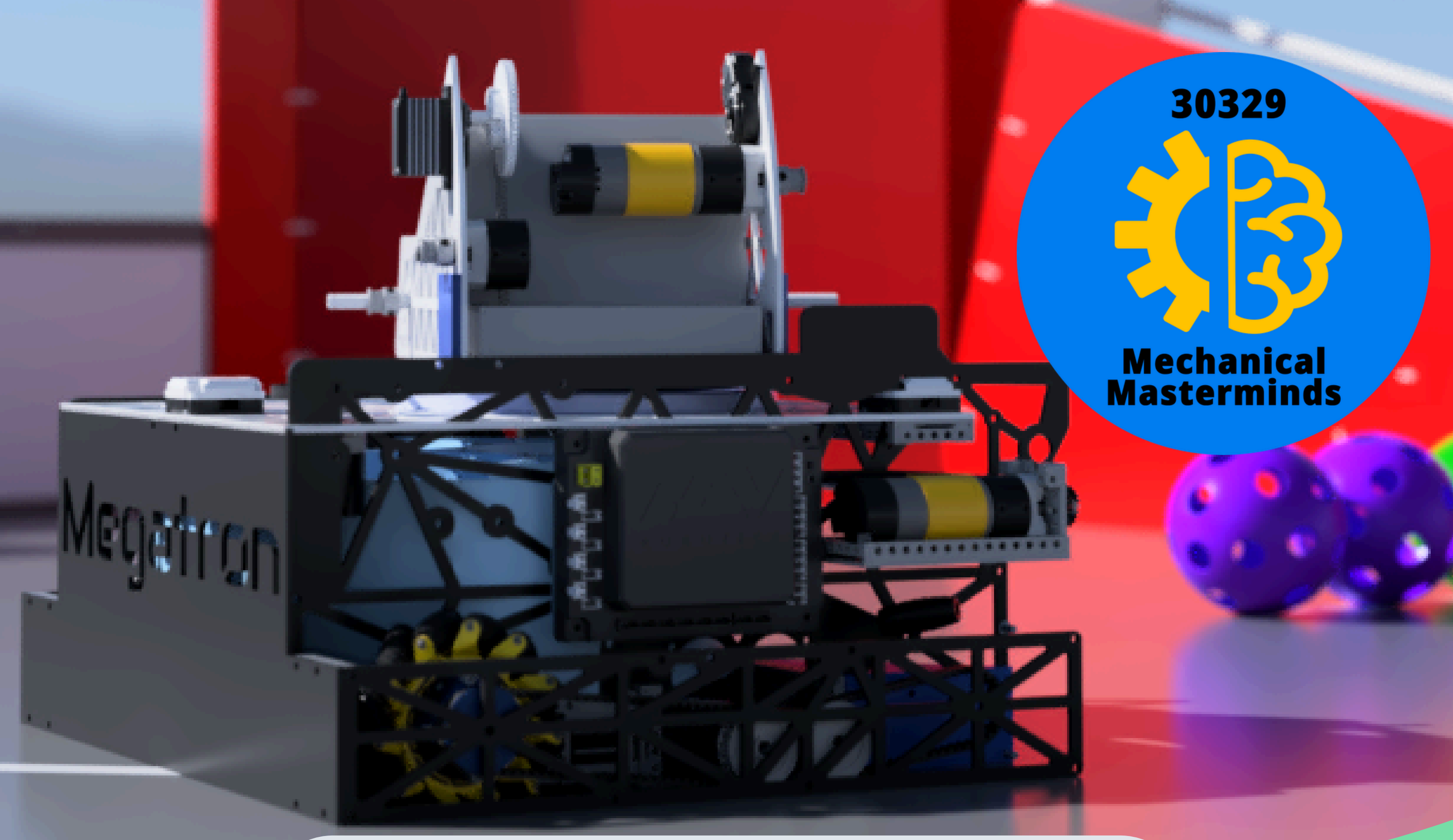


30329
Mechanical Masterminds

MECHANICAL MASTERMINDS #30329

DECODE™
PRESENTED BY RTX

Engineering Portfolio 2025 – 2026

Team Introduction

Team 30329, Mechanical Masterminds, from Howard County is a **diverse rookie** team made up of 8th and 9th graders. Our goal is to inspire students to pursue STEM while fostering an environment of **inclusion** and **diversity** while always striving to improve in any way we can. Being a rookie team meant that we had **no experience** for First Tech Challenge going into the season. We worked very hard learning how to work as a team, getting work done and about how FIRST Tech Challenge runs.

Team Management

Our team is fully **student-led**, which means that meetings are scheduled and ran by **team leaders**. We have a lead and a deputy that oversees the agenda, as well as individual leads for different departments.

Our coaches are there to teach us things we may not know, make sure we get to competitions, and communicate to parents!

Team Goals

Our team goal is to **inspire kids** to **explore robotics** and the opportunities it can open up. We want everyone to experience how **exciting** and **meaningful** robotics and FIRST really are through doing projects, **working together**, and solving problems. FIRST creates a community where students can **grow**, learn new skills, and have a great time!

Team Members

Alex
Team Lead
8th Grade

Adi
Team Deputy
9th Grade

Shriya
8th Grade

Jenna
8th Grade

Dominic
8th Grade

Arnav
8th Grade

Yuvraj
8th Grade

</> Software - codes our robot

Outreach - organizes events and social media

Hardware - builds and assembles

CAD - designs and modifies

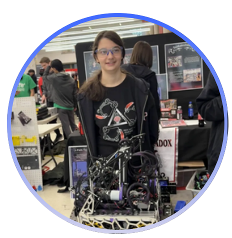
Drive team

DEVELOPING LEADERS

- We have sub-team **leaders** who prioritize tasks for the team and manage their schedule
- We have a team lead and team deputy who help lead our team to **success**
- During this robotics season we had people lead different things in their **sub-teams** while helping to contribute to every aspect of our team.

Coach Luke
Lead Coach

Coach Rabi
Assistant Coach



Miriam
Mentor

Recruitment

Getting involved with FIRST

We are team 30329 Mechanical Masterminds. Want to get involved in FIRST? If so, fill out this form so we can help you get involved with FIRST and a program fit for you!

Contact us at mechanicalmastermindsftc@gmail.com or visit out website at https://www.team30329.org/

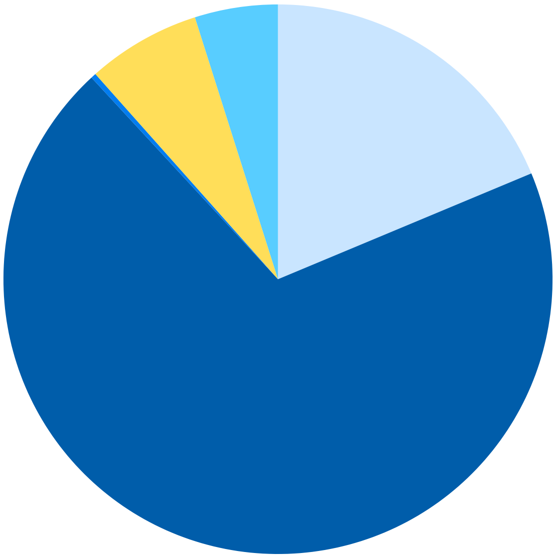
- In order to get children of all ages into FIRST we made a form
- At our library class, we gave it to kids to fill out so we can help them sign up for FIRST
- We helped find a team for them, or help make a team if they fill out the form

We have helped **9** people get involved with FIRST

- In the summer, we joined a class for rookie teams, where we met Miriam, an FTA
- Previously, she was part of 4 teams
- She helps us through our communication server (discord)

We had several people reach out expressing interest in joining our team, and we have been actively helping them find teams to join

Our Expenses



Off the shelf parts	$4345
Custom parts	$901
Competition fees + materials	$858
Other (Jerseys, mouses,)	$834

- Worked with a tight budget, prioritizing only essential purchases
- Reduced costs by reusing hardware and 3D printing custom parts
- As a rookie team, invested in setup items like field elements and learned effective budget management
- Secured funding through fundraising and sponsorships

Getting Sponsors & Mentors

- As a rookie team, we needed significant funding to get started
- Our team lead contacted multiple organizations and companies for sponsorships
- Noticed many sponsors preferred tax-deductible donations
- Registered for a Fiscal sponsorship with a 501(c)(3) to make all donations tax-deductible and increase sponsor support

BUILD & BRIGHTEN AROUND THE WORLD

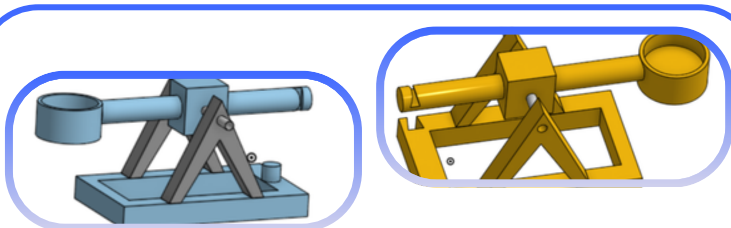
Our Goal

- Create boxes that can be sent to kids in hospitals **around the world**
- Kids in hospitals are able to have fun, **learn about robotics** and **engineering**

We created a discord server where many teams from around the world can communicate and help us **bring boxes to hospitals**.

- **Our boxes teach them different STEM principles**
- We spent time researching and designing each model so they are **fun** yet **teach** children something new!
- We are recruiting teams from around the world to join us!

22 teams have joined to help the initiative so far!



Catapult

- Teaches the Engineering principles of **Energy Storage** and **projectile motion**

Teams that are participating in B&BAW

We have recruited teams from
The United Kingdom
Libya
Australia
The United States
Canada
Kazakhstan

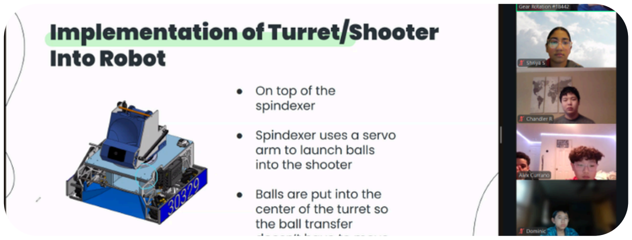
We recruited teams by sharing our discord through other discord servers. All of the teams help us by giving ideas and feedback for our designs and delivering boxes to their local hospitals.

This encourages the kids to feel that they are able to engage with the STEM community and be a part of FIRST.

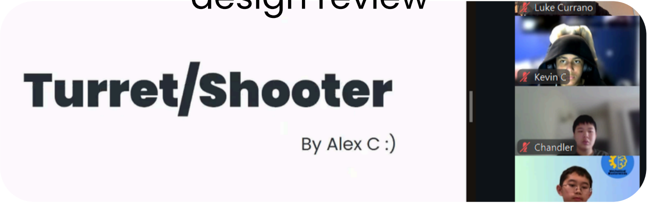
Because we are bringing the boxes to hospitals, we need to make sure that we follow hospital guidelines and sterilize them so they are safe to bring to the children.

- Connected with and met 2 professionals from Johns Hopkins APL
- Educated us on **soldering** and **wiring**

- Met with 2 teams on zoom for **design reviews**
- Extremely helpful, answered our questions and gave **feedback** on our robot design
- There was significant **design improvement** from this collaboration
- More experienced engineers helped us with our designs



Zoom call with Gear Rotation for our design review

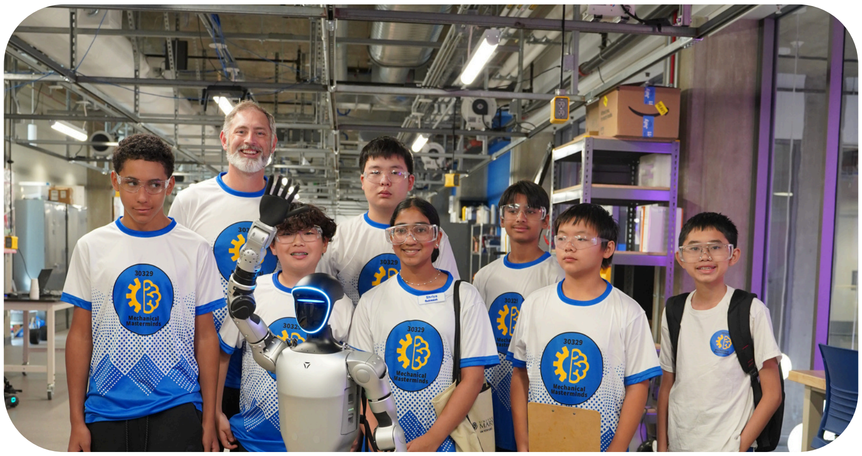


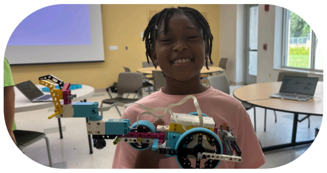
Zoom call with Equilibrium.exe for design review

Our goals for each team members development

- **Hardware + CAD -**
  - Learning how to solder
  - Using the engineering process
  - Designing a fully custom robot,
  - Learning how to CAD in Onshape
- **Software -**
  - Learning how to code in Java
  - Learning how to use object oriented programing
- **Outreach -**
  - Learning how to manage events,
  - Managing and updating social media and our public website
  - Creating a portfolio

- Went to the **Maryland Robotics Center** and learned a lot about **robotics** from **college students**
- Connected with **professional engineers**
- Had a lot of fun learning about robotics and competitions and their **work with robots**





SumoBot designed at a StemTree camp

- Created a **public website** (**team30329.org**)
- Stayed **updated** with social media

- **321** people reached
- **202** hours spent on outreach
- **10** outreach events

- Attended 3 **STEMTree** camps to **present** our robots to students
- Led hands-on activities where kids:
    - Drove and played with our robot
    - Designed and built SumoBots
- Reached **27 kids** total

- Helped 2 schools plan their first STEM night.
- Participating in 3 other STEM nights

Building Legos for our FLL robot at the Joint Base Andrews Air Force Show.

Photos from the Joint Base Andrews Air Force show

- Had a library class at a Howard country library
- Over **50** people attended
- Taught kids how to join FIRST

- Attended to the Joint Base Andrews Air Force Show
- Presented with **FIRST Chesapeake** and Equilibrium.exe
- **Met** and **impacted 167 people**
- Showed kids about how fun FIRST is

- Hosted a FTC Scrimmage
- Had **4 other teams attend,** (around **50** people attended)
- Learned and **connected with other teams!**
- Teams were able to practice and prepare for competition

Outreach - Sustainability

- Each person has their own **role** in the team based on what they are best at
- We have 4 **sub-teams**, one team lead and one deputy lead
- Sub-teams are **Hardware**, **CAD**, **Software** and **Outreach**

- **~$6910** in donations
- ~**$4625** in sponsors
- ~**$240** in fundraising
- Used Hack Club Bank to keep track of our savings to make sure we don't spend more than we can afford

These photos from some of the STEM camps we went to

These are from our Soldering Workshop

- Use documents to **track** what we get done during the meetings
- Use ClickUp to track our **goals** and **deadlines**

- Before becoming an FTC team, we participated in **FLL** (FIRST Lego League)
- Because FTC requires more work, we needed to get **more members** for our team. Recruited **4 new members** to our team, **1 new coach,** and **1 judge!**

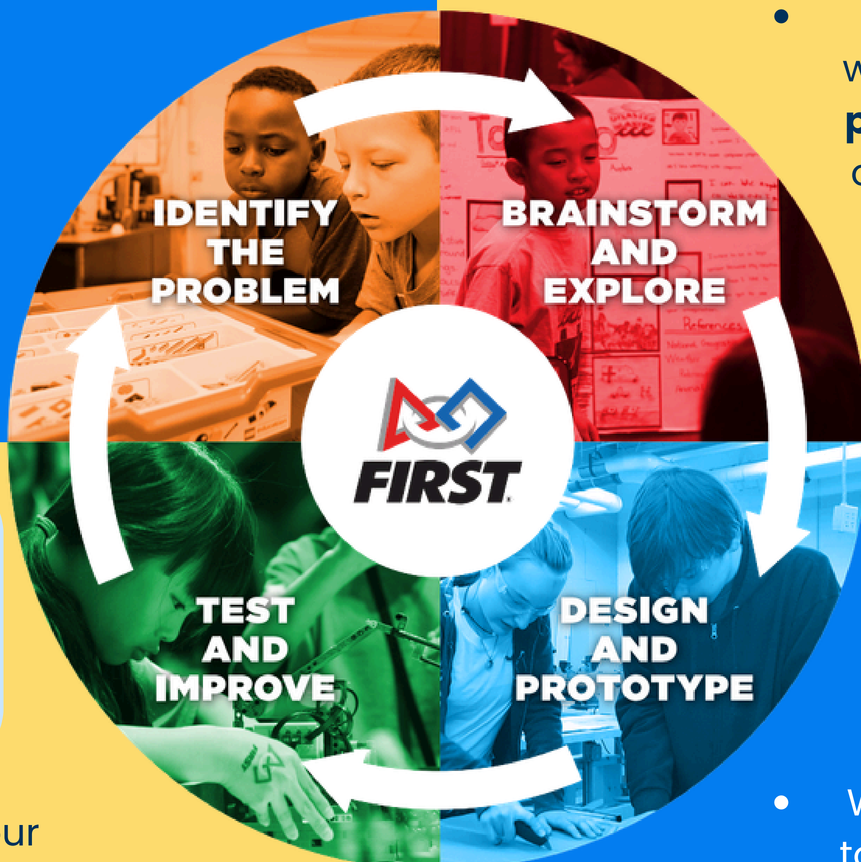
- Plan to **keep our team going for years** more
- Once we outgrow FIRST, we plan to have other members take our spots to keep our team going

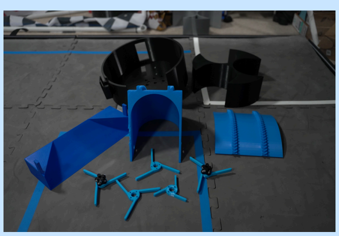
Robot Design Process

Solutions:	Metric 1	Metric 2	Metric 3	Metric 4	Metric 5	Totals
Strafer	1	1	2	3	2	27.
Parallel Plate	2	4	3	2	2	45.
Swerve	3	2	4	1	1	36.
Traction Wheels	3	3	1	4	2	41.

- Used the **Engineering Design Process** for robotics to design all subsystems.
- To show this process, we can use one of our sub-systems, the **Shooter**
- **Our problem**: we needed a **reliable** way to shoot artifacts in order to score points.
- We used this to **revise** and **update** our bot for our second qualifier for more **efficiency**

- After identifying the problem, we spent time determining what we needed to do to solve it by trying to come up with as **many ideas** as we could.
- **Decision Matrices** were used to assign **point values** to rate options against key criteria.





- We used **Onshape** to CAD and **design** our subsystems, spending 7 weeks on the robot design.
- Two team members created different shooter designs, and we voted on which one we preferred.
- After choosing a design, our CAD and Hardware teams worked together to complete it in **CAD**.

- We tested all of our different versions by either **3D printing** or laser cutting.
- When we identified issues in an early version, we returned to Onshape, made the fixes, and **continued testing** until it worked.
- This is how we applied the engineering design process and adapted it for robotics!

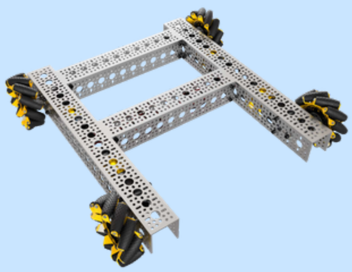
Design - Drivetrain & Intake

Requirements

- Highly versatile with many mounting opportunities
- Efficient with space
- Swift movement at a notice (less gear play)

Drivetrains Considered

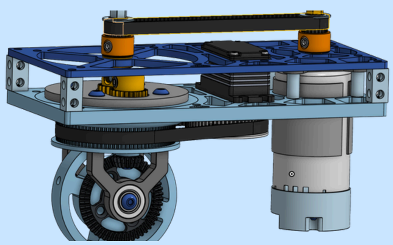
We switched from 223 rpm to 435 rmp to collect artifacts quicker!



Strafer kit w/ traction wheels

Pros: Great at defense
- Easy to build and design
- Great modularity

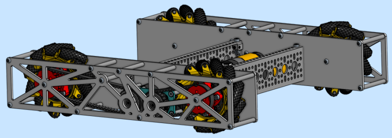
Cons:
- Not maneuverable
- Heavy
- More gear play



Swerve Drivetrain

Pros: High traction wheels
- easy maneuverability
- Great at defense

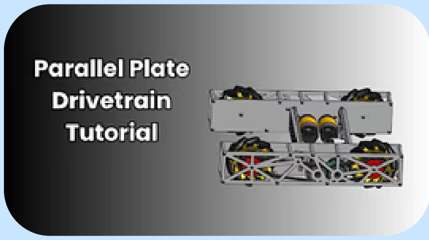
Cons:
- Not Reliable
- No starting ground for code
- Custom Parts are expensive



Parallel Plate Drivetrain

Pros: More maneuverability
- Lightweight
- Reliable
- Less gear play

Cons:
- **Mecanum wheels** are bad at defense
- **Custom Parts** are expensive



What: We made our own video to help other teams design a competitive drivetrain
Why: When we were making our **drivetrain,** we didn't see many reliable videos to CAD them.

Requirements

- A fast and effective intake that pulls in balls.
- Reliably feed the balls into the Spindexer (Sorting system)

What:
We chose to use two different sizes of gecko wheels for the first and second stage. We added a second stage of intake.

Why?
- We chose gecko wheels because our other forms of intake got stuck and didn't allow the balls to get into the spindexer
- The second stage helps keep artifacts from getting stuck in the intake

Requirements

- Fast sorting and cycle time
- Efficient with space
- Simple and consistent

Sorting Systems Considered

- **Spindexer (chosen) - spinning divider cycles through artifacts**

  Pros: Takes up less space, simpler design

  Cons: Slower cycle time, uses one more servo
- **Tube Sorting System - two tubes divide artifacts into separate colors to sort**

  Pros: Faster sorting, less servo use

  Cons: Takes up more space, less simple design

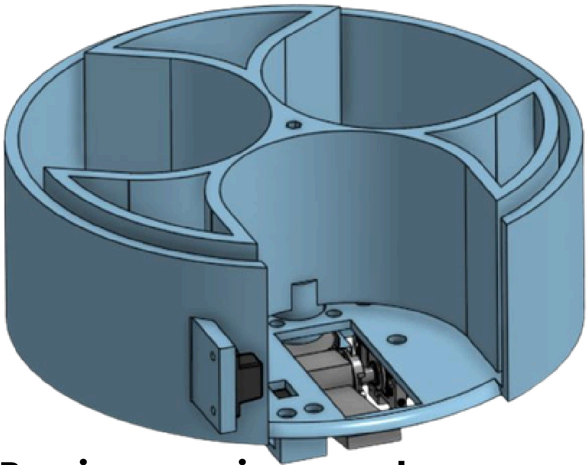
Spindexer V1 + V2

Detection: Only had a **color sensor**, color sensor could detect other things besides artifact, Color sensor was now oriented to hit the top of an artifact

Handoff: Used an **omni-wheel** which couldn't effectively move artifacts + A servo arm was added

Spindexer

Detection: A **Color Sensor** detects when artifact has entered and detects the artifact's color

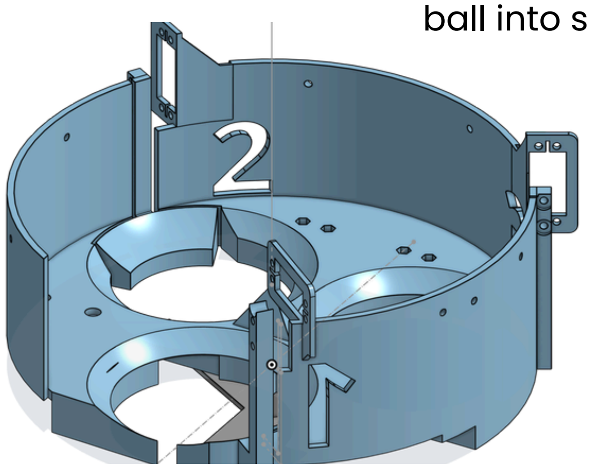


Sorting: Spinning divider cycles through artifacts until desired color is in front slot

Power: Servo spins the divider. **Previous versions used a motor, which was unnecessary**

REV Through Bore Encoder needed because servo can't track position continuously

Handoff: Servo arm launches ball into shooter



Freesort (Future Plans)

- Faster than Spindexer
- Does not need to rotate to shoot
- Faster color sensing
- 3 different transfers resulting in faster output
- Still working on integrating into robot

Design - Turret & Shooter

Why do a turret?

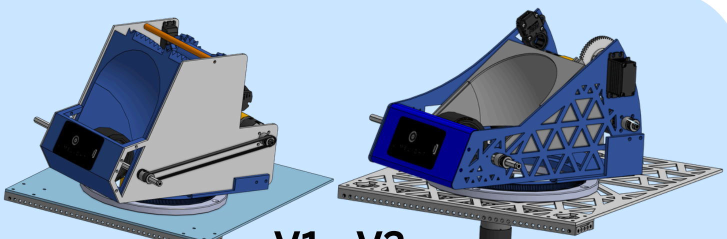
Cycle time: We wanted a turret because we didn't want to go to a specific angle due to human error

Accuracy in shooting: Having a turret allows us to be more accurate in shooting

Transfer: Since we transfer to our shooter through the center of rotation of our turret, we can make an easy arm transfer

Requirements for shooter

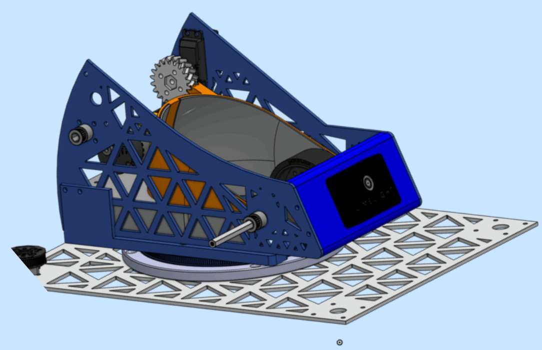
- Able to shoot anywhere in both launch zones
- Quick cycle time



V1 - V2

Key Features:

- Axon servo **changes angle of shooter**
- Dual Motor flywheel shooter
- Camera mount for turret
- 6 thrust bearings for turret
- **Collapsible hood** inspired by a **knights mask**
- Camera mount hit the **flywheel**
- When hood moves there is a gap of contact
- Andymark COTS barring for V2
- Flywheel weight for **rotational inertia**



V3

Key Features:

- **Lower minimum angle for hood in order to shoot from closer distance**
- Gear moving shooter angle improved
- Added more weight to flywheel to increase inertia (1 ½ pounds)
- Bare motors for more efficiency

- Our flywheel weight helps increase our rotational inertia.
- Increased rotational inertia means that it takes less time for a wheel to get back up to speed for the next ball
- Created our own flywheel with added weight (total assembly weight ~ 1 ½ pounds)

Spindexer

- The Spindexer is used to **sort the balls** for the correct ball color. It “memorizes” what color ball is in each slot.
- The code also defines each slot's “position” in servo ticks

Drivetrain

- Contains logic for **field centric** and **robot centric*** driver controls
- Contains path for auto park*

Turret and Shooter

- Contains Logic for **transfer arm**
- Uses **Robot Pose** to calculate velocity
- Aims turret to correct goal
- Changes the angle of the hood
- Updates RGB Indicator

Intake

- Sets the power of our intake
- Stops intake when the spindexer is full
- **Reverses intake** if a ball is stuck

```
ic void setMotifPattern(char color1, char color2, char color3)
if (color1 == 'P') firstRequestedColor = Color.PURPLE;
else if (color2 == 'P') secondRequestedColor = Color.PURPLE;
else if (color3 == 'P') thirdRequestedColor = Color.PURPLE;
else if (color1 == 'G') firstRequestedColor = Color.GREEN;
else if (color2 == 'G') secondRequestedColor = Color.GREEN;
else if (color3 == 'G') thirdRequestedColor = Color.GREEN;
```

We use this function to set the motif pattern once the obelisk ID is detected. it takes in the motif pattern in order and sets it to what color needs to be shot in order.

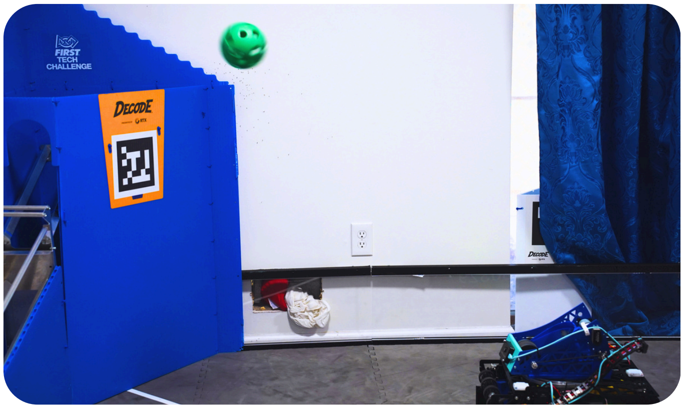
- Need to figure out what slot has the desired color
- Color for each slot is remembered based on the color sensor output
- Compare it to the desired color requested and load the ball in the outtake slot.

```
LOADGREEN:
greenBallRequested = false;
if (slot1 == Color.GREEN) {
    goToSlot( setSlotNumber: 1);
    spindexerTime.reset();
    shootColorBall = ShootColorBall.INSLOT;
} else if (slot2 == Color.GREEN) {
    goToSlot( setSlotNumber: 2);
    spindexerTime.reset();
    shootColorBall = ShootColorBall.INSLOT;
} else if (slot3 == Color.GREEN) {
    goToSlot( setSlotNumber: 3);
    spindexerTime.reset();
    shootColorBall = ShootColorBall.INSLOT;
```

Pinpoint Tracker

- Get the current heading of the turret
- Does a arctangent to figure out to the required heading to point at the goal
- Set’s the encoder position to the calculated angle for a perfect shot every time!

Field Centric - A type of driving that allows driver to control the robot relative to the field
Robot Centric - A type of driving that controls the robot based on the robot’s orientation
Auto Park - A feature we made that creates a path to go to the park zone perfectly



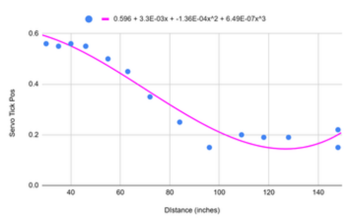
Velocity equation

$$v = c + 1538 - 16.8x + 0.264x^2 - 0.000889x^3$$

- Equation created from **Regression** of the optimal motor speed for a given shooting distance
- Adjusts **ball velocity** in real time to provide us with the most accurate shots every time
- We get the robot's position by using **dead wheel odometry,** tracked using Pedro Pathing, a localization path library
- We calculate the distance to goal in real time using the robots position
- The constant C allows driver two to adjust the velocity during a match if needed

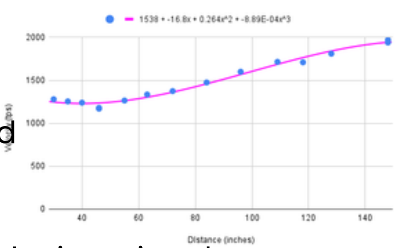
Finding optimal shooting angle

- Experimented with different shooting angles at different distance to find the most accurate shots and minimizes bounce out
- Fit a regression equation to this angle by distance to the goal (x)
- Set the shooter angle in real time to this angle based on robot location
- Add a compensation angle to correct for velocity error of the flywheel

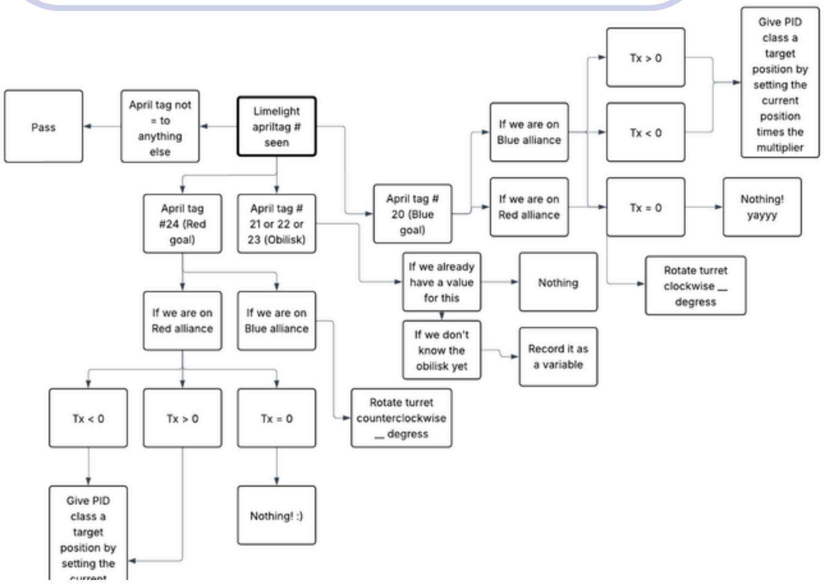


Regression of the motor speed to initial ball velocity

- We set our flywheel motor to different speeds and recorded how far the ball went from the robot



- Calculated Launch Velocity using the projectile motion equation
- Used regression to find the relationship between the ball speed and motor speed

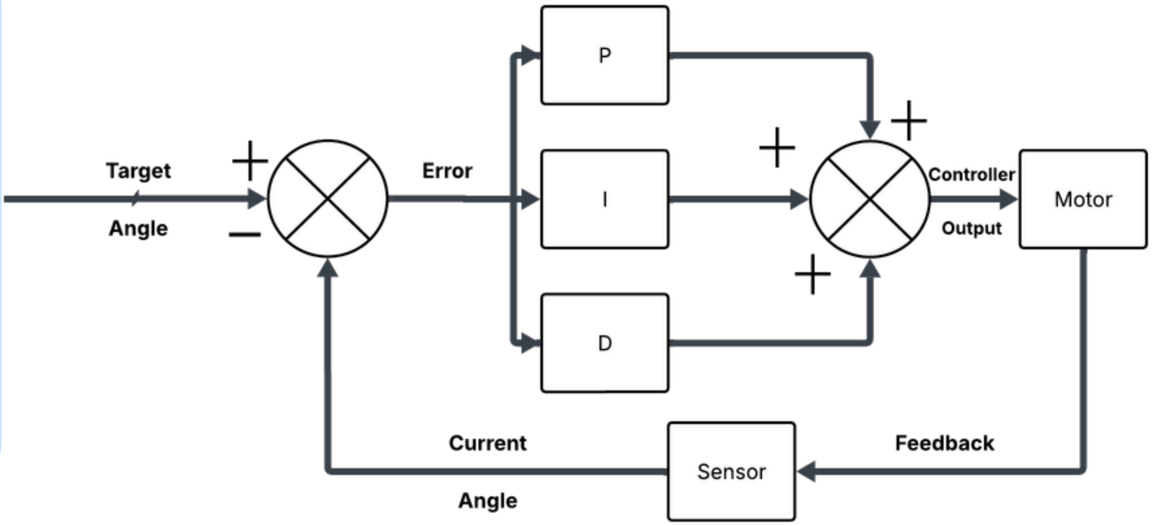


Turret Tracking State Machine

- Created a **logic map** for turret rotating logic.
- There are many tasks our turret needs to do, and mapping out our logic help us organize so that it's very efficient and easy to work with
- Represented in our code as if statements based on limelight results.

PIDs are used in various subsystems in our robot, such as:

- drivetrain (using Pedro Pathing),
- fly wheel,
- spindexer motor.



PID helps our robot get to a target position. Using the error, or the distance between the current position and target position (P), the sum of all of our errors overtime (I), and the derivative of changes in error(D). A PID changes the input based on results from the output.

Pedro Pathing is a big part of our autonomous code, it allows us to make quick paths that react to external forces

- Pedro also allows us to get the position our robot is on the field, allowing us to calculate velocity and alpha for our shooter

We use the Pedro Pathing visualizer to plan our auto paths

Software Organization

- We use a tool called GitHub to help us collaborate easily and maintain version control.
- We need to add new features but often that process will have some bugs. We create a branch which is a copy of our safe code to design new features, we have a **branch called main with code that works**
- GitHub **enhances collaboration** by letting multiple users edit code concurrently, then resolve differences during the merging process.