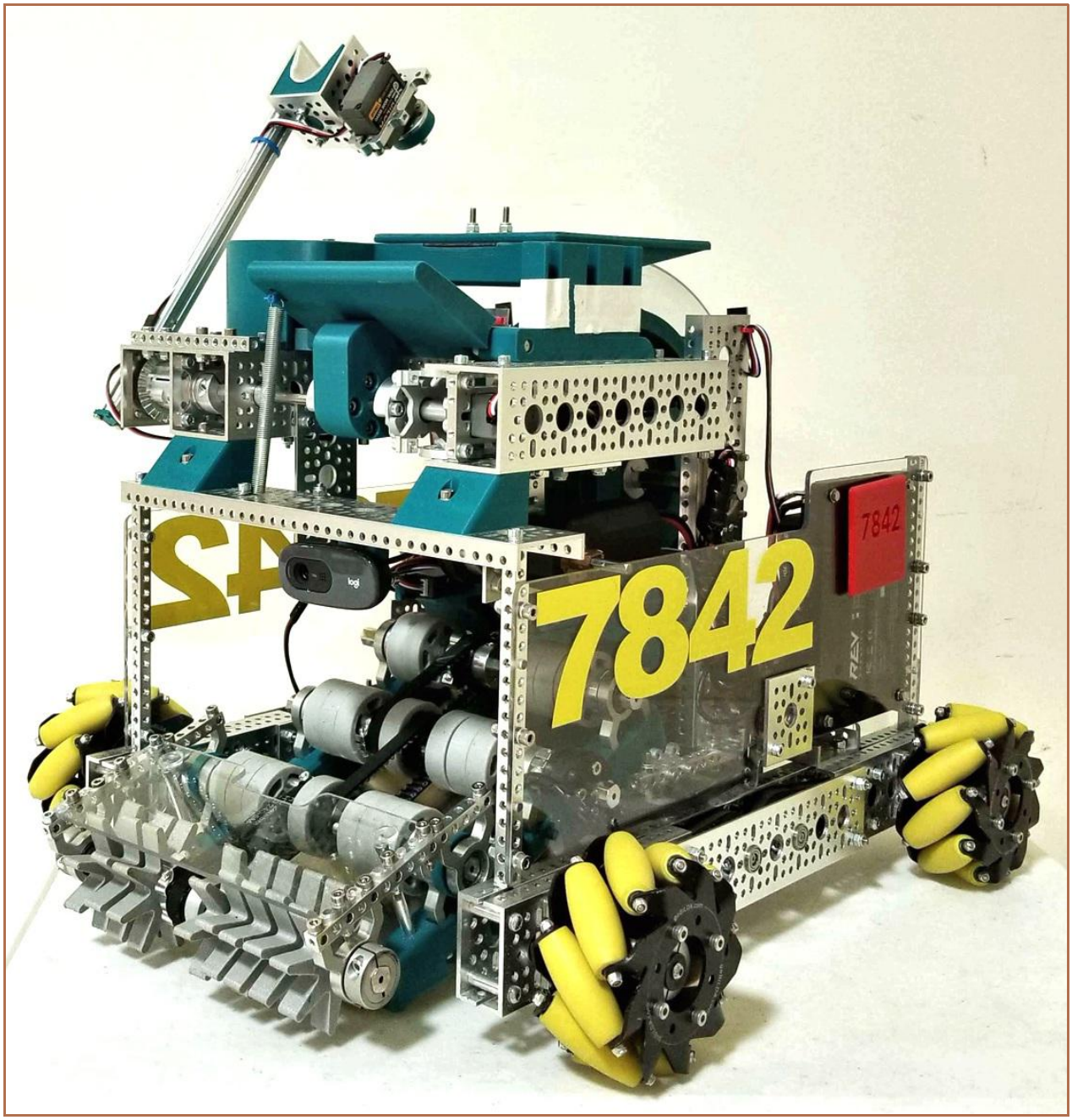
Browncoats

FTC Robotics
Team 7842

Huntsville, AL

Team 7842
Engineering Portfolio
2020-2021



MEET THE BROWNCOATS

The Browncoats are a community *FIRST* Tech Challenge robotics team from the Huntsville, Alabama area. This year, the team is made up of six students, ranging from 7th to 12th grade. These students represent public, private, and home schools from across the North Alabama area. Our team name came from our favorite science fiction television series, *Firefly*. This is our eighth year competing as a *FIRST* Tech Challenge Team.

3rd Year on Team

2nd Year on Team

2nd Year on Team

1st Year on Team

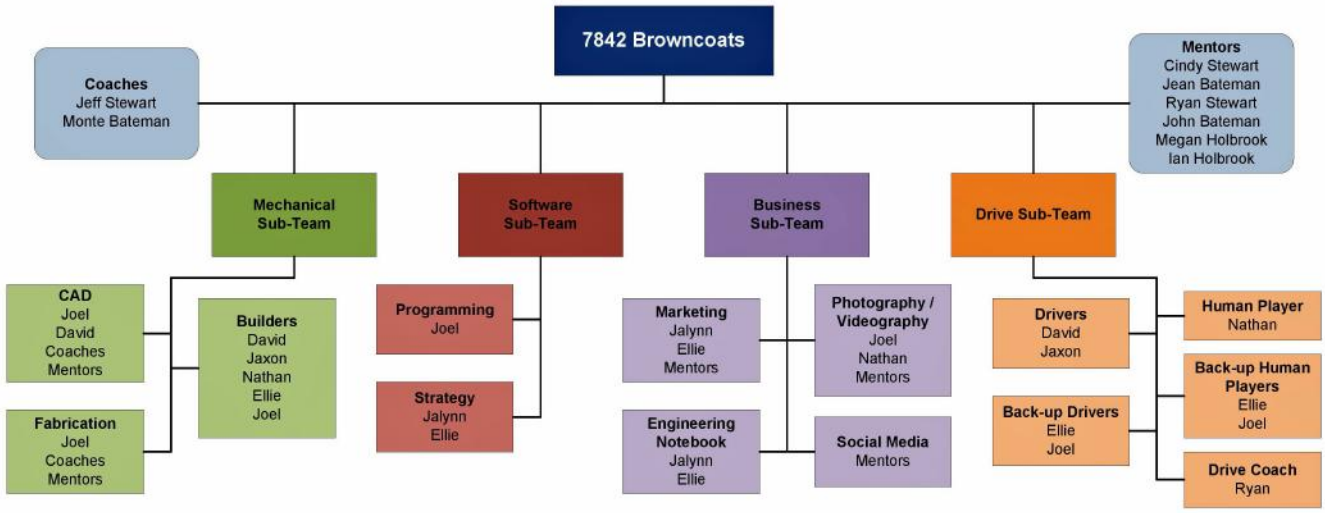
1st Year on Team

1st Year on Team

Team Organizational Structure

Our team has been fortunate to have mentors from many different fields, including a physicist who has worked with NASA; several Engineers who coach the students on engineering principles, computer programming, and construction; and former team members, currently in college, who share their experience with the current team. Team members take on the responsibility of managing the team, deciding roles, and completing project tasks.

The Browncoats are organized into four main sub-teams. Every team member has to choose one sub-team to be a part of, but they can opt to be a part of several, or even all of them, if they wish. The list of sub-teams includes Mechanical, Software, Business, and the Drive sub-teams.



TEAM PLAN

Sustainability Goals and Actions

Expand the *FIRST* Program in Alabama: One of our biggest goals as a team is to spread an awareness of *FIRST* and to help expand the program in Alabama. Every year we host informational meetings at our local libraries to inform people about the program and how to start a team. We also encounter many school teachers at our outreach events, and we give them materials to encourage them to start a team in their school district.

To sustain the program in Alabama, we mentor FTC teams to help them get on their feet and keep them going. We're always willing to help them if they need us for anything! We also do a lot of volunteering with *FIRST* Lego League; We love volunteering at their qualifiers and state competitions, and we've done FTC robot demonstrations encourage kids to join the next level of *FIRST*.

Host FTC scrimmages and build days for teams: We typically host build days and scrimmages, along with our end of season Rocket City FTC Invitational, to give teams more opportunities to interact and run their robots. We have found that this is a great way to encourage teams to share ideas and help one another out. The more teams meet throughout the season, the more chances we all have to improve!

Recruit and Retain New Team Members: One of the main reasons the Browncoats have been a team for eight full years is our recruitment plan. Our team makes a special effort to attend *FIRST* and STEM-related outreach events, reach out to local home school groups, encourage friends and family to participate, and host informational meetings at local libraries. To retain our members, we get them involved in the team as soon as possible. One of the hardest parts of joining a team is learning how to build and program once the season starts; so, early in the off-season, we begin offering classes to give the newer members a solid foundation. We have in the past taught building, programming, CAD, soldering, public speaking, and the engineering process. During the build season, we use a "divide and conquer" process and make sure that everyone is responsible for some part of the robot. We also pair our veteran members with our rookie team members to work on their subsystem assignments together as another way of training our new members, and hopefully better preparing them to design and build the next season's robot.

Fundraising: As a team, we work together to raise money and come up with ideas for fund raising opportunities. We've put together sponsorship packets that we take to local businesses and STEM companies, and we reach out to as many people as we can. Some of our methods for fund raising include:

- Yard sales
- Kroger's Community Rewards
- Amazon Smiles
- AUVSI Pathfinders Grant
- Facebook fundraisers
- Grants from local companies

OUTREACH

As a *FIRST* robotics team, we aim to embody the principles of *FIRST* and to expand the program in our community. Through our outreach events, we hope to inspire others to start their own team or participate in *FIRST*.

Engineering Connections: Due to the pandemic, we were unable to attend many of our regular summer outreach events such as the Space & Missile Defense Symposium, Huntsville Hamfest, and the AUVSI Pathfinder Symposium. But before the pandemic shut everything down, the Browncoats attended Robotics Day at the Space & Rocket Center hosted by Women in Defense. FRC, FTC, and FLL teams were given the opportunity to demonstrate and share their experiences with *FIRST*. Also, the team member who wrote our software last year graduated. Joel, who was our team photographer last year, agreed to learn and has been tutored by our software mentor, a NASA engineer, all season.

Giving back to FIRST: Giving back to *FIRST* is one of our highest priorities. This is usually accomplished by hosting build days and scrimmages to local FTC team and by volunteering at FLL events. This year, senior member Jalynn and recent alumni Megan collaborated to make a video for the Robotics Video Contest hosted by TN Valley Robotics. They were able to share the personal influence robotics and the *FIRST* program has had on each of their lives. Jalynn and Megan's video made the top 10 and earned $100 for the team!

Community Connections: We were able to hold an Informational Meeting at our local library before it was closed by the pandemic, and we held a robot demonstrations during our fundraiser yard sale and at one of our team member's school. At these events, we were able to introduce *FIRST* robotics to people by demonstrating last season's challenge with our competition robot, Vera, and letting kids of all ages drive our outreach robot. We also distributed flyers with information about our team and *FIRST* to all who were interested in robotics.

Community Service: Members of our team was able to provide Personal Protection Equipment to many different people for protection against COVID-19, including face shields, "stress relief straps", and face masks. The Browncoats 3D printed 50 face shields, over 1,100 straps, and sewed over 100 face masks, all of which went to local area hospitals, doctors, and grocery store workers. For the fourth year, the Browncoats held a Winter Clothing Drive where we collected bags and boxes filled with winter clothing and blankets for the Huntsville's Downtown Rescue Mission.

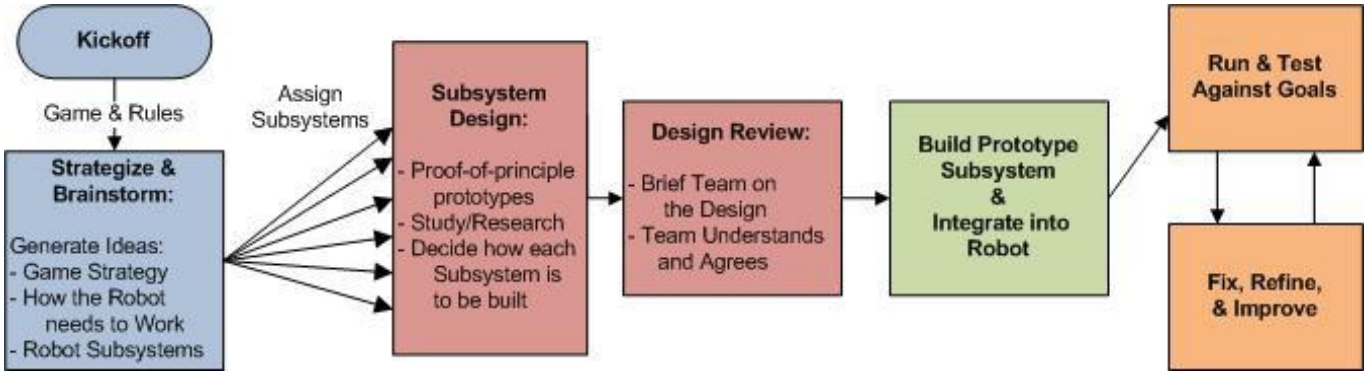
Informational Meeting at Library

Yard Sale and Robot Demonstration

Face Shield Frames

Winter Clothing Drive

Engineering Overview – Development Process

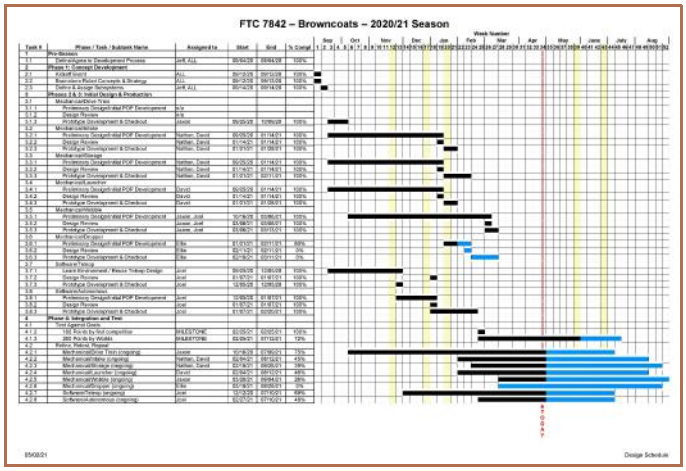


The Browncoats designed and built their robot using a simple but well-defined engineering process, which was agreed to prior to kickoff. At kickoff, the game and its rules were revealed to the team. Immediately afterwards, the team met and began brainstorming about how the game might be played (game strategy) and what the robot needed to do to play the game that way. Over the course of multiple sessions, the components of the robot (subsystems), and the jobs they needed to perform, emerged from the brainstormed ideas.

Each subsystem was then assigned to a student, or a small team of students, to design. Where possible, rookie team members were teamed with more experienced members so that they were helped through the process. This step of the process is where the "Divide and Conquer" process name came from.

As the design work was completed, each student or sub-team conducted a design review; typically a brief show-and-tell where each student shows off his preliminary prototypes, explains how his subsystem is going to work, and describes how it will be built. Once the entire team understands the idea and agrees, final prototypes of the subsystem were built and integrated into the robot.

The final steps of the process consist of a continuous improvement loop: The robot is run with an eye towards meeting team goals (scoring goals and the ability to execute the desired game strategy, for instance) and any identified fixes, refinements, and improvements are incorporated. Progress metrics were kept with a Gantt chart development schedule that was updated weekly. In addition, several team milestone goals were defined during the brainstorming (usually of the form of points scored by a specific date/event) and tracked throughout the season.



DRIVE TRAIN SUBSYSTEM ENGINEERING AND DESIGN

DESIGN DECISION: Compact, low profile chassis featuring Mecanum wheels and a timing belt transmission.

Pros: Speed and maneuverability. Plus, the team is familiar with this type of drive train from past seasons.

Cons: Opponents may be able to push the robot relatively easily. May be an issue while trying to score.

Solution: Since most, if not all, of the events this year will be remote, we don't believe this will be a problem.

ASSIGNED TO: Jaxon

DESIGN HISTORY:

Early in the summer, the decision was made to purchase and build this year's robot from goBILDA parts, thinking that the predominantly rookie team members would find them easier to work with. Mentors John and Ryan worked with the

team to assemble a drive train as a get-acquainted-with-the-new-build-system exercise.

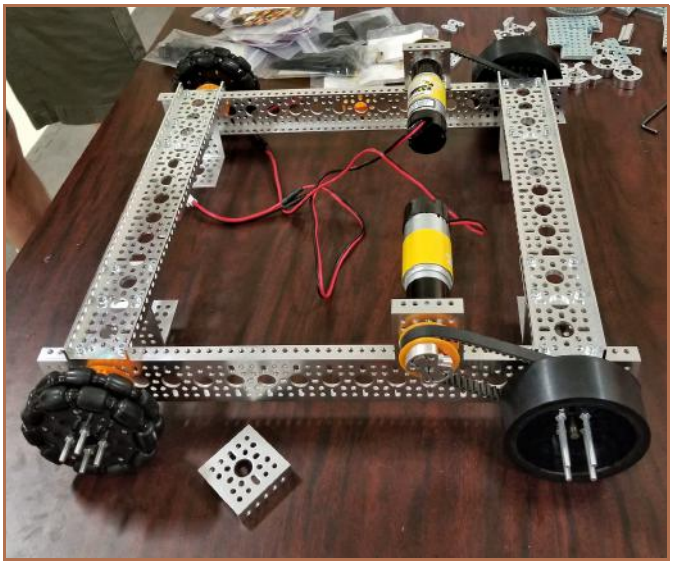
The first drive train was a pushbot with two powered Rhino wheels and two Omni wheels. Power was transmitted from the motors to the wheels with a set of old belts and 3D printed pulleys. Once driven, all agreed that compared to past robots, this drive train was crude, overly large, painfully slow, and not very maneuverable.

The second drive train featured sides of full-sized goBILDA U-Channels, four motors, mecanum wheels, and Bevel gear transmissions. Maneuverability was vastly improved due to the Mecanum wheels, but it was still very slow, more than 18 inches wide, and very heavy. The team determined that the slow speed was due to gearing; the Bevel gears provided a 2:1 speed reduction on top of the 20:1 gearing of the motors, and much of the weight was due to the four, 12mm stainless steel hex axles the wheels were mounted on.

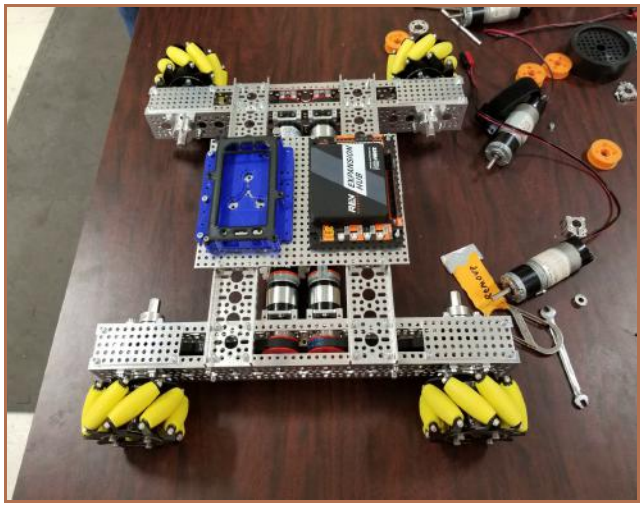
The third revision of the drive train featured timing belts running on new 3D printed pulleys as the transmission. The weight was reduced by substituting aluminum axles for the stainless steel, and low-profile goBILDA channel for the full-sized. This robot was faster and more responsive than the earlier models, but was still too wide to be FTC-legal.

The final revision of the drive train came about when mentor John demonstrated how CAD software (SolidWorks) could be used to design and virtually assemble mechanisms from STEP files of the goBILDA parts. This led to an optimized design that incorporated thrust bearings to reduce the friction caused by the Mecanum side-to-side motion and a very narrow beam profile just wide enough for the pulleys and timing belts. The resulting chassis retained the center mounted motors for balance and easily fit within the 18-inch FTC size constraints.

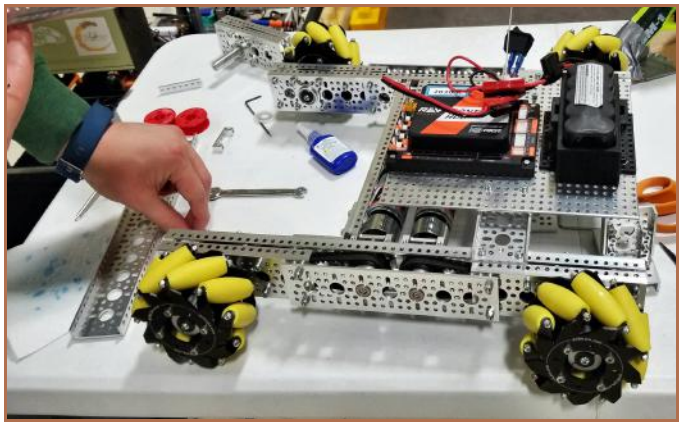
After kickoff, the team decided to reuse the summer project drive train in the competition robot-they were that happy with it. All fasteners were treated with thread locker and, based on past team experience, the transmissions were tweaked by changing the pulleys to 24-teeth on the motor and 22-teeth on the drive axles, yielding an overall 17.6:1 gear ratio. Testing showed that this arrangement provided a good balance between acceleration and speed.



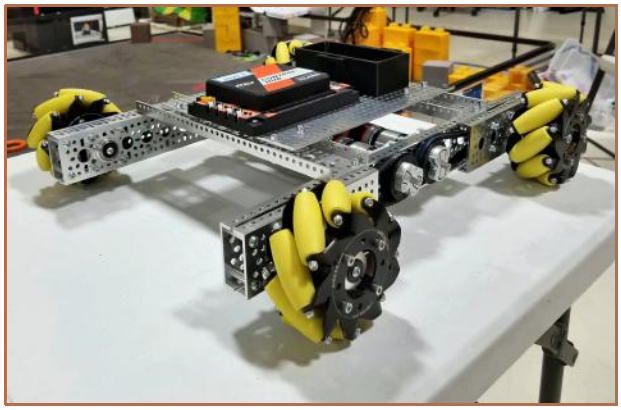
First Drive Train prototype



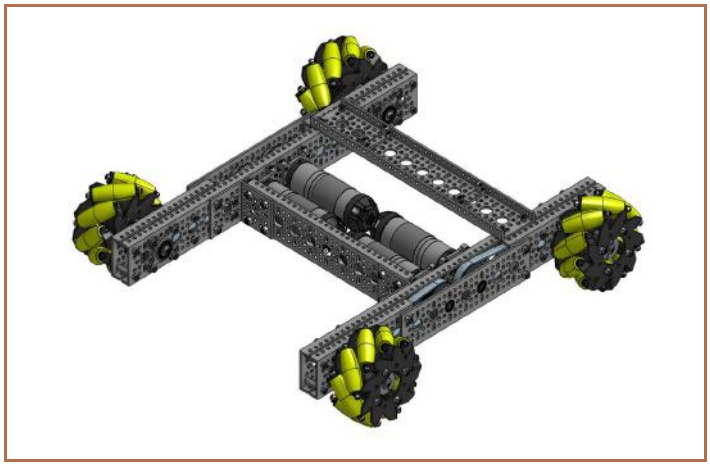
Third Drive Train prototype



Final Drive Train, under construction



Final Drive Train, nearly complete



CAD rendering of the Final Drive Train

INTAKE SUBSYSTEM ENGINEERING AND DESIGN

DESIGN DECISION: Mechanism made up of a series of flexible wheels that rapidly pulls rings into the robot.

Pros: Speed

Cons: Can only pick up rings from the front of the robot

Solution: Driver practice

ASSIGNED TO: Nathan

DESIGN HISTORY

Nathan began with a concept similar to last year's robot, with side mounted wheels pulling rings into the robot.

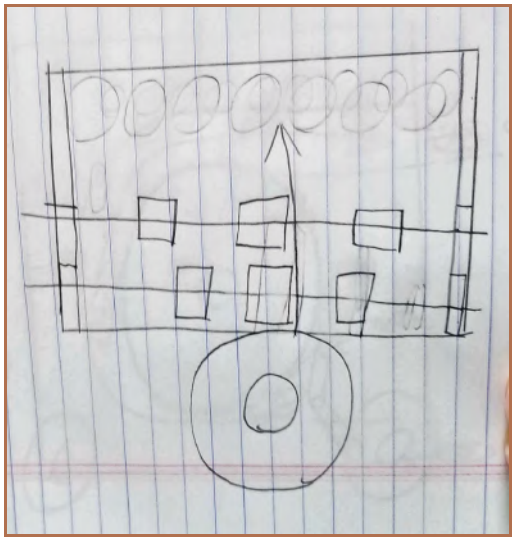
Through a series of proof-of-principle models, the design evolved into a top-mounted set of grippy, conformal wheels that slide rings into the robot and up a ramp. The models were also used to experiment with several different types of wheels the team had on hand. Two inch diameter, gray VEX flexwheels were eventually chosen for their grippiness and ability to pull the rings.

The proof-of-principle model eventually evolved into a motorized, belt-driven device that used several sets of flexwheels to slide rings up a long, aluminum ramp made of goBILDA parts. From this final model, the width of the intake was set at about four inches, and the distance between the wheels and the aluminum slide to 18mm.

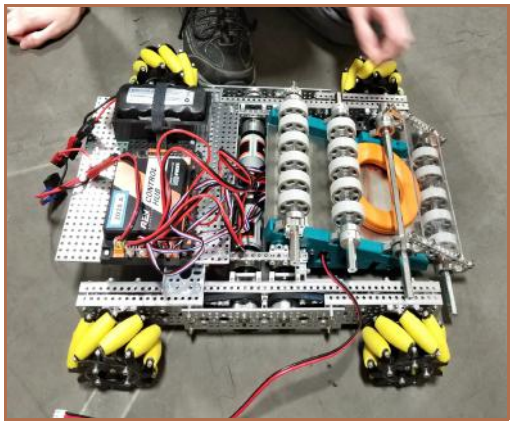
The final design consists of two 3D printed rails which provide properly spaced mounting points for goBILDA pillow blocks and a smooth aluminum ramp. The ramp is long enough to lift the rings off of the floor and over the drive train motors. The pillow blocks hold four axles, onto which the flexwheels are attached. The first axle is hinged and extends out in front of the robot, so as to better reach rings and pull them in toward the ramp. The drive motor for the mechanism is housed under the ramp, and timing belts are used to distribute the power.

While testing the final design, it was found that the edge of the ramp almost had to scrape the mat for the rings to smoothly slide onto it. This was a problem, since if the ramp was too high the rings would hit the blunt edge and refuse to go up the ramp, but if the ramp was too low it could scrape and damage the field tiles. The solution eventually adopted was to allow the ramp to ride just off the mat, but a roller made of a 3/32" aluminum tube was suspended just in front of it on a fixed, music wire axle. The rings slide over the roller and up the ramp with very little encouragement.

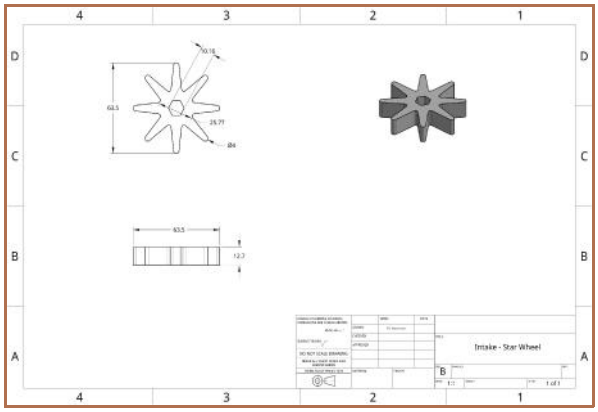
Improvements added to the final design included a color sensor to detect when rings have been picked up and the addition of star-shaped wheels, 3D printed from flexible TPU filament, to the first stage of the collector, improving it's ability to quickly snag rings, even when they get dirty.



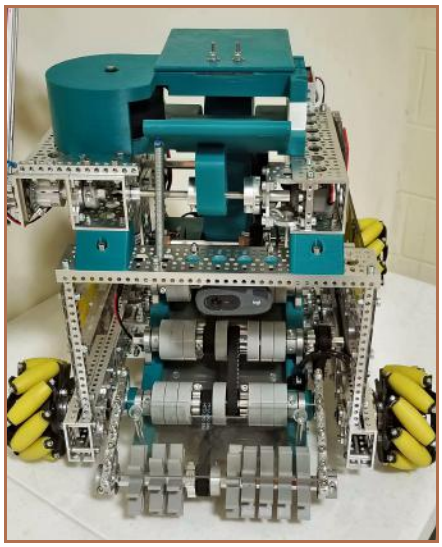
Nathan's intake idea



Initial testing of a prototype Intake



CAD drawing of the Star Wheels



Robot sporting new Star Wheels

LAUNCHER SUBSYSTEM ENGINEERING AND DESIGN

DESIGN DECISION: Single flywheel based mechanism that launches rings at the goals while imparting spin for stability. A software-controlled flap is used to vary the height of the launched rings.

Pros: Simplicity of the hardware.

Cons: Sometimes inconsistent in how far and high the rings are thrown.

Solution: Implement a PID controller in software to stabilize the velocity of the flywheel and make the shots more consistent.

ASSIGNED TO: David

DESIGN HISTORY:

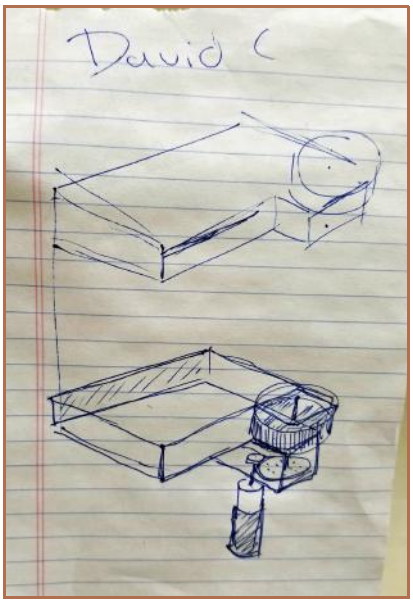
During the initial brainstorming, a flywheel-based launcher was suggested, mostly due to the success the team had with a similar mechanism in the Velocity Vortex season. It was felt that a single-flywheel design would work best, since it would impart spin to the rings and stabilize them in flight.

Several prototypes of a flywheel mechanism were experimented with. Most consisted of a goBILDA hardware frame, a NeverRest 3.7 Gearmotor, a 96mm goBILDA Rhino wheel as the flywheel, and a transmission containing goBILDA gears arranged in various configurations. Most of these prototypes were found to be impractical—they spun much too fast, throwing the rings much too far to be legal, or the hole spacing between the limited set of goBILDA parts from which the frame was build was always off. In addition, the exposed gears of the transmission proved to be rather dangerous around fingers. These prototypes did help David determine the spacing between the flywheel and the opposite, static wall that yielded the best range and spin--about 120mm.

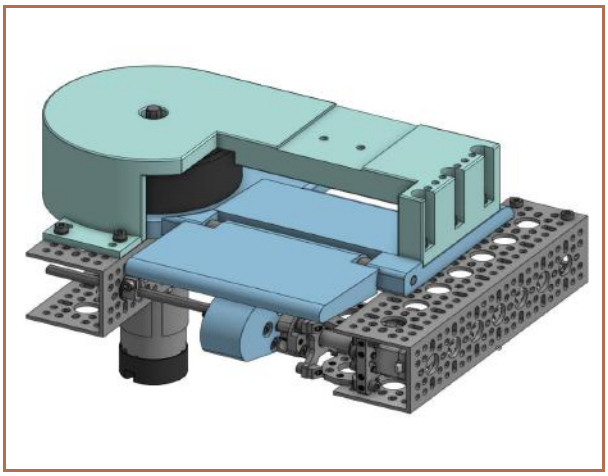
To simplify mounting, the team purchased a direct-drive goBILDA motor, specifically intended for this kind of application. When the prototype was rebuilt, several new things were tried, including mounting two Rhino wheels on top of one another in an attempt to increase the mass of the flywheel (spin-up took much too long) and adding a very long ramp, made of goBILDA hardware and cardboard, so that the effects of launching multiple rings rapid-fire could be checked (the rings were launched very consistently). Also tried at this stage was the addition of a flap to the launcher's exit, which could redirect the rings into an adjustable, higher trajectory and help with aiming for the goals.

David began sketching how he envisioned a finished launcher would look and fit onto the robot, and eventually transferred his sketches to CAD (Onshape), which he was using in one of his school classes. Initially, the Baseplate was to be a relatively simple piece cut from Delrin plasic, chosen for its smooth, slick surface. However, in CAD, the design quickly evolved into not just a Baseplate, but also incorporated a safety wheel cover, the motor, the flywheel, the stationary wall, hinge points for a servo-controlled flap, a notch for the Storage System's wheel to pass through, and mount points for attaching to the metal robot frame. This became too complex to cut out of Delrin, so the parts were 3D printed.

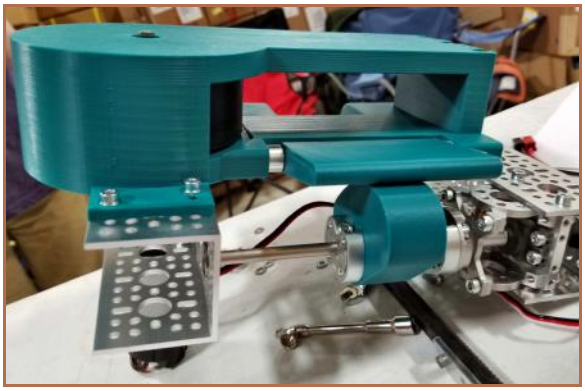
Improvements eventually incorporated into the design included a 5 degree upward slope of the baseplate; a 3D printed cam, driven by a servo, to control the height of the flap; narrowing the space between the baseplate and the roof of the wheel cover so to more consistently align the rings as they were being thrown; and the addition of fastening points for the Storage Subsystem's Lexan strap.



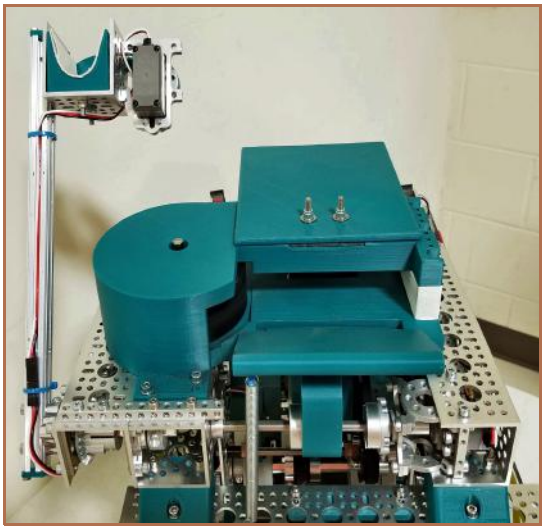
David's original drawing of his launcher design



CAD rendering of Launcher Subsystem



Early 3D printed Launcher Subsystem



Final Launcher installed on the robot.

STORAGE SUBSYSTEM ENGINEERING AND DESIGN

DESIGN DECISION: Mechanism that receives rings from the Intake Subsystem and stores them on a large wheel. Capable of rapidly feeding the stored rings to the Launcher Subsystem.

Pros: Simplicity

Cons: Rings can jam as they are passed from the Intake, which moves rings very quickly, to the storage wheel, which runs much more slowly.

Solution: Sensors detect when rings are in the Intake and the position of the teeth on the storage wheel. Software synchronizes the hand-off of rings between the two subsystems.

ASSIGNED TO: David and Nathan

DESIGN HISTORY

During brainstorming, a large wheel was proposed to transport the rings from the front-facing intake up to a forward firing launcher on the top of the robot. The team wanted both of these mechanisms to face forward so that the drivers could clearly see them and to minimize the number of times the robot had to turn around during the course of a match.

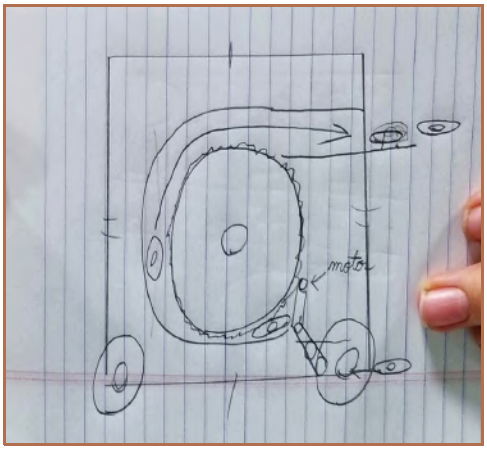
A mentor brought in an eight inch plywood disk mounted on a stand (used in past seasons as a gear-ratio demonstration) so that the team could see how disks might fit on the outer perimeter. This visual demonstration inspired David to sketch the initial first storage wheel design. It featured a 7-inch diameter, 2-inch wide, spoked wheel with a curved surface to fit the interior contour of the rings and to better hold them in place. The wheel was designed to include an integral large diameter timing belt drive pulley. The diameter was specifically chosen to balance the number of rings that could be stored and the robot interior volume required to house it. The original concept called for the outer edge of the wheel to be covered with microfiber carpet tape, so as to grip the rings and hold them in place as the wheel turned.

After the first wheel was 3D printed and could be handled, several improvements were suggested, including the addition of four short, rounded teeth around the outer perimeter. The new teeth inspired the team to rename the part the "water wheel".

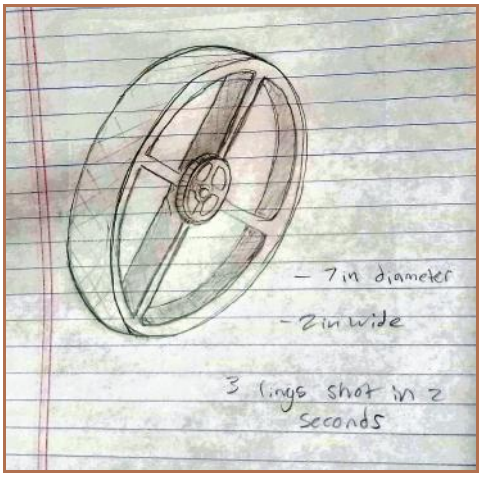
To better envision how the water wheel would fit inside the robot, it was mounted on an axle and suspended between two 9-hole goBILDA low-profile U channels equipped with bearing blocks. The team began experimenting with a couple of different ideas about how to hold the rings against the wheel, and settled on the idea of using a thin Lexan strap bent around the contour of the wheel.

The next major revision of the wheel was the replacement of the four short, rounded teeth (which were found to be largely ineffective) with eight larger, saw tooth shaped teeth. These teeth had a large flat surface that could jam rings into the Launcher mechanism with some force.

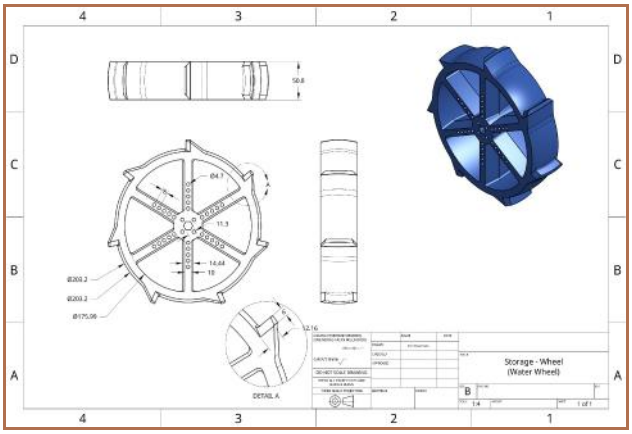
Over time, several improvements have been made to the subsystem. It was found that rings were likely to jam as they were handed off between the Intake and Storage subsystems because of the difference in speed that the two subsystems operated at. To address this, the water wheel was changed to a seven-tooth design which better nestles the rings and the wheel was sped up by changing the drive pulley from 80 to 50 teeth. A color sensor was also added so that the software could detect the position of the teeth and coordinate the hand-off of rings between the two subsystems.



Nathan's original Storage Subsystem concept featuring a large storage wheel

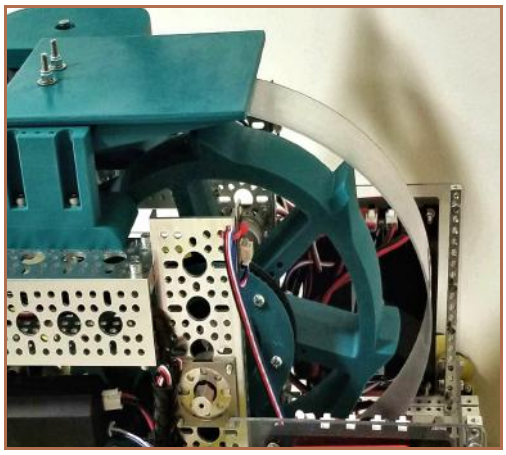


David's original water wheel design



CAD drawing of "water" wheel

Visual Demo: 8-inch plywood disk and stand



The final Storage Subsystem integrated into the robot

WOBBLE GOAL SUBSYSTEM ENGINEERING AND DESIGN

DESIGN DECISION: A pivoting arm and grasping mechanism that can catch Wobble Goals by their masts, lift them, and deliver into either a Target Zone or over the field wall into the Drop Zone.

Pros: Simplicity of the hardware.

Cons: Difficult to precisely position the arm to capture the Wobble Goals.

Solution: Software automation positions the arm to pre-programmed positions. Driver practice.

ASSIGNED TO: Jaxon and Joel

DESIGN HISTORY

During brainstorming, it was realized that a mechanism would be required that could lift Wobble Goals over the field wall. Another consideration was the ability to right tipped-over Wobble Goals. At kickoff, it was discovered how easily the Goals could tip over and lay on their sides, and also how difficult it could be to right them without tipping them onto their opposite side.

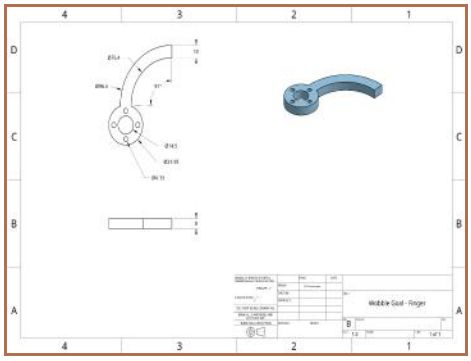
Many different mechanisms were considered, including servo-actuated fingers designed to right tipped over Wobble Goals; forks that would slide under and lift the base of the goals; a horizontal arm with a rotating wrist that could grasp

the mast of the goal and rotate the base up high enough to get it over the wall; and a multi-jointed arm with a very human-arm like motion. Several rough prototypes were built but all had some flaw, such as requiring too many motors, being too heavy or bulky, or requiring a mounting point on the robot that was already occupied by another mechanism.

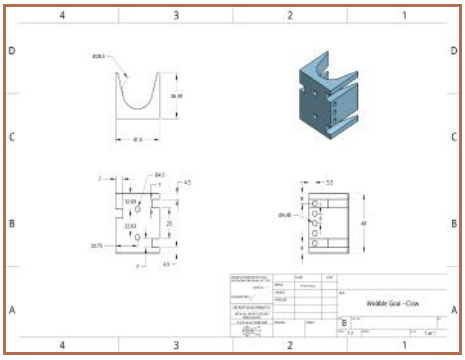
Quite a bit of thought also went into the grasping mechanism, including a multi-fingered claw, a very wide V-shaped funnel into a simple clamp, a device inspired by a Chinese Finger Trap that would use the weight of the Wobble Goal to grasp it, and a simple socket shaped to form-fit the goal's mast with a single servo-actuated finger to hold it in place. This final, simple approach was ultimately chosen.

The final arm design is also very simple: A 12 inch length of 15mm aluminum extrusion mounted on a bearing-supported shoulder, driven by a 30 RPM motor and a goBILDA Bevel gear set all mounted on the upper superstructure of the robot. The arm can be rotated back to stow within the FTC 18-inch size limit and rotates over the top of the robot to reach forward and grasp Wobble Goals by their masts. The shoulder is powerful enough to lift the Wobble Goals, making deposits into Drop Zones a very simple matter.

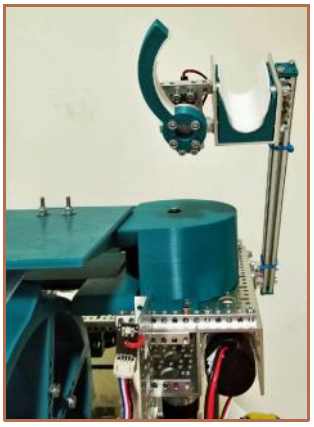
An improvement made to this subsystem was the addition of a magnetic limit switch that can detect when the arm reaches its home position. This allows the software to better control the arm and prevent it from jamming into the robot or the floor.



CAD drawing of finger



CAD drawing of claw



Final Wobble Goal Subsystem on the robot

SOFTWARE SUBSYSTEMS ENGINEERING AND DESIGN

DESIGN DECISION: Development of Java software in the Android Studio environment that allows the robot to navigate during Autonomous and allows the drive team to operate during Teleop.

Pros: Use of Android Studio made software reusable from past season's work. It is also the most capable environment for implementing complex software, including machine vision.

Cons: Android Studio is also the most difficult programming environment to use and the assigned student was new to the job this season.

Solution: A great deal of mentoring and software expertise was available.

ASSIGNED TO: Joel

SOFTWARE REUSE:

The Mecanum drive train software algorithms were reused from last season's robot.

Ring height detection using the Webcam was accomplished with OpenCV and started with an approach by team 9794, Wizards.exe (EasyOpenCVExample). Their example pipeline was refined, streamlined, and clarified for use in Vera.

Launcher flywheel motor PID tuning was accomplished using a standalone OpMode (VeloPIDTuner) posted by Noah Bresler (team 10940), utilizing the FTC-Dashboard and RoadRunner packages.

LEARNING AND MENTORING:
The student assigned to work on the software subsystems this season was Joel. He already had some experience with PyCharm, a modern Integrated Development Environment (IDE) for Python, but had no experience with Java, programming FTC robots, or with Android Studio.

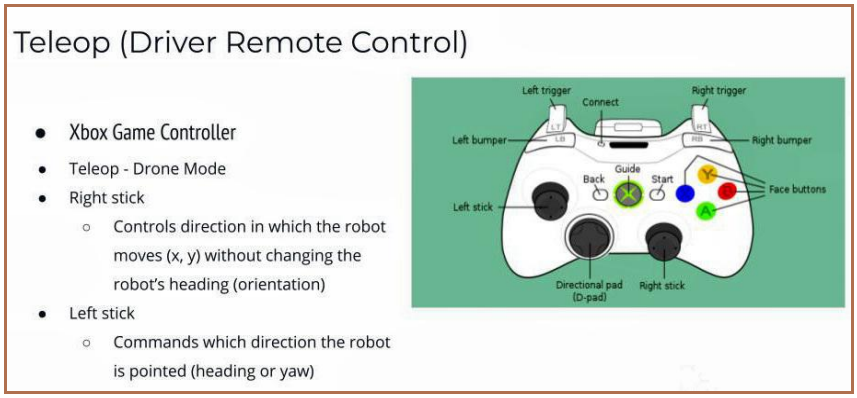
Joel and his software mentor (his Dad) were provided with the on-line book "Learn Java for FTC" by Alan G. Smith and with a Hardware Test Bed, which included a REV Control Hub, an Android phone, a game controller, a motor, servo, and an assortment of sensors. Joel worked closely with his software mentor to study the book and to work through all of the examples in the first 12 sections. From this work, Joel became familiar with how to add and control hardware through software as well as how to properly structure the Java for maintainability (i.e., the concepts of abstraction and compartmentalization).

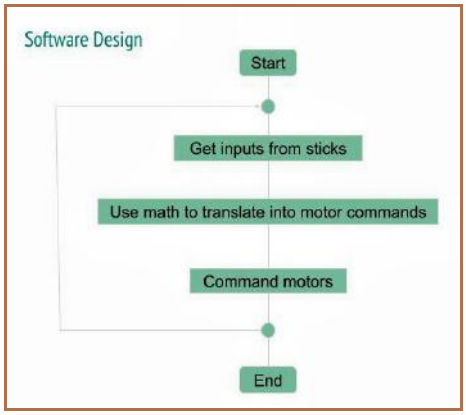
The Android Studio Guide was used as a reference resource to fill in gaps not covered in the book and to ensure currency with the 2020/2021 library requirements.

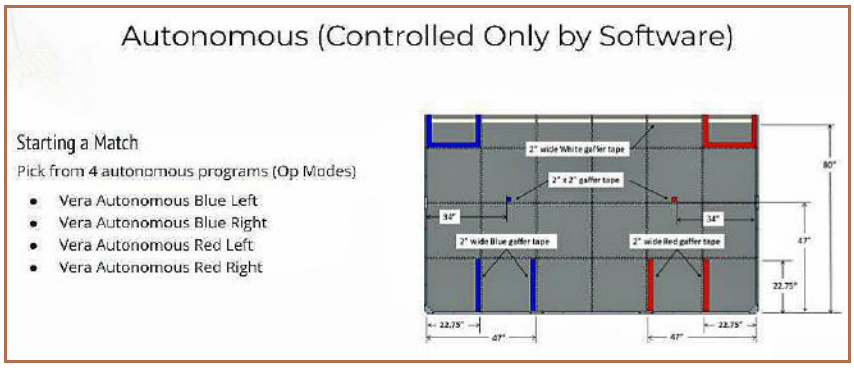
Since Joel was brand new to *FIRST* robotics programming, his mentor assisted with the design of the more complex portions of the code (such as the state machine and machine vision). The goal of the mentor was to help Joel code from scratch as much as possible in straightforward, understandable chunks so as to maximize his learning.

DESIGN HISTORY:
Early software design work focused on learning Java, then building basic but very useful Teleop and Autonomous OpModes. The Teleop OpMode allowed driver control of the bare robot Drive Train, using the joysticks on the game controller as inputs. The first Autonomous functionality used the drive motor encoders for speed and distance control, and allowed the robot to navigate to and park on the Launch Line from any of the possible starting positions.

Joel presented the following charts at his software design review.





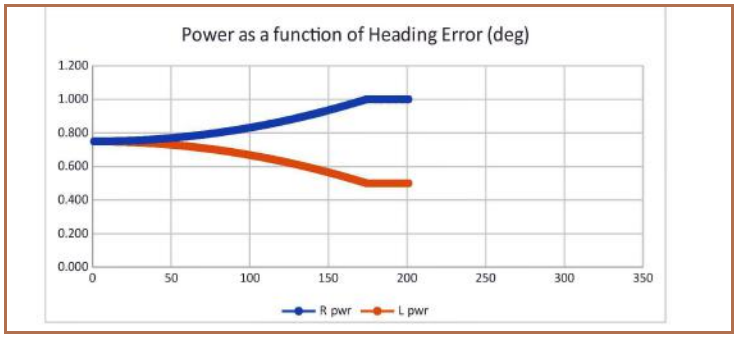


Following the design review, Joel and his mentor worked to develop and add numerous capabilities and refinements to the software base, and also added the necessary hardware control functionality as the mechanical robot subsystems were completed. These capabilities included:

Game Controller: During the design review, an initial set of controller button assignments were proposed to the team. As more hardware subsystems were added to the robot, button assignments were added and reassigned based on feedback from the drive team.

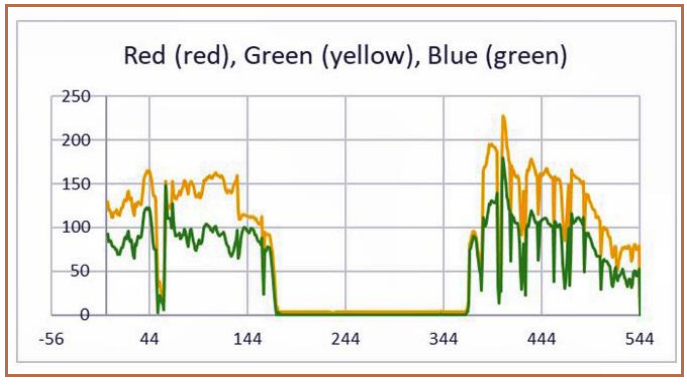
Machine Vision - Ring Detection: Once the Webcam became available, an OpenCV pipeline was built to implement the ring height detection capability required by autonomous. This pipeline extracts a small rectangular region where the ring stack is expected to be and takes the average of how much "Red" (close enough to Orange for our purposes) appears in that region. Comparing that average to experimentally determined thresholds determines whether zero, one, or four rings are present.

Autonomous Navigation: Each Autonomous route was choreographed as a simple series of discrete, sequential steps. Functions were written to "Rotate to Heading (degrees)", "Drive to a Field Position (X, Y)", "Drive Forward/Backward (distance)". The Control Hub IMU provides heading information, and the average of the four drive motor encoders are used to track distance. Joel came up with the mathematical approach to steer the robot in a straight line using IMU heading error as feed back into the commanded power for the left or right motors (see the Error Squared relationship graph). In addition, a trapezoidal acceleration/ cruise/deceleration curve is used to ensure that the robot does not start, stop, or travel fast enough for wheel slippage to degrade the positional accuracy on the field.



Error squared relationship of motor power inputs to IMU heading error

Machine Vision – Aiming: The web cam is also utilized to aim the robot when launching rings. The approach taken is based upon the how very black the low goal is, and that nothing as black and wide would appear in the camera's field of view when pointed at the goals. To detect the goal, a narrow horizontal band of the image that cuts across the low goal is processed and a contiguous stretch of black is searched for. If that band is large enough, the tower goal is considered detected.



Collected RGB Data showing a clearly distinguishable black region of the Tower Goal

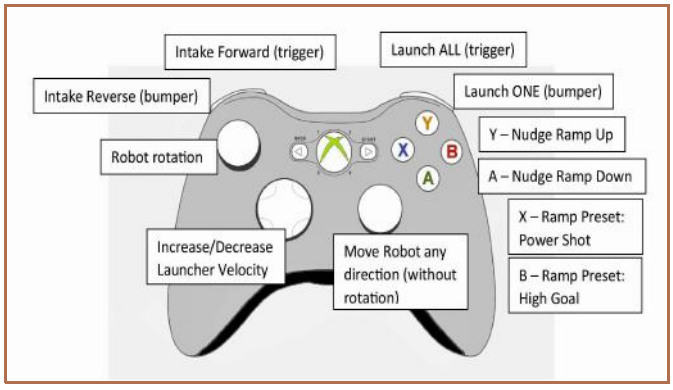
Intake, Storage, and Launcher Subsystem Integration: A state machine is used to control the ring-handling interactions between these three subsystems. Sensors on the Intake and Storage subsystems detect when rings are ingested and where the teeth of the water wheel are with the goal of preventing jams as rings are handed off from one subsystem to the other. Several other capabilities are also incorporated into the state machine, including the ability to initialize the system; allowing the drivers to command rings to be fed to the Launcher subsystem; allowing the drivers to nudge up or down the velocity of the Launcher flywheel; and a fail-safe system "clear" that should remove any rings that get jammed or stuck.

Both the Launcher Flywheel motor and the Storage Wheel drive motors were PID tuned to provide precision control for velocity and position respectively.

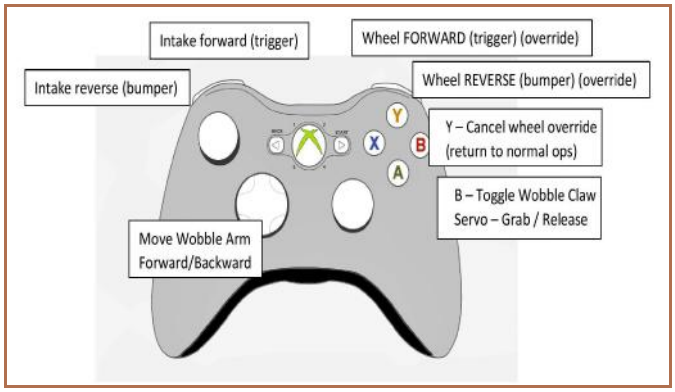
Launcher Flap: The flap is driven by a servo and controls the elevation that rings are launched at. Preset elevations for the High Tower Goal and the Power Shot Targets can be selected by buttons on the game controllers. In addition, there are buttons that permit the drivers to nudge the flap up and down for fine tuning based on robot performance.

Indicator Lights and Telemetry: Information is reported to the drivers via indicator LEDs on the robot and by telemetry text printed on the Driver Station display:
LEFT LED/Teleop: Indicates the number of rings the robot has loaded (Off=0, Red=1, Amber=2, Green=3).
LEFT LED/Autonomous: Indicates the number of rings detected (Off=failed, Red=0, Amber=1, Green=4).
RIGHT LED/Teleop: Off=Tower Goal not detected, Amber=detected but not aimed, Green=aimed.
RIGHT LED/Autonomous: Green=Goal detected, Red=Goal not detected, aiming based on IMU alone.
Driver Station Telemetry: Messages include Tower Goal detected/not detected, Tower Goal estimated distance, Aiming error in pixels and degrees, Launcher Flap position, and Launcher flywheel commanded/actual velocity.

Wobble Goal Subsystem: The secondary game controller operates the arm motor and a button toggles a servo to open and close the gripper. The software limits the travel of the arm to prevent the arm from going too far back and striking the robot or too far forward to strike the ground.



Current Design: Game Controller - Primary Driver



Current Design: Game Controller - Secondary Driver