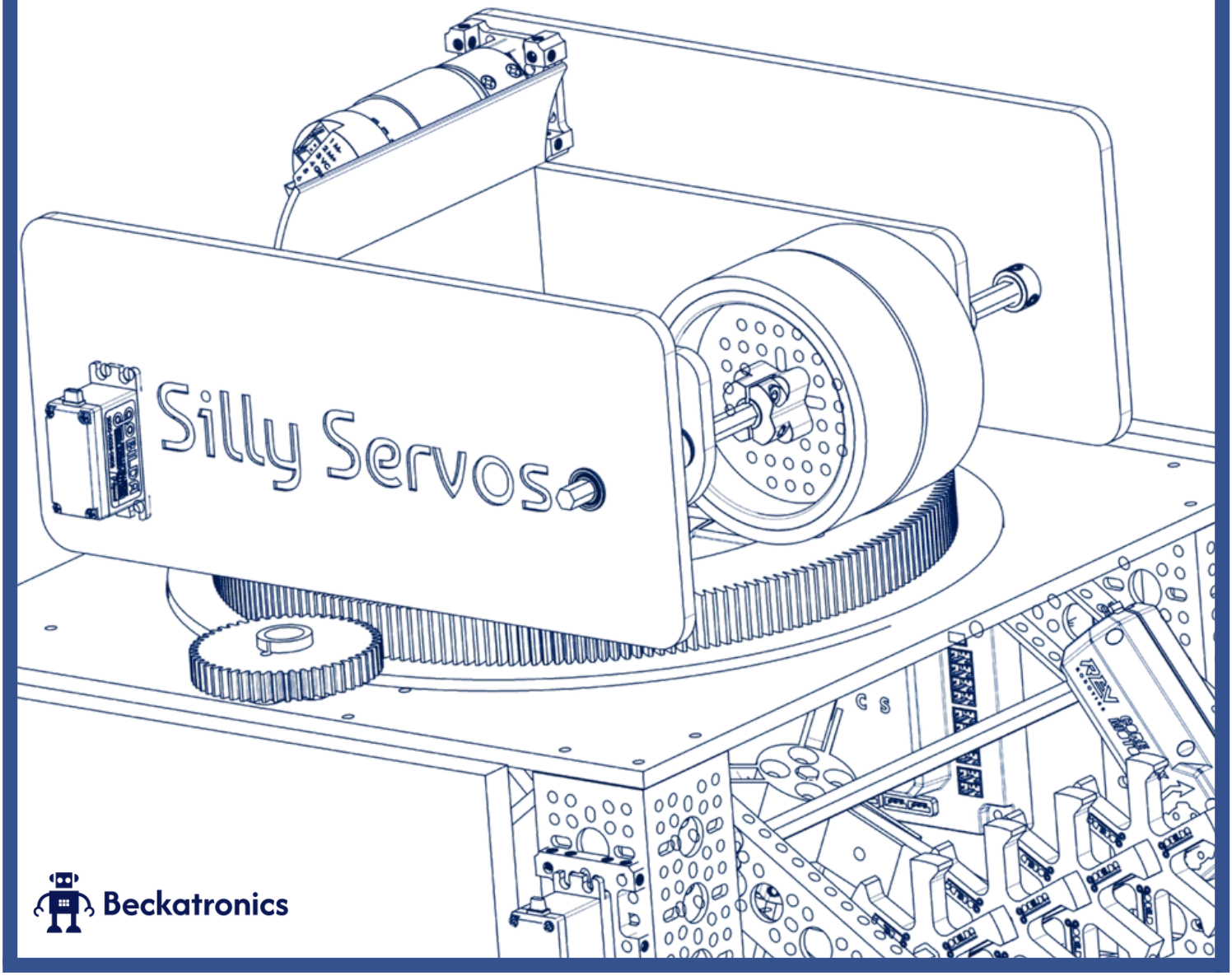
SILLY SERVOS ROBOTICS

FTC robotics team at Highland Park High School

24213

www.sillyservos.org



Beckatronics

Team Introduction

We are FTC Team #24213, Silly Servos, from Highland Park, Illinois. We started in 2023 to **promote STEM education at Highland Park High School**. We're a student-led team dedicated to designing, building, and programming competitive robots for the FIRST Tech Challenge, while actively promoting STEM education in our school and local community. Our team emphasizes engineering excellence, collaboration, and outreach. Through hands-on learning and real-world problem solving, **we strive to grow both as engineers and as leaders**.

Leadership

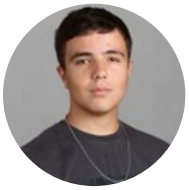
Our team is fully **student-led**, which means that all decisions are made by team members. We have **two captains** who oversee the team and agenda, as well as members who work on either programming, building, or CAD. Our **mentors** are there to manage finances, give advice, and communicate to parents. Additionally, we have structured meetings at the beginning of every year to determine who will be the captains for that year. Our captains also work to **teach new team members** everything they know to ensure we always stay a competetive team.

Recruiting

We are a **school team** with mainly school funding and a dedicated workspace. To support this team, we recruit new members at the beginning of every season via our **school's activity fair**. This involves members presenting previous robots, outlining what we do, and explaining the **core values of FIRST**.

Team Members

Jacob K: Captain & Lead Coder

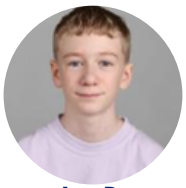


Bora K: Captain

Zachary G: Coder

Brian M: Head Designer

Abel C: CAD Designer



Jay D: Builder

Ashley P: Builder

Annie V: Builder

Elisha P: Builder

Ja' Kyra D: Builder

Our team's success is built on the foundation provided by our mentors, Mr. Mak and Ms. Edwards. Their combined years of **teaching PLTW engineering and degrees in engineering** create a culture of excellence that challenges us to push the boundaries of our capabilities.

Ms. Edwards

Ms. Edwards has been teaching for 24 years, having earned her Mechanical Engineering degree at Marquette University. She ensures our team follows a **rigorous design process**, from initial sketches to final assembly

Mr. Mak

Mr. Mak has been teaching for 15 years, with a degree in Technology Education from Illinois State University. He challenges us to treat our FTC season like a **professional engineering firm**, pushing us to meet deadlines and **conduct thorough testing** before every competition.

Our Core Values

We are defined by:

- **Gracious Professionalism**: We assisted our sister team Mighty Motors with programming and robot debugging as well as helping other teams during competitions.
- **Teamwork**: Subsystems were developed collaboratively across programming, CAD, and build teams to ensure that everyone works together toward one common goal.
- **Inclusion**: We recruit new members every year through a school-wide activity fair.
- **Learning**: Our captain created Code-A-Robot to teach Java to new and external FTC students.
- **Respect & Conflict Resolution**: Team decisions are made through structured design reviews and consensus voting to keep discussions respectful.

Code-A-Robot

Our captain, Jacob K, created **codearobot.org** over the summer as a way to teach new members of Silly Servos and other FTC teams how to program robots in Java. The site includes **80+ lessons** covering basic Java topics, Object-Oriented programming, and **FTC-tailored coding lessons**. One user of Code-A-Robot stated that **"I could put code on one of my team's robots within a week"** of starting Code-A-Robot and that after completing the course **"I now feel like I have the foundation to be a programmer."** In total, Code-A-Robot has over **480 users** across **10 countries,** with over **27,000 total views!**

Website & Sponsorship Packet

This year, we launched a website, sillyservos.org, to **showcase our robots, team progress, and achievements**. On it, we created a **sponsorship packet** that explains our team and what sponsorship options we have, helping us work towards future sponsorship and funding opportunities. The website is maintained and updated each season to reflect current robot development, outreach efforts, and possible sponsorship opportunities.

Mighty Motors

Our sister team, Mighty Motors (32690), which formed this year at Highland Park High School, has benefited greatly from **contributions from Silly Servos**. Members have assisted in code, technical difficulties, and team relations. From this, their autonomous accuracy increased from making about 33% of shots to making about 70% of shots.

Beckatronics

Beckatronics is a local startup founded by Kevin Beck, focusing on low-cost, high-durability robotic arms. As a 2021 graduate of Highland Park High School, Beck reached out to our mentors, securing a place as our **primary sponsor**.

Throughout the year, Beck has used his experience in robotics to provide us with **guidance and mentorship** each and every day. Beck also juggles volunteering as an FTA at local FIRST tournaments, making an effort **to help the community**.

At the start of the DECODE season, we performed a full **scoring and risk analysis** of the game manual, focusing on:

- ARTIFACT scoring criteria
- PATTERN scoring multipliers
- Autonomous vs TeleOp point distribution
- Endgame robot scoring
- Robot efficiency

Our observations:

- **Pattern scoring** significantly increases match points
- Controlling artifact launching **order is more important than intaking quickly**
- The ability to **score fast** is important

From this, we concluded that the robot needed to be **strategic and accurate**, not just fast.

Based on our scoring analysis, we created these strategy options:

Option 1: Controlled Pattern Sorting

Reasoning:

- Pattern bonuses
- Increases total scoring potential

Implications:

- We must control which artifact is launched

Engineering requirements:

- Selectable indexer system
- Artifact retention
- No reliance on intake order

Option 2: Optimize Cycle Time

Reasoning:

- Matches are time-constrained
- More cycles = more scoring

Implications:

- With speed can come inaccuracy

Engineering requirements:

- Continuous intake
- Low jam risk
- Fast intake to launcher transfer
- Easy to drive

Option 3: More Shot Options

Reasoning:

- Shooting from one position reduces adaptability

Implications:

- Difficult to calculate target distance while moving

Engineering requirements:

- Shoot from anywhere
- Close & far shots

On the next page, you will find the design options derived from this analysis

Engineering Decision Matrix

Strategy	Solution	Goal
Pattern Control	Spinning Indexer	Ability to select any color ball
Faster Cycles	Active Intake	<10 sec to intake an artifact
Faster Cycles	Adjustable Hood & Turret	2 distance presets
Positioning Reliability	Pedro Pathing	In correct position 80% of the time
Jam Prevention	Rotating Retention Ramp	<5% artifact ejection

Option A: Conveyor Intake

Pros:
- Simpler
- Fewer moving parts

Cons:
- No artifact selection
- Requires careful intake order

Rejected - Does not align with our strategy

Option B: Gravity Feed

Pros:
- Simple
- Lightweight

Cons:
- Poor retention at speed
- Centrifugal ejection likely
- Reduced shot control

Rejected - Low reliability, less control

Option C: Angled "Spindexer"

Pros:
- Artifact selection
- Compact
- Supports pattern strategy

Cons:
- Requires retention solution
- More software integration

Chosen - Aligns most with strategy, reliable

Design Philosophy

From our analysis, we adopted this **engineering philosophy**:
- Control > Speed
- Reliability > Complexity
- Iteration > Assumption

Every subsystem design was evaluated against 3 questions:
1. Does this improve strategic control?
2. Does this reduce failure risk?
3. Does this increase scoring potential?

If not, it was redesigned or removed

Strategic Outcome

Our final robot reflects our strategic decisions:
- Active rotating intake → fast intaking
- Rotating retention ramp → Prevents centrifugal ejection
- Spinning indexer → enables pattern control
- Adjustable hood & Turret → flexible scoring distances
- Pedro Pathing → Drive consistency

Instead of building mechanisms first and strategy second, we **derived mechanisms from our game analysis.**

Methodology

Throughout the season, we used **controlled testing** to validate design decisions. Each configuration was evaluated using timed scoring trials, shot accuracy tracking, and failure-rate observation. Trials included intake, indexing, shooting, and human player interaction, with **multiple runs averaged to reduce variability**.

Intake & Storage Comparison

Starter Bot (No Storage)

- Avg. time to score 3 balls: ~45 seconds
- Avg. shots made: 2/3 (~67% accuracy)
- Limitation: No pattern control

Conveyor-Style Storage

- Avg. time to score 3 balls: ~25 seconds
- Shot accuracy: ~80–85%
- Limitation: No artifact selection; increased jam risk at higher speeds

Spinning Indexer (Final Design)

- Avg. time to score 3 balls: ~25 seconds (10 trials)
- Shot accuracy: ~85–90%
- Artifact ejection after retention ramp: <5%
- Minimal full jams during controlled testing

The indexer matched conveyor cycle time while enabling artifact selection, increasing total scoring potential per match.

Design Testing Summary

Design	Time to score 10 points	Time to build	Failure Rate Per Cycle
Intakeless (starter bot)	1 minute	3 weeks	5%
Conveyor storage (other teams)	45 seconds	1.5 months	20%
Indexer storage (our design)	30 seconds	1.5 months	15%

*Failure rate represents jam or misalignment events per 20 controlled scoring cycles.

Iterative Improvements & Competitive Benchmarking

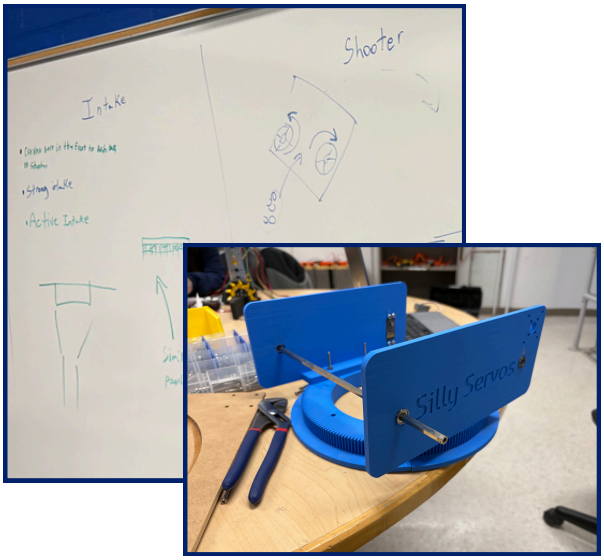
Testing also extended beyond our own robot. During meets, we actively **observed and analyzed other teams' designs**, tracking cycle times, jam frequency, structural failures, and scoring consistency. **Instead of redesigning blindly**, we used this real-world benchmarking to speed up iteration.

For example:

- We observed conveyor-based systems achieving fast cycle times but **struggling with intentional pattern control.**
- We noted **centrifugal ejection** issues in high-speed indexers and designed our rotating retention ramp to mitigate that failure mode.
- We saw **structural weaknesses** in heavily 3D-printed drivetrains and made high-stress components use stronger materials.
- We studied turret designs with fixed hoods and recognized their **limited shot adaptability**, reinforcing our decision to implement an adjustable hood.

By **combining our own testing with competitive observation**, we avoided repeating common failure modes and focused our iteration cycles on **proven improvements**. This approach allowed us to move directly toward optimized designs and avoid rediscovering known limitations.

Early Concepts

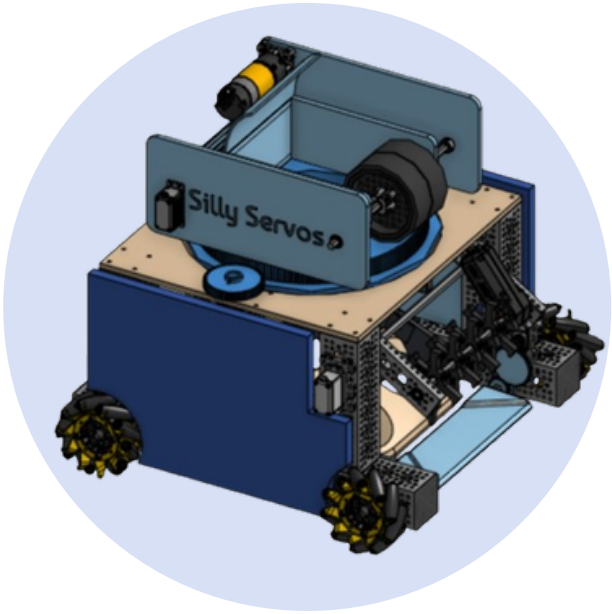


At the start of the season, we wrote and modeled any and all ideas and presented them to the group as a whole. With these, we took the **best parts from each design** and combined them as best we could into one.

Custom Parts

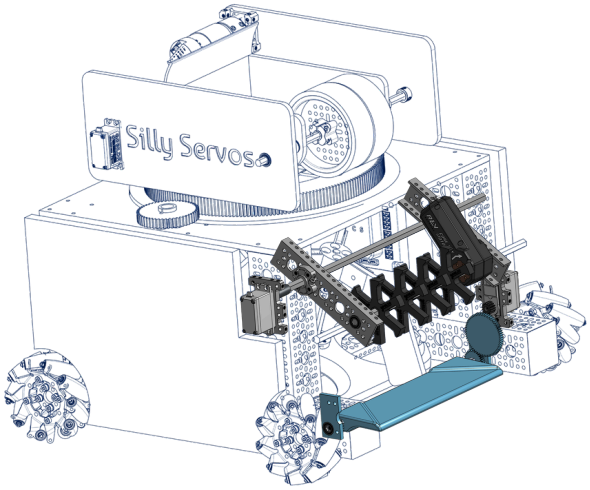
We designed a custom metal lift arm in CAD to move balls into the launching mechanism. The part was metal 3D printed by JLC, giving us a **strong and reliable component** that improved **consistency and durability** in our system.

CAD Modeling



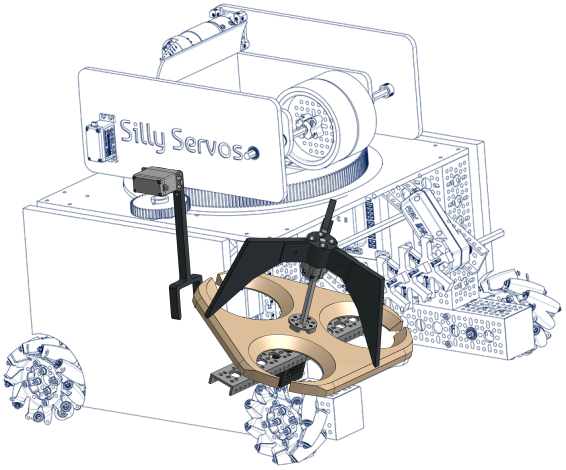
We used CAD extensively before making any large decisions or changes to our robot, testing everything we could. Utilizing **Onshape**, we modeled the entire robot and fully assembled all subsystems digitally before manufacturing, allowing us to **check fit, motion, and clearances early in the design process**. This helped us identify **potential issues** before they became physical problems, saving time, materials, and effort during the build season. **CADing our robot also made iteration faster**, as we could easily modify designs, compare different mechanisms, and optimize performance without rebuilding parts. Overall, CAD played a **critical role** in improving reliability, organization, and efficiency, resulting in a more refined and competition-ready robot.

Intake



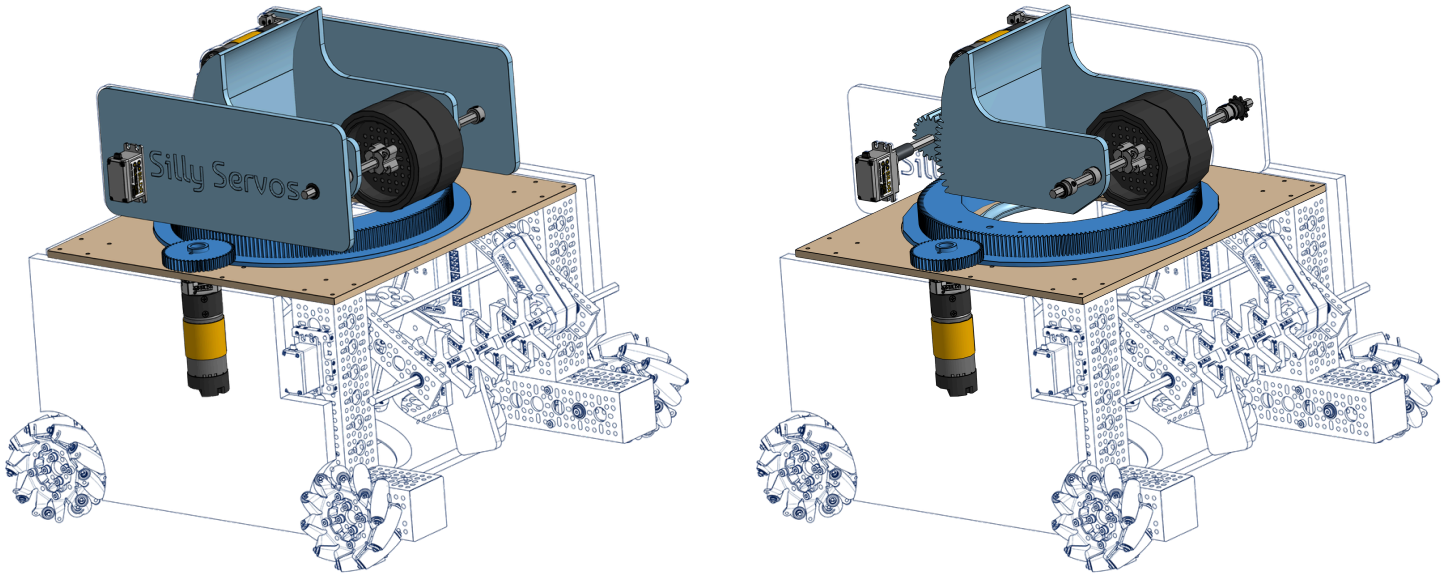
After identifying the lack of an intake system in our first robot, we knew that the new iteration needed to have an intake. **Using Onshape for iterative CAD modeling**, we developed a compact, servo-actuated **rotating intake assembly**. To minimize size, we selected a REV Core Hex motor to drive the intake roller. The intake is mounted on a servo-controlled pivot, allowing vertical articulation. **This articulation enables a scooping motion, improving intake reliability** compared to a fixed horizontal roller that relies solely on frictional pull-in. During high-speed indexer testing, we observed that **artifacts were ejected from the system due to centrifugal forces**. To combat this, **we created a rotating ramp**. When the intake is active, the ramp rotates downward to allow artifact entry into the indexer. Once an artifact is captured, the ramp retracts upward **to create a physical retention barrier.**

Indexer



Although many robots have a simple “conveyor” style connector between the intake and launcher of the robot, **we opted to instead use a spinning indexer**. This choice lets us **choose which ball we launch**, which helps us easily score pattern points where other teams would have to carefully intake the balls in the right order. Furthermore, instead of having the indexer be flat, we decided to have it **angled, sloping upwards towards the back of the robot**. We did this because it naturally brings the balls from the lower intake to a higher position as they spin around, making it easier to push them into the launcher. We also chose to use **a pivoting fork to push the balls upwards into the launcher** as unlike a complex rotating “intake” style design, the fork is far simpler to build and maintain and is far less likely to fail or break down. Additionally, we decided to make the **rotating indexer out of wood** as it allows for precise shapes while staying cheap (unlike metal) and strong (unlike 3D printed parts).

Turret



The launcher of our robot, which we call the “turret” has many mechanisms that make it easier to make shots from various distances. This launcher has a **rotating hood** (spun by the gray servo on the left) which lets us **vary what angle we launch from**. Additionally, the entire launcher mechanism (everything above the tan square), **can pivot around the robot to aim in various directions, letting the robot automatically aim at the goal**. This pivoting motion is achieved with the yellow jacket motor spinning the small blue gear, thus rotating the large blue gear which the entire launcher sits upon. Although many other teams decided to have a small section of the robot rotate, usually just the hood and flywheel, we instead decided to have a far larger section of the robot rotate. We did this because it lets us **spread out the various mechanisms of the launcher**, namely the flywheel and ability to rotate the hood to various positions. This gave us more room for these mechanisms, **making it easier to design and iterate on** without having to worry about the tight tolerances and small parts that would come from a smaller turret design. Moreover, we decided to have a **rotating hood as it lets us more accurately vary the distance we shoot from**. If we were to have a constant position hood and only varied the speed of the flywheel, we would be constrained in viable shooting positions, as very close or far distances would have a far too high and low hood angle respectively. This would make the tolerances to make these shots far tighter, thus making it more likely that we would miss the shot.

Odometry

Early drivetrain testing revealed that relying solely on motor encoders resulted in **significant positional drift**. Wheel slip and acceleration changes caused **cumulative error that reduced shot accuracy and autonomous consistency.** To address this, **we integrated goBILDA dead-wheel odometry pods** to independently measure robot translation and rotation. Unlike drive motor encoders, dead wheels are mechanically isolated from drivetrain slip, which gives a more accurate measurement of true robot movement. With odometry integrated into our pathing system, **rotational error** was significantly reduced, autonomous **shot alignment** became more consistent, and **path repeatability improved** across various trials.

Camera

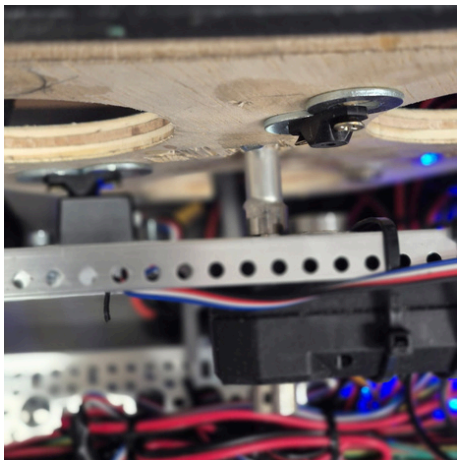
We designed a **vision system using AprilTag detection to provide field-relative awareness**. While full integration and tuning were not completed before competition, the majority of the detection and control code was developed and tested independently.

The camera system was designed to serve three engineering purposes:

1. **Goal Localization**
   a. AprilTag pose estimation provides real-time bearing and distance (Δx) to the target. This data feeds into turret PD alignment, kinematic launch calculations, and automatic distance-based hood adjustment
2. **Motif Pattern Detection in Autonomous**
   a. At the start of autonomous, the robot scans the field motif to determine scoring configuration and select the correct artifact to launch.
3. **Vision provides an external field reference to complement odometry, reducing accumulated error.**

Magnet Sensor

We used a **magnet sensor in the indexer to make it rotate more accurately**. This works by **telling the indexer exactly where it is each time it passes a magnet**. Normally encoders would be used to keep track of the motor's position, but over time **encoders accumulate error**, while a magnet will always be in the same place, meaning **it can't accumulate error like an encoder**. This prevents our indexer from getting stuck and **ensures that it is always properly aligned.**



Magnet sensor underneath the indexer

Color-Based Indexing

To enable **controlled pattern scoring**, we implemented an **automated color-detection** subsystem integrated directly with our spinning indexer. Rather than relying on intake order, the robot **actively scans each launch position** and builds an **internal model of stored artifacts.**

System Architecture

We used a REV NormalizedColorSensor mounted at the indexer inspection point. The sensor reads normalized RGB values, which we convert to HSV using:

```
Color.colorToHSV(colors.toColor(),
hsvOut);
```

HSV was chosen over raw RGB because hue is more **stable under varying lighting** and better isolates color type.

State Machine

The system operates as a structured state machine:

IDLE → MOVE → SETTLE → SAMPLE → (BLACK_NUDGE) → DONE

For each of the three launch slots:

1. The indexer **rotates to a known launch position.**
2. The system waits for motion to stop (!indexer.isMoving()).
3. **After a controlled settle delay, sampling begins.**
4. Multiple HSV samples are collected and averaged.
5. The **averaged color is classified.**
6. The result is **stored in an internal array**

This array becomes the robot's real-time inventory model.

Hole Detection

Early testing **revealed that ball holes could be falsely detected as empty**. To address this, we implemented a retry-nudge mechanism:

- **If a slot reads black** (empty) repeatedly,
- The **indexer nudges** a small number of ticks (nudgeTicks)
- **The system re-settles and re-samples**
- Up to maxBlackRetries = 3

This **prevents false negatives** caused by slight misalignment

Strategic Impact

By **integrating color detection** with indexer homing, the robot **enables intentional pattern scoring** and removes dependence on intake order.

Pedro Pathing

We **implemented Pedro Pathing to improve our autonomous routines** and driver-controlled performance. In autonomous, **Pedro Pathing allowed us to generate smooth, continuous trajectories** instead of simple point-to-point movements. This resulted in more **accurate positioning and reduced slop.** During teleop, the same motion framework enabled smoother acceleration and more predictable handling.

Subsystem Architecture

To improve **reliability, scalability, and maintainability**, we structured our entire codebase using a subsystem architecture. **Each major mechanical** system — intake, indexer, turret, color detection, and drivetrain — **operates as an independent Java class** responsible for its **own hardware control and internal logic.**

For example:

- The IndexerSubsystem handles motor control and slot positioning.
- The CheckPatternSubsystem runs a scanning process without blocking the main loop.
- The Turret subsystem manages aiming and hood positioning.

Indexer Homing

To ensure **repeatable and accurate indexing**, our robot uses a **three-magnet homing system integrated into the indexer mechanism**. Each launch position has a dedicated magnet mounted beneath the triangular indexer, and a hall-effect sensor detects when a magnet passes. We found that **encoder counts can drift** due to backlash and mechanical slop in the direct-drive Core Hex motor. Instead of relying solely on encoder counts, we compute the center of each magnet by measuring both the rising and falling edges of the sensor signal. **The system then dynamically adjusts a phase offset** so that the virtual indexer position is snapped precisely to the known launch phase. This approach **eliminates cumulative encoder error** and ensures that every launch position is physically aligned to the same absolute mechanical reference each time, **resulting in consistent ball positioning** and improved shot reliability throughout a match.

Agentic Engineering

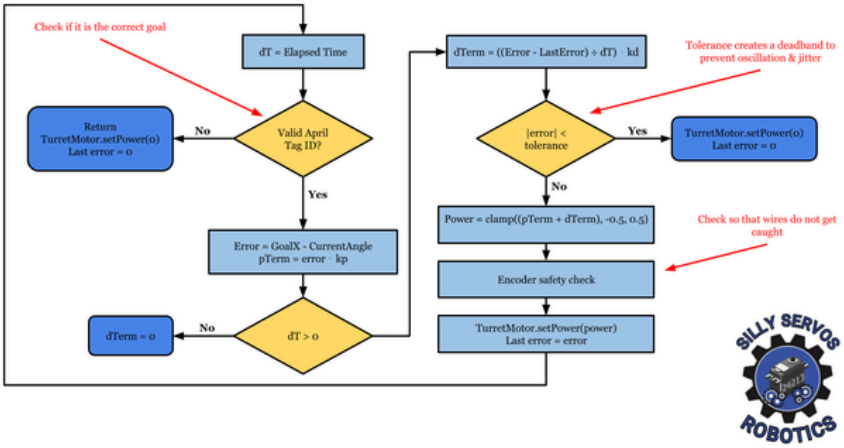
This season, we **incorporated agentic engineering tools** such as GitHub Copilot and ChatGPT to enhance our **software development process**. Rather than using AI to write our entire codebase, we used it strategically as a tool to **generate initial drafts, get feedback, debug logic, and brainstorm**. All major architectural decisions were carefully planned and reviewed by our team before implementation. **AI assisted with efficiency, allowing us to iterate faster** and focus more time on testing and tuning. By combining intentional engineering design with AI tools, **we significantly improved our development speed and workflow organization.**

Turret Aiming

This season, we designed a **vision-based turret auto-alignment system using a Proportional–Derivative (PD) control loop**. The goal was to allow the turret to **automatically rotate toward an AprilTag target** using real-time bearing data from our camera. The controller calculates angular error (goal − current angle), applies proportional control to drive the turret toward the target, and uses a **derivative term to dampen motion and prevent overshoot**. A tolerance band (deadband) stops the motor once alignment is within an acceptable range, reducing oscillation and jitter. **We also designed the system to include safety extensions such as encoder-based rotation limits to prevent cable damage.**

Although we fully developed the control architecture, flowchart, and implementation plan, time constraints prevented full integration and tuning on the competition robot. However, this system demonstrates closed-loop control theory, modular subsystem architecture, and real-world considerations like derivative spikes, loop timing, and mechanical safety.

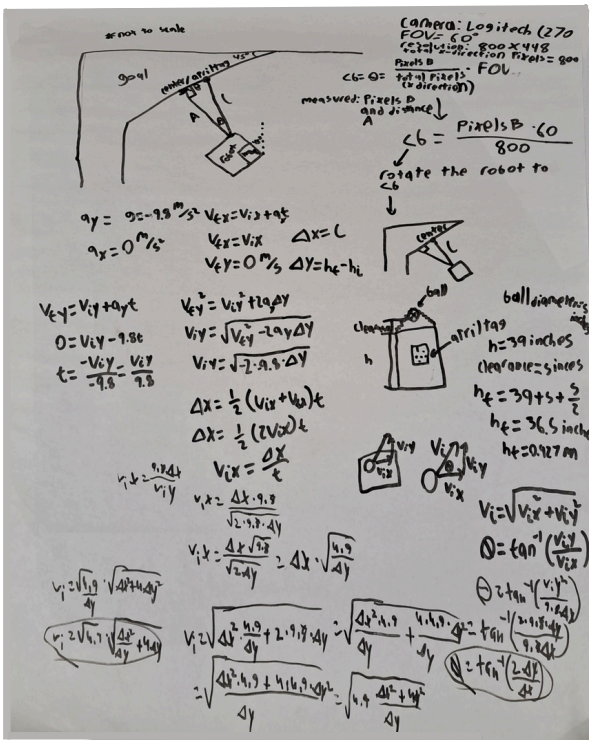
Turret PD Loop



Version Control & Git

This season, we significantly improved our software workflow by **fully adopting GitHub for version control and code management**. In past years, our codebase was often disorganized, with files stored locally and changes difficult to track. By using GitHub, we were able to **manage branches, document updates, and maintain a structured repository**. Additionally, we leveraged **GitHub Codespaces**, allowing team members to work directly in a cloud-based development environment. This was especially useful for students **using school-issued Chromebooks, which typically cannot run the full FTC development environment locally.**

Calculations



These following equations were derived to model projectile motion and validate feasible launch distances for hood preset selection:

launch angle = $tan^{-1}(\frac{2\Delta y}{\Delta x})$

launch velocity = $\sqrt{4.9} \cdot \sqrt{\frac{\Delta x^2}{\Delta y^2} + 4\Delta y}$

We implemented the kinematics shown on the left to **calculate the exact launch velocity and launch angle**. It uses primarily kinematics along with some trigonometry. We leveraged the **following basic kinematic equations** to derive our final equations:

- $V_f = V_i + at$
- $V_f^2 = V_i^2 + 2a\Delta x$
- $\Delta x = \frac{1}{2}(V_i + V_f)t$

Our equations take in several variables (all in SI units):

- Δy is the height from the center of the launcher to the top of the goal with some clearance(constant).
- Δx is the horizontal distance from the center of the launcher to the apriltag(x and z directions). We can measure this using our camera.

While full integration was not completed before competition, the modeling informed our hood angle presets and validated shot feasibility.