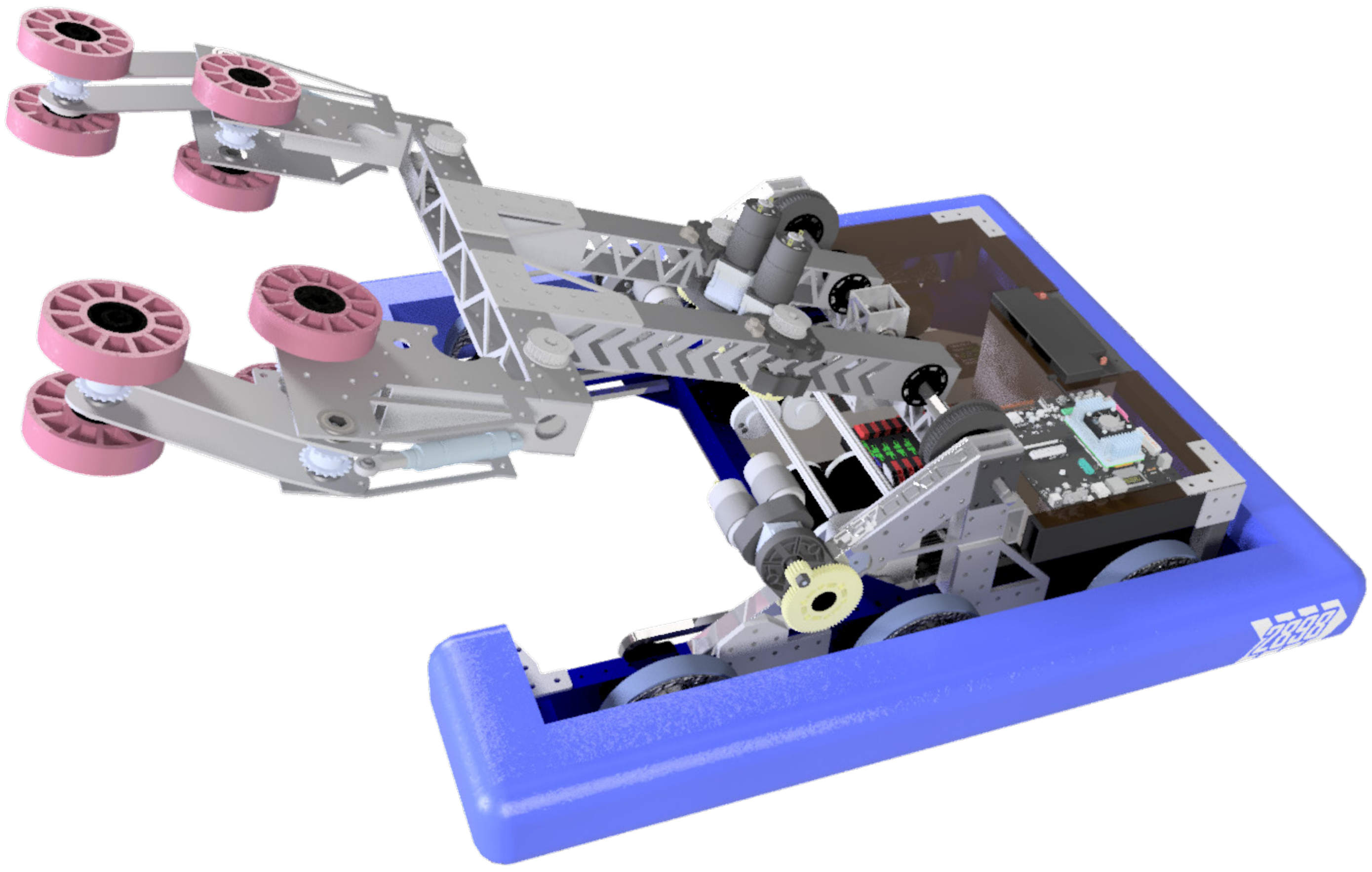
Flying Hedgehogs

School of Science and Technology • Health and Science School • BPS Robotics



2018 Technical Binder

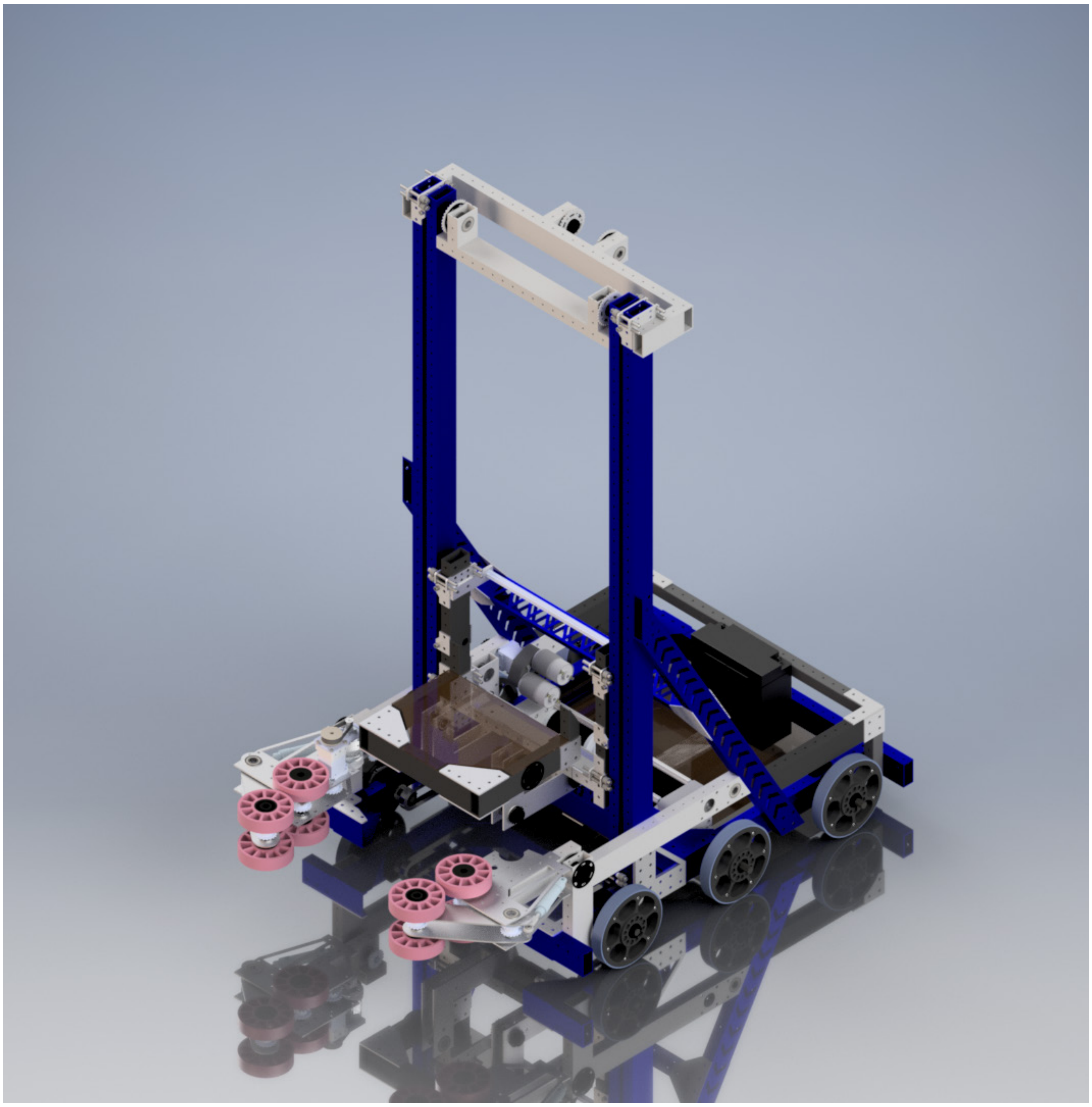
Contents

2898's Process

As a team, we are student led and mentor advised. We strive to elevate students to their highest level through leadership and teamwork.

One hundred percent of design, programming, manufacture, and wiring is proudly student led.

Robot v1


Not the one in front of you

Strategy

Robot v1

Overview

While a switch-only robot was appealing, we were hesitant to accept the possibility of a switch-only alliance in quals. We decided to try to build a robot able to primarily place cubes in the scale, but that could also effectively play roles involving the exchange and switch, allowing us to be a versatile alliance partner at any event.

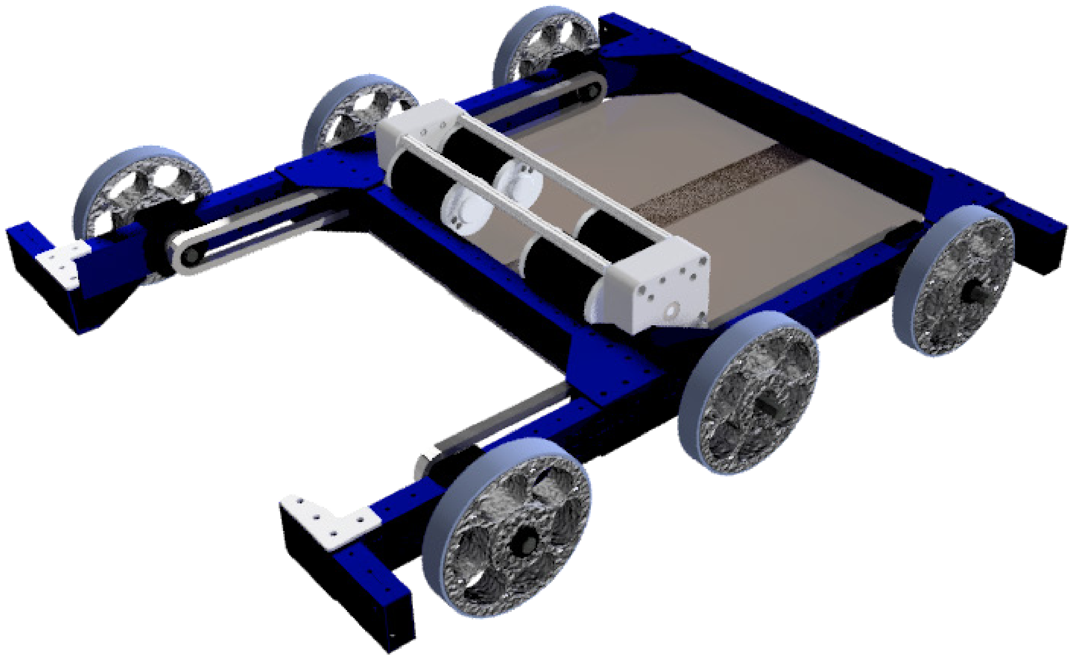
Game Strategy

We decided that the ability to complete the Auto quest was essential to the success of our robot, but the autonomous contestance of the scale would be just as important.A quick, light single speed drivetrain, optimized center of gravity, and a seperate cube lift and intake allowed a theoretical two or more cube autonomous.

While we did not decide to include a climber or set of ramps, we opted to use large, especially high-grip 6” Nitrile wheels in order to allow our robot to drive onto near any set of ramps provided.

Design

Robot v1 — Drivebase



A quick and versatile drivebase was needed. Our traditional "west coast drive" style drivetrain was chosen, but with 6" wheels to aid maneuverability.

— 6 wheel, 4 CIM drivetrain

— 6" Nitrile wheels for additional traction and ramp traversal ability

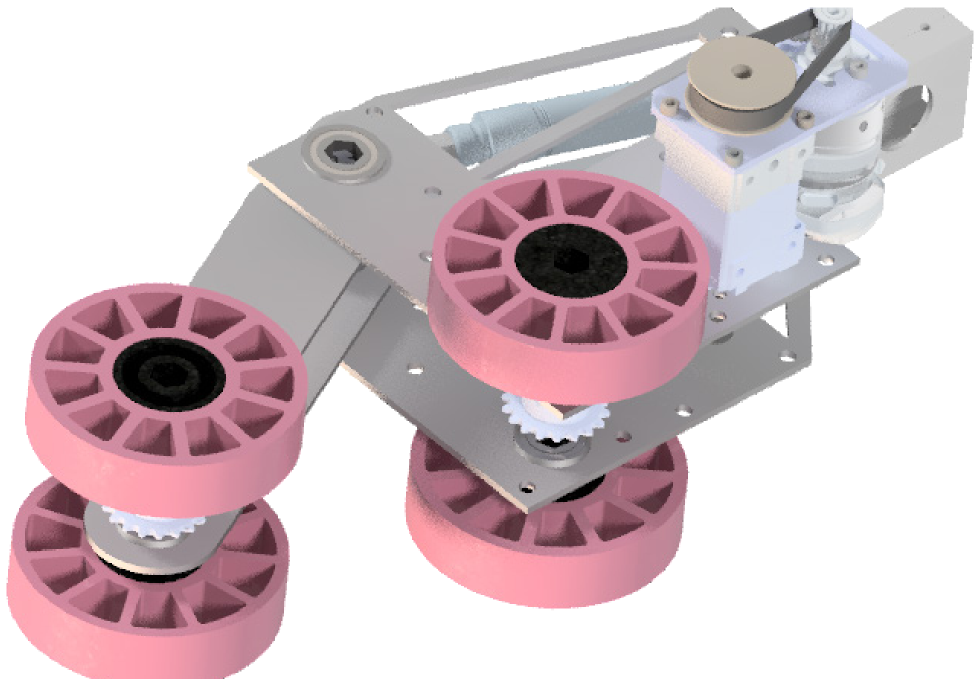
— Cantilevered "West Coast" design increases repairability

— 14.9 ft/sec top speed optimized for sprint distances found in POWER UP

— Sliding bearing blocks allow chain tensioning

Design

Robot v1 — Intake



Offcenter pivoting wheels allow for quick, painless intaking in most cube orientations

— "1114" style

— 0.5" Compression

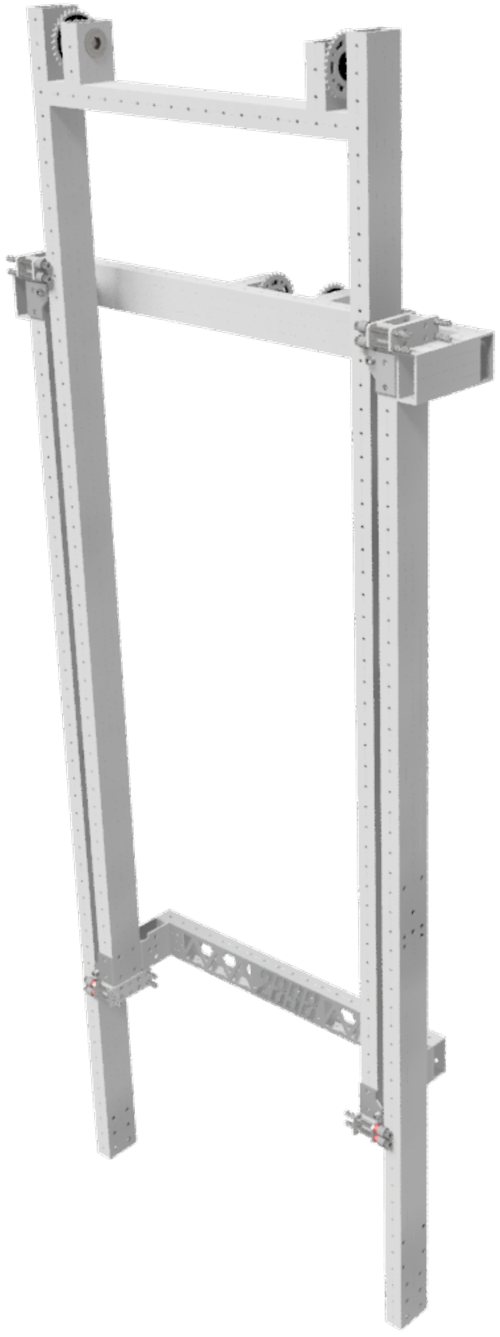
— Can intake any 13" alignment + diagonal

— 775pro in flipped 10:1 gearbox to optimize packaging

— 32.3 ft/s surface speed

Design

Robot v1 — Elevator



— Combination lift
- — Continuous first stage
- — Cascading second stage

— 2x1 aluminum with custom linear slides

— Driven by 2x 775pro motors, 50:1 reduction

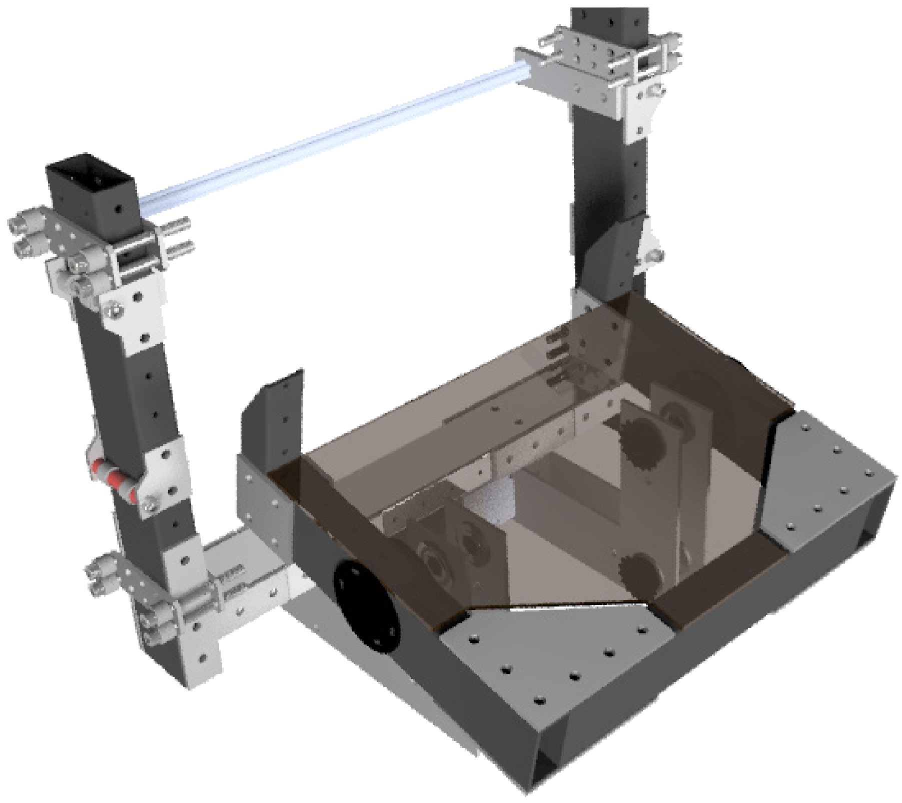
— Lift time of 0.75s

— Constant force springs to help holding torque

A combination lift would provide quick lift times with the prospect of adding a climber in the future

Design

Robot v1 — Manipulator



A light manipulator helped aid lift times and holding torque requirements

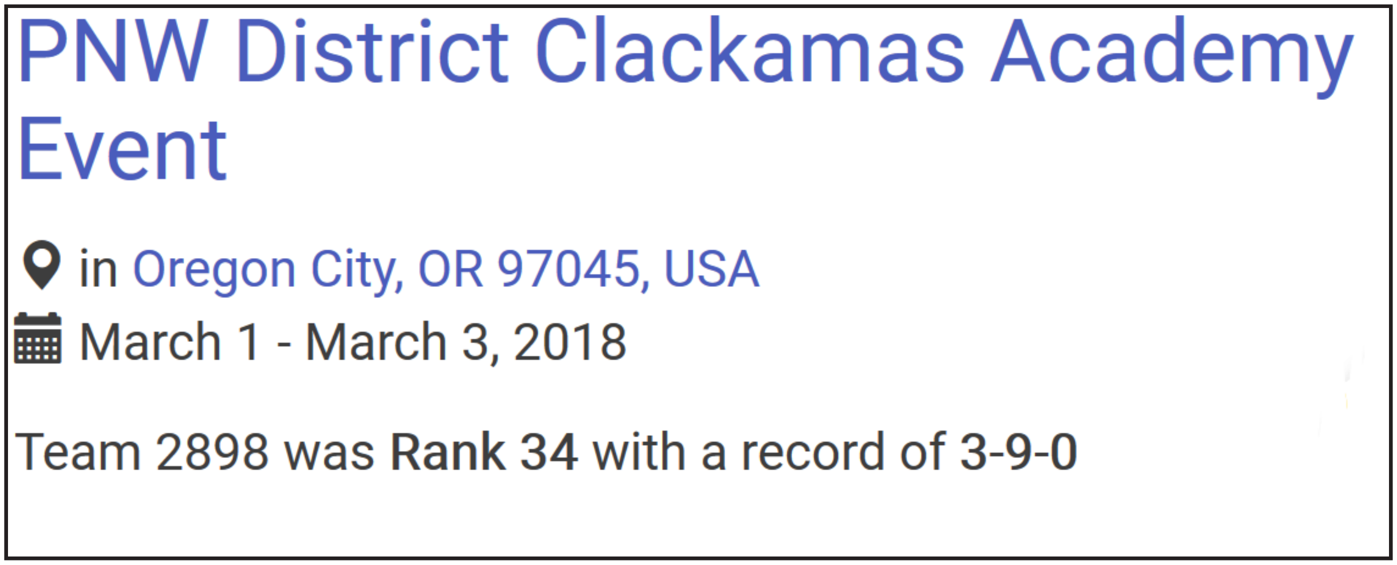
— Side+Backing Polycarbonate not shown

— Pivoting bucket

— Simple 775pro pivot

Results

Robot v1



PNW District Clackamas Academy Event

in Oregon City, OR 97045, USA

March 1 - March 3, 2018

Team 2898 was **Rank 34** with a record of **3-9-0**

Fig 1. Oof

— Massive elevator issues

— Max height impacted due to poor design of cascading stage

— Manipulator misalignment

— Intake misalignment

— Sensor failure

Two Choices

Choice #1

Jury rig elevator repair, redesign manipulator.

Pros:

— Easy
— Safe
— Cheap

Cons:

— Not much able to be done in 6h unbag

— Still many fundimental issues

Choice #2

Redesign for switch only, implement on practice chassis.

Pros:

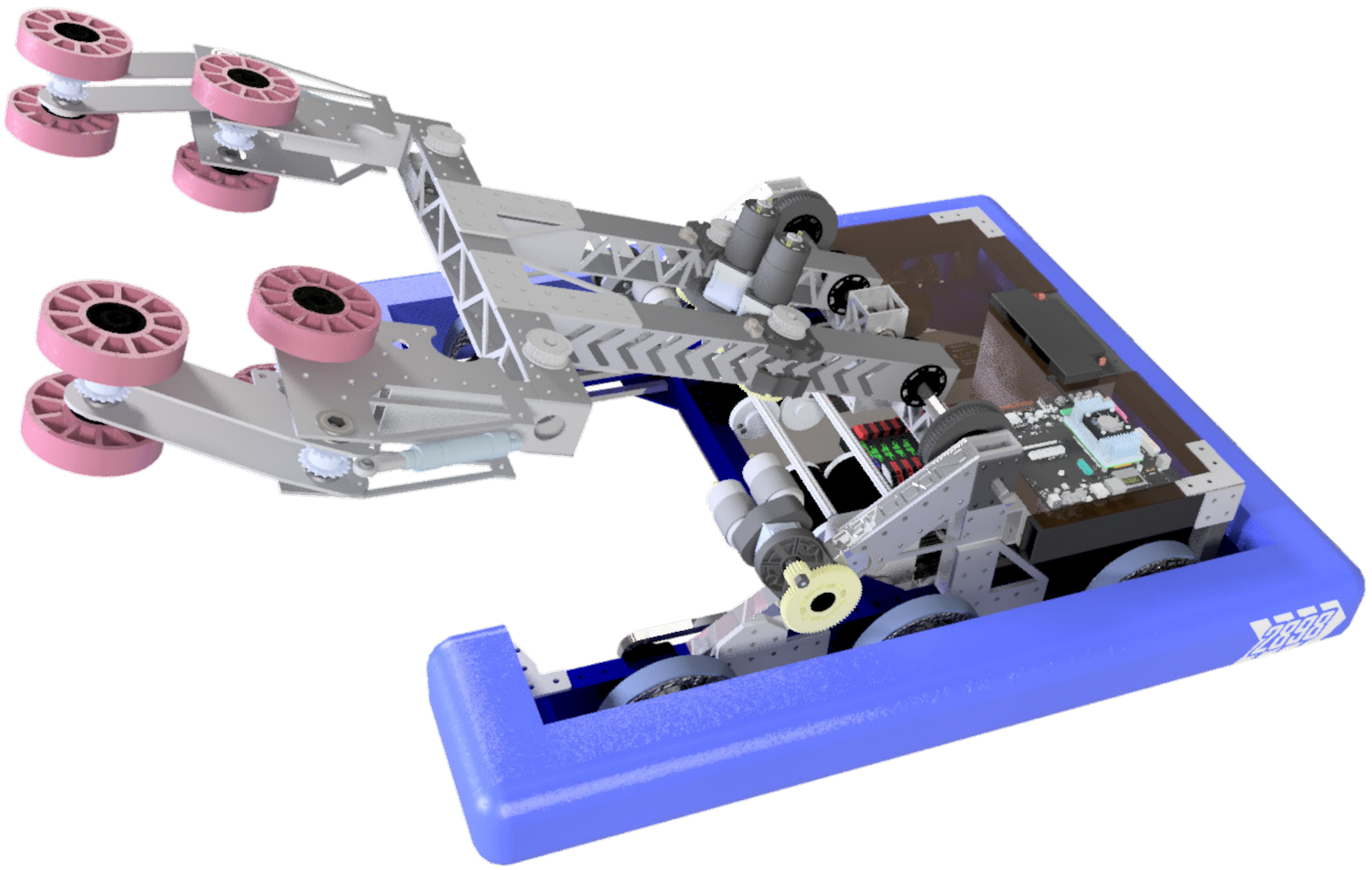
— Can build off other designs
— Intakes, drive base already made
— Practice chassis to implement
— Filling niche role, largest chance of winning event
— Time to test autonomous, new sensors+methods

Cons:

— Extremely risky, not much time

— Losing ability to play scale

Robot v2



Switch+Exchange specialist

Strategy

Robot v2

Overview

A light, fast switch-only robot can be played much differently than a top-heavy scale robot. V2 was designed to be fast, light, and robust, meshing with our aggressive, relentless driving style.

Game Strategy

As always, the Auto quest is even more necessary. V2 posesses the ability to place one or more cubes in the switch during autonomous. With simple partner autos, we can reliably score a ranking point during auto.

Our sensor and camera suite lends towards an aggressive strategy, putting pressure on the opposing switch. However, we remain fully capable of effectively defending our switch or placing cubes in the exchange/vault.

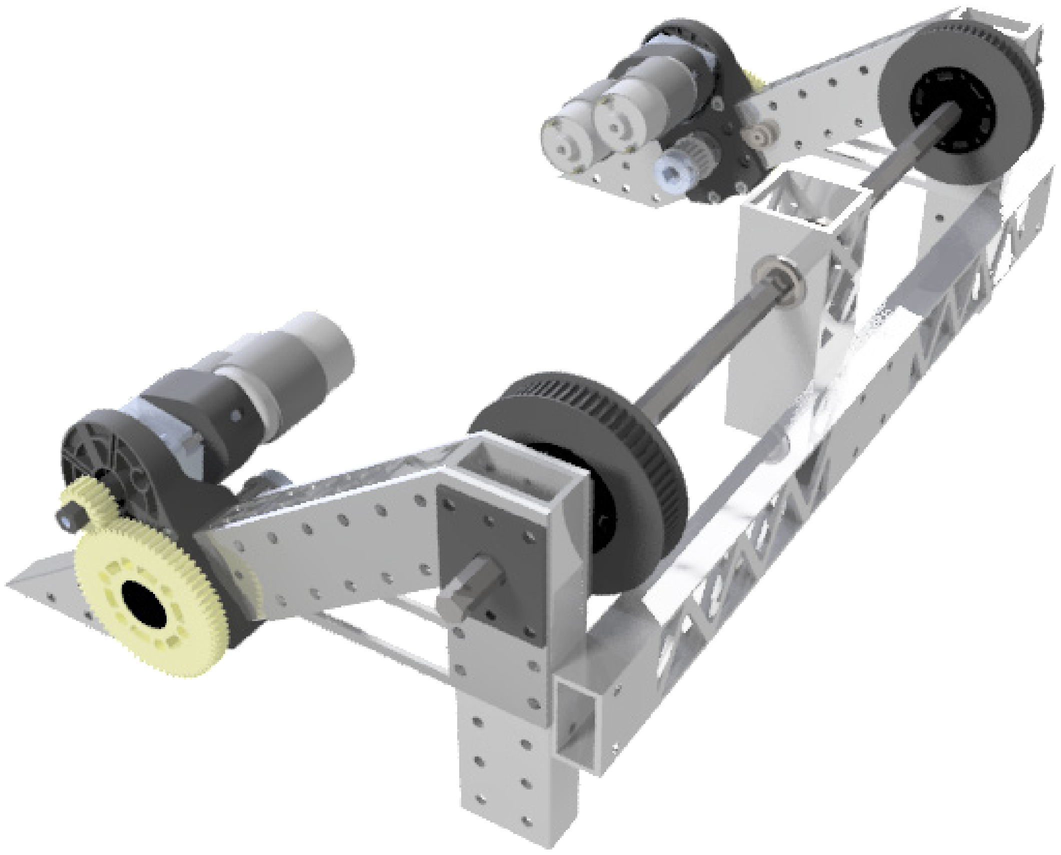
Manufacture strategy

For us to perform well on the field, we needed to get the robot built first. We relied heavy on newfound sponsors and resources, including the use of CNC milling and routing to enable the use of precision timing belts and large, highly precise assemblies, as well as the pocketing and weight optimization needed to slip under the 30lb witholding allowance.

The arm was assembled out of bag, tested on a practice chassis, and then added to the main robot during the 6h unbag time. The intakes and drivebase was retained from the main robot.

Design

Robot v2 — Arm Pivot

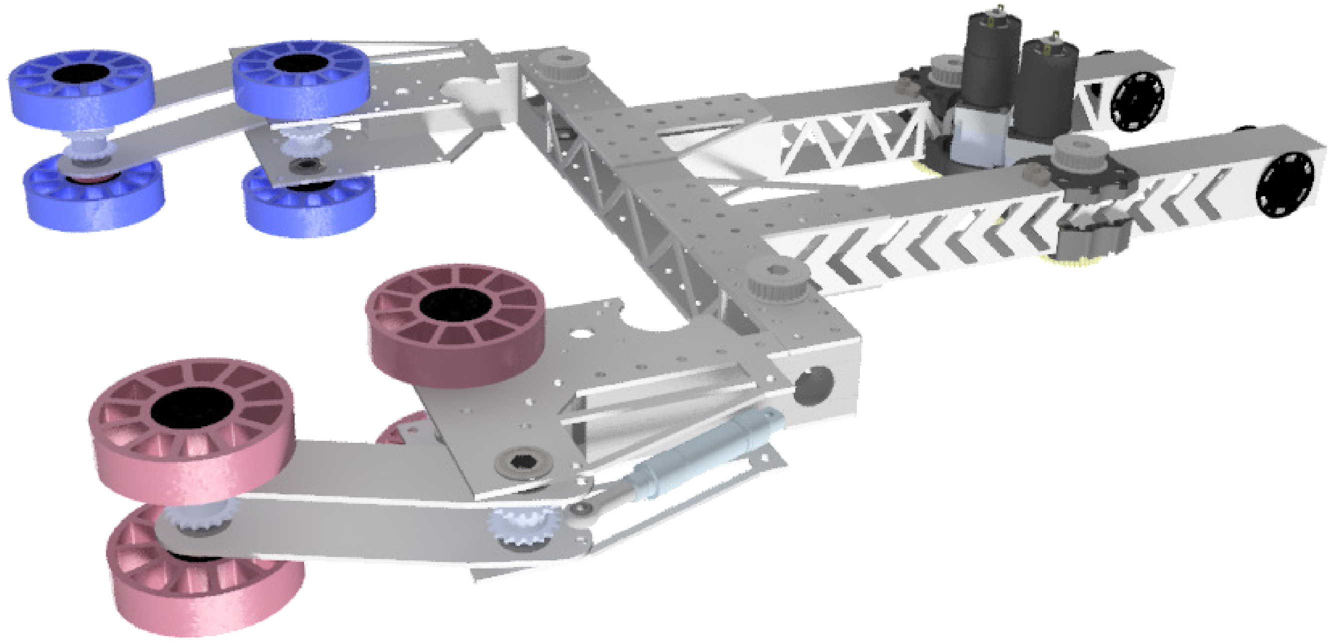


"Do you really have to engrave '2898' into everything?"
— A Concerned Mentor

— Quad 775
— ~1:96 reduction
    — 1:10 Versaplanetary
    — 18:66 Gearing
    — 24:60 Belt reduction
— Absolute + Relative encoder
    — Removes need to zero
— Sliding gearboxes allow belt tensioning

Design

Robot v2 — Arm



"Seriously, more *chevrons?"*
— A Concerned Mentor

— Intakes from Robot v1
— Motors mounted to optimize CoG
— 120 degree range of motion
  — Can place in switch from behind
— Slightly different durometer intake wheels found to ease alignment
— Cool LEDs

Weight Optimization

— Strict 30 lb witholding allowance

— Needed to optimize for weight while maintaining strength

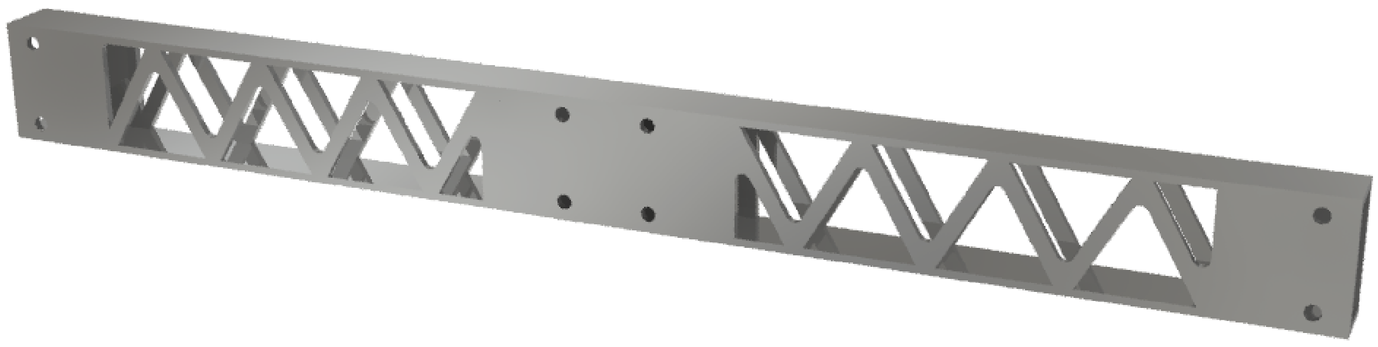
— Autodesk Inventor shape generator for weight reduction proposals

— Autodesk Nastran FEA for strength and design verification

Example:

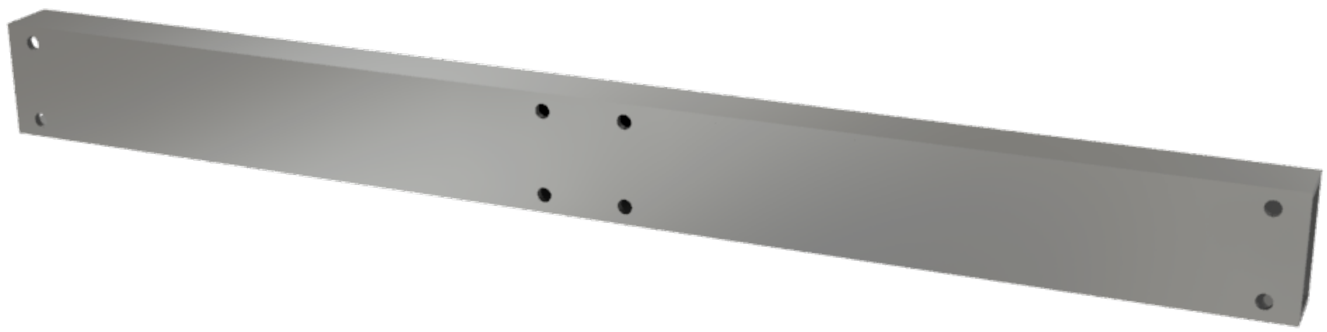
Arm pivot crossbar

Final Part:

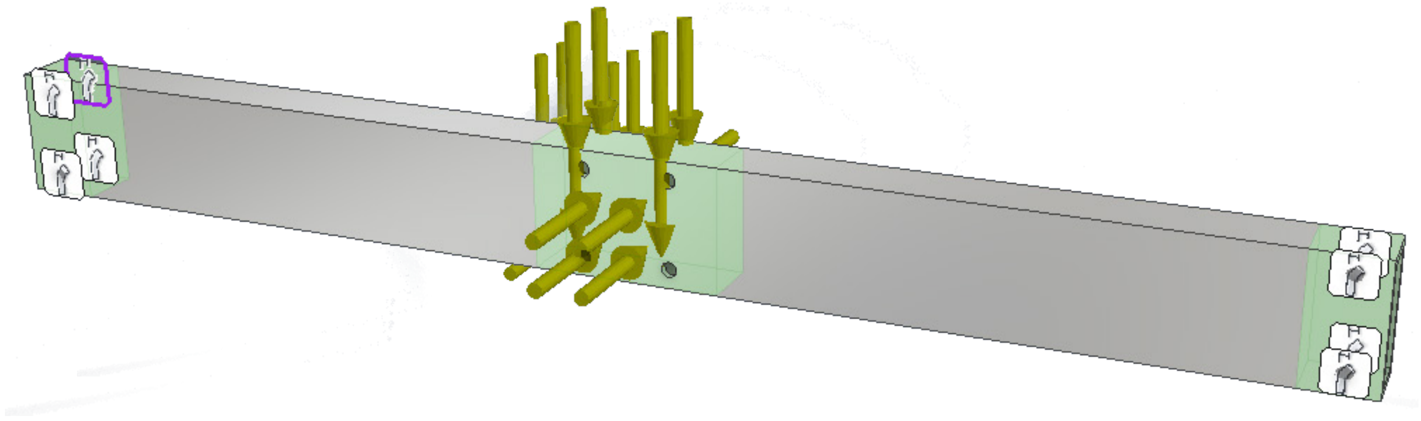


Weight Optim., Cont.

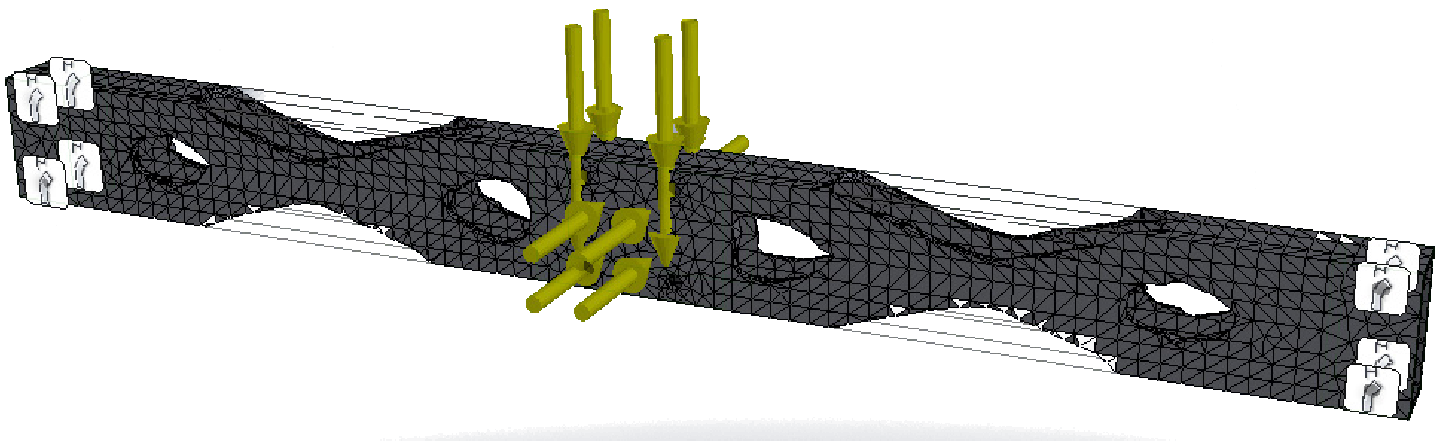
Base part design



Load definition

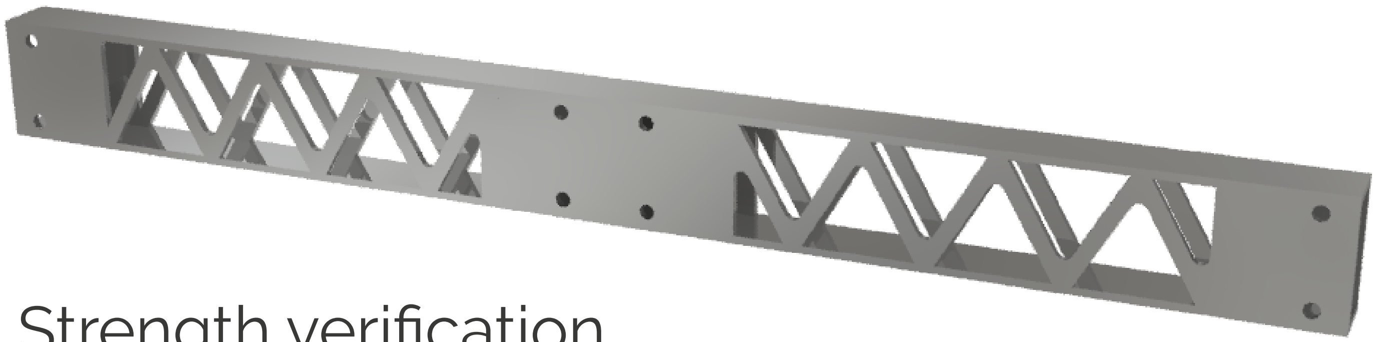


Shape generator

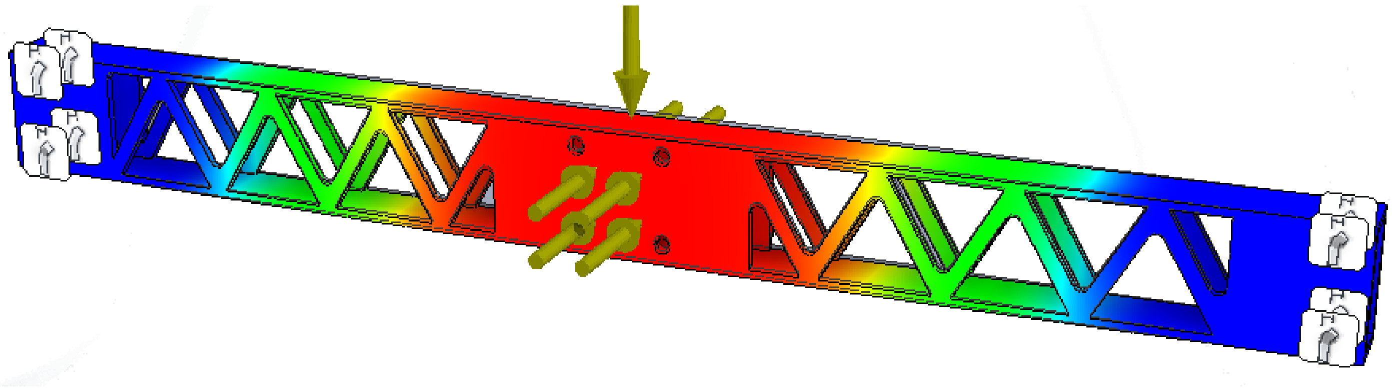


Weight Optim., Cont.

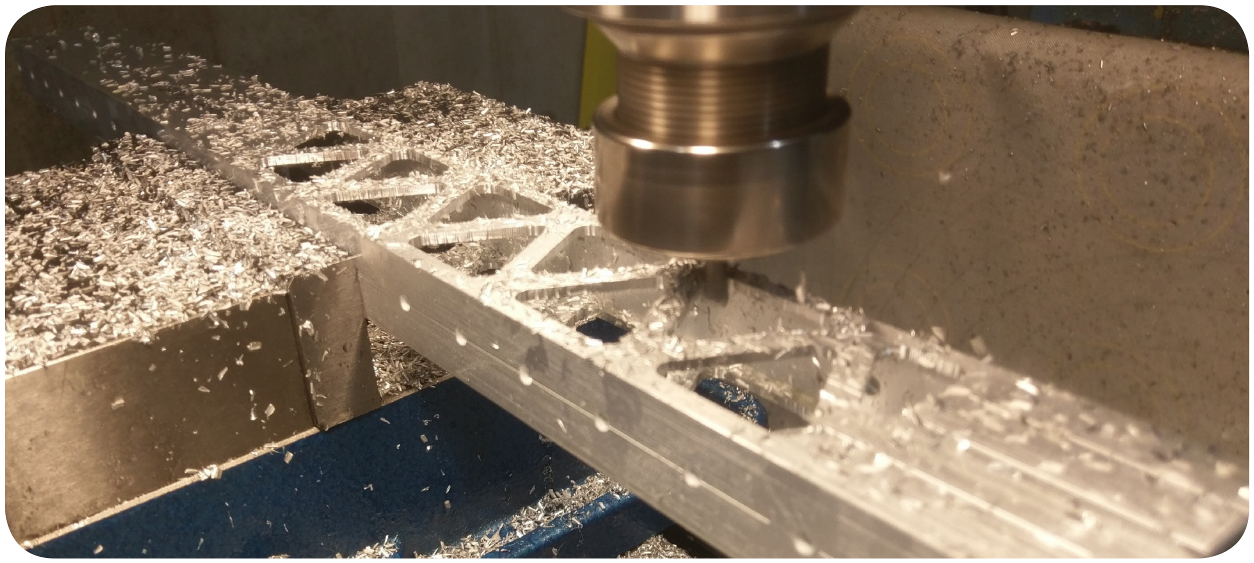
Pocketing



Strength verification



Manufacture



Manufacture

— Major tolerance issues with all manual machining in v1

— v2 is 100% CNC manufactured

Control & Software

Team 2898 is fortunate enough to have the ability and resources to cultivate a strong, adept software team. We're driven to continue pushing the boundaries and raising the bar of programming in FIRST, whether it be incorporating Deep Neural Networks into autonomous routines or adding 3D cameras to better estimate robot position.

Included is a selection of notable robot and software features.

Control & Software

Hedgehog Engine

The result of hundreds of person-hours of development, Hedgehog Engine is the core library and framework powering Team 2898's and Team 1510's robots. Incorporating advanced subsystem management, motion control algorithms, and an extensive mathematics library, Hedgehog Engine allows us to propel our software to the next level with ease.

Kotlin Core

In our quest to bring our code to the next level of functionality and maintainability, we adopted the Kotlin programming language. Originally created by IDE developer Jetbrains as a JVM (Java compatible) scripting language for internal tools, Kotlin's popularity in the development community is taking off. Its intrinsic null safety, conciseness, and incredibly powerful lambda and coroutine systems help elevate new members to our level of code faster than ever before.

Asynchronous Programming

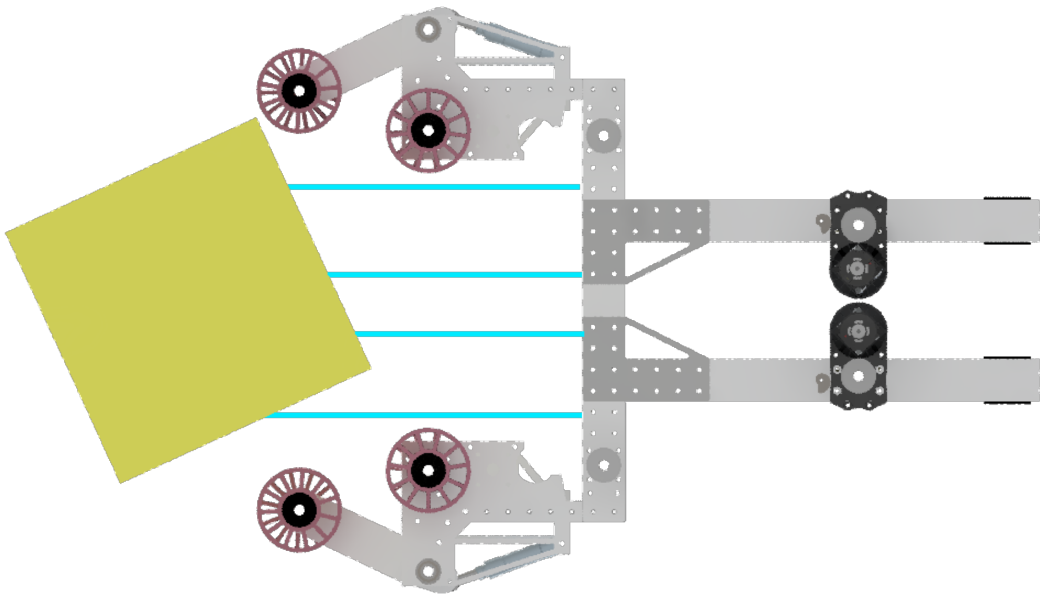
Robots are complex, with many things happening at the same time. So why subscribe to the paradigm of synchronous, lock-step code? Our codebase embraces Kotlin's lightweight coroutine system, allowing for near zero-overhead software scheduled threads, allowing us to isolate subsystems, spawn off ephemeral tasks, and create advanced control loops, all easier than ever before.

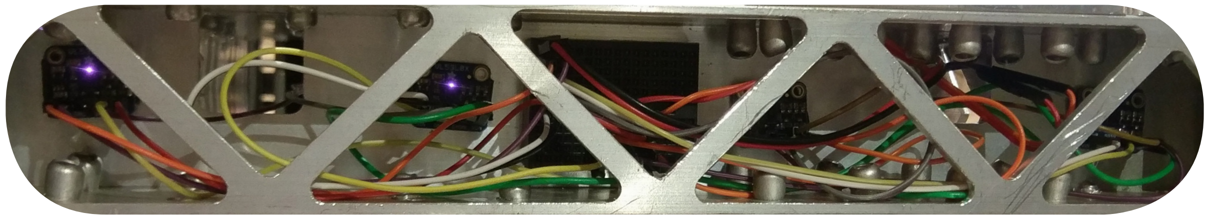
Reactive diagnostics

With many sensors and motors comes many points of failure. Upon power on, the initialization of autonomous mode, and the start of teleop, our code runs a thorough self test, checking subsystems for nominal functionality, ensuring optimal network health and CPU load, and more. If any discrepancies are found, corrective actions are made, from enabling fallback autonomous modes to disabling entire subsystems to prevent damage to the robot.

Control & Software

LIDAR Intake Assist



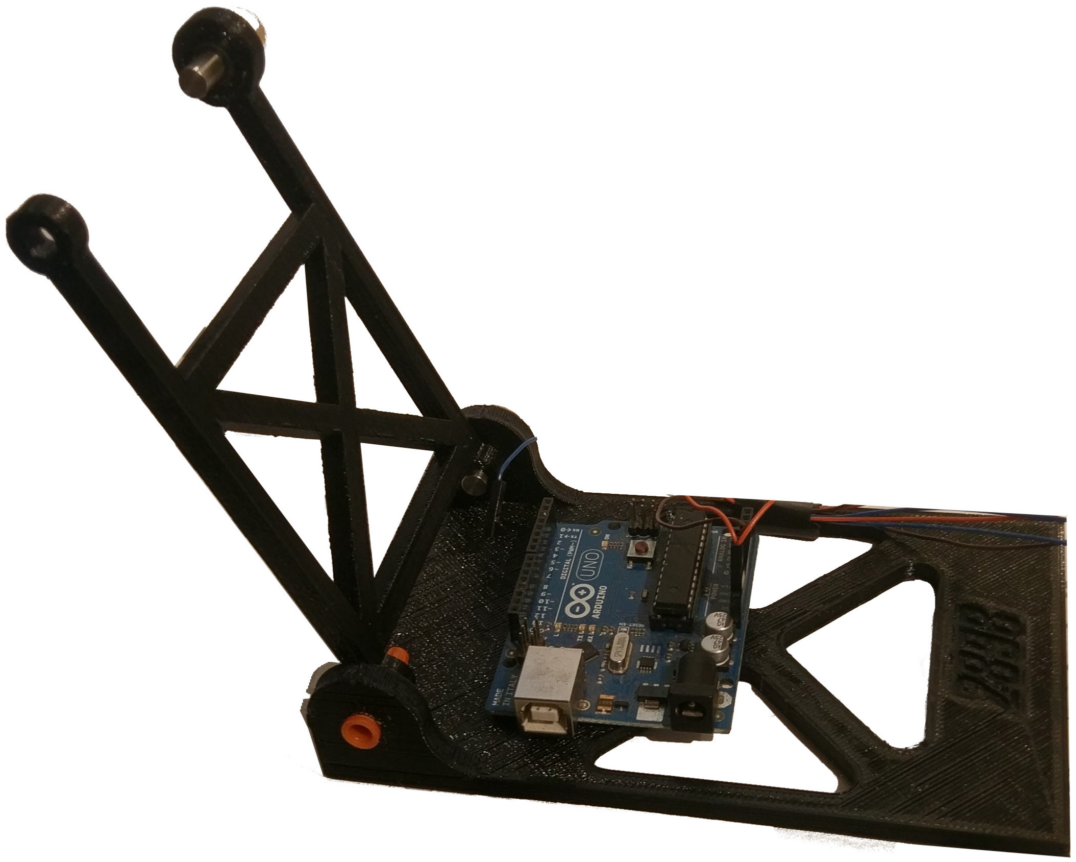


"Wait, how many lasers are on the robot?"
— A Concerned Mentor

Four discrete LIDAR (LIght Detection And Ranging) sensors inside the intake backing bar gather data about the environment immediately in front of the intake. When enabled, the sensors send distance data to a powerful NVIDIA Jetson TX1 embedded computer. There, the data is parsed, and a 3D model of the Power Cube is fit to the sensed distance readings using a Least-Squares algorithm. Information about the orientation of the cube is sent to the RoboRIO, where intake speeds are modulated to automatically align the cube when intaking, making for an easier, more reliable intake system.

Control & Software

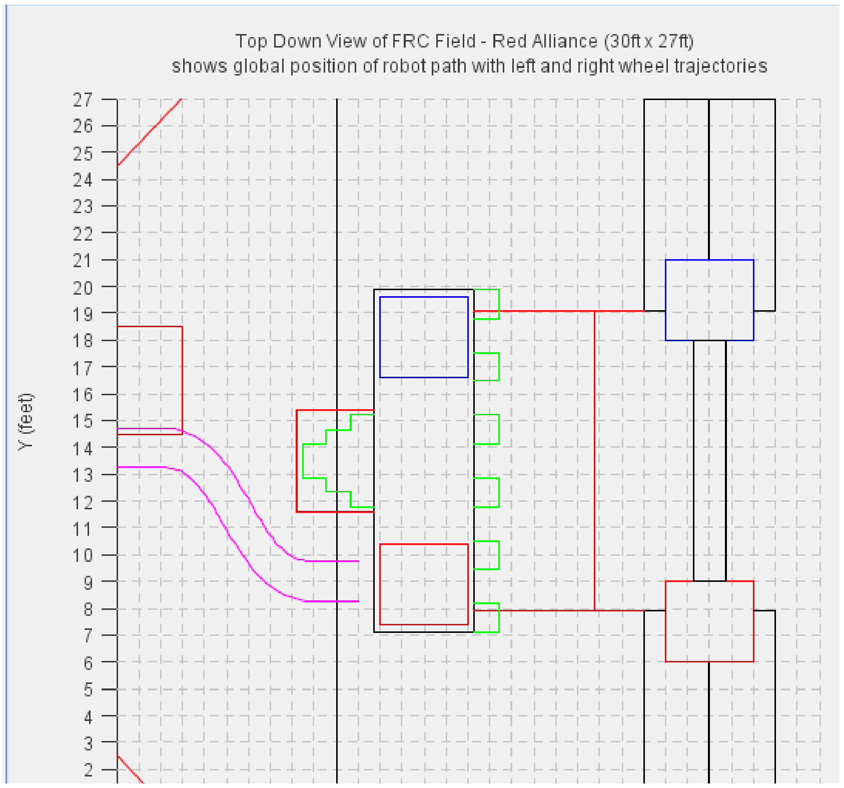
Driver & Operator



When practicing, we found that arm setpoints just won't cut it. With that in mind, Team 2898 has innovated and designed a 3D printed controller resembling our robot's arm. A magnetic absolute encoder communicates with an Arduino, relayed over the network to the robot, which then moves the full sized arm in order to match the orientation of the scale model. This way, we have full, intuitive, fast control over the arm at all times.

Control & Software

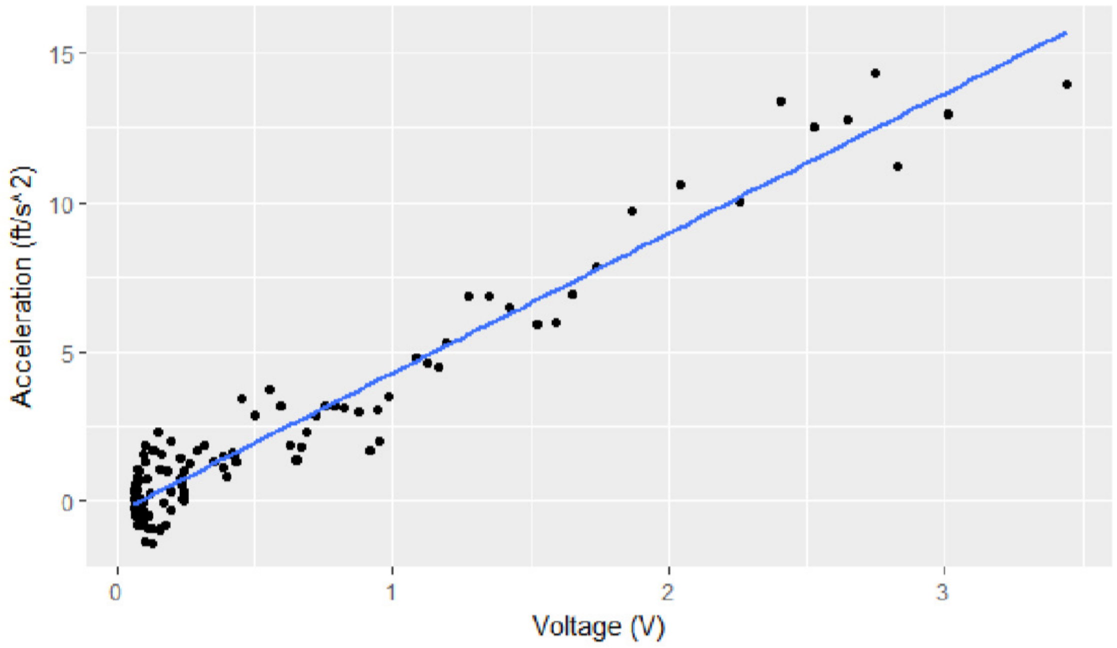
Motion Profiling



The key to a successful auto is smooth movement. Using a spline-based approach, target robot waypoints and headings are converted into a precise, continuous, acceleration- and jerk-limited trajectory. That trajectory is then followed using PVA (Position-Velocity-Acceleration) controllers on the robot to execute our autonomous modes. An easy to use waypoint planning interface makes creating various profiles a breeze.

Control & Software

Drivetrain Modeling

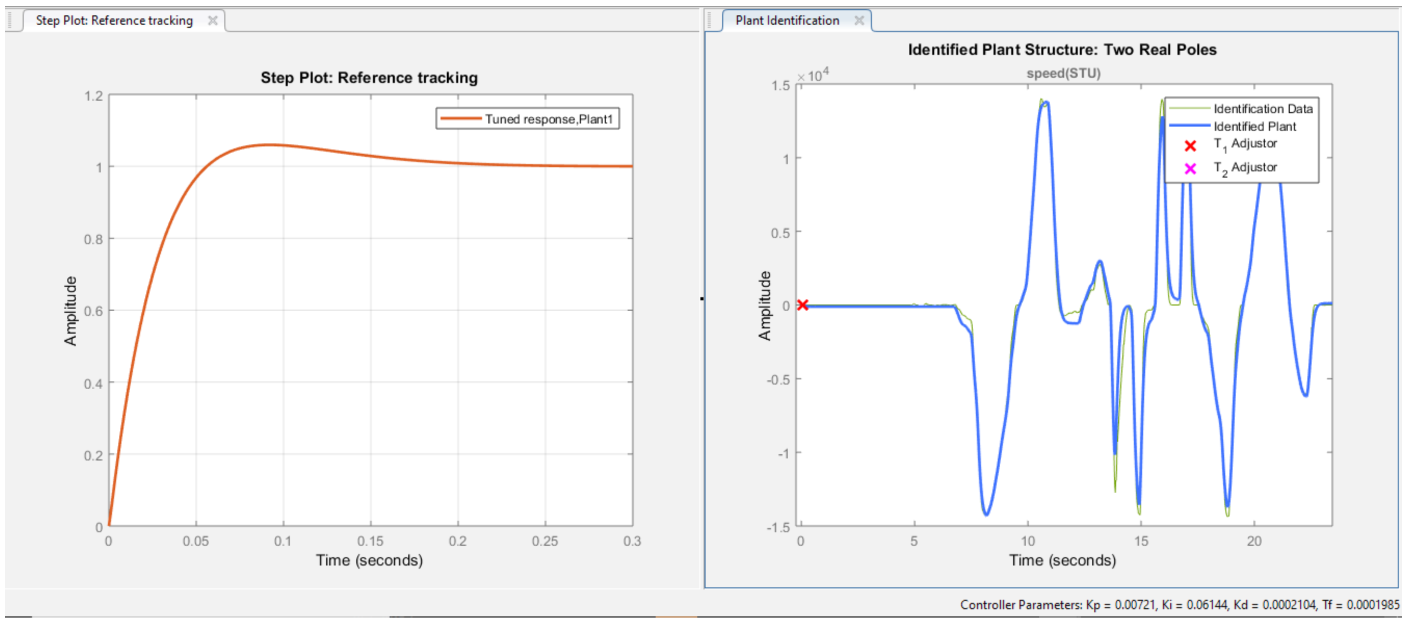


Example graph adapted from Team 449

Even the best constructed drivetrains have a level of inconsistency and angular drift. To compensate, we run extensive testing to generate a mathematical model of our drivetrain, creating equations to output the optimal motor voltage given a target velocity and acceleration for each side of the drivetrain. This way, not only are motion profiles and driver velocity commands followed more precisely, but all minute inconsistencies are compensated for and the motion platform drives measurably perfectly straight.

Control & Software

State-Space Control



Controller Parameters: Kp = 0.00721, Ki = 0.06144, Kd = 0.0002104, Tf = 0.0001985

"I haven't seen this stuff since college!
— An Amused Mentor

The crown jewel of this year's software is our state-space based control loops. We decided that having precise, optimal control over every aspect of robot movement was absolutely vital — so much so that "classical" control loops like PID (Proportional-Integral-Derivative) do not provide necessary performance, no matter how well tuned.

PID loops and similar variants simply react to the error in a system (or "plant") between the setpoint (reference) position and measured position. These reactions are governed and defined by simple coefficients, with no underlying model or knowledge of the system. While robust and easy, these controllers provide decisively suboptimal control.

We made the decision to devote time and effort into developing controllers based on "modern" control theory, incorporating complete mathematical models of robot subsystems into error correction processes. Combined with real-time motion profiling of subsystem movements, we can ensure fast, stable, and robust control for every aspect of our robot.

In fact, if an encoder comes unplugged during the match, our systems can operate with reasonable accuracy for up to fifteen seconds through the accuracy of our governing mathematics.

Control & Software

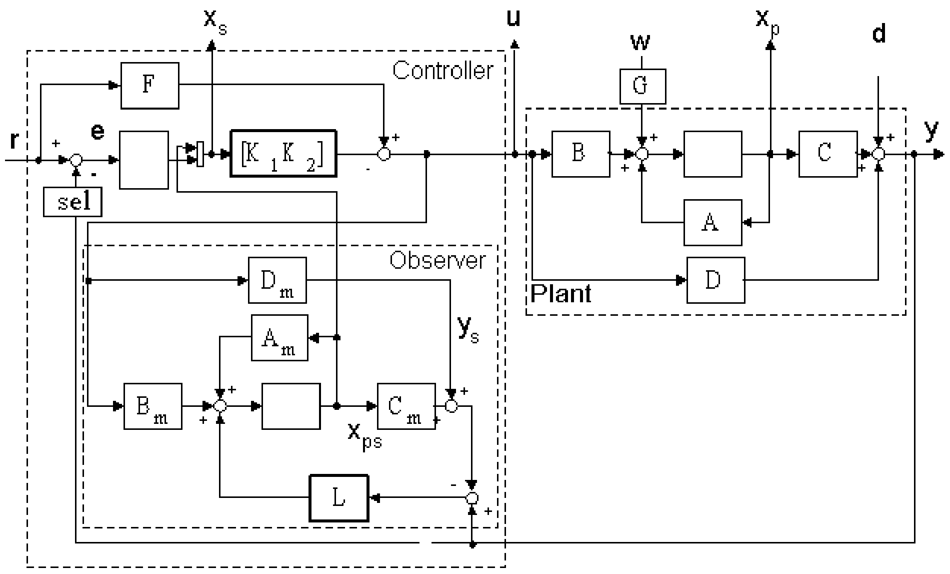
State-Space Control Cont.

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B(\mathbf{r} - K\mathbf{x}(t)) \\ &= (A - BK)\mathbf{x}(t) + B\mathbf{r}\end{aligned}$$

A basic controller incorporating feedback

$$\frac{d}{dt}\begin{bmatrix}\theta \\ \dot{\theta} \\ i\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta} \\ i\end{bmatrix} + \begin{bmatrix}0 \\ 0 \\ \frac{1}{L}\end{bmatrix}V$$

Model for a simple DC motor



A block diagram for a state-space controller incorporating a model of the system (AKA Kalman Filter)

FLYING HEDGEHOGS 2898