Checkmate 12961 Portfolio

Table of Contents

Robot Design and Process

Team Summary p 12

Outreach to the Community

Connecting with Mentors and FIRST Community

Business Plan 15

HIGHLIGHTS

Checkmate Current Design Summary

Our robot is designed for high efficiency ring scoring. With an incredibly versatile drivetrain, a speedy intake and our high accuracy flywheel shooter, we quickly score rings in the high goal.

Intake

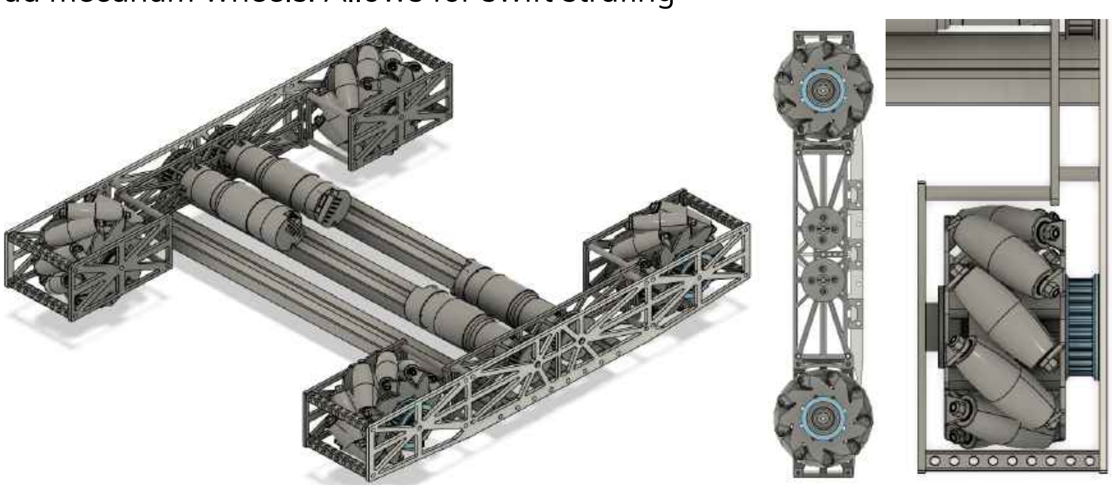
Our intake mechanism is ***quick*** and requires little precision to accurately pick up rings, allowing drivers to quickly pick up rings without manipulating the robot much.

- One long shaft spins 4 evenly spaced 1.625” Vex compliant wheels
    - Leads directly into bottom roller that lifts ring up into transfer
- Custom 3D printed structure
- 435 rpm motor
- Rings do not jam up in the intake

Drivetrain

Our custom fabricated drivetrain is very simple and robust. It was designed for strength and longevity.

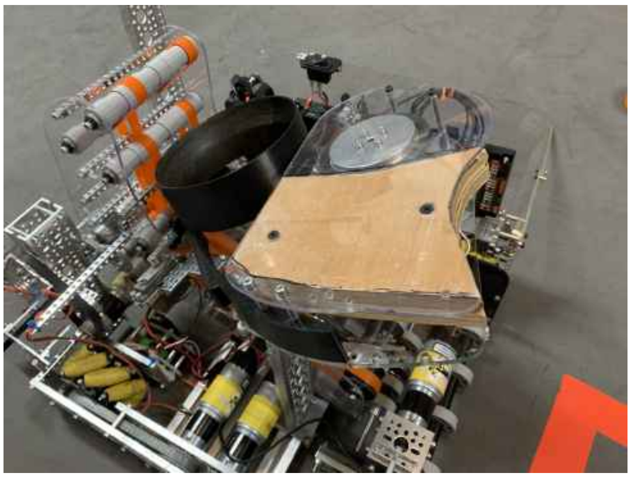
- Based on modular parallel plate design
    - Allows plates to be redesigned and remade to suit our needs
- Indirect 1:1 belt drive + 19.2:1 planetary motors (312 rpm)
    - Custom 3d printed belt pulleys
    - Dead shaft wheel design
- GoBilda mecanum wheels: Allows for swift strafing



Shooter

Our shooter is created with a 96mm GoBilda Rhino wheel, a 3D printed 45 degree track and a sheet of custom cut acrylic.

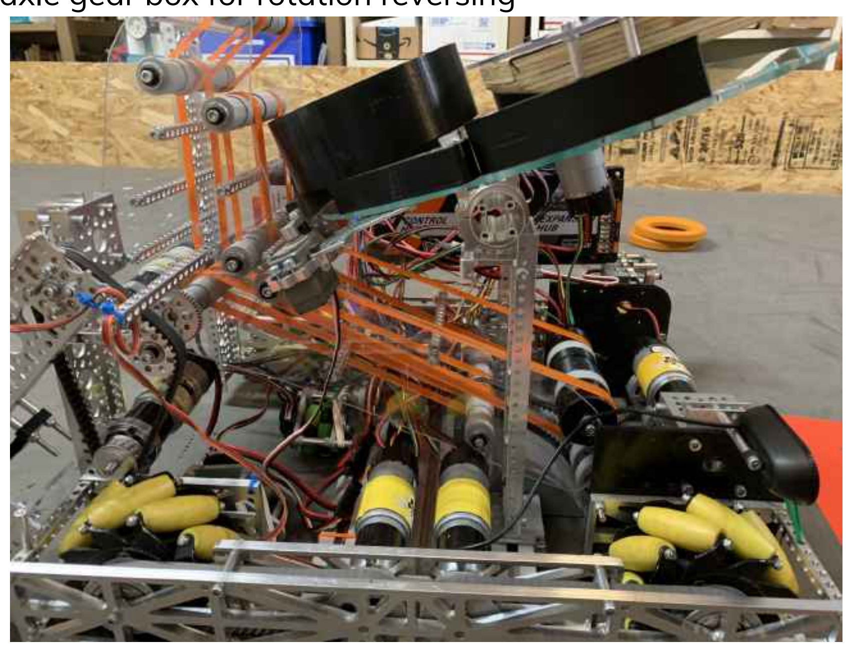
- Two modes
  - High goal and power shots
- 5800 rpm effective flywheel speed
  - Motor encoder to control precise rpm of flywheel
- 3D printed ring pusher for smooth shooting
- Attached with angle adjustable GoTube for easy tweaks




Transfer Mechanism

Our transfer mechanism consists of two curved tracks of polyurethane belting to form a fast and effective conveyor that serves the purpose of moving rings from the intake to the flywheel hopper.

- Custom 3D printed rollers
- Custom fabricated polycarbonate sheet structure
- Powered entirely by one 435 rpm motor
  - Flat belting serves as power transfer between sets of rollers
- Custom dead-axle gear box for rotation reversing



Checkmate Design Process

Our design process from season to season has remained consistent. However, this year we have largely met virtually or in small groups, and this has forced a more linear design process. We still put extreme emphasis on foresight throughout our design process. We have learned throughout the team's 4 years of experience in FTC that thinking about potential future designs is imperative to having a good robot. We continue to use a "design funnel", which entails brainstorming, research, prototyping, and testing and improvement through multiple iterations.

Problem Identification

Identifying problems is the first step in our design process. We carefully define what the problem is and assess priorities within the finite set of problems.

For the Ultimate Goal season, these are the problems that we quickly identified: After the season kickoff, we came together and compiled a list of robot requirements as well as what to prioritize.

- Autonomous:
  - Powershots or high goals at the start.
  - Hug the wall and park over the launch line.
- Tele-Op:
  - Launch as many rings into the high target as possible.
  - Stack 3-6 rings on wobble goal.
  - Gather 3 rings for endgame.
- Endgame:
  - Lift wobble goal out of field
  - Powershots

Information gathering

Gathering information is the second crucial step in the design process. Researching existing solutions to the problems we've already defined is a method that we have found very productive for idea generation. We often look to industrial machinery for ideas for our robots, and most of the time we find concepts that work well with our problem.

These are the things that we have researched so far this season:

- Polyurethane piping and ultimately used urethane belts
- Spindle/wheel designs for belts and ultimately designed grooves in our spindle design
- We reviewed the design of each robot component for electrostatic discharge problems we encountered last season
- Code algorithms for pathfinding and settled on RoadRunner
- Vision detection systems and settled on using OpenCV
- Motors for the shooter and settled on a bare DC motor from GoBilda that runs at about 6000 RPM

Brainstorming and Concept Design

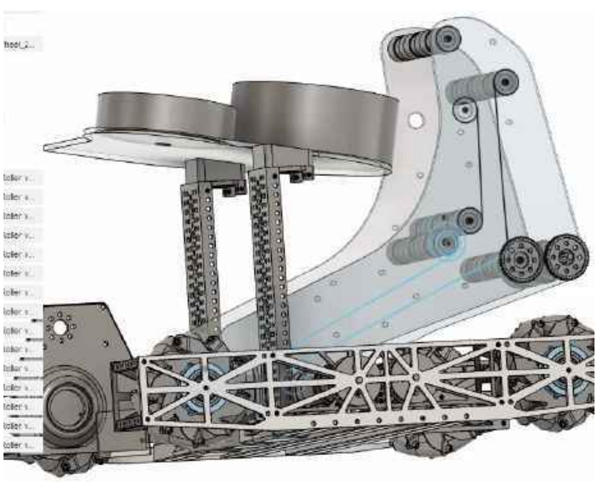
Designing concepts is the third and most time consuming step in our design process. This is the stage where we define well researched conceptual solutions to the problems we have defined.

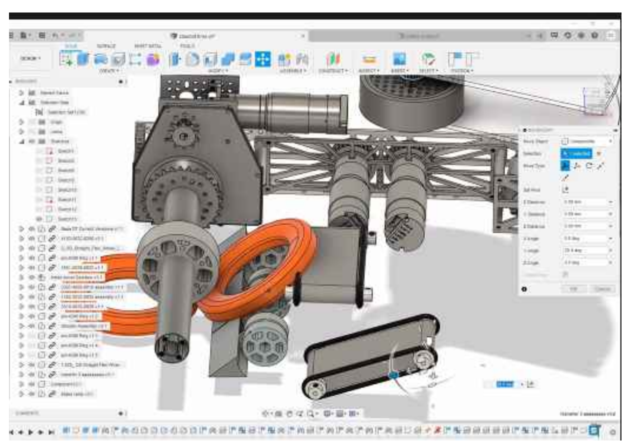
From the start of the season, we had a few central concepts that we wanted to stick with throughout the season:

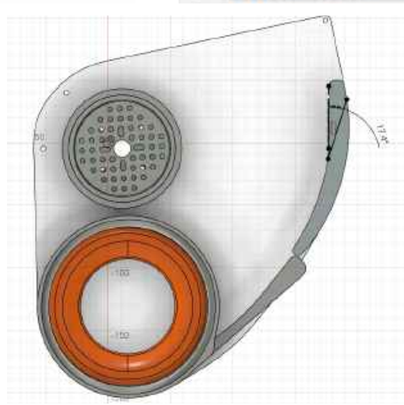
- We wanted to give the robot a design which allowed it to both pick up and launch disks from the front.
- We wanted to design a launcher which would be able to aim up and down, as well as rotate so we wouldn't have to steer the robot to aim.

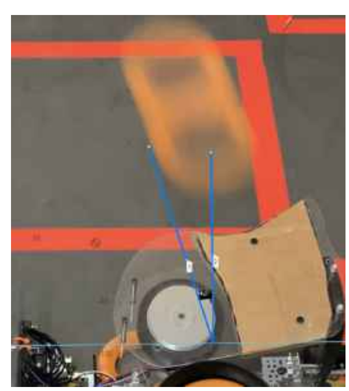
We made early prototypes during our post-kickoff brainstorming session:

Then we refined these ideas with research and CAD:









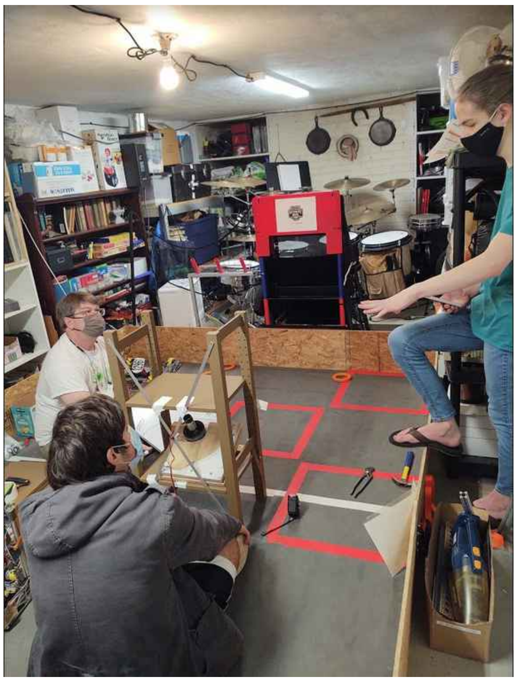
Testing

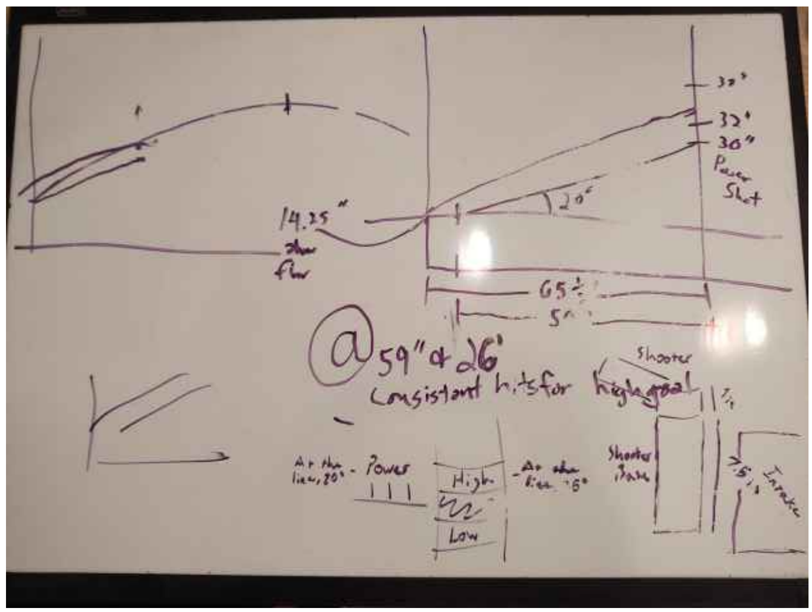
Testing our concepts in the form of prototypes is the fourth step in our design process. Through prototyping and testing, we can reach a more refined final product.

For our team specifically, we have created an identical copy of our robot's drivetrain for us to prototype new systems on without affecting the main robot. This has proven incredibly useful throughout the season and has probably saved us countless hours of time.

There were a couple of things that we tested and prototyped throughout this season:

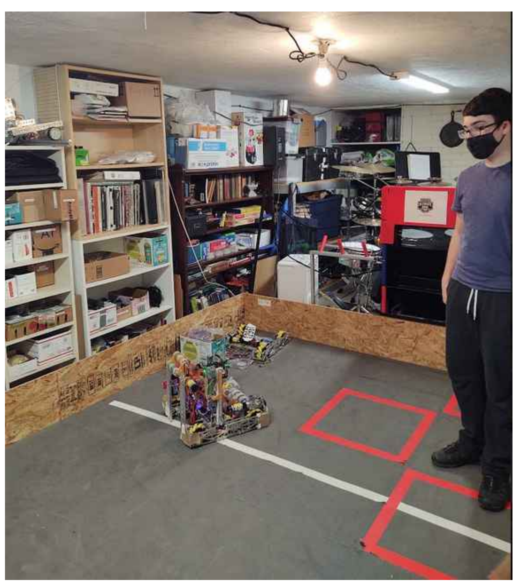
An early prototype of the shooter allowed us to test velocity, angle and 22 degrees:





Testing the shooter prototype

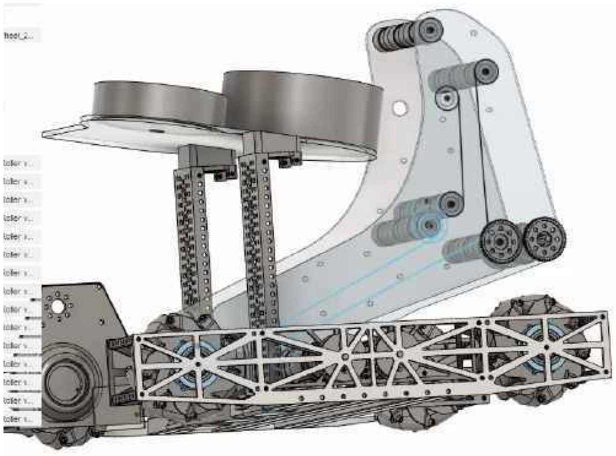
Some calculations to determine the angle of the shooter



The Roadrunner code required a lot of testing and "tuning" to work reliably

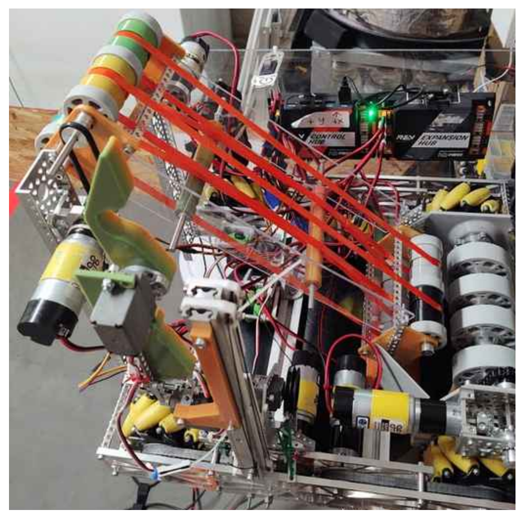
System Level Design

This year we spent a lot of time on system level design. We had the basic shooter and intake designs fleshed out early in the season. However, it took some time to create a transfer mechanism that would smoothly take rings from the field to the intake to the conveyor belts to the hopper and finally to the shooter.

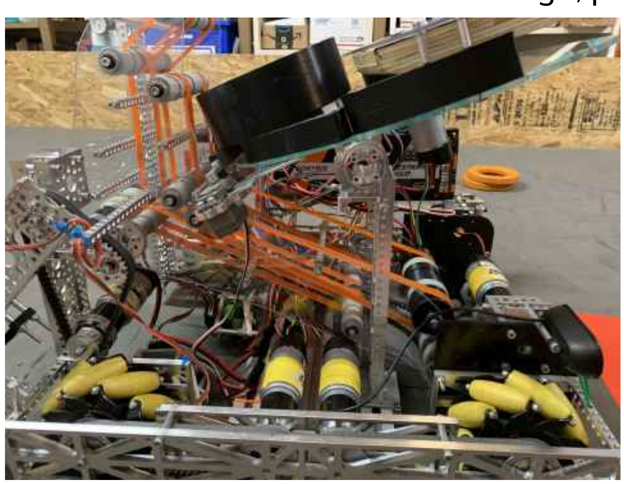


Transfer Iterations

1st design, rubberbands came off track



2nd design, polybelts still slipped



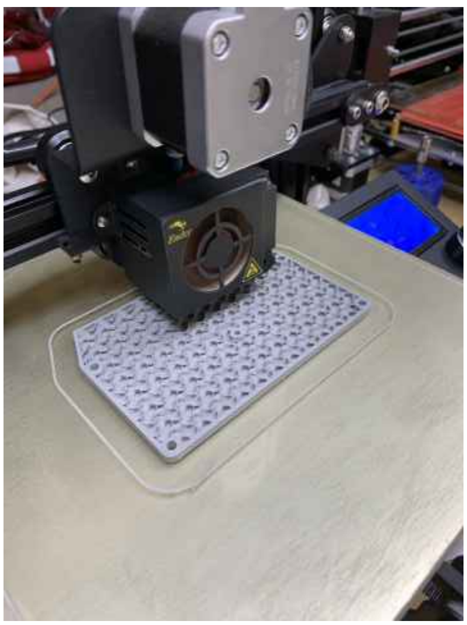
Final Design is reliable and fast with grooves for the belts grooves hold the belts in place!

Production

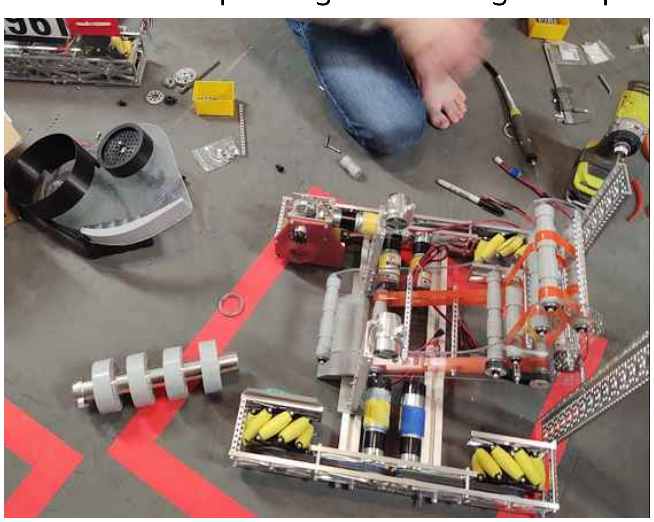
Production is the fifth step in our design process. Because of our extensive CAD and testing process, production takes up much less time than in prior seasons. This step basically involves fabricating and assembling the refined versions of our prototypes from the previous step.

We have 4 main methods of production that we utilize:

- Free building
  - GoBilda parts are very versatile
- 3D printing
  - Endless possibilities and durable results
- Band saw + printed template
  - Polycarbonate/acrylic plexiglass
- CNC/water jetting
  - Aluminum/polycarbonate




3D printing and cutting a template on the bandsaw




Detailed CAD designs allow assembly to happen quickly once the parts are created/purchased.

Improvement

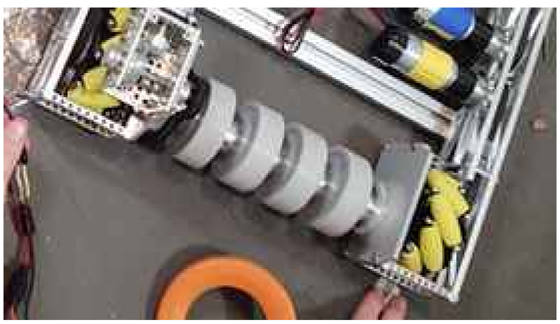
After production is complete, the next stage in the process is improvement or tweaking. This mostly involves small changes to individual systems in order to help them work more smoothly. Sometimes, this improvement stage means going back to the drawing board entirely because a better mechanical method is discovered.

Nearly every system on our robot has undergone minor improvements. These include:

- More accurate intake funneling
- Geometrical tweaks to wobble gripper
- Completely reworking transfer mechanism to allow for shooter

Intake

Prototype

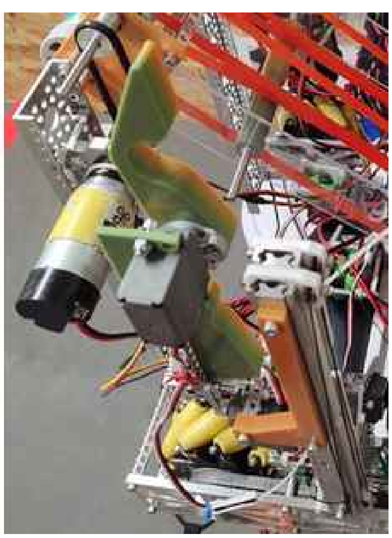


Design 1

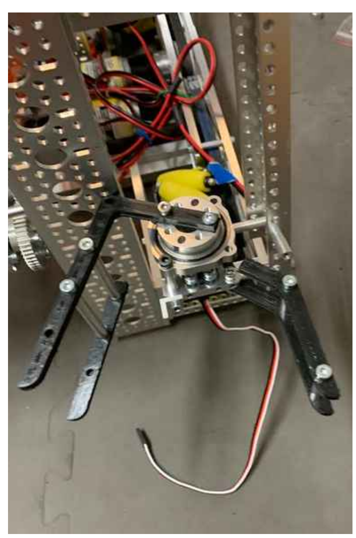
Final Design

Wobble Goal Grabber

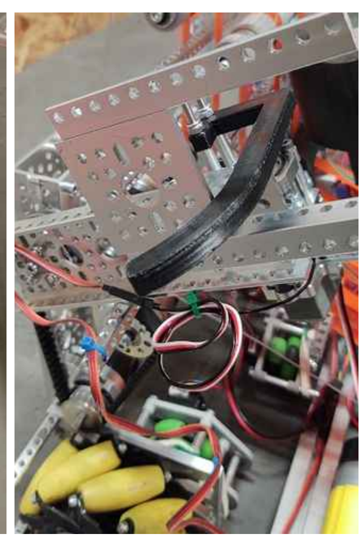
1st design



2nd design



3rd design



4th design

Checkmate Programming Summary

Key Features

- Pathfinding: Roadrunner and Odometry
  - Roadrunner is a library used for motor feedback based control and odometry for accurate field positioning.
  - Pathfinding: We used Roadrunner to make pathfinding in autonomous and for the Driver-Assist Features.
- Relative Driving
  - This deprecated class handles all of the hardware initialization instead of the opmode. This means if we add a new servo or motor, we don't have to go change every opmode.
  - We developed this over the summer before we implemented Roadrunner.
    - We scrapped this because it was too complicated, and not very helpful.
    - It will probably work better when we have time to layer in Vuforia targets
- Environmental Sensing
  - Uses odometry pods to measure distance traveled, which is much more accurate than time based instructions.
  - Uses OpenCV to program a webcam to detect the number of rings.
  - Our ring detection is completely dependent on the camera not being jostled though and needs a solid mechanical design to keep this consistent.
- Driver-Assist Features
  - We made “seek to” positions to aim the shooter for the high goal and the powershots during tele-op and endgame to make the job of the drivers easier.
    - These are bound to buttons on the D-pad of the movement controller
    - These work great during tele-op for the high goal and the powershots and are responsible for our high tele-op and endgame scores.

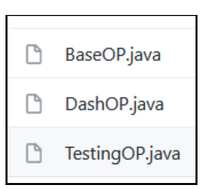
Practical Impacts

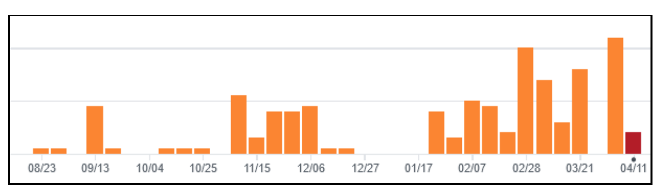
Our goal this year was to fully implement the field positioning system the team started last year. Our veteran programmers graduated, so this was no small feat for our new coders.

It was a ton of hard work, hard lessons learned regarding version control, semi-successful attempts to code remotely while various members were in COVID quarantine, but **our use of pathfinding has led to consistent high scoring in autonomous and end game**.

Strategies for Success:

- Codacy
- Simplify code into classes
- Github for Xtreme™ version control (217 commits!)





Testing Op Modes | Commits throughout the season

Custom Classes and Notable Code

```
    // These numbers tell the robot where it is when the
    //  endgame powershots get triggered.
    public static double setPointX = 2.5;
    public static double setPointY = -61.25;
    public static double setPointHeading = -2.6;
```

Endgame Powershot starting position

```
case AUTO:
    robot.setMode(DcMotor.RunMode.RUN_USING_ENCODER);
    // Replace false here with a check to cancel the trajectory
    //noinspection ConstantConditions
    if (false) robot.cancelTrajectory();
    if (!robot.isBusy()) controlMode = ControlMode.TELE;
    break;
default:
    // If we end up here, something went horribly wrong.
    // Generally, the best plan of action is to ignore
    //  it and move on.
    controlMode = ControlMode.TELE;
    // Mission accomplished.
    break;
```

Failsafe Measures In Teleop Mode Control

```
// BIND:
//  gamepad1.left_stick_x, gamepad1.left_stick_y
//  gamepad1.right_stick_x, gamepad1.right_stick_y
//  gamepad1.right_trigger
protected void runDrivetrain (DrunkenHippoDrive robot, double rotationalOffset, boolean relative){
    // Read pose
    Pose2d poseEstimate = robot.getPoseEstimate();
    double offset;
    if (relative) offset = -rotationalOffset - poseEstimate.getHeading();
    else offset = 0;

    double xin = gamepad1.left_stick_x * Range.scale((gamepad1.right_trigger), -1, 1, 0, 1);
    double yin = gamepad1.left_stick_y * Range.scale((gamepad1.right_trigger), -1, 1, 0, 1);

    // Create a vector from the gamepad x/y inputs
    // Then, rotate that vector by the inverse of that heading
    Vector2d input = new Vector2d(
            -yin,
            -xin
    ).rotated(offset);

    // Pass in the rotated input + right stick value for rotation
    // Rotation is not part of the rotated input thus must be passed in separately
    robot.setWeightedDrivePower(
            new Pose2d(
                    input.getX(),
                    input.getY(),
                    -gamepad1.right_stick_x * Range.scale((gamepad1.right_trigger), -1, 1, 0, 1)
```

Controller Button Mappings

```
// BIND: gamepad1.dpad_right
private void runSeekPowerShots (DrunkenHippoDrive robot){
    if (gamepad1.dpad_right) {
        controlMode = ControlMode.AUTO;
        while (setPointHeading > 360){
            setPointHeading -= 360;
        } while (setPointHeading < 0){
            setPointHeading += 360;
        }
        //Changes robot position estimate to the side of the field, so roadrunner is more consistent
        robot.setPoseEstimate(new Pose2d(setPointX, setPointY, Math.toRadians(setPointHeading)));
        //Revs flywheel in advance.
        robot.revFlywheel(-EndgamePowerConstants.veloRight);
        //One trajectory defined for each of the high goals.
        rightShot = robot.trajectoryBuilder(new Pose2d(setPointX, setPointY, Math.toRadians(setPointHeading)))
                //Move to shooting locations
                .lineToSplineHeading(LauncherUtils.getPowerPose(LauncherUtils.Position.RIGHT))
                .addDisplacementMarker(() -> {
                    //Wait for the flywheel to get to full speed. This line is unique to this block.
                    robot.waitForFlywheel(LauncherUtils.flywheelThreshold);
                    //Virtually presses trigger
                    robot.pressTrigger(true);
                    //Gives the trigger time to finish shooting.
                    sleep(LauncherUtils.triggerActuationTime);
                    //Retracts trigger
                    robot.pressTrigger(false);
                    robot.revFlywheel(-EndgamePowerConstants.veloCenter);
                    robot.followTrajectoryAsync(midShot);
                })
```

Seek-To Positions

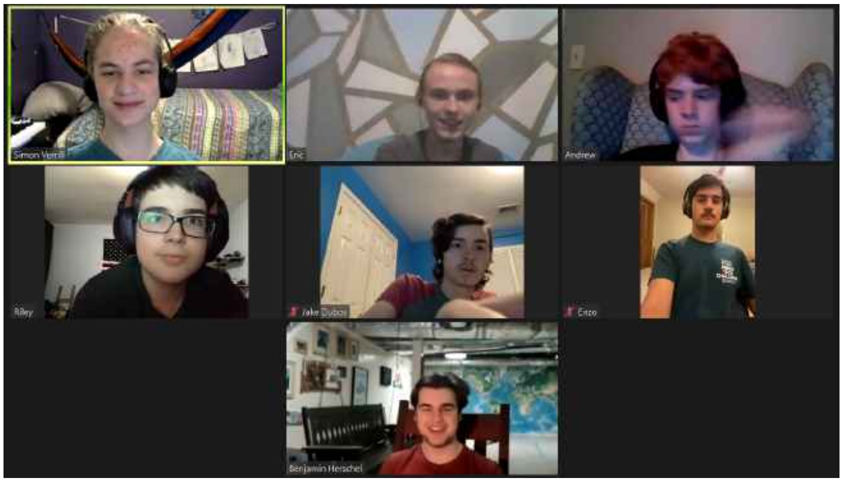
```
while (opModeIsActive() && !isStopRequested()) {
    switch (currentMode){
        case FIRST:
            runTrajectory(missRings);
            currentMode = RunMode.RUNNING;
            break;
        case SECOND:
            runTrajectory(grabWobble);
            currentMode = RunMode.RUNNING;
            break;
        default:
            break;

    }
```

Converging Auto Paths

```
Trajectory toFollow;
switch (ringPosSaved){
    case NONE:
        toFollow = dropA2;
        break;
    case ONE:
        toFollow = dropB2;
        break;
    case FOUR:
        toFollow = dropC2;
        break;
    default:
        toFollow = null;
        break;
    }
runTrajectory(toFollow);
```

Vision Based Path Following

Checkmate Team Summary



Checkmate is a 4th year community team based out of Kennesaw, Georgia. Our team is committed to bringing innovative, ambitious and interesting ideas to life. We try to use the most effective methods of making our visions into reality, including AGILE project management, CAD Software, 3D Printing and collaborating directly with local companies. We have many contacts in the engineering and CAD fields who also assist our team.

Our mission is to inspire young people to pursue STEM careers through robotics competitions and by partnering with our community to host engaging educational events. Our goal is to excel at the robotics competitions, but also to show Gracious Professionalism and build a cohesive team where each person learns and can contribute.

Our team is student lead. We not only program, build and conceptualize our robots, but we also fundraise for the team and manage relationships with professionals with minimal mentor input. Today the team consists of seven students and two regular mentors, a PHD student from Georgia Tech and an engineer at Burrell Aerospace. The pandemic has been a challenge for us because limited in-person or no in-person contact has meant that we needed to get creative and re-think some of our approaches.

Skills we're working as a team: We're both teaching basic skills to new members and also developing more sophisticated skills for experienced members**:**

- **CAD**
- **Programming**
- **Design Methodology**

	Strategy to improve #1:	Work more on cross training members and will work with team members on coding and CAD
	Strategy to improve #2:	Reach out to STEM professionals who can help us learn about careers as well as help with the robot

Checkmate Community Outreach

Summary of Experiences

This year, we wanted to directly reach people, while also keeping our local community safe. This unfortunately meant a temporary suspension of our ongoing STEAM events with Southern Train Museum for the 2020-21 season. Instead we really tried to focus on activities that would get the most good done.

Early on in the pandemic, we used our army of 3D Printers to create face shields for the first responders in the metro-Atlanta community. We continued this program through the summer of 2020, ultimately printing **1200** face shields between our team's 4 printers.

After the Ultimate Goal Kickoff, we initiated other forms of virtual outreach:

- Georgia Connections - we meet with the schools statewide STEM club to help them formulate a plan to start several FTC teams
- Atlanta Homeschool and Virtual School Community - Organized virtual STEM fall break activities for students in Cobb, Marietta, Douglas and Cherokee counties.
- Thingiverse - Released custom designs to the greater FTC community, which have been downloaded over **1300** times this season

Lessons Learned

With the global pandemic, community outreach became significantly trickier. First we focused using our STEM skills to do the most good. Then we attempted other virtual outreach. Our fall break and school outreach met with mixed success - by fall, we think many kids were already tired of virtual activities.

Outcomes

Faceshield Donations	
Kennestone Hospital	165
Atlanta Medical Center ER	35
VA Hospital ER	50
Grady ER	25
Devereaux Kennesaw	40
Devereaux National	225
All Hands In	115
Homeschool Consortium	28
Elevator inspection company	50
Devereaux National July	200
Albany Putney ER	60
Atlanta Legal Aid	160
Grady Pulmonology	40
Total Donations	**1193**

Checkmate Outreach to FIRST

Summary of Experiences

Like other forms of outreach, we have relied heavily on virtual formats, where in prior years we have focus on in-person meetings:

- **Thingiverse** - To assist the FIRST community, we have released our CAD files for our 3D-printed designs to Thingiverse to be used by anyone who wants it as open source files, ready to be printed. Released custom designs to the greater FTC community, which have been downloaded over **1300** times this season.
  - For the small effort it takes to post these designs (once we are satisfied with them), it has had a remarkable impact.
- **Discord**: Not being able to communicate with teams in-person during Meets or otherwise, this year we relied heavily on the FTC Discord community
  - In addition to bouncing ideas off of the minds of fellow teams through the Discord server, we also assisted countless teams from across the world in their own robotics endeavours
- **GitHub**: We keep our code open and available for other teams to use...however, this form of outreach hasn't been nearly as successful as Thingiverse!

Checkmate STEM Mentors

Summary of Experiences

STEM connections:

- **Andrew Schulz -** Georgia Tech, PHD Student | Design and strategy input.
- **Jeremy Crowley -** Burrell Aerospace, Engineer | CAD assistance and fabrication of parts.
- **Dr Ian Wong -** pulmologist and biomedical engineer working on open source ventilator parts

Lessons Learned

What we learned by working with mentors this year is that we can develop skills that compare to the skills of adults working in STEM fields. This tells us that we are building extremely valuable skills through participating in FTC.

Outcomes

It was great to have professionals give us feedback on our designs. The suggestions made by our mentors helped us improve our designs and our skill sets.

Checkmate Business Plan

Team Organization and Management

	Team Roles	
Jake: Business VP	Simon: Engineering VP	Riley: Software VP
Jake: Notebook & HR	Enzo: Autonomous Lead	Enzo: Tele-Op
Eric: Fundraising	Ben: Tele-op and Fabrication	Andrew: Vision Detection
Jake: Community outreach - kids	Riley: Tele-op	Riley: Autonomous Lead
Jake and Ben: Strategy	Simon: Tele-op Lead and CAD	
Eric: STEM connnections	Jake: Autonomous	

Checkmate's goal is to learn to use Agile Project Management, which is designed to involve the whole team in short "sprints" that will allow the team to create components that will score early in the season. Our plan uses a version of Agile, modified for FTC. **Trello** is the team's project management tool. Due to COVID-19, our team met mostly virtually and in small groups. As a result our design process was more linear and time consuming.

SWOT Analysis and Skill Development Plan

SWOT Analysis		
	Strengths	Weaknesses
Internal	We have 5 experienced members who have been cross-trained and have multiple skills	Small meeting space and the inability to have everyone meet in-person at the same time (COVID)
	We have 3 great new members, including great coders	Health limitations for some team members
	Our shop has many tools and mentors and members who know how to use them	Members are great friends, and that can potentially inhibit productivity
	Well stocked with basic building materials and 3D printers	Limited in-person interaction for majority of team.
	A lot of connections to the community and outreach experience and ideas for doing virtually	Our coders are new to FTC
	Most of our team members won't age out.	
	We have a solid foundation for our notebook format	
	Opportunities	Threats
External	We have the ability to support the community with our	Other teams with higher skills
	We have a lot of financial support for the team through individual donations.	School and other obligations have increased for many members
	Mentoring opportunity through Georgia Connections	We're joining a new league and don't know the other teams as
	Multiple connections to STEM professionals	We will have a small space to run our meets
	We have access to CNC milling through a team sponsor	Our regular community outreach has stopped due to COVID