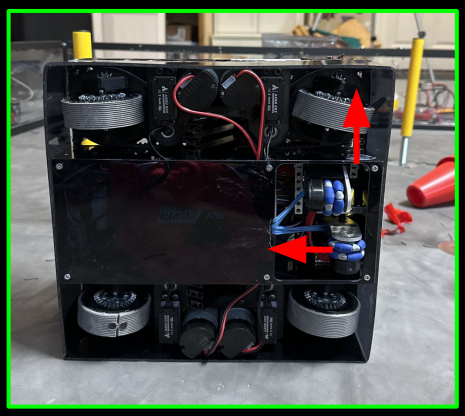
CONTROL AWARD



This is what allows us to reliably run our autonomous at such high speeds.

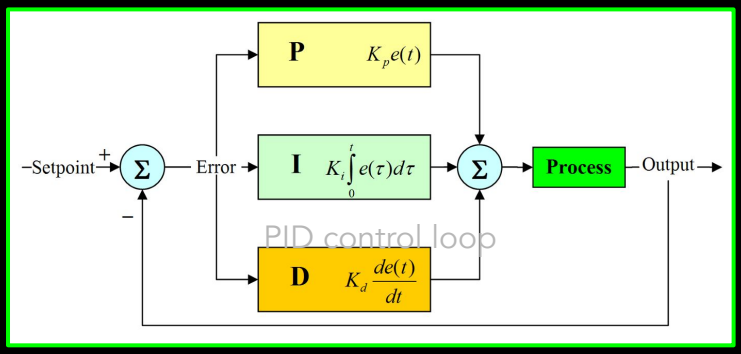
ROBOT LOCALIZATION

Problem: Required accurate (±0.5 inches) & precise (10x in a row) localization method
Solution: Our odometry is made up of two spring-loaded deadwheels, put in a perpendicular formation at the bottom of our robot. One tracks x displacement, and one tracks y displacement. We utilize the REV Hub IMU to track heading. With this setup, we are able to get our robot consistently within ±0.1inches of error from the desired target position.

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \varphi\end{pmatrix} = \begin{pmatrix}\cos(\theta_0) & -\sin(\theta_0) & 0\\ \sin(\theta_0) & \cos(\theta_0) & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\frac{\sin(\varphi)}{\varphi} & \frac{\cos(\varphi)-1}{\varphi} & 0\\ \frac{1-\cos(\varphi)}{\varphi} & \frac{\sin(\varphi)}{\varphi} & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\Delta x_c\\ \Delta x_\perp\\ \varphi\end{pmatrix}$$

PATH FOLLOWING

Problem: Needed an fast and precise way to go from point A to point B
Solution: By utilizing a PIDF controller on our follower, we are able to drive to a position with millimeter accuracy. This helps increase the reliability of our autonomous by ensuring a consistent pickup and dropoff position every time.



ERROR STATE FLAGGING

Problem: Interference from the other alliance in Autonomous can cause transfer failures which stall the auto and result in missing valuable parking points
Solution: By constantly checking the state of our subsystems, we can detect when a certain sequence such as scoring a cone fails. We then trigger an error flag, causing our autonomous to automatically stop everything and park so we are always guaranteed points, regardless of what the other alliance does.

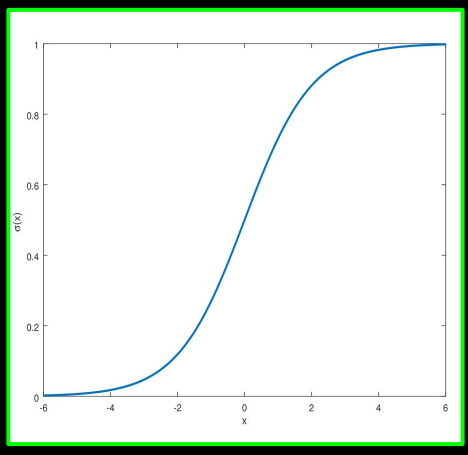
SLEEVE DETECTION

Problem: Needed a quick and easy to tune vision implementation for detecting the signal sleeve
Solution: We use OpenCV, along with a custom vision pipeline to detect which position we need to park in, at the end of autonomous. Our pipeline is adaptable to any lighting conditions, and can work even when the signal sleeve isn't fully in the frame. This is due in combination to the large number of algorithms we use, such as Gaussian blurring. We also open sourced this code, and it is currently being used by hundreds of teams.

CONTROL AWARD

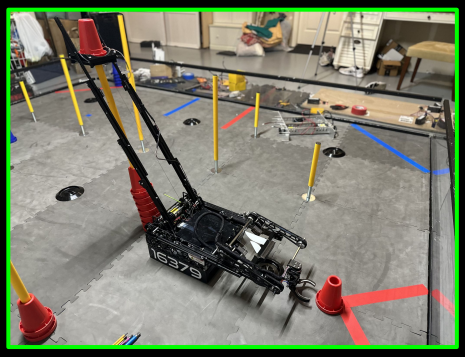
MOTION PROFILING

Problem: When extending and retracting our slides, we found that the movements were not as smooth as we wanted, with just a generic PIDF implementation.
Solution: Motion profiling is the process of creating a precise, smooth motion profile for a system to follow. It allows the creation of custom motion paths, making it a great tool in the real world. Additionally, it is possible to control the position, velocity, and acceleration of a system with high accuracy, enabling the creation of complex and precise motion patterns.



S-Curve Profile

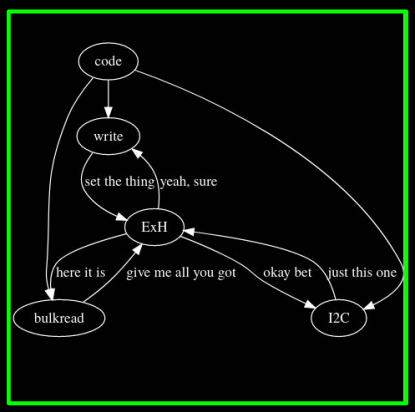
DRIVER AUTOMATION



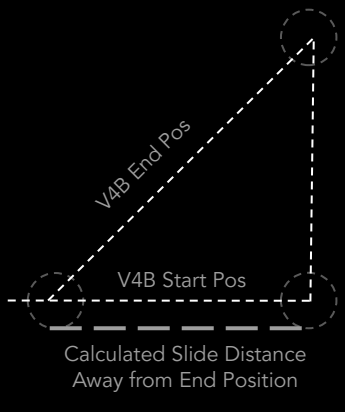
Problem: Driver inaccuracies and error resulting in slower and less fluid movements in our actuators and subsystems
Solution: We created several commands to run multiple subsystems in parallel. For example, we automated the scoring sequence from picking up a cone, to scoring it on the high pole. Due to its well tuned and high degree of automation, we are able to consistently cycle a cone in 2 seconds. This same automation is also applied to other movements, such as lowering and raising our virtual-fourbar from its intake position, to is depositing position.

LOOP TIME OPTIMIZATION

Problem: Due to a large degree of I2C calls, our loop times decreased significantly, resulting in much lower accuracy for our drivetrain and other subsystems
Solution: We now bulk read the control hub, and utilize a parametric writing system to help reduce how often motors are written to. Along with this, we thread the IMU, to allow us to call it asynchronously. By restructuring our code, we also do all of our software tasks sequentially, by reading from the hubs, then doing computations for systems like PIDFs and motion profiling, and then write that output to the hubs.



Bulk read diagram



V4B INVERSE KINEMATICS

Problem: Kept knocking over the cone stack during the autonomous period
Solution: Once we introduced inverse kinematics (IVK) into our subsystems, our autonomous no longer knocked over the cone stack. Utilizing a simple set of equations, we can perfectly calculate the exact movements for the slides to extend out to not knock over the cone stack. This allows much more precise linear slide and fourbar movements.