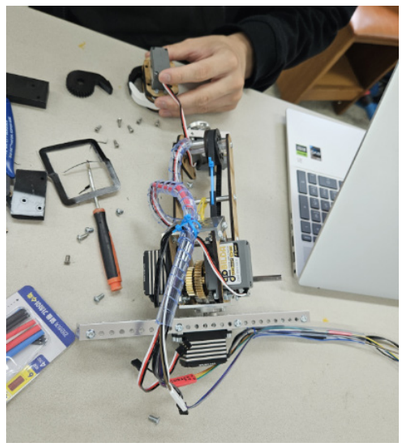
- Spring set x1 (link)
- 8mm black/white heat-shrink tube 1m x10 (link)
- 8mm white heat-shrink tube roll 100m x1 (link)
- 40-pin 20cm jumper cables x20 (link)
- 40-pin 40cm jumper cables x20 (link)

Five MDF boards from an earlier order also arrived, so we can start assembly in earnest.

2. Turret cable management

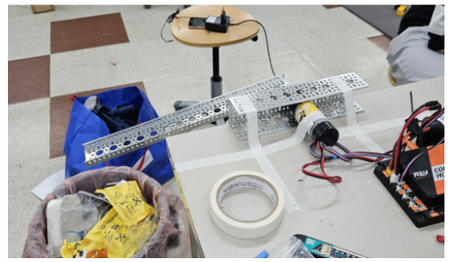
Before fully testing the turret control, we set the initial angles for the five servo motors and reassembled. Extended the servo wires and finished cable management using cable ties and sleeves made from cut rubber hoses.

Though the sleeves are clear, so the inside shows and isn't visually perfect, they're practical—wrapping evenly regardless of wire bundle thickness and protecting the cables. However, because of the sleeve form, there's some risk of snagging on protruding parts during movement.



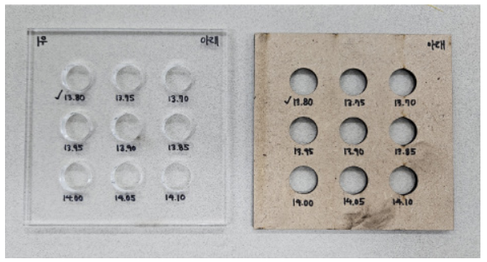
3. DC PID setup

Mounted a Gobilda DC motor on a U-frame. Attached a long U-beam and sonic hub to the motor shaft and secured it.



4. Laser cutting bearing hole test

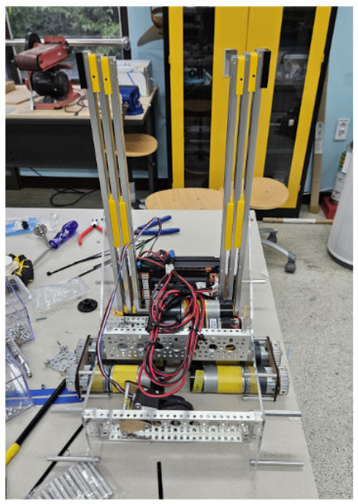
We mainly use 14mm outer-diameter bearings for 6mm/8mm shafts inside the robot. To fix them precisely in laser-cut parts, we printed test holes with slight diameter adjustments. After fitting the bearings, we determined that a 13.8mm cut gave the best fit.



2025.06.20. (Fri)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Driving test after assembly



Learning from early-season mistakes where we didn't account well for center of gravity, we decided to partially assemble heavier parts first and test movement in all directions. We mounted a total of 7 DC motors: 4 for the wheels, 2 for the vertical linear, and 1 for the horizontal linear. Then we placed the control hub and expansion hub symmetrically front and back, secured them with cable ties on mounts, and installed them between the inner plates.

However, due to a design error, two wheel holes were drilled farther apart than the actual timing belt length, so we had to pause. With no spare 5mm acrylic to cut new plates, we instead disassembled the inner plate, marked the correct positions, aligned the side plate on MDF, and used a laser cutter to make additional holes—avoiding a full reprint.

Also, since one of the timing belt pulleys from our previous competitions went missing, we had to print a new one, which delayed the work.

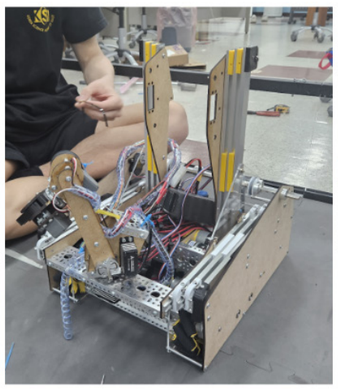
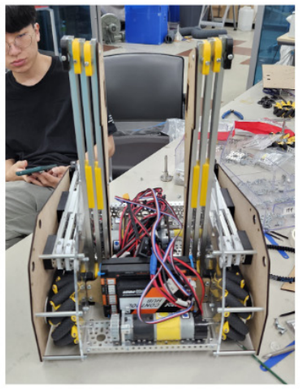
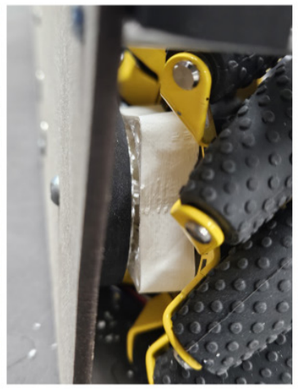
2025.06.21. (Sat)

구분 | SUMMER_EN_BUILDING
책임자 | 22 C.S.

1. Transportation booking

Booked transportation while sitting at a jjimjilbang. Rough route: Hanyoung Science HS → (large taxi) → Busan Station → (KTX) → Seoul Station → (Airport Railroad) → Incheon Airport Terminal 1.

2. Driving test after assembly

Continued from yesterday by assembling the horizontal linear and connecting it to the outer plates using standoffs and 8mm hex shafts. Inserted the mecanum wheels, DC motors, and odometry between the inner and outer plates as designed, making it ready for driving tests. Used 3D-printed spacers between the mecanum wheels and inner plate to ease assembly. Learned it's best to pre-wire hard-to-reach parts like the odometry. The photo below shows the first partial assembly.

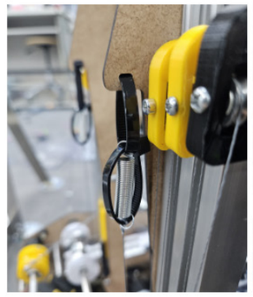
Since programmers weren't available, some builders tried running the driving test code themselves, going through trial and error. It took extra time, but eventually they succeeded and the driving test showed no major issues. Gradually increased DC motor output up to 100% to feel the max speed. However, one timing belt pulley on the wheels couldn't handle the torque and broke, so driving had to stop until the pulley could be replaced. The photo top right shows the broken pulley.

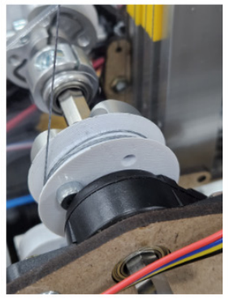
3. Vertical & horizontal linear string work

Printed 8 string spools on the 3D printer beforehand, each fitted with a sonic hub. Forgot to drill holes to tie the strings, so used a soldering iron to make them. Fixed the spool side facing the plate with button-head bolts. To keep things clear, we agreed to call them "winding spool" and "unwinding spool" depending on how they move when the linear extends. Both vertical and horizontal consider the unextended state as initial.

Measured minimum string length for both linears, added 10–15cm slack, then wound them. The winding and unwinding spools must wrap string in opposite directions.

• For the vertical linear: there's an inner plate between the winding and unwinding spools, and both sides must be strung to prevent tilting.
• The key: keep enough tension on both winding and unwinding strings, and balance tension between the two sides.






So we did it in this order:

1. Ran string from one side's winding spool through the pulley and tied it to the end of the linear, tightened strongly while fully extended (since tension drops as it extends).

2. Ran string from the other side's winding spool the same way, tied it off with a cable tie loop, and adjusted tension to match.

3. For the unwinding spools, tied them to the end effector in the fully extended position, then tightened with cable ties so the tension was set at the lowest point.

4. Repeated for the other unwinding string.

For the horizontal linear: both winding and unwinding spools are on a single sonic hub. Since there's almost no gravity load here, we only strung one side. This left the other side free, letting us secure a cable carrier later and making wire management cleaner.

4. Arena assembly

Assembled the field tiles and walls for the driving test. Built the center Submersible following reference videos. Some old tape markings had peeled off, so they'll need to be redone.

2025.06.22. (Sun)

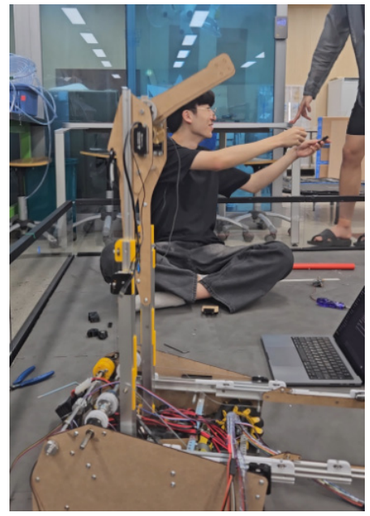
구분 | SUMMER_EN_BUILDING
책임자 | 22 C.S.

1. Australian visa issued

Applied for a tourist visa through the Australian ETA app. Did it after lunch, and the visa was approved in about 2 hours.

2. Packing prep

Planned to reuse the box from the last FTC trip, but this robot is taller, so had to switch to a bigger box. The small box is 16.28 kg, the large one 21.78 kg — just barely fitting under the 40 kg extra baggage limit.



3. Vertical/horizontal linear test

Finished stringing the vertical and horizontal linears and wrote a quick test code to check stability and max speed. Tested manually without encoder length limits, and under sudden load, the string and cable tie snapped and tangled. Spent about 3 hours redoing the vertical linear wiring.

2025.06.23. (Mon)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

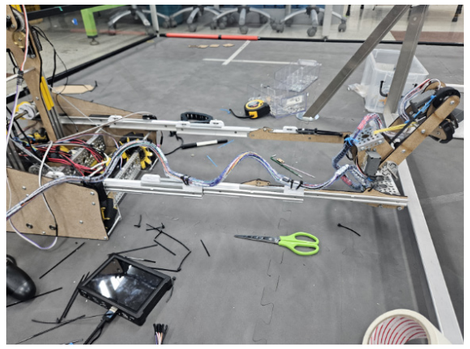
1. Vertical/horizontal linear encoder test

Tested how high the vertical linear extends by coding set values and checking movement. Measured encoder values at fully retracted and fully extended states, then programmed limits so it wouldn't exceed them.

Gradually increased speed from 0.05 up to 1, confirming it moved well even at full speed. Noticed, however, that faster speeds caused more slack. During tests, the string loosened, so we found the point where tension dropped the most—roughly mid-extension—and adjusted to pull the string tight there. This kept the string taut throughout the slide's motion and prevented it from slipping off pulleys.

Did the same for the horizontal linear: measured encoder values at full retraction and extension, then coded it not to exceed these limits.

2. Vertical/horizontal linear wire management



The main tools we had were 16 mm sleeves and cable carriers. Also cut rubber hose to organize wires on the horizontal linear. Since only one side of the horizontal uses pulleys, we wrapped rubber hose in coils on the other side, like a spring. This keeps wires from twisting and simply stacks up when the linear retracts, preventing tangles.

For the vertical linear, after reviewing other teams' designs, decided to use a cable carrier. Added wire clips on the carrier's side to ensure it folded in a consistent shape.

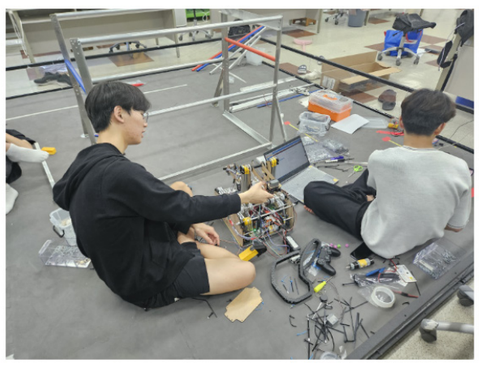
3. Wheel pulley replacement

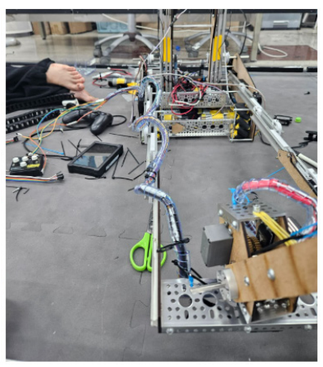
Printed a replacement for the broken wheel pulley, increasing infill from 30% to 70% for extra strength—heavier but sturdier. Since replacing it requires disassembling most of the chassis and the transfer isn't fully tuned yet, we decided to delay swapping it out until we upgrade the intake, outtake, and frame together.

4. Hub repositioning

To make wiring easier, shifted the Control Hub to better use the space. One side had most wires, while the other had free room since it didn't drive the linear. Moved the hub to balance space and ease cable routing.

5. Transfer test



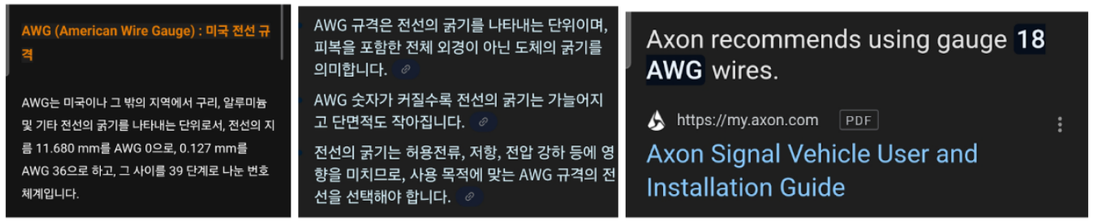


Before doing full transfer tuning, tested if moving samples and specimens between mechanisms was even feasible. Mapped motions to the triangle button on the controller and ran dry tests. Found that the precise angles of intake and outtake are crucial for reliable transfer. Adjusted angles by 0.02 increments to find the sweet spot, and plan to slightly modify the intake shape to reduce potential issues.

2025.06.25. (Wed)

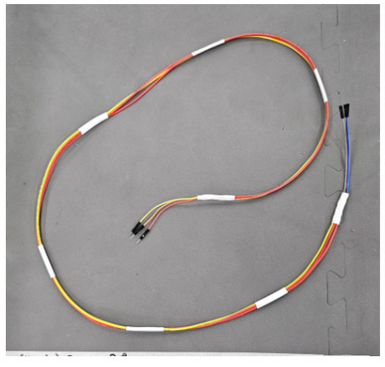
구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

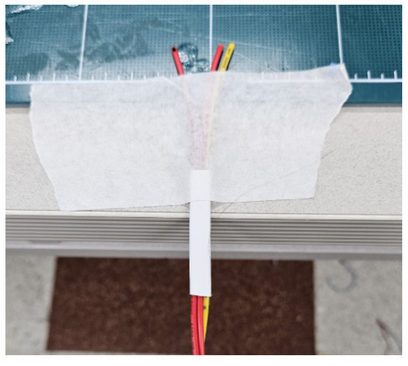
1. Made Axon cable extensions



Early this morning, tested servos using regular Arduino jumper wires. Found that with SRS tests, issues appeared from 7 cables at 20 cm, and even with just 1 cable at 40 cm. This confirmed it was a wire problem.

So cut 20 AWG (0.5 mm^2) wire to the needed lengths and made 5 Axon extensions. Measured the required length, made 3 wires of that length, fixed them together with super glue, then soldered 3 jumper wires to each end to complete one extension. Repeated this process 5 times.





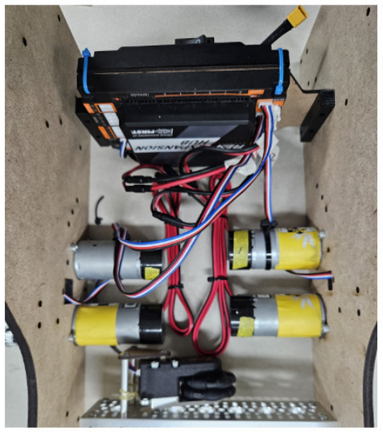
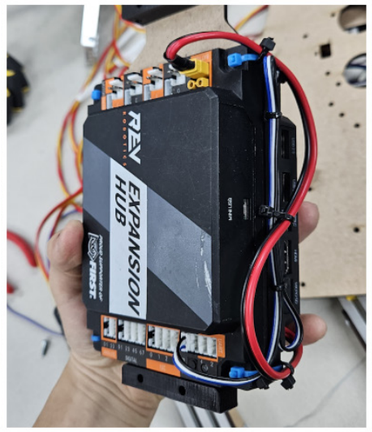
2025.06.26. (Thu)

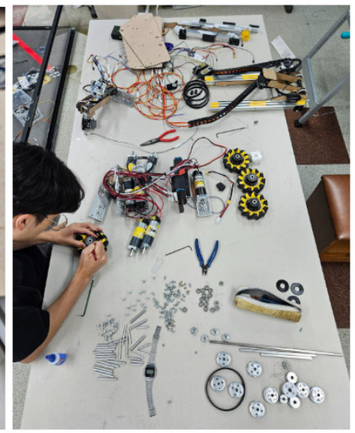
구분 | SUMMER_EN_BUILDING
책임자 | 22 C.S.

1. 1st reassembly

Started disassembling the robot at 8 PM and finished in just 40 minutes, but reassembly took much longer due to carefully organizing the wiring. Previously, motor power cables ran upward to the control and expansion hubs, but during reassembly, routed them under the frame to keep them mostly hidden. Connected the control and expansion hubs compactly and secured them between the inner plates, as shown below.

After reassembly, ran the vertical linear wires through the right side of the frame (relative to the horizontal linear direction) and temporarily fixed them at the center where the SPM will be mounted. Noticed the control hub power cable was hitting the horizontal linear's DC motor frame, so shifted the control and expansion hubs about 10 mm upward to clear it.



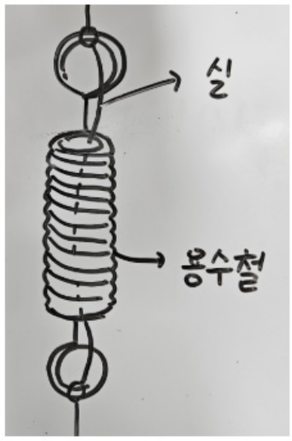
2025.06.27. (Fri)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Changes to linear string setup

Spent the entire day refining string and cable management. Developed a new method that avoids cable ties entirely on the end effector, using only string. Previously, the end effector either required a bulky spring setup or relied on cable tie tension, so it didn't work reliably. Now, by tying the string with slack at both ends of the spring and threading it through the spring, it stretches only as much as the slack allows, staying compact and neat.

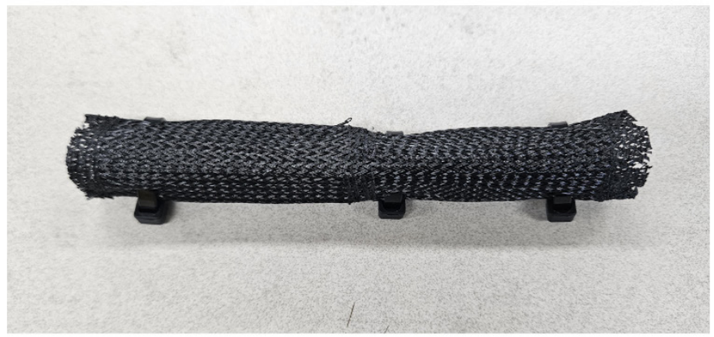
Also revised the string arrangement: originally, the winding pulley pulled directly, while the unwinding pulley was linked by cable ties and springs. This could be unstable during hanging, so switched to mounting the spring on the winding pulley and using the cable tie on the unwinding side to adjust tension.

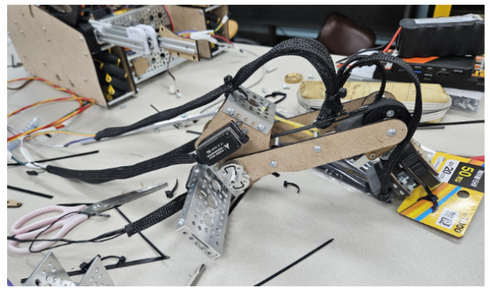
2. Wire management

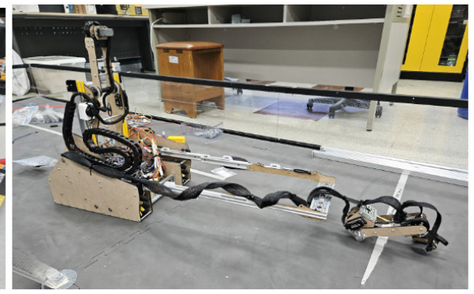
Finished the first reassembly by dawn, then took the day more slowly to work on horizontal linear cable management. Fortunately, the new black sleeves (16 mm, 25 mm, 30 mm) arrived. Unlike the Gobilda sleeves used before, these from Coupang frayed easily if not heat-sealed and were flat tube-style, making them trickier to work with. Used a soldering iron to seal ends, and inserted cable ties inside to help keep shape.

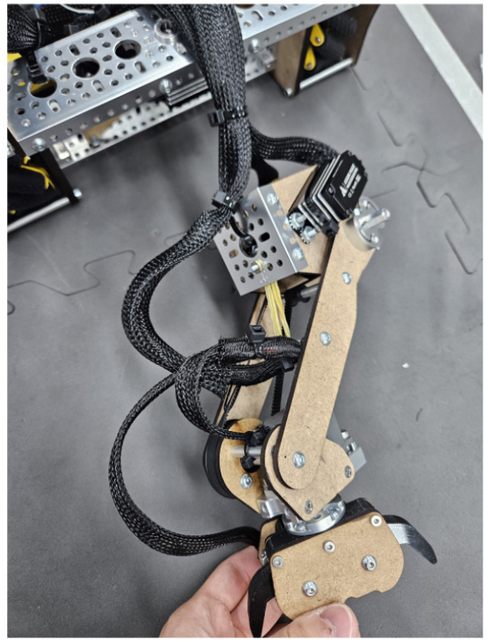
Disassembled the turret first to make threading easier. Cut the 30 mm sleeve to length, heat-treated it, and anchored it to an unused string pulley mount on one side of the horizontal linear, so wires stayed secure as it moved.

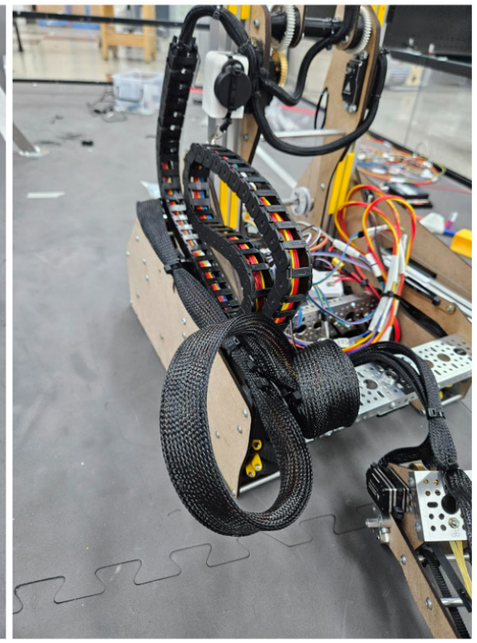


After routing the horizontal linear wires, fed them through holes in the inner plate to the SPM. The SPM port holes were too small for Arduino jumper connectors, so pre-installed 20 cm extensions on all Input and Output ports, then plugged servo lines into those. Found that some turret servos weren't connecting properly due to SPM contact issues, so plan to swap to proper 3-pin servo extensions soon. Also concluded that, like many foreign teams, mounting the Control Hub and Expansion Hub directly on the side plates might simplify cable management more than routing everything below.









3. Full system test

Ran all functions once. The transfer was surprisingly fast. However, kept running into a problem where the intake turret's servo screws loosened, making the whole assembly wobbly. Will need to secure this more firmly.

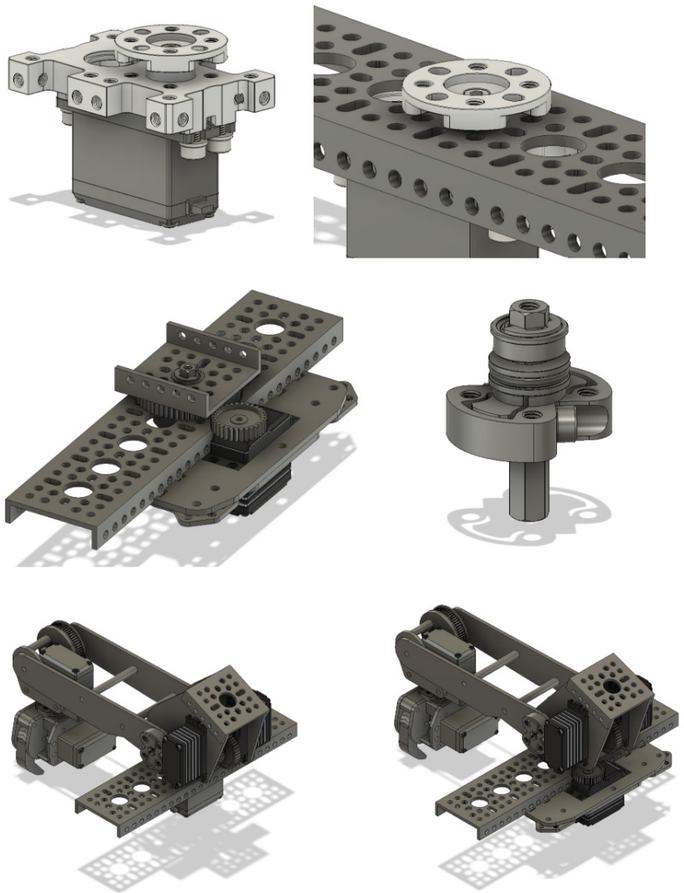
2025.06.28. (Sat)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.S.

1. Turret redesign

The original turret was entirely supported by a Gobilda servo block, which relied on a single basic bearing in the vertical axis. This meant all vertical load was borne by the servo mounting screws, making them loosen over time as the turret rotated.

To solve this, added a separate dead axle and incorporated both a thrust bearing (for vertical forces) and a standard bearing (for horizontal forces). This new design can handle both vertical and horizontal stresses — and turned out quite clean.

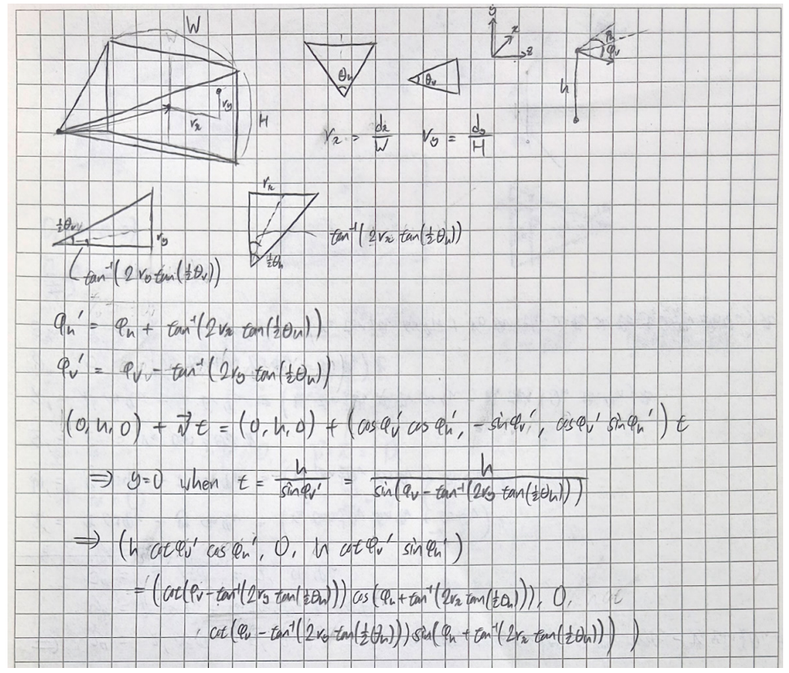


2025.06.29. (Sun)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Vision position conversion

The Limelight processes on-screen data well, but to use it in real-world coordinates, set up equations. Since samples only exist on the floor plane, derived a formula using the Limelight's height, angle, and FOV to convert image points into actual coordinates.



2. Field cleanup

Everyone stayed up all night yesterday and only went to bed around 5 AM, then collectively overslept and woke up at 4:30 PM. Before dinner, started thoroughly cleaning up the field.

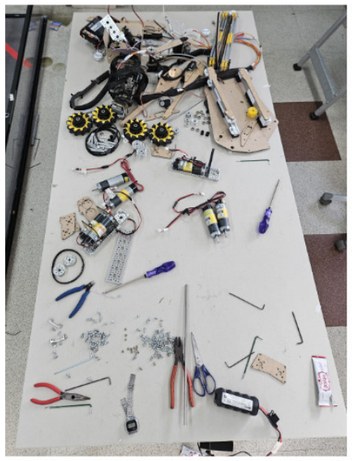
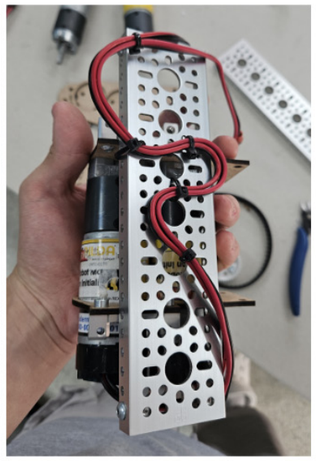
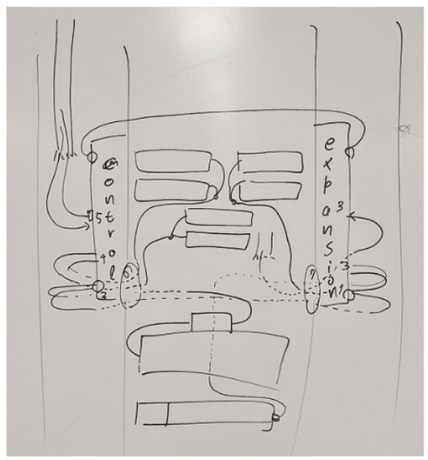
2025.06.30. (Mon)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.S.

1. Second reassembly attempt; failed (design flaw)

Started disassembling the robot for a second reassembly. Carefully took it apart, keeping the vertical and horizontal linear wires attached to the standoffs to preserve cable management. Secured the vertical linear DC motor wires as shown below, and made a new cable routing plan (bottom right photo). Unlike before, the control and expansion hubs would now be mounted on the side plates instead of the back, so routed wires through the inner plate holes and across the horizontal linear mounts to reach the hub ports.

However, ran into a design flaw: two drive motors ended up too close, causing the timing belts to cross. This meant we had to revise the design and reprint the side plates and motor mounts, delaying reassembly by a day.

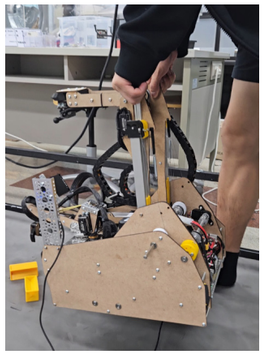
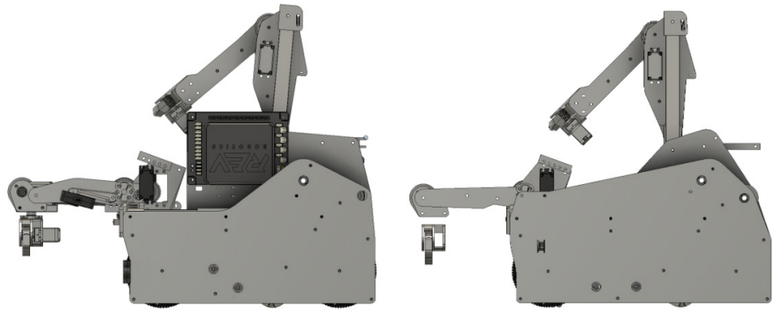




2025.07.01. (Tue)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

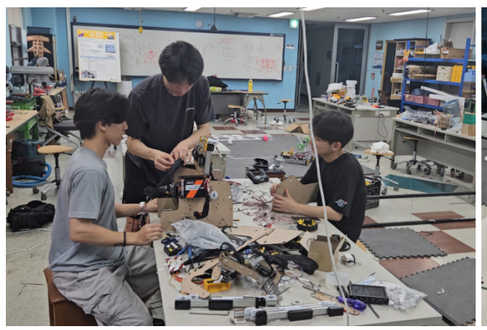
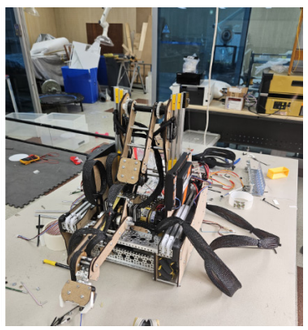
1. Hanging test and robot redesign

Tested hanging and found that with the current structure, hooking onto the bar wasn't easy. So redesigned the robot with the vertical linear tilted about 5 degrees. Also shortened the front-to-back length by around 2 cm. Adjusted the whole design so when depositing forward, the robot could sit flush against the submersible and place the specimen reliably. Additionally, moved the hubs from the inner side to the inside of the side plates to simplify wiring.

2. Second reassembly

After the KSASF opening ceremony, started the second reassembly at 10 PM. Took under an hour to disassemble and about 1.5 hours to put back together, but spent over 5 hours on cable management. Plan to optimize wires again during the final rebuild. Expect it'll be much easier once we switch to a custom aluminum frame, which will have more space for cables. Stayed up all night, reworked horizontal wire routing from noon to prep for KSASF booth demos—but without a coder, couldn't do a demo, and ended up going to sleep at 3 AM.

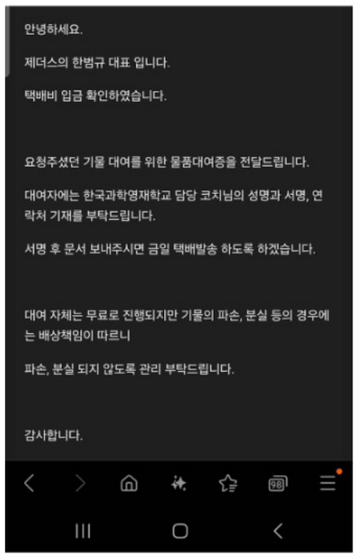



2025.07.03. (Thu)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.S.

1. Finding ways to borrow missing field elements

This season we only bought half the field, so we lacked enough samples and clips. Decided to contact teams that competed in the KRC to see if we could borrow or buy unused elements. Searched for Korean teams on FTC Scout (https://ftcscout.org/) and tried reaching out to nearby teams. Since FTC Scout only lists schools, called their school offices and admin departments for contact info. Inquired with SHT32 (Suwon Hightech HS), Ex Machina (Yesan HS), and ROG United (Sejong Academy for Gifted Science & Arts). Learned from ROG United that they didn't buy field elements but rented them from a company called Xeders (https://mall.xeders.com/). Contacted Xeders and they agreed to lend us the needed field elements free of charge—only paying 4,000 KRW for shipping. Sent them a list of the quantities needed by email.



2. Ordered required items

Because we moved the Limelight camera from the main body to the intake arm, now needed a USB C to A cable over 1.2 m long. Also ordered two spare cables on Coupang to replace unstable connections between the Driver Hub and gamepad.

이름	링크	개당 가격	개수	배송비	총 가격	용도
듀얼 90도 USB A to C 케이블	https://www.co	11,000	1	0	11000	Limelight - 컨트롤 허브 연결용
USB C to 5pin 케이블	https://www.co	4,130	2	0	8260	게임 패드 - 드라이버 허브 연결용
발포지 PE 폼 50T 1m x 2m	https://www.co	23,000	5	6,000	121,000	APOC 패킹용
구경모 세트	https://www.co	11,750	2	2,500	26,000	APOC
실	https://www.co	7500	1	0	7500	로켓배송 19800원 이상 무료배송

2025.07.04. (Fri)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

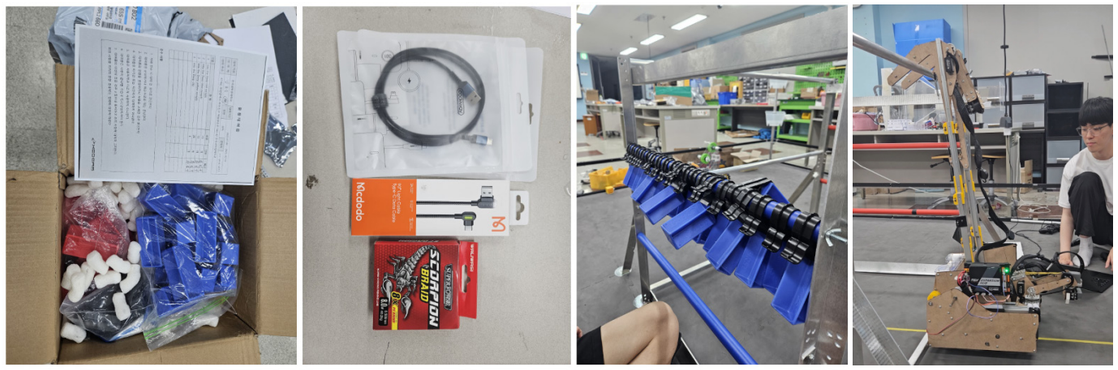
1. Replaced vertical linear mount

To properly test hanging after the second reassembly, swapped the old vertical linear mount for one with a hook that can latch onto the bar. During the last hanging test, only changed the mount itself to preserve wiring, leaving the servo in place. This time, found a way to remove the servo without disconnecting wires, so installed it together with the new hanging mount.

2025.07.05. (Sat)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.S.

1. Received packages



2. Hanging test

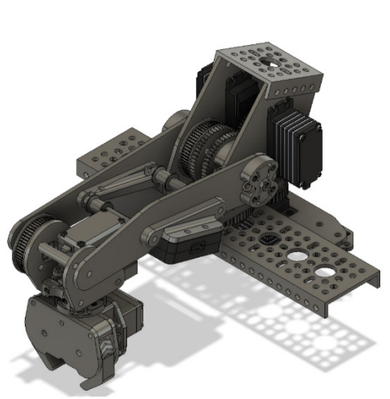
After the second reassembly, tested whether the 2nd Ascending worked as planned. Result: it worked well.

2025.07.06. (Sun)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

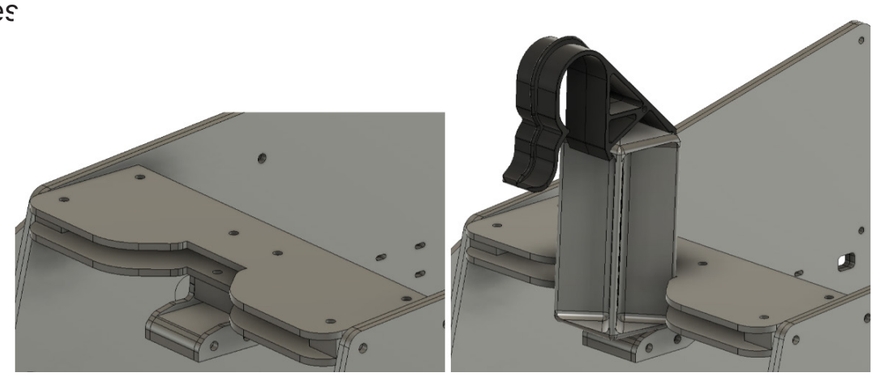
1. Limelight camera repositioning

Initially planned to mount the Limelight fixed on the main body, so it could detect samples from afar and pick one by extending the horizontal linear. But ran into a problem: when the linear extended, the turret blocked the samples. Inspired by Technoz's design, moved the camera onto the Intake Arm, solving the occlusion issue, though it narrowed the detection range. Decided to partly adopt Technoz's game strategy: the player roughly positions and orients the robot and extends the linear, then the camera fine-tunes to detect and grab the sample. Considered mounting the camera upright or on the side of the Intake Arm, and finally chose to embed it partially through the side plate of the Intake Arm.



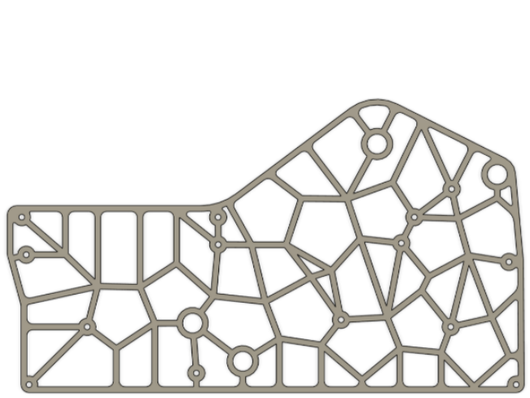
2. Liner design & assembly

To reliably grab specimens hanging on the wall, the robot's lateral position and height need precise adjustment. Since doing this slowly hurts cycle time, designed and attached a liner that aligns the specimen to the desired position in both axes

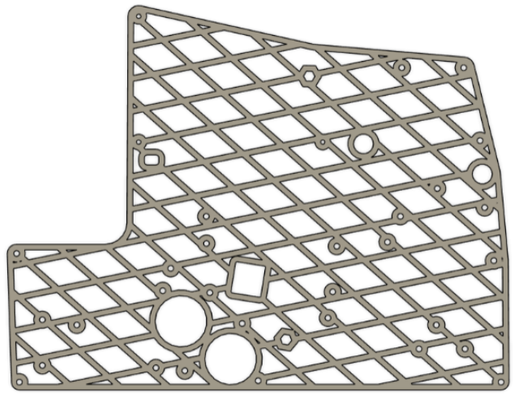


3. Aluminum custom frame design

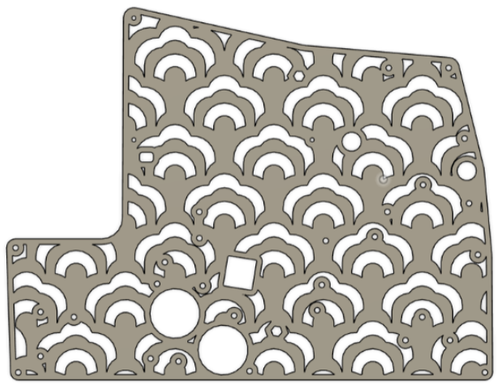
Haven't yet requested a quote from Korea Metal Laser, so still don't know the cost or lead time. Earlier KRC prep suggested it might take ~3–4 days. Since their HQ is in Sasang-gu, Busan, it could be faster to pick up directly if shipping delays.



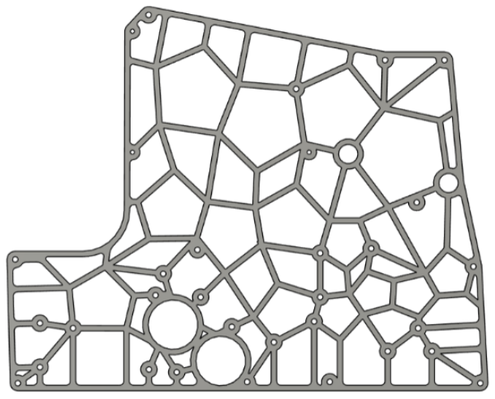
▲ OuterFrameR 패턴 디자인 (108g)



▲ InnerFrameR 패턴 디자인 (193.7g)



▲ InnerFrameR 패턴 디자인 (309.1g)



▲ InnerFrameR 패턴 디자인 (139.7g)

Aluminum has a density of ~2.7 g/cm³, roughly 5× that of MDF (~0.5 g/cm³). The aluminum parts would include: OuterFrameR, OuterFrameL, InnerFrameR, InnerFrameL, two VerticalMounts, two HorizontalMounts, and two DriveMotorMounts—7 types, 10 pieces total. Set thickness to 2.5 T, referencing overseas teams. Fully solid frames would weigh ~2 kg, so plan to cut out ~☰ of the area to lighten it.

Used mostly random shapes to create openings, but added pentagon patterns for better overall rigidity, arranging them centrally and drawing perpendicular lines from vertices to fill gaps. Tried to align bolt holes with pattern vertices and rounded edges near bolts for stronger support.

Thought about repetitive patterns to speed up cutting vs purely random fills, but smaller repetitive holes needed to maintain bolt support would add weight and look less clean. So stuck with this hybrid pentagon-random approach.

2025.07.07. (Mon)

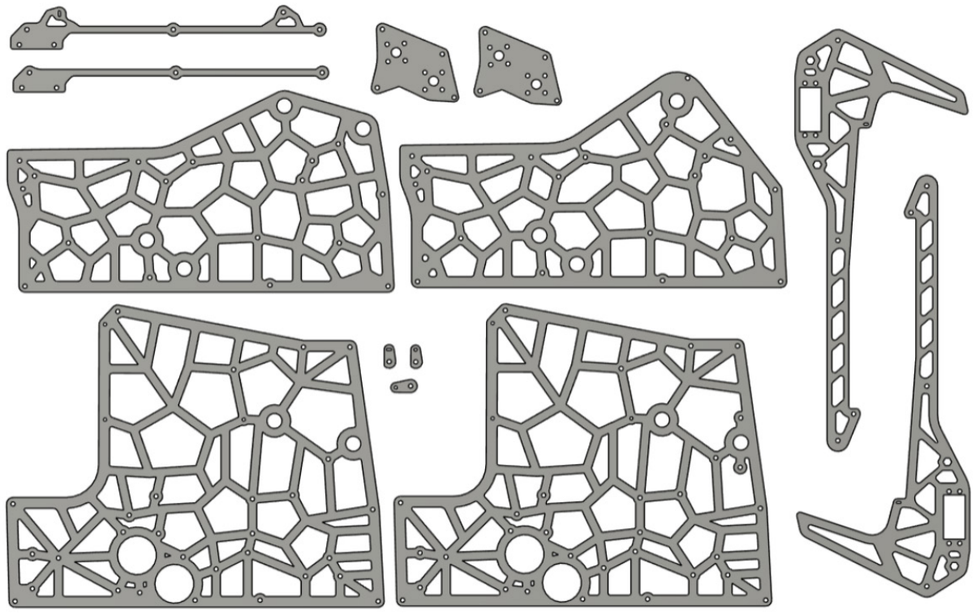
구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

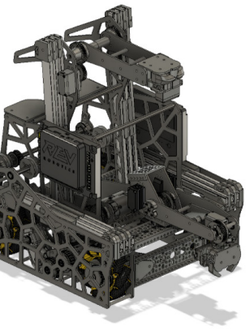
1. Field setup & TeleOP demo

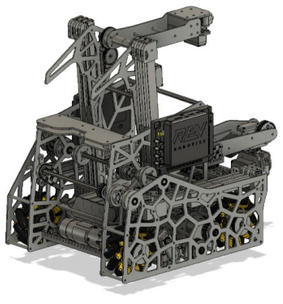
For the first time since staying on campus, ran TeleOP in a field-like environment. Didn't have red/blue masking tape to mark it precisely, but still tested picking up samples and specimens, then rotating the turret right to see if samples were accurately delivered to the human player. Re-tuned the vertical linear's PID so the deposit claw lined up exactly with the specimen hanging height, and confirmed it could hang a specimen on the submersible while also grabbing samples.

2. Aluminum custom frame design & order

Decided to go with Korea Metal Laser for aluminum cutting—same company contacted during KRC prep. Last time, tight timing and a weekend meant we couldn't get the frames in time. This time, determined to use aluminum, so skipped online quote forms and called directly. Explained our schedule and confirmed that if we ordered Mon–Tue, we could pick up Thu–Fri. They requested .dwg files.







That evening, finalized which parts to make in aluminum, prioritizing high-load areas: main frame plates, vertical/horizontal linear mounts, and DC motor mounts.

Previously designed with 4 mm outer edge width and 4 mm internal beams, but compared to overseas teams felt too thin. Redesigned with 10 mm outer edges and 8 mm internal beams. Made the lower half of the inner plates denser with more pattern holes since it bears more load, and left the upper part sparser (just holds control/expansion hubs). Designed vertical/horizontal linear mounts with 4 mm throughout.

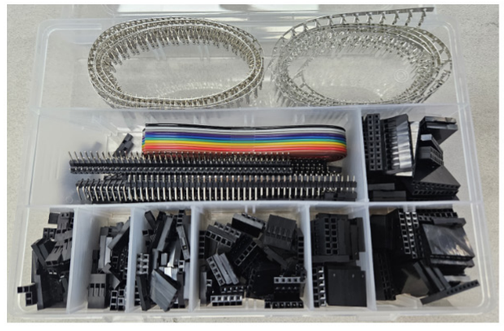
Then arranged all parts on a single sketch, exported to .dwg (note: personal Fusion360 licenses can't export to .dwg, so used the education license).

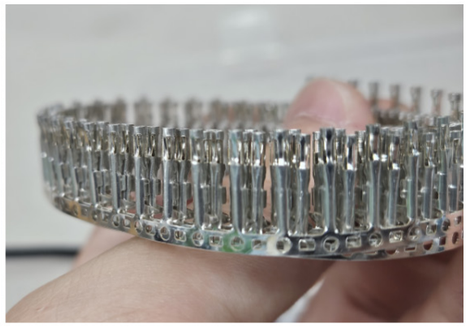
3. Static stress simulation

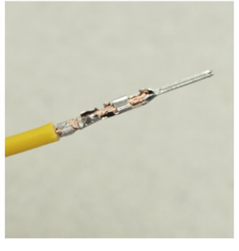
Curious about Fusion 360's simulation tools, so followed a quick tutorial and ran static stress tests to see how the aluminum frames would distribute load.

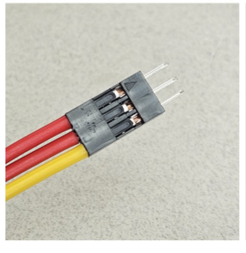
4. Connector pin header kit / making servo cables

After a meeting with TC Team (FTC team in Korea), decided that instead of soldering connectors onto Dupont cables, it'd be cleaner to use proper pin header terminal kits and crimp directly onto AWG wires. Ordered a kit on 7/6 (Sun) via Coupang, which arrived that evening. Stripped each wire, inserted into crimp terminals, secured with a clamp, then slotted them into 3-pin housings—made 20 AWG servo extensions at exactly the lengths needed. Over 5× faster, neater, and simpler than soldering. Used this to rebuild and reconnect the IntakeClaw cable.









2025.07.08. (Tue)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Aluminum custom frame quote & order

At 7:30 AM today, sent the final .dwg files compiling all the needed part drawings to the email listed on the Korea Metal Laser website (hml727@naver.com) to request a quote. During Into the Deep KRC prep, even with zero cutouts, the quote came to around ₩120,000, so expected something like ₩200,000–300,000 this time. Was surprised to see almost no price difference despite all the pattern holes.

Asked about payment; they said both prepay and pay-on-pickup were fine. Decided to pay on-site with a corporate card when we pick it up Thursday afternoon.

2. Packing prep: cutting PE foam

Started cutting the delivered PE foam sheets. The big box's inside is 555 mm × 555 mm × 580 mm, so cut ten 50 T PE foam squares of 555 mm × 555 mm to stack tightly and fill it. Need at least one sheet at the bottom and top to cushion, and should also brace the frame inside so load spreads evenly (not only on the vertical linear) in case the box flips during shipping. Since the large box leaves space around the robot, plan to pack bolts and small tools there too. Last FTC season used 50 T styrofoam from DDC, but it didn't cut cleanly with a box cutter and needed a hot wire cutter—plus made a mess with dust everywhere. This time chose PE foam, which cuts neatly with a knife.

3. Replaced IntakeClaw motor: Gobilda → Axon

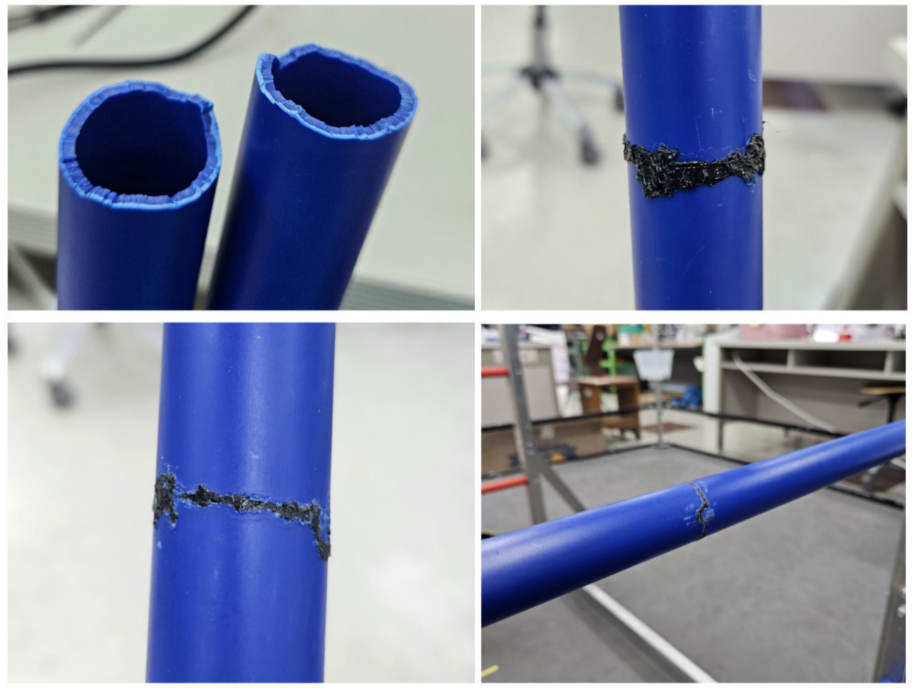
The Limelight detection algorithm can't guarantee 100% accuracy, and even when it detects, sometimes the claw still fails to grab. From the driver's perspective, the claw and sample are blocked by the robot, so it's hard to see if the claw actually grabbed it. Initially tried solving this via vision, but then remembered the Axon servo's built-in encoder values can always confirm if it's holding something. So swapped the Gobilda servo on the Intake Claw for an Axon servo to test this approach.

2025.07.09. (Wed)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. Repaired broken Submersible part

Last night, one of the blue rods on the Submersible that holds specimens snapped under pressure. Used a soldering iron, wire, and scrap cable ties to fix it. Like in YouTube tutorials: heated the wire with the soldering iron to fuse and reinforce the break, carved grooves along the crack line with the iron, then melted cable tie plastic into them. Trimmed excess plastic with a knife and finished by sanding.



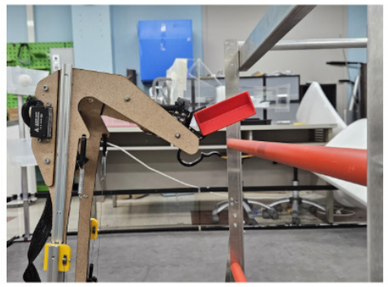
2. Removed SPM & reorganized wiring

At first, everything worked fine without the SPM, so planned to add it later just to boost overall servo speed. Midway, suspected some Gobilda servos were underpowered, so hooked only them to the SPM. But the SPM kept acting up — near the expansion hub, it vibrated, lost connection, and its LEDs flickered constantly. Couldn't find the root cause, so decided to remove the SPM entirely.

3. Reset servo positions

Helped the programmer re-tune servo angles and linear heights for grabbing specimens with the Intake Claw, transferring them, and finally hanging them on the rear rods.





4. Robot sign design

Robot signs must be at least 16.5 cm × 6.4 cm for the number area, show the alliance color as the background, and be placed on two sides or adjacent faces. We chose the sides, meaning two red and two blue signs.


Most teams print on coated paper or use stickers, then mount on holders or bolt them directly. For this season's KRC, decided to attach signs with Velcro for easy removal.

Set the number area at 16.5 cm × 6.4 cm, added 6 mm top/bottom margins and 10 mm side margins, making the final size 17.7 cm × 8.4 cm. Plan to print on coated paper and wedge it into place without making separate sign plates. Also designed the outer plates to be minimally bolted for quick removal during string work or rewiring. Still finalizing the actual sign design.

2025.07.10. (Thu)

구분 | SUMMER_EN_BUILDING
책임자 | 22 K.H.

1. TeleOP Test

To achieve high scores during TeleOp, we conducted practice matches. We ran a total of 12 sessions, and after discussions with our Alliance, we determined it was most efficient to focus on either specimens or samples, so we practiced both. Each TeleOp practice lasted for 2 minutes. If about 10 seconds remained, we attempted a 2nd ascending; if within 5 seconds, we attempted a 1st ascending; and if there was no time left, it was considered a failure.

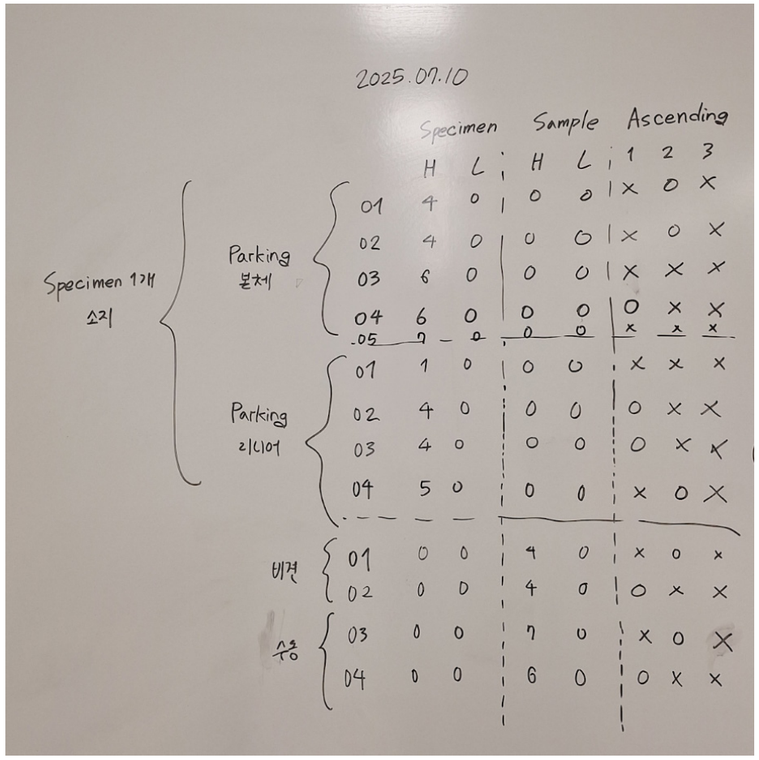
Specimen practice

For specimens, there are generally two possible situations after autonomous ends. The first is when the entire robot chassis is parked. In this case, we can immediately grab the specimen or, if there is none with the human player, retrieve it right away. The second is when only the linear slide is extended to touch the parking line. In this case, as TeleOp begins, we first need to retract the linear slide and rotate, so the operation flow changes slightly.

Sample practice

For samples, quick movement is crucial. Since moving efficiently is key, we tried to minimize the buttons needed along the route to pick up and place samples. Because combining straight and rotational movements can be confusing, we optimized paths to rely as much as possible on straight movements alone. Also, since vision detection was not working reliably at the moment, we practiced manually controlling the turret.

1st overall TeleOp results



Type	Avg Score
specimen (Parking – body)	60.6
specimen (Parking – linear)	40.25
sample – Vision	41
sample – Manual	61
Total Avg	51.38

ENGINEERING NOTE

PROGRAMMING

APOC 2025

This section is about last season's programming using OnBot Java.

APOC 2025
Engineering Note

Part 1: Preparation

Participating in the FTC South Korea Regional Competition, the South Korea Championship, we applied a variety of new technologies for the first time, such as the FTC Dashboard, vision systems, Roadrunner, and PID controllers. However, as these were our first attempts, we were not yet proficient with these technologies and encountered multiple issues. To address this, we took time beforehand to learn and practice these techniques. This allowed us to utilize these features more effectively and implement more advanced code, resulting in better performance.

1. Development of a Precision PID Class

A PID controller is a classic feedback control method used to drive a system's output toward a desired setpoint. PID stands for Proportional, Integral, and Derivative, each responding to the current error, accumulated error, and rate of change of error respectively to generate a control signal. Proportional control (P) reacts strongly to larger errors but can leave a residual steady-state error. Integral control (I) accumulates persistent errors to eliminate residuals over time. Derivative control (D) reacts to rapid changes to dampen oscillations and stabilize the system. A well-tuned combination of these allows precise and stable control across many systems. The PID feedback value is defined by:

$$MV(t) = k_P E(t) + k_I \int_0^t E(t)dt + k_D \frac{dE}{dt}$$

In the last competition, we first employed a PID controller to precisely operate the linear slide, but faced several issues.

Ideally, the feedback can take any value. In reality, however, it is limited. For example, a DC motor is capped at a maximum output of 1.0. Thus, the full feedback cannot always be applied. This disproportionately impacts the integral term; if the error decreases more slowly than expected, the accumulated integral builds excessively, leading to overshoot. To address this, we added limits to the PID feedback output and designed the class to exclude excessive contributions from integration.

When the target position changes, the accumulated error should be reset. Previously, this had to be manually specified by the user of the object, which led to mistakes. We resolved this by storing the previous target value and comparing it, automatically resetting integration only when the target changes.

Ideally, the error function is continuous. In practice, it is discretized over time. This caused the derivative term to fluctuate: after a large change, a small subsequent change would reduce the derivative term's impact, increasing variation again, leading to oscillations. We solved this by applying a low-pass filter to the derivative term, so it depends not only on the immediate derivative but also on historical values. The filter defaults to 0.1 and can be adjusted via the FTC Dashboard.

2. Development of Custom Hardware Classes

The hardware classes provided by the FTC SDK only cover low-level manipulation, making it difficult to control complex systems directly. To overcome this, we developed custom hardware classes to encapsulate advanced features, reduce bugs, and improve code readability.

Smart Servo

A standard servo only allows specifying the target position, not the speed. We created a Smart Servo class to specify not just the target position, but also the time over which it moves. This overloads the setPosition function to accept duration as an additional argument, enabling smooth motion over a given period. The class uses an internal timer to incrementally move the servo. While individual updates can be called on each servo, we store all Smart Servos in a static dynamic array, allowing a single updateAll function to update all Smart Servos at once, minimizing errors or performance drops due to missed or duplicate updates.

Additionally, in parts like robot arms where multiple servos handle large torque, they must move in sync. Controlling two servos simultaneously every time is prone to human error, and if physically linked, miscoordination can go unnoticed yet stress the hardware dangerously. To prevent this, our Smart Servo class includes a synchronization feature. The static list of servos allows identifying and pairing servos by name. When synchronized, child servos are fully controlled by the parent, and any attempt to manually move a child throws an exception to prevent bugs. We also allow configuring offsets and directions so that when the parent moves, each child correctly reflects its own characteristics.

Smart Motor

Unlike servos, DC motors control only power, making precise position control difficult without PID. Writing a separate PID for each DC motor is cumbersome. Our Smart Motor class combines a DC motor, an encoder, and a PID controller into a single object. It auto-initializes, allows setting target positions with setPosition, and computes PID feedback via update to move smoothly. We also provide functions to adjust PID coefficients and DC motor settings. Since keeping PID always on increases computation and hardware requests, we allow turning it on or off, freezing the last power, or manually setting power to optimize computing resources.

Like Smart Servo, each Smart Motor is stored in a static dynamic array to enable batch updating, avoiding missed or duplicate updates. It also supports synchronization for physically linked DC motors, applying the parent motor's power—including PID output—directly to the children and preventing independent manipulation.

Smart Gamepad

The FTC SDK's default gamepad object only provides booleans for button states and values for triggers. To detect a new press, release, or hold, you must compare the current and previous gamepad states, which is tedious and reduces readability. Failing to handle this correctly can result in unintended repeated executions. Thus, we created the Smart Gamepad class.

KRC's previous version managed gamepads but required manually fetching and comparing values, not improving readability much. We advanced it to process everything with intuitive chained calls.

For example:

```
SmartGamepad.isPressed (
        smartGamepad1.gamepad().circle,
        smartGamepad1.prev().circle
)
```

The new Smart Gamepad takes an FTC gamepad object in its constructor, updates automatically via update, and allows checking conditions with concise, chained calls.

SmartGamepad has internal classes for Button, Trigger, and TouchPad, each created as temporary objects initialized with current and previous states to provide intuitive methods. For example, Button has isPressed(), isReleased(), isHeld(), isFree(), Trigger adds getValue() and getDifference(), and TouchPad includes Finger classes for each finger's touch data.

This means you can do:

smartGamepad1.buttonCircle().isPressed()

to efficiently check gamepad state.

3. Scheduler

When controlling robot behavior, you often want to delay between sequential actions. Typical procedural programming uses sleep, which is dangerous in a robot: it halts the main thread, freezing all simultaneous processes and risking major failures. Thus, asynchronous programming is essential.

Java threads are a common solution, but misuse can degrade performance or introduce memory-access bugs. Hardware operations have overhead that can cause fatal issues if accessed concurrently. Writing a fully thread-safe system is complex and sometimes inefficient, so we developed a Schedule class to achieve asynchronous programming by periodic calls on the main thread.

The static Schedule class accepts tasks of type Runnable and their execution time, storing them in a priority queue. Each update checks the internal timer to execute tasks precisely. It also supports BooleanSupplier conditions for conditional tasks, stored in a linked list to check conditions on each update. Thus, we implemented time-based asynchronous programming on the main thread.

For special needs, we also created thread-based concurrency functions. These allow higher performance but must be used cautiously in thread-safe scenarios.

4. Telemetry System

Telemetry is crucial for debugging by outputting the robot's current state. As our robot grew more complex, we needed standardized, easily controlled telemetry outputs. Thus, we developed a Telemetry System.

The static Telemetry System initializes once during the OpMode startup by assigning telemetry, then updates collectively during OpMode updates. This removes the need to pass telemetry instances around, preventing missed outputs or null pointers.

It provides two main output types:

- addClassData accepts a class name, data caption, and value, formats them into telemetry messages, and lets you toggle outputs by class name for easy filtering.
- addDebugData provides temporary debug outputs, controllable via a global toggle for streamlined debugging.

5. Limelight

At KRC, we previously performed sample detection using webcams and the FTC SDK's standard or OpenCV-based pipelines, which rely heavily on color. This requires frequent recalibration under different conditions and is prone to interference from similar-colored objects.

To overcome this, we purchased a Limelight for the APOC competition. We studied the Limelight 3A official docs and tested both color-based and neural-network-based detection pipelines. Color pipelines were easy to configure but struggled when samples overlapped. This confirmed the need for neural network detection.

We attempted to train our own model using Roboflow resources and Limelight's Colab training code, but the limited free resources led to insufficient training (about two hours), producing a model that could not distinguish samples well. It often crashed the Limelight due to high computational demands. Possible reasons include:

- inadequate training on our specific samples,
- mismanaged Limelight training protocol,
- or potential mistakes in our understanding of the training process.

We factory-reset the Limelight and explored using models from other teams. We found one by team 6165 MSET Cuttlefish trained on 6000+ samples with ~91% accuracy. Applying this model to our Limelight achieved high-accuracy sample detection. We thank team 6165 MSET Cuttlefish for publicly sharing their AI model.

Part 2: Code Architecture Plan

To write efficient, maintainable, and readable code, we carefully structured our project, dividing it into Part, Feature, and OpMode packages based on roles.

1. Part Package

A robot consists of many subsystems. Handling everything in one object is complex and inefficient. We divided the robot into parts, each managing its own hardware.

A valid Part:

- must operate independently of other parts,
- should control around 3-6 hardware components,
- must have exclusive control over its hardware.

Inside the Part package, each subsystem has its own sub-package containing:

- **Part Class**: manages hardware, stores state, initializes, updates, provides emergency stop, and exposes command() and adjustment().
- **Constants**: stores all constants—positions, values, hardware names, default settings—centralized for easy maintenance and FTC Dashboard tuning via @Config.
- **Commands**: defines high-level commands, manipulating hardware directly. External calls use partInstance.command().someCommand().
- **Adjustment**: extends an abstract Adjustment class, offering modes like ADJUST_SERVO, ADJUST_PID, and PRINT_ENCODER_VALUE to inspect and fine-tune hardware via FTC Dashboard.

Main Part objects implement the Part interface, enforcing init, update, and stop methods, which OpModes handle polymorphically. Parts abstract operations via Commands, achieving OOP goals of polymorphism, encapsulation, and abstraction for robust maintenance.

With Adjustment objects, we can quickly inspect and tune servos, view real-time encoder data, and safely adjust PID controllers, streamlining not only robot operation but also tuning.

2. Feature Package

This contains global utility classes like Smart Servo, Smart Motor, Smart Gamepad, Scheduler, Telemetry System, and PID. Designed per SOLID principles, these objects prevent code duplication and bugs, offering strong maintainability benefits.

3. OpMode Package

This houses the main OpModes, subdivided into:

- **auto**: final AutoOp classes executing preplanned protocols.
- **tele**: final TeleOp classes driven by gamepad control.
- **manipulate**: runs Parts in Adjustment mode for tuning.
- **test**: for testing features or parts, excluded at build to simplify the Driver Station.

Each OpMode cleanly manages Parts without directly controlling hardware, relying only on abstracted Commands to prevent unforeseen issues.

Class Name	Role
Part (Name)	Directly manages the hardware of the part and stores the part's state. Handles hardware allocation, initialization, updating, and also manages emergency stop functions. Provides its own Commands object through the command() function. Also provides its own Adjustment object through the adjustment() function.
Constants	Stores and manages various constants for the part. This includes hardware positions and states, as well as various robot parameters, hardware names, and default hardware settings. By consolidating these constants into a single file, maintenance becomes easier. Supports value adjustments in the FTC Dashboard through the @Config annotation.
Commands	Defines various commands for the part. Most of the part's movements are abstracted and implemented through the functions of this class. It takes the part class as a constructor argument and, being in the same package, can directly manipulate the part's hardware and internal variables. Externally, commands are executed in the form partInstance.command().someCommand().
Adjustment	Directly tests and adjusts the hardware values of the part. Inherits from an abstract class named Adjustment and has three modes: ADJUST_SERVO, ADJUST_PID, and PRINT_ENCODER_VALUE. Allows real-time monitoring of servo positions and PID control states through the FTC Dashboard, enabling precise adjustments by modifying constants. Also provides real-time encoder values to check the current state through encoder output.

Package	Role
auto	This is the package that contains the AutoOp classes for the final robot. It can control all parts and operates according to pre-planned protocols. Various AutoOp classes can be created depending on the situation.
tele	This is the package that contains the TeleOp classes for the final robot. It can control all parts and operates based on gamepad inputs. Various TeleOp classes can be created depending on the situation.
manipulate	Closely related to the Adjustment function, this package includes OpModes used to adjust servo positions, PID values, and various constants for each part. The OpModes in this package run all Part objects in Adjustment mode.
test	Contains various test OpModes, which exist to partially test features or parts and can be freely created and removed without special constraints. When the final robot is completed, this package can be excluded from the build using gradle to simplify the Driver Station screen.

The static Schedule class accepts tasks of type Runnable and their execution time, storing them in a priority queue. Each update checks the internal timer to execute tasks precisely. It also supports BooleanSupplier conditions for conditional tasks, stored in a linked list to check conditions on each update. Thus, we implemented time-based asynchronous programming on the main thread.

For special needs, we also created thread-based concurrency functions. These allow higher performance but must be used cautiously in thread-safe scenarios.

4. Telemetry System

Telemetry is crucial for debugging by outputting the robot's current state. As our robot grew more complex, we needed standardized, easily controlled telemetry outputs. Thus, we developed a Telemetry System.

The static Telemetry System initializes once during the OpMode startup by assigning telemetry, then updates collectively during OpMode updates. This removes the need to pass telemetry instances around, preventing missed outputs or null pointers.

It provides two main output types:

- addClassData accepts a class name, data caption, and value, formats them into telemetry messages, and lets you toggle outputs by class name for easy filtering.
- addDebugData provides temporary debug outputs, controllable via a global toggle for streamlined debugging.

5. Limelight

At KRC, we previously performed sample detection using webcams and the FTC SDK's standard or OpenCV-based pipelines, which rely heavily on color. This requires frequent recalibration under different conditions and is prone to interference from similar-colored objects.

To overcome this, we purchased a Limelight for the APOC competition. We studied the Limelight 3A official docs and tested both color-based and neural-network-based detection pipelines. Color pipelines were easy to configure but struggled when samples overlapped. This confirmed the need for neural network detection.

Part 3: Development of Part Objects

1. Development of the Part Interface and Adjustment Abstract Class

We first developed the Part interface, which serves as the foundational framework for creating each main Part class named after its respective part. By creating this interface, we were able to leverage polymorphism, one of the powerful features of object-oriented languages. The Part interface enforces the definition of the following functions:

init: a function for initializing the Part, recommended to be called by the OpMode's init() function.
update: defines special tasks that need to be performed regularly within the Part class.
stop: defines tasks that should be executed in emergencies or when the OpMode ends.

Additionally, each Part class must include its own command() and adjustment() functions to provide the command and adjustment features of its Part package. However, since the return types of these functions vary by class, they were not defined in the interface. By having all main Part classes inherit this interface, we were able to store them in a list of type Part interface and process them all at once through a loop, without needing to call functions individually in the OpMode.

Next, we developed the Adjustment abstract class. As described above, the Adjustment class is included in each Part class and provides three main features: Adjust Servo, Adjust PID, and Print Encoder Value. Since these functions must be applied consistently across all Adjustment classes, we created an abstract class to be inherited, thereby maintaining code consistency and avoiding redundant code.

The Adjustment abstract class includes an enumeration defining the Adjustment States, an activate() function to control activation, and an update() function for regular updates. Tasks to be performed by each adjustment mode, which must be independently implemented by each class, were defined as protected abstract functions to be created within each specific class. These protected abstract functions include adjustServo(), adjustPID(), and printEncoderValue(), with their contents defined in each Part's Adjustment class to specifically control the hardware needed by that part.

2. Development of the Drive Package

This is the Part package that manages the robot's drive system. It handles translational and rotational movements of the robot via the gamepad and supports localization and trajectory following using Road Runner.

By finely controlling the mecanum wheels, it enables free movement in all directions—not just forward, backward, left, and right—and allows simultaneous rotation and translation. Furthermore, by utilizing localization, it supports field-centric driving, enabling the robot to move within an absolute coordinate system regardless of its orientation, providing a powerful driving system distinct from previous implementations.

The main class of this part, the Drive class, inherits the Part interface and implements init, update, and stop functions. It also includes getter functions to directly access TrajectorySequenceBuilder, SampleMecanumDrive, and the Commands object. In addition, it contains a Constants class for defining various constants and a Commands class for controlling the Drive part within the Part package. Unlike typical Part packages, this package does not include an Adjustment class since special adjustment is unnecessary.

The list of commands provided by this part is as follows.

CMD Name	Explanation
drive	Receives x, y, and omega components to move in the specified direction.
stop	Stops the movement of the robot's drive system.
followTrajectory	Receives a TrajectorySequence and executes that sequence.
changeDirection	Reverses the forward direction of the robot.

2. Development of the Intake Package

Developed the Intake Package, which controls the robot's intake part responsible for picking up samples. This part consists of the robot's horizontal linear slide, turret servo, arm up-down servo, wrist up-down servo, wrist orientation servo, and intake claw servo. Each of these hardware components is finely controlled to support the movements required by various commands.

All hardware manipulated by this Part is declared using the Smart Servo or Smart Motor class, allowing it to leverage the powerful features provided by those classes. This enables precise extension of the linear slide and exact servo positioning to accurately pick up samples at desired locations.

The Intake package includes an IntakeState enumeration to represent the current state of the intake part. This makes it possible to check the current state, restrict movements to avoid unnecessary or dangerous operations, and quickly determine the next state based on the current one for efficient operation.

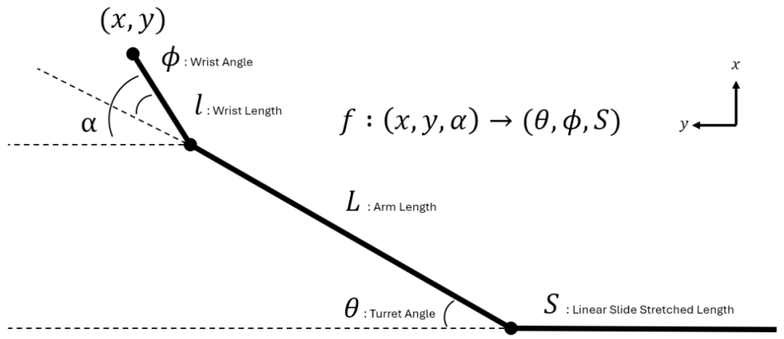
The main class of the Intake package, the Intake class, provides in addition to the functions defined by the Part interface, a command() function that returns the Commands object, an adjustment() function that returns the Adjustment object, a state() function that returns the current state of the intake part, and an isLinearSlideInside() function that checks whether the linear slide is fully retracted inside the robot. The Constants class stores the names of the hardware belonging to the intake part, servo positions for each movement, linear slide extension limits and target positions, and delay values for various steps. The Commands class defines various commands so that precise hardware control can be achieved through a single explicit command call from the outside.

The list of commands provided by this part is as follows.

CMD Name	Explanation
openClaw	Opens the intake claw.
closeClaw	Closes the intake claw to its standard position.
closeClawMaximum	Closes the intake claw to its maximum tightness.
ready	Sets the intake part to a ready state for picking up a sample. Moves only the arm section without moving the linear slide. Changes the intake state to READY_FOR_PICKUP.
automaticTarget	Receives color data, detects the sample of the corresponding color, moves to that position, and performs pickup in one sequence. Uses the vision class to perform sample detection and controls the intake part based on the results. Optionally accepts a cautious boolean value; if true, it assumes potential collision with a wall, reducing the operating range and moving more conservatively. Changes the intake state to AUTO_DETECTING.
automaticTargetForAllianceSample	Executes the automaticTarget function based on the current alliance information defined in a global value, optionally accepting a cautious value.
automaticTargetForYellowSample	Targets and picks up a yellow sample using the automaticTarget function, optionally accepting a cautious value.
pickup	Lowers the arm to pick up a sample positioned below. When first grabbing, closes the claw to its maximum, then uses the axon servo's analog input to determine how tightly the claw is closed to check if the sample was successfully grasped. If successful, closes the claw to its standard position, proceeds with the readyForTransfer process, otherwise performs discard and then repeats the ready process. After performing the grab operation, changes the intake state to PICKED_UP.
discard	Opens the claw to discard the held object and returns to the ready state. Changes the intake state to READY_FOR_PICKUP.
compactReady	Similar to the ready function but includes fully retracting the linear slide inside the robot. Changes the intake state to READY_FOR_PICKUP.
readyForTransfer	Performs all transfer preparation movements except raising the intake arm to prevent collisions with the submersible when immediately executing transfer. Changes the intake state to READY_FOR_TRANSFER.
transfer	Executes the transfer process for the intake part. Raises the intake arm, opens the claw at the right timing, and when the transfer is complete, runs ready. During transfer, changes the intake state to TRANSFER, and after finishing, returns to READY_FOR_PICKUP by executing the ready command.
drop	Drops the held sample into the observation zone. Moves the robot arm as far right as possible, releases the sample midway, and throws it to the back right through inertia. During the drop process, changes the intake state to DROP_SAMPLE, and after finishing, returns to READY_FOR_PICKUP by executing the ready command.
setPosition	Receives x, y, and angle values, and when the intake part is fully retracted, moves the linear slide and turret so that the claw's coordinate becomes (x, y) with the turret's rotation center as the origin. The angle sets the wrist orientation, taking the turret's rotation into account. How each hardware component is controlled based on x, y, and angle will be described later.
setPositionDelta	Receives changes in x, y, and angle, adds them to the current x, y, and angle values, and moves to that position.
movePositionXY	Receives x and y values and moves to that position, keeping the current angle.
movePositiondXdY	Receives changes in x and y, moves to that position, keeping the current angle.
rotateOrientation	Receives an orientation value and sets angle to that orientation, maintaining x and y.
rotateOrientationDelta	Receives a change in orientation, moves to that orientation, maintaining x and y.

The most essential function of this part is the ability to receive a specific coordinate (x, y) with the turret's rotation center as the origin when the horizontal linear slide is fully retracted, and move the claw precisely to that position. It also accepts an orientation value as angle to set the claw's orientation independently of the turret's rotation. Through a mathematical approach, our team was able to implement this feature, enabling a powerful sample detection system as well as a manual control system.

First, we can simplify the current system into the following form.



Our goal is to achieve the claw's target (x, y, alpha) values, where L and l are known through measurement. This means we need to determine theta, phi, and S from the diagram. Considering the above diagram, the claw's coordinates x, y, and the wrist orientation alpha can be expressed using l, L, S, phi, and theta as follows.

$$x = L\sin\theta + l\sin\alpha$$
$$y = L\cos\theta + l\cos\alpha + S$$
$$\alpha = \theta + \phi$$

Since L and l are constants and x, y, and alpha are given values, we can rearrange these equations to ultimately derive expressions for theta, phi, and S. From this, we can determine the positions of each servo and the extension distance of the linear slide. The rearranged equations are shown below.

$$\theta = \sin^{-1}\left(\frac{x - l\sin\alpha}{L}\right)$$
$$\phi = \alpha - \theta$$
$$S = y - L\cos\theta - l\cos\alpha$$

By implementing these equations in functional form, we created the setPosition(x, y, angle) function. When actually putting this into practice, we also accounted for the 5.2-degree angle offset between the claw and the turret's rotation center by applying this offset to the calculated theta and phi values, allowing us to determine the claw's exact coordinates relative to the turret's rotation center.

However, the real robot cannot reach every possible x, y coordinate, since the linear slide's extension length, the turret's rotation radius, and the arm length are all limited. Therefore, the actual setPosition code was written to incorporate these constraints: after first calculating the theta, phi, and S values, if any of these violated the constraints, they were clamped to their limit values, and then the current x, y, and angle were recalculated through inverse computations.

Through this entire process, our team developed a method to move the claw to a specified (x, y, angle) position. This made it possible to precisely adjust the claw's location simply by calling this function. It also enabled us to accurately move to coordinates tracked by the vision system when detecting a sample, and during manual operation, allowed the driver to move to exact positions by adjusting only x and y values without worrying about turret and wrist orientation angle changes.

2. Development of the Deposit Package

Developed the Deposit Package, a Part package designed to place samples or specimens onto a basket or into a submersible. This package controls hardware such as the robot's vertical linear slide, deposit arm servo, and deposit claw servo. Each piece of hardware is declared using the Smart Servo or Smart Motor class, enabling precise control of movements to safely and efficiently deposit samples.

The Deposit package includes a DepositState enumeration representing the current state of the deposit part. This enum allows real-time monitoring of the deposit part's state and restricts operations to only those permitted in that state, preventing unnecessary or hazardous actions. It also enables quick determination of the next action based on the current state, allowing for an efficient sample deposit process.

The main class of the Deposit package, the Deposit class, provides various additional features beyond the functions defined by the Part interface. These include a command() function that returns a Commands object for issuing commands to the deposit part, an adjustment() function that returns an Adjustment object for fine-tuning deposit operations, and a state() function that returns the current state of the deposit part. The Constants class stores the names of the hardware components needed for the deposit part, servo and motor positions for each operation, linear slide extension limits and target positions, and delay values for each step. This allows precise control of deposit-related hardware through a single external command.

The commands provided by this part are as follows.

CMD Name	Explanation
openClaw	Opens the deposit claw.
closeClaw	Closes the deposit claw to its standard position.
closeClawForSpecimen	Closes the deposit claw to a position for gripping a specimen.
rest	Resets the deposit part to a state ready for transfer. Sets the deposit state to REST.
transfer	Receives a sample or specimen from the intake part. Changes the deposit state to TRANSFER, and after the transfer process completes, sets it to LOAD_SAMPLE or LOAD_SPECIMEN depending on the object received from the intake part.
poseForSpecimenPickup	Adjusts to a posture for directly picking up a specimen with the deposit part. Changes the deposit state to READY_FOR_PICKUP.

pickupSpecimen	After loading a specimen, executes poseForHighSpecimenScoringForward() and sets the deposit state to LOAD_SPECIMEN.
poseForDiscard	Shifts to a posture for discarding the object loaded in the deposit. Changes the deposit state to READY_FOR_DISCARD.
discard	Internally executes poseForDiscard(), discards the loaded object, and sets the deposit state to REST.
poseForLowBasketScoring	Controls the deposit part for low basket scoring. Executes only if the state is LOAD_SAMPLE via addConditionalTask, and afterward sets the deposit state to READY_FOR_DEPOSIT_BASKET.
poseForHighBasketScoring	Controls the deposit part for high basket scoring, similarly restricted by state, and then sets the deposit state to READY_FOR_DEPOSIT_BASKET.
scoringBasket	In the basket scoring pose, opens the claw to place the sample into the basket and returns to the original state, setting the deposit state to REST.
poseForHighSpecimenScoringForward	Controls the robot to position it for placing a specimen onto the submersible from the front. Executes only if the deposit state is LOAD_SPECIMEN, and afterward sets the state to READY_FOR_DEPOSIT_SPECIMEN.
poseForHighSpecimenScoringBackward	Controls the robot to position it for placing a specimen onto the submersible from the rear. Executes only if the deposit state is LOAD_SPECIMEN, and afterward sets the state to READY_FOR_DEPOSIT_SPECIMEN.
scoringSpecimen	In the specimen scoring pose, opens the claw to separate the specimen from the robot and returns to the original state, setting the deposit state to REST.
ascendingReady	Extends the linear slide above the height of the submersible to perform Level 2 ascending.
ascend	Retracts the linear slide to suspend the robot from the submersible while the linear slide hook is latched onto it for Level 2 ascending.

5. Overall Testing and Debugging

After implementing all of the above features, we conducted tests to ensure each function operated correctly. First, we ran an AdjustOpMode with adjustments activated for all parts, allowing us to adjust each servo's position, the maximum extension distance of the linear slide, and the extension distances used in each movement, then set these as constants. This process was carried out efficiently using the FTC Dashboard, enabling us to quickly determine values at runtime and apply them directly into the code.

Next, we created a Test TeleOp. This TeleOp mapped all commands to the gamepad without any state-based restrictions for each part. By running this, we verified that each command functioned properly, adjusted delay values for each section, and debugged any malfunctioning areas to ensure every command executed correctly.

During this process, we discovered a new bug related to synchronization. Specifically, in Smart Servo or Smart Motor, synchronized hardware sometimes did not move with the parent hardware but remained stationary. The key feature of this problem was that it did not occur when the robot code was built and the OpMode was executed for the first time, but consistently appeared when rerunning the OpMode afterward. Initially, we suspected a hardware issue, but given this consistent pattern, we identified it as a code problem. Investigation revealed that because static memory only resets when the robot fully shuts down and reboots, values set by previous OpModes persisted. Synchronization locates the parent hardware by searching for its name in static lists of Smart Servo and Smart Motor objects. As static memory remained, previous hardware objects were not destroyed but stayed in the list, causing synchronization to link to incorrect hardware and fail to move together. We resolved this by ensuring these lists were also initialized whenever the OpMode was initialized.

Additionally, we found minor issues that were immediately fixed by updating the code. Through this process, we completed the Part code to a level that could be expected to operate flawlessly without errors.

Part 4: Vision Feature Development

Sample Detection
→ Position Detection Algorithm
→ Sample Orientation Detection Algorithm
→ Develop Test Op
→ 1st Orientation Modification
→ 2nd Orientation Modification (SUG's Algorithm)

Our team attempted to use sample detection with Limelight to accurately pick up samples. We first tried machine-learning-based sample detection.

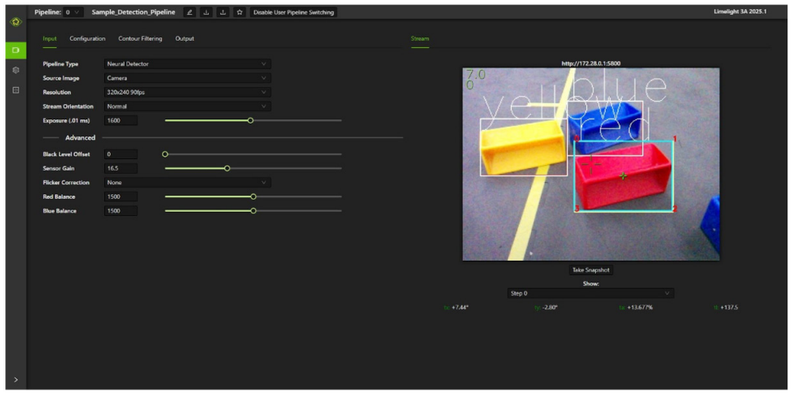
1. Sample detection and position identification using Limelight's Object Detection

Our team attempted sample detection using the model tested during the preparation phase in Part 1. Limelight returns the offset angles of a detected sample relative to its front-facing vector on the x and y axes. We studied mathematical methods to calculate the sample's position on a plane using this data.

By drawing a diagram of the current situation in a three-dimensional coordinate system and applying trigonometry based on the angles, we derived the following final equation.

$$x = \frac{h \sin \phi_x}{\tan \phi_y}, \quad y = \frac{h \cos \phi_x}{\tan \phi_y}$$

Based on this equation, we output the position of the sample detected by the Limelight and compared it to the actual value, confirming that we could detect the sample's position with high accuracy within less than 1 cm. Therefore, our team decided to use this algorithm to determine the sample's position.



2. Determining sample orientation through Limelight's color filtering

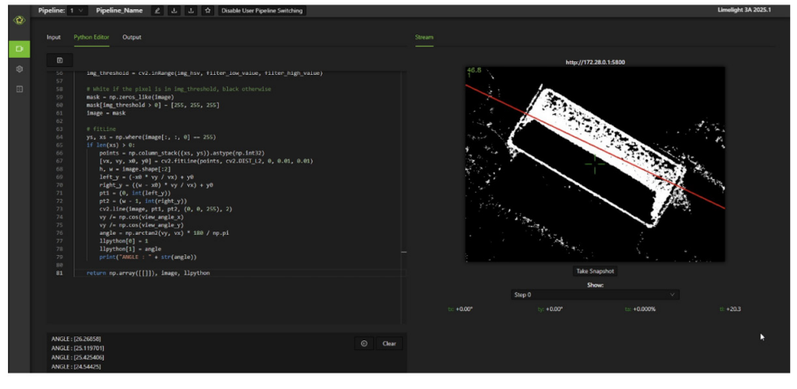
The algorithm using Limelight's machine-learning-based detection to identify the sample's position showed very high accuracy. However, we also needed to determine the sample's orientation, and unfortunately, the detection algorithm did not provide information about the sample's angle. For this reason, we decided to use color filtering alongside detection to determine the sample's orientation.

The reasons our team initially tried color-filter-based neural network sample detection were as follows:

- Color filtering is heavily influenced by the environment and requires tuning depending on conditions.
- Because multiple samples with similar colors are placed together, it is difficult to filter only a specific sample when viewed from the side, making it prone to interference and reduced accuracy.

We addressed this issue by leveraging the results of the neural network-based sample detection. When sample detection is performed, the result includes not only the position vector of the object but also the rectangular boundary of the detected object. By using this, we could extract only the specific targeted sample area and perform color filtering exclusively on this region, which we expected would yield much higher accuracy.

We implemented custom color filtering using Limelight's python snapcode pipeline. By loading the targeted sample's region as an llrobot input, we masked all areas outside this region to black. Then, we identified the HSV color range corresponding to the sample and created a filter, applying it so that the filtered areas turned white while non-matching areas remained black. Finally, to determine the sample's orientation, we analyzed the distribution of the white areas. By applying the least squares method to draw a trend line through the white pixels, we found that this line closely matched the sample's orientation. By calculating the slope angle of this line, we were able to determine the sample's orientation.

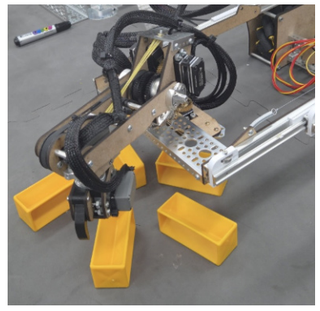


This method showed quite high accuracy. Even if the filtering was not perfectly performed, since it ultimately relied on observing the trend of the white points created by the filter, as long as most of the sample was correctly filtered, these points dominated, causing other noise to be ignored, allowing us to determine the sample's exact orientation within about 5 degrees.

However, when the camera viewed the sample at an oblique angle, perspective caused the sample's angle to be measured smaller than its actual value. To solve this, we applied a mathematical mechanism similar to the one used to calculate the sample's position, multiplying the slope of the trend line representing the orientation by x and y weights based on the sample's position.

At first glance, this approach seemed to achieve high accuracy, and when compared to the actual orientation, we confirmed that it extracted a significantly improved angle over the previous method.

3. Development of Vision Test OpMode and testing of the vision algorithm



To test the vision algorithm we had built, we developed a Vision Test OpMode. The first algorithm we created continuously calculated the sample's position and angle through a loop and output these values via telemetry. Using this, we initially analyzed the accuracy of the vision algorithm.

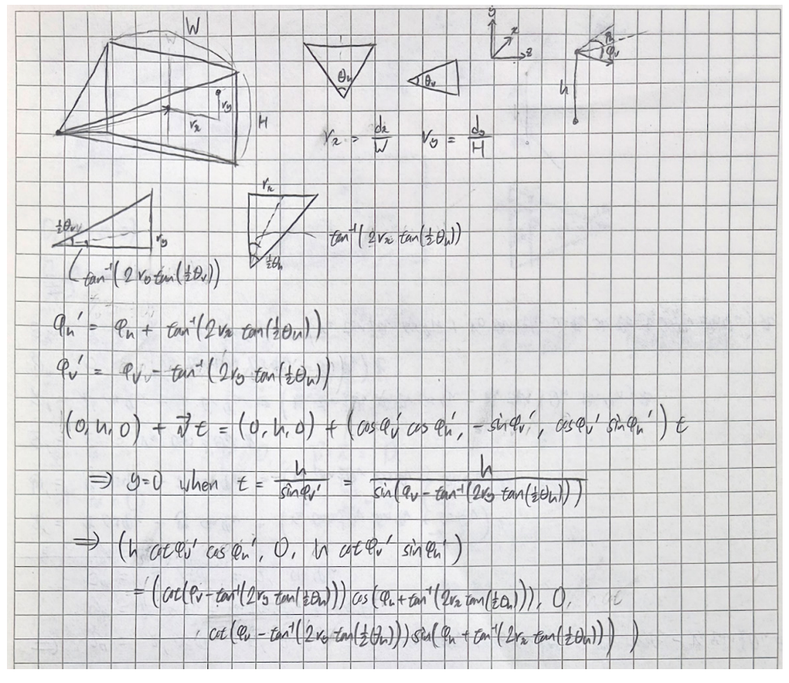
Afterward, to more rigorously evaluate the effectiveness of this algorithm, we developed a Detection Test OpMode. This OpMode continuously extracted sample data through the vision system and then used intake commands to move to the detected position and attempt to pick up the sample. With this, we were able to verify whether the sample could be accurately picked up and test its real-world performance.

As a result, we identified inaccuracies in the position and orientation measurements. By recognizing these trends, we applied appropriate offsets and adjusted various parameters such as the height of the Limelight, the length of the arm, and the rotation radius of the turret. Through this process, we also identified issues within the algorithm and explored solutions to address them.

4. First modification of the sample orientation detection algorithm

Testing the algorithm revealed slight inaccuracies in determining the sample's orientation. To address this, we re-approached the estimation of the sample angle using a mathematical method.

Our team derived equations to convert points on the Limelight image into real-world coordinates by using the Limelight's height, angle, and field of view. Then, from the line calculated by the original algorithm, we extracted two points not far from the center of the sample, converted these points using our calculation method, and obtained a new line representing the sample's orientation. Through this approach, we were able to determine the sample's angle more accurately.



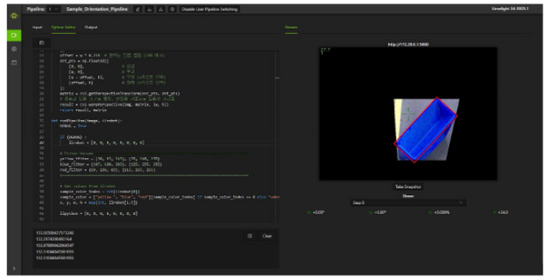
5. Second modification of the sample orientation detection algorithm

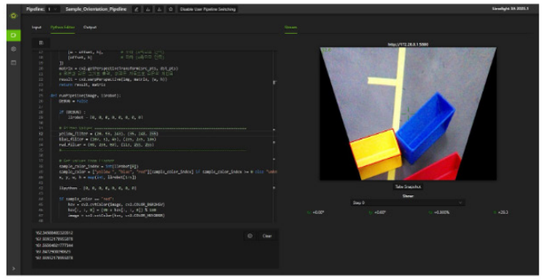
However, despite this mathematical approach, we still observed multiple cases where the detected orientation differed from the actual orientation depending on the situation.

To identify the root of the problem, we examined it from various perspectives and found that the underlying issue was the assumption that the trend line obtained through the least squares method accurately represented the actual direction of the sample. In reality, there was a definite discrepancy between the line passing through the true center of the sample and the trend line, meaning that no matter how precisely we adjusted this line through mathematical transformations, it inevitably differed from the real orientation. This also led to situations where the algorithm produced completely incorrect sample orientations, highlighting the need for a revision.

To solve this, our team decided to use OpenCV's perspective transform feature. By applying this, we could transform the camera's view into a top-down perspective. We then replaced the least squares method with OpenCV's minRect function to find the minimum bounding rectangle around the largest contour of the filtered image, allowing us to measure the sample's orientation more accurately.

While this method required more precise filtering and thus had some limitations, it offered higher accuracy by directly focusing on the actual sample area, unlike the least squares approach that was somewhat sensitive to noise. Additionally, by converting the camera image into a top-down view, we could determine the actual sample orientation without needing complex mathematical transformations or highly precise coordinate measurements.





6. Development of the Vision class

To use vision information effectively, we developed a Vision class dedicated solely to handling Limelight vision functions. This class abstracts the detailed calculations, Limelight pipeline switching, and data processing involved, allowing other parts of the code, such as Part classes, to easily utilize vision capabilities.

The Vision class returns data in the form of a class called Sample. This class encapsulates the sample's color, position, and orientation, making it straightforward to pass this data around.

Because the Limelight has its own internal CPU, it performs computations independently of the robot controller. Thus, receiving results from the Limelight requires explicitly requesting them, and changing the pipeline involves sending a request to the Limelight and then waiting until it processes this and generates new results. Simply using a while loop to wait would block control of other parts of the robot during this time, which could lead to serious accidents. To solve this, we designed it so that after making a request, the creation of the Sample data is deferred until the Limelight has sent back the desired results.

To implement this, we defined internal states within the Vision class. When a request for data is received, tasks such as changing the Limelight pipeline or analyzing results are handled through the Vision class's own update method, which is called regularly. This ensures other parts of the code continue to operate normally while waiting. We also actively used the scheduler to execute code when certain conditions were met, effectively implementing delayed operations.

7. Conclusion

Through the above process, we developed a method to accurately detect samples using the Limelight and created a Vision class to efficiently utilize vision capabilities. This enabled the robot to autonomously recognize samples, move to their locations, and pick them up without direct control from the driver, significantly enhancing performance in matches and greatly improving the accuracy of the autonomous period.

Part 5: OpMode Development

1. Introduction to OpMode Class

Content..

2. Development of TeleOpMode

Content..

ENGINEERING NOTE

BUILDING KRC 2025

This section is about the FTC 2024-25 season, which we competed in the Korea Robot Championship.

This chapter is mostly written in Korean.

Introducing UI

시즌 캡틴 이름의 뒷글자를 따서 로봇 이름을 짓는 TALOS 팀의 전통에 따라, 캡틴 홍정우의 이름으로 로봇 **"우이"**를 명명했습니다.

우이는 독창적이면서도 빠른 이터와 팔 설계를 통해 효율적으로 점수를 낼 수 있는 구조를 가지고 있습니다.

로봇 사진 + 설계도 → 설명

25309

Vertical Linear + Deposit Claw
180도 회전하는 팔이 연결되어
High Basket과 Specimen이 모두 가능함.

Control / Expansion Hub
로봇의 서보, 모터 등을 모두 연결.
OnBot Java로 코딩함.

Servo Power Module
서보에 충분한 전력을 공급하는
동시에 과전류로부터 보호함.

REV Hex Motor
수평 리니어를 빠르게 확장.

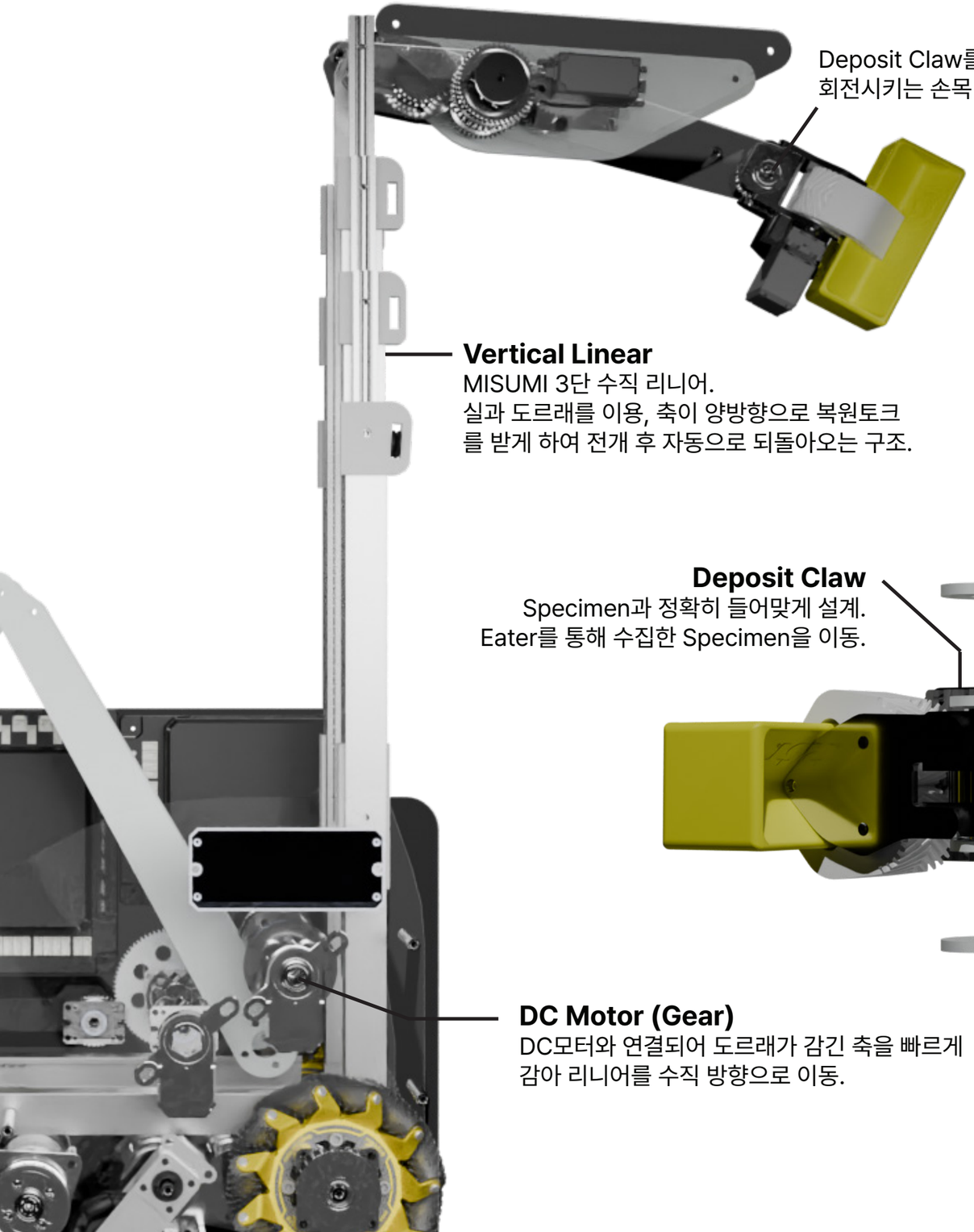
Horizontal Linear + Eater
집게 방식 active intake로 속도 극대화,
Deposit Claw로 전달.

Mecanum Wheel
Belted Drivetrain 구조로 연결,
전후좌우 모든 방향으로 이동.

본 3D 모델은 22 김수기가 Fusion 360으로 모델링한 설계도를 Blender로 채색한 결과물입니다.

해당 모델을 바탕으로 다양한 영상을 제작하여 공유할 수 있었습니다.

Horizontal Slide + Deposit Arm



Deposit Claw를
회전시키는 손목

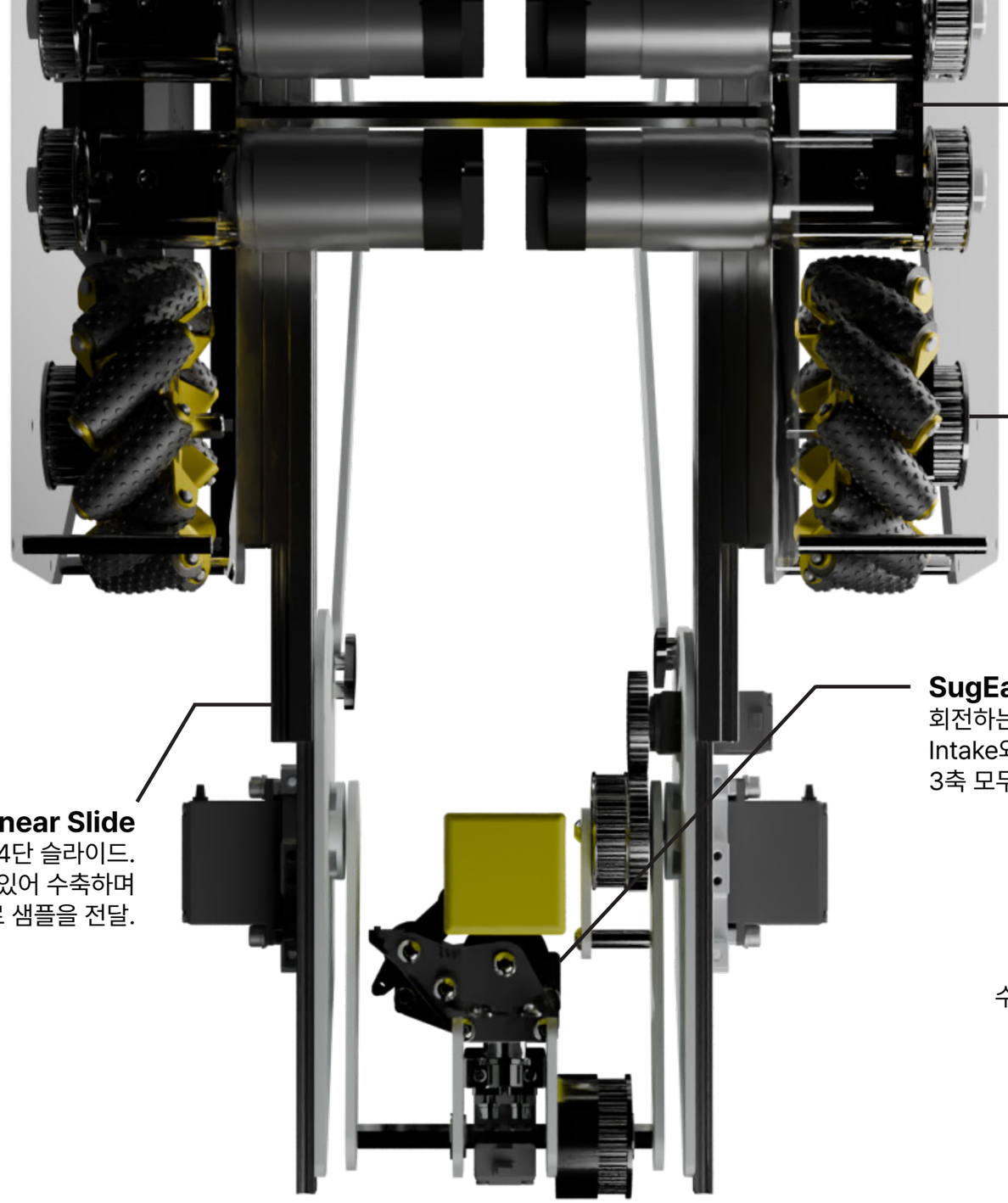
Vertical Linear
MISUMI 3단 수직 리니어.
실과 도르래를 이용, 축이 양방향으로 복원토크
를 받게 하여 전개 후 자동으로 되돌아오는 구조.

Deposit Arm
Deposit Claw가 연결되어 있음.
수직 리니어에 고정되어 서보 2개를
사용해 180도 회전.

Deposit Claw
Specimen과 정확히 들어맞게 설계.
Eater를 통해 수집한 Specimen을 이동.

DC Motor (Gear)
DC모터와 연결되어 도르래가 감긴 축을 빠르게
감아 리니어를 수직 방향으로 이동.

Deposit Arm 전체를
회전시키는 모터



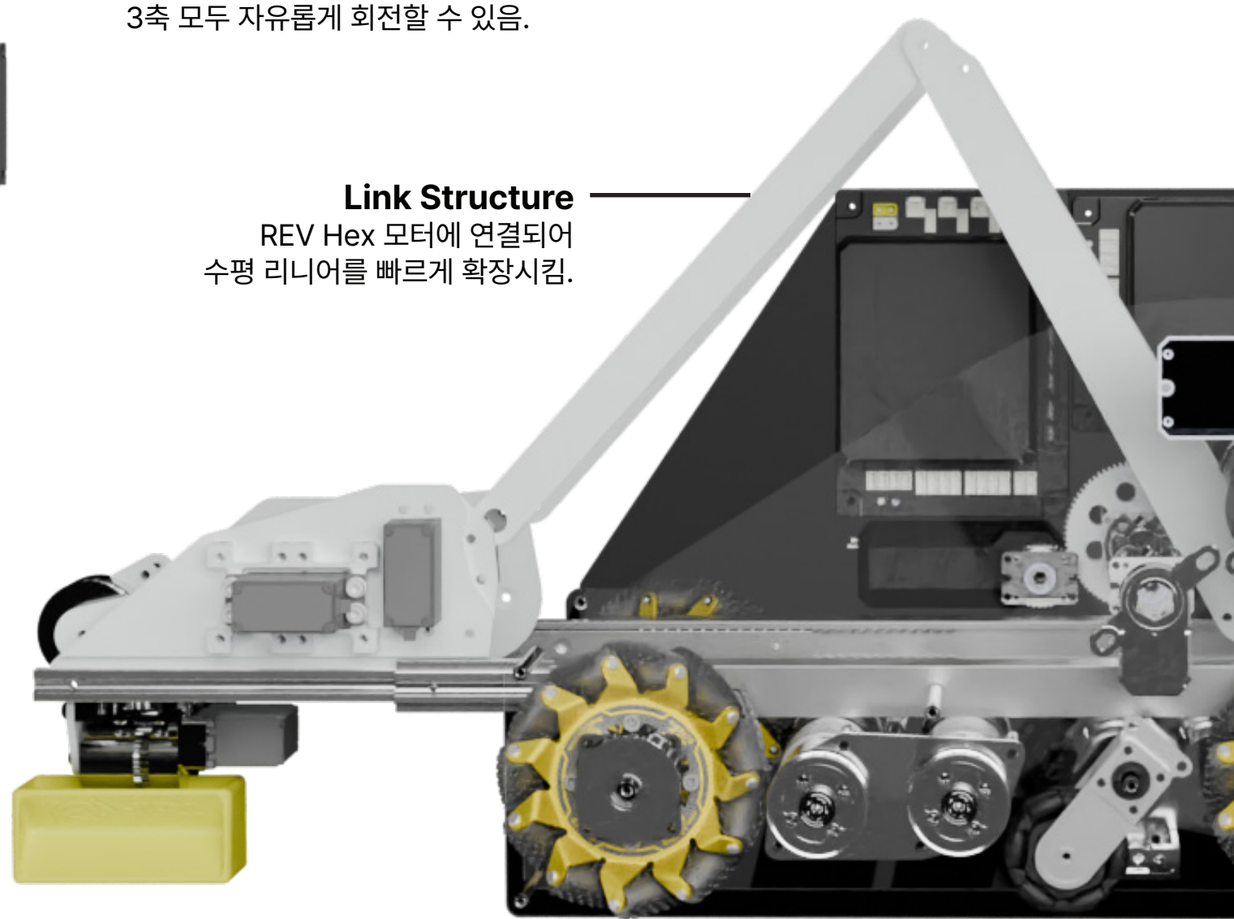
Localization
Odometry를 사용해 경기장 위에서의 정확한 위치 확인 가능, Auto 주행 시 높은 정확도.

SugEater
회전하는 롤러를 집게처럼 사용하여 Active Intake와 집게의 장점을 모두 살림.
3축 모두 자유롭게 회전할 수 있음.

SugEater
회전하는 롤러를 집게처럼 사용하여 Active Intake와 집게의 장점을 모두 살림.
3축 모두 자유롭게 회전할 수 있음.

Horizontal Linear Slide
MISUMI 4단 슬라이드.
SugEater가 연결되어 있어 수축하며 Deposit Claw로 샘플을 전달.

Link Structure
REV Hex 모터에 연결되어 수평 리니어를 빠르게 확장시킴.



Vertical Slide + SugEater

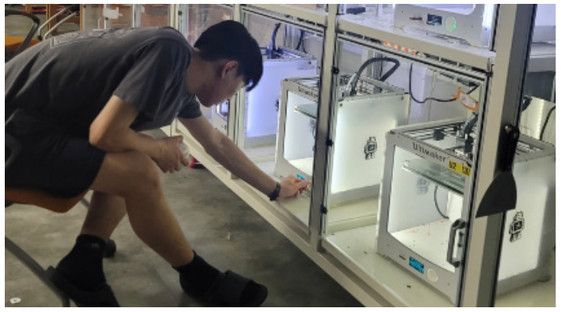
2024년 07월 02일 (화)

구분 | SUMMER_EN_BUILDING
책임자 | 23 홍정우

📝 **작성/편집 |** 23 홍정우
작업 참여 | 22 유태우, 22 이규진, 23 김준성, 23 김진용, 23 홍정우 24 이소민, 24 이재빈

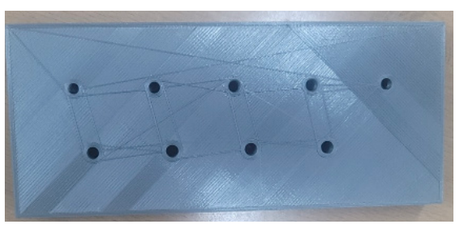
이전까지는 기존에 고빌다 혹은 다른 사이트에서 제공해준 철제 프레임을 기반으로 로봇을 제작하였었는데, 이번 대회에서는 로봇을 구체적으로 모델링하여 철제프레임을 쓰지않고 로봇 차체를 만들어볼 계획임. 그렇기에 시간이 여유로운 여름방학 중에 학교에 잔류하여 여러가지 3d 프린팅과 아크릴 등을 활용하여 다양한 테스트를 해보았음.

1. 3D 프린팅 기어 테스트



3D 프린팅한 기어를 사용할 수 있을지를 확인해보기 위해 기어들을 모델링 한 후 프린팅 해보았음. 그러나 3D 프린터의 퀄리티가 좋지 않아 잘 되지 않았지만, 기어가 기본적으로 맞물리고 돌아가는 기본적인 역할은 할 수 있을것이라 판단하였음. 추가적으로 Double helix gear를 만들어보기로 계획함.

2. 3D 프린팅 M4 고정 테스트



3D 프린팅을 했을때 M4가 고정되게 하려면 어느정도의 구경을 가지는 구멍인지를 확인하기 위해 다음과 같은 모델을 프린팅했음. 하나씩 M4를 끼워본 결과 4.05~4.15mm 정도의 구경정도에 고정이되는 것을 확인할 수 있었음.

2024년 07월 03일 (수)

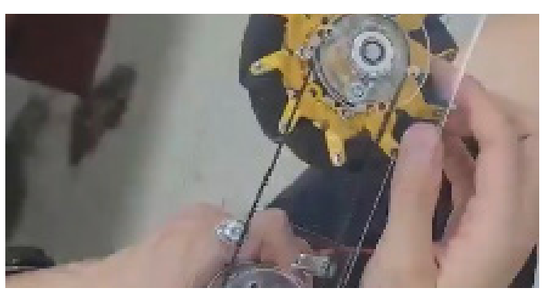
구분 | SUMMER_EN_BUILDING
책임자 | 23 홍정우

📝 **작성/편집 |** 23 홍정우
작업 참여 | 22 김수기, 22 최성빈, 23 홍정우

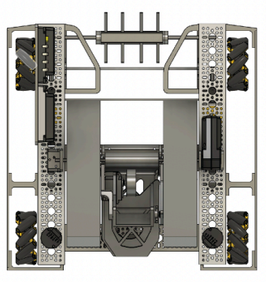
1. Prototype 차체 제작

새롭게 차체를 모델링해서 뽑아보기로 하였음. 기존에 사용하였던 고빌다 스타터 키트의 경우에는 DC모터와 기어를 사용하는데 이때 유격이 발생하는 등의 여러 문제점이 존재하였음.

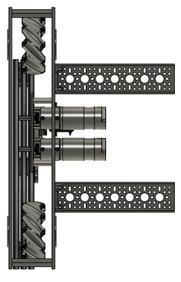
그렇기에 타이밍벨트 기반으로 DC 모터와의 연결 매커니즘을 바꿔서 설계해보았음. 이로 인해서 심한 유격을 줄일 수 있고, 공간 효율과 바퀴 축의 안정성을 향상 시킬 수 있었음.



📌 **CENTERSTAGE 차체와 비교!**



△ CENTERSTAGE



△ Prototype

Prototype 차체는 Belted Drivetrain으로 제작하였습니다.

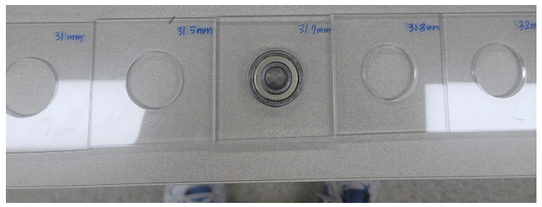
특징 / 장점 | 모터에 바퀴를 직접 연결하지 않고, 기어를 통해 연결하기에 차체 크기를 줄일 수 있으며 유격을 줄임. 타이밍 벨트로 연결하여 사용함.

2024년 07월 04일 (목)

구분 | SUMMER_EN_BUILDING
책임자 | 22 최성빈

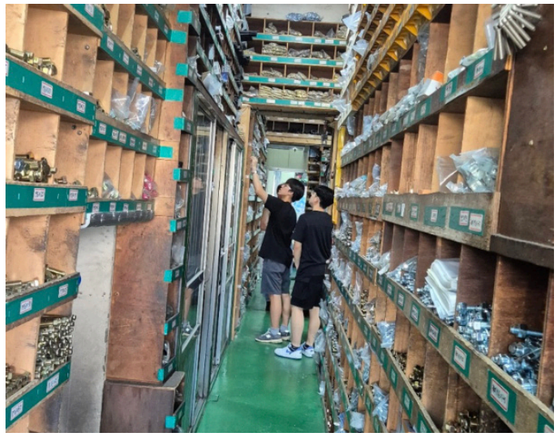
📝 **작성/편집 |** 22 최성빈
작업 참여 | 22 김수기, 22 최성빈, 23 홍정우

1. 아크릴 프린팅 베어링 고정 테스트



31,5mm 베어링이 들어가는 아크릴을 찾기 위해 다음과 같이 직경 31mm, 31.5mm, 31.7mm, 31.8mm, 32mm 로 직경에 변화를 주어 확인해본 결과 31.7mm이 작은 유격으로 가장 안정적이었고, 이보다 작은 직경에는 들어가지 않았음.

2. 공업단지 방문



부산사상공업단지에 방문하여서 필요한 부품들을 구매하고 프레임을 외주를 맡길 수 있는지 확인해보았음.

2024년 07월 05일 (금)

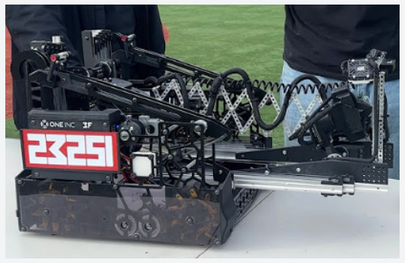
구분 | SUMMER_EN_BUILDING
책임자 | 22 김수기

📝 **작성/편집 |** 22 김수기
작업 참여 | 22 김수기, 22 최성빈, 23 홍정우

1. 시저 리프트 제작

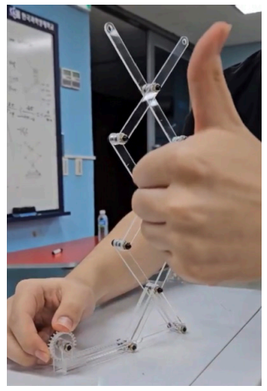
링크 구조의 기본이 된다고 할 수 있는 시저리프트를 아크릴로 제작해보았음. 작동은 되었지만 많은 힘이 필요하였음.

✅ **시저 리프트 (Caesar Lift)란?**



X자로 연결된 구조물이며, 모터 등을 통해 수직 방향으로 확장됩니다. 많은 해외 팀들이 시저 리프트를 사용하여 수평, 수직 리니어를 확장하는 데 보조합니다.

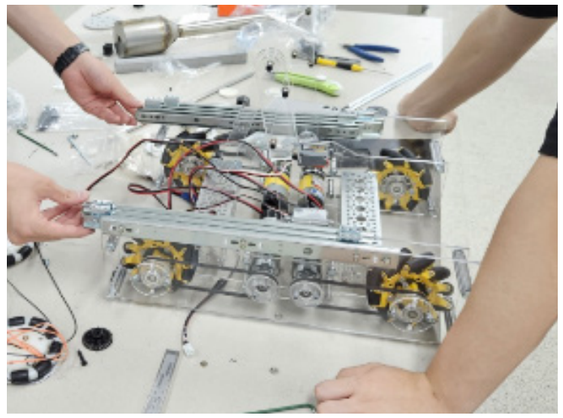
왼쪽은 23251 Triple Fault의 로봇입니다. 동력은 수평 리니어를 사용하며, 시저 리프트는 케이블 정리를 위해 보조만 한다는 사실을 알았습니다.



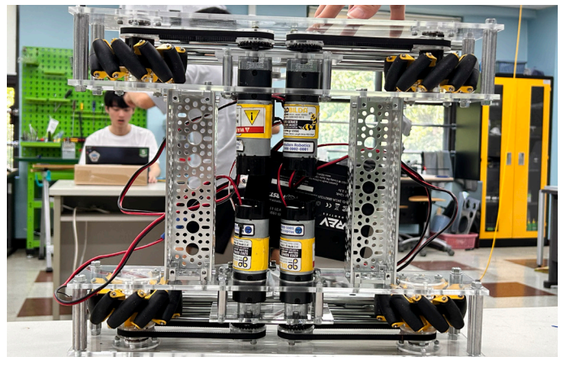
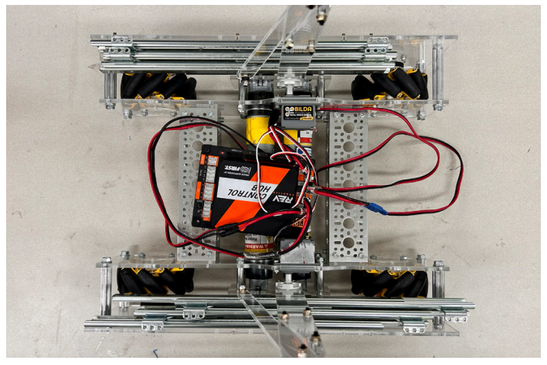
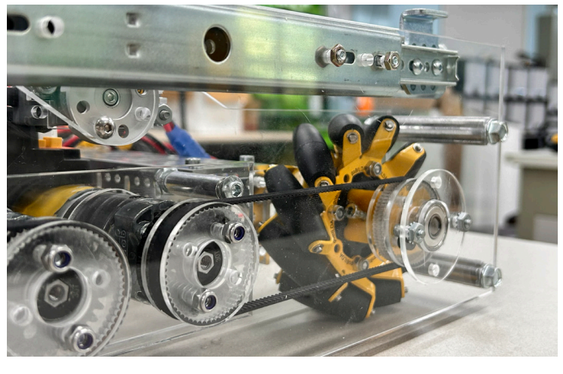
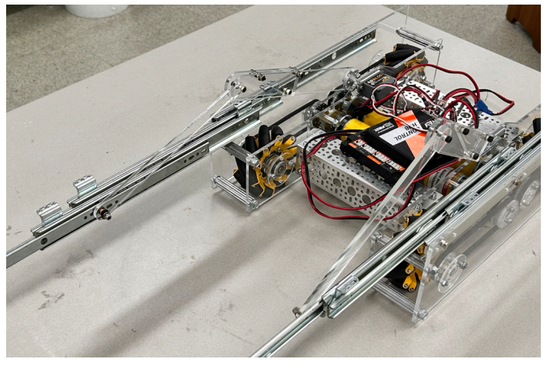
왼쪽과 오른쪽 사진은 각각 시저 리프트의 확장 전, 후의 사진. 케이블 정리를 할 때 어떤 식으로 고정하면 좋을지 실물을 보고 고민하기 위해 Rack and Pinion 구조를 사용해서 테스트해 보았다.

2. 프로토타입 차체 완성

타이밍 벨트를 기반으로 아래 차체부분을 제작 완료하고, 수평 리니어를 위에 부착하였음. 수평 리니어를 다음과 같이 서보와 아크릴을 통해서 만들게 됨.




타이밍 벨트를 기반으로 아래 차체부분을 제작 완료하고, 수평 리니어를 위에 부착하였음. 수평 리니어를 다음과 같이 서보와 아크릴을 통해서 만들게 됨. 완성된 사진은 아래와 같음.

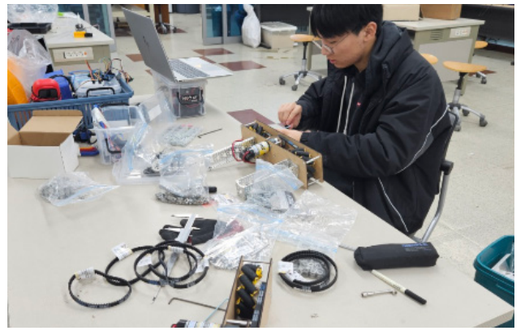
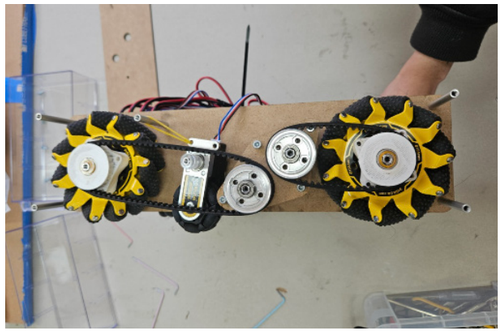





2024년 12월 11일 (수)

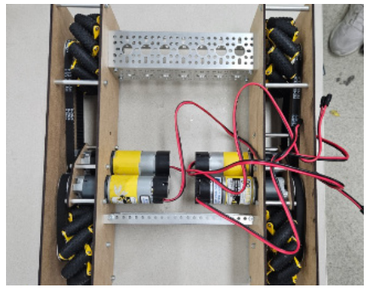
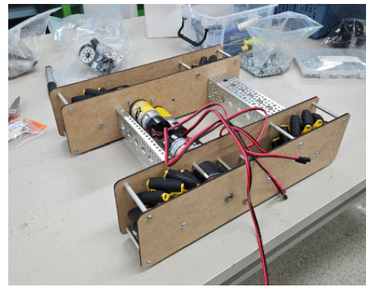
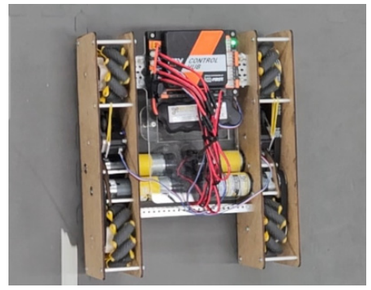
구분 | SEMESTER_EN_BUILDING
책임자 | 22 최성빈

작성/편집 | 22 최성빈
작업 참여 | 22 김수기, 22 최성빈, 23 홍정우

1. RoadRunner 테스트용 차체 제작

CAD 파일을 기반으로 로드러너 테스트용 차체를 제작함. 로봇의 폭을 최대한 줄이기 위해 메카넘 휠, 타이밍 벨트 풀리, 베어링과 DC 모터 등 구동부를 만드는 데 필요한 부품을 밀도 있게 배치하여 옆판 사이의 거리를 56mm 정도로 줄일 수 있었음. 옆판은 mdf로 제작하였으며 옆판은 56mm Standoff로 고정함. 설계에 맞추어 구입한 닫힘형 타이밍 벨트를 사용하여 DC 모터의 동력이 바퀴로 전달되도록 했으며 오도메트리는 일차적으로 차체를 조립해본 뒤 나중에 추가함. 오도메트리 세트에 사용되는 용수철 대신 고무줄을 사용하여 오도메트리가 바닥과 안정적으로 밀착될 수 있도록 함. 그리고 옆판끼리는 U 채널 두 개를 사용하여 연결하였으며 연결한 U 채널 위에 아크릴로 Control hub와 배터리가 들어갈 공간을 만들었음. 차체 완성 후에는 제대로 동작하는지 확인해보았고 잘 움직임.

2025년 01월 06일 (월)

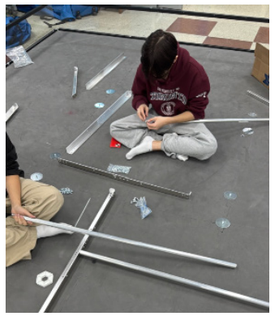
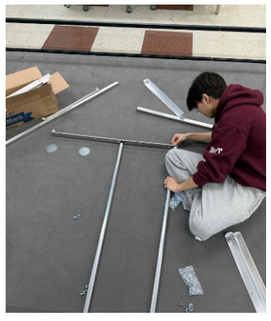
구분 | WINTER_EN_BUILDING
책임자 | 23 홍정우

📝 **작성/편집 |** 23 홍정우
작업 참여 | 22 유태우, 22 이규진, 23 김준성, 23 김진용, 23 홍정우 24 이소민, 24 이재빈

1. 경기장 제작

1월 6일 잔류를 시작하였으며, 먼저 도착한 일부 인원끼리 Into the DEEP 경기장의 조립을 시작함. Andy-Mark에서 Partial game set를 사전에 주문하여, 배송된 부품들을 조립함.

6*6 크기로 바닥 타일을 깔고 주변에 벽을 세움. 나머지 바구니와 가운데 잠수정을 조립하여 설치함.

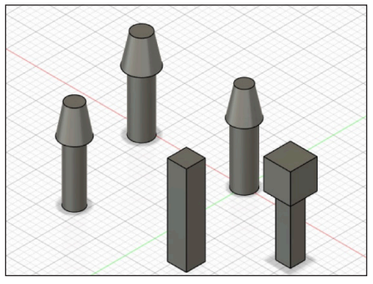
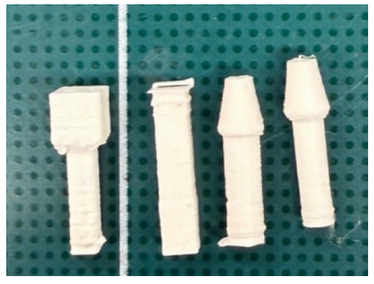
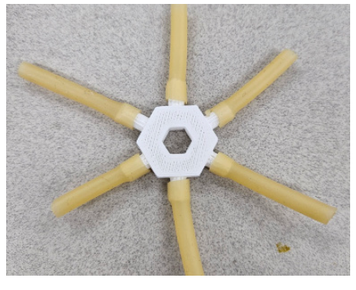

2024년 01월 07일 (화)

구분 | WINTER_EN_BUILDING
책임자 | 23 김준성

📝 **작성/편집 |** 22 최성빈, 23 홍정우, 24 이소민
작업 참여 | 전체

1. 롤링 Eater 제작

기존의 해외팀들 중에서 로터와 비슷한 형식으로 specimen을 가져가는 모델이 있었고, 로터를 막대에 surgical tube를 붙여서 만들기로 계획하였음. 막대에 surgical tube가 고정되려면 돌기가 필요하다고 판단하였고, 다양한 모양의 돌기를 제작해보았음. 기본적으로 4mm 지름을 가지는 막대에 돌기형태를 다르게 하여서 적합한 막대기 모양을 찾아보았음.

가장 오른쪽의 사진에서 순서대로 1. 정육면체 모양의 돌기 2. 돌기가 없는 형태 3.아래는 5mm, 위는 2mm 의 돌기 4. 아래는 4.5mm 위는 2mm 의 돌기임. 왼쪽의 사진과 같이 Surgical tube를 각 막대기에 끼워본 결과 돌기가 없이 잘 들어간다는 것을 알아내었음.

2. 전체 회의

연구회원 모두가 학교에 도착하여 로봇 제작 일정이 대해 김호숙 선생님과 회의를 진행함. 업무를 분담하고 매일 오전 10시에 작업을 시작하기로 결정함.

2025년 01월 08일 (수)

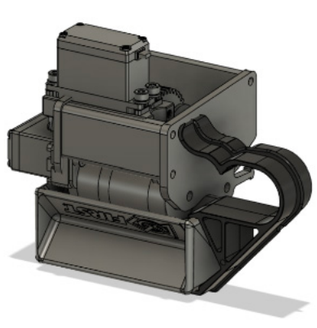
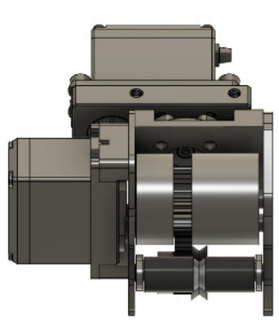
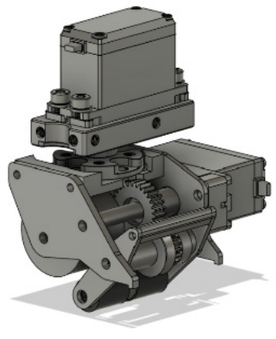
구분 | WINTER_EN_BUILDING
책임자 | 22 김수기

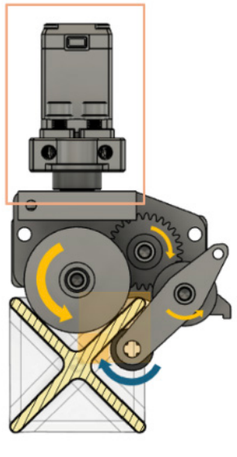
📝 **작성/편집 |** 22 이규진
작업 참여 | 22 김수기, 22 이규진, 22 강현빈, 22 유태우

1. SugEater (수기 Eater) 제작

1) Eater 설계

이터가 여러 각도와 관련없이 블록을 집을 수 있도록 롤러를 통해 블록을 빨아들이는 원리로 이터를 설계함. TALOS 팀원인 김수기가 이터를 설계하여 수기 이터라고 명명함.

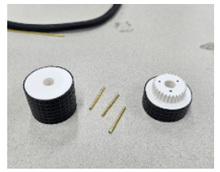


서보모터와 3개의 기어를 사용해 하나의 동력으로 블록을 잡는 두 방향의 힘을 만들도록 설계함. 3D프린터로 기어를 제작할 경우, 유격이 발생하여 기어가 서보의 스피드에 따라 헛도는 경우가 발생하기 때문에, 이를 보안하기 위해 장력을 사용함.

이터의 앞부분이 벌어지면 부착된 고무줄이 장력을 통해 다시 이를 잡으면서 블럭을 잡는 안정성을 추가함. 또한, 고무줄의 장력을 통해 일정한 유격을 유지함. 서보 모터에 연결된 축을 단단하게 만들기 위해 반대 부분에 standoff와 베어링을 연결함.

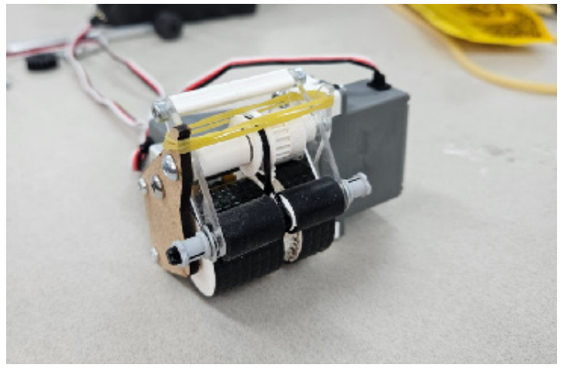
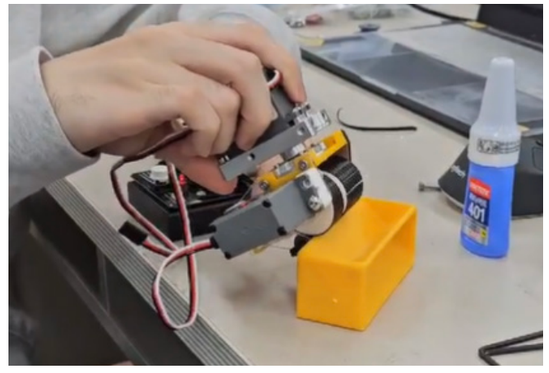
2) 이터 제작





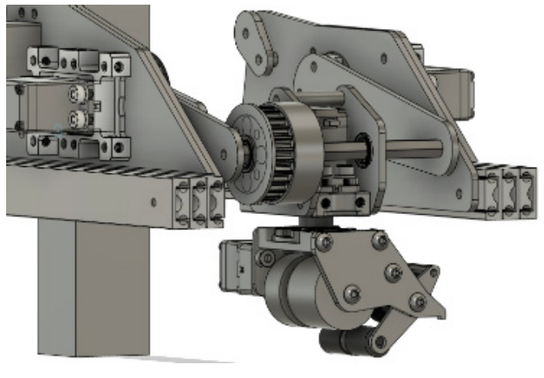
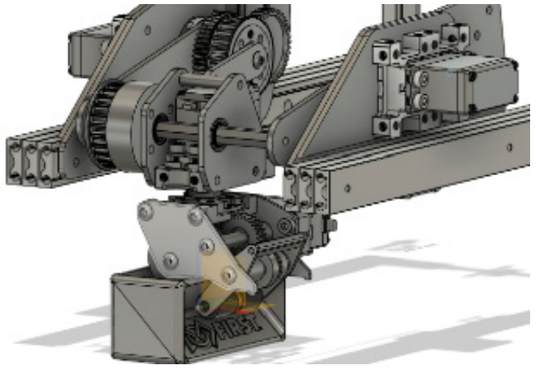
이터를 제작하는 과정에서 서보와 연결되는 3D프린팅 모델 부분이 서보와 맞물리는 부분이 잘 맞지 않아서 다시 모델링을 진행함. 기존 서보 톱니 부분의 1.0배, 1.05배, 1.1배, 1.2배로 맞물리는 부분을 제작하였고, 서보에 끼워본 결과, 1.05배가 가장 적합하다고 판단하고 교체하였음.

해당 이터를 작동시켜본 결과, 살짝 기울어져 있는 샘플이라도 쭉 밀면 샘플을 안정적으로 잡을 수 있음을 확인함. 다만, 위에서 내려찍으며 샘플을 먹을 수 있을 것으로 기대하였지만, 실제로 위에서 샘플을 먹기 위해선 비교적 위치의 제약을 받는다는 것을 확인함. 따라서 teleOP 때 빠른 동작을 위해서는 미는 방식으로 픽셀을 먹어야한다는 결론을 도출함. 따라서 잠수정 내부의 샘플을 먹기 위해서 이터의 높이를 조절할 수 있는 팔이 필요함을 인지했고, 이를 설계하기 시작함.

3) 이터 팔 설계

이터와 수평 리니어를 연결하는 설계도를 작성.

서보에서 두개의 freedom을 가지도록 설계를 했기 때문에 수평 리니어를 달아주면 먼 거리를 이동하지 않더라도 샘플을 집을 수 있다는 장점을 가지고 있음. 수평 리니어와 이터를 연결하는 과정에서 총 3개의 서보 모터를 사용하는데, 대칭적으로 있는 서보를 통해 이터의 손목을 움직이는 동력으로 사용하고, 하나는 팔을 움직이도록 설계함.

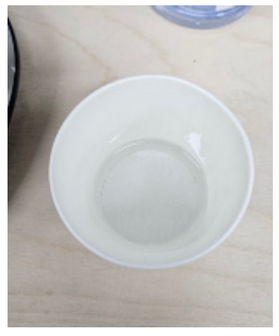
📝 **작성/편집 |** 22 최성빈
작업 참여 | 22 최성빈

2. 에폭시 레진을 이용한 mdf 경화 테스트

MDF를 에폭시 레진에 적셔서 안쪽까지 스며들게 만든 다음 경화시키면 MDF의 단점인 낮은 내구성 문제를 해결할 수 있는지 확인하기 위해 진행함 (테스트 목적).

원래는 알루미늄 차체 키트를 사용하여 로봇의 구조를 만들었으나 이번에는 자체 제작 차체를 사용하게 되었으므로 레이저 커터를 사용하여 옆판을 만들게 됨. 그런데 아크릴은 충격에 쉽게 깨지고, 단단한 집성목이나 원목은 5T 미만으로 얇게 가공이 어려우며, 알루미늄 판 가공은 CNC가 필요한데 장비가 없으므로 결국 mdf를 사용하게 됨. 차체 옆판으로 사용하는 것에 있어서 그렇게 큰 문제는 없지만 지속적인 부하로 인해 휘거나 충격에 의해 일부가 뜯겨나갈 수 있으므로 두께를 바꾸지 않으면서 경도만 올릴 방법을 궁리하게 됨.



일단 10cm*10cm 3T mdf를 5장 준비하여 그중 3장에 주제와 경화제를 3:1 비율로 혼합한 에폭시 레진을 박스 조각을 이용하여 고르게 펴바름. 작업 전후 무게 차를 측정하여 10cm*10cm 한 면을 코팅하는데 사용된 레진의 양을 계산해보니 대략 1g 미만이 도출됨. 걱정되는 것은 단순히 겉에 레진을 바른다고 안쪽까지 레진이 스며들지는 잘 모르겠다는 것. 레진이 경화되는데 보통 12~24시간이 소요되므로 다음 날에 결과를 확인함.

경화 ⭕

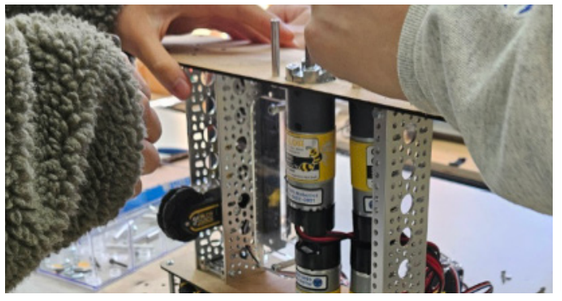
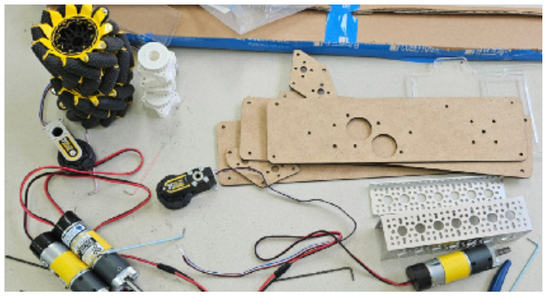

경화 ❌


결과적으로 경화한 MDF와 경화하지 않은 MDF 모두 같은 강도의 충격에 부서진다는 것을 확인할 수 있었음. 즉, 추후 MDF를 사용할 때 경화하지 않고 사용하기로 결정함.

📝 **작성/편집 |** 23 김준성
작업 참여 | 22 이규진, 22 최성빈, 23 김진용, 23 김준성, 23 홍정우

3. 테스트용 차체 분해

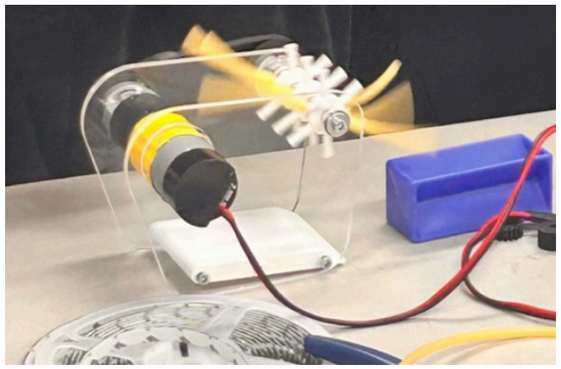
여름 잔류기간 동안 제작했던 차체를 분해하여 정리함. 부품도 종류별로 정리하였으며, 디자인한 모델을 기반으로 새로 제작할 수 있도록 준비함.

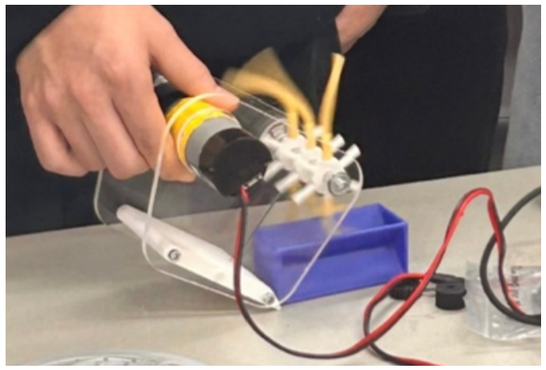



📝 **작성/편집 |** 24 이소민
작업 참여 | 24 이소민, 24 함주원

4. 롤링 Eater 제작

Surgical tube에 적합한 막대기를 찾고 그후 롤링이터를 제작함. 롤링이터는 DC모터와 타이밍 벨트를 이용하여 만들었음.





롤링 이터는 양옆의 판을 5T 아크릴을 사용하여 제작했고 이외의 부품들은 3D 프린터로 출력함. Specimen이 롤링이터에 의해 끌려가는 것은 확인했지만 바닥에 부딪혀 specimen이 올라가지 못함. 그렇기에 바닥 부분에 specimen이 경사로를 올라갈 수 있는 추가적인 장치가 필요하다고 판단됨.

📌 **SugEater과 비교!**

장점 | SugEater는 집게를 사용한 방식이기에 정확한 위치에 집어야 합니다. 반면 Rolling Eater와 같은 active intake 방식은 sample 근처에만 있으면 비교적 부정확한 위치에서도 집을 수 있다는 장점이 있습니다. 또한 서보 모터 대신 DC 모터를 사용하여 속도를 훨씬 높일 수 있습니다.

단점 | 그러나 집을 수 있는 방향이 세로로 한정되어 있으며, 현재로는 바닥에 부딪힌다는 문제가 있습니다.

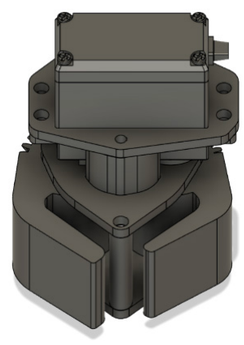
2025년 01월 09일 (목)

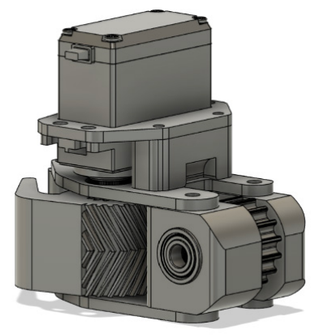
구분 | WINTER_EN_BUILDING
책임자 | 23 홍정우

📝 **작성/편집 |** 22 최성빈, 22 장호원
작업 참여 | 22 이규진, 22 최성빈, 22 장호원, 22 강현빈

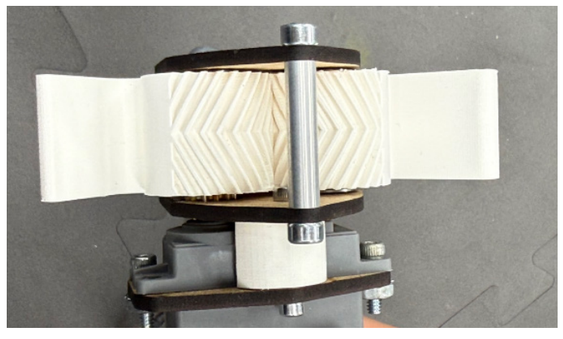
1. Deposit 집게 제작

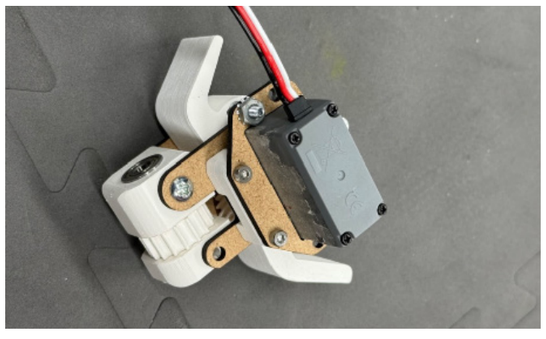
롤러를 통해 잡은 샘플을 다시 잡을 deposit 집게를 제작함. 톱니 구조를 활용해 하나의 서보모터로 집게를 구현함. 샘플이 들어가 있는 구조를 활용해 정확히 잡을 수 있는 구조를 설계함. 이후 팔에 연결할 기어 또한 철사를 통해 고정함.





기어는 helical 기어를 사용해 반대로 된 헬릭스 각도에 의해 축방향 힘이 상쇄되어 더 안정적이고 빠르게 서보모터 작동이 가능함.





위 사진과 같이 조립을 완료함.

📝 **작성/편집 |** 23 김준성
작업 참여 | 23 김준성, 23 김진용, 23 류승완, 23 홍정우

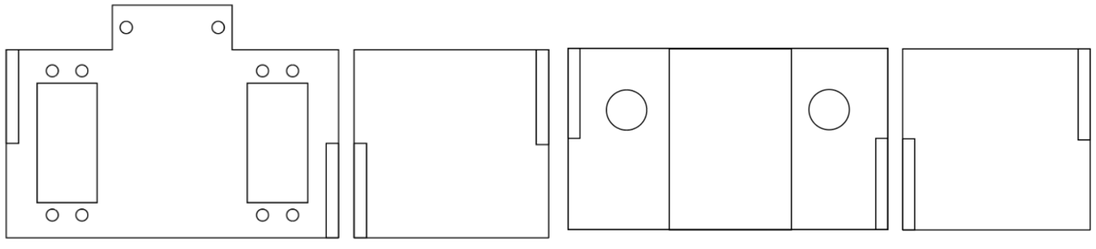
2. 사이드 Eater 제작

롤러를 사용하는 해외 팀들의 영상을 참조하여 이터를 제작함. 다른 이터에 비해 훨씬 빠른 속도로 intake할 수 있을 것이라고 기대하였으며, 운전자가 보이지 않는 곳에서 조종하기에 가장 좋은 방법이라고 생각하였음.

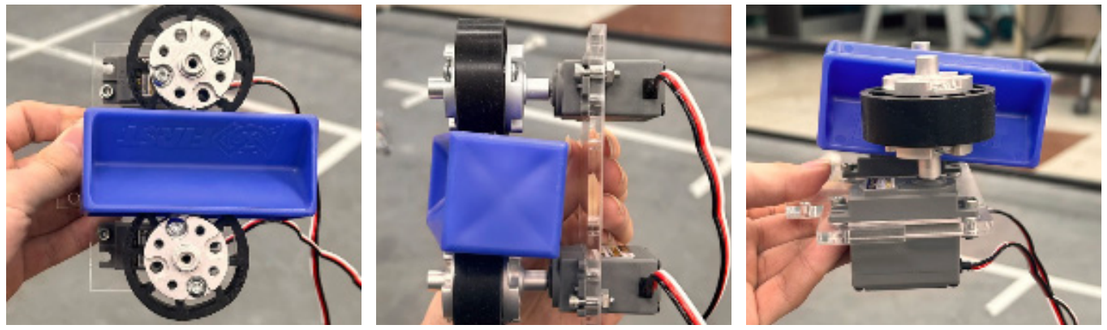
✅ **참고한 해외 팀 영상**

왼쪽부터 순서대로 14380 Blue BotBuilders, 14872 Orbit Knights, 18228 Red BotBuilders의 active intake. 이중 Orbit Knights의 이터를 참고하였음.

첫 시도에서 서보는 GoBilda의 Torque 서보를 Continuous Rotation으로 설정하여 사용하였으나, 속도가 느려 Superspeed 서보로 변경하였음. Adobe Illustrator로 아크릴 도면을 그려 출력하였으며, 서보를 끼워 테스트해보았음.



아크릴 도면은 위와 같음.



위의 사진과 같이 샘플을 집은 후 모터의 회전을 멈춤으로써 집은 상태를 유지할 수 있었음. 컬러 센서는 코드가 완성되지 않은 관계로 테스트해보지 못하였음.

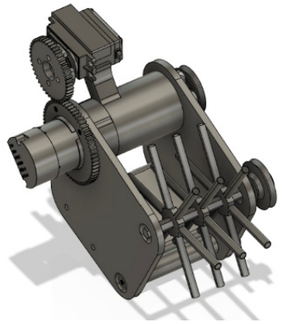
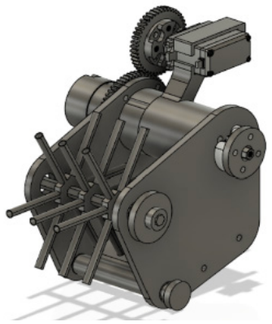
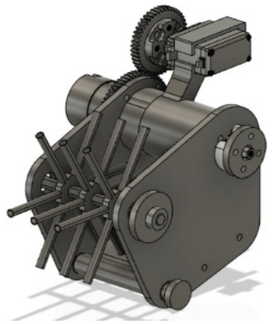
샘플을 먹는 과정은 아래과 같음. 제작 후 몇 번에 걸쳐 테스트해본 결과 가로와 세로 등 모든 방향에서 샘플을 집을 수 있음을 확인함. 샘플 여러개가 서로 이웃하여 모여있는 경우에도 잘 작동함을 확인함.

📝 **작성/편집 |** 24 이소민
작업 참여 | 24 이소민

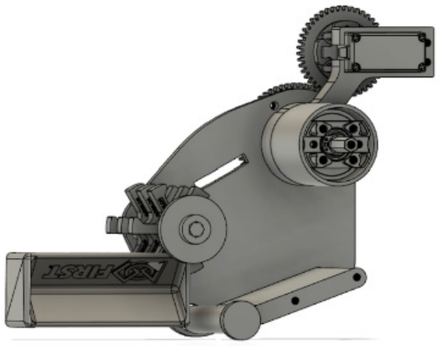
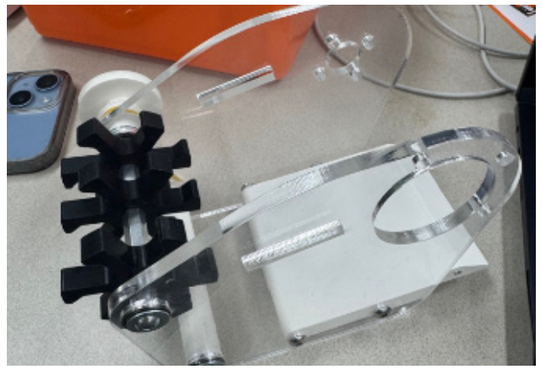
3. 롤링 Eater 개선

앞서 만들었던 롤링 intaker가 specimen을 먹지 못해서 앞에 롤러를 붙여 specimen을 먹는 방법을 고안하여 제작을 시도함. 또한 기존에 DC 모터에서 동력을 전달하기 위해 사용한 타이밍 벨트 방식이 과도하게 공간을 차지해, 고무줄을 이용한 풀리형태로 설계를 수정함.

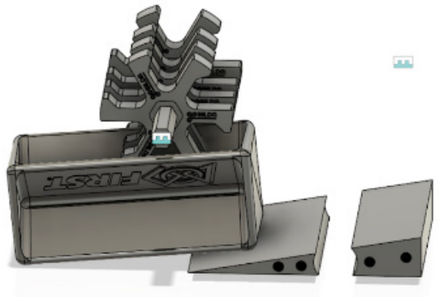
Intaker의 회전을 위해서 DC 모터를 축으로 회전하는 서보모터를 이용한 회전부를 설계함.

제작하여 실험해보니, 고무 튜브를 사용한 intaker는 롤러가 있어도, 롤러의 높이를 넘어 specimen을 잡기 위한 힘을 제공하지 못함. 때문에 GoBilda 사의 intake 전용 OD Wheel을 사용하여 재설계함.

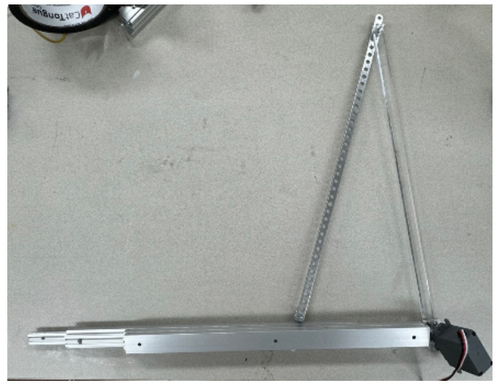
하지만 본 모델도 intaker wheel이 롤러의 높이를 높기위한 충분한 힘을 제공해주지 못함. 이를 해결하기 위해 다른 사례들을 찾아본 후, 전면부에 롤러를 없애고 매우 낮은 경사형태로 바꾸어 specimen이 쉽게 intake 되도록 수정함. Intaker에 들어간 specimen이 로봇으로 운반되기 위해서 경사 중간에 롤러를 배치함.



작성/편집 | 23 김진용
작업 참여 | 23 김진용, 23 류승완, 24 함주원

4. 3단 리니어 가이드 조립

앞서 제작한 롤러 이터가 장착된 헤드가 앞뒤로 자유롭게 움직이며 샘플을 다음 단계로 전달할 수 있도록 하기 위해 MISUMI사의 수평 리니어를 3단으로 조립하여 제작함. 리니어의 전개는 아래 그림과 같이 팔 막대를 달고 이를 모터로 회전시키는 방식. 이 때 리니어를 펼치고 접을 막대를 리니어와 연결하기 위해 연결부를 따로 아크릴 판을 이용하여 제작함.



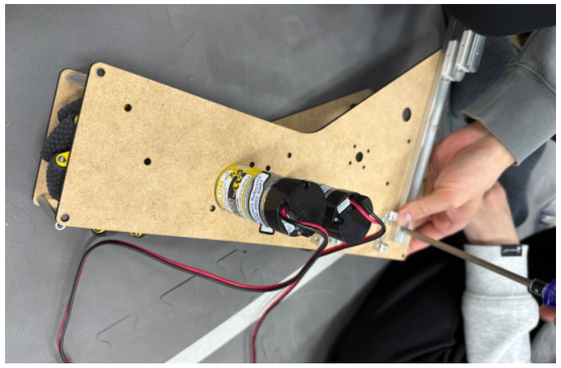

2025년 01월 10일 (금)

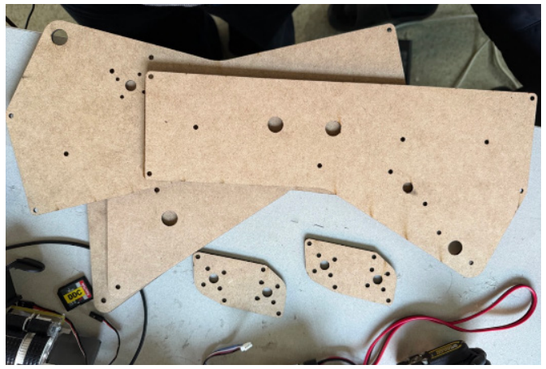
구분 | WINTER_EN_BUILDING
책임자 | 22 김수기

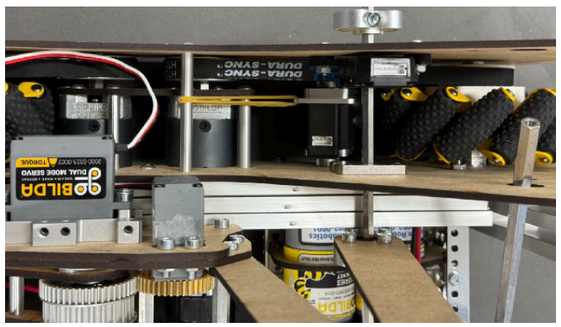
📝 **작성/편집 |** 23 홍정우
작업 참여 | 전체

1. 차체 재구성

RoadRunner 테스트를 위해 만든 차체를 분해하고, 단순히 주행의 역할만 하는 차체에서 더 발전시켜 수평, 수직 리니어를 부착할 수 있는 공간을 추가하여 새로 차체를 출력 및 조립함. 실제로 보다 좁은 폭을 위해 차체에서 남는 공간이 최대한 효율적으로 사용함. 구동하는 DC 모터를 세로로 두어 바퀴에 바로 연결했을때, 수평 리니어가 들어갈 공간이 없었음. 이를 인지하고, 모터를 가운데에 두어 타이밍 벨트로 동력을 전달하는 차체를 구축함.

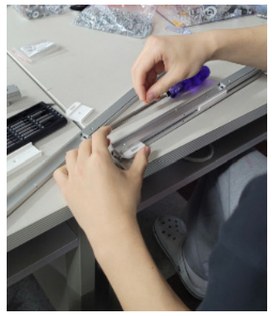


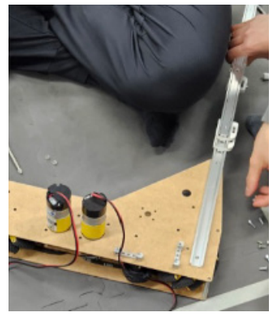




완성된 결과를 보면 타이밍 벨트를 통해 모터와 바퀴가 직접 연결되지 않아도 굉장히 폭이 좁게 설계되었기 때문에, 접시 모양 나사를 사용해 리니어를 차체에 밀착시켜 제작함. 정확한 Auto를 위한 오도메트리도 standoff에 고무줄을 걸어 설치함. 이를 통해 굉장히 작은 사이즈의 차체를 제작할 수 있었음.

2. 수직 리니어 조립 및 설치

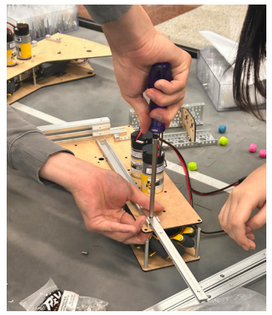


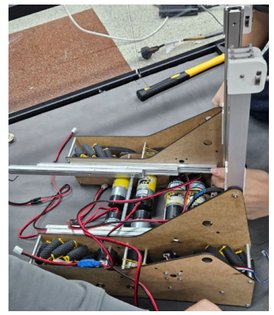


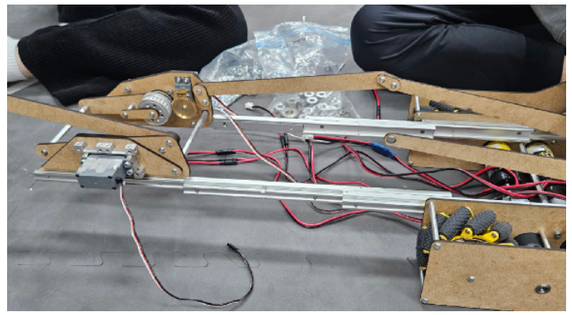
3d 프린터로 출력한 리니어 가이드와 풀리 마운트를 차례대로 고정하여 3단 리니어를 조립함. 10mm M4 접시머리 나사를 이용하여 인접한 리니어와 리니어의 슬라이드 부분을 결착하였고 풀리는 따로 고정하지 않았음.

본래의 차체 위에 수직 리니어를 부착함.

3. 수평 리니어 제작 및 설치







차체가 샘플을 집는 용도로 수평 리니어도 제작함. 이는 수직 리니어와 같은 방법으로 조립했으며 차체의 안쪽에 수평하게 부착함. 수평 리니어는 4단을 사용하였으며, 팔을 부착할 계획을 가지고 있었음.

수평 리니어를 부착한 이후에는 수평 리니어를 접었다가 펼 수 있도록 팔을 설치함. 리니어의 끝과 끝을 하나의 관절을 사이에 둔 2개의 긴 막대를 활용하여 연결했고, 차체쪽 끝의 막대에는 헥스모터를 연결하여 모터가 제공하는 토크로 수평 리니어를 접었다가 펼 수 있도록 만듦. 결과물은 오른쪽 사진의 모습.

2025년 01월 12일 (일)

구분 | WINTER_EN_BUILDING
책임자 | 23 김준성

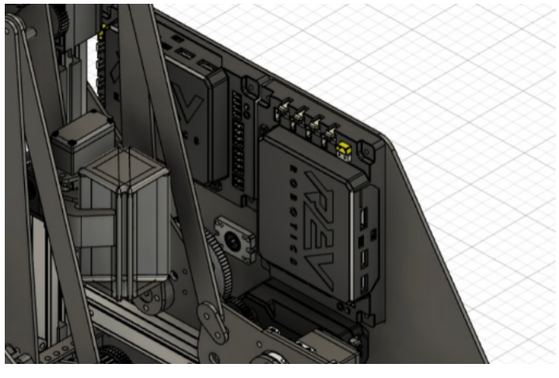
📝 **작성/편집 |** 23 김진용, 23 류승완
작업 참여 | 전체

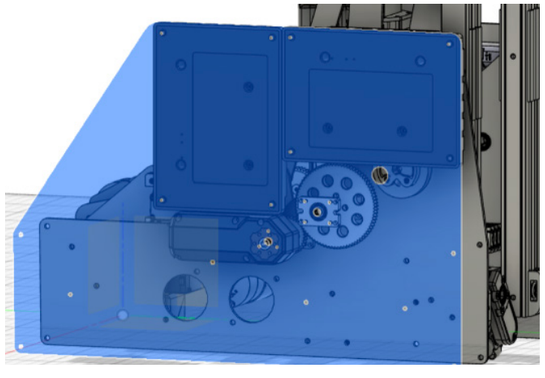
1. 설계 및 차체 골격 수정

리니어를 구동시키는 기어를 수정한 새로운 모델을 cad로 설계하여 기존에 제작한 차체를 분해하고 수정하여 재조립하는 과정을 거침. 재조립하는 과정은 이번대회가 처음인 빌더들이 차제의 구조와 조립과정을 배우게 하는데에도 의의가 있었음. 재조립하는 과정에서 몇가지 주의사항을 깨달을 수 있었음.

1) 리니어 전개 방식 수정

앞서 제작한 차체에서, 굽은 팔을 모터로 회전시켜 수평 리니어를 전개하는 방식을 사용하였는데, 이 때 모터의 토크가 부족하여 전개가 제대로 이루어지지 않음. 이에 따라 기어비를 조정하여 토크를 크게 할 수 있도록 설계를 수정함. 또, 모터 조작을 위한 허브를 장착할 공간이 없어 허브를 장착할 수 있는 공간을 추가하여 전체 차체 골격을 수정함. 수정된 차체 골격을 레이저 커터를 통해 MDF로 프린트함.

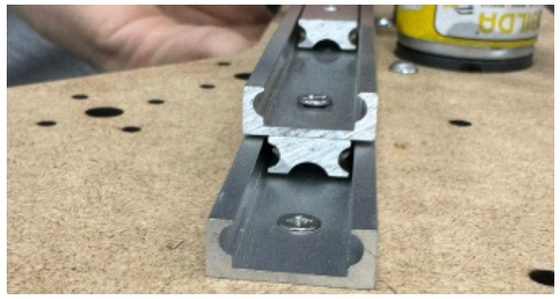


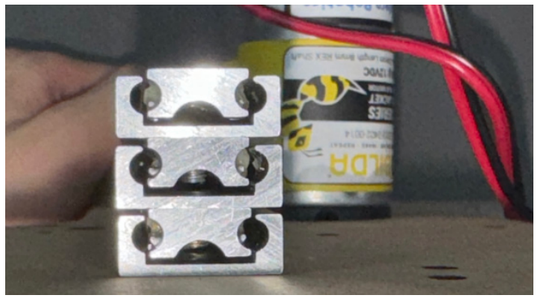


2) 수평 리니어 나사 걸림 문제

Horizontal Linear 결합시 나사가 서로 걸리는 문제점이 발견됨. 리니어에 평소에는 수평 방향으로만 힘이 가해지기에 큰 문제는 없었으나, 수직 방향으로 힘을 가하면 나사에 걸려 확장에 방해가 된다는 것을 확인하였음.

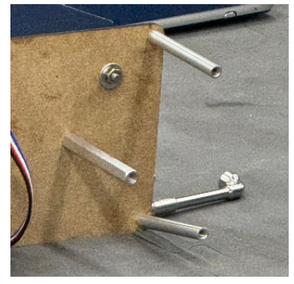
이는 첫번째 나사를 타이트하게 조여야 리니어가 걸리지 않고 부드럽게 넘어갔음.





3) MDF 강도 문제

Stand off를 먼저 조립을 하고 조립을 하는데 stand off가 조립되어있는 나사구멍과 가장자리의 간격이 작아 MDF가 파손되는 일이 빈번하게 일어남. 이를 위해 최종 prototype에서는 MDF보다는 강화된 재료를 사용하거나 가장자리의 나사 구멍을 조금 여유있게 설계할 필요성을 느낌

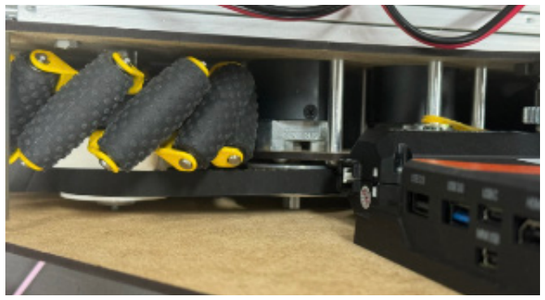


4) HEX 모터 나사 구멍 위치

설계상에서 Hex Motor를 설치하였을 때 맞은 편 stand off의 나사구멍이 막히는 문제점을 발견하게 됨. 재조립과정에서는 stand off를 한쪽만 고정하여 문제를 해결함. 이는 추후 Hex Motor의 위치를 조금 이동시켜 해결을 할 예정임.

5) 베어링 구멍.

CAD파일상에서는 motor의 베어링 구멍이 두개를 설계되어 있었으나, 레이저 커팅 과정에서 문제가 발생하여 구멍이 커팅 되지 않았음.

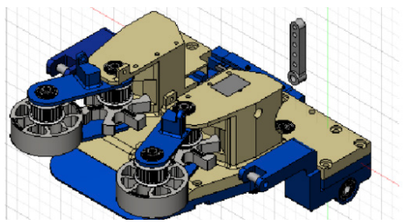


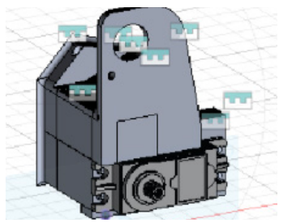
📝 **작성/편집 |** 24 함주원
작업 참여 | 24 함주원

2. 개선된 롤링 Eater 설계

1월 9일에 롤링 이터 제작을 진행하며 조금 더 다양한 각도의 샘플을 원활히 먹기 위해 새로운 형태의 이터를 제작함. Bluebot과 미팅을 통해 전달 받은 CAD 파일을 기반으로 설계하는 방안으로 결정함.

Bluebot이 사용한 부품을 기반으로 설계를 진행하려고 했으나, DC모터를 Superspeed servo 2개로 대체하고 REV color sensor의 위치를 수정하는 등, 일부 역설계를 시작함. 특히 3D 프린터로 출력하기 위해 돌기와 홈을 이용해 조립하는 형식으로 수정 방향을 설정함.





다음과 같이 서보를 위해 새로 마운트를 설계하였고, 타이밍 벨트를 이용해 동력을 전달할 수 있도록 수정함. 샘플의 이탈을 방지하기 위해 이터의 뒷면에 Torque servo를 닮. 컬러센서가 색을 인식하여 색이 다르다면 배출하고 원하는 색상이면 갖고 있도록 함.공간 상 컬러센서의 위치를 뒤쪽으로 옮김. 판과 판 사이를 고정하기 위해 축을 프린트하여 고정하는 방식을 채택함. 나머지 부품 역시 이미 보유한 부품을 활용하여 제작할 수 있도록 설계를 수정함.

2025년 01월 13일 (월)

구분 | SUMMER_EN_BUILDING
책임자 | 22 김수기

📝 **작성/편집 |** 22 최성빈
작업 참여 | 22 최성빈, 22 강현빈

1. 번호판 제작

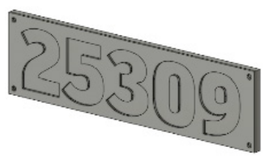
ROBOT SIGN 관련 규정을 참고하여 일단 가장 간단한 직사각형 판에 가로로 팀 번호를 붙여 놓은 형태로 제작에 들어감. 팀 번호판의 숫자 부분을 양각(Emboss)이나 음각(Engrave)으로 표현할지, 아니면 종이나 코팅지를 붙이거나 마스킹 처리를 해서 숫자 부분의 색만 다르게 도색하는 방법 등 다양한 방식을 고민해 봄.

양각으로 표현할 경우 숫자 부분의 아크릴 조각들을 글루건이나 아크릴 접착제를 이용하여 고정시켜야 하므로 추후 내구성 문제가 발생할 수 있다고 생각하였고 이것을 해결하고자 볼트를 사용하게 되면 볼트 구멍을 뚫어야 하므로 미관 상 좋지 않다고 판단하였음.

음각으로 표현할 경우에는 마스킹 작업 없이 숫자 부분을 다른 색으로 표시할 수 있고 도색 작업도 숫자와 바탕 부분으로 나누어 간단히 처리할 수 있음.

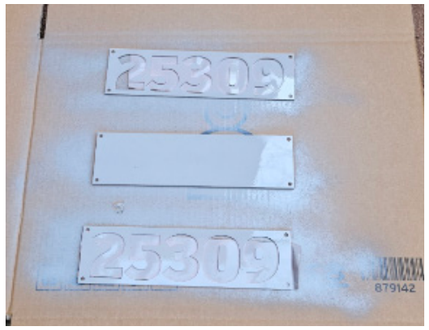
시트지에 팀 로고와 번호를 프린트한 후 아크릴에 붙이는 방법은 결과물을 깔끔하겠지만 작업이 번거롭고 시트지 인쇄를 따로 외주를 맡겨야 함. 마스킹 처리를 하는 방법은 몇 번의 시행착오를 겪을 것이라고 생각하지만 결과물의 두께가 한 장으로 얇고 작업이 순조롭게 된다면 좋은 결과를 기대할 수 있다고 생각함. 이중에서 음각과 마스킹을 사용하는 방법을 시도해보기로 결정함.

1) 음각



작년에 아크릴에 래커를 이용하여 도색하였을 때 외관상 문제가 없었던 것으로 기억함. 그래서 따로 서페이서를 뿌리지 않고 아크릴의 보호 필름을 제거한 후 바로 래커로 도색함. 그 결과는 아래 사진과 같이 도막이 충분히 두껍지 않아 도포량에 따라 아크릴 뒤의 색이 비춰보였음. 그래서 그냥 서페이서를 뿌리고 도색하기로 결정함. 작은 크기의 조각들은 래커를 뿌릴 때의 힘 때문에 밀려 도색이 어려웠으므로 몇 번의 시행착오 끝에, 도색할 조각들을 테이프로 고정하여 도색하는 방법을 이용함.

서페이서 ⭕

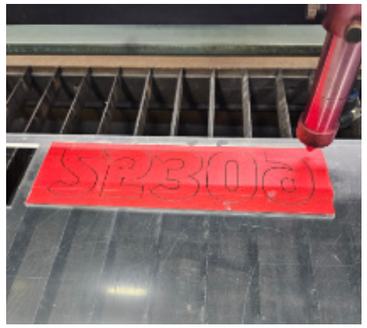


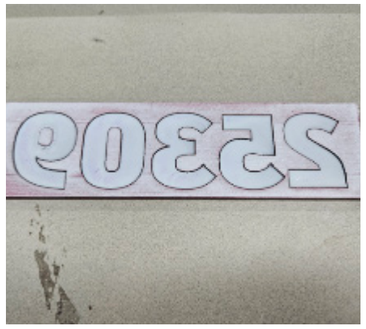
서페이서 ❌

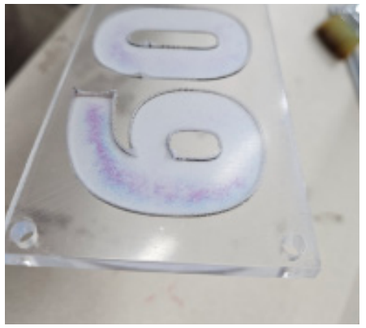
이전에 결정한 내용대로 서페이서를 먼저 아크릴 번호판 표면에 도포하고 충분히 건조한 다음, 락카를 이용하여 빨간색, 파란색으로 도색함. 서페이서 없이 도색했을 때보다 도막이 두꺼워져 뒤가 비치는 문제는 해결되었으나 오히려 표면에 도료가 부분부분 뭉쳐 거칠게 변함. 하지만 색이 섞이지 않는 것이 더 중요하므로 서페이서를 뿌리는 편이 낫다고 생각함. 서페이서를 뿌리고 1200방 사포로 표면을 살짝 정리해주는 것이 좋을 것 같음.

2) 마스킹

마스킹 테이프를 일일이 재단하여 도색하는 것은 번거롭고 시간도 오래 걸리므로 레이저 커터를 활용하는 방법을 시도해봄. RDWorks에서 숫자 부분의 외곽선을 Cut 레이어로 설정하고 Speed를 50mm/s, Power를 20%로 설정하여 작업함. 화재가 난다거나 하는 등의 안전 사고는 발생하지 않았고 결과물은 깔끔하게 잘 나왔음.







그런데 흰색 래커로 도색하고 마스킹 테이프를 제거해보니 생각만큼 작업물의 질이 좋지 않았음. 빨간색과 파란색 래커 도색을 옆에서 먼저 하고 그다음에 흰색으로 도색을 했더니 착색되어 얼룩이 생겼고, 레이저 커팅으로 파인 부분이 백화되는 등 약간의 문제가 생김. 그래서 레이저로 인해 파이는 깊이를 최소화하기 위해 레이저 커터의 Speed와 Power를 조금씩 바꾸면서 테스트해 봄. Speed를 4배, Power를 1/2배인 200mm/s, 10%로 설정해서 레이저 커터를 가동해 보았지만 파이는 깊이가 같았음. 이와 같은 결과로부터 레이저 커터 소프트웨어 상에 속력 제한이 걸려있다는 사실을 알 수 있었음.

📌 도색 작업 과정!

1. 레이저 커터로 5T 아크릴 판재를 번호판 크기의 직사각형으로 절단 및 레이저 커터 위치 고정.

2. 절단한 직사각형의 아크릴 조각 한면의 보호 필름을 제거하고 서페이서 도포.

3. 서페이서가 충분히 건조된 후에 고운 사포로 표면을 살짝 정리.

4. 마스킹 테이프를 아크릴 조각 크기에 맞추어 꼼꼼히 붙여줌.

5. 1번에서 레이저 커터로 절단한 부분에 맞추어 아크릴 조각을 배치하고 레이저 커터로 번호 부분을 절단(Speed 100mm/s, Power 10%로 설정).

6. 마스킹 테이프의 이음새를 글루건으로 고정한 후, 번호 부분(면적이 작은 조각들)을 먼저 제거하고 락카로 도색(제거한 마스킹 테이프 조각들은 따로 보관).

7. 번호 부분에 수성 바니쉬를 발라주고 열풍기로 건조(5분이면 마름).

8. 번호 부분 마스킹 조각들을 다시 붙이고 배경 부분의 마스킹을 제거, 그리고 배경 부분을 락카로 도색.

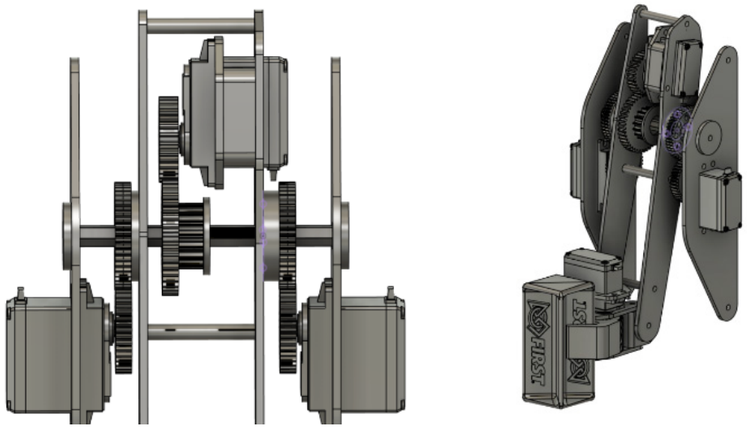
9. 배경 부분도 수성 바니쉬를 바르고 열풍기로 건조.

10. 작업 마무리.

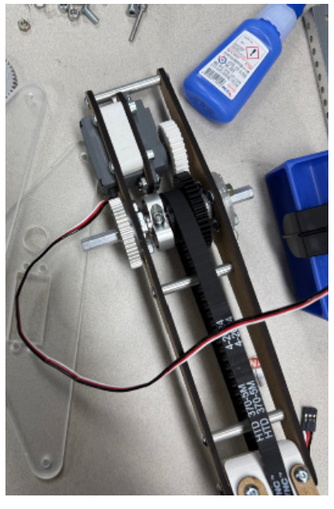
📝 **작성/편집 |** 24 이소민
작업 참여 | 전체

2. 수평 리니어 팔 제작

설계를 토대로 수평 리니어에 부착될 팔 제작을 시작함. 서모 마운트와 기본적인 판은 MDF로 제작하였고, 기어의 위치를 고정하기 위한 스페이서를 아크릴로 제작하였음.



안쪽 서보모터에 부착되는 검정색 기어는 3D 프린터로 출력하고 나머지 스퍼기어들은 메탈기어를 사용함.



본체에 부착되는 양쪽의 서보모터에 달린 기어가 40 tooth 기어인데 2개가 필요한 40 tooth 기어가 1개밖에 남지않아, 30 tooth 기어로 설계를 수정함. 서보모터 마운트에서 서보모터 구멍과 볼트 구멍의 위치를 +y 방향으로 약 4mm 이동하여 수정함.

또한 회전축이 본체에 고정이 되어야 하는데, 기존 방법은 와셔 모양의 MDF를 순간접착제로 붙여 고정하는 방법이었음. 하지만 접착제를 사용하는 설계의 단점이 너무 많아, 본체에 축이 연결되는 부분의 구멍을 축에 맞는 hex 모양으로 수정하여 고정함.

모두 제작한 다음, 팔 전체의 회전을 담당하는 기어 부분이 서보모터의 위치 문제로 180도 이상 회전하지 못하는 문제점이 발생함. 이를 해결하기 위해 축의 길이를 늘리고, 스페이서를 추가하여 팔 회전부가 360도 회전할 수 있게 설계를 수정함.

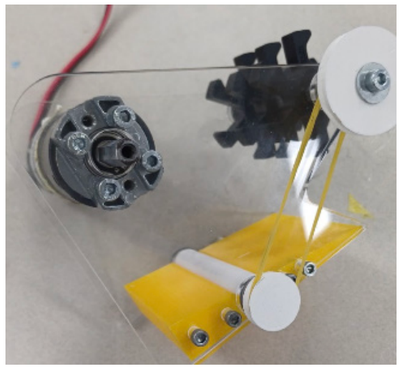
2025년 01월 14일 (화)

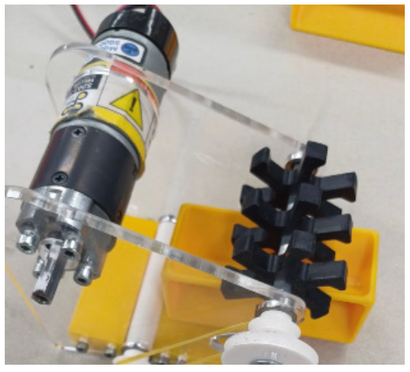
구분 | SUMMER_EN_BUILDING
책임자 | 24 이소민

📝 **작성/편집 |** 24 이소민
작업 참여 | 24 이소민

1. 롤링 Eater 완성

기존에 제작을 시도했던 롤링 이터를 총 6회정도의 시도 끝에 완성함.



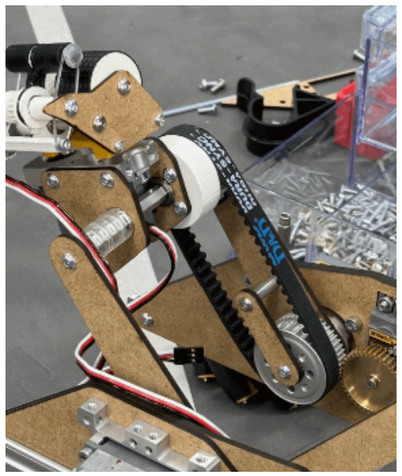


앞서 말했던, 롤러 높이에 의한 문제와, 고무 튜브의 탄성력 부족등을 보완한 모델을 설계하여 specimen을 완벽하게 intake 할 수 있는 모델을 설계했고 출력하여 완성함. 양옆의 판은 5T 아크릴로 제작하였고, 나머지 부품들은 3D 프린터로 출력함.

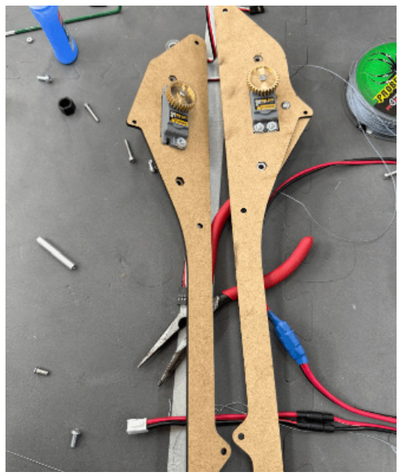
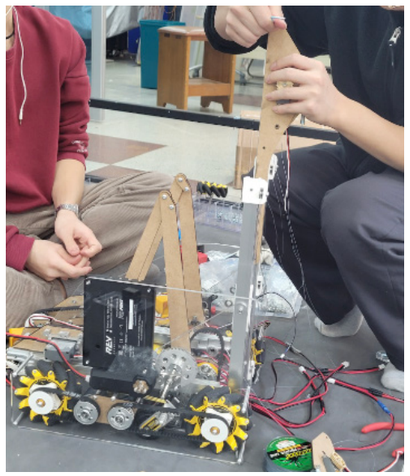
📝 **작성/편집 |** 22 장호원
작업 참여 | 전체

2. 팔 설계 수정

전날 팔에 있는 서보 모터와 기어가 걸려서 팔이 360도 돌아가지 않는 문제를 서보와 기어의 위치를 조정하여 설계한 것을 제작함. 설계를 수정한 결과 팔이 360도 원활하게 잘 돌아가는 것을 확인함.




팔 뿐만 아니라 설계에 맞게 수직 리니어쪽 마운트도 수정하여 다시 수직리니어와 팔을 부착하여 연결함. 이외에도 실제 작동시켜보는 과정에서 생긴 마이너한 문제들을 해결함.

이렇게 최종적으로 다시 조립한 후 구동시켜본 결과 잘 작동하는 것으로 확인되었으며, 마지막으로 선 정리를 진행함.

이로써 로봇을 완성함. 완성한 로봇은 이터, 수평 리니어, 수직 리니어, 팔이 유기적으로 연결되어 견본을 옮기는 가장 기본적인 움직임을 비롯해 Into the Deep의 미션을 수행하기에 적합하며 차체 또한 메카넘 휠과 오도메트리로 자유롭게 이동이 가능함. 직접 경기장에서 경기 시뮬레이션을 해본 결과 미션을 문제 없이 효율적으로 잘 수행하는 것으로 확인됨.

ENGINEERING NOTE

PROGRAMMING

KRC 2025

This section is about last season’s programming using OnBot Java.

This chapter is mostly written in Korean.

APOC 2025
Engineering Note

I. 사전 준비

저번 대회보다 더욱 효율적이고 강력한 기능을 가진 로봇 컨트롤러를 제작하기 위하여 개발 환경, 라이브러리, 개발 방식 등 다양한 방면에서 새로운 기술을 학습하고 연구하는 과정을 거침. 공부한 내용을 바탕으로 기존보다 더욱 발전된 코드를 작성하여 로봇의 구동, 특히 자율 주행 측면에서 큰 이점을 얻을 수 있었음.

1. Android Studio 기반의 개발 환경 구축

📌 **Rev Client에서 빌드했을 때의 단점!**

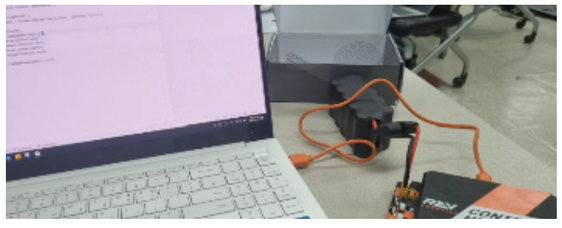
] 코드를 전부 옮기는 과정이 **번거로우며**, 그 과정에서 **실수가 발생**할 수 있음.

] **여러 사람이 함께** Control Hub에 연결하여 코드를 빌드할 수 없음.

] **상수값 수정 및 버그 수정 등**은 코드를 다시 옮기는 작업을 최소화 하기 위해 Rev Hardware Client에서 직접 코드를 수정할 수 있음. 하지만 이 경우 Android Studio로 **다시 코드를 옮겨야 하는 불편함**이 있으며, 이를 잊어먹는다면 수정한 값이 날아가 추후에 **불필요한 노동**을 수반하며, 심지어는 더 큰 버그로 발전하는 경우가 있음.

이러한 단점을 극복하기 위하여 이번 시즌부터는 Android Studio에서 직접 Robot Controller를 빌드하는 방법을 찾아내었음.

Android Studio에서 Control Hub를 디바이스로 인식하면 Control Hub에 Robot Controller를 빌드할 수 있음. Android Studio에 Control Hub를 연결하기 위해서는 연결선을 이용해서 직접 연결하는 방법과 Control Hub WiFi 연결과 Android Studio의 ADB Wi-Fi 플러그인을 통한 무선 연결이 가능함. 연결선을 활용한 방법은 인터넷을 사용하면서 동시에 빌드가 가능하다는 점에서 공부를 병행하며 테스트를 할 때 활용하기 적절한 방법임. 반대로 무선 연결은 원거리에서도 쉽게 Robot Controller를 빌드할 수 있다는 점에서 로봇을 제작한 후 그에 맞는 코드를 작성해 빌드할 때 유용하게 쓰일 수 있음.



Rev Hardware Client는 Driver Hub 및 Control Hub의 앱 및 펌웨어 버전 관리, 로그 확인, 그리고 WiFi 설정 관리를 위해서만 사용하고, 그 외의 소스코드 빌드 작업은 Android Studio로 바로 빌드하는 이러한 시스템은 기존의 단점을 보완하고, Android Studio의 검증된 최적화를 사용할 수 있으며, Github를 통해 공유되는 코드 스페이스를 직접적으로 활용하여 동시에 소스코드 빌드가 가능하다는 점에서 기존에 방식에 비해 큰 이점이 있으리라 기대됨.

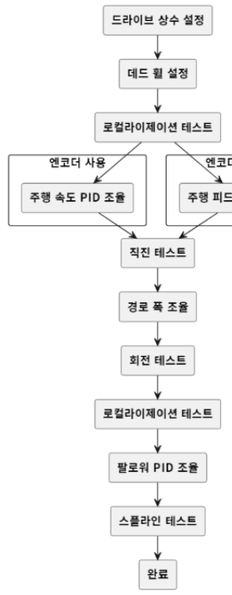
2. Road Runner

Road Runner는 자율 주행시에 오도메트리와 IMU 센서를 활용한 자율 주행 기능을 편리하게 제공하기 위해 ACME Robotics 등이 제작한 모션 플래닝 라이브러리임. 대부분의 해외팀은 정확한 Localization과 자율 주행을 위해 Road Runner를 사용하며, 이를 통해 AutoOp에서 높은 점수를 얻을 수 있음. 저번 대회에서는 IMU와 로봇 바퀴 엔코더에 기반한 자율 주행 코드를 작성했기에 정확도가 상당히 떨어져 추가적인 점수 없이 Randomization만을 목표로 하는 간단한 자율 주행만 구현할 수 있었음. 세계 대회 출전을 기점으로 오도메트리를 구매하였으나, 당시에는 시간적인 한계로 인하여 Road Runner에 대한 공부 없이 자체적인 Localization 코드를 작성하였으나, PID를 사용하지 않아 단계별 정확한 정렬을 위해 많은 시간이 소요되었고, 정확한 튜닝이 불가능하여 정밀도의 한계가 명확하였음.

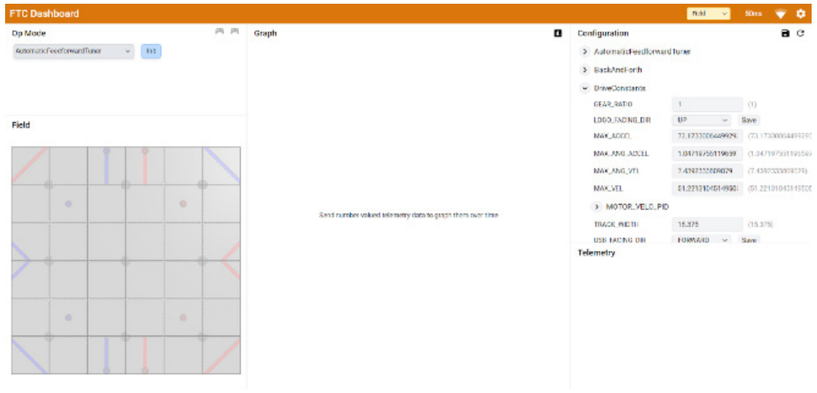
저번 대회의 한계를 극복하고, 정확한 Localization을 통해 AutoOp의 효율을 극대화하기 위하여 Road Runner를 공부함. 또한, 추가로 한국의 FTC 팀들이 Road Runner를 쉽게 접하고 공부할 수 있도록 한국어 기반 Road Runner 가이드 웹사이트를 제작함 (Outreach 섹션 참고).

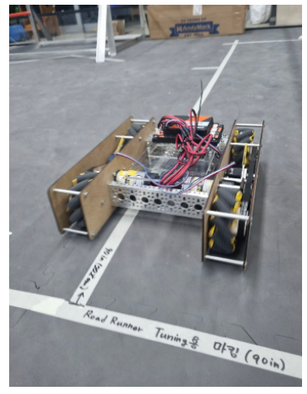
1) 튜닝 (Tuning)

Road Runner의 튜닝을 시도하였음. ACME Robotics의 Road Runner Quickstart 레포지토리를 가져와 개발 환경을 구축하고, 절차에 맞게 튜닝 과정을 수행함. 튜닝 절차는 오른쪽 그림과 같다. 튜닝 테스트를 통해 필요한 공간의 크기, Localization의 성능을 극대화하기 위한 오도메트리의 위치 설정 등 로봇 제작 시 고려해야할 점, 그래프와 로봇의 움직임 해석한 후 이에 기반한 PID k값 수정 방법 등을 터득하였으며, 이는 추후 로봇을 제작하여 자율 주행 시스템을 구축하는데에 큰 도움이 되리라 생각됨.



튜닝 결과 상당히 정확하고 빠르게 원하는 위치로 이동하고 정렬할 수 있음을 확인하였음. 하지만 첫 튜닝인만큼 각 과정마다 약간의 오류가 있었고, 주행이 여러번 반복되면 될수록 오차가 커지는 것이 발견되었음. 그럼에도 불구하고 자율 주행이 정확하게 되는 것을 통해 각 과정에서 튜닝을 확실하게 하고 넘어간다면 실제 로봇에서의 자율 주행 성능이 크게 향상될 수 있으리라 결론 지을 수 있음.

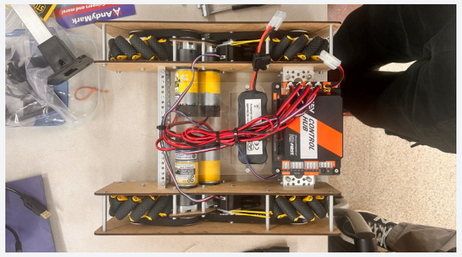




2) 경로 설정 (Trajectory)

경로를 설정하고 Road Runner가 작동하는 모습을 직접적으로 확인하였음. 정확한 위치와 방향으로의 움직임, 직선적인 움직임 뿐만 아니라 곡선적인 움직임도 가능한 시스템, 움직임 중간에 마커를 추가하여 다른 장치를 움직이는 등 추가 장치를 조작할 수 있는 기능 등 다양한 기능을 테스트 해봄으로써 Road Runner의 가능성을 확인해보고, 이를 기반으로 자율 주행 코드를 어떻게 구성할 수 있을지를 생각해봄. 움직임이 매우 정확하고 빠르며, 마커 등 강력한 기능을 적절히 사용한다면 높은 성능의 자율 주행 시스템을 구현할 수 있으리라 생각됨. 특히 기존의 코드 구조는 비동기적인 처리를 위해 오버헤드가 많이 발생하는 시스템인지라 성능 저하가 많이 발생하였으나, Road Runner의 시스템을 적극 활용한다면 AutoOp는 물론 TeleOp 시기에도 높은 효율을 낼 수 있을 것으로 기대됨.

📌 **RoadRunner 테스트에 사용한 차체**



RoadRunner 테스트는 프로토타입으로 제작한 차체를 사용했습니다. 자세한 내용은 학기 중 작성한 Building EN에 기록되어 있습니다.

실제로 사용할 Drivetrain과 같은 구조를 차용하였으며, 보정 등 다양한 테스트를 진행했습니다.

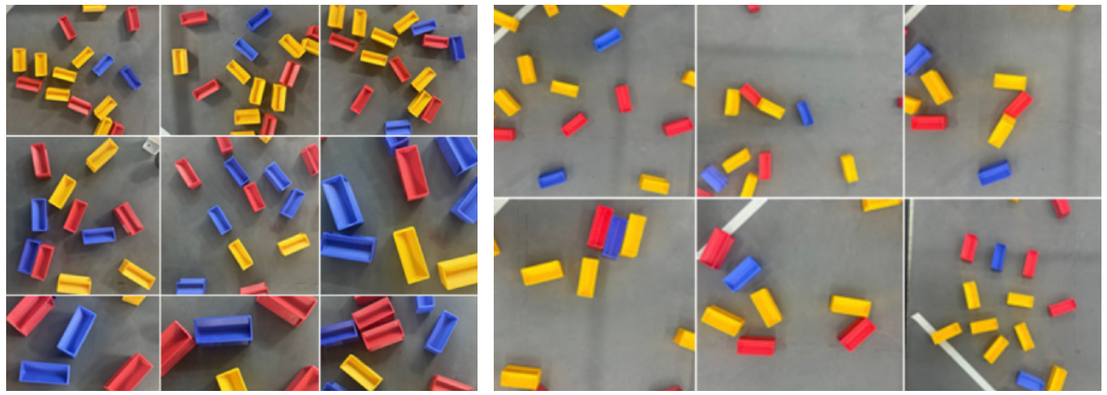
3. Vision

샘플을 정확하게 인식하고, 위치와 색깔 정보를 활용하여 자율 주행과 원활한 드라이버의 로봇 조종을 위하여 웹캠을 활용한 비전을 공부하고 연구하였음.

1) 머신 러닝

실시간 웹캠 피드에서 블록을 정확하게 인식하고 구분하기 위해 YOLO (You Only Look Once) 모델과 Roboflow를 활용함. Python을 이용해 이러한 시스템을 구현함으로써 실시간 객체 인식의 가능성을 탐구함.

YOLO 모델을 효과적으로 사용하기 위해서는 충분한 학습 데이터가 필요하기에, 세가지 색상의 블록을 다양한 각도와 조명 조건에서 촬영된 블록 이미지들을 수집함.



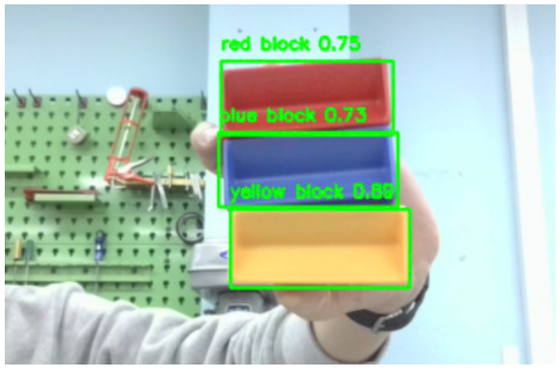
✅ **YOLO 모델이란?**

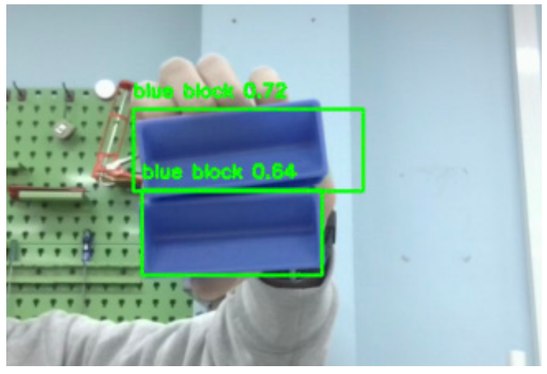
YOLO는 객체 인식 분야에서 널리 사용되는 실시간 객체 탐지 모델입니다. YOLO의 주요 강점은 이미지를 한 번에 처리하여 여러 객체를 동시에 인식할 수 있다는 점입니다.

기존의 객체 인식 모델 | 이미지를 여러 번 분석

YOLO | 단일 신경망을 통해 전체 이미지를 한 번에 처리

수집된 데이터는 Roboflow를 통해 전처리되었으며, 이미지에 바운딩 박스를 그리고 클래스 라벨을 지정하는 주석 작업을 거쳤음. 이러한 과정을 통해 모델이 블록 객체를 다양한 상황에서 정확하게 인식할 수 있도록 데이터셋을 준비하였음. YOLOv8 모델을 선택하여 Roboflow에서 제공하는 도구를 사용해 학습을 진행하였으며, 하이퍼파라미터 튜닝을 통해 모델의 성능을 최적화함. 학습된 모델은 높은 정확도로 블록 객체를 인식할 수 있었음.





Roboflow는 컴퓨터 비전 프로젝트를 위한 데이터 관리 및 전처리 도구로, 본 프로젝트에서 핵심적인 역할을 하였음. Roboflow를 통해 이미지 데이터셋을 효율적으로 관리하고, 주석 작업을 간편하게 수행할 수 있었음. 데이터 증강 기능을 활용하여 데이터셋의 다양성을 높였으며, 전처리 과정을 자동화함으로써 모델 학습에 최적화된 데이터를 준비할 수 있었음. 또한, Roboflow의 API를 통해 학습된 모델을 손쉽게 배포하고, Python과 연동하여 실시간 객체 인식 시스템을 구축하는 데 기여하였음.

2) 두 번째 모델 제작 및 테스트

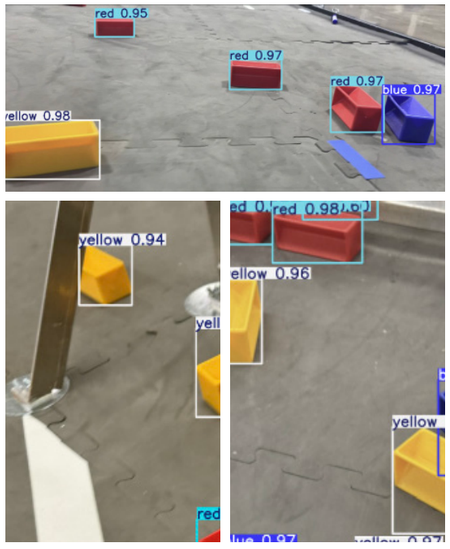
다음으로는 YOLO v8보다 더 높은 성능의 최신 버전인 YOLO v11 모델과 Roboflow에 올라와있는 오픈 데이터셋을 활용하여 심층 학습을 진행하였음. Roboflow의 데이터 증강 기법을 활용하여 기존의 데이터셋을 회전, 확대, 축소 등의 변형을 거쳤으며, 색조, 채도, 밝기 등에 다양성을 추가하여 다양한 환경에서의 데이터를 학습할 수 있도록 데이터셋을 구축함. 이를 통해 총 203개의 이미지를 487개의 이미지로 증폭시켜 학습에 활용하였음.

학습은 4번에 걸쳐서 진행하였음. 각 학습에서 사용한 데이터셋 원본은 동일하지만, 데이터 증강 기법을 활용해 서로 다른 세 가지의 데이터셋을 제작함. 학습 결과는 아래와 같음.

Version	mAP	Precision	Recall
Ver 1	99.2%	95.4%	99.1%
Ver 2	99.2%	95.1%	100.0%
Ver 3	99.2%	94.3%	99.8%
Ver 4	97.5%	95.1%	91.3%

학습 결과 2번째 버전에서 모든 지표의 값이 가장 높게 확인되었음. 학습이 거듭되며 2번째 버전까지는 학습 결과 발전이 있었지만, 그 이후로는 오버피팅에 따라 정확도가 감소한 것으로 추정됨. 아래 그림은 2번째 버전 YOLO 모델의 객체 감지 모습이며, 이와 같이 매우 높은 정확도로 샘플의 종류와 위치를 특정할 수 있는 것이 확인됨.

YOLO 모델을 학습시키면 PyTorch 가중치 파일을 얻을 수 있고, 이를 FTC SDK에서 이용하기 위해서는 TFlite 파일로 변환시켜야 함. 파일을 변환시키는 건 손쉽게 가능했지만, Output Layer 구조가 달라서 FTC SDK에서 이를 활용할 수 없었으며, 찾아본 결과 FTC SDK에서 지원하는 TensorFlow의 버전이 낮아서 호환이 안 되는 것이 문제였음. 결론적으로 인공지능 기술이 발전함에 따라 TensorFlow가 빠르게 변화함에도 불구하고 FTC에서 이를 즉각적으로 반영하는 것이 어려워 이번 시즌을 기점으로 TensorFlow를 활용한 객체 감지 기능의 공식적인 지원이 중단되었고, 이에 따라 우리 팀은 Limelight 또는 HuskeyLens 등 다른 하드웨어를 활용한 머신러닝 또는 Open CV의 이미지 처리를 통한 객체 감지를 수행해야 했음.



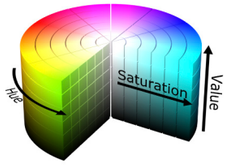
3) 결론

결론적으로 머신 러닝 테스트 통해 YOLO와 Roboflow를 활용한 실시간 객체 감지 모델을 생성하는 것이 가능하며, 충분히 쓸 수 있을 정도의 높은 정확도를 가짐을 확인하였음. YOLO의 빠른 객체 인식 능력과 Roboflow의 효율적인 데이터 관리 및 전처리 기능을 결합함으로써 높은 정확도와 실시간 처리를 동시에 달성할 수 있었음. 최종적으로, 실시간 객체 인식 기술의 실용성과 가능성을 확인하였으며, 이를 기반으로 더욱 발전된 응용 시스템을 개발할 수 있는 기반을 마련하였음. 물론 이번 시즌부터 FTC가 TensorFlow 기반 인공지능 모델 지원을 공식적으로 종료함에 따라 현재는 당장 사용할 수 없는 기술이지만, 이후에 Limelight 또는 HuskeyLens 등 인공지능 카메라를 구매한다면 이 때의 경험을 바탕으로 높은 수준의 객체 인식 시스템을 구성하여 자율 주행에 활용할 수 있을 것으로 기대됨.

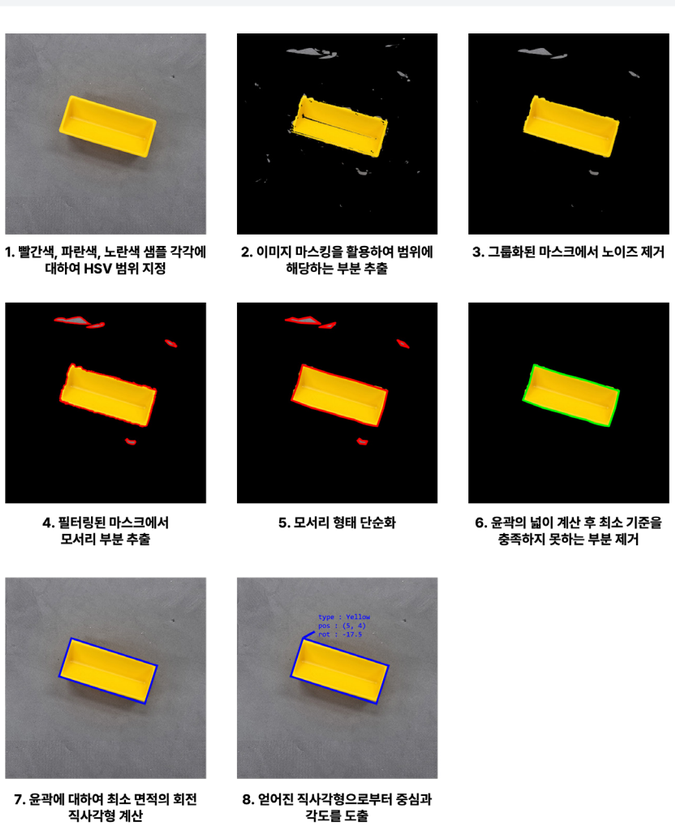
4. OpenCV

머신러닝 외에도 머신러닝에 비해 빠르고 가벼운 이미지 처리를 통해 물체를 감지하는 방법을 연구하였음. 이를 위하여 이미지 처리 라이브러리인 OpenCV를 사용함. 기본적인 아이디어는 샘플과 바닥 사이의 급격한 색상차를 활용해 감지하는 것으로, 색의 특성을 잘 반영하는 색상 표현 방식인 HSV 형식을 사용하여 바닥과 샘플을 분리함.

샘플은 경기장 바닥에 비해 상대적으로 채도가 높다는 특성을 활용해 채도(Saturation, S Value) 값을 중심으로 바닥과 샘플을 분리하였으며, 추가적으로 다양한 조명 환경에서 샘플이 가질 수 있는 최소 명도(Value, V Value)를 실험적으로 구하여 더욱 정밀한 샘플 탐지를 구현함. 샘플 사이의 색상 구분은 색조(Hue, H Value)를 사용하였으며, 그 범위를 넉넉하게 하여 샘플 감지에 방해가 되지 않도록 처리함.



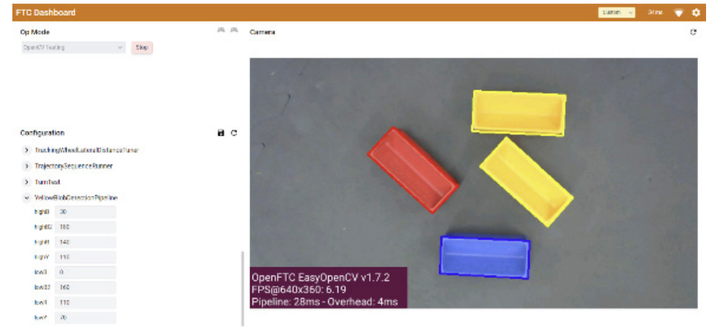
📌 알고리즘 작동 방식



1. 빨간색, 파란색, 노란색 샘플 각각에 대하여 HSV 범위 지정
2. 이미지 마스킹을 활용하여 범위에 해당하는 부분 추출
3. 그룹화된 마스크에서 노이즈 제거
4. 필터링된 마스크에서 모서리 부분 추출
5. 모서리 형태 단순화
6. 윤곽의 넓이 계산 후 최소 기준을 충족하지 못하는 부분 제거
7. 윤곽에 대하여 최소 면적의 회전 직사각형 계산
8. 얻어진 직사각형으로부터 중심과 각도를 도출

첫 테스트는 코드 작성이 편리한 Python에서 진행했으며, Python에서 잘 작동함을 확인한 후 Java 코드로 옮겨 테스트를 진행함. Java의 'imageproc' 모듈과 'easyopencv' 모듈을 활용하여 웹캠에서 받아오는 실시간 영상을 처리하여 샘플을 인식하도록 코드를 작성함.

이후 Control Hub에 웹캠을 연결하여 테스트를 진행하였음. HSV 범위는 상황에 따라 달라질 수 있기에 FTC Dashboard를 활용하여 소스코드를 새로 빌드하지 않더라도 값을 수정하고 테스트할 수 있도록 구성함. 테스트 결과과 Python에서 실행했던 환경과 마찬가지로 샘플만을 정확하게 인식할 수 있음을 확인하였음.



다만, 색상의 차이를 기반으로 한 방식인 만큼, 같은 색의 샘플이 붙어있으면 두 샘플을 분류하지 못하는 문제가 발생하였음. 같은 색 끼리 묶어서 하나의 그룹으로 인식하는 방식을 사용하였기 때문에, 같은색 샘플이 2개 붙어있는 경우에 하나의 큰 사각형으로 인식하였음. 이를 해결하기 위해 2가지의 방안을 고안함.

1) 노이즈 제거 연산의 간소화

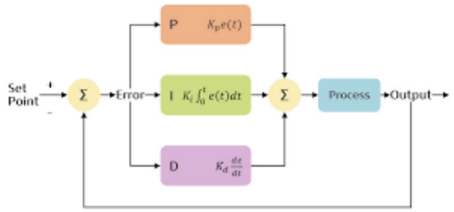
마스크의 노이즈를 제거하기 위해 튀어나온 부분을 제거하는 Open연산과 들어간 부분을 제거하는 Close연산이 활용함. 하지만 가까이 있는 2개의 샘플을 구분하려면 그 사이에 있는 민감하게 인식해야할 필요성이 있으나, 픽셀 사이의 공간이 굉장히 얇기 때문에 Close연산을 통해 제거됨. 따라서 이 연산을 하지 않음으로써 가까이 있는 2개의 샘플을 구분할 수 있음.

2) 크기를 기반으로 한 구분

두 샘플이 완전히 붙어있어 그 사이의 틈새를 인식하기 어려운 경우에는 위의 방법으로 샘플을 구분할 수 없음. 하지만 카메라가 지면으로부터 항상 같은 거리 떨어져 지면을 수직으로 비춘다는 가정 하에, 샘플 하나가 만드는 사각형의 크기는 동일하므로, 사각형의 크기를 비교하여 특정 부분에 개의 샘플이 겹쳐있는지 확인할 수 있음.

5. PID 제어기

PID 제어기(Proportional-Integral-Derivative Controller)는 제어 시스템에서 널리 사용되는 피드백 제어 방식으로, 목표값(설정값)과 실제값(측정값) 간의 오차를 최소화하기 위해 사용됨. PID 제어기는 세 가지 구성 요소(비례, 적분, 미분)를 결합하여 시스템의 동작을 제어함. 로봇에서 정밀하게 움직여야하는 부분은 주로 서보 모터를 사용하지만, 서보 모터는 힘과 속도에 한계가 있어서 리니어를 움직이거나 무거운 팔을 회전시키는 동작에는 필연적으로 DC 모터를 사용해야함. DC 모터는 서보 모터와 다르게 전압에 기반해 속도가 조절되며, 원하는 위치로 회전시키기 위해서는 직접 엔코더 값을 활용하여 모터의 전압을 조절해야함. 또한, DC 모터는 브레이크가 없기에 외력을 무시하고 특정 상태를 유지하기 위해서는 엔코더 값을 지속적으로 모니터링 하며 전압을 조절해야함. 이러한 과정을 빠르고 효과적으로 수행하기 위하여 PID 제어기의 사용이 필수적임.

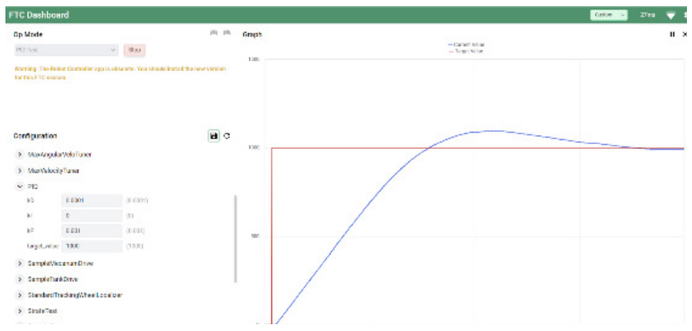


PID는 비례(Proportional), 적분(Integral), 미분(Derivative) 3가지 매개변수를 조절하여 목표 위치를 향하기 위해 필요한 힘을 결정하는 제어 기법으로, 이에 대응하는 3가지 변수는 kP, kI, kD이며, 시간에 대한 오류함수 E(t)에 대하여 아래와 같은 관계식을 통해 PID 값을 결정할 수 있음.

$$MV(t) = k_P E(t) + k_I \int_0^t E(t)dt + k_D \frac{dE}{dt}$$

✅ **상숫값들의 의미**

kP | 현재 상태에서의 오차값의 크기에 비례한 제어작용을 수행
kI | 정상상태(Steady-State) 오차를 없애는 작용을 수행함
kD | 출력값의 급격한 변화에 제동을 걸어 오버슈트을 줄이고 안정성을 향상



FTC Dashboard에서 현재 엔코더 값 (current_position)과 목표 엔코더 값 (target_position)의 변화를 그래프를 통해 시각적으로 확인하면서 k값을 조절해보았음. 그 결과 k값을 적절하게 조절하여 빠르고 정확하게 목표 값에 도달하는 PID 제어기를 설계할 수 있었음.

6. 시각화

프로그래밍 시에 로봇의 상태를 시각적으로 확인하여 효과적으로 소스코드를 작성하고 수정할 수 있도록 FTC Dashboard 사용법을 공부함. FTC Dashboard는 ACME Robotics에서 제작한 로봇 모니터링 웹 앱으로, 아래와 같은 다양한 기능을 제공하는 강력한 툴임.

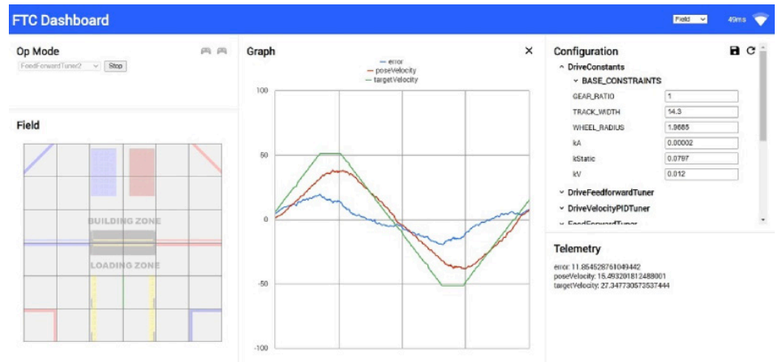
📌 **FTC Dashboard의 기능**

] Telemetry를 통한 그래프 플롯과 필드 **그래픽 제공**

] 실시간 구성 **변수 수정** 및 확인

] **카메라** 스트리밍

] **원격 OpMode 제어** 및 게임패드 지원



FTC Dashboard를 Road Runner와 PID 제어기, 그리고 Vision을 테스트 할 때 사용하며 사용법을 익히고 유용하게 활용할 수 있도록 공부함. 다양한 기능으로 테스트 과정에서도 큰 도움이 되었으며, 로봇 소스코드 작성 과정에도 유용하게 쓰일 수 있으리라 기대됨.

FTC Dashboard 외에도 Road Runner에서 로봇의 경로 설정시에 유용하게 활용할 수 있는 MeepMeep을 설치하고, 사용법을 익힘. MeepMeep은 Noah Bres가 만든 Road Runner Trajectory 시각화 프로그램으로, trajectory 코드를 작성하여 집어넣으면 그 경로를 시뮬레이션 하여 보여줌. 로봇의 자율 주행 코드를 작성할 때, 직접적인 로봇의 주행 없이 경로를 빠르고 안전하게 확인할 수 있는 프로그램이라 생각함.

II. 코드 구조 기획

코드를 효율적으로 작성하고, 유지 보수 및 가독성을 높이기 위하여 코드 구조를 기획함. 특히 역할에 따라 파트(Part), 기능(Feature), 그리고 OpMode로 세분화하여 객체 구조를 디자인함.

1. 파트 (Part)

전체 로봇은 그 작동 방식이 복잡하기에 하나의 객체가 관리하기에는 무리가 있음. 따라서 우리는 로봇을 그 역할에 따라 나누었으며, 이 각각을 관리하는 객체를 'Part Class'라 명명하였음.

📌 **Part가 되기 위한 조건!**

1. **파트의 동작**은 다른 파트들과 **독립**되어야 함.
2. 한 파트는 3개에서 6개 정도의 하드웨어를 제어하는 **적당한 규모**여야 함.
3. 해당 파트의 **하드웨어 제어 권한**을 전적으로 그 파트 객체에 할당할 수 있어야 함.

파트는 할당된 하드웨어의 동작을 전적으로 관리하며, OpMode에 의해 통제될 수 있는 적절한 "명령어 함수"를 제공함. 이를 통해 객체 지향 프로그래밍(OOP)의 다형성(Polymorphism), 캡슐화(Encapsulation), 그리고 추상화(Abstraction)을 달성하여 코드의 관리를 용이하게 함.

2. 기능 (Feature)

다양한 파트 및 OpMode에서 사용할 수 있는 유용한 기능들을 구조화하여 객체로 제작함. 이 객체들은 그 기능에 따라 다양한 형식과 형태를 가질 수 있음. 여러 곳에서 사용될 수 있는 기능들을 객체화하여 객체 지향 프로그래밍 (OOP)의 주된 특성인 캡슐화(Encapsulation)를 달성하여 코드의 중복 사용과 그로 인한 실수를 방지함.

3. OpMode

로봇 컨트롤러 코드의 가장 중추적인 부분으로, 자율 주행 시기의 AutoOp와 원격 조정 시기의 TeleOp 코드를 작성하는 부분임. OpMode에 해당하는 객체들은 각 파트들을 통합하여 전체적인 로봇의 움직임을 총괄 및 제어하지만, 이들은 각 파트가 지원하는 "명령어 함수"를 실행할 수 있는 권한만 가질 뿐, 하드웨어를 직접적으로 제어할 수 없음.

기능을 분리하여 객체를 생성하고, 이들 사이의 유기적인 관계를 구성한 객체지향적인 프로그래밍을 통하여 관리하기 쉽고, 오류를 구조적으로 예방할 수 있는 코드 구조를 구축함.

✅ **기존 방식과의 차이점!**

1. 하드웨어 객체

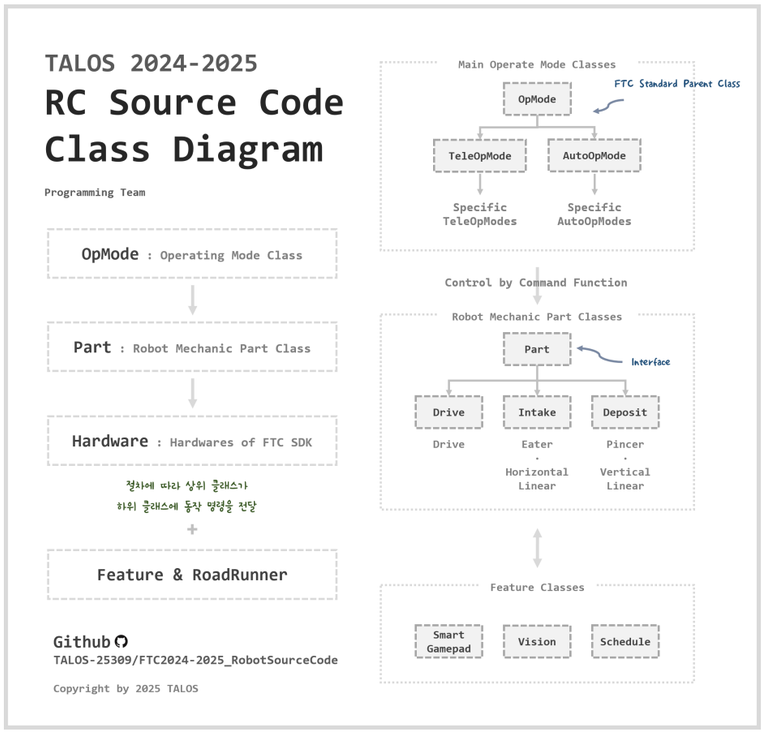
기존의 방식은 하드웨어 객체를 새로 정의하여 파트에서의 하드웨어 사용을 용이하게 하였으나, 파트가 하드웨어를 직접적으로 조작할 수 없어, 그 기능이 제한되고 하드웨어 객체가 복잡해지는 문제점이 있었음. 이로 인하여 이번 시즌 코드에서 하드웨어 객체 레벨은 제거됨.

2. 파트 객체의 명령어 작동 방식

현재 파트 객체가 지원하는 "명령어 함수"는 Java의 기본 메소드 형태이며, 즉각적인 명령을 지원하고, 필요한 경우에만 Update 함수를 통한 후처리를 지원함. 하지만 기존 파트 객체는 비동기 프로그래밍에 중점을 두어 문자열 형태의 명령어를 전달하고, 명령을 예약했다가 Update 함수에서 이를 처리하는 방식을 활용하여 오버헤드 발생, 잘못된 명령어 실행, 실행시간 지연 등의 문제가 발생하였음. 이번 시즌에는 이러한 문제를 해결하여 소스코드의 구조를 새로 디자인함.

3. 기능 객체

기능 객체 레벨을 새로 개발함. 이로 인하여 기존에 비해 중복되는 기능에 대한 코드 재사용이 줄어들었으며, 객체를 통한 체계적인 관리로 코드의 가독성을 높이고, 효율을 극대화 시킴.



III. 파트 객체 개발

1. 파트 인터페이스 및 기본 구조 개요

모든 파트는 파트 인터페이스 (Part Interface)를 상속받아야 함. 파트 인터페이스는 파트에 필수적으로 존재해야하는 함수를 강제하며, 이를 통해 다형성 (Polymorphism)을 구현하여 OpMode에서 공통된 기능을 쉽게 접근할 수 있도록 제작함. 이에 해당하는 함수는 아래와 같음.

📌 **공통된 함수 목록!**

Init | 초기화 함수, Hardware Map과 Telemetry를 매개변수로 받음

Update | 주기적으로 확인해야하는 작업을 진행하는 업데이트 함수

Stop | 비상 상황에서 모든 기능을 정지하고 수동 모드에 진입시킬 수 있는 함수

그 외에 모든 파트는 각자 고유한 일반 함수와 "명령어 함수"를 가질 수 있음. 고유 함수는 내부에서 사용되는 Private 함수이며, "명령어 함수"는 외부에서 명령을 내릴 수 있도록 파트가 제공하는 함수로, Public 함수이며 함수명이 cmd[명령어]로 구성됨.

모든 파트는 3가지 종류의 클래스에 의해 정의됨.

메인 파트 클래스
Main Parts Class

서브 파트 클래스
Sub Parts Class

파트 상수 클래스
Part Constants Class

모든 파트는 파트 명에 해당하는 메인 파트트 클래스를 Public 클래스로 가지며, 외부에서는 이 메인 파트 클래스에만 접근할 수 있음. 그 외에는 파트를 작게 쪼개어 객체화한 Default 클래스인 서브 파트 클래스가 존재하며, 서브 파트 클래스들은 메인 파트 클래스에 의해 제어됨. 대부분의 경우 하드웨어는 해당 서브 파트 클래스가 제어하며, 서브 파트 클래스가 메인 파트 클래스를 위한 "명령어 함수"를 제공하고, 메인 파트 클래스는 서브 파트 클래스의 "명령어 함수"를 통합한 고유의 "명령어 함수"를 가짐. 파트 내의 유기적인 연결과 로봇이 구동중인 상태에서 빠르게 상수를 바꿀 수 있도록 파트 상수 클래스를 제작하였으며, 내부의 값은 정적(Static) 값으로 모든 클래스가 그 값을 공유함.

2. Drive Part

로봇의 구동부를 관리하는 파트 클래스임. Gamepad를 통한 로봇의 병진 및 회전 운동을 관리하며, Road Runner를 사용한 Localization과 Following Trajectory를 지원함.

메카넘 휠을 세밀하게 조정하여 상하좌우 뿐만 아닌 모든 방향으로의 자유로운 움직임을 지원하며, 회전과 병진 운동을 동시에 수행할 수 있음. 뿐만 아니라 Localization을 활용하여 로봇의 방향과 관계 없이 절대적인 좌표계에서 움직일 수 있는 필드 중심 주행을 지원한다는 점에서 차별화된 강력한 주행 시스템을 제공함.

1) Class 구조

클래스	구분	역할
Drive	Main Part	구동부 및 Road Runner 관리
DriveConstants	Part Constants	속도, 위치 데이터 등 구동 속성값 관리

2) 명령어 목록

명령어 함수	매개 변수	기능
Drive	x, y, omega	입력을 기반으로 주행
DriveSlowly	x, y	병진 운동을 통한 미세 조정 지원
AutoAlignBasket		그물(Basket) 위치로 이동하는 매크로
AutoAlignSpecimen		관찰 구역으로 이동하는 매크로
AutoAlignSubmersible		Vision을 활용한 Intake 자동 조준 매크로

3. Intake Part

Sample과 Specimen을 습득하는 장치 (Intake)를 관리하는 클래스임. 엔코더를 활용한 수평 리니어 제어와 서보 모터 제어를 지원함. 수평 리니어는 PID 제어기기를 활용하여 특정 위치에 고정될 수 있도록 하였으며, 이터는 서보의 회전 범위를 파트 상수 클래스를 통하여 지정하여 세밀하게 움직임을 제어함.

조작을 간편히 하기 위해 습득 과정을 4가지 단계로 구분함. currentStep 변수에 현재 단계를 저장하고, 단계에 따라 (0) ~ (3) 의 함수를 실행함. 실행 순서는 선형적이지 않기 때문에 runNextStep와 runPrevStep 함수로 단계를 변경함.

1) Class 구조

클래스	구분	역할
Intake	Main Part	Intake 전체 관리 및 총괄
IntakeConstants	Part Constants	리니어 엔코더 값, 서보 범위 등 관련 속성값 관리
Eater	Sub Part	샘플을 빨아들이는 이터의 작동 및 움직임 관리
HorizontalLinear	Sub Part	PID를 사용한 수평 리니어의 움직임 제어

2) 명령어 목록

명령어 함수	매개 변수	기능
(0) StretchLinear		리니어 확장 및 리니어의 자유로운 조작
(1) Eat		이터 하강 및 샘플 습득
(2) Vomit		샘플 제거 및 이터 상승
(3) Transfer		Intake에서 Deposit으로 전달
runPrevStep		이전 step의 함수 실행
runNextStep		다음 step의 함수 실행
AutoStretch		정해진 위치로 수평 리니어 확장
AutoRetract		로봇 안으로 들어오도록 수평 리니어 수축
ManualStretch ManualRetract ManualStop EaterRun		컨트롤러의 명령에 따른 수동 리니어 확장 컨트롤러의 명령에 따른 수동 리니어 수축 수동 상태에서 PID로 리니어 고정 이터 회전을 통한 샘플 습득
EaterStop		이터 정지
AutoRotate		샘플 감지에 따른 이터 회전
ManualRotate	direction	컨트롤러의 명령에 따른 이터 회전
ArmUp		이터 상승
ArmDown		이터 하강
runCurrentStep		현재 step에 맞는 함수 실행

4. Deposit Part

Intake에서 전달받은 Sample과 Specimen을 각각 Basket과 Chamber에 놓고 오는 클래스. 엔코더를 활용한 수직 리니어 제어와 서보 모터 제어를 지원함. 수직 리니어는 PID 제어기기를 활용하여 특정 위치로 정확히 이동할 수 있도록 하였으며, 집게는 고정된 회전 범위에서 운동함.

1) Class 구조

클래스	구분	역할
Deposit	Main Part	Intake 전체 관리 및 총괄
DepositConstants	Part Constants	리니어 도달 위치 등 관련 속성값 관리
Claw	Sub Part	샘플을 운반하는 집게의 움직임 관리
VerticalLinear	Sub Part	PID를 사용한 수직 리니어의 움직임 제어

2) 명령어 목록

명령어 함수	매개 변수	기능
DepositSample	location	샘플을 2가지 바구니 중 하나에 득점
DepositSpecimen	location	견본을 2가지 체임버 중 하나에 득점
Return		수직 리니어 수축 및 집게 조정
ManualStretch		컨트롤러의 명령을 통한 수직 리니어 확장
ManualRetract		컨트롤러의 명령을 통한 수직 리니어 수축
ManualStop		수동 상태에서 PID로 리니어 고정

IV. 기능 객체 개발

1. Smart Gamepad

FTC SDK에서 기본적으로 제공하는 Gamepad 객체는 각 버튼이 눌렸으면 참, 눌리지 않았으면 거짓의 값을 나타냄. 따라서 어떤 버튼이 클릭되었을 때 어떠한 명령을 처리하기 위해서는, 현재 버튼 상태가 참임을 확인하는 것 뿐만 아니라, 이전의 버튼 상태가 거짓임을 추가로 확인해야함. 이를 위하여 게임패드를 복사하고, 조건을 구성하는 것은 번거롭고 실수를 유발하기 쉬우므로, 이 기능을 객체화하여 처리함.

Smart Gamepad는 자동으로 이전 게임 패드의 상태를 저장하며, 직관적인 함수 호출을 통해 다양한 상태에 대한 참/거짓 값을 알아낼 수 있음. 또한, Getter를 활용해 등록한 게임패드에 직접 접근하여 값을 확인하는 것 역시 가능하도록 디자인 함.

📌 **Smart Gamepad의 함수!**

isPressed | 버튼이 새로 눌렸는지 확인 (Now True + Prev False)
isReleased | 버튼이 떼어졌는지 확인 (Now False + Prev True)
isHeld | 버튼이 눌려있는지 확인 (Now True + Prev True)
isFree | 버튼이 눌려있지 않은지 확인 (Now False)

2. Vision

OpenCV 모델을 이용한 샘플 구분 기능을 구현한 클래스. Intake 파트에서 타겟 샘플을 찾는 과정을 단순하게 사용할수 있도록 설계함. setTargetColor은 목표 색을 설정하고 detectTarget은 가장 가까운 샘플의 정보를 전달함. Sample의 정보를 쉽게 관리하기 위해 Sample 클래스를 만들어 위치와 각도, 색을 저장함. 웹캠 관리 및 타겟 샘플 감지는 독립적인 pipeline을 만들어서 영상 처리에 필요한 기능을 순차적으로 진행함. 상세한 과정은 Part 1 - 4.openCV 에 설명되어 있음. 카메라의 위치에 가장 가까운 샘플을 찾아내는 기능은 추가로 구현함.

전체적인 로봇 작동의 효율을 높이기 위하여 telemetry를 이용해 영상을 전달하는 기능을 최소화함. 또한 샘플을 감지할 필요가 없는 경우에 Vision 파트의 작동을 중지할수 있도록 turnOn, turnOff 함수를 설정함.

3. Schedule

일련의 명령을 특정 시간을 간격으로 하여 처리하기 위하여 단순히 명령어를 나열하고, 그 사이에 딜레이를 넣으면, 그 시간동안 로봇이 작동 불능 상태가 되어 위험할 수 있음. 따라서 특정 일이 진행중일 때 다른 일을 동시에 처리할 수 있는 비동기적 프로그래밍을 수행해야함.

비동기 프로그래밍을 하는 가장 대표적인 방법은 Java의 쓰레드를 사용하는 것임. 그러나 쓰레드는 잘못 사용하면 오히려 성능이 떨어질 뿐더러, 메모리 동시 접근 시에 버그 및 오류가 발생할 수 있음. 또한, 하드웨어 접근은 특히 오버헤드가 많이 발생하기에, 동시에 하드웨어에 접근하면 치명적인 오류가 발생할 수 있음. 쓰레드 안전 (Thread Safety)한 시스템을 작성하는 것은 매우 어려우며, 때로는 비효율적이므로, 메인 쓰레드에서 주기적인 호출을 통해 비동기 프로그래밍을 구현할 수 있는 새로운 시스템을 객체 형태로 개발함.

Schedule은 정적 Public 클래스로, 모든 영역에서 접근 가능함. Schedule 객체는 일(Task)를 Runnable 타입의 수행할 일과 수행할 시기를 입력받아, 우선 순위 큐 (Priority Queue)에 저장하여 주기적인 Update 함수가 호출될 때, 내부 타이머를 확인하여 정확한 시기에 수행할 일을 실행시킬 수 있음. 이를 통해 시간 기반 메인 쓰레드에서의 비동기 프로그래밍을 구현함.

V. OpMode 개발

1. OpMode 객체 개요

OpMode 객체는 로봇을 작동시키는 메인 클래스로, 로봇의 전체적인 구동을 총괄함. OpMode는 Part 객체를 정의하고, 드라이버의 입력 혹은 자율주행 절차에 따라 Part의 명령어 함수를 활용하여 적절한 명령을 내림. 또한, 각각의 파트들과 Schedule 객체를 지속적으로 업데이트 하면서 메인 루프 안에서 비동기 명령을 처리하는 역할을 수행함. 또한 Part에 직접적인 명령을 내리는 것 외에도 Feature 객체에 해당하는 Vision과 Smart Gamepad 객체의 정보를 통합하여 복합적인 명령을 구성하는 기능을 수행함.

2. TeleOpMode (Driver Controlled Period)

TeleOpMode 객체는 원격 조정 기간 (Driver Controlled Period) 동안 게임 패드를 통한 드라이버의 명령을 전달받아 이를 재조합하여 Part에 명령을 하달하는 역할을 수행함. 전체적인 구조는 FTC SDK의 OpMode를 상속받아 구성됨.

TeleOpMode의 가장 중심적인 기능은 컨트롤러 (게임 패드)의 입력값을 통해 드라이버의 명령을 정확히 전달받고, 이를 활용해 로봇의 전체적인 구동을 총괄하는 것임. 따라서 이 객체를 정의할 때에는 게임 패드의 각 버튼에 무슨 역할을 할당하고, 이것을 어떤 파트의 어떤 명령어 함수, 혹은 어떤 Feature의 기능을 사용할지를 정의하는 것이 중요했음.

프로그래밍 팀은 드라이버들과 긴밀히 소통하며 최대한 로봇을 조작하기 편리하고 효과적인 방법을 찾기 위해 많은 토의를 거쳤으며, 다양한 테스트를 통하여 최적의 방법을 찾고자 노력함. 이를 통해 최종적으로 컨트롤러의 각 입력에 역할을 부여하고, 이를 명령어 함수들과 연결시킴으로써 TeleOpMode 객체를 제작함.

3. AutoOpMode (Autonomous Period)

AutoOpMode 객체는 30초 동안의 자율 주행 시간 (Autonomous Period) 동안 로봇이 자율적으로 구동할 수 있도록 하는 일련의 명령을 관리하는 객체임. 기본적으로 주어진 절차에 따라 명령을 순차적으로 이행하며, 각종 센서와 카메라 입력값을 활용하여 정확하게 명령을 이행하는 것을 목표로 함. 전체적인 구조는 FTC SDK의 LinearOpMode를 상속받아 구성됨.

AutoOpMode의 가장 기본적인 구조는 Road Runner의 Trajectory Sequence 생성의 형태를 가짐. Trajectory Sequence를 빌드하고 실행시킴으로써 정확한 위치로의 자율 주행을 구현하고, 그 사이에 마커를 삽입하여 Drive 파트를 제외한 다른 파트에 명령을 전달함으로써 자율적인 로봇 구동을 달성함. 만들어진 Trajectory Sequence는 Drive 파트에 의해 실행됨.

1) [전략 1] Basket에 Sample 넣기 전략

이 전략은 로봇이 중립 샘플 가까이에 배치되었을 때 사용하는 전략으로, 로봇에 사전에 로딩되어있는 샘플을 먼저 가장 높은 Basket에 넣고, 그 뒤로 바닥에 있는 3개의 중립 샘플을 순차적으로 집어 Basket에 넣는 것을 목표로 함. 또한, 이 모든 과정이 끝나면 잠수정에 붙어 주차를 수행함.

2) [전략 2] Chamber에 Specimen 걸기 전략

이 전략은 로봇이 자신의 컬러에 해당하는 샘플 가까이에 배치되었을 때 사용하는 전략으로, 로봇이 사전에 로딩되어있는 견본을 먼저 높은 Chamber에 건 후, 바닥에 배치된 모든 컬러 샘플을 순차적으로 관찰 구역으로 밀어넣어 견본으로 만들어 최종적으로 모든 견본을 높은 Chamber에 거는 것을 목표로 함. 또한, 이 모든 과정이 끝나면 잠수정에 붙거나 혹은 상황에 따라 관찰자 구역으로 들어가 주차를 수행함.

이 과정에서는 Road Runner의 정확한 구동이 필수적임. 따라서 이를 위해서는 정확한 튜닝과 함께 정확한 경로 설정이 뒷받침 되어야 함. 이를 위하여 프로그래밍 팀은 많은 시간을 투자하여 어떠한 전략이든 최대 점수를 달성하기 위해 최선을 다함.

TALOS

APOC 2025
Engineering Note

We are Team TALOS from South Korea,
and we've come to tell the final story to conclude
our three seasons of FIRST Tech Challenge.