#25309

TALOS

FIRST Tech Challenge
2023-24 Season

Engineering Notebook

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

This Engineering Note for the FIRST Tech Challenge 2023-24 Season was written by the team members of Team TALOS (#25309) and was edited by Joonsung Kim (23-029@ksa.hs.kr).

Contents.

#25309
TALOS

Talos #25309

About Us

We’re the Champions from Korea.
Take a look at what we’ve been through.

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

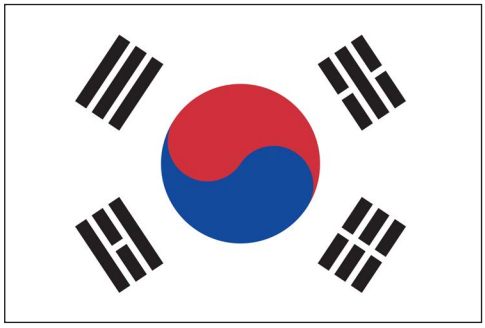
About TALOS

TALOS is a Robotics Engineering team from **Republic of Korea**. We are part of a research group KROS, based in **Korea Science Academy of KAIST**. We are registered as Team #25309.

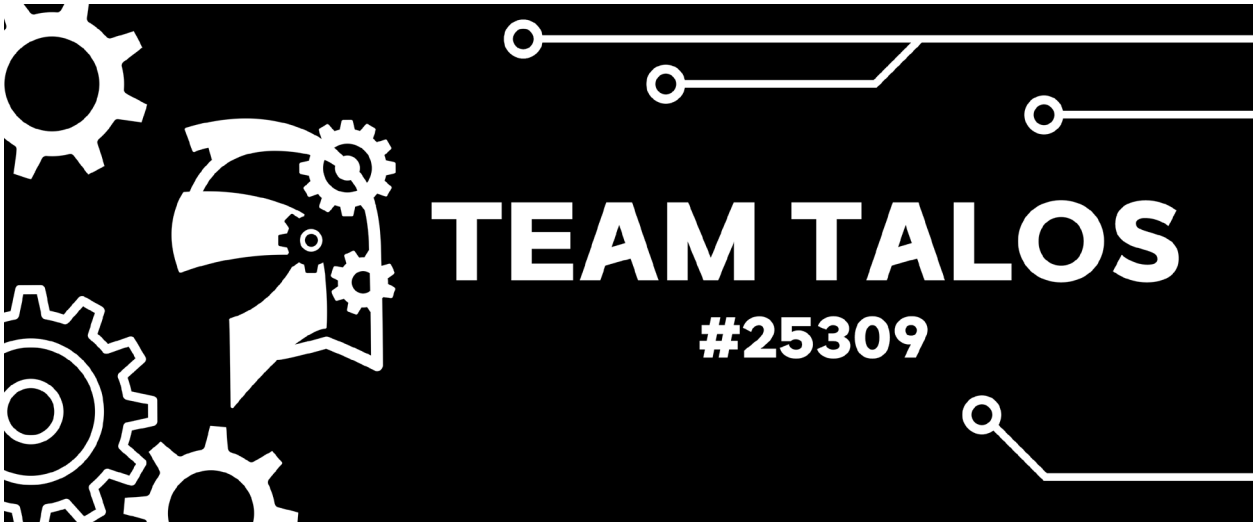
We are sponsored by **Korea Science Academy of KAIST** and the **Ministry of Science and ICT**.

KAIST부설 한국과학영재학교

Ministry of Science and ICT



Talos is the first robot created by Hephaestus in Greek mythology, and represents our determination to create robots as awesome as Talos, whom Hephaestus created through reason.



History.

TALOS has participated in multiple KRC (Korea Robot Championship) competitions since the 2018-19 season.

2018-19 Season
Rover Ruckus

2019-20 Season
SKYSTONE

2021-22 Season
FREIGHT FRENZY

2022-23 Season
POWERPLAY

We won the **Inspire Award** at the 2023-24 Korea Robot Championship. It's our first time participating in the FIRST Tech Challenge World Championship.

Our team.

We have a team of 15 students; 3 programmers and 12 builders. Students of the 21st, 22nd, and 23rd batch have participated, and seven of us will be participating in the World Championship.

Coaches.

Hosook Kim (Advisor)

Department of Math and Computer Science,
Korea Science Academy of KAIST

Seungchan Lee (Advisor, Alumni)

19th batch, Korea Science Academy of KAIST
2021-22 Season Captain

FTC Members.

Howon Chang (FTC Captain)

Builder, 22nd batch
Design and engineering of eater part

Gyujin Rieh

Builder, 22nd batch
Robot frame engineering

Sugi Kim

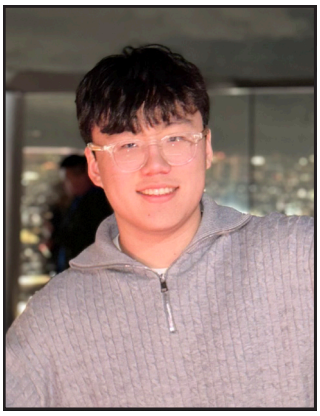
Builder, 22nd batch
Design and engineering of eater part

Mingwon Kim

Builder, 22nd batch
Design and engineering of eater part

Joonsung Kim

Builder, 23rd batch
Design and engineering of eater part



Gyuseo Shim

Builder, 21st batch
Robot frame engineering

Minjun Oh

Builder, 21st batch
Design and engineering of eater part

Not all of our team could join in the World Championship due to school's midterm schedules. However, all of our members contributed in revising our robot for the FIRST Tech Challenge.

Each and every contribution for this competition was crucial for this to work. After competing for KRC and preparing for FTC, we are more than grateful to be in this team.

KRC Members.

Taewoo Yu (KRC Captain)

Builder, 22nd batch
Design and engineering of eater part

Jeongwoo Hong (KRC Captain)

Builder, 23rd batch
Robot frame engineering

Hyeonbeen Kang

Builder, 22nd batch
Design and engineering of eater part

Uijin Jung

Builder, 22nd batch
Design and engineering of eater part

Sungbin Choi

Builder, 22nd batch
Design and engineering of eater part

Jihong Min

Builder, 21st batch
Robot frame engineering

Sanghyeon Park

Builder, 21st batch
Design and engineering of eater part

Chan Park

Builder, 21st batch
Design and engineering of eater part

#25309
TALOS

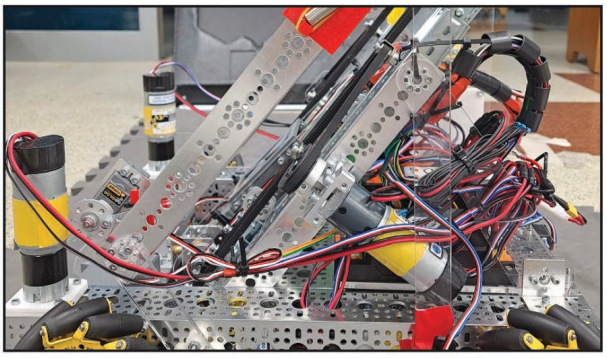
KRC Preparation

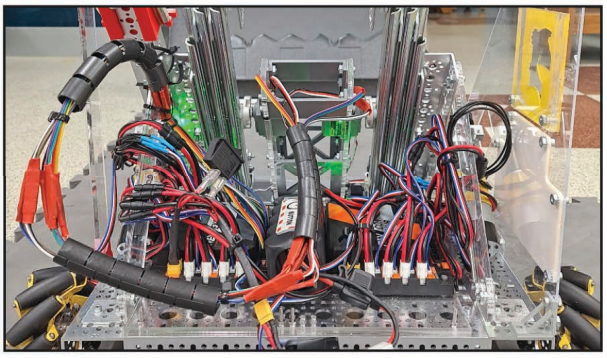
Robot & Strategy

We’re the Champions from Korea.
Look at what we’ve been through.

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

Say Hi to our robot.





FIRST Tech Challenge 2023-24 Season - Robot “Ui”
Duration of work - 13 days total (January 12-24)

TEAM TALOS designed and built the robot “Ui” to perform the mission as efficiently as possible. We analyzed videos of foreign teams’ matches to see how they played and the realistic problems they faced, and designed the robot with our own original ideas. In a total of 13 days, we were able to complete the robot based on the collaboration and efforts of all team members.

The robot is based on the body of goBILDA and is made by combining various parts such as plastic (3D printed) and acrylic flat plates, and above all, we focused on making it stable to play with symmetry and center of gravity within the specified specifications.

The robot “Ui” consists of a body, linear slide, pincers, eater, and airplane launcher.

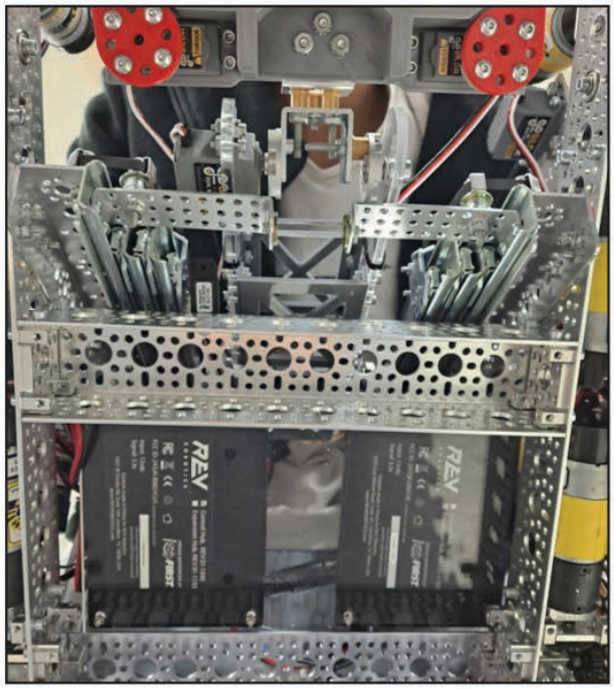
1. Body

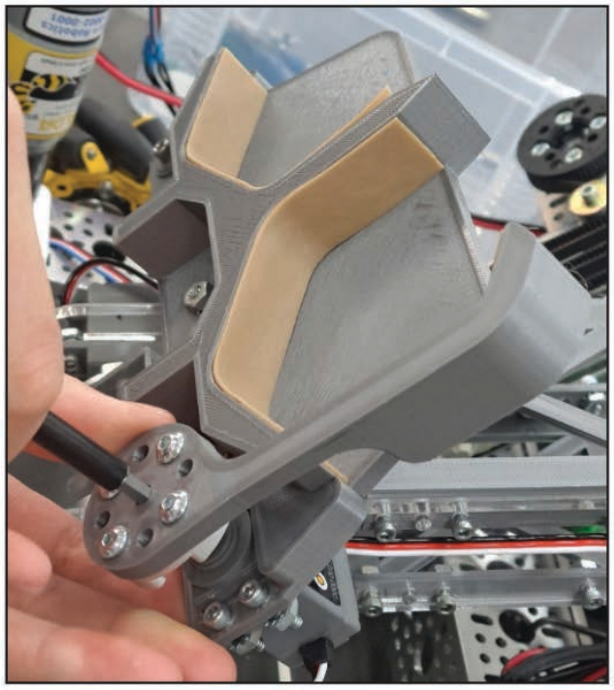
The body is a structure consisting of the basic frame and wheels, and is connected to the linear slide, eater, and airplane launcher. It is also equipped with a Control Hub, Expansion Hub, optical distance sensor, and battery.
Optical sensors are mounted on the right and front to recognize the Team Prop without rotating during the Autonomous Period.

2. Pincers

This is the part that picks up and flips the pixel, and is connected to the linear with a servo motor.

- **Arm:** Rotates the pincer part
- **Wrist:** Adjusts the angle of the pincer
- **Finger:** Grabbing and releasing the pixels





Each part is driven by a servo motor, and we tested various versions to design the most stable way to pick up hexagonal-shaped pixels.

3. Linear Slide: 4-stage viper-slide (belt-type, GoBlida)

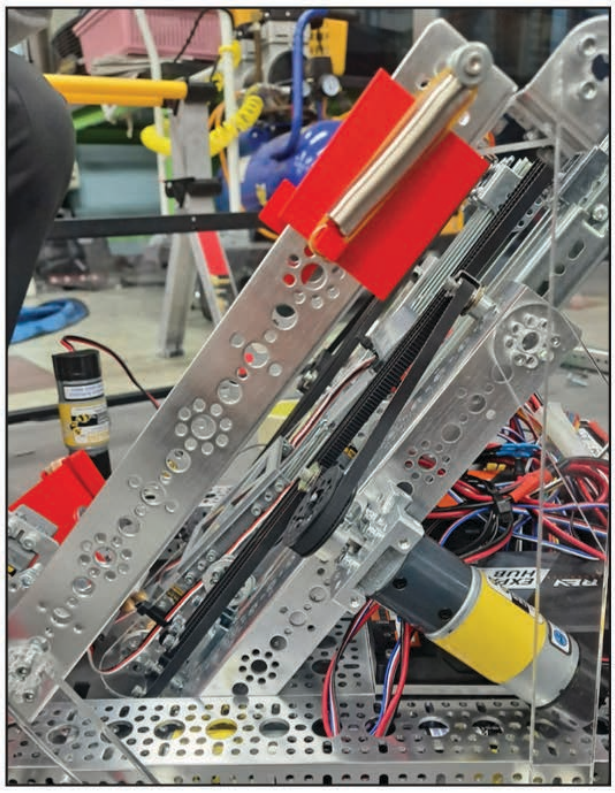
This is responsible for raising the tongs to the proper height to place the pixels on the backboard, and for hanging the robot body on the truss in the endgame. It extends up to 3 levels by connecting chains to DC motors on either side of the body.

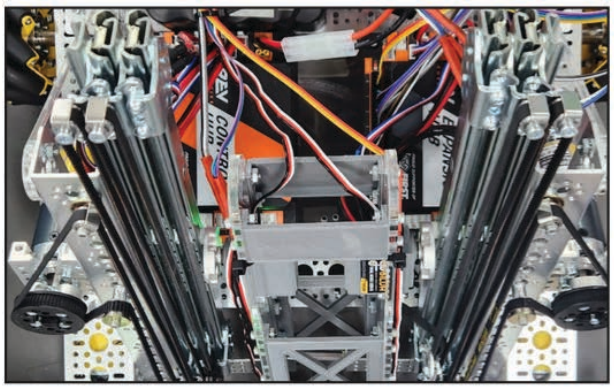
4. Stack Eater

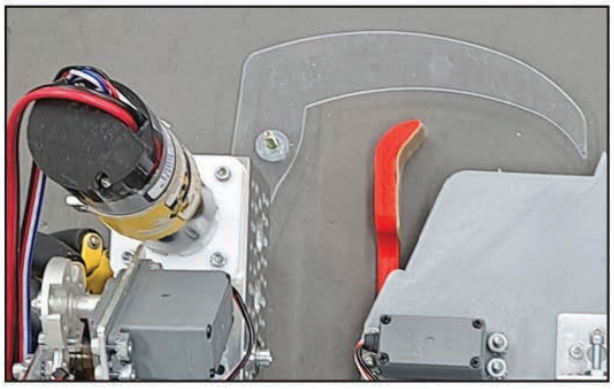
The Stack Eater is used to collect stacked pixels or pixels that are difficult to pick up with a pincer. A sickle-shaped structure made of acrylic is connected to a DC motor and rotates within a set angle.

5. Airplane launcher

This is responsible for launching the paper airplane that is pre-mounted on the robot and fixed to the body. The 3D printed airplane case is connected to an elastic band and a spring, and its position is fixed using a servo motor. It is adjusted to the appropriate angle to ensure that the paper airplane reaches a certain position.







Game Strategy.

Strategies

TEAM TALOS conceived a strategy and built a robot that projected the strategy in order to achieve a good score at Center stage. Once the robot was built, they continued to practice and refine their strategy.

TEAM TALOS continues to improve with each practice, reducing their flaws and mistakes. Currently, they have run 13 practices and the results are shown in the graph.
As the practice progressed, we were able to improve our score as shown in the graph.

While the driver-controlled period is important on the center stage, we believe that the autonomous period and the

endgame period are the most important, so we designed our strategy to get the most points possible in the autonomous period and the endgame.

1. Autonomous Period

In this period, the goal is to detect an object with a distance sensor within 30 seconds, place a purple pixel on the line where the object is located for 20 points, place a purple pixel in the backdrop at a predetermined location for 20 points, and then park the car within the line for 5 points, for a total of 45 points.

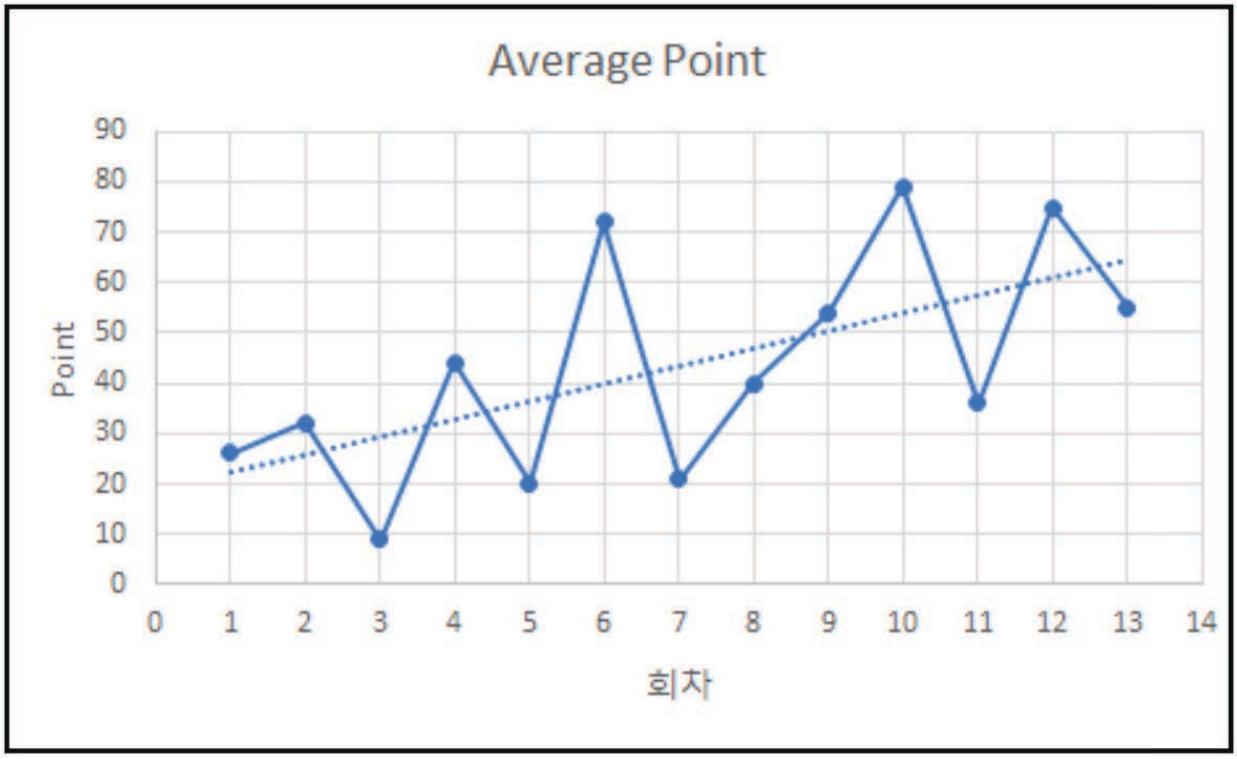
2. Driver Controlled Period

During this period of 1 minute and 30 seconds, we plan to take two pixels from the human player to complete one mosaic, and then repeat the process of taking a stack of pixels from the stacker that is relatively close to the BACKDROP, with the goal of placing 8 pixels on the BACKDROP. We aim to score 24 points for 8 pixels, 10 points for completing one mosaic, and 10 points for height, for a total of 44 points.

We plan to be flexible and change our strategy to match our alliance team's total and score as many points as possible.

3. Endgame

In this segment, we are aiming to score 30 points for launching the airplane in the right place for 30 seconds and 20 points for attempting to hang on to the pole for the remaining time, for a total of 50 points.


Average Point
Point
90
80
70
60
50
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
회차

#25309
TALOS

Builder

KRC Preparation Engineering Note

We’re the Champions from Korea.
Look at what we’ve been through.

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

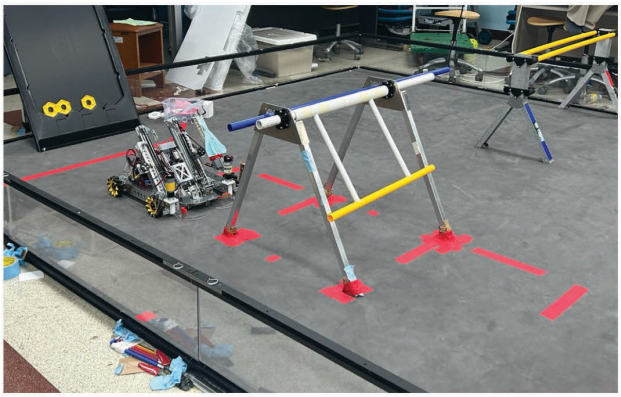
Jan 13th, Sat

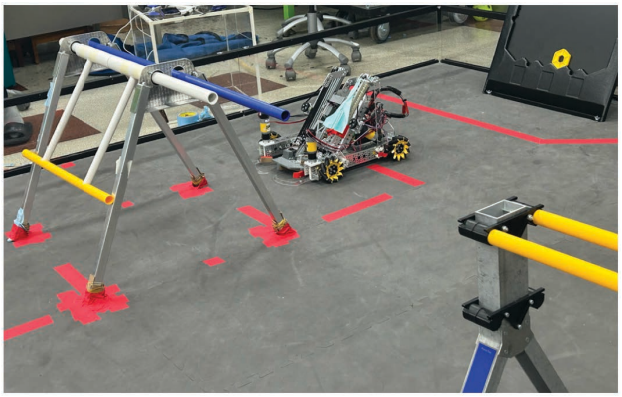
1. Making a Linear Slide

After purchasing goBUILDA's 4 Stage Viper-Slide (Belt-Type) Linear, we made it by referring to the production manual on the website. When installing the linear slide on the robot body, we checked that the height of the truss is higher than 35cm.

2. Making the playing field

While building the robot, we built a stadium to test its operation in an environment similar to a real competition. We purchased a quarter of the stadium set to build the center truss, and used some of last year's stadium set and PVC pipes to build another smaller truss.





The backdrop and obstacles were fixed in place. Also, parking space and positions for the autonomous period were marked with red and blue colored tapes.

Jan 14th, Sun

1. Overall robot design

While designing the overall robot, we considered various ways to pick up and discharge pixels and discussed the advantages and disadvantages of each.

1) Pincher type vs. rotor type

[pincer type]

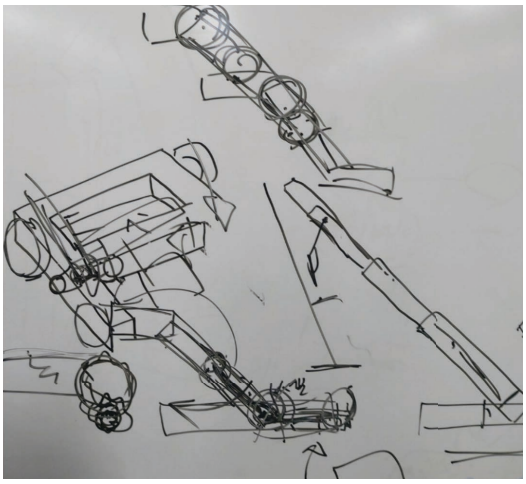
Structure :: Uses pincers and uses an arm shape that connects the linear and pincer.

Pros :: Can drop pixel far away from linear while using arm form in autonomous period.

→ By dropping from a low position, you can place the pixel wherever you want, and it doesn't need to be raised to a high position, reducing the time it takes to place it.

Cons :: The increase in torque when using the arm puts a lot of strain on the motor, increasing instability. The use of gears to overcome this means that it takes longer to turn the pincers. Additionally, when the pincer is built, it is not possible to build a horizontal pincer without first building a vertical pincer to match the width.

Improvements ::



Conventional pincers require servomotors for both the arm, wrist, and pincer parts, which puts a lot of weight on the servomotors fixed to the linear slide. It also takes a lot of time to rotate because you need to apply a steady force against gravity.

→ By passing the tongs under the linear, gravity is used to turn the servo motor, which reduces the load on the motor and saves time by eliminating the wrist servo motor.

[Rotor type]

Structure :: A rotating rotor grabs the pixels from the floor and lifts them onto the treadmill, the treadmill spins to move the pixels to the basket, which is fixed to the linear, which opens up and drops them from the bottom onto the plate

Pros :: Relatively time saving as you only need to move near the pixel to load it onto the robot. Also, no additional strain on the motors due to the lack of arms, i.e. more stable than graspers

Cons:: When moving pixels from the eater part to the basket on the linear slide, it is difficult to expect accurate loading into the basket. Also requires the use of a rotor or conveyor belt as a power source from the floor to the basket, which adds weight. Also, since the basket is directly fixed to the linear slide, the linear slide needs to be maximized in

order to reach the plate and safely place the pixels, which takes time.

2) Basket horizontal vs. vertical

Mockup and acrylic drawing of a basket to hold the pixels from the ether.

Part of the discussion :: When the pixels from the ether go in one by one, can two be placed horizontally exactly where you want them.

Alternatives :: Creating an opening in the center that only allows a single pixel to enter, and then aligning the triangles to limit how far the pixel can move. It was also suggested to restrict the path of the pixel eaten by the ether, making it a simple rectangular model.

To minimize the size of the basket that would be attached to the linear slide, a mockup was created with a box to cover the rectangular model.

Further design and fabrication of the sides and base in acrylic. Added screw holes in each part to determine where to attach the servo motor, considering opening the bottom with a servo motor. To be printed and assembled later.

[Horizontal] : Loading 2 pieces horizontally

Advantages :: Low height makes it easier to attach to the linear. With 2 servos, pixels can be dropped one by one, allowing for fine control when making mosaics.

Cons :: When 2 pixels come in, there is a high chance that they will overlap.

[Vertical] : Loading 2 vertically

Pros :: No jamming

Cons :: Longer vertical distance from the ether, so the pixels can't be raised by the force of the ether alone. Also difficult to link to linear slides.

3) Basket → Pixel loading method.

[Discharge to the back part of the basket]

Receive the pixels from the eater into the basket, linearly raise the basket at a fixed angle, and open the lower part of the basket to discharge the pixels.

[Discharge by rotating the basket]

Receives pixels from the eater in a basket, raises the basket to the linear, and then flips the basket itself to discharge the pixels.

2. Modify the Linear Slide and Build the Body

We modified the linear slide from 4 steps to 3 steps, changed the position of the motor, and discussed whether to stick to the 35cm hurdle height. As a result of the discussion, we chose a method that does not exceed 35cm.

1) Modifying the linear slide

i) 4 stage → 3 stage

After comparing the height of the plate and the height of the linear slide, we decided that using only 3 out of 4 stages would be enough to reduce the weight. We modified the linear slide to have 3 stages.

ii) Motor Position / Support Change

We decided that the size of the plate supporting the linear slide does not need to be large. Also, if the volume of the linear slide is reduced, more space can be freed up. Therefore, the length of the support was shortened, and the position of the motor was changed so that it could be connected to the car body from the existing position. We also considered cutting the support, but decided that this would not be practical.

2) Linear slide attachment location

The height of the hurdles in the stadium is 35 centimeters, and the robot body needs to be built below this height for ease of movement. However, the length of the linear slide is fixed, so the linear slide must be attached to the bottom of the body as much as possible.

[if we give up the 35cm height]

Pros :: The linear slide can be hung above the frames on either side, leaving a large space in the center to attach a pincer that grabs two pixels horizontally at once.

Cons :: The robot is taller than the hurdle, making it impossible to move under the hurdle. Therefore, you must enter the support area next to the obstacle and move to the other side to pass through, i.e. zigzag through the obstacle. Accumulated errors in maneuvering through the obstacle can lead to a crash, which can be very inconvenient and restrictive.

[if height is kept under 35cm]

Pros :: Much easier to maneuver and allows for faster movement.

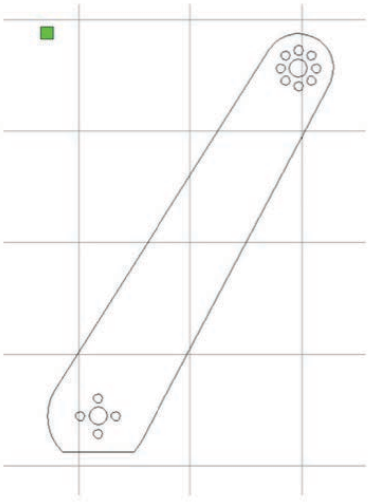
Cons :: There is no space between the linear slides for a pincer to pick up two pixels horizontally. In this case, to use a pincer that grabs two pixels horizontally, the arms of the pincer must be made very long, at least half the length of one linear slide. → Less safe

[Conclusion] → 35 cm or less

Therefore, we removed the motor and its support from the bottom of the linear, and then removed the plate connected to the linear. The remaining part of the linear was connected using a D-shaped frame. At this time, the hole size of the D-shaped frame and the linear part did not match the diameter of the bolts, so I enlarged the holes of the

D-shaped frame with sharp-threaded bolts and connected them. We connected the opposite side of the D-shaped column and the side of the main body at an angle to make it easier to reach the inclined plate. To solve the problem of the first stage linear falling out as it descends, we attached a G-shaped bracket at the bottom of the D-shaped column, but we also used a small flat plate to attach bearings to facilitate the connection with the external motor, which created space and solved the problem of the linear falling out at the same time.

3) Strengthening the stability of the linear slide



The linear slide is attached to the body by laying it down, and we found that it would be very unstable if it was only secured by the frame of the body. Therefore, the linear slide was made of acrylic and attached to the frame with an L-shaped bracket to securely hold it. The frame and acrylic form a triangular shape for added stability.

4) Creating Stadium Parts

Built the supports for the truss structure out of boxes. Installed the existing rods in the truss structure by cutting PVC pipes of the appropriate caliber to the appropriate length.
Cut, partially processed and connected existing rods and PVC pipes of different sizes to realize

the function of a door (a structure that allows only one-way traffic).

4. Where is the front of the car body

[This is the part that spits out pixels]

Why:: The front is the side that does the most difficult part. It's easier to think of the front as the pixel spitting part because it needs to be the most precise.

[It's the eater part]

Why :: Our bodies also have an ingestion part in the front and an expulsion part in the back, so this makes sense. We discussed where the "front" of the body should be, and after discussion, we decided that the front of the body is where the pixels are spit out.

Jan 15th, Mon

1. Finalize the Linear Slide and Mount the Motor

1) Finalize the linear slide

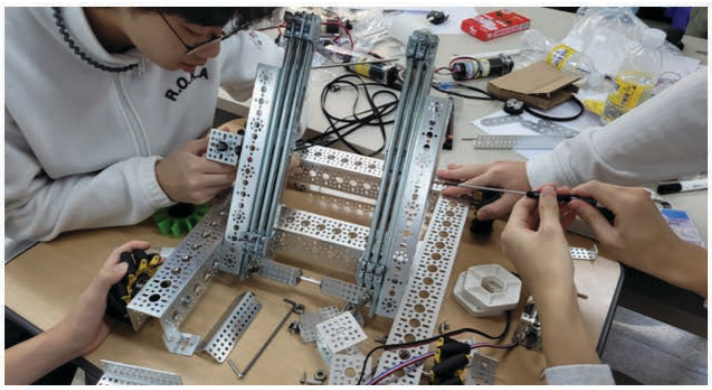
i) Add pulleys and modify their position

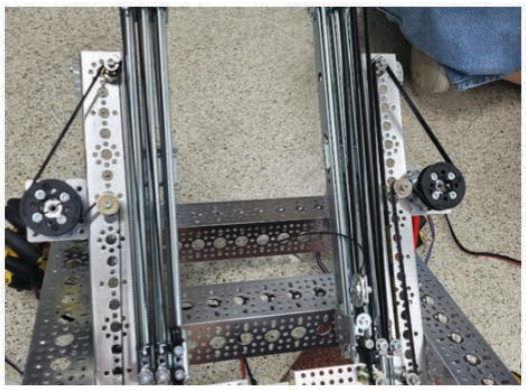
On the linear slide of the main body, pulleys that change the direction of the timing belt are attached to each step, but the position of the pulleys was not maximized when the G-shaped bracket was attached to prevent the linear from falling off. In addition to the pulleys attached to each end of the linear, we also needed an independent pulley attached to the body that would not move and would help the timing belt to wrap around the motor. To solve the lack of space, two D-shaped frames were used, each taking on the role of a G-shaped bracket. By using two D-shaped frames, the independent pulley installation was possible, and the problem that the G-bracket could not be raised to the maximum height was solved by relocating the pulley.

We also added another large pulley in the center of the frame where the linear slide is attached, which allows the timing belt to engage the motor more closely and receive the motor force better.

ii) Structural reinforcement

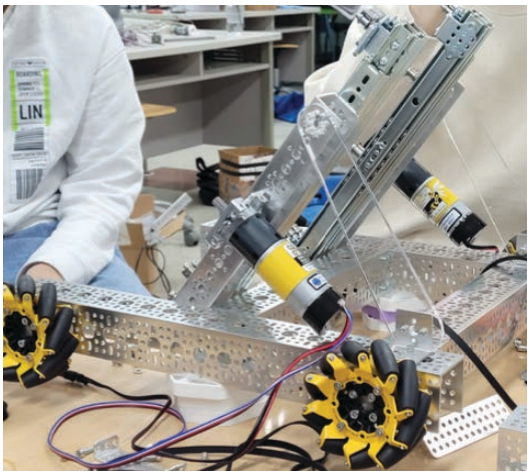
The use of two D-shaped frames not only solved the problem of the linear falling out, but also created space for independent pulleys to be installed so that the timing belt





could better engage the motor. However, it didn't solve the problem of the linear slide having a small area connected to the body, which can be structurally unstable. Therefore, we used acrylic and a laser cutter to create a structure that could connect the linear to the main body, and fixed it using an L-shaped bracket.

2) Mounting and testing the motor



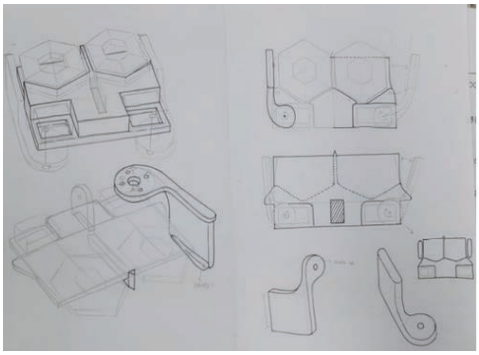
The location of the motor for the linear slide needs to be found and mounted in a suitable space where it can run parallel to the timing belt and pulleys without affecting the space for the control hub or battery. We determined that the best space was parallel to the linear slide, so we mounted the motor there.

2. Eater planning and team project progress

There were three main ideas for an iterator that would carry the pixels inside the robot and into the basket. Each method had its own pros and cons, so we decided to divide into

teams, give each team a set amount of time to create an iteration, and then put the most successful iteration on the robot.

1) Pincer Eater



We used two 38cm aluminum frames to make a frame of about 70cm, and we found that the servo motor rotated smoothly even with considerable torque. This shows that it is possible to simplify the tongs enough to make it run smoothly even with longer arms.

Since the arms were long enough, the design moved forward with the horizontal loading method, which is more efficient for mosaic creation, and the pixel molds were made on a 3D printer.

2) Rotor Eater - Using a conveyor belt

[Selection of belt type]

Timing belt :: Grooves are properly spaced and have good friction, but cannot be made into a loop of proper length because it is not in the form of a loop (X)

Rubber band (length: 12 cm, width: 0.5 cm) **::** Recognizes that the rubber band is appropriate in terms of friction and loop. However, the existing 12cm rubber band is too short and has too much tension.

=> Buy several 23 cm long and 0.5 cm wide rubber bands.

[Structural Design]

Motor selection :: Although the servo motor has enough power to be used as a chain, we chose a hexagonal DC motor (REV) to secure a strong power with the structure of the car body.

Axis setting :: I chose a hexagonal axle suitable for the above motor, and cut 2.3cm of the 40cm axle with a metal cutter to attach it to the car body.

Attach the structure :: Further resembling the structure to run the conveyor belt as close to the floor as possible. Attach a 3*5 plate with bearings to the first compartment of the back of the car body. We want to attach an axle to the bottom of the plate to attach a rubber band.

[Rubber Band Experiment]

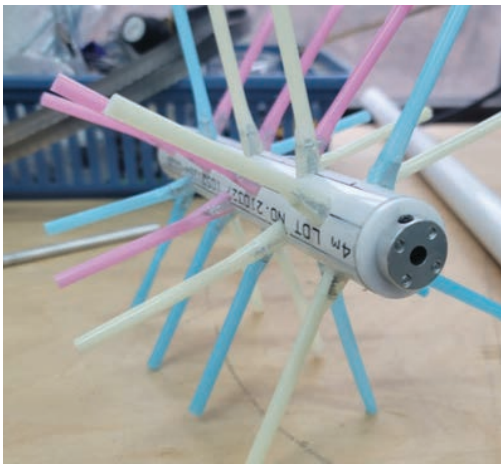
We connected a 12cm rubber band that we had on hand to the hex shaft on the motor and the opposite shaft on the structure, and experimented to see if the pixels would move upwards. As a result, it was confirmed that it is possible to transport pixels through the conveyor belt method.

3) Rotor Eater - without conveyor belt

We made an iterator by drilling several holes in a pipe, nailing it, and cutting off the end of a straw to connect the nail and the straw. We connected it to the motor and found that it sucked up the pixels well.

[Structural Design]

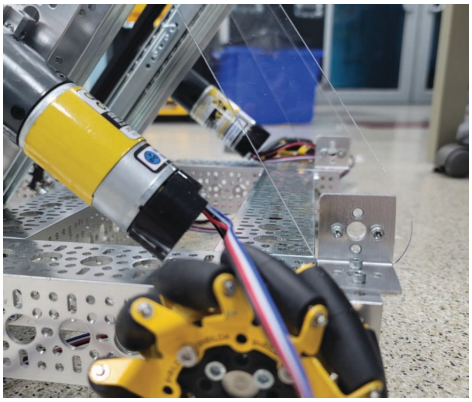
Motor selection :: Although the servo motor has enough power to be used as a chain, we chose a hexagonal DC motor (REV) to secure a strong power with the structure of the car body.



Since we don't use a conveyor belt, the pixels have to go up the ramp using only the force of the eater, so we focused on making the eater stronger.
We tested this by building our own eater, as shown in the photo on the left.

Jan 16th, Tue

1. Space at the back of the body



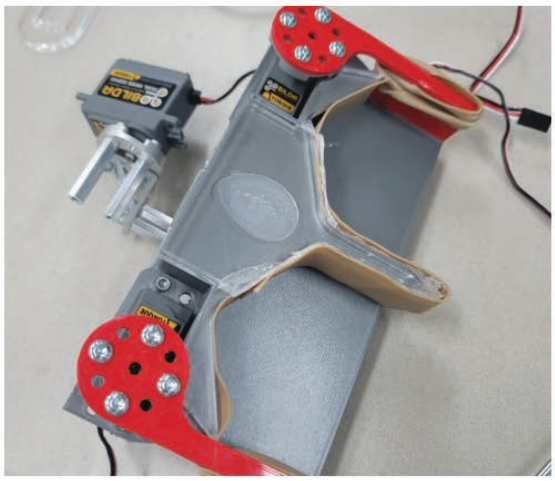
In order for both the pincer and rotor iterators to be located in the rear part of the robot, a lot of space was needed at the rear of the robot, but the design of the linear and the main frame itself made it unavoidably lacking. To solve this problem, the position of the D-shaped frame and the linear located in the center of the body was pulled forward by one unit of space to free up space at the back of the body.

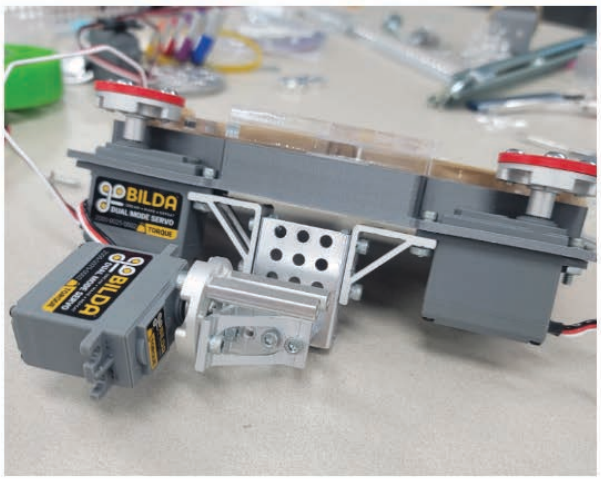
However, as the linear slide and the D-shaped frame were pulled forward by one space, the acrylic and L-shaped bracket attached to secure stability protruded from the body, so the acrylic had to be repositioned and rebuilt.

2. Eater team project progress

1) Pincer Eater Team

Using a 3D printer, we made a mold that can be loaded horizontally and used acrylic to stabilize the pixels. To drop the pixels one by one, we made pincers at both ends and connected them to the servo. Then, I fixed the servo with an angle bracket and a D-shaped frame, and connected the servo motor to the wrist.





However, the weight was heavier than I expected, so I plan to reduce the weight by adjusting the density of the 3D printed frame and drilling holes in the acrylic.

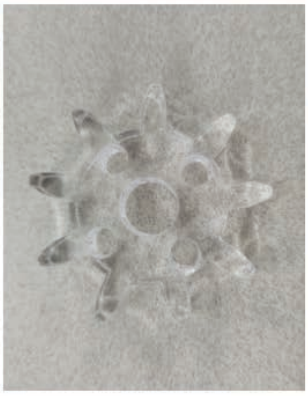
2) Rotor Eater - Conveyor Belt Team

[Conveyor Belt Design]

Using a single hexagonal DC motor, the team plans to install the motor on one part of the shaft and fix the shaft with bearings on the other part. Since the shaft that transmits power has to be located in a narrow space where the linear and the body form an acute angle, the team had to use a 3D printer to fix the position of the motor and bearing. To do this, we printed the motor and bearing mount and used a drill to drill holes to connect it to the body.

[Selecting an ether rotor material]

The following material was chosen for the rotor because it sits at the right height when the axis of the rotor is directly connected to the body. In addition, it is made of silicone, which is very elastic, which is a great advantage for the ether itself, and its diameter is not large, so it was expected


that it would minimally snag on the conveyor belt.

To create the rotor above, we wanted to connect the DC motor vertically to the body frame and use gears to convert the power through 90 degrees to turn the rotor. The number of gears was insufficient, so they were made of acrylic. For the connection of the shaft, the bearing that connects the body frame to the hexagonal shaft was lacking, so we used acrylic to make a hole the same diameter as the hexagonal shaft. The DC motor was planned to be connected using a mount, but the location of the hole was not convenient, so it was also made of acrylic to connect it.

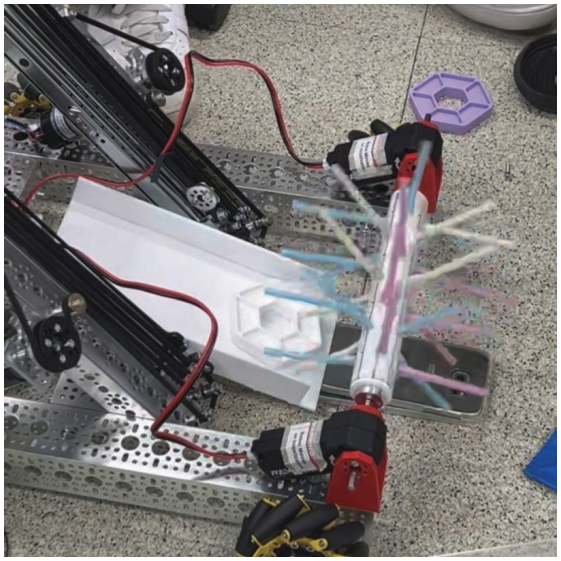
[Fixing the rubber band]

When we made the conveyor belt using rubber bands, the rubber bands often derailed from the linear axis, so to prevent this, we made acrylic plates to fix the rubber bands from both sides. If the plates protruded above the elastic, the pixels could slide off, so we made sure that the plates only protruded by the thickness of the elastic.

Also, to make the bearing fit the width of the newly purchased elastic (1cm wide and 23cm long), we made an acrylic ring the same size as the existing bearing and fixed it to the bearing. The finalized shaft using these materials is

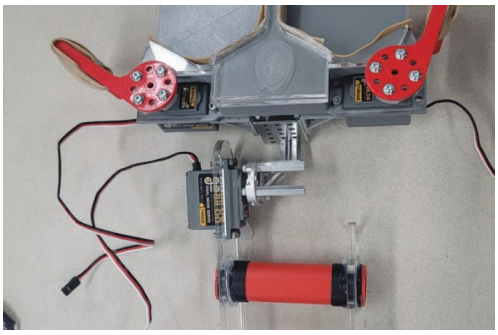
shown in the photo on the right above.

3) Rotor Eater - Converyor X Team



Pixels kept bouncing off the end of the ramp. To compensate for this, we attached a ruler to the end of the ramp to make it easier for the pixel to climb up. However, we found that the pixels would not move smoothly up the ramp unless the ramp was exactly flush with the ground, and we needed a way to adjust the height of the ramp.

Jan 17th, Wed



1. Eater Team Project

1) Pincer Eater Team

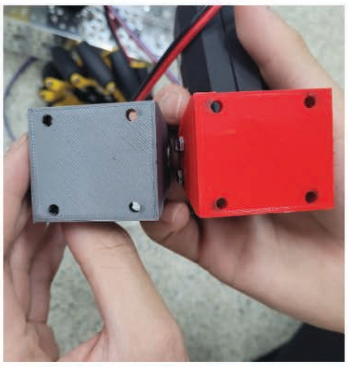
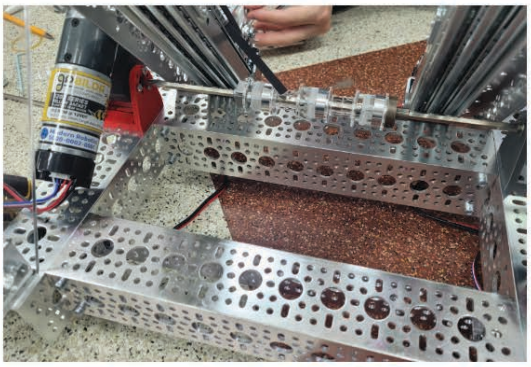
i) Adding / positioning pulleys

The arms were made by connecting two pieces of acrylic to two pipes for the tongs, and two servos were attached to each side to move the arms.

ii) Attach the pincer body

The pincer arms were connected to the endstop plate in the center of the linear slides with two screws. During the connection process, one of the screws broke inside the linear slide. We found it difficult to remove the screw, so we replaced it with another linear slide that was created by reducing the linear from four to three.

2) Rotor Eater - Conveyor Belt Usage Team

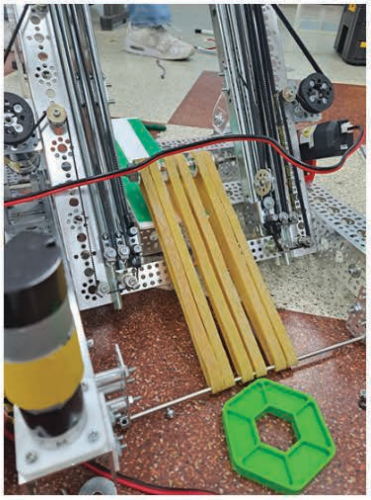
[Fixing the motor, making the base].

The motor that will turn the rotor was fixed using gears made of acrylic, a mount, and a plate for the hexagonal shaft.
I 3D printed a base to hold the hex DC motor that will turn the linear side of the axis that will turn the conveyor belt. (I designed it on 1/16) After making it, I drilled holes to secure it with screws.

[Basket design]

The basket I designed for use with the conveyor belt was to be fixed to a linear slider, so that the pixels would enter the basket horizontally from the conveyor belt, fall at an angle





along the inner surface of the basket, and then be fixed to the linear, which would open the bottom of the basket using a servomotor as it rose, allowing the pixels to fall onto the plate. We prototyped using a box to see which shape was most efficient, and found that the pixels fell off better when the surface was rounded, so we 3D printed the same shape.

[Making the rubber band conveyor]

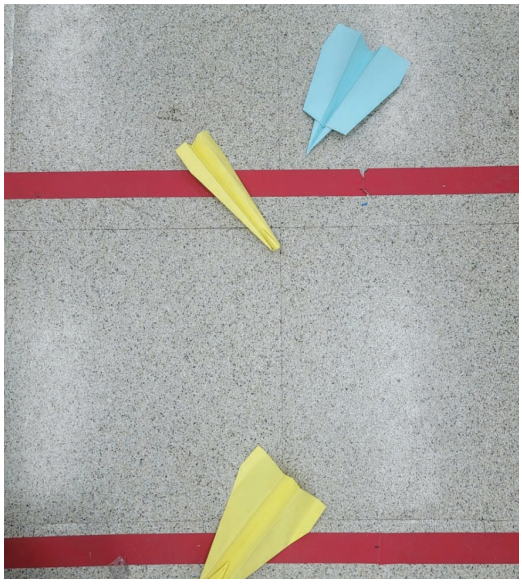
The new order of rubber bands (1cm wide and 23cm long) arrived and I tried them on the shafts I had made so far. Each bearing has two rubber bands, so I used a total of six rubber bands.

I had two options: buy new, shorter rubber bands, or adjust the position of the axis to increase the tension. Since repositioning the axis would affect the placement of the other structures, the elastic had to be purchased about 20 centimeters shorter. To see if the conveyor belt would work better with more tension, we first tried moving the axis for a while and tensioning the 23-centimeter elastic band, but the conveyor still didn’t run smoothly.

I decided that if I wanted to use the conveyor belt method, I would be better off buying an actual conveyor belt instead of utilizing a rubber band. However, since ordering a new shipment now would cause too much time to be lost while waiting for the shipment to arrive, we decided not to use the conveyor belt method.

Jan 18th, Thu

1. Drone Mission



1) Folding the Drone

In this paper airplane mission, the paper airplane must land directly in front of you outside the arena for the highest score, so it shouldn't fly far with a lot of lift. To find the right paper airplane, I built a few, and the ones that seemed to work best for high scores were the ones that flew up and then curved down and fell, and the ones that flew up and then came back to where they came from.

2) Plan for drone launch pad

We started planning our paper airplane launcher. First, he attached a spring to both ends of a D-shaped frame about 30 centimeters long. After that, he made a frame to insert the airplane, and then planned to fly the airplane through the elastic force of the spring. Then, the back of the frame is made into a N-shaped hook, and a G-shaped hook is attached to the servo part to pull the frame hanging on the spring, and then the hook hanging on the back of the frame is attached to the servo hook to load the airplane, and when the servo moves and the hook is released, the airplane is launched.

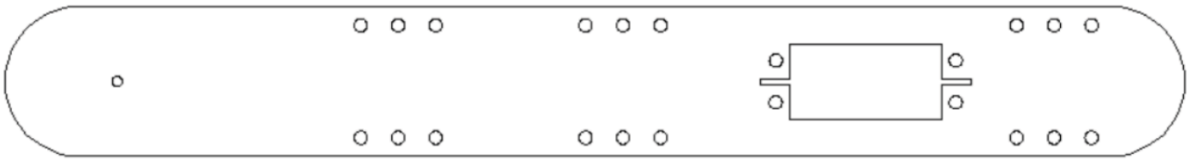
However, due to the limited length of the frame, the maxi-

mum length that the spring can be pulled is limited, so the elastic force may be insufficient.

2. Building the Stack Eater

In order to remove only two pixels from the pixel tower stacked in the arena, we started to build a “clicker”, which uses the elasticity of the spring steel to move the pincer at a high speed to remove only one layer of pixels from the pixel tower. It is similar to pulling out a tablecloth.

3. Modifying the pincer structure



The existing pincer structure was unstable because the pincer was attached to only one acrylic plate on the arm. We had to explore how to fix the pincers on both acrylic plates while using only one servo motor.

First, the existing L-shaped bracket connected to the servo motor was changed to a U-shaped bracket, and the servo horn was attached and fixed with bearings so that the non-motorized part could rotate well.

In addition, one side of the acrylic plate had to be pulled out again to put in the bolts for fixing the servo horn, and it was designed and manufactured as shown below. The connection between the acrylic plates was also changed from the existing cylindrical type to a stable connection with a truss structure.

4. Finalizing the Eater

[Rotor Eater - Without Conveyor Belt]

When using a rotor ether without a conveyor belt, the pixels are received through the ether and then loaded into the basket via a ramp, but there was a constant bouncing phenomenon at the front of the ramp, and we tried various materials such as rulers, acrylic, cable ties, etc. to eliminate this phenomenon, but we could not find a way to move the pixels smoothly onto the ramp.

We decided not to adopt the rotor eater method without using a conveyor belt because there are too many variables that can occur due to various steps, such as pixels not entering the rotor eater, not making it from the ramp to the basket, and not loading into the basket.

[Rotor Eater - With Conveyor Belt]

This method was also rejected due to the problems caused by rubber band tension and shaft bending.

[Pincer Eater]

On the other hand, the pincer iterator was more promising than the other team projects because it was able to grab and move pixels, and it was the most complete iterator among the current team projects, so we decided to adopt the pincer iterator.

[Re-move linear slide]

When using the pincer iterator, we thought that the center of gravity was more important than being close to the back-stage, so we moved the linear slide back one space and pulled it forward again to make room for the control hub and battery.

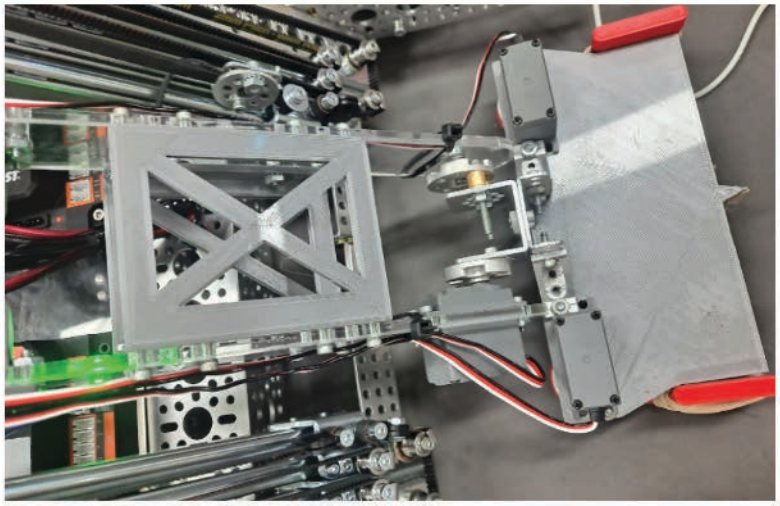
Jan 19th, Fri

1. Reassemble Pincer

We've reassembled the finalized Pincer Part to the body of our Robot.

2. Bottom Panel for Control Hub

For a full-fledged driving implementation, we wanted to attach the Control Hub and Expansion Hub to the body. Since we decided not to use an iterator, we didn't need the space for a box to load the pixels, so we decided to make an acrylic plate that would fit into the empty space in the front of the body and put the hubs on it. After measuring the dimensions, the acrylic plate was designed in fusion 360 as shown below and cut to a thickness of 0.5 cm.



Jan 20th, Sat

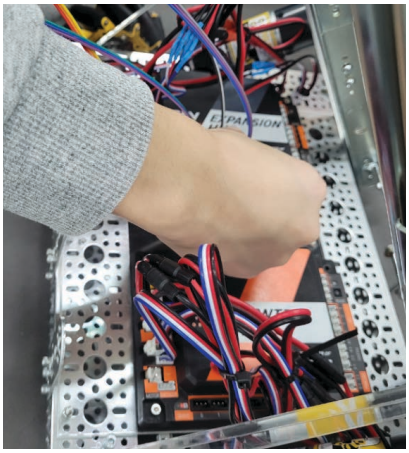
1. Body Maintenance

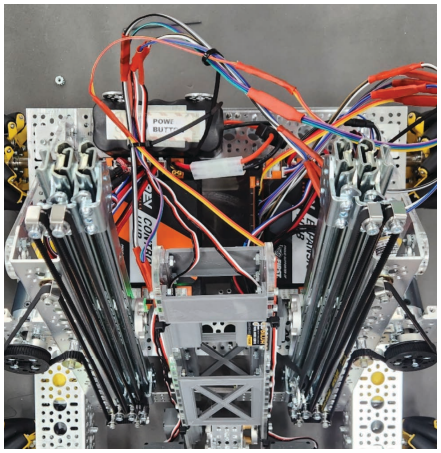
1) Modify the position of the hubs and place the motor

When we tried to organize the lines, we found that placing the hubs in the existing position (under the body frame) was inconvenient for organizing the lines, and the position of the motor was also inconvenient when placed between the hubs, which fit the space perfectly but made it difficult to connect the webcam. Therefore, in order to solve the above problems while recycling the existing acrylic plate, the plate was fixed on the body frame, and the motor was placed on the front frame of the car body and fixed with long bolts and small plates to prevent it from moving.

2) Organizing the lines

After fixing the unstable motor wires with tubes, we finished organizing the wires.

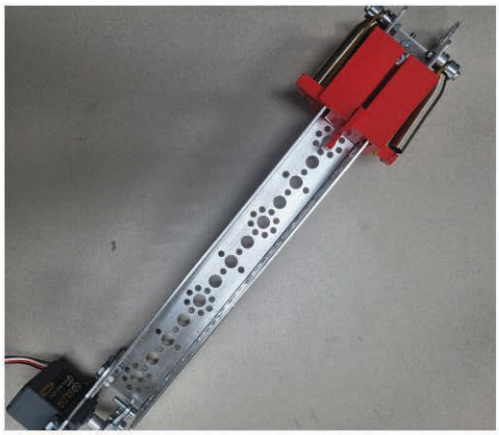


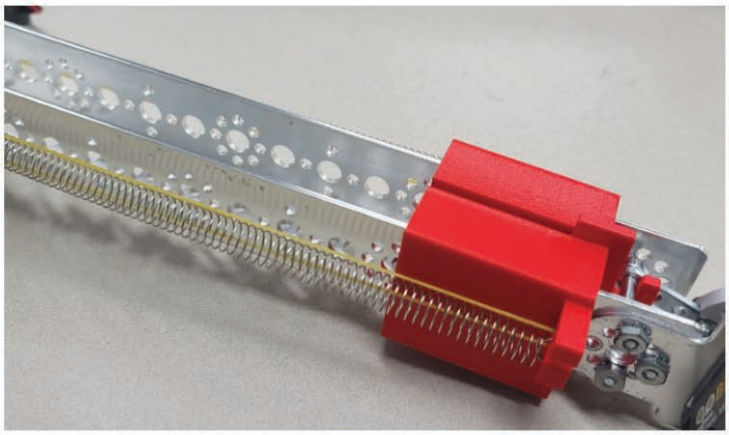


2. Build an airplane launcher

To launch the paper airplane, we used a 3D printer to create an airplane launcher. As planned, I attached a spring to both sides of the mold, but the tension wasn't strong enough to launch the airplane, so I added rubber bands to both sides.

I launched the airplane from the launcher and it flew with the desired force. I'll adjust the angle and attach it to the body. The hooks that connect to the servos are at risk of breaking, so I plan to make extra hooks in advance.





Jan 21st, Sun

1. 1st successful demonstration

You have successfully demonstrated driving the sequence of picking up a pixel with pincers from the starting position on the competition, moving to the front of the board, dropping the pixel at the appropriate location via linear, and returning to the board. In a second demonstration, we will try dropping the pixel from various positions, flying a paper airplane, passing under a hurdle, and hanging from a hurdle pole.

In the process of practicing, we realized that the moving “fingers” of the pincer were relatively low, which prevented the pixel from fitting snugly into the pincer.

Jan 22nd, Mon

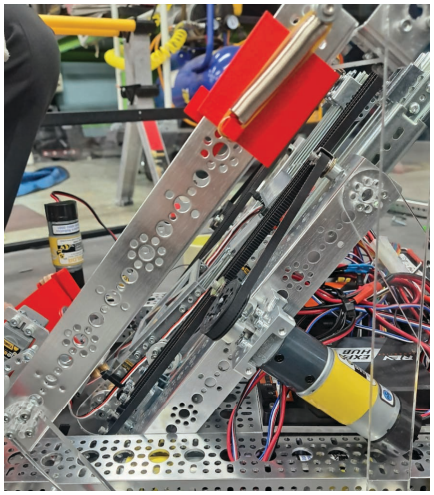
1. Attach Drone Launcher

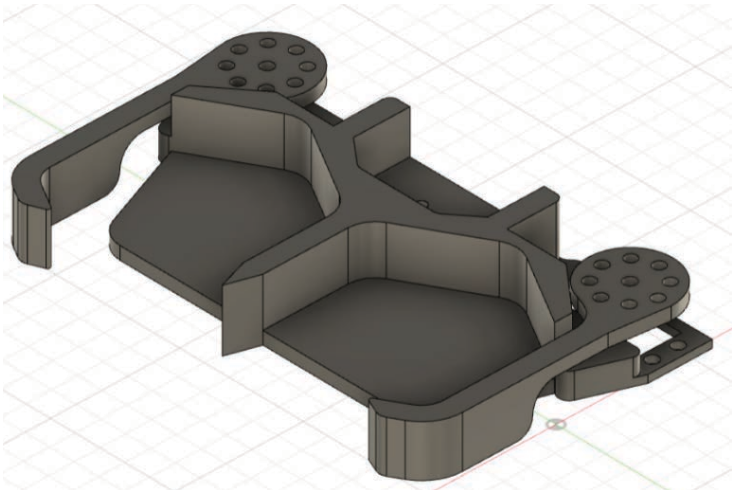
In order to attach the paper airplane base to the body, we needed to connect the body to the 3D printed compartment where the paper airplane is placed. The acrylic plate was designed considering the angle of the bolt holes, and two different lengths of acrylic plate were cut out and connected to the body. The team realized that the hooks connected to the servo motors could not reach the 3D printed compartments, and came up with two options: cutting off part of the acrylic plate design or relocating the motors. They chose to reposition the servo motors, and the paper airplane was able to fly successfully.

2. Change Pincer Design

1) Change the pincer plate

The pincer plate was getting in the way of the pincer finger as it gripped the pixel, preventing the finger from grasping





the pixel properly. Therefore, we beveled both ends of the pincer plate so that the pincer grabs the pixel directly.

2) Changing the Pincer finger

The pincer's finger was not gripping the pixels well, so we lengthened the finger and made a 120-degree bend at the end to make it easier to grip.

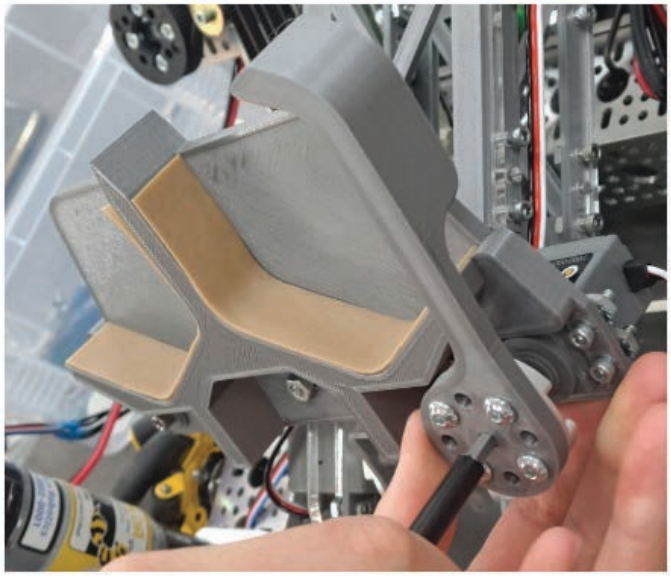
Jan 23rd, Tue

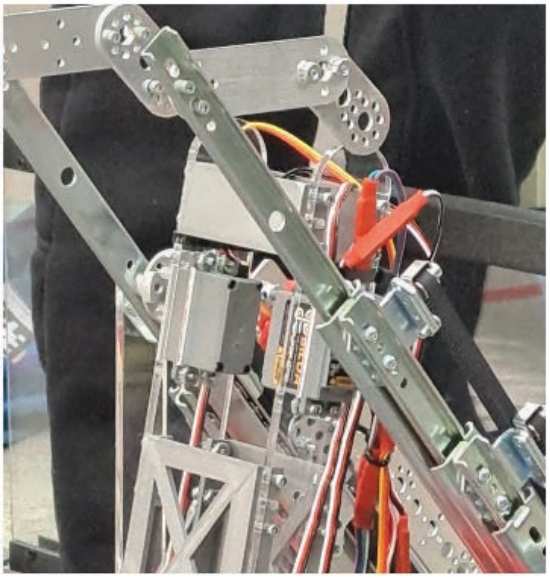
1. Reattaching the pincers

Yesterday, I finished 3D printing the new design of the pincer, so I changed the pincer. Initially, we used a servo horn with a long threaded connection between the fingers of the modified pincer and the motor, but the fingers were too low in relation to the pincer hand and did not grasp the pixels properly during testing. The builder in charge of the design confirmed that the original design considered using a servo horn with a shorter screw connection, and later replaced it with a shorter part to ensure that the hand and fingers were at the correct height.

2. Hook Attachment (Linear Slide)

We installed a hanging hook, a simple method of fixing two thin plates with screws, and we checked the completion and successful hanging of the robot as shown below.





#25309
TALOS

Programmer

KRC Preparation Programming Note

We’re the Champions from Korea.
Look at what we’ve been through.

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

1. Preparation

Our team decided to use JAVA to increase programming efficiency and create a more finished robot controller. We identified the On Bot JAVA development environment, which utilizes the Hardware Client to insert and build source code directly into the Control Hub of the robot, and checked the official manual and the methods and packages of our own SDK.

Pulled the SDK and project provided by FTC, and created a repository to store the robot source code for this competition to facilitate collaboration and future code inheritance.

2. Hardware Testing

During the preparation for last year's competition, the encoder of the DC motor did not work properly, making it difficult to move it to the correct position when writing the auto code. To compensate for this, we tested the encoder with a pre-made drive part. When no special force was applied, we found that the motors rotated at the same angular velocity when using the encoder, and the encoder values were accurate, so all the motors stopped at the exact same moment. In addition, while reading the SDK documentation, we found several functions that can be utilized more effectively with DC motors, and tested them.

1) IMU (Inertial Measurement Unit)

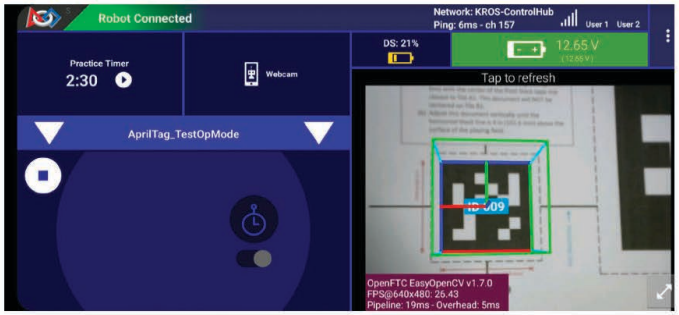
The IMU (Inertial Measurement Unit) consists of an acceleration sensor and an angular velocity sensor, and is built into the Control Hub and Expansion Hub. Last year,

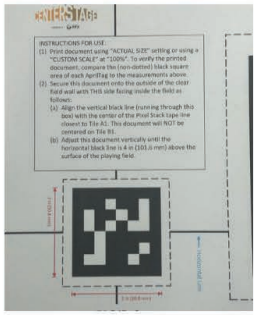
we didn't realize the existence of the IMU until after the fact and didn't utilize it properly in the actual competition. Therefore, this time, we tested the IMU early on and confirmed its accuracy. IMU sensors are prone to errors because they calculate the distance and angle moved by double integrating the acceleration, and we found that the errors accumulated during the actual test. The error rate will increase even more if it is mounted on a robot with shaking, and we have considered several compensation methods to solve this problem.

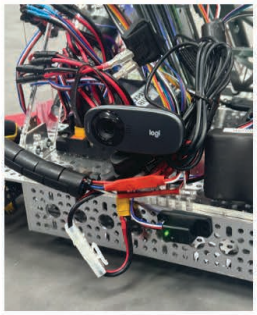
2) AprilTag

AprilTag is a recognition tag developed by the University of Michigan that is designed to recognize robots accurately and efficiently. The FTC utilizes AprilTag to identify the current location and orientation of the robot.

We wanted to measure the current position of the robot by recognizing the AprilTag attached to the backdrop of the stadium, so we tested whether the AprilTag was recognized correctly using a webcam. Since we need to recognize the position of the Team Prop in the Autonomous Period and then drop a pixel on the corresponding position in the Backdrop, we expected that recognizing the AprilTag attached to each position would improve the accuracy of the position measurement. We also expected to be able to calibrate the current position of the robot by calculating the position and angle of the tag.







Using the built-in library, we were able to retrieve the IDs and names of several types of tags, and we tested the web-cam on a real robot. However, we had difficulty calculating the current position and angle of the tag, and measuring the position of the backdrop was not significantly different from the results using the optical distance sensor, so we decided not to use tags.

3. Code Structure Planning

We planned the code structure to write the code efficiently, and to increase the maintainability and readability. We de-signed the object structure by subdividing the program into Hardware, Part, and OpMode according to their roles.

Hardware directly utilizes the FTC SDK to handle hard-ware input and output. Part is an object that divides the robot into parts according to function and manages the movement of the parts. A Part has Hardware as a member variable and is responsible for issuing commands to and managing the Hardware corresponding to its Part. Finally, OpMode is a child class of OpMode provided by the FTC SDK that issues commands to Parts based on user input or automated procedures. The Part receives commands from the OpMode, and the OpMode periodically updates the Part to fulfill the commands.

KROS 2024

Robot Code Class Diagram

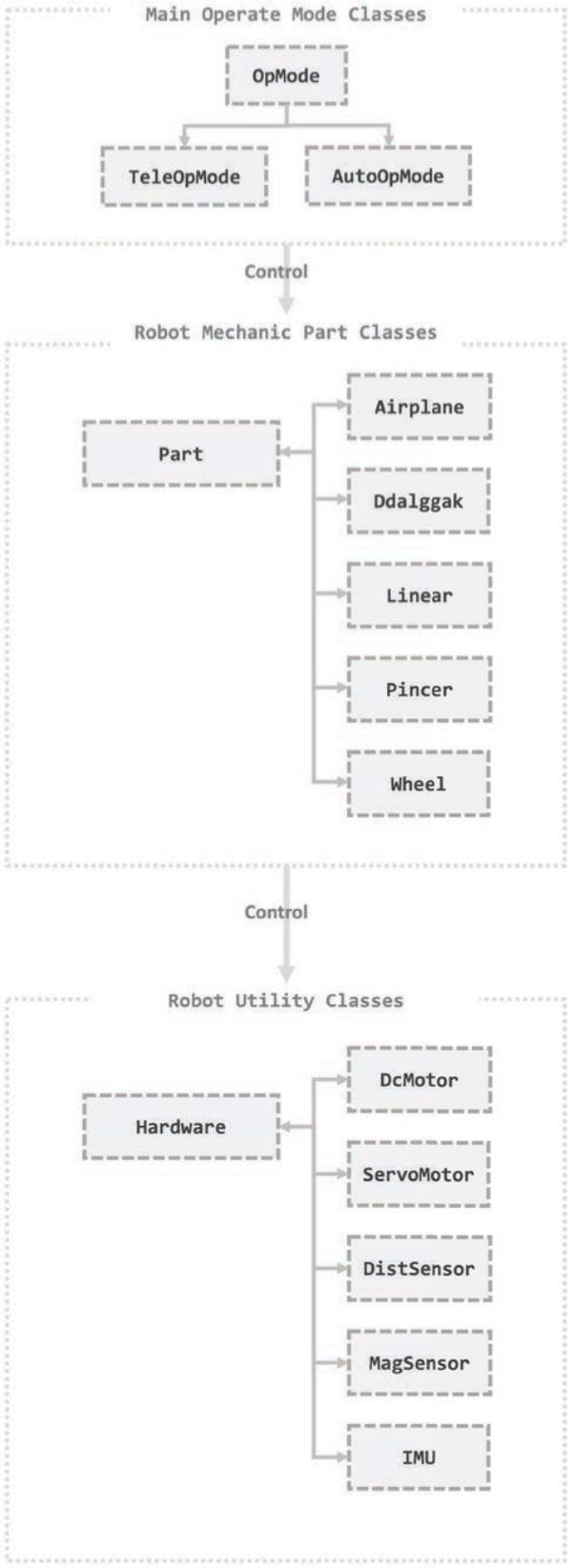
Programming Team : 유태우 & 김준성

OpMode : Operating Mode Class

Part : Robot Mechanic Part Class

Hardware : HW Managing Class

절차에 따라 상위 클래스가
하위 클래스에 동작 명령을 전달



Github
https://github.com/KSA-KROS/FTC2023-2024_RobotSourceCode

This allows OpMode to issue abstracted commands to perform movements without directly touching the hardware, making the code more readable and reducing complexity. It also allows for multiple actions to be performed simultaneously by scheduling specific actions and updating them to execute them, rather than performing them immediately upon receiving a command.

4. Developing Hardware Objects

1) Overview of the hardware abstract class

Hardware is a set of objects that directly handle the input and output of hardware through the FTC SDK. These objects all inherit from an abstract class named Hardware.

By inheriting from a common parent class called Hardware, we enable polymorphism, a key feature of object-oriented programming, and prevent programmer mistakes based on object-oriented logic by declaring abstract functions to override functions that must be in the Hardware class, such as update() and isFinished().

2) DcMotorHW

DcMotorHW is a class that controls a DC motor. It makes it easy to change the setting values of the DC motor through functions such as setDirection, setUsingEncoder, setUsingBrake, and setUsingFixation, and allows these functions to return themselves, so that the setting functions can be used in succession. This improves programming convenience and readability.

We also overloaded the move function, which controls the motor, to keep it moving at a certain power without a stop signal, and a function that allows you to enter an encoder tick value and make it move until it reaches a certain tick value.

We also created a function that measures the change in the encoder value and makes the motor move until it stops moving, so that we can physically control the range of the DC motor accurately, while not overworking the motor. Another key aspect of this class is the implementation of a locking function for the DC motor. In general, DC motors do not have the ability to be locked to a specific position, so external forces can cause them to change position. To prevent this, we devised a fixation algorithm and implemented a fixation function. If the target tick value deviates from the target tick value by more than a certain range, the algorithm rotates in reverse with a power proportional to the deviation to reach the target tick value. To be able to withstand strong external forces, the reverse rotation is performed by merging the current motor power value with the calculated power value.

3) ServoHW

ServoHW is a class that controls servo motors. Servo motors have the advantage that once you set a target angle, they move to that position and lock in place. However, a servo motor cannot set its speed, it only knows the target value, but has no way of knowing if it has actually reached it. Therefore, we implemented a function called moveWithInterval to control the speed and check the actual position. After entering the time and final target angle for the motor

to run, the target angle is changed every time ServoHW is updated, so that the target angle gradually reaches the final target angle over a certain period of time. This feature is useful when the servo motor is lifting heavy objects, and has the advantage of making the rotation of heavy objects more precise and more likely to succeed.

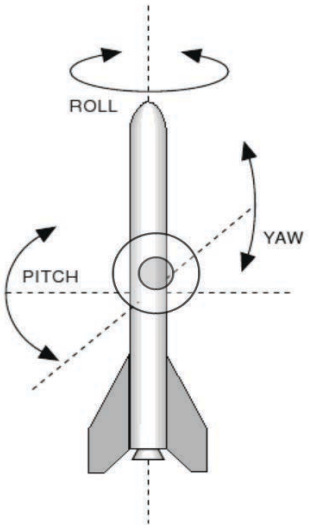
4) DistSensorHW

The DistSensorHW controls an optical distance sensor. The sensor is a REV Robotics 2m Distance Sensor, which can detect distances up to 2 meters. The optical sensors are mounted on the front and right side of the body and are responsible for detecting Team Prop and Background using the following functions. The isObjectDetected function returns true/false if an object is detected within 10cm or a given distance.

5) MagSensorHW

MagSensorHW controls the magnetic field sensor. It uses a Magnetic Limit Switch, which is switched on when a strong enough magnetic field is detected. However, the input value does not depend on the magnetic field strength, but only on the presence of a strong enough magnetic field. Since the encoder value of the DC motor cannot be used to determine exactly how much it has rotated, a magnetic sensor was used on the linear slide to determine if the motor has rotated enough. The isActivated function detects whether the magnetic switch sensor is triggered and returns true or false.

6) IMUHW



The IMU has a built-in function called getAngularVelocity that can retrieve the angular velocity in the Yaw, Pitch, and Roll directions as zRotationRate, xRotationRate, and yRotationRate. Non-zero angular velocities were measured even when stopped, and to solve this problem, we defined the getVelocity function to correct the measured angular velocity to zero if it is greater than -2.0 and less than 2.0.

We also defined a getAngle function so that we can measure the angle we are currently facing using the new getVelocity function we defined above to compensate for the accumulated error. In OpMode, we measure the time (Δt, ms) it takes for the code to execute each time it goes through one loop, and set

(current angle) = (current angle) + (angular velocity) x Δt

and set it to ±360 if the current angle was outside the range [-180, 180).

In our tests, we found that the actual angle deviated from the measured value only to the first decimal place, and when the robot moved very slowly, values below 2.0 were replaced by 0, which increased the error. Since the robot is moving/rotating at a high speed, the actual error was found to be very small.

Initially, only the IMU of the Control Hub was used for the calculation, but later, the IMUs of the Expansion Hub were imported into imu_sub and each was weighted by 0.5 to calculate the average value. As a result, the orientation was more accurately measured in AutoOpMode and the mission was stable.

5. Developing Part Objects

1) Overview of the Part abstract class

Part is a set of classes that control hardware and execute commands received from OpMode. It is responsible for dividing the robot into functions and controlling each of them. These classes all inherit from an abstract class named Part.

The reason for having Part inherit from an abstract class is slightly different from Hardware above. The Part abstract class is focused on encapsulation and abstraction, whereas Hardware is for polymorphism.

First, we need to understand how Part works. A Part receives instructions from OpMode through a function called startStep. startStep calls an abstract function called nextStep, and each Part can override this nextStep to tell it what to do step by step for each instruction. The Part has a variable called step, which starts at 1 and represents the step in the execution of a particular command. In nextStep, you assign commands to Hardware objects, and when each Hardware has been assigned a command and has finished executing it, step is automatically incremented by 1, and the nextStep function is called to sequentially exe-

cute the next step of the specific command. The functions for updating the hardware, checking for completion, and automatically incrementing step are predefined in the Part abstract class, so each Part defines its own constructor to get the hardware for the Part, defines the nextStep function, and writes the step-by-step command execution procedure in a switch-case statement. This object abstraction maximizes the efficiency of writing code, increases readability, and makes it easier to maintain by defining the same functionality in a parent class.

The commands that Part is assigned are defined as an enum type named Command, and we utilized JAVA's Interface to give enum types object-oriented polymorphism, so that command processing functions can be defined in the Part abstract class, which is the parent class.

2) Hardware Manager

Parts manage hardware as a group. However, since we don't know what hardware each Part uses, and it's all different, it would be unnecessarily trivial to create update and shutdown checks for each Part in order to update all of the Part's hardware, or to check if it's finished. This is bug-prone and inefficient, so we created the HardwareManager class to solve this problem.

The HardwareManager class is based on the principle that Hardware classes are polymorphic because they inherit from the Hardware abstract class. Because the ability to check for updates or termination is a common feature of all Hardware classes and is defined in the Hardware abstract class, different types of objects can be managed by using

polymorphism to store them all in a list of objects named Hardware. The HardwareManager receives all the hardware classes for a particular part through the registerHardware function and stores them in a list. By utilizing polymorphism, it has the advantage of being able to easily update all hardware classes in bulk through a loop and check for termination.

This HardwareManager class is an object of the Part abstract class, and all parts have this object, which makes it easy to perform batch processing on the hardware.

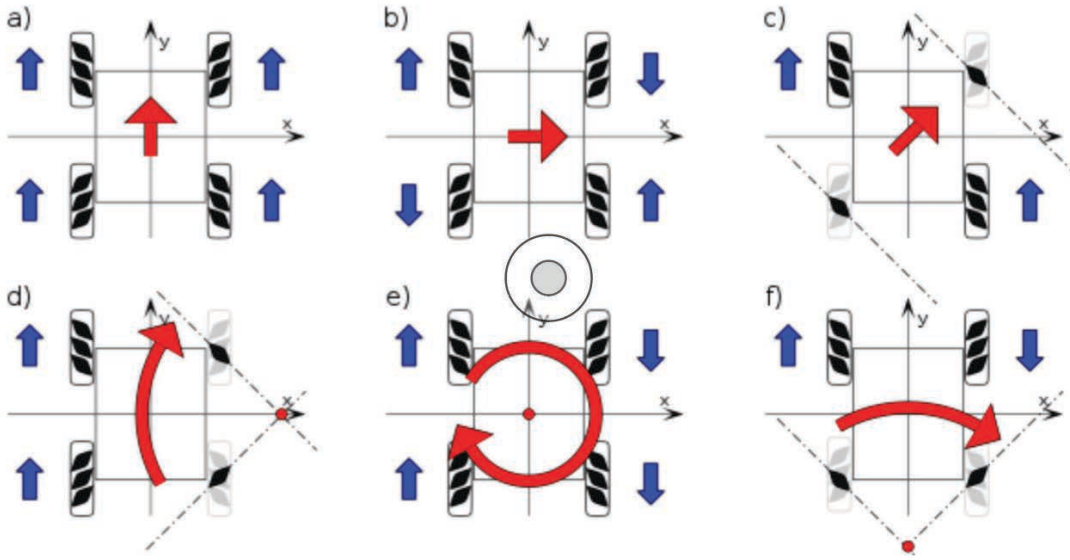
3) WheelPart

This class manages the driving part of the robot. It controls the four DC motors for the wheels and the two distance sensors used to detect team props when autonomous.

Command	Function
MOVE_FORWARD MOVE_BACKWARD MOVE_RIGHT MOVE_LEFT	Continuous movement (forward, backward, sideways)
VIEW_FORWARD VIEW_BACKWARD VIEW_RIGHT VIEW_LEFT	View certain angle, relative to starting position. (Utilizes IMU)
TURN_RIGHT TURN_LEFT	Continuous rotation (left, right)
STOP	Stop
AUTO_MOVE	Move along the direction with length (both static variables) (Reserved for AutoOp)

Also, the most powerful feature of this Wheel part is its omnidirectional drive. By varying the rotation direction and rotation speed of the four wheels, the Mechanum Wheel can be driven not only forward and backward, but also in other directions. In previous competitions, only the front, back, left, and right feet could be driven in four directions, but this time, the wheel can be driven in all directions by varying the angular velocity of each wheel according to the angle through mathematical calculations.

By utilizing trigonometric functions, the direction of movement can be obtained from the direction of movement. The magnitude of the vector is the traveling speed. The wheels on the top left and bottom right are proportional to the magnitude of the vector projected onto the x-axis after rotating the travel direction vector by -45 degrees, and the wheels on the top right and bottom left are proportional to the magnitude of the vector projected onto the x-axis after rotating the travel direction vector by +45 degrees. This allows for omni-directional drive.



4) LinearPart

This class manages the robot's linear. It controls two DC motors to drive the linear and a magnetic switch to limit the lower height limit of the linear.

Command	Function
MOVE_UP MOVE_DOWN MOVE_DOWN_POWERFUL	Moves linear slide up and down, with powers 0.6, 0.4, 0.8 (POWERFUL → Rigging is possible)
MOVE_DROP_POSITION MOVE_PSEUDO_DROP_POSITION MOVE_ORIGINAL_POSITION	Move linear slide to preset position.
STOP	Stop linear movement.

The main feature of a linear slide is the fixed value synchronization of the two motors. First of all, the linear slide has to overcome the strong external force of gravity. Therefore, the DC motors that drive the linear slide have a locking function enabled. However, the fixed positions of the two motors are rarely precisely matched, and it has been observed that the two motors wander and move incorrectly. To solve this problem, a synchronization function was added to the motors.

Only the synchronized motor determines the fixed position and manages the up and down movement. The synchronized motor does not determine the fixed position, but moves in synchronization with the movement of the synchronized motor. This prevents the two motors from moving apart and allows them to successfully fix the position.

4) PincerPart

This class manages the pincer part of the robot. The pincer consists of an Arm, Wrist, and Finger. Each is driven by a servo motor, and each servo has a variable GrabPosition (the position when picking up a pixel) and DropPosition (the position when dropping a pixel).

Command	Function
OPEN_OR_CLOSE_DDALGGAK	Recognizes the current position of the iterator and opens or closes it. It has a power of 0.8 to make it move strongly. and is responsible for fetching pixels from the stack.
OPEN_OR_CLOSE_DDALGGAK_GENTLY	Recognizes the current iterator position and opens or closes the iterator. Weakly animated with a power of 0.2, responsible for accurately grabbing pixels released by the human player into the pincer.
RESET_DDALGGAK	Opened the iterator as far as possible to reset the state.
CLOSE_PERFECTLY	Close the iterator as much as possible.

We found that the servos on the arm were not getting enough torque to rotate, so we set the run time to a constant to solve the stalling problem. We ran the TeleOp for 3 seconds for both raising and lowering, and the AutoOp for 7 seconds and 5 seconds, respectively, and found that it worked fine.

5) Ddalggak Part

This class manages the robot's stack eater. The stack eater is powered by two DC motors and is responsible for pushing the pixels with the pincers. While developing the stack eater, our team informally referred to it as the "click part" and didn't decide on a formal name until the programming was complete, so we coded it as the Ddalggak Part.

Command	Function
OPEN_OR_CLOSE_DDALGGAK	Recognizes the current position of the iterator and opens or closes it. It has a power of 0.8 to make it move strongly. and is responsible for fetching pixels from the stack.
OPEN_OR_CLOSE_DDALGGAK_GENTLY	Recognizes the current iterator position and opens or closes the iterator. Weakly animated with a power of 0.2, responsible for accurately grabbing pixels released by the human player into the pincer.
RESET_DDALGGAK	Opened the iterator as far as possible to reset the state.
CLOSE_PERFECTLY	Close the iterator as much as possible.

Initially, we tried to limit and reset the driving range of the ether using a magnetic limit switch, but it was difficult to utilize the sensor-based driving range limitation because it failed to recognize in some cases. Therefore, we solved this problem by limiting the angle that can be physically rotated and detecting the situation when the motor physically stops and stops.

6) Airplane Part

The Airplane Part is fixed to the paper airplane holder and is driven by a servo motor. The servo motor, registered with the name airplane, rotates to release the holder with the rubber band and water iron, which launches the paper airplane.

Command	Function
FLY	Blasts paper airplane.

6. OpMode Development

1) Overview of how OpMode works

OpMode is the main class that operates the robot, and it is responsible for overseeing the operation of the robot. OpMode mainly defines Part objects, assigns commands to them, and updates them to execute commands. In some cases, OpMode also controls hardware directly from OpMode.

TeleOpMode is the class that receives commands from the driver via controller inputs and assigns commands to the Part accordingly. As such, this class primarily utilizes conditional statements to check for controller inputs and perform certain actions based on them.

AutoOpMode is a class that assigns commands to Parts and performs automatic missions according to a preplanned operation procedure. This class is similar to Part in that the program works based on Commands and Steps. It executes the Commands sequentially as specified, and executes the command through the startStep function after each Command is executed. When the part finishes executing the command, it increments the step variable by 1 and executes the commands in the Command sequentially.

2) TeleOpMode

TeleOpMode allows two drivers to control the robot using a controller (gamepad) during the Driver Controlled Period. The drivers are in charge of the robot's movement and

pixel-related driving, respectively, and use joysticks and buttons to control the robot. The development process of TeleOpMode and how to use the controller are described below.

i) Development process

The most important part of the development process was the “where is the front of the robot” problem mentioned above. Since the robot must be controlled using the joystick and directional keys on the gamepad, the control method changed depending on where the front of the robot was determined. Previously, the “front” was defined on the robot body and the controllers were mapped to it, but at the request of the driver, the “front” was determined based on the direction the driver was facing. Therefore, we created TeleOpRight and TeleOpLeft separately so that the steering direction changes depending on the direction the driver is looking.

To move the robot, we used the arrow keys to move forward, backward, right, and left, but later added a joystick for more intuitive control. The speed of the robot changes depending on how hard you push the joystick, and by allowing it to move diagonally, we were able to shorten its travel speed.

During the development of TeleOp, we had many discussions with drivers about the buttons on the gamepad. We tested different control schemes to make them more comfortable as they practiced on the built arena, and we believe this helped them to consistently improve their scores.

ii) Using the controller

We created a guide for using the controller, as shown in the photo below.

3) AutoOpMode

AutoOpMode was written to allow the robot to operate autonomously during the 30-second Autonomous Period. It recognizes the Team Prop as a distance sensor and drops a purple pixel in the right place, and it also recognizes the position of the Backdrop as a distance sensor and drops a yellow pixel, then folds the tongs and parks.

In our practice matches, we were able to reliably score 45 points with AutoOpMode, but we found that you may need to change your strategy depending on where you start. We aimed to score 45 points by placing and parking the yellow pixel when starting close to the background, and only placing the purple pixel when starting farther away.

The most important parts of AutoOpMode are the distance sensor and IMU.

i) Distance Sensor

AutoOpMode relies on the input from the distance sensors. Initially, we used only the front-facing distance sensor to arrive at a position in the Team Prop, then looked to the front, left, and right in that order to see if the distance sensor recognized the object. However, this was time inefficient, so we added an additional sensor on the right side so that we could determine the position of the Prop without rotating. (If neither the front nor the right side detects an object, the Prop is on the left side.)

The distance sensor can also be used to determine the position of the Backdrop. After dropping the purple pixel,

move left until the Backdrop is out of sight (relative to the Blue Team). Move right until the distance sensor detects the Backdrop, at which point it will move to where the yellow pixel will be placed based on the Prop's position.

ii) IMU

The IMU helps the robot to move in the correct direction during AutoOpMode. In TeleOpMode, if the robot is not traveling correctly, the driver can correct the direction by rotating the robot. For Auto, however, the human driver cannot intervene and must ensure that the robot is traveling in the correct direction.

We utilized the IMUs in the Control Hub and Expansion Hub to determine the angle the robot was looking at. After some trial and error, we were able to compensate for the error by calculating the average value using the two IMUs.

#25309
TALOS

KRC Preparation

Conclusion & Plans

We’re the Champions from Korea.
Look at what we’ve been through.

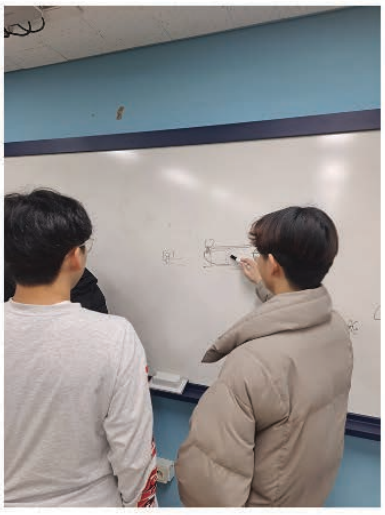
Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

Conclusion

We stayed at school from January 12th to 26th to finally finish writing the robot “Wuyi” and code, and we were able to fully test the Autonomous Period and Driver control Period. The team members who played the driver and human players also had plenty of time to practice and prepare for the competition.

We are very proud of the fact that we built a robot that was almost perfect for our initial plan and strategy, and completed the programming in a short time with many revisions and rechecks. I would like to thank all the team members of TALOS who participated in the whole process of designing and building the robot and planning the strategy, as well as our coach Hosook Kim, who provided a lot of help and guidance.

Although we achieved a high score in the trial match in the stadium we built, I think it is most important to make sure



that there are no errors or mistakes in the actual competition. Even now that the robot has been built, we are continuing to test the robot's control and autonomous driving, and we hope to achieve good results in the actual competition by working hard until the end.

For this engineering notebook, each team member wrote a description of the part they were responsible for or built, and one person was responsible for overseeing it each day. Changes made to the robot, problems encountered, and discussions with teammates were all recorded in the EN so that we could have a record to refer to in future competitions. We compiled them in a shared document on Google Docs and edited them using Adobe InDesign once they were complete.

My only regret is that I was not able to realize all of my ideas due to the short time I had left, but I was able to learn more about the technology and programming of robot building, as well as the overall process of robot building and project implementation in preparation for the FTC 2023-2024 season. As KROS has participated in the FTC several times, I think it was a great opportunity for us to build on the experiences of our seniors and learn from their mistakes to improve as a team.

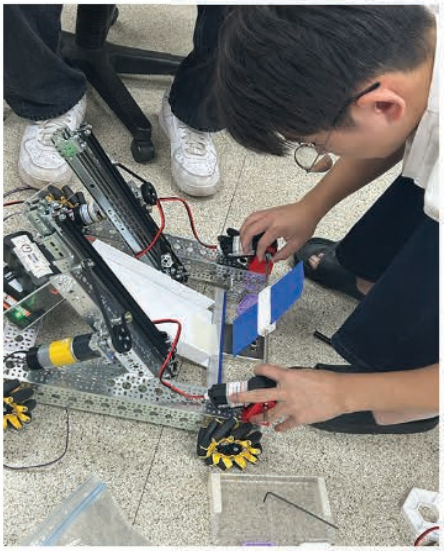
After winniing the Inspire Award, we were given a chance to participate in the FIRST Tech Challenge World Championship, which we cover in the next part of our Engineering Note.

#25309
TALOS







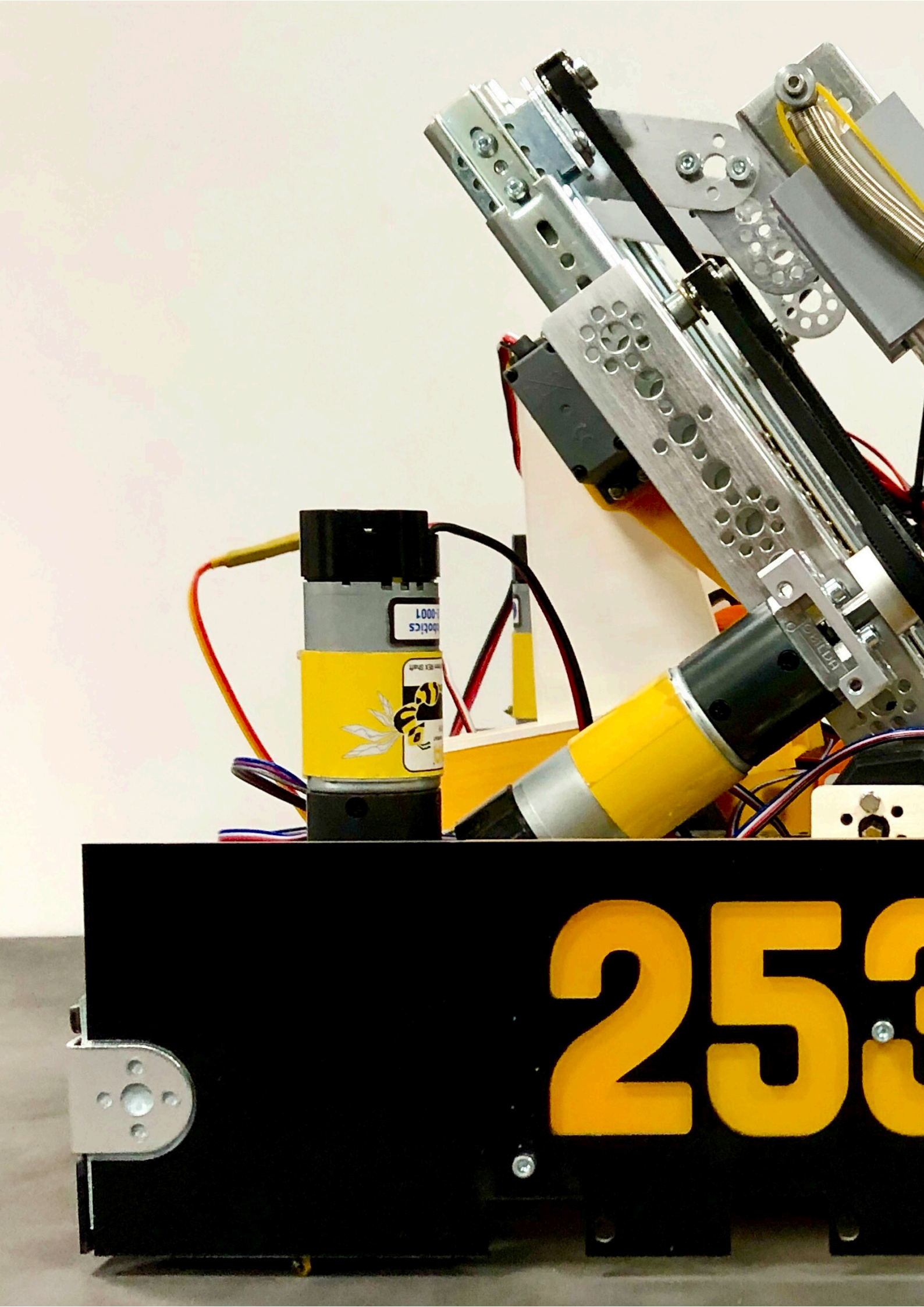



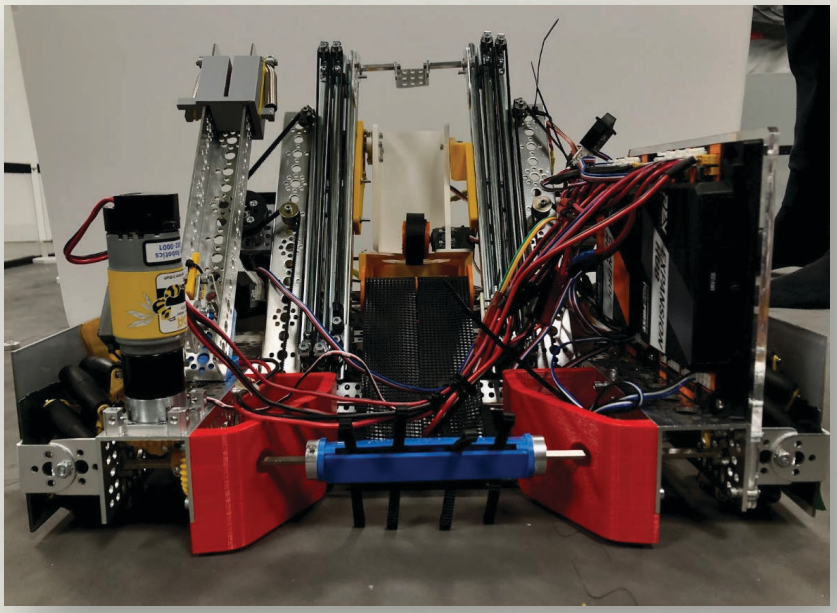
Builder

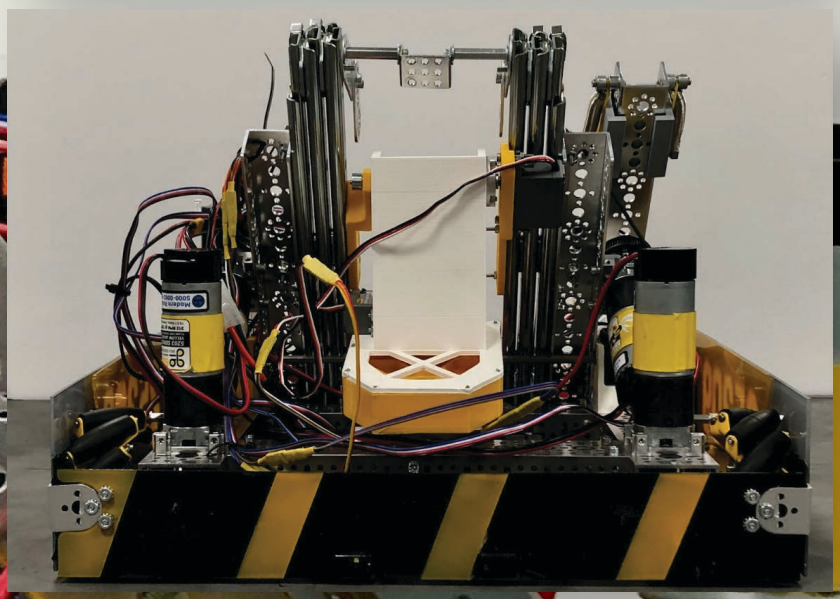
FIRST Tech Challenge Engineering Note

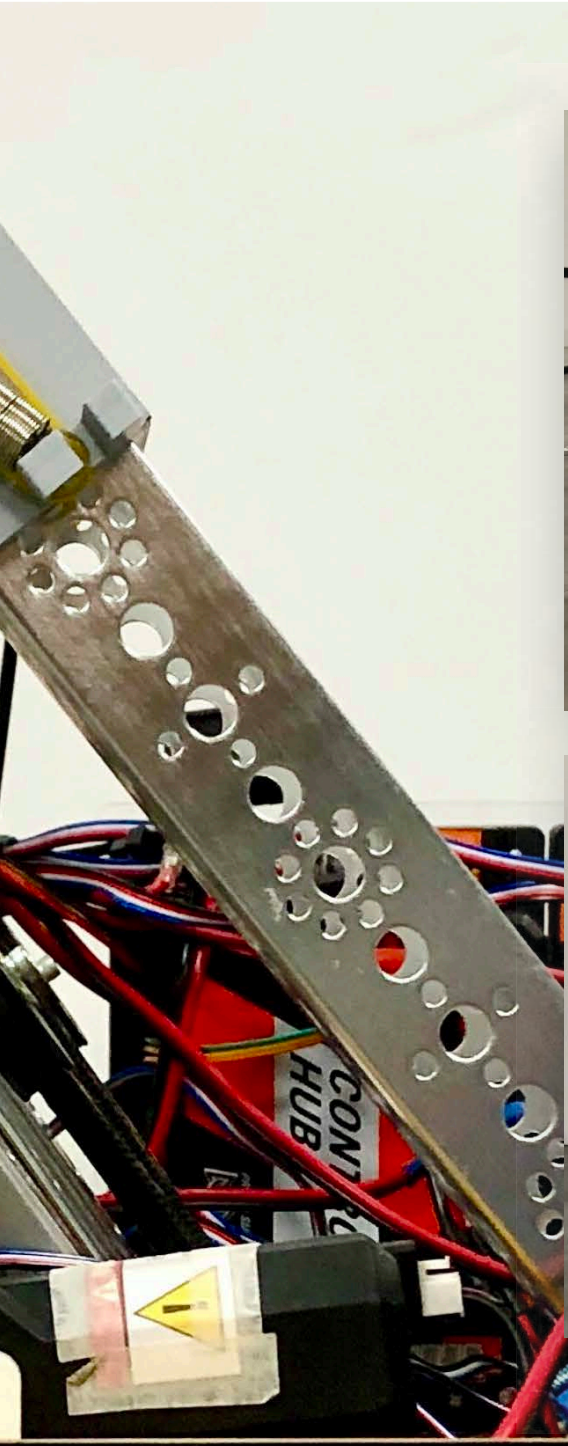
We're the Champions from Korea.
Take a look at what we've been through.

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST


253



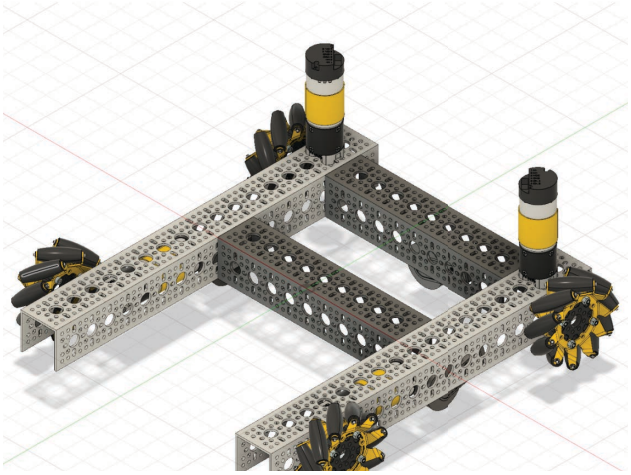



CONTR
HUB


309

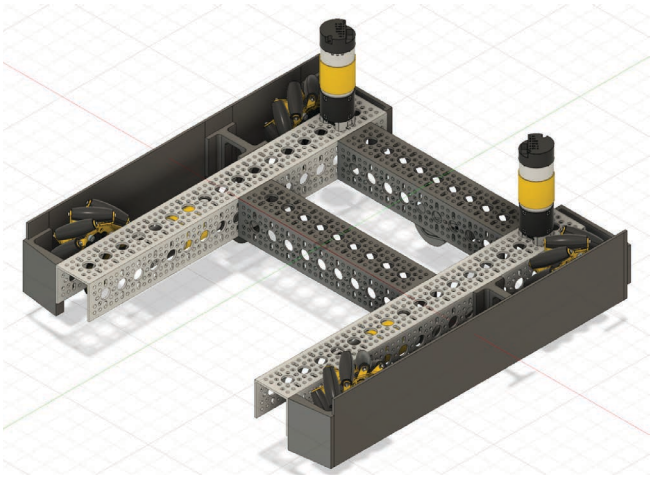
3D Modeling

1. Vehicle body modeling



When participating in KRC, we used the same chassis frame as before, but we made changes to the mechanism that captures pixels and places them on the backdrop. Considering this change, we moved the positions of the two “⊏” beams that support both frames forward and changed them to a shorter type. As a result, the width of the robot decreased slightly, allowing space to be created for the conveyor part and rotor part at the back. Instead of using the previous method of calculating the robot’s position using gyro and accelerometer sensors embedded in the Imu, we placed three odometries inside the chassis to more accurately track the robot’s movement during the autonomous period. Additionally, to secure space for odometry, we vertically placed and reconnected the two DC motors of the front wheels.

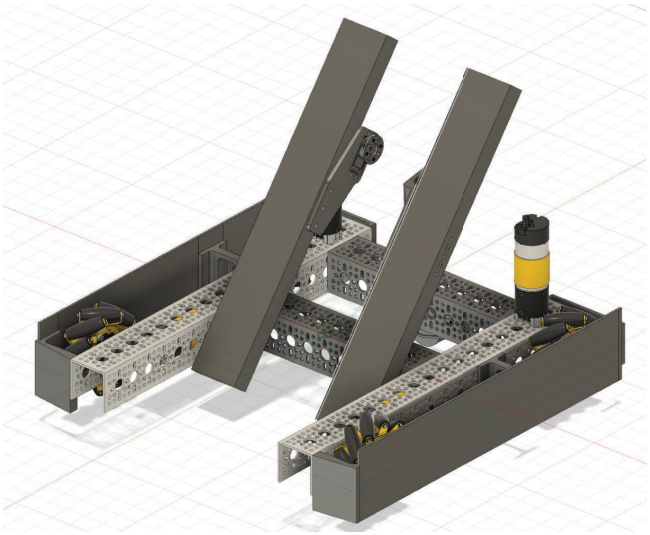
2. Guard Manufacturing

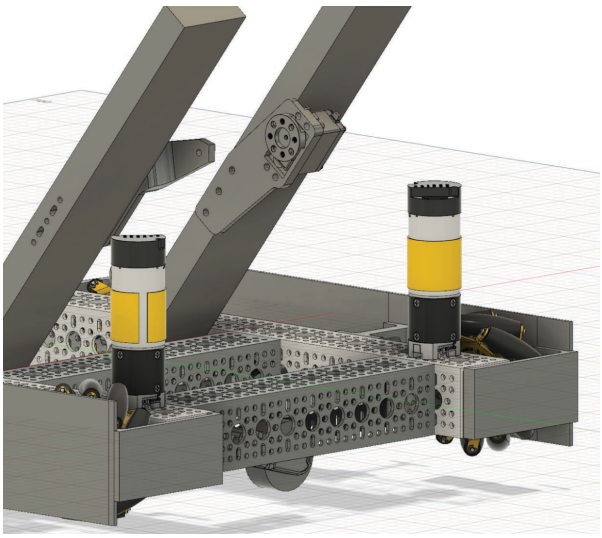


While participating in the KRC competition, we noticed a problem of pixels entering under the chassis during the game. In order to address this issue, we have designed a guard as shown in the picture below. The main pur-

pose of the guard is to prevent pixels from entering under the chassis by surrounding the sides of the robot and the front and back of the four wheels. The longer parts of the guard are made using acrylic or MDF, and the parts connecting the side panels and the chassis are all printed using a 3D printer. One additional advantage of installing the guard is that there is no need to install plates on the chassis to attach the team alliance and unique numbers, as was required for previous robots.

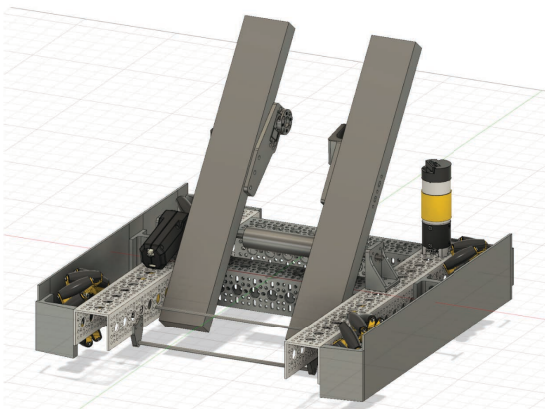
3. Attaching Linear and Servo Motors

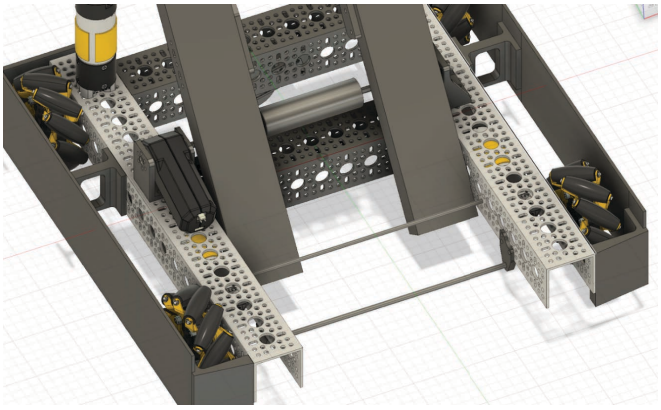




The persistent problem with previous robots, including last year's, was the size of the chassis frame. Since the chassis itself used in competitions was almost identical to the one provided for inspection, it was difficult to attach additional components to the frame. This caused inconvenience when maneuvering through obstacles on the field. To address these issues, we reduced the gap between the two linear motors to match the width of a single pixel and designed and assembled a servo motor mount to rotate the basket based on this adjustment.

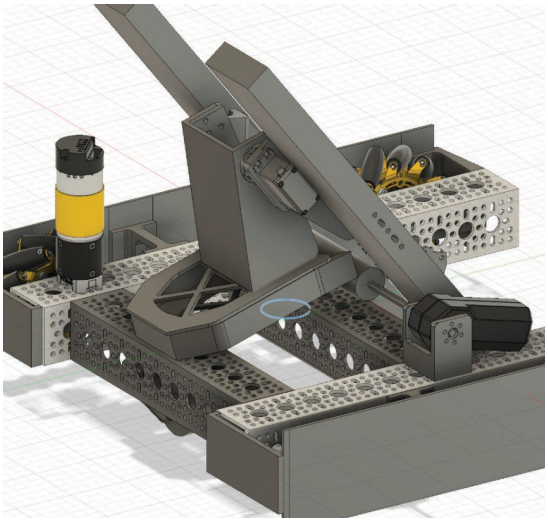
4. Design of the axis that rotates the conveyor belt

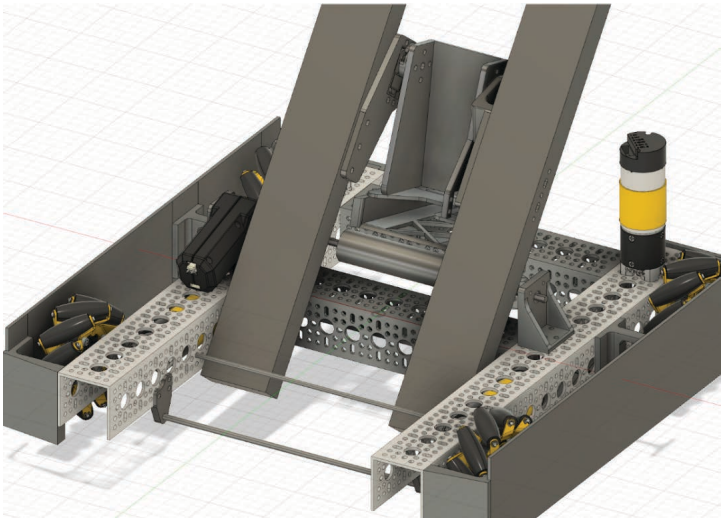




For the conveyor part, we used two axes. Since one rotating axis is sufficient to provide power, we designed a conveyor roller attached to the linear front. The roller is made by cutting a PVC pipe to fit the width between the linear sections and fixing hubs on both sides with instant adhesive. We then designed a hexagonal DC motor mount to provide power and also created a mount on the opposite side to insert bearings. The lower axis of the conveyor is designed to minimize the diameter and reduce the height difference of the pixels. Therefore, we designed it to be able to move pixels from the floor to the conveyor stably without additional rotor parts or slopes.

5. Designing the box that eats pixel

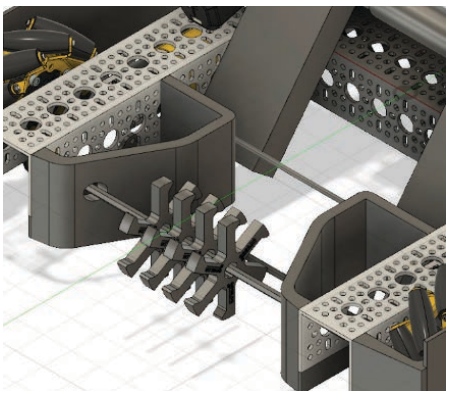




Designed a box structure for placing pixels transferred through a conveyor onto a backdrop. To eliminate dependence on gravity, a servo motor was driven in DC mode in front of the box to create a mechanism that eats pixels. An intake wheel was used, and the height of the box was made slightly larger than the height of the pixels to prevent the pixels from coming out when the intake wheel is not moving. After printing the box, the ease of pixel intake was confirmed. It was discovered that the end of the box was not blocked but open, causing the intake to not work as intended. The box was reprinted after making modifications with tape.

A box arm was designed and printed to rotate the box to an angle that allows it to be inserted into the backdrop. The servo motor was mounted to fit the holes on the linear guide. After attaching the servo motor to the linear guide, a test was conducted. The test results showed that if the chassis is too long, the length of the arm connecting the servo motor and the box needs to be extremely short in order to position the linear guide as close to the backdrop as possible. Therefore, the length of the box arm was made longer to ensure that the height of the linear guide is not too high, allowing for accurate placement of pixels within the appropriate boundary during driving.

6. Eater production



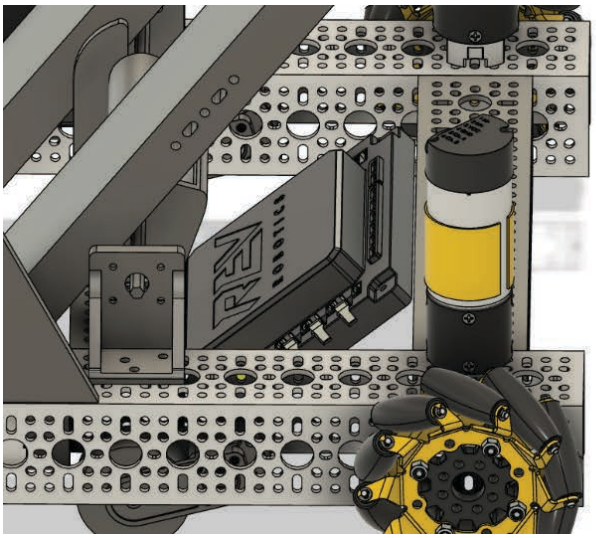
Because we decided to create a conveyor belt, it was necessary to efficiently utilize the front space during the inspection of Eater. Due to the conveyor belt, even by slightly raising the pixels in contact with the floor, the pixels could reach the box. There-

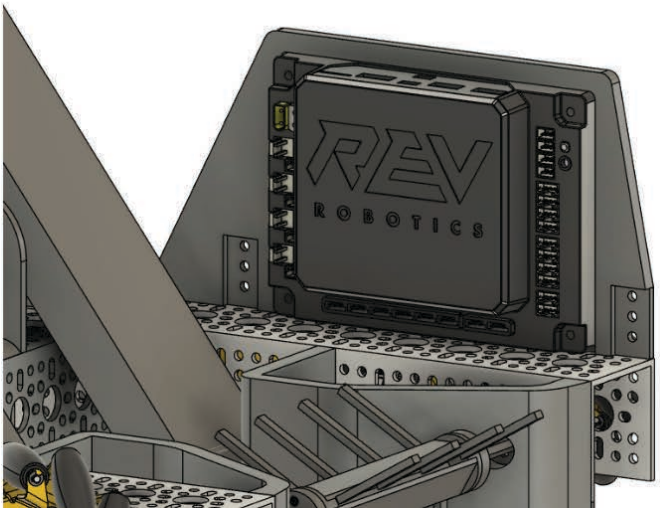
fore, we decided to place the motor in the front part of the chassis without including it in the body, and rotate the axis vertically at a 90-degree angle through a bevel gear. Thus, we purchased four intake wheels with a diameter of 8mm and a length of 72mm from gobuilda to use for Eater. However, there were many difficulties in actually manufacturing it. [Bevel gear, acrylic fixing plate manufacturing photo] Additionally, to resolve the issue of the intake wheels touching the ground by about 5mm, we trimmed the front part of the intake wheel.

7. Eater Guard Production

When the iter eats pixels, a path was designed next to the conveyor to prevent it from eating inaccurately and jumping. It is a structure that does not obstruct the axis of the iter and conveyor, and prevents pixels from falling down and receiving penalties.

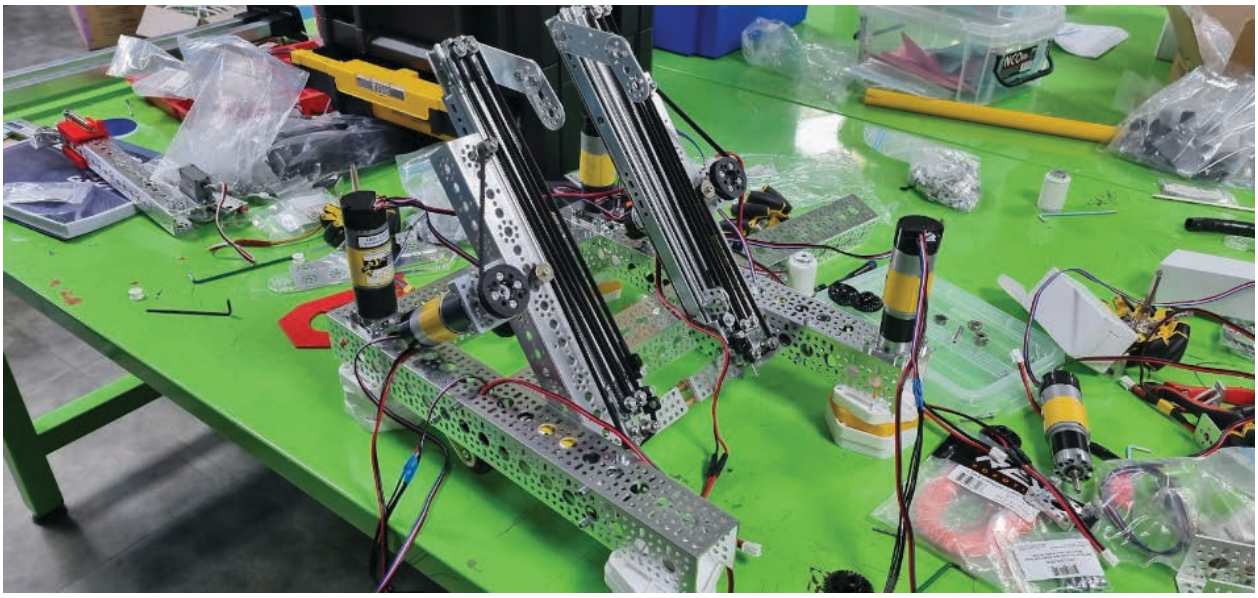
8. Attach the control hub and expansion hub





Due to insufficient space inside the robot, a plate was designed to attach the hubs on the side. Initially, the plan was to attach the control hub at the back of the conveyor and the expansion hub on the side plate. However, there was a problem with the size of the back plate being slightly smaller than the vertical length of the control hub, so it was decided to attach them vertically.

Actual Building Process

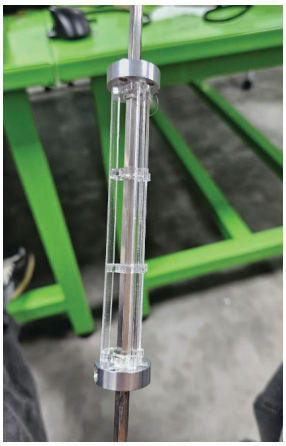
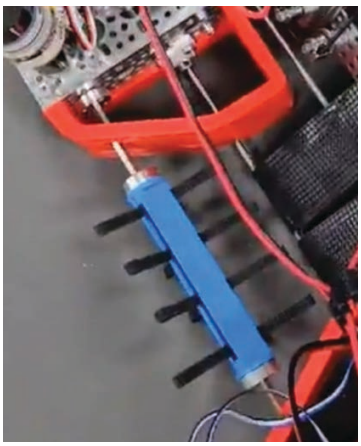


We used a horizontal frame one size smaller to reduce the width of the vehicle body. When the race was actually conducted, the width of the vehicle body was almost the same as the width of the hurdles, causing significant difficulty in driving. This was used as a measure to address this issue.

1. Eater

On the right is the picture of the initial rotor using Gobilda's intake wheel (using a 6mm hex shaft). The intake wheel was made using Gobilda's intake wheel because we wanted to create a simple intake. The diameter of the intake wheel was 8mm, but since our team did not have an 8mm hex shaft, we designed and made a connector using acrylic to connect it to the 6mm shaft. The connector was designed with acrylic printed with an 8mm diameter, with both ends fixed with 6mm hubs, and the intake wheel attached in between. Then, the entire assembly was connected by inserting a long piece of acrylic.



To reduce weight, we replaced the intake wheel part of the initial rotor with a timing belt and assembled a prototype of the new rotor using acrylic. By cutting the timing belt to the appropriate length and bonding it at regular intervals, the rotor was completed as shown in the right picture. In order to improve durability, we plan to replace it with a 3D printed version in the future.

The newly created rotor using a 3D printer is as on the right. It was more stable than when using acrylic. At this time, we did not attach the timing belt to the 3D printed part with any separate adhesive, as it was sufficient to just insert it and leave it in place.

2. Conveyor Belt Manufacturing

After consuming the pixels, we wanted to use a conveyor belt to move the pixels into boxes. To do this, we used a rubber tape with friction. When using the tape as a conveyor, it was necessary to remove the strong adhesive parts on the opposite side in order for it to rotate smoothly. In our search for a solution to remove the adhesive strength of the tape, we found that washing the adhesive surface with hot water and applying sunscreen can partially alleviate the adhesive strength. This is because a sunscreen layer covering the ad-

hesive part of the tape is formed, which reduces the adhesive strength.

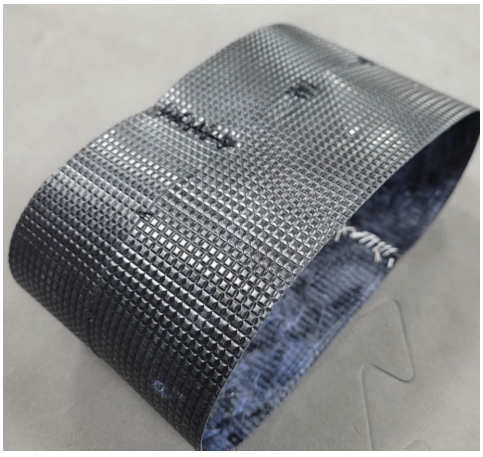
In order to find the optimal way to connect the conveyor as follows, we conducted various experiments:

1) Melt the rubber

We attempted to melt and bond the rubber using a soldering iron or adhesive. However, this caused the conveyor belt to become uneven and inefficient. We concluded that using adhesive would result in the belt falling off when subjected to tension, unless the belts were overlapped. If they were overlapped, there would be areas where the conveyor belt would get stuck while rotating.

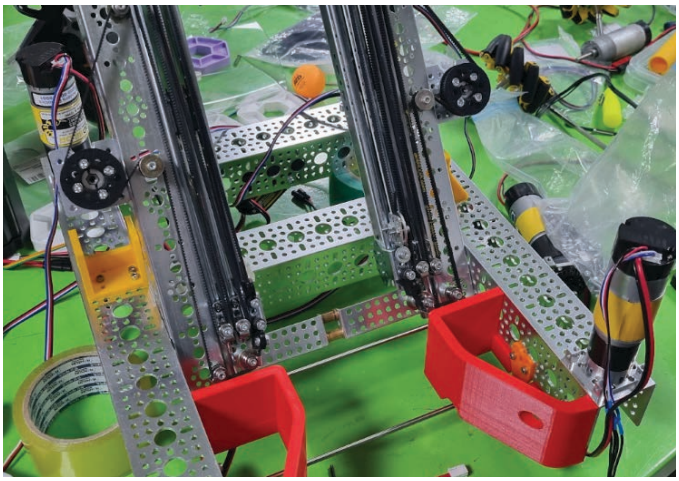
2) Sewing the ends

We considered using sewing to connect the ends of the conveyor belt. We believed that this method would prevent any separate bending at the bonded area and would be able to withstand the tension. In fact, when we tried it, it worked well.



The final improvement solution that was ultimately thought of was to use two conveyor belts, but not to connect them separately with sewing. This way, it was possible to use conveyor belts with the same width as the two conveyor tapes, and there was no phenomenon of the conveyor

belts leaning to one side due to connection errors. Therefore, it was deemed the most ideal solution and it was decided to apply this method.

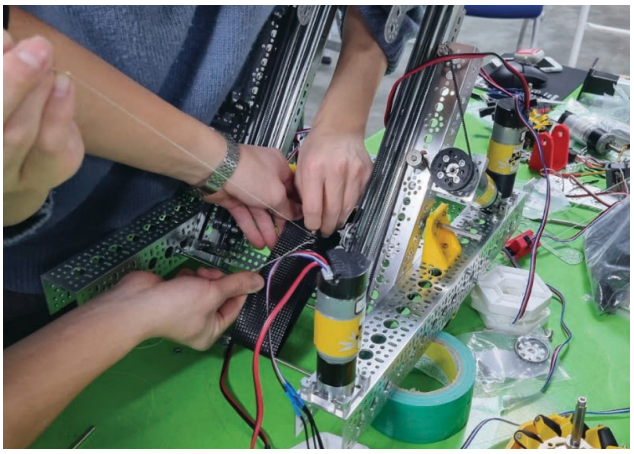


Connecting the bottom axis to turn the conveyor belt. The narrow spacing between the linear inevitably means that the width of the conveyor belt is the same as the horizontal length of a single pixel. Therefore, pixels eaten by the etcher could fall off the conveyor belt, so we printed a red 3D print-out as a wall to prevent this. The axis in the center was installed to hold the conveyor under tension.

A hub was attached to a PVC pipe to connect to the hexagonal shaft to create a roller. I designed the width to be smaller than the width of the linear, and when I glued the pipe and the hub to both ends of the pipe, I observed that it fell apart due to lack of adhesion, so I took advantage of the fact that the diameter of the PVC is slightly smaller than the hub to fit inside. In this case, I made a small hole where the screws go in so that I could tighten the screws that secure it to the axle.

You have connected the conveyor belt to the robot by stitching. We tested the conveyor by running it to see if there were any problems. Since we needed a conveyor belt that was two tapes wide to stably eat the pixels, we made a wider convey-

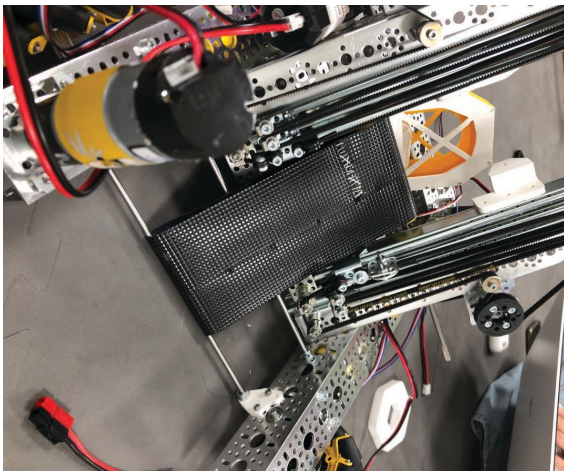
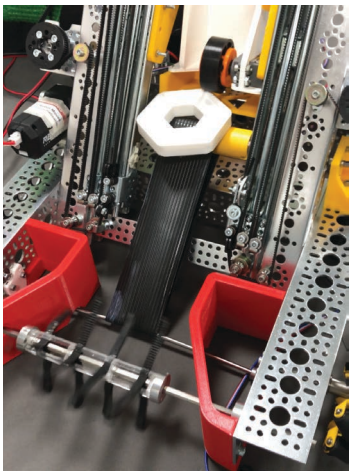
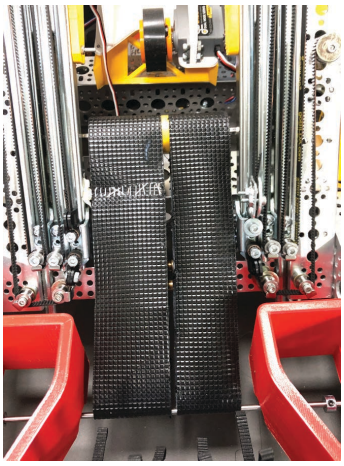
or belt by sewing two tapes that were slightly narrower than the width of the existing tape, as shown in the photo above. However, the slightly different lengths of the two conveyor belts caused the conveyor belt to tilt to one side as it turned.



We tried to see if it was possible to move the pixels with a single conveyor belt, and it worked, but it was unstable because the width was narrower than the pixels, so there was too much possibility that the pixels could fall to the sides. We also observed that if the tension was strong enough, we could see the problem described above with the conveyor moving.

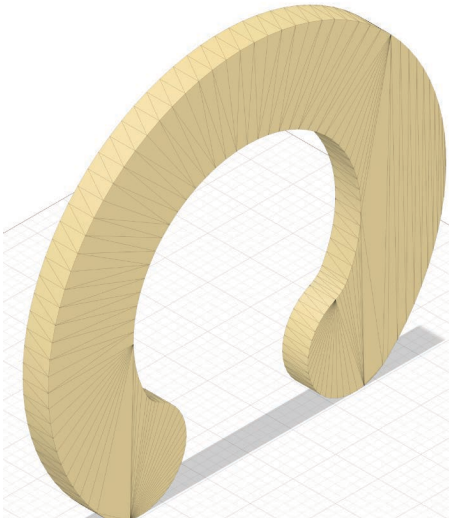
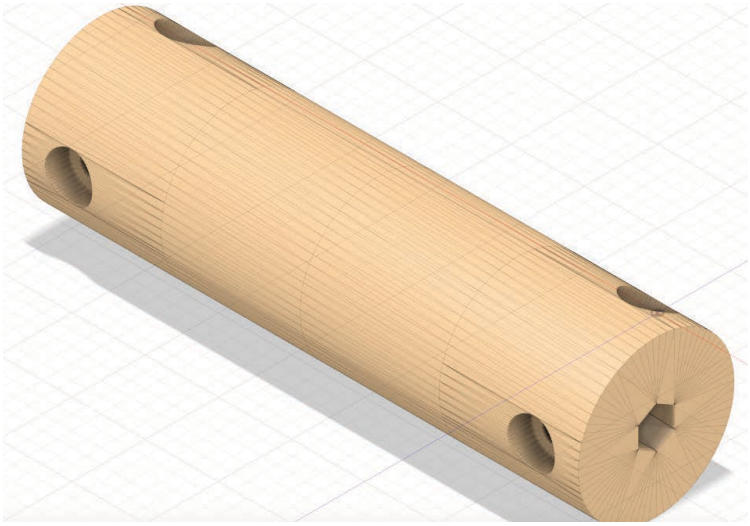
The first completed version of the conveyor belt. After changing the length several times to adjust the tension, we realized that when two tapes were attached back-to-back, they tended to move to one side when the belt was turned due to a slight mismatch in the lengths of the two tapes. Therefore, we modified it by reducing the width of each tape so that two conveyors could run simultaneously.

We decided that this would be the ideal solution because we could use a conveyor belt as wide as the width of the two conveyor tapes, and the conveyor belt would not be tilted to

one side due to the connection error.

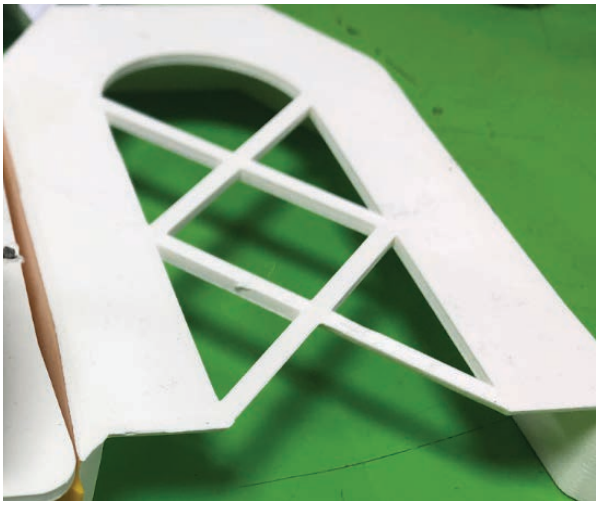
In the process of experimenting with this method, we observed that the rollers and hubs made of PVC pipes moved away from each other, so to solve this problem, we designed and printed an integrated roller with the same diameter as the hexagonal shaft and a screw line to fix it on the shaft through 3D modeling. (See the figure below)

3. Box

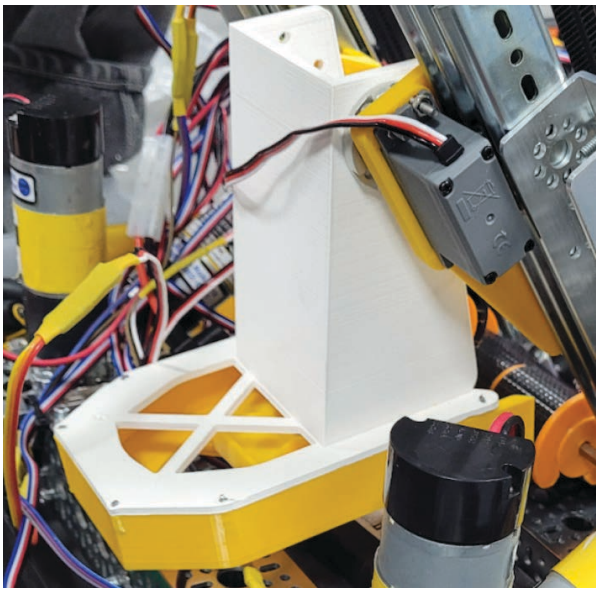
The box was printed and tested using the 3D modeling created above.

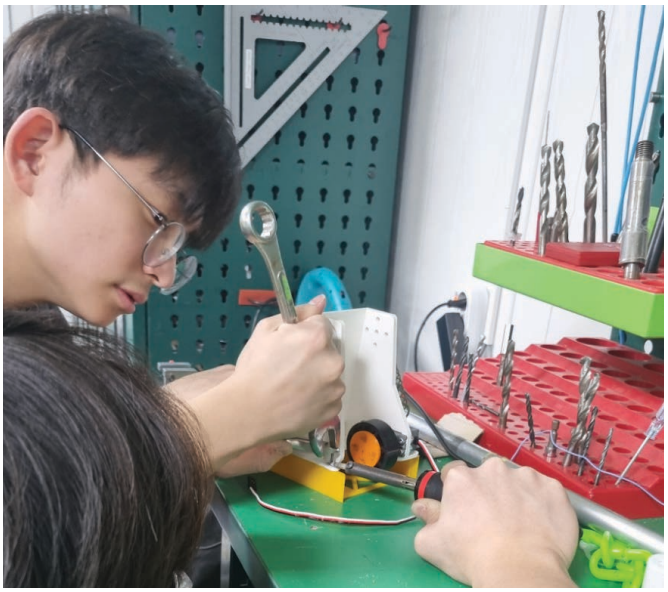
In the initial design, the end of the box was not blocked, but was open, so it was reprinted after realizing that it would get caught in the jaws and not intake. I expected the pixels to fit better if I left more space at the front, but after testing, I realized that the pixels were getting stuck at the end of the initial box when they went in, so I temporarily taped the bottom and that solved the problem. So we corrected this and re-printed it.



Two KROS members heated the nails with an iron to connect the box to the box arm.

The submotor was attached to the linear and tested. The test results showed that if the length of the arm connecting the submotor and box is too short because the car body is too long, the linear must be raised too much and positioned as close to the backdrop as possible. Therefore, the length of



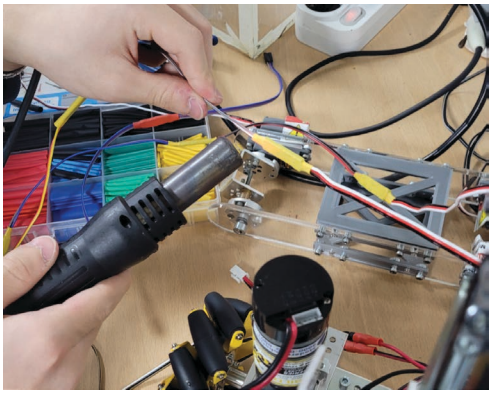


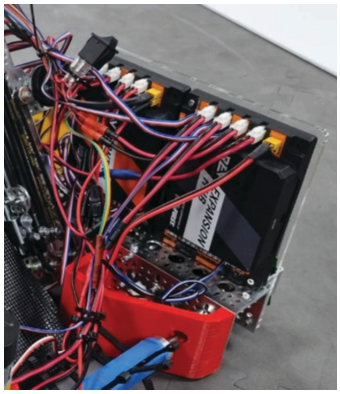
the box arm was lengthened so that the height of the linear could not be raised too high, and the pixel could be placed accurately within the proper boundaries when driving.

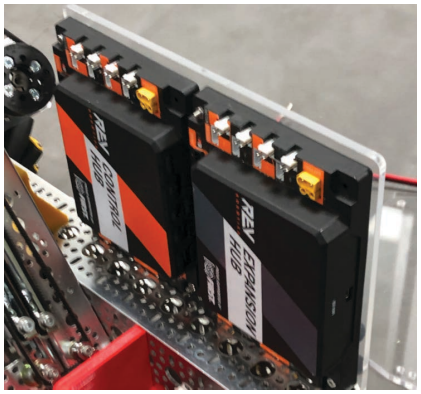
4. Clean Up Wires

The wire of the servo motor connected to the linear slide was extended to prevent it from breaking after the linear slide was raised, and it was fixed using a hot air blower and shrink tube.

On the left side of the robot, the control hub and expansion hub were mounted vertically using an acrylic plate, and the wires of each motor were connected to the appropriate locations, and cable ties and shrink tubing were used to secure them.







Endgame - Drone

<DR05> Construction Material Constraints:

a) The *Drone* must be made of a single, continuous sheet of paper no larger than a single sheet of 8 ½ x 11 or A4 size uncoated printer paper. The paper weight can be no more than 20lb (75 g/m2). Card stock, construction paper, cardboard, photo paper, etc. are not allowed.

b) Graphite pencils, ink pen, and/or felt-tip markers may be used to color or apply the *Team* Number. Crayons, colored pencils, paint, chalk, and similar items are not allowed.

c) Laser or ink jet printers, or similar technology, may be used to apply the required red or blue color, printed decorations, images, *Team* number, etc. on the paper.

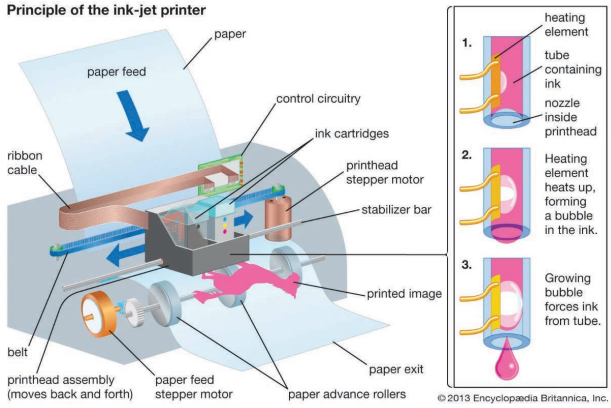
d) No other materials are allowed.

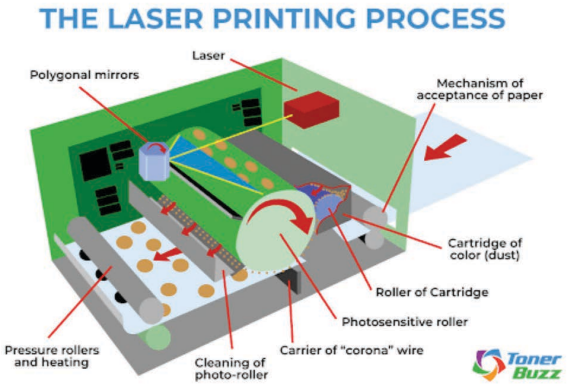
When searching for regulations regarding the type of paper to be used in the production of drones, it is stated that paper weighing 75 m/2 (referred to as 75 gsm) or less should be used. The existing drones were made with 90 gsm paper, so they cannot be used. Therefore, it was necessary to purchase 75 gsm coated paper to make the drone, but it was found that 75 gsm coated paper is not sold in Korea. As a result, it was necessary to use a printer to produce 75 gsm coated paper. Printers can be broadly divided into two types: inkjet printers and laser printers.

An inkjet printer is a type of printer that is widely used and works by spraying ink through fine nozzles to sequentially print the output onto paper. It mainly uses liquid ink and is characterized by relatively high resolution compared to laser printers.

On the other hand, a laser printer is an electrophotographic printer that uses a laser to create an image on a photosensitive drum in the form of powdered toner, which is then heated and transferred onto paper for output. To explain the principle of a laser printer in more detail, it generally operates using the principle of electrostatics. The powdered toner is recorded on a metal rotating drum and then transferred back onto paper, which is then heated to produce the out-

put. Specifically, the drum, which carries an electric charge, is exposed to a laser to indicate the output state, and the powdered toner, which carries an opposite charge, is applied only to the areas where the laser was applied, resulting in the desired shape of the output. The toner is then attached to the paper, which carries a stronger opposite charge, and is fused to fix the shape, resulting in the desired output.





After examining the operation principles of the two printers, it was determined that a laser printer, which utilizes the principles of electrostatics and toner, would be more helpful in producing the color scheme for the Drone compared to an inkjet printer. Before proceeding with the actual printing, a common 80 gsm paper was printed with a laser printer. The results of printing color on both sides revealed two critical errors. The first is the fatal flaw that the color does not print perfectly up to the edges of the paper, and the second is the need to consider the weight of the toner when producing the color scheme.

First, the excess weight due to the ink can be easily solved by using paper with a lighter gsm. Since there is no paper lighter than 70 gsm according to the search results, we used YES 70g paper. Next, to print the paper perfectly up to the edge, we resolved it by setting B4 paper for printing and inserting A4 paper for printing. The printer used for this is CN/ApeosPort

C3070G.

In addition, a precise electronic scale was needed to measure the weight of the paper. A scale capable of measuring up to 0.001g was required, and the WBA-620 electronic scale was used for the measurement. From assembling the electronic scale to basic settings and operation methods, the experiment was conducted independently without the assistance of the teacher.

To compare the measured value with the actual weight of the paper, the weight per sheet of A4 paper was calculated for each GSM. The calculation method is as follows:

(known GSM of the paper) * 1m : 10^4 cm^2 = x : 21.0cm * 29.7cm

Calculating using the above proportion equation, we get:

80 gsm	4.98 g
75 gsm	4.67 g
70 gsm	4.36 g
65 gsm	4.05 g

[Table] Mass per sheet of A4 paper

According to the regulations, the specified weight of the paper is 75 gsm, so the total weight including the paper and ink should be 4.67 g. Since the lowest gsm paper available is 70 gsm, we can determine the weight of ink (toner) that can fit on one sheet of paper. This is an unrealistic weight of 0.31 g for an inkjet printer. Therefore, we have decided to use a laser printer. As a trial, we printed red and blue on both sides of a 70 gsm paper and measured the weight.

[Fig] 70 gsm paper

[Fig] 80 gsm paper

	Blue	Red	A4
80 gsm	5.544	5.967	4.9896
ink weight	0.5544	0.9774	0
70 gsm	5.102	5.213	4.3659
ink weight	0.7361	0.8471	0
average	0.64525	0.91225	-

[Table] Weight of Ink

The measurement results are values that far exceed the target value, with 0.645 g and 0.912 g of ink used, which is well above the required ink of 0.31g. Based on these results, two issues were identified.

The first issue is that the amount of ink (toner) used is excessive. When folding a paper airplane, there are many parts that are folded and hidden, and it was determined that ink was excessively used even in these hidden areas. To address this issue, it was attempted to redesign the paper airplane as shown in the diagram below, by folding and confirming the hidden parts to solve the problem.

The second problem that was discovered is the moisture. When measuring the gsm of paper, usually only the weight of the paper itself is measured, excluding the weight of moisture. Even if the weight of the paper with a moisture content of less than 4% is measured as 70 gsm, when the paper comes out of the warehouse and is distributed, it will

absorb moisture from the air as soon as it is exposed. To prove this, we measured the weight of empty paper with 70 gsm and 80 gsm using WBA-620.

The measurements were taken on different days for two types of paper, with the intention of considering the condition of the air in South Korea, where the humidity is relatively high. Calculating the average moisture content of the two types of paper reveals that it is approximately 0.13 g. Based on this, it was deduced that a paper composed of 75 gsm A4, when printed, should weigh 4.80 g. Taking into account the previous two issues, the paper airplane was remade, and its weight was measured again.

In conclusion, the printing on 70 gsm paper resulted in the production of 75 gsm paper with a relative error rate of 0.05% for Blue and -0.09% for Red, demonstrating precise production results of less than 0.1%. It is worth noting that it takes approximately 10 minutes to fold one drone, and for the purpose of experimentation and testing, we produced nearly 100 sheets, which required approximately 1,000 minutes, or close to 16 hours of production time. When factoring in measurements, printing, and other calculations, we invested approximately 20 hours solely in the production of the drones.

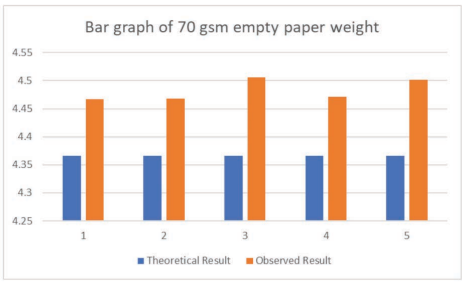
	Blue	Red
average 70 gsm colored	4.8026	4.7956
Expected 70 gsm colored	4.8	4.8
error(%)	0.05413734	-0.0917508

[Table] 70 gsm *Drone* Error Rate

All of our experiment data and charts are shown on the next page.

70 gsm A4 (Empty)							(unit: g)
	1	2	3	4	5	avg	
Theoretical Result	4.3659	4.3659	4.3659	4.3659	4.3659	4.3659	
Observed Result	4.467	4.468	4.506	4.471	4.501	4.4826	
Moisture	0.1011	0.1021	0.1401	0.1051	0.1351	0.1167	

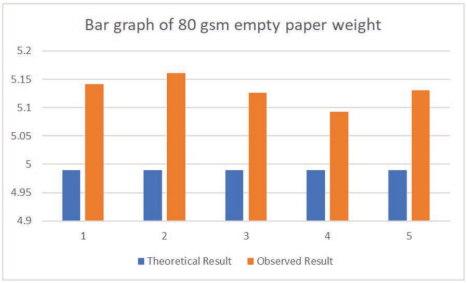
[Table] 70 gsm Empty Paper Weight



[Fig] Bar graph of 70 gsm empty paper weight

80 gsm A4 (Empty)							(unit: g)
	1	2	3	4	5	avg	
Theoretical Result	4.9896	4.9896	4.9896	4.9896	4.9896	4.9896	
Observed Result	5.141	5.161	5.126	5.093	5.13	5.1302	
Moisture	0.1514	0.1714	0.1364	0.1034	0.1404	0.1406	

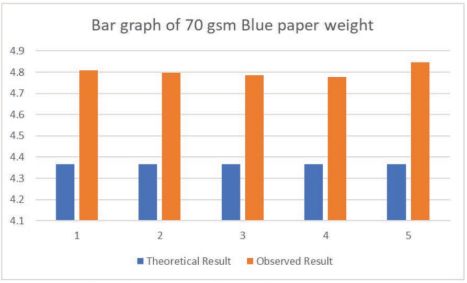
[Table] 80 gsm Empty Paper Weight



[Fig] Bar graph of 80 gsm empty paper weight

70 gsm A4 (Blue)							(unit: g)
	1	2	3	4	5	avg	
Theoretical Result	4.3659	4.3659	4.3659	4.3659	4.3659	4.3659	
Observed Result	4.809	4.798	4.785	4.776	4.845	4.8026	
Moisture + Ink	0.4431	0.4321	0.4191	0.4101	0.4791	0.4367	
Ink	0.3264	0.3154	0.3024	0.2934	0.3624	0.32	

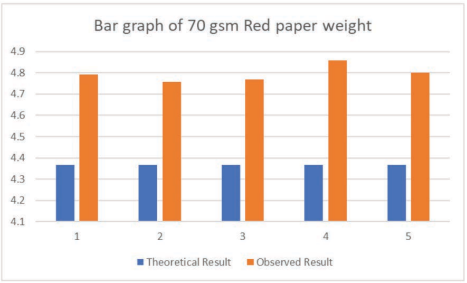
[Table] 70 gsm *Drone* Blue Weight



[Fig] Bar graph of 70 gsm Blue paper weight

70 gsm A4 (Red)							(unit: g)
	1	2	3	4	5	avg	
Theoretical Result	4.3659	4.3659	4.3659	4.3659	4.3659	4.3659	
Observed Result	4.791	4.757	4.769	4.859	4.802	4.7956	
Moisture + Ink	0.4251	0.3911	0.4031	0.4931	0.4361	0.4297	
Ink	0.2951	0.2611	0.2731	0.3631	0.3061	0.2997	

[Table] 70 gsm *Drone* Red Weight



[Fig] Bar graph of 70 gsm Red paper weight

#25309
TALOS

Programmer

FIRST Tech Challenge Programming Note

We’re the Champions from Korea.
Take a look at what we’ve been through.

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST

TALOS 2023-24

Robot Code Class Diagram

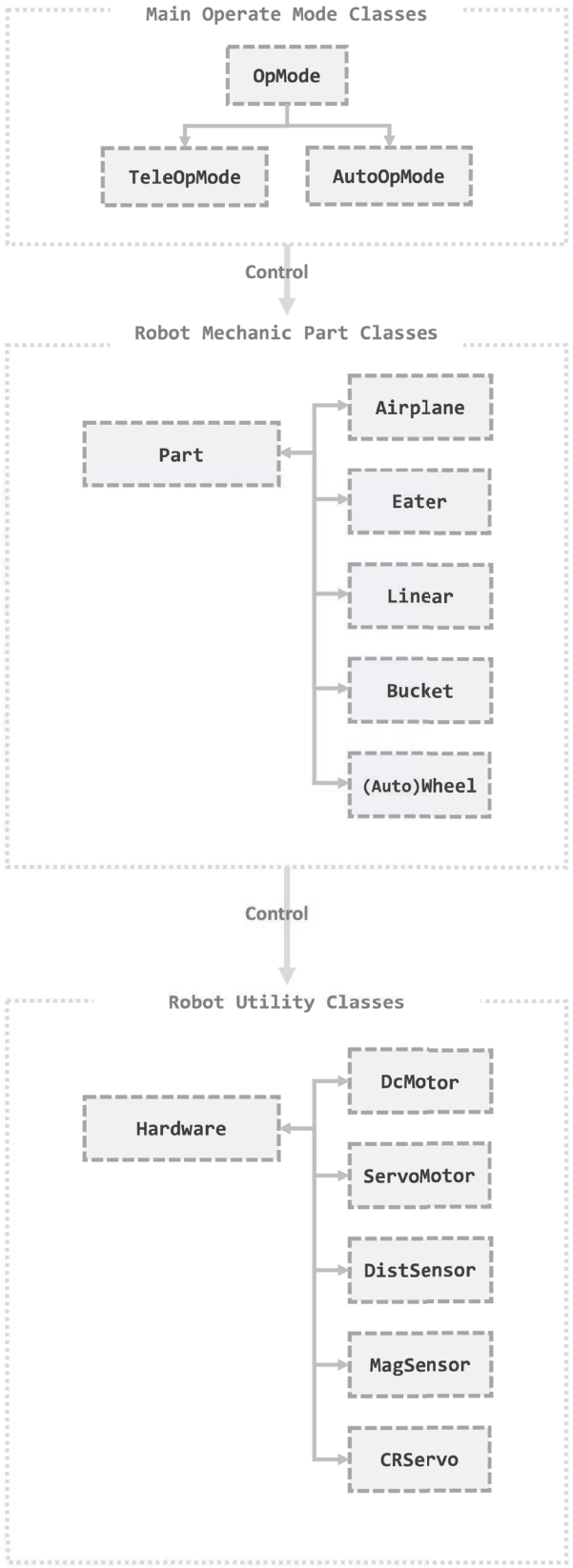
Programming Team : Taewoo Yoo, Joonsung Kim

OpMode : Operating Mode Class

Part : Robot Mechanic Part Class

Hardware : HW Managing Class

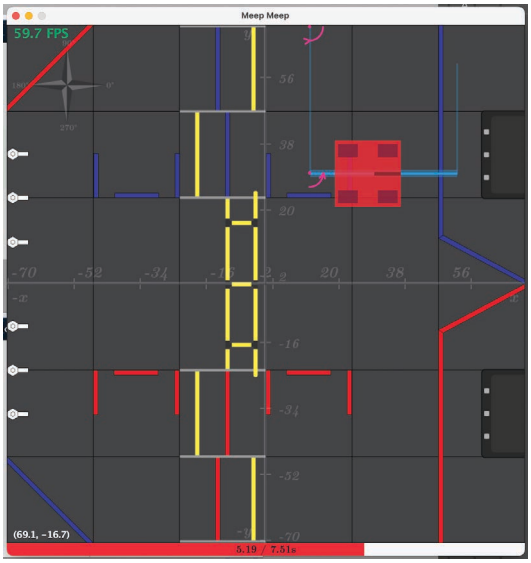
Procedure for a parent class to pass behavior commands to a child class

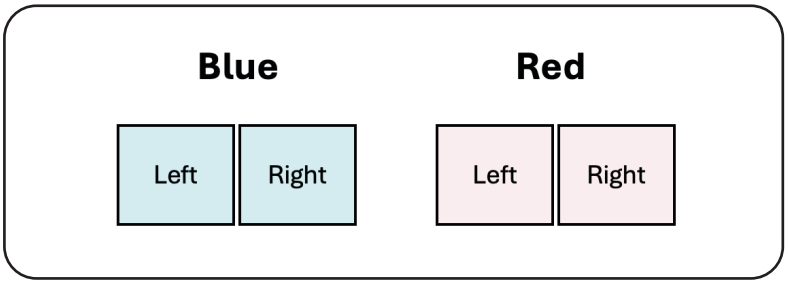


Github
https://github.com/KSA-KROS/FTC2023-2024_RobotSourceCode

Robot Movement

We visualized our robot's movement with MeepMeep, an open-source tool on GItHub. We developed 4 different AutoOpModes for different starting locations.





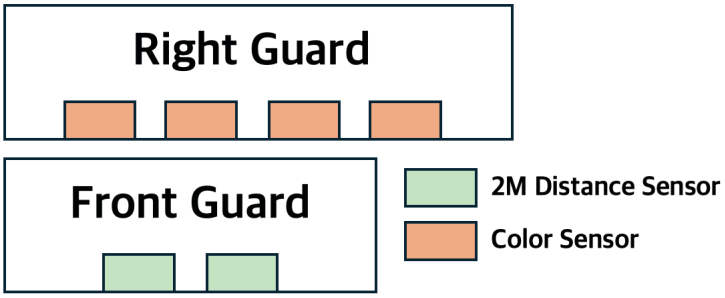
Randomization Objects

We used red and blue cubes made from foam boards, to keep it light-weight but still rigid.

Sensor Placement

In the autonomous period, we depend on sensors to locate our robot and detect randomization objects.

We use 2M distance sensors and color sensors attached on sides to detect objects. Placement of the sensors is shown on the right.



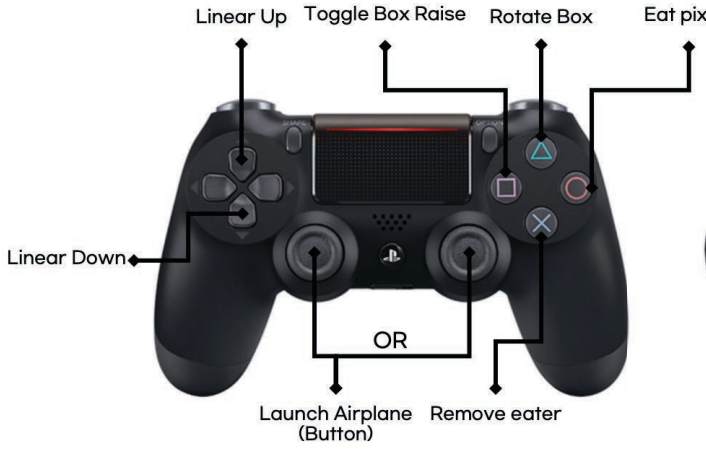
Autonomous Scores (KRC)

	Q1	Q2	Q3	Q4	Q5
Navigation Points	20	20	0	20	20
Pixel Points	5	5	0	5	5
Purple Bonus Points	5	5	0	5	5
Yellow Bonus Points	20	0	0	0	20
Total	50	30	0	30	50

Driver Controls

Controller Guidebook
DRIVER #1 : Robot Movement

Controller Guidebook
DRIVER #2 : Linear, eater, bucket



Controller Guidebook
EMERGENCY MODE

Key changes in FTC

Robot programming hasn't changed a lot from our code written for KRC, since we structured it to be maintainable and reusable. Some changed components are described below.

1. Odometry Wheels

We used new odometry wheels to increase accuracy, and wrote the AutoWheelPart to control it. AutoWheelPart is an upgraded version of WheelPart, and can make the robot return to original position even when pushed.

2. New BucketPart

After changing the pixel dropping mechanism from pincer to bucket, we removed the PincerPart code and wrote a new BucketPart code.

BucketPart consists of a set of commands for rotating the bucket and storing / dropping pixels. It controls a single servo and a single continuous-rotation servo.

3. TeleOp Change

As we changed some parts such as the bucket, we changed the driver controls as shown on the left.

#25309

TALOS

FIRST Tech Challenge
2023-24 Season

Presented by
KROS Robotics Research Group,
Korea Science Academy of KAIST