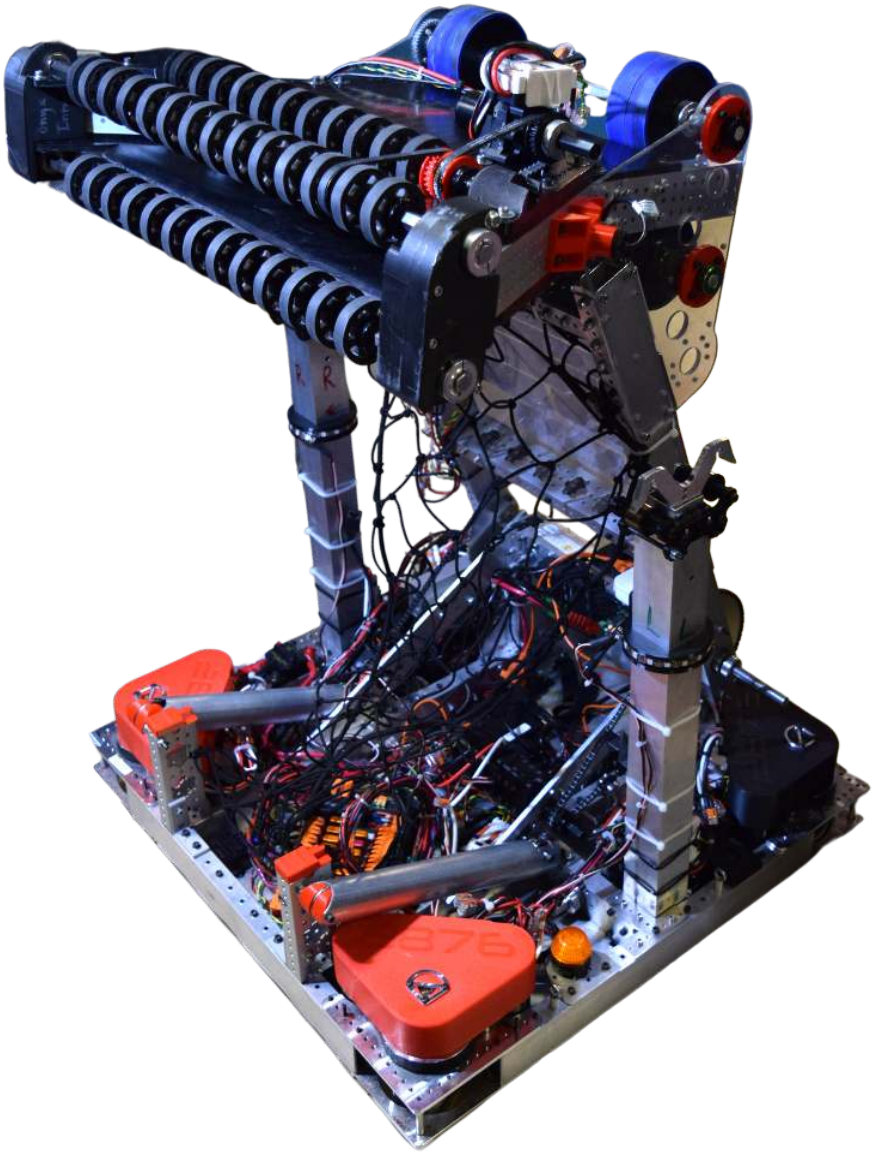
2876 DevilBotz 2024 Technical Binder



INFERNO

Version 1.3

2024-04-03

2024 Crescendo Tech Binder

DevilBotz 2876

Game Analysis **4**
- Strategy 4
- Robot Research and Design 5
  - Implemented Improvements 6

Subsystems **7**
- Drivetrain 8
- Intake 9
- Shooter 11
  - Wheels vs Rollers Testing 12
  - Wheel Configuration Testing 13
- Arm 14
  - Arm Calculations 15
- Climbers 16
- LEDs 20

Programming **21**
- Hardware Definitions 21
- Driver Controls 23
  - Main Driver 23
  - Assistant Driver 24
  - Driver Shuffleboard GUI 24
- Pit Shuffleboard GUI 25
- Vision 26
  - Requirements 26
  - Final Components 26
  - Connection Diagram 26
  - Hardware Placement Diagram 27
  - Camera Mounts 28
    - ArduCam 28
    - Microsoft LifeCam 28
  - Development Phases 29
    - Phase 1: Human Vision Only 29

Phase 2: Automatic Robot Alignment using yaw to April Tag.....29
Phase 3: Automatic Arm Angle using distance to April Tag.....29
Phase 4: Drivetrain Odometry Updates using 3D April Tags.....33
Drive to Amp.....34
Autonomous.....34
Field Position/NOTE Naming Key.....34
Routine Overviews.....35

Game Analysis

Crescendo, the FRC game of the 2024 season, focuses on two main goals, shooting ring shaped NOTES into a SPEAKER and an AMP, and hanging on a chain called the STAGE. ROBOTS can also score into a TRAP, which requires a complex mechanism to hold the TRAP door open, and also simultaneously score the NOTE.

Strategy

Kickoff:

- After kickoff, we split into groups to discuss robot rules, actions, and archetypes
- Each group compiled a list of rules, actions, and archetypes
- After each discussion, we came together as a team to make a master list of robot actions, and archetypes

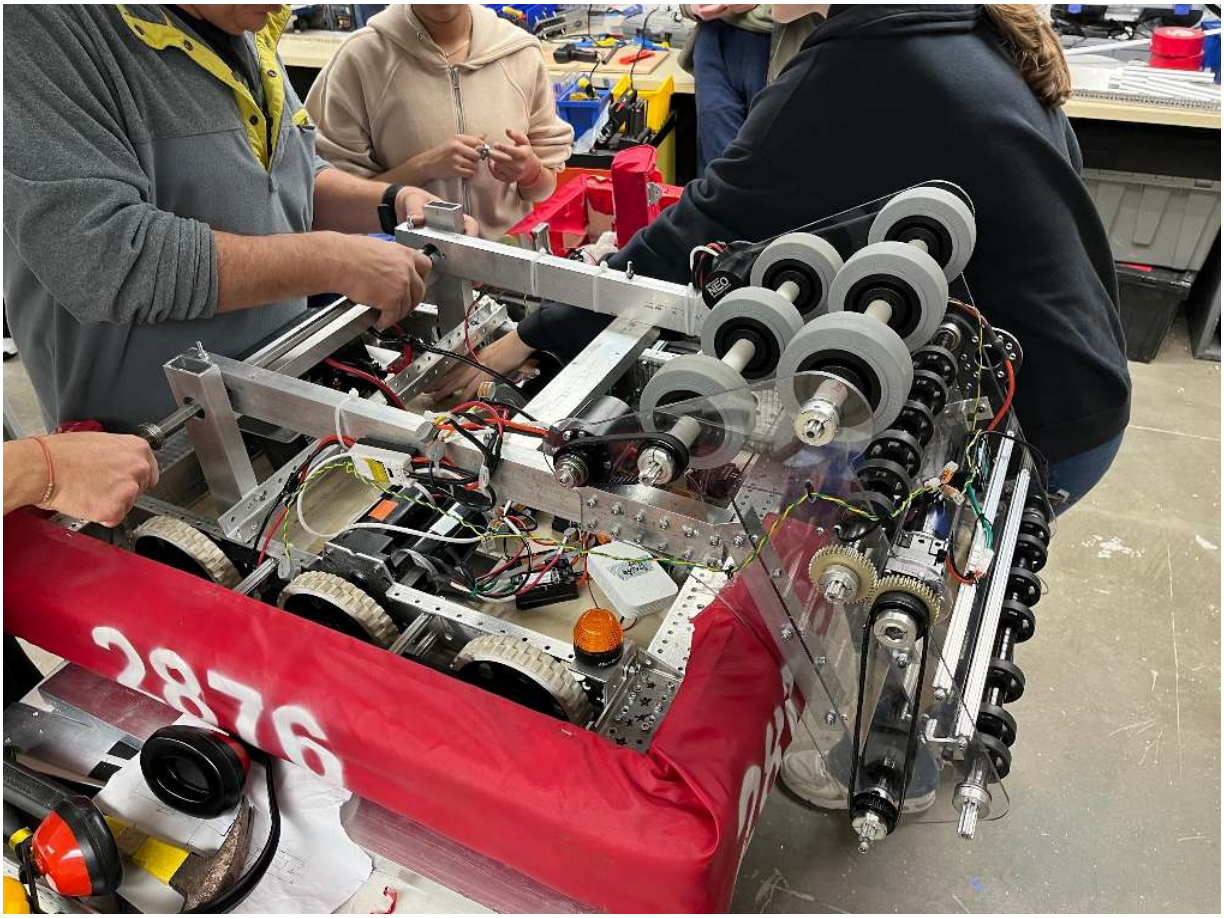
Week 1:

- The executive board led open meetings to put together a Master Robot Actions List, with **the goal of returning to New England District Championship**
- We organized the actions into a MoSCoW list:

Must Haves	Should Haves	Could Haves	Won't Haves
• Be fast • Drive in any direction • Intake NOTES from ground • Score NOTES in SPEAKER consistently • Score NOTES in AMP • Score from center SUBWOOFER • Score from sides of SUBWOOFER • Climb ONSTAGE • Mobility auto • 1 piece auto (SPEAKER) • 1 piece auto (AMP)	• Score NOTES in AMP consistently • Be able to climb harmoniously • 2 piece auto (SPEAKER only) • 2 piece auto (AMPonly) • 2 piece auto (SPEAKER & amp) • April tag alignment	• Directly intake from SOURCE • Score from podium • Score from AMP zone into SPEAKER • Score from anywhere • Be able to go under STAGE • 3+ piece auto	• Score in trap

Robot Research and Design

- We did extensive research into many Robot in 3 Days Teams (Cranberry Alarm, FAMNM, Unqualified Quokkas, and more)
- The Unqualified Quokkas robot stood out to us because it met most of our initial requirements documented in the MoSCoW list
- We decided to build the original Quokka Robot as a practice robot and to find flaws in the design
- **For our competition robot, we made the decision to build an improved Unqualified Quokka robot**



The Unqualified Quokka robot that we built as a practice robot

Implemented Improvements

After testing on Original Quokka Bot, researching flaws published by Unqualified Quokkas, and looking into other robots, we implemented the following improvements to our competition robot:

Intake:

- Widened intake for easier pickup
- Moved secondary intake wheels towards primary intake wheels for less shooter drag
- Constructed top and bottom intake sections with UHMW plastic to prevent friction
- All side aluminum extrusion was covered in UHMW tape to further reduce friction

Shooter:

- Narrowed Shooter to reduce side to side movement, leading to more consistent shots
- Top and bottom shooters geared together for more consistent shots
- Better arranged and different shooter wheels for better shot accuracy

Arm:

- Use of round shaft instead of hex shaft for arm axle to allow for looser tolerancing
- More robust structure and construction to better hold up in high speed collisions
- Faster arm for better Autonomous and faster AMP scoring
- Springs added to hold arm in both frame and stow position with no power
- Deadaxled Arm to protect shaft from stress
- Added arm absolute encoder on output axle

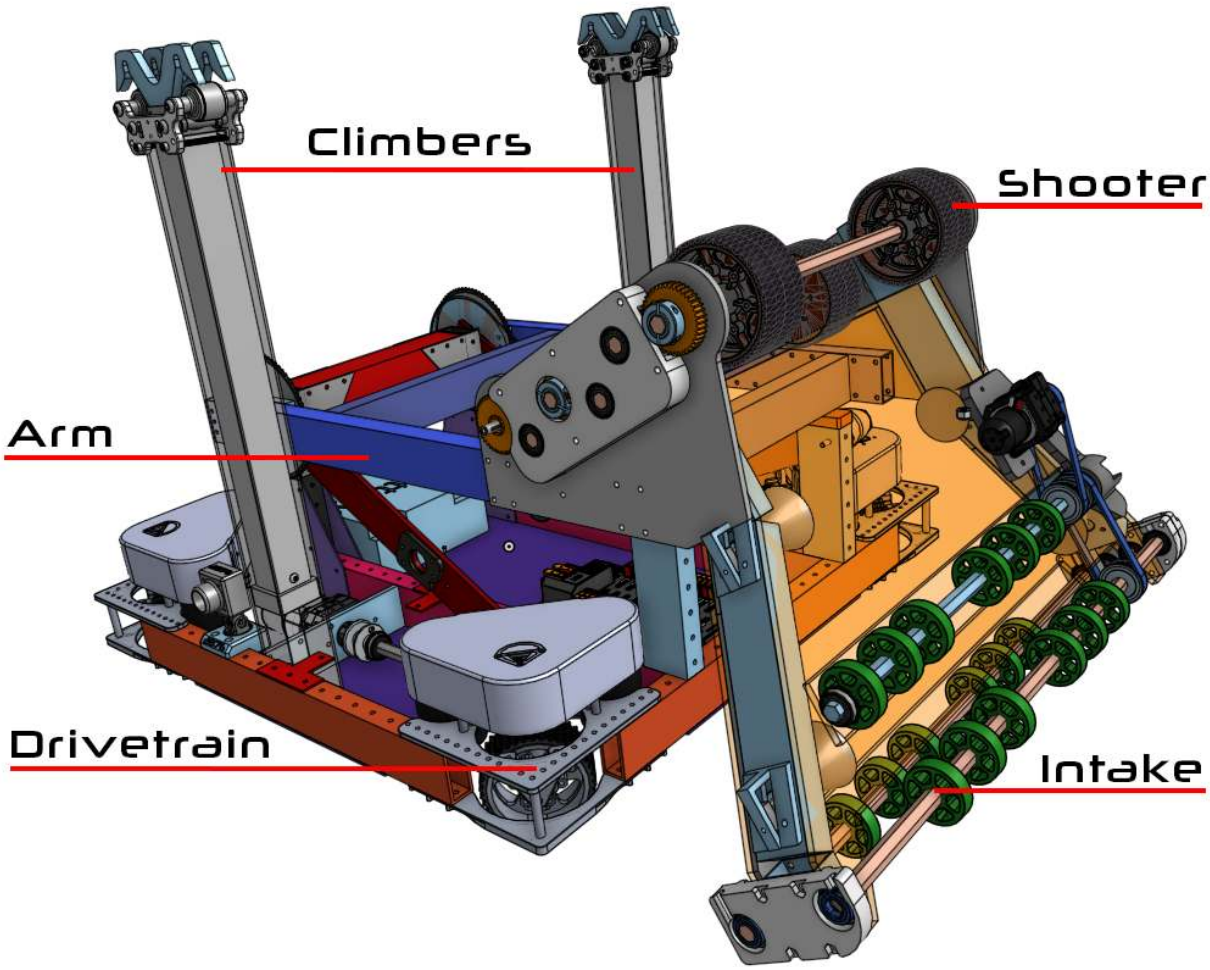
Climbers:

- More robust and reliable Climbers to ensure harmony

Electronics:

- Changed electronics placement for centered CG and better wire management
- Added LED’s for robot to driver communication

Subsystems



Drivetrain

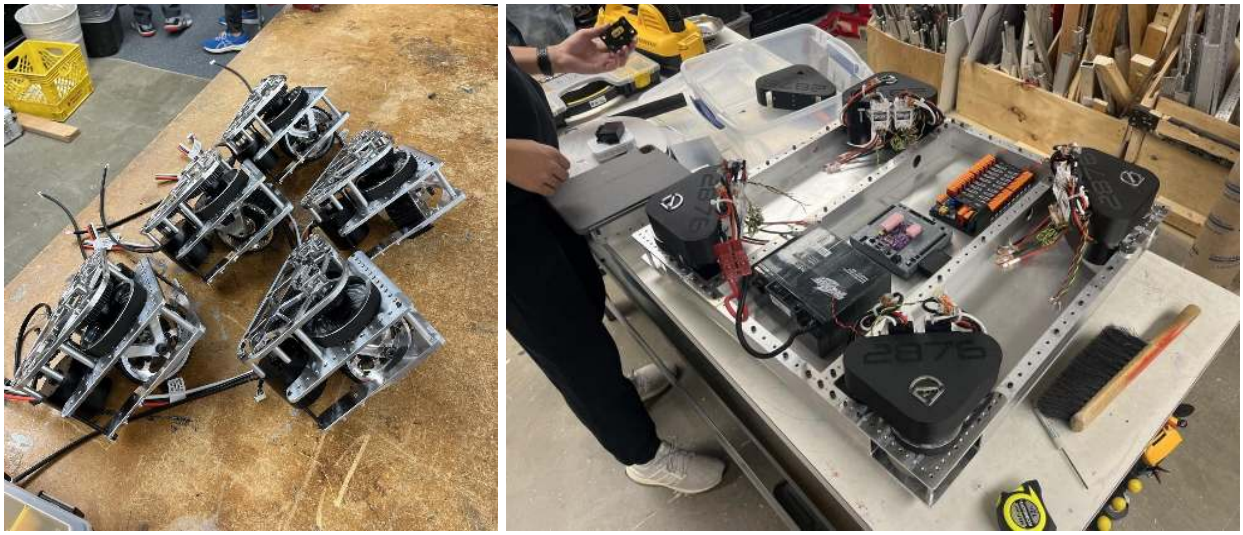
The robot drivetrain is the most critical part of any robot. Without a robust drivetrain, it is impossible to build a reliable and competitive robot.

We therefore set the following drivetrain requirements for our robot this season:

- Robust and reliable
- Fast and agile
- Modular and easy to replace

Given this we decided on the following drivetrain configuration:

- Swerve Drive Specialties Mk4i with NEO V1.1s, L2 Gearing
    - Allows for a heavy robot but high acceleration
- Black Nitrile Tread
    - Allows for more wear resistance than blue neoprene tread
    - Allows for more grip compared to colson wheels
- 28x28 Size
    - Allows for maneuverability and easy harmony chain climbing, but enough room for subsystems and electronics
- ⅛” Aluminum Bellypan
- Battery in the back center, facing flat against the bellypan
    - Allows for a balanced but low CG

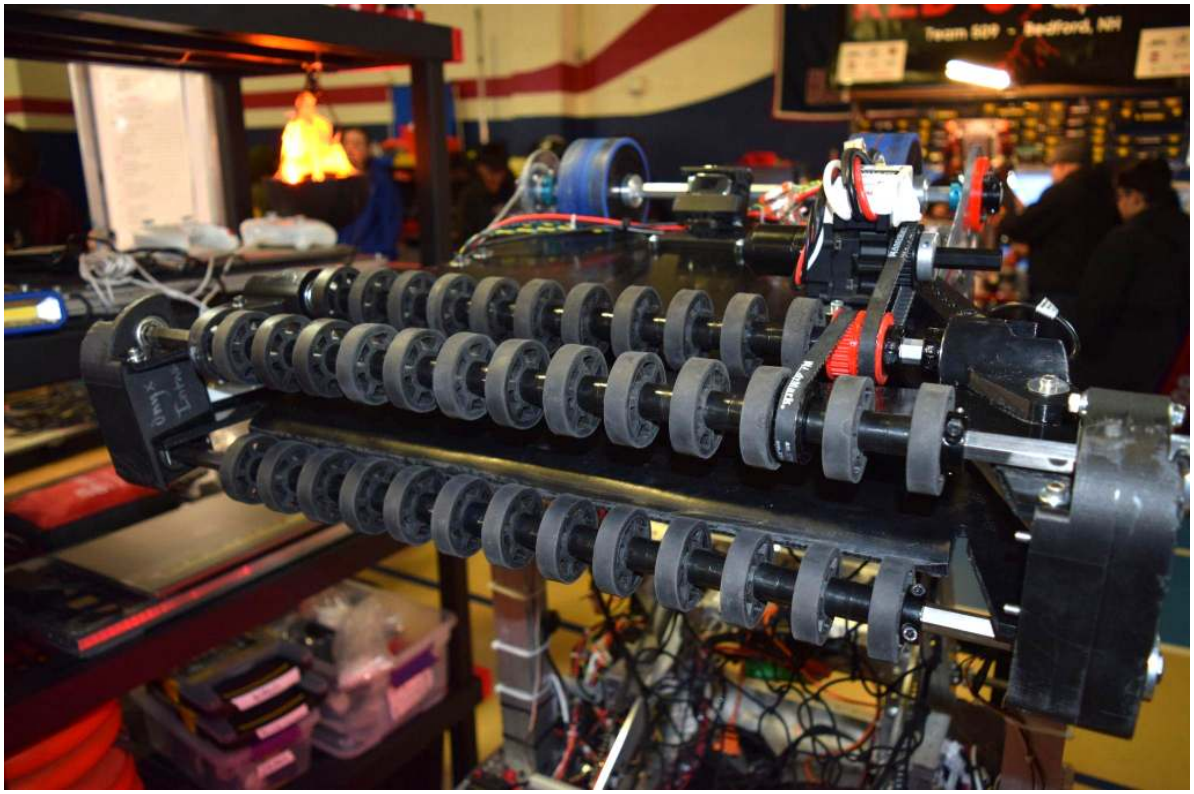


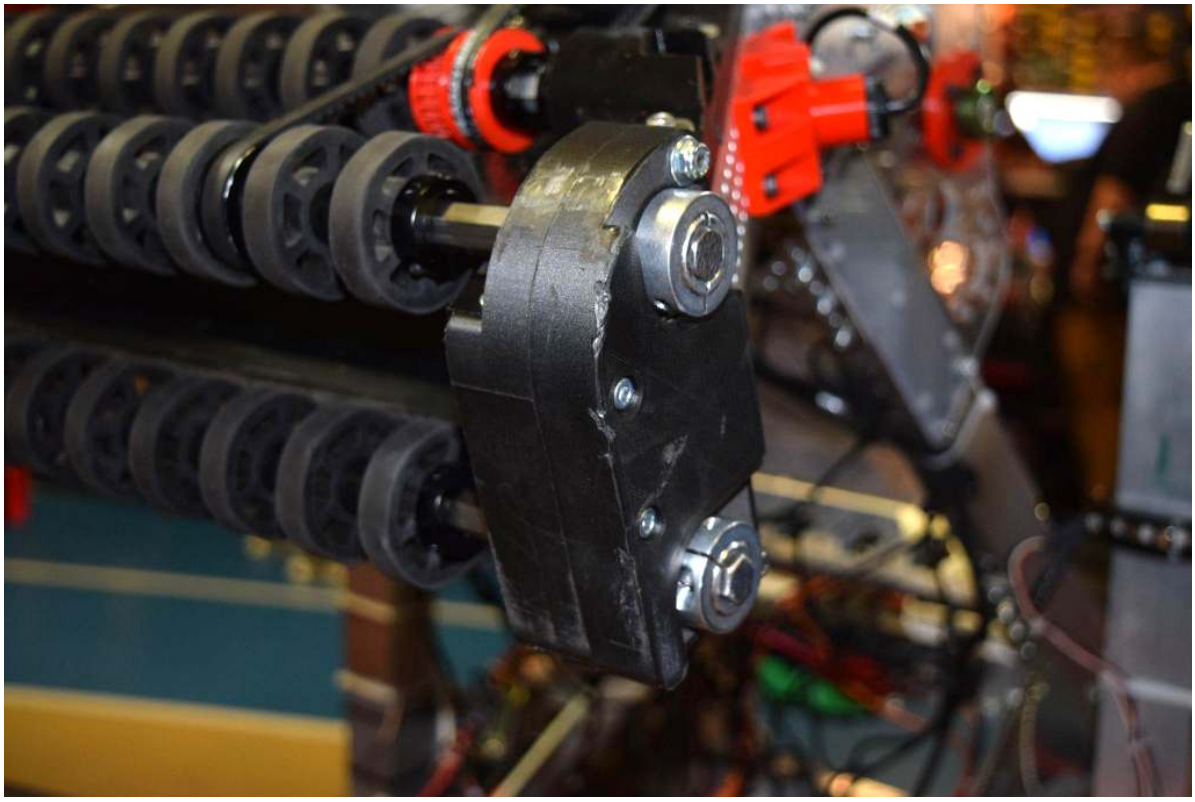
Left: Our SDS Mk4i modules before they were attached to the drivetrain
Right: Our drivetrain prior to wiring and assembly of other subsystems

Intake

The intake is the first part of the robot that the NOTE interacts with. It is responsible for picking up game pieces off of the ground, as well as storing the NOTE after intaking it.

- 23” Wide
    - Allows NOTES to be intaked easier by autonomous routines and drivers
- 2” Black Compliant Wheels, with 0.5” of Compression
    - Compresses the NOTE to ensure reliable intaking
- 2 Primary Rollers, Steel Hex Shafts
    - Steel hex shaft is more impact resistant and is less prone to bending compared to aluminum
- 1 Secondary Roller
    - Placed forward so that it does not interfere with the NOTE when it being shot
- All rollers are mechanically linked with timing belts, gears, and custom gearboxes
    - Carbon fiber reinforced nylon (Onyx) gearboxes to avoid them from breaking when colliding with the field or other robots
- To avoid friction between the intake and the NOTE
    - Intake made of UHMW plastic to allow NOTE and UHMW tape placed on sides of intake to avoid friction
    - Rounded front section to avoid NOTE from getting stuck
- All CNCed and 3d printed parts, avoids tolerance stackup
- Electronics
    - Powered by 4:1 NEO 550
    - SparkMAX Motor controller
    - Photoelectric sensor to detect NOTE



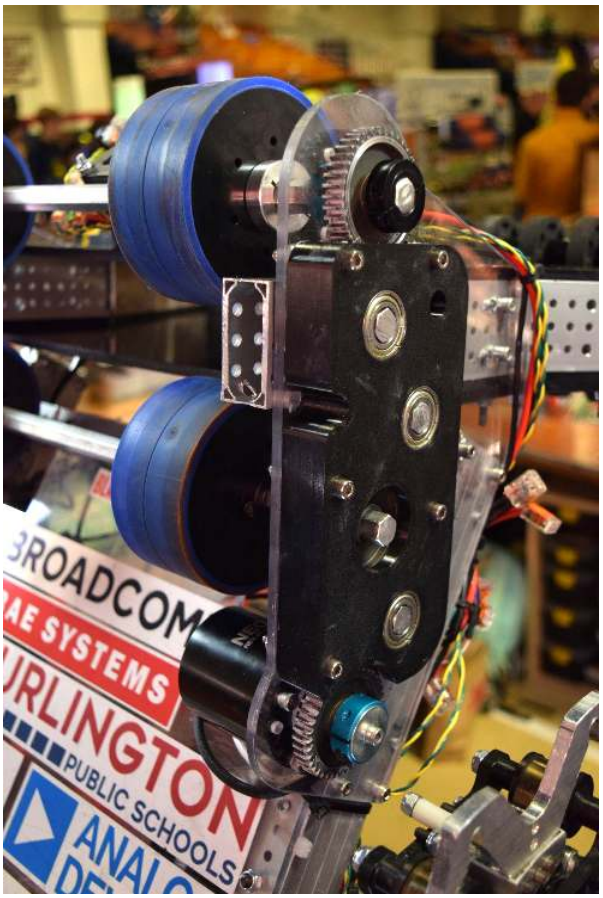


Custom carbon fiber reinforced nylon intake gearbox

Shooter

The shooter is an important mechanism of our robot, as it is how we score most of our points. It takes the NOTE from the intake and shoots it into the SPEAKER or amp.

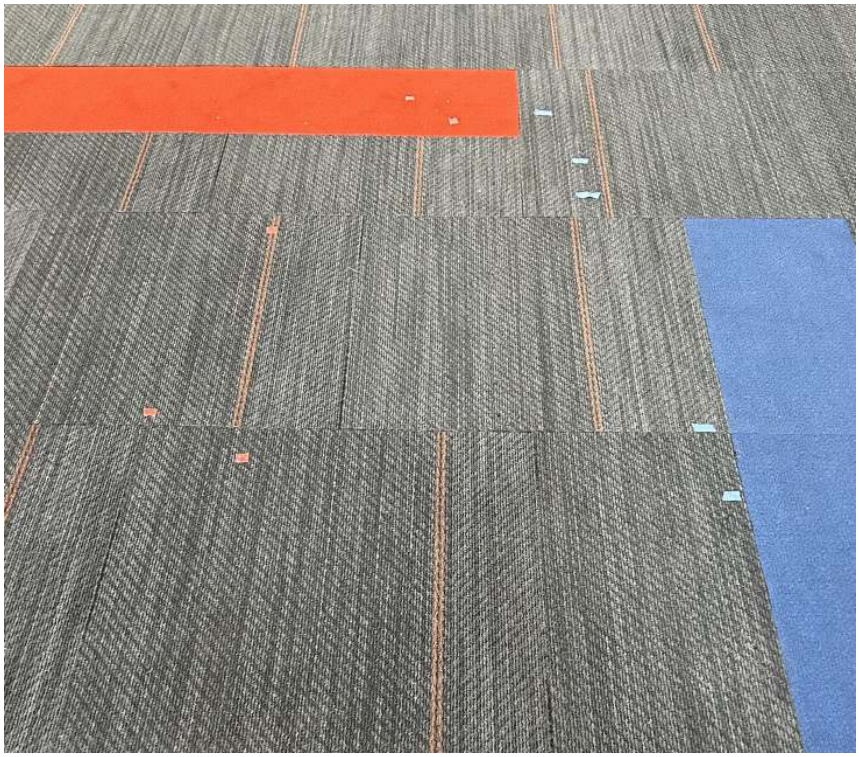
- Top-Down 0.5” Compression
    - Contacting the top and bottom of the NOTE allows for more contact area compared to contacting the sides of the NOTE
    - More contact area leads to more contact time, which leads to a greater transfer of inertia
- 14” Side Compression
    - Prevents side-to-side movement of the NOTE
    - UHMW tape placed on sides to avoid friction
- 4” 50A Stealth Wheels
    - Greater size has a greater mass, which generates more inertia
    - 50A durometer allows for the best grip on the NOTE
- Wheel Configuration: 2 on each end
    - Allows for a straight shot, no side spin
- 2 Aluminum Hex Shafts
    - Light weight allows shafts to spin up quicker
- Controlled by 1 NEO Motor
    - Shafts geared together, allowing them to spin in opposite directions



Custom shooter gearbox

Wheels vs Rollers Testing

	Wheels	Rollers
Mean	138.80”	137.25”
Standard Deviation	12.34”	17.95”



Red Tape: shots by rollers
Blue Tape: shots by wheels

We shot NOTES through a prototype Quokka shooter and a set of 3.5-inch rollers, keeping a half-inch compression on the NOTE in both trials. We recorded the horizontal distance the NOTE traveled from the shooter and calculated the mean and standard deviation.

The mean represents the average distance the NOTE traveled, and the standard deviation shows the accuracy of the shooter. The higher the mean, the further it can shoot the NOTE, and the lower the standard deviation, the more accurate it is.

After this testing, we came to a conclusion that the wheels are the best choice and moved on with them.

Wheel Configuration Testing

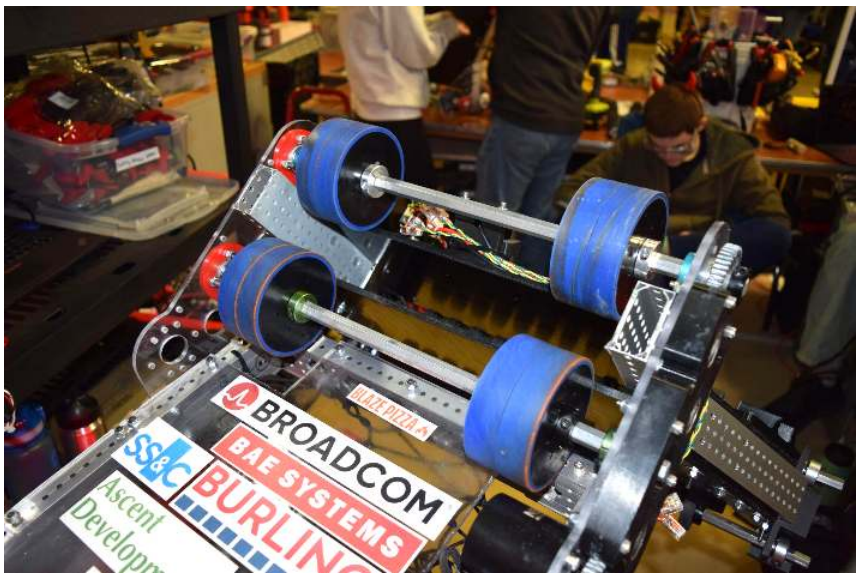
At first, we had the Quokka shooter wheel configuration (1 on each side + 1 in the middle). After some research on Chief Delphi, we saw that wheels on either side of the NOTE were shooting far with consistency and accuracy, so we tried it out.

However, we noticed that the NOTE begins to drop in the air when you shoot from a far-enough distance. To solve this issue, we moved all the wheels to the right side of the shaft, giving the NOTE a good amount of side spin as it slices through the air.

Although this allowed us to shoot farther without the NOTE dropping, we observed that the NOTE was spinning far left and would rarely shoot straight. We did not like this accuracy, so we changed the wheel configuration to 3 on the right and 1 on the left, which still spun the NOTE, but the wheel on the left counteracted some of the spin, which allowed the NOTE to shoot straighter.

We tested the 2x2 and 3x1 wheel configurations from the maximum distance we are allowed to shoot, from the wing, and compared the results.

	2x2 Wheel Configuration	3x1 Wheel Configuration
Accuracy from Wing	50%	5%



Final shooter wheel configuration

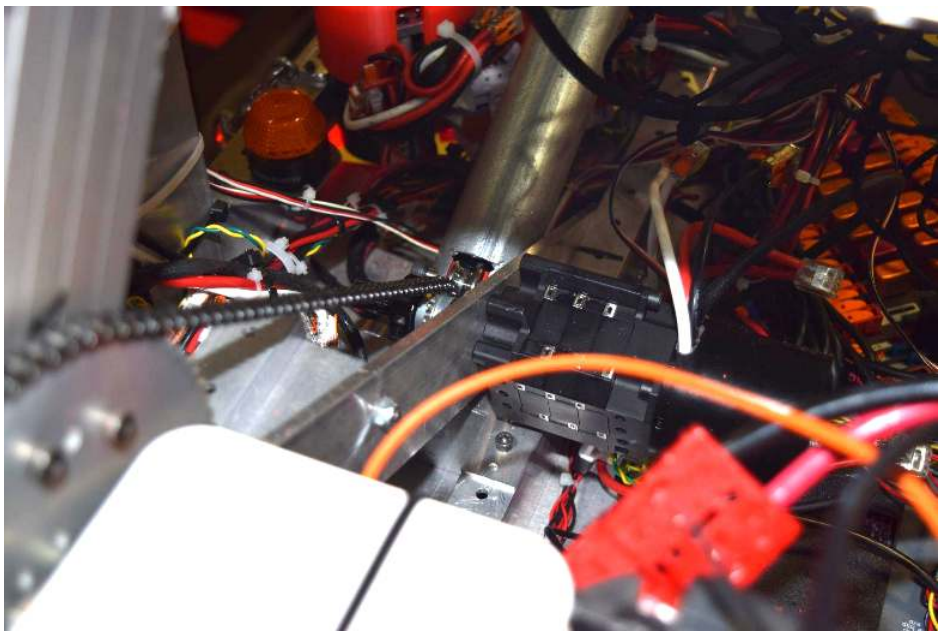
The 2x2 configuration was much more accurate from the wing, so we decided to implement that into our final design.

Note: Both configurations had similar accuracies from close to the subwoofer.

Arm

The arm

- Axle: Steel Round Shaft, Dead Axle Arm
    - Round shaft is ~~stronger and~~ manufactured to tighter tolerances than hex shaft
    - Dead axle allows us to reduces stress on the round shaft
- Chain
    - Two #25 chains, one driven/for springs and one just for springs
    - Arm is driven by the chain, connected via sprockets
    - Gear ratio due to sprockets is 3.5:1calc
- Arm Springs
    - Reduces torque on motor
    - Allows for much faster arm
    - Allows robot to be held in frame perimeter as well as barely above the ground (Stow Position) with no motor power
    - Springs attached to chains via UHMW string
- Reinforced Arm
    - Sponsor and polycarb board on both sides of the arm serve as a reinforcer to the arm
- Final Gear Ratio (between sprockets and gearbox) is 210:1
- Electronics
    - NEO V1.1 with a 60:1 gearbox
    - Spark MAX Motor Controller
    - Absolute encoder on axle



Arm motor, chain, and springs. Springs are stored inside the silver pipe

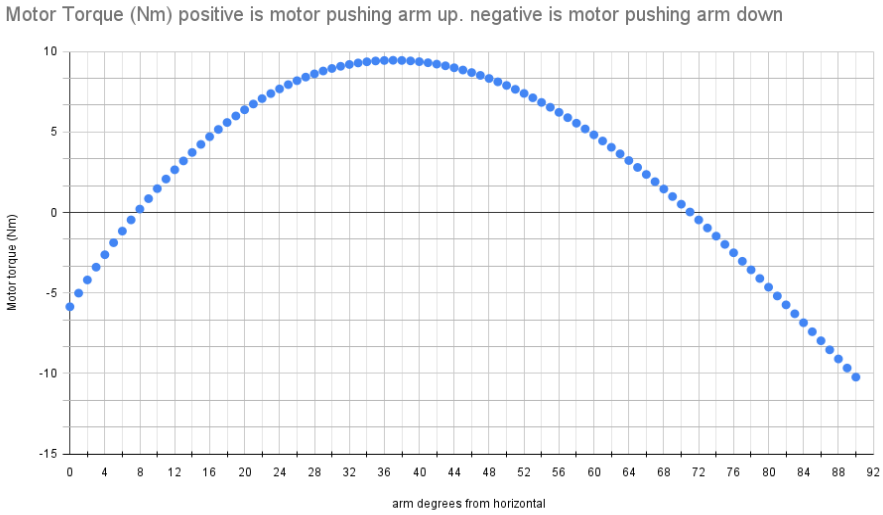
Arm Calculations

moment arm from CG	24	in	NEO Motor capability	0.16	Nm
moment arm from CG	0.6096	m	(should be conservative)	4,000.00	RPM
Arm Weight	30	lbs		66.67	rev/s
Arm Weight	133.7727273	N		266.67	cycles/s
green boxes are inputs					
yellow boxes are outputs					
		required gear ratio	63.9	:1	
		required gearbox ratio	12.09	:1	
		resulting motor speed	1.04	rps	
		resulting lifting time	0.26	seconds	
		max motor torque	10.22880676	Nm	

Desired arm cycle time	0.5	s/cycle	**Chain reduction**		74 tooth sprokect PD	5.89	in
resulting rev/s	0.5	rev/s	big Sproket teeth	74	chain tension radius	2.95	in
target gear ratio	133.3333333	:1	pinion teeth	14	chain tension radius	0.07	m
resulting max motor torque	21.33333333	Nm	chain ratio	5.29	chain travel @ 90deg	4.63	in
					Manual twiddling (add design margin)		
					gearbox selection	25	:1
					total reduction	132.1428571	:1
					max torque	21.14285714	Nm
					motor speed	0.5045045045	rev/s
		for chart			cycle time	0.4955357143	S

Chosen spring			century 81158		century 81159		century 81160	
rate	25	lbs/in	28	lbs/in	25	lbs/in	23	lbs/in
initial tension	11	lbs	11	lbs	11	lbs	11	lbs
max tension	132	lbs	132		132	lbs	132	lbs
max extension	4.8	in	4.3		4.8		5.4	
double spring rate	**50**	**lbs/in**	**56**	**lbs/in**	**50**	**lbs/in**	**46**	**lbs/in**
double spring max force	264	lbs	264	lbs	264	lbs	264	lbs
Initial deflection	4.8	in	4.3	in	4.8	in	4.8	in
initial force	262	lbs	262.8	lbs	262	lbs	242.8	lbs

Calculations that were done to determine the proper spring



Graph of arm position versus motor torque. When torque is 0, the arm will hold position with no power

Climbers

The climber is necessary for endgame, and we wanted to prioritize a steady, reliable, fast, and easily implementable design

When deciding on climbers, we went through the pros/cons of multiple options, including

- Andymark Climber in a Box
- ThriftyBot Telescoping Tube
- Elevator
- Pneumatics
- Pulley
- Hooked Design
- Winches

We ended up comparing (in the table below) AndyMark and ThriftyBot since both were:

- Easy to implement
- Extended quicker/faster
- Allowed the Arm to move more quickly
- Opportunity to extend to different heights and harmonize

The cons that both designs raised, including cost, space, and weight were reasonable for our team as we had the resources and ability to support both

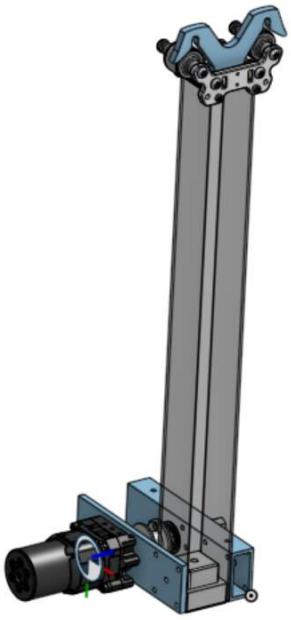
AndyMark Climber in a Box	Thrifty Bot Telescoping Tube Kit

• More detailed instructions
• Starting height is 3ft, so will require a turning point from axis point if we want to go under stage and be under 2”4’
  ○ More CAD / resources needed for implementation that we may not have
	• Thrifty's design allows us to be under the 2” 4’ stage limit easily (no axis/rotation point needed)
	• Can reach the 4 foot limit with two tubes
	  ○ Can be slightly under with just 1 tube

We decided on the Thrifty-bot climber kits, we realized we could have a 1 or 2 stage climber. Due to the calculations outlined in the table, we decided on one stage over two stage climber, as the one-stage telescoping arm satisfied these goals:

- Maneuverability under the stage
- Attempt harmony with other teams
- Climb on the higher end of the chain

	1 stage thriftybot	1 stage thriftybot	2 stage thriftybot	
frame height (in)	4	4	4	
first tube height (in) (2'x2")	20	19	19	Max Height: 27" (4 + 3 climber, leaves 20in max for 1st tube)
1st stage (1.5"x1.5")	18	17	10	
2nd stage (if necessary?)	-	-	10	
hook height (in)	3	3	3	** an overestimate, probably 2 in
max height	45	43	46	
notes	NOT CONSERVATIVE (2 foot, 3 inches)	CONSERVATIVE 2 foot, 2 max height	CONSERVATIVE 2 foot, 2 max height	

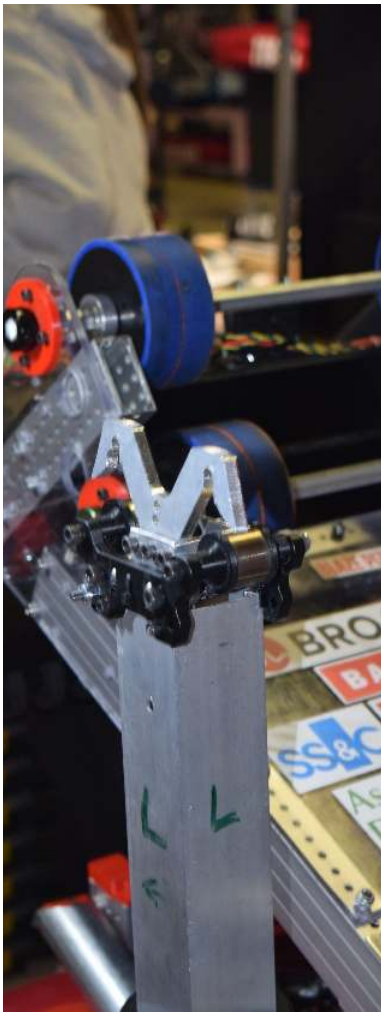


After CADing our design (modeled off of Purdue Ri3D) and designing custom hooks, we created & implemented the model shown on the left

The calculations on the next page were meant to verify gearbox ratio to ensure we could stay up for 5 seconds without power and sustaining speed

This allows us to climb onto the chain and perfectly fit onto the width of it, locking us in place, even after the game ends

Weight of robot	110	lbs
const force springs	12	lbs
# of climbers	2	
total rqd pull force	61	lbs
total rqd pull force	272.0045455	N
capstan diameter	1.2	in
rope factor	0.25	in
estimated actual capstan diamter	1.45	in
estimated actual capstan diamter	0.03683	m
capstan radius	0.018415	m
required torque	5.008963705	Nm
Motor capability	0.16	Nm @ 3000 RPM
speed	4000	rpm
Required gear ratio for torque	31.30602315	
actual gear rat	25	
capstan speed	160	RPM
capstan speed	2.666666667	rot/s
capstan surface area	4.555309348	in
	12.14749159	in/s



Left: Calculations to ensure that we used the correct gear ratio on the climber motor
Right: Final climber hook design

LEDs

The LEDs are essential to optimize cycle times in a match. They allow fast and easy communication between the driver and the robot. We wanted to prioritize visibility and easy-to-implement design.

When selecting the LED, we decided on a 5V AddressabLe Strip based off:

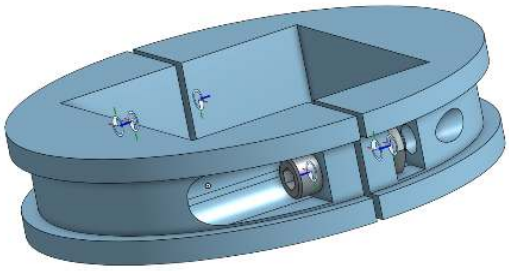
- Easy to implement
- Kept from previous years' robot design
- Dense LED Emitters for visibility
- Simple wiring

We wanted the LEDs to be visible for the driver and the assistant driver during a match. For this, we placed them on the climbers on each side of the robot. This allowed for visibility from all angles of the robot, and the circular design of the mount allowed for it to be seen from all sides. The LEDs get wrapped around the 3D printed mount.

The LEDs have different colors to indicate what is happening to the robot:

- Solid White - Constant
- Blinking Orange - NOTE in Intake
- Blinking Green - AMP Mode
- Blinking Blue - SPEAKER Mode
- Solid Red - Issue with Intake/Shooter/Arm

Because our driver is color-blind, we chose colors that would not be affected by red-green color blindness.





Left: The 3D printed mount that the LEDs get wrapped around
Right: LEDs installed on the robot

Programming

All code is available and published on GitHub:
https://github.com/DevilBotz2876/Crescendo2024

The following critical software tools and libraries were used in the DevilBotz 2876's Crescendo 2024 software:

- WPILib
- AdvantageKit + AdvantageScope
- Pathplanner
- YAGSL
- Photonlib + PhotonVision

Hardware Definitions

In order for the *programming* and *electrical* teams to stay in sync and understand the various hardware components that need software control, the team maintained a "Main Electrical Components" document:

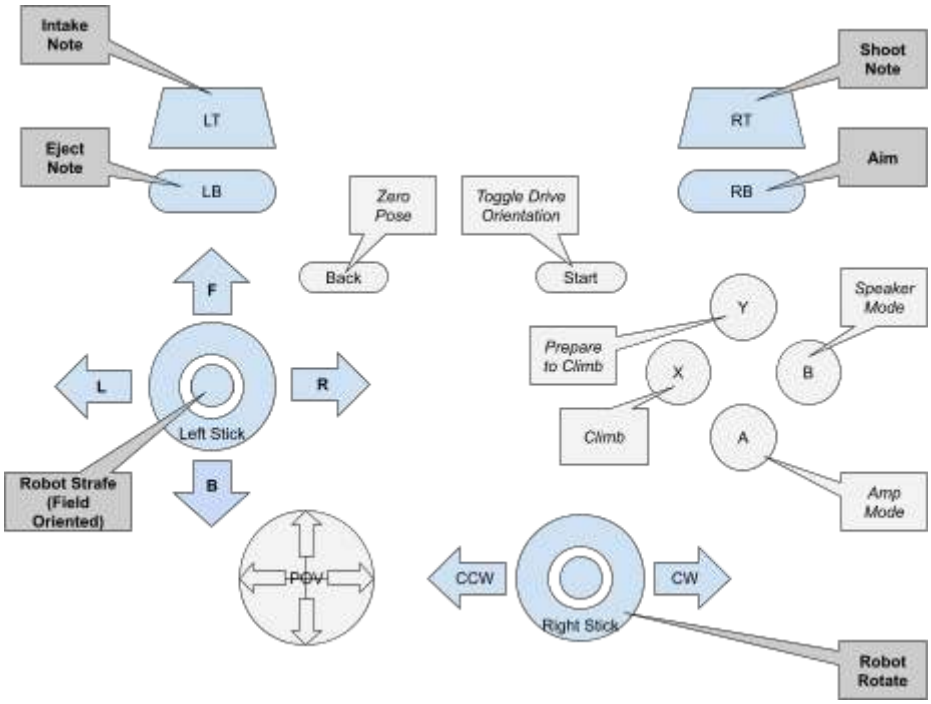
Qty	Item	Desc	Sub	I/O	Port(s)	Purpose
1	NEO Brushless	Motor	Arm	Motor Controller	*SparkMax*	Control intake/shooter position
1	REV Thru Bore Encoder	Absolute Offset Encoder	Arm	RoboRio DIO	DIO:0	Measures arm position, set to absolute mode, use a JST PH 6-pin to 4 Channel PWM Cable to connect to RoboRIO DIO pins, documentation here(https://docs.revrobotics.com/through-bore-encoder/application-examples) we are using it in Duty Cycle Encoder (Absolute) mode
1	SparkMAX	Motor Controller	Arm	RoboRio CAN	**CAN ID: 4**	Control intake/shooter position
2	Built-in NEO encoder	Encoder	Climber	Motor Controller	CPR: 42	Measures the distance the climber has climbed
2	NEO Brushless	Motor	Climber	Motor Controller	SparkMax	Winch for deploying/retracting telescopic arm
2	SparkMAX	Motor Controller	Climber	RoboRio CAN	**Left CAN ID: 6** **Right CAN ID: 7**	

Qty	Item	Desc	Sub	I/O	Port(s)	Purpose
4	CTRE CanCoder	Absolute Offset Encoder	Drive	RoboRio CAN	FL CAN ID: 18 FR CAN ID: 19 BL CAN ID: 20 BR CAN ID: 21	
1	NavX	Gyro	Drive	RoboRio MXP	MXP	Odometry
8	NEO Brushless	Motor	Drive	Motor Controller	*SparkMax*	Swerve
8	SparkMAX	Motor Controller	Drive	RoboRio CAN	FL Drive CAN ID: 10 FL Angle CAN ID: 14 FR Drive CAN ID: 11 FR Angle CAN ID: 15 BL Drive CAN ID: 12 BL Angle CAN ID: 16 BR Drive CAN ID: 13 BR Angle CAN ID: 17	Swerve
1	NEO 550 Brushless	Motor	Intake	Motor Controller	*SparkMax*	Intake Piece from Ground
1	SparkMAX	Motor Controller	Intake	RoboRio CAN	CAN ID: 3	Intake Piece from Ground
1	HiLetgo Photoelectric Switch	Intake sensor	Intake	RoboRIO DIO	DIO:1	Detect NOTE in intake
1	USB Thumb Drive	Logging	Main	RoboRio USB		Robot Logs
1	Built-in NEO encoder	Encoder	Shooter	Motor Controller	*SparkMax* CPR: 42	Measures rotational speeds of the shooter wheels
1	NEO Brushless	Motor	Shooter	Motor Controller	*SparkMax*	Score in SPEAKER/Amp
1	SparkMAX	Motor Controller	Shooter	RoboRio CAN	CAN ID: 2	Score in SPEAKER/Amp
3	Arducam B0332	Camera	Vision	RPi USB	Vision Processor	AprilTag Detection Mounted on Swerve Module Corner (Back Left or Back Right)
1	Microsoft Life-Cam HD-3000	Camera	Vision	RPi USB	Vision Processor	Driver Camera (Visual, Color Object Detection, extra AprilTag detection) Mounted on Intake Arm
1	Raspberry Pi 4	Vision Processor	Vision	Ethernet Switch	Ethernet: ?	Odometry Updates using April Tags How was this powered last year?
1	REV 5V Addressable LED Strip	LED Strip	???	RoboRio PWM	PWM 0	Indicate information to Driver and Operator

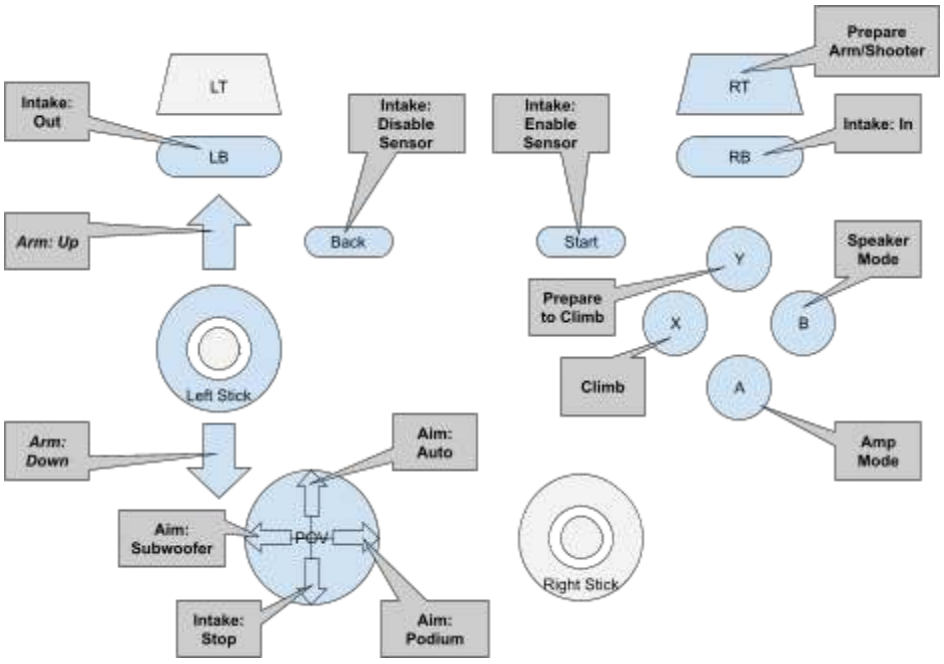
Driver Controls

In order for the *programming* and *drive* team to stay in sync and implement driving controls that make sense, a “Driver Controls” document was maintained to ensure the controls and functionality are understood. The document included the following diagrams, along with more details on specific functionality.

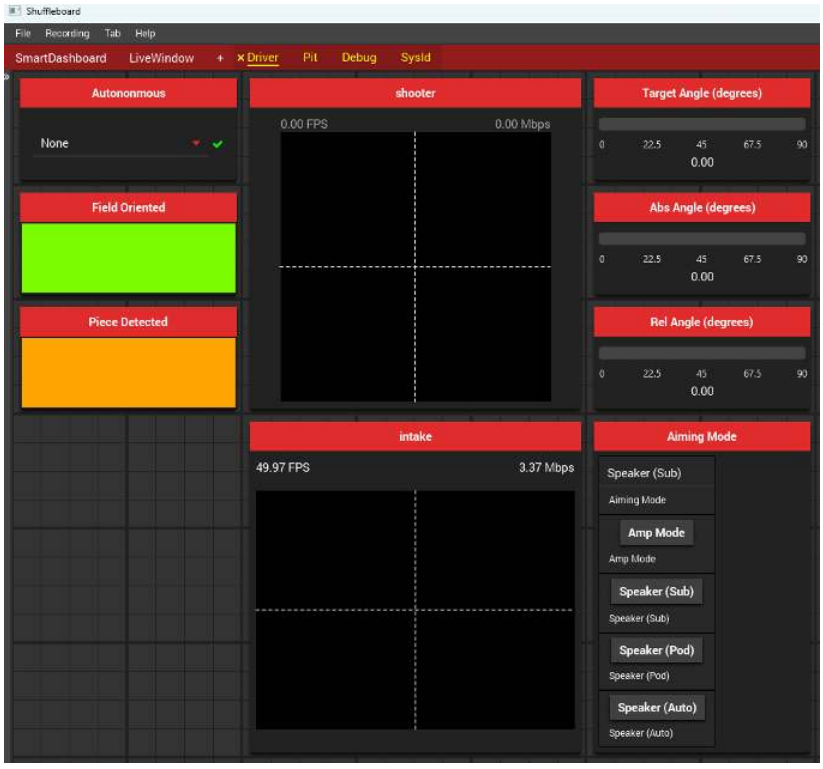
Main Driver



Assistant Driver

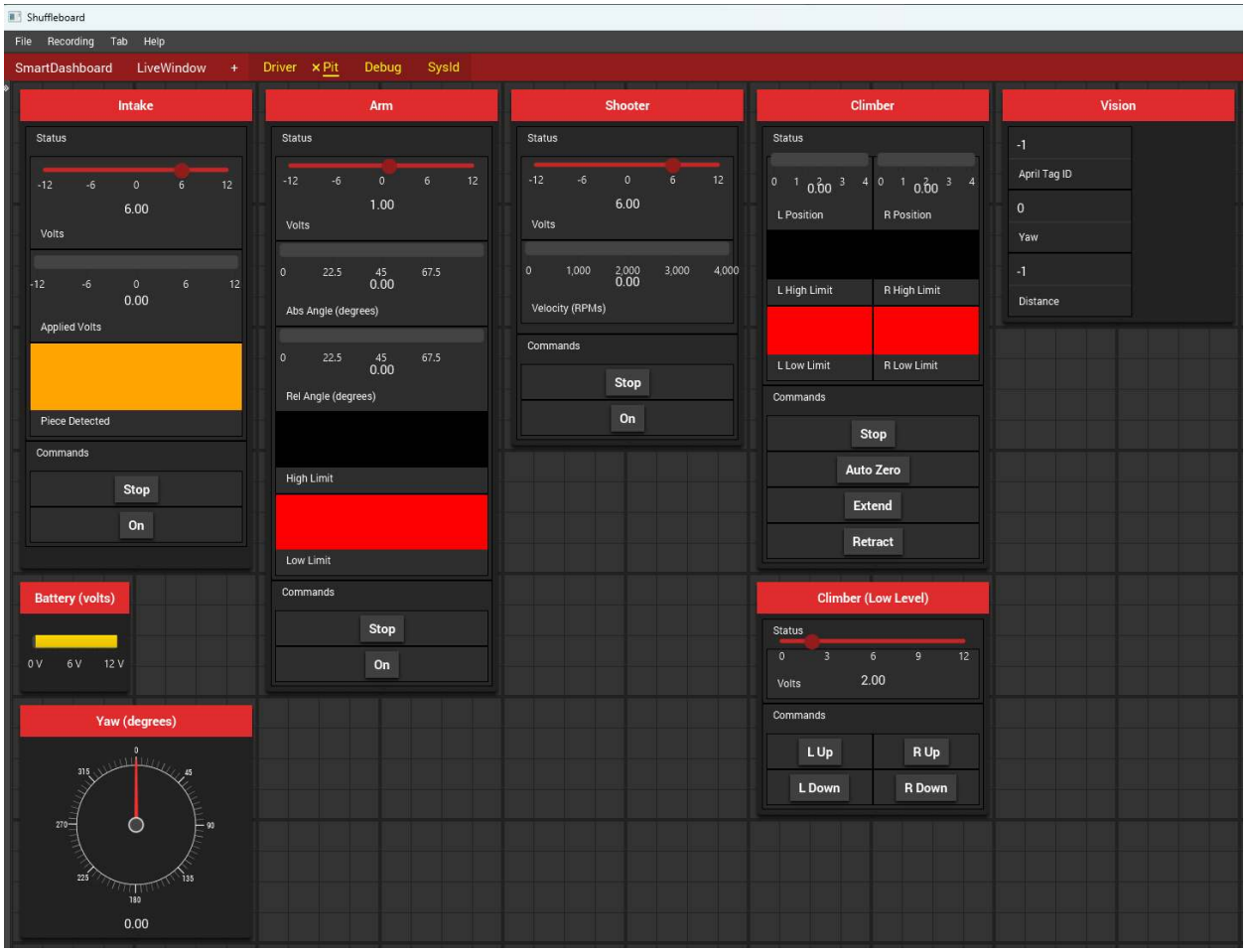


Driver Shuffleboard GUI



Pit Shuffleboard GUI

Shuffleboard GUI controls were created to assist in the pit crew's pre-post match checklist. The pit is given low-level controls that allows them to set the voltages directly, thus minimizing control issues related to software.



Vision

Requirements

At first, we outlined the minimum system requirements, which included:

- ArduCam #1 (Back Center) connected to the Raspberry Pi via USB
    - *for AprilTag-based SPEAKER/AMPTarget Alignment*
- ArduCam #1 (moved to Back Right) connected to Raspberry Pi via USB
    - *for AprilTag-based Odometry Updates*
- MS LifeCam connected to RoboRio via USB
    - *Primary Use: for Human Driver Camera*
    - *Secondary Use: Vision Camera (Color Detection/Node Alignment using vision running on RoboRio)*
- Raspberry Pi connected to RoboRio via Ethernet

Upgrade #1

- Added ArduCam #2 (Front Left) connected to Raspberry Pi via USB
    - *Augment AprilTag-based Odometry*

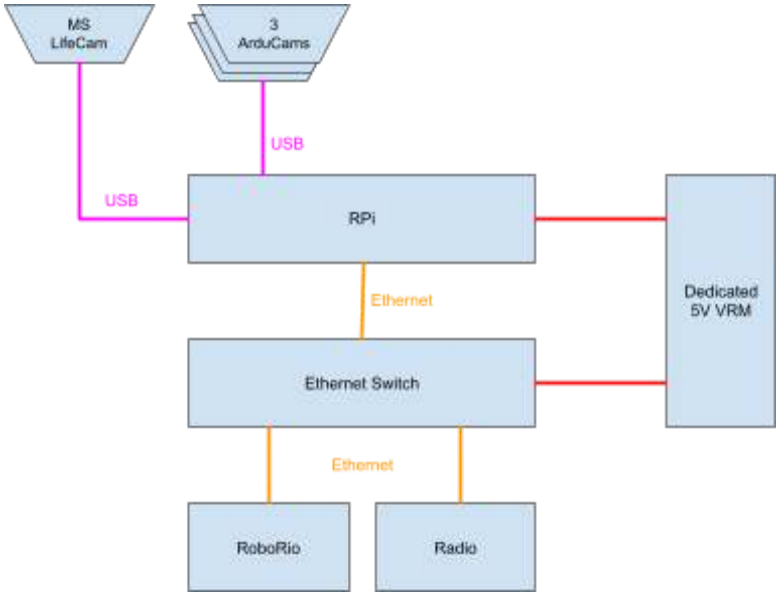
Update #2

- Added ArduCam #3 & #4 (Back Left & Front Right) connected to Raspberry Pi via USB
    - *Further Augment AprilTag-based Odometry*
- MS LifeCame connected to RoboRio via USB
    - Was no longer being used as a secondary Vision Camera, unless
        - We add a second USB hub to Raspberry Pi
        - Implement node detection/vision directly onto RoboRio

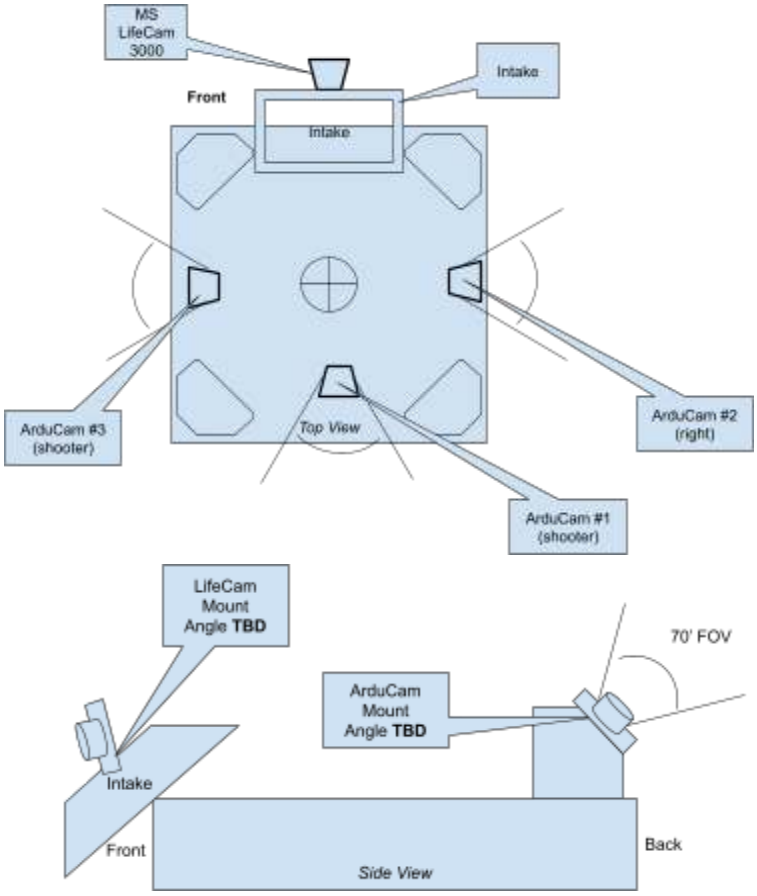
Final Components

- 1x Raspberry Pi 4 (RPi)
    - Connected to the RoboRIO via ethernet
    - Communicates via NetworkTables
    - Running PhotonVision
- 3x Arducams + 1x Microsoft LifeCam
    - All connected to the RPi via USB
- Custom designed and 3D printed camera mounts

Connection Diagram



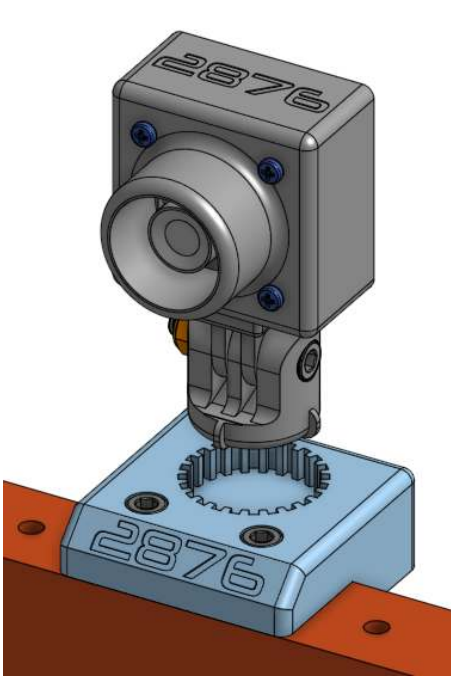
Hardware Placement Diagram

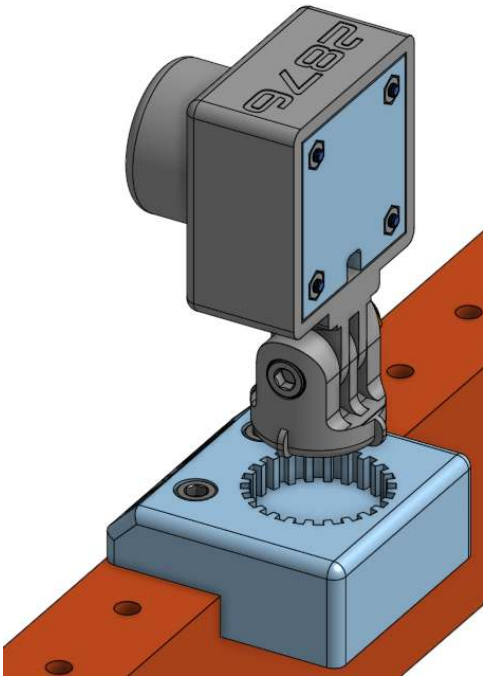


Camera Mounts

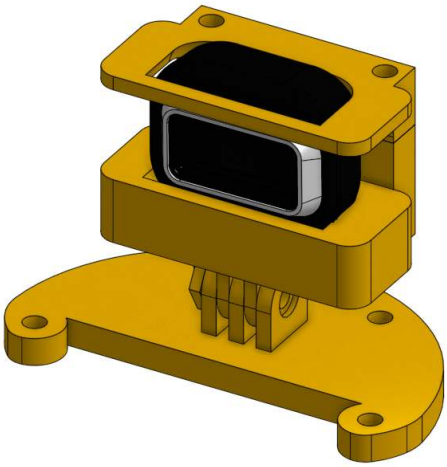
The camera mounts were designed with adjustability in mind, so we developed a GoPro compatible mounting mechanism that would allow us to easily adjust the pitch of the camera as needed. It will also allow us to use off-the-shelf camera mounts in the future as needed.

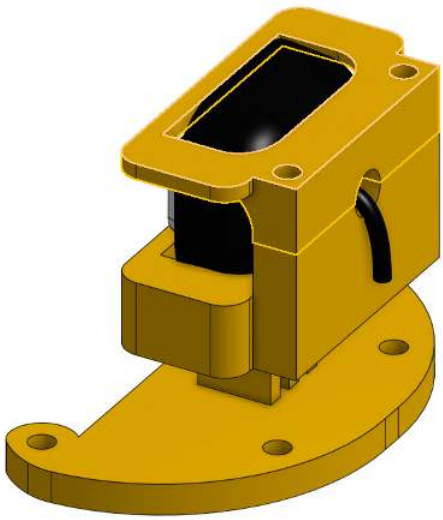
ArduCam





Microsoft LifeCam





Development Phases

This is DevilBotz 2876's first year trying to implement any sort of vision processing for competition. The team split up the development into various phases with expectation that each one builds upon the stability of the previous one.

Phase 1: Human Vision Only

In this phase, only the intake (MS LifeCam) and shooter (ArduCam #1) cameras are used. No vision processing is done. The camera streams are viewable in the Shuffleboard GUI to help the driver align with NOTE pickup and SPEAKER/AMPalignment.

Phase 2: Automatic Robot Alignment using yaw to April Tag

In this phase, the shooter camera (ArduCam #1) is running PhotonVision's 2D AprilTag tracking. The camera is centered with the shooter, so PhotonVision's off-the-shelf code to compute the relative yaw to the target is used to automatically rotate the robot to center itself (left/right) with the target AprilTag.

A rotation PID was implemented and tuned to reach the desired robot yaw as fast and reliably as possible.

```
public Optional<Double> getYawToAprilTag(int id) {
  PhotonTrackedTarget target = findAprilTag(id);

  if (target != null) {
    return Optional.of(target.getYaw());
  }

  return Optional.empty();
}
```

It is important to NOTE that, other than being centered in the robot, no other calibration was required.

Phase 3: Automatic Arm Angle using distance to April Tag

In this phase, the shooter camera's "transform 3D" was measured as accurately as possible. The transform 3D consists of translation and rotation 3D components as documented in our code:

```
  /*
   * Location of the camera view relative to the robot (in
meters)
   *
   * Translation(x,y,z) - location of the camera relative to
the center of the robot (in meters)
   *    x +/- is forward/back from center
   *    y +/- is left/right from center.
   *    z is relative to the ground (and should be positive)
   *  E.g. Translation(0.0, -0.5, 0.5) is 1/2 meter right of
center and 1/2 meter above the ground
   *
   * Rotation(roll, pitch, yaw) - orientation of the camera
view (in radians)
   *    roll +/- is CCW/CW around x-axis (should generally be
0)
   *    pitch +/- is down/up around y-axis (should generally be
<= 0)
   *    yaw +/- is left/right around the z-axis
   *  E.g. Rotation(0, Units.degreesToRadians(-20),
Units.degreesToRadians(-90)) is pitched up by 20 degrees, and
facing to the right
   */
```

We also had to import the Crescendo 2024 April Tag Field Layout as provided by the PhotonVision library.

Once the shooter position and field layout were configured in PhotonVision, we could rely on the “distance” that PhotonVision’s PhotonUtils computed:

```
  public Optional<Double> getDistanceToAprilTag(int id) {
    PhotonTrackedTarget target = findAprilTag(id);

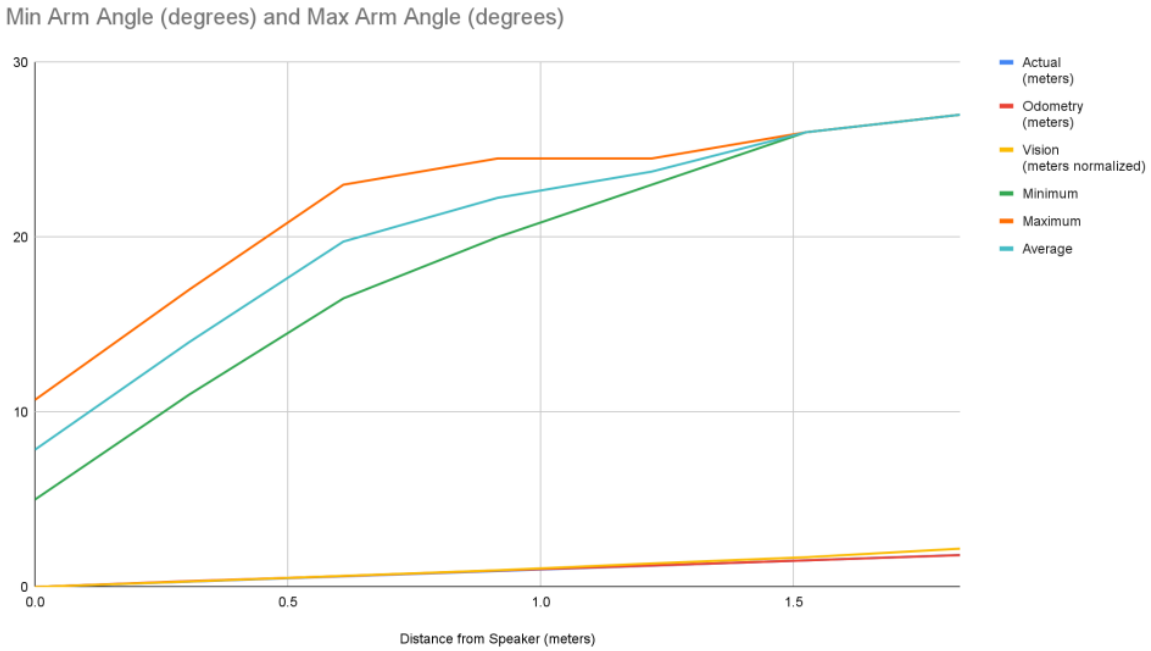
    if ((target != null) && (primaryCamera != null)) {
      return Optional.of(
          PhotonUtils.calculateDistanceToTargetMeters(
              primaryCamera.getRobotToCamera().getZ(),
```

```
fieldLayout.getTagPose(target.getFiducialId()).get().getZ(),

-primaryCamera.getRobotToCamera().getRotation().getY(),
                Units.degreesToRadians(target.getPitch())));
    }
    return Optional.empty();
  }
```

Once we had this data, we had to create a mapping of “distance to arm angle” for scoring in the SPEAKER. We measured the following information at various distances from the SPEAKER:

- Distance from subwoofer
    - Actual
    - Reported by Drivetrain Odometry
    - Reported by Vision Odometry
- Angles for Scoring Reliably (at least 5 in a row)
    - Minimum
    - Maximum



Based on this data, we determined that we can reliably score from at least 1 meter

away, so put together a piecewise function to compute the ideal arm angle given the distance:

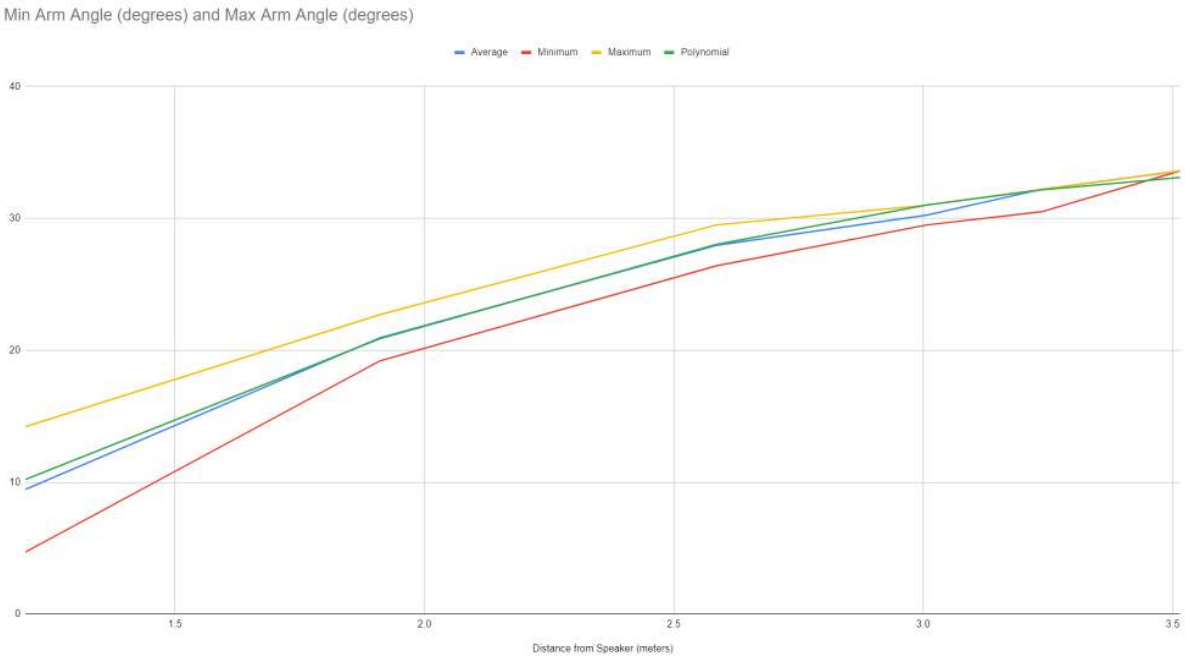
```
  public Optional<Double> getArmAngleFromDistance(double
distanceInMeters) {
    /* TODO: Insert mapping of distance to arm angle for
scoring in SPEAKER */
    distanceInMeters -= Units.inchesToMeters(38);
    Optional<Double> angle = Optional.empty();
    if (distanceInMeters < 0.6) {
      angle = Optional.of(8.0 + 6.0 / 0.3 * distanceInMeters);
    } else if (distanceInMeters < 1.2) {
      angle = Optional.of(2.0 / 0.3 * distanceInMeters + 16.0);
    } else if (distanceInMeters < 1.5) {
      angle = Optional.of(25.0);
    } else {
      angle = Optional.of(1.0 / 0.3 * distanceInMeters + 21.0);
    }

    if (angle.isPresent() && angle.get() <
ArmConstants.subwooferScoreAngleInDegrees) {
      angle =
Optional.of(ArmConstants.subwooferScoreAngleInDegrees);
    }

    return angle;
  }
```

As shown in the graph, we did detect some drift in the drivetrain and vision reported distances, but found that in practice, at ~1 meter from the SPEAKER, we did not need to account or adjust for this discrepancy.

2024-04-03 Updates:
The team gathered the distance to arm angle measurements again and decided to use a 2nd-degree polynomial regression instead of a linear regression:



Using an online 2nd-order polynomial regression solver, we came up with the following equation to model the arm angle to the distance:

```
armAngle = -3.2*(distance^2) + 23.7*distance + -10.3
```

Phase 4: Drivetrain Odometry Updates using 3D April Tags

This phase requires at least the following known steps to achieve:

- Perform camera calibration using a grid pattern to enable “3D” April Tag tracking
- Verify the reliability of the April Tag based odometry
- Feed the vision-based odometry estimates to the drivetrain

2024-04-03 Updates:
We were fortunate to have time after our 2nd qualifier to get basic 3D vision-based pose estimation working.

Since we are using the common “ArduCam OV9281” camera, www.calibdb.net already had calibration data available to download which was added to the PhotonVision camera configuration.

We then re-measured the camera's position/rotation and then periodically added the vision's pose estimate to the drivetrain's odometry. This allowed us to quickly implement a new feature that was requested by the drive team: "Drive to Amp"

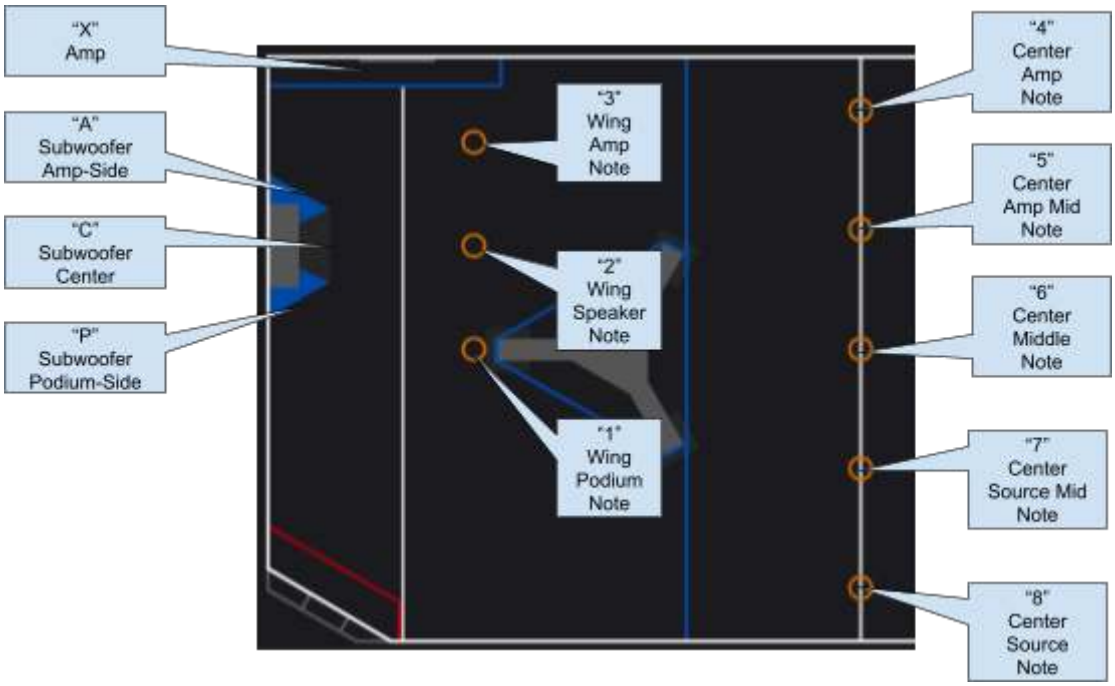
Drive to Amp

With drivetrain odometry getting periodic updates from the vision plus using PathPlanner's on-the-fly path generation, we were able to implement a "Drive to Amp" driver assist for teleop mode. When the main driver wants to score in the Amp, the right bumper on the controller now generates a path to the amp and commands the drivetrain to follow the trajectory. Pathplanner also automatically handles the required translation to handle red vs blue alliance field differences.

Autonomous

PathPlanner was used exclusively to develop the autonomous routines for Crescendo 2024.

Field Position/NOTE Naming Key



In order for the strategy and programming teams to communicate clearly and concisely regarding the required autonomous routines, we developed a naming/numbering scheme for the various locations on the field. Then, we used the sequence of letters and numbers to define the goal of the routine.

E.g. “C-2-3-1” means it is a 4-NOTE auto:

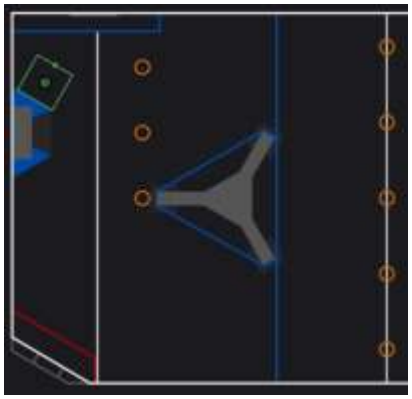
1. Score pre-loaded NOTE from the **C**enter of the Subwoofer
2. Pick-Up and Score NOTE “**2**” (Wing SPEAKER NOTE)
3. Pick-Up and Score NOTE “**3**” (Wing AMPNOTE)
4. Pick-Up and Score NOTE “**1**” (Wing Podium NOTE)

Routine Overviews

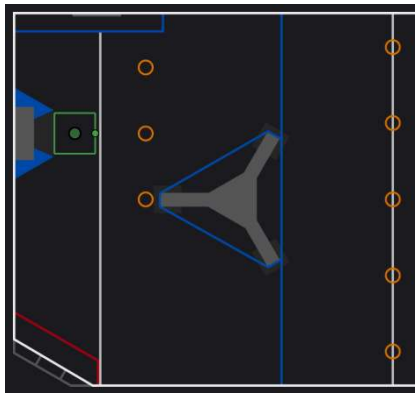
In order to develop the configurability and workability of our autonomous routines to work alongside other teams, we focused on developing several auto configs are outlined below

1-NOTE Autos

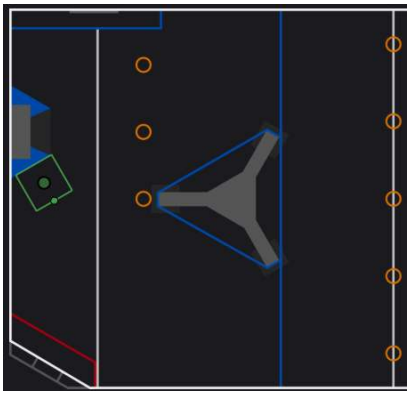
Score a pre-loaded NOTE, while not moving; ability to shoot from the stage from 3 different sides



Amp



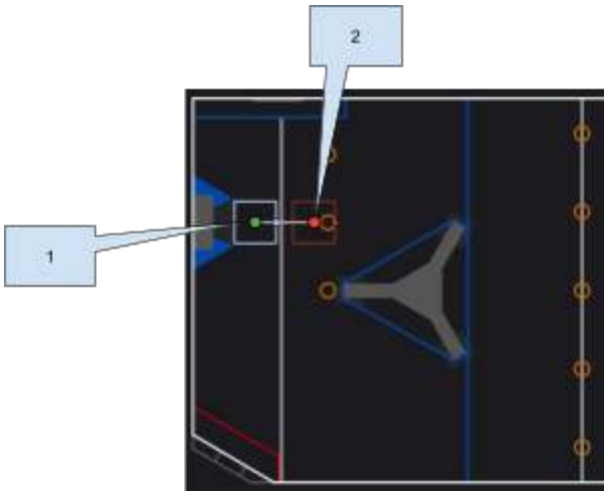
Center



Podium

2-NOTE Autos

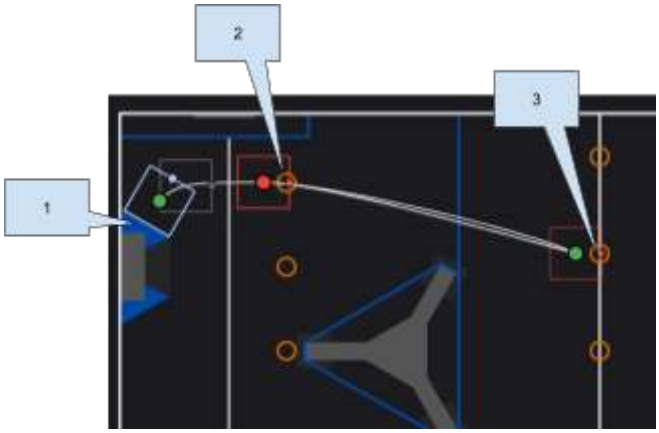
Score a pre-loaded NOTE, and then pick up & shoot the center NOTE



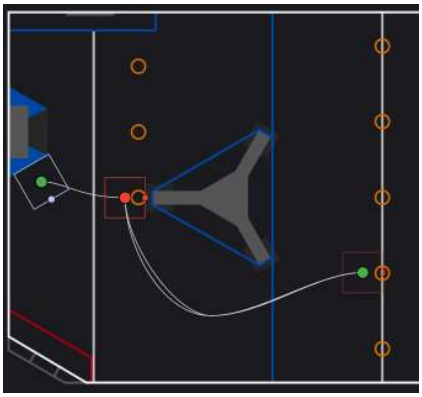
Center

3-NOTE Autos

Score a pre-loaded NOTE, pick up a NOTE from starting line & score it, then pick up a NOTE from the centerline & score it as well; 2 different routines



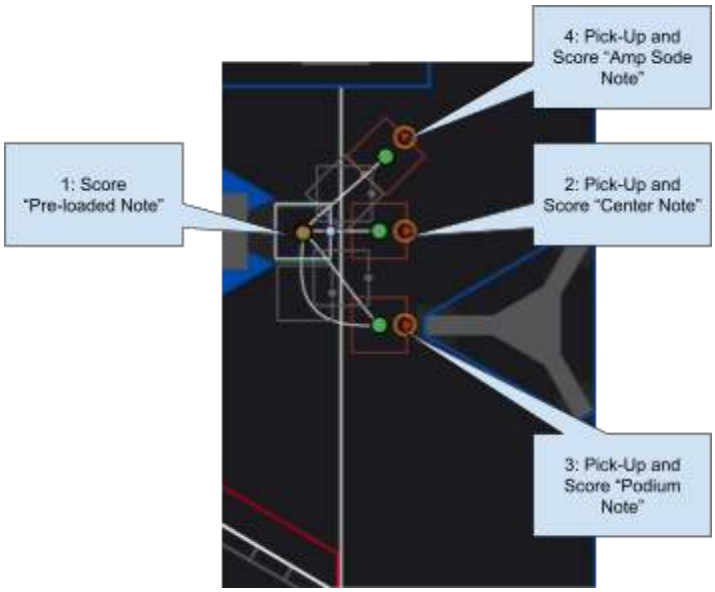
Amp



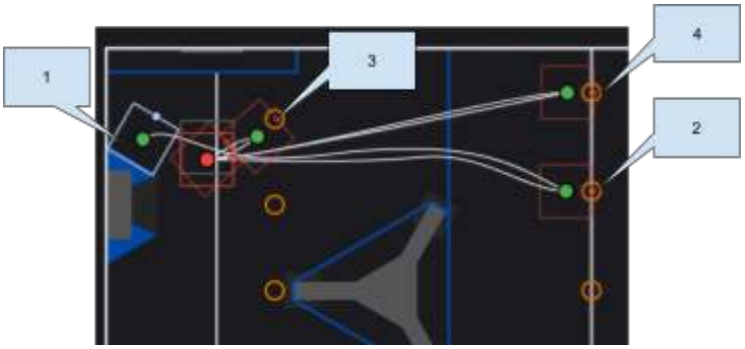
Podium

4-NOTE Autos

Score a pre-loaded NOTE, shoot from 3 NOTES from side of robot placed against the front/center of subwoofer or shoot from the side of robot placed against the AMPside of subwoofer



Center



Amp