THE HIGHLANDERS

#4499

2023 TECHNICAL BINDER

CONTENTS

ANALYSIS

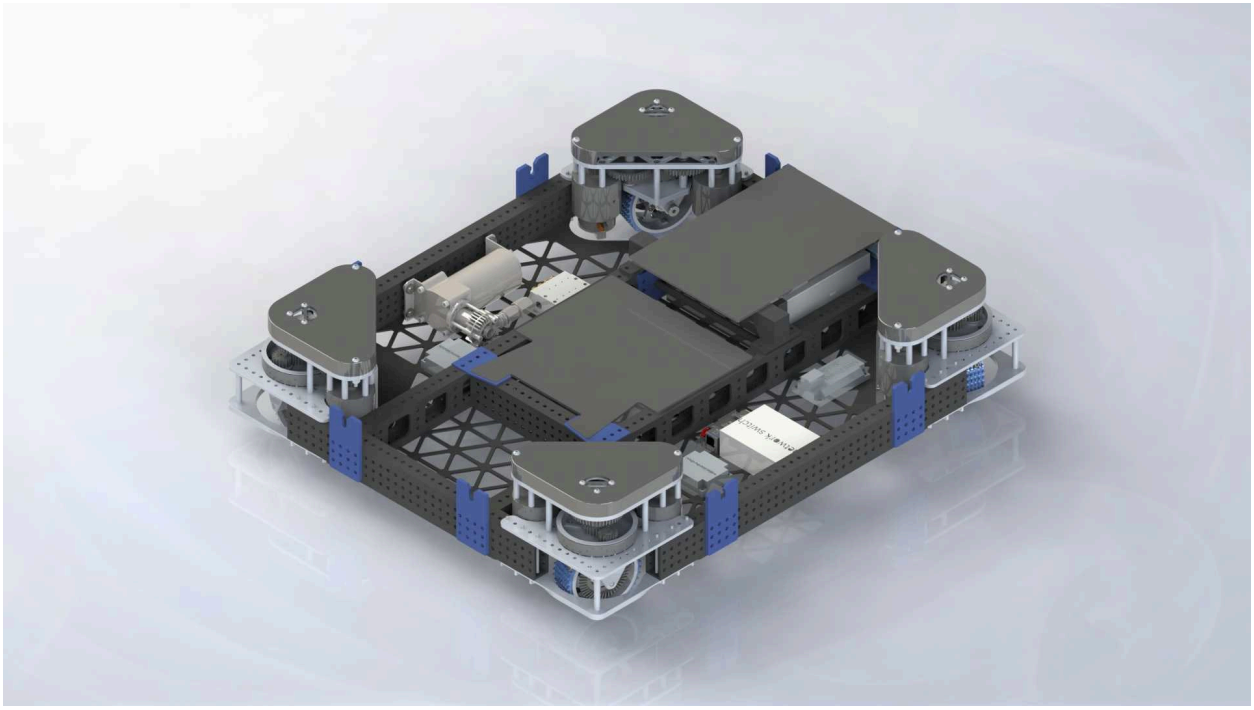
This year, our game focus was control. To achieve our goals and be adaptable to any alliance, we needed to create links on all three levels. It was crucial for us to be able to meet teams at any level to maximize points and utilize our alliance abilities to the fullest. Having a robust robot was necessary because the charging station takes up a significant portion of the field, and being a robot enabled us to overcome this obstacle. Being able to pick up game elements at any time was also beneficial in improving our cycle time and gaining control of the game elements with other robots around. Therefore being able to pick up from as many sides as possible is vital.

Priorities

1. Omnidirectional fast drive train
2. Pick all game elements in al orientations
3. reach all 3 levels
4. 2 Piece auto both sides, red and blue, leaves community
5. Balance on charge station
6. Climb with two alliance partners
7. Single button placement

ROBOT DESIGN

DRIVE TRAIN



This years drive base is optimized for fast cycles and the ability to fit 3 robots on the charging station during endgame.

Being similar to last years chassis, the team was able to devote more time on other subsystems

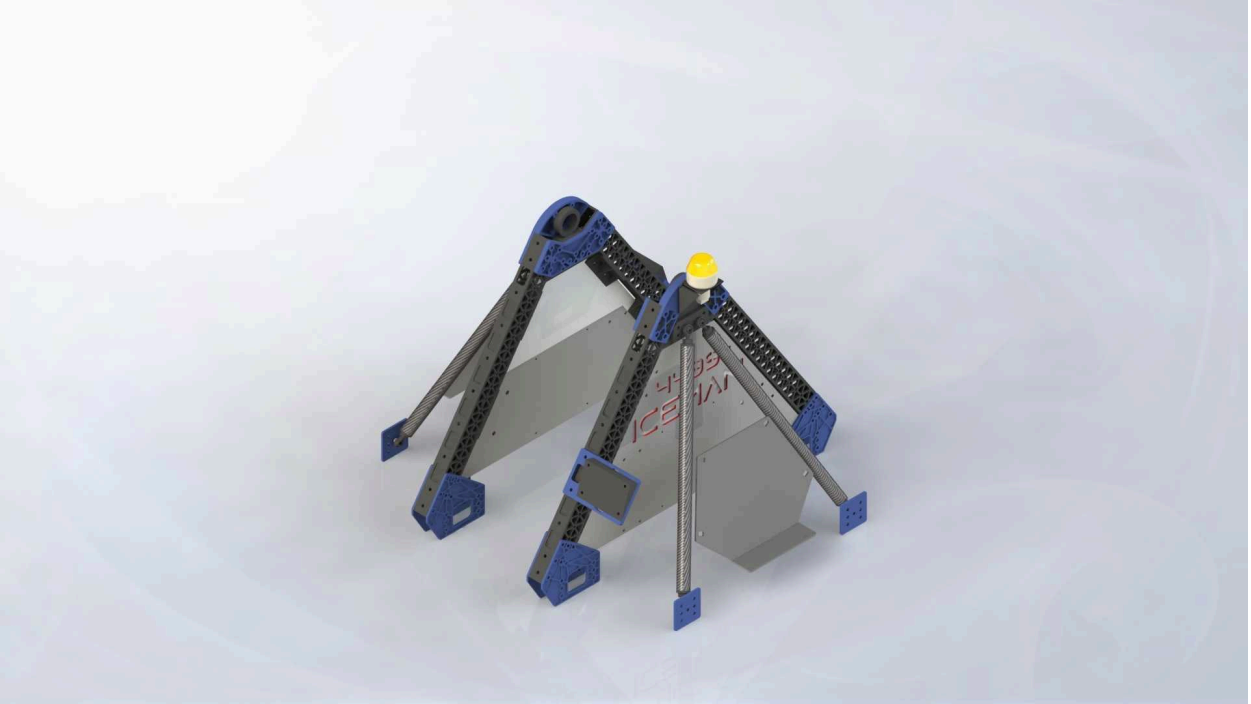
Chassis

- Built with 2"x1"x1/8" with standardized hole pattern for easy subsystem mounting
- 25"x29" frame
- 2 crossbeams for better structural integrity

Swerve

- Driven with 4 MK4i-L2 modules for maneuverability around the field
- Top speed of 16.3ft/s

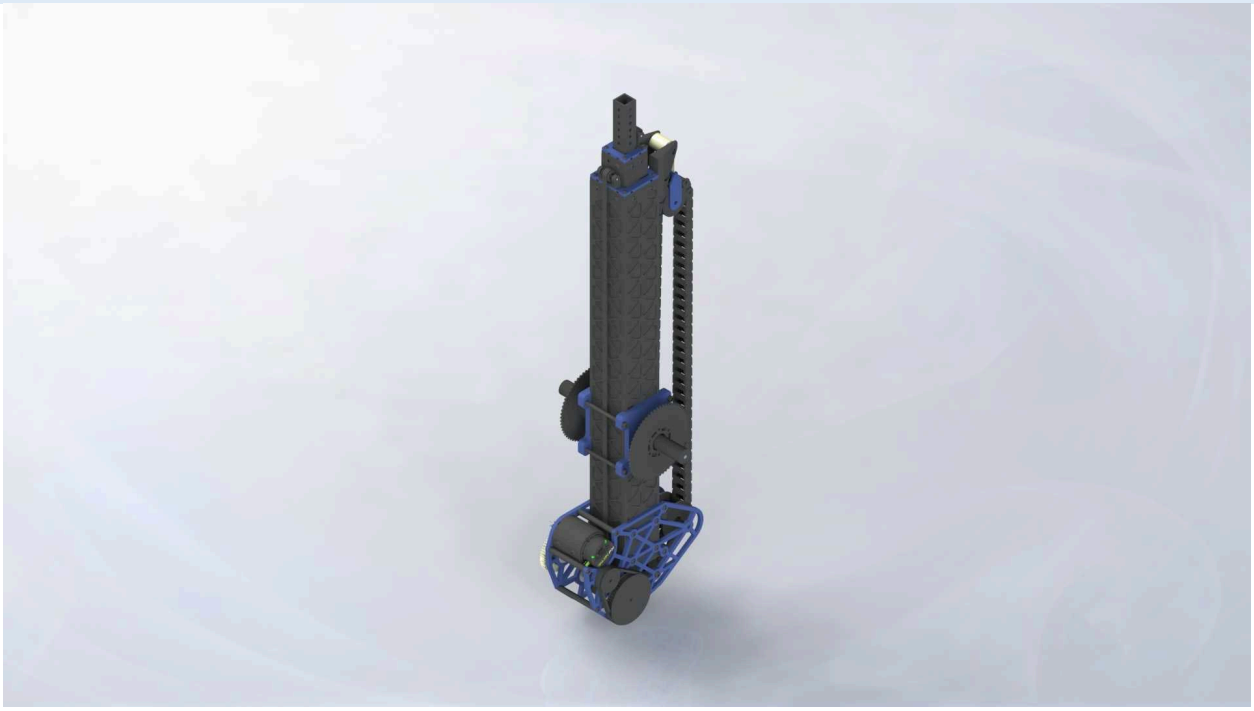
A-FRAME



This is the support structure for the arm, which has also been designed for the ability to hot-swap the arm. With multiple support structures, it is robust enough to withstand impacts from other robots and items on the field.

- Centered pivot point for symmetrical arm positions
- Mounted on chassis crossbeams with tube blocks
- Easily removable arm pivot for quick arm swaps
- Lights to indicate preferred game object to human player
- Carbon fiber tie rods for increased rigidity

ARM



The arm contains a weighted gearbox and weight savings to maintain balance across the robot. It also has a rigid structure capable of reaching all levels and withstanding impacts from robots on the field. We also made it capable of picking up both sides of the robot.

- 2 stage telescoping 3"x3", 2"x2", and 1"x1" tubes
- Cascading belt rigging ran inside the tubes
- Driven by 1 Falcon 500 with a 10:1 reduction for an extension speed of 0.5 seconds
- Ability to pick up and place game pieces on both sides

Rotation

- Driven by 2 Falcon 500 with a total reduction of 186.67:1
- Completes full range of motion under 1 second
- Uses worm gear to prevent back drive

WRIST

We designed this part to be compact in order to keep the weight at the end of the arm as light as possible. By minimizing the weight, we can improve the arm's maneuverability and precision. Additionally, reducing the weight at the end of the arm can help us avoid tipping over during the game.

- Driven by 1 Neo 550 with a 280:1 reduction
- Rotates intake on dead axle
- Uses magnetic encoder for accurate positioning
- Uses 3D printed herring-bone gears to reduce backlash from wrist to encoder

INTAKE

The team chose a durable and versatile design, equipped with two 12-inch rollers on the intake that can intake and place from both sides, allowing for quick and efficient collection and precise depositing of game elements from any orientation on the field.

- Driven by 1 Falcon 500 with a 2:1 reduction
- Motor mounted on wrist to maintain motor position
- Roller surface speed of 13.09 ft/s
- Dead axle carbon fiber rollers with silicon rubber tubing to increase grip
- Amsteel rope to hold intaked cube

PROGRAMMING

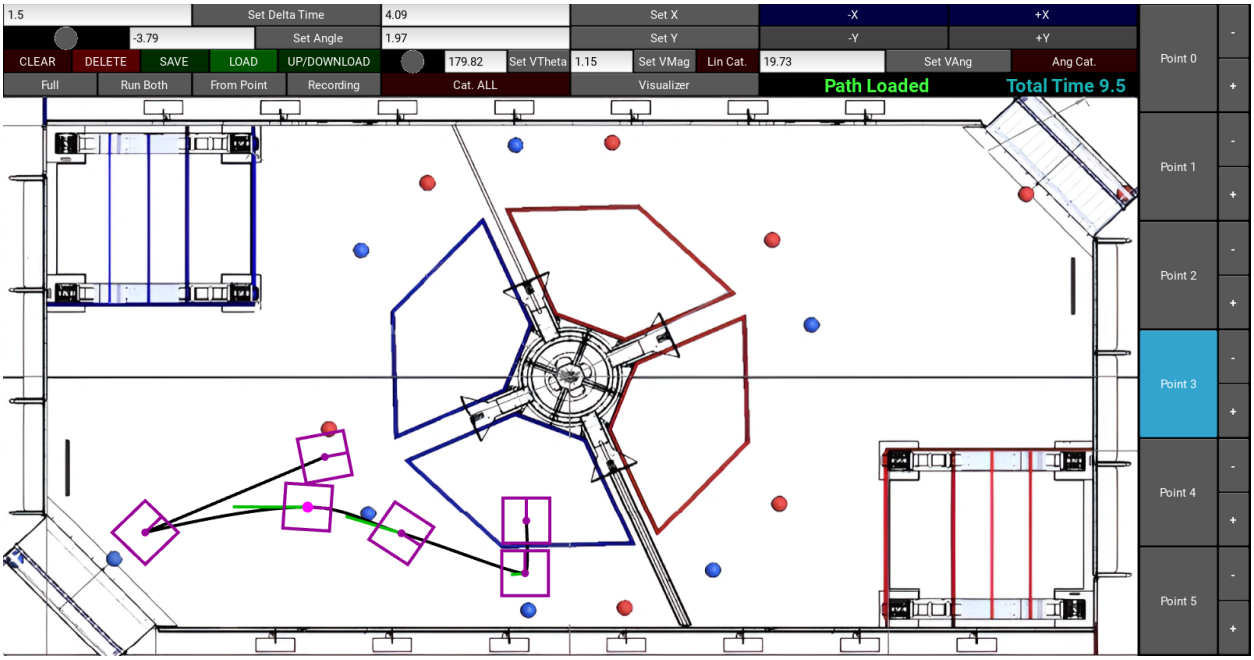
AUTONOMOUS

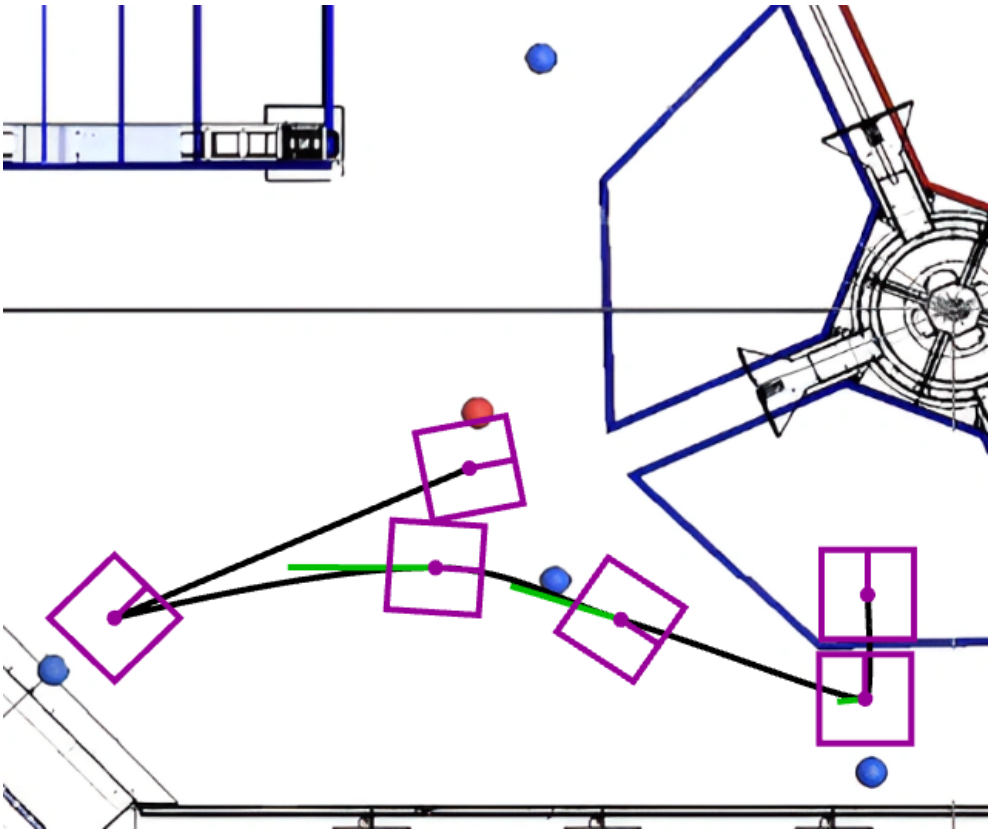
Our autonomous plan is as follows: during the autonomous period, we aim to establish as much of a link as possible. We also aim to get onto the charging station for RP (ranking points), and we will try to balance if possible.

Autos:

- 2 pieces placed high on clear and cable chain side, then dock
- 1 piece placed high in center and dock

PATHING TOOL





SET UP

Install Python 3.10.2

Clone the “2023-Pathtool” repository on HighlandersFRC GitHub

Navigate inside of the 2023-Pathtool folder

In a command line, navigate to the desired parent folder and run “pip install -r requirements.txt”

To launch the application run “python main.py”

DESIGN REQUIREMENTS AND SOLUTIONS

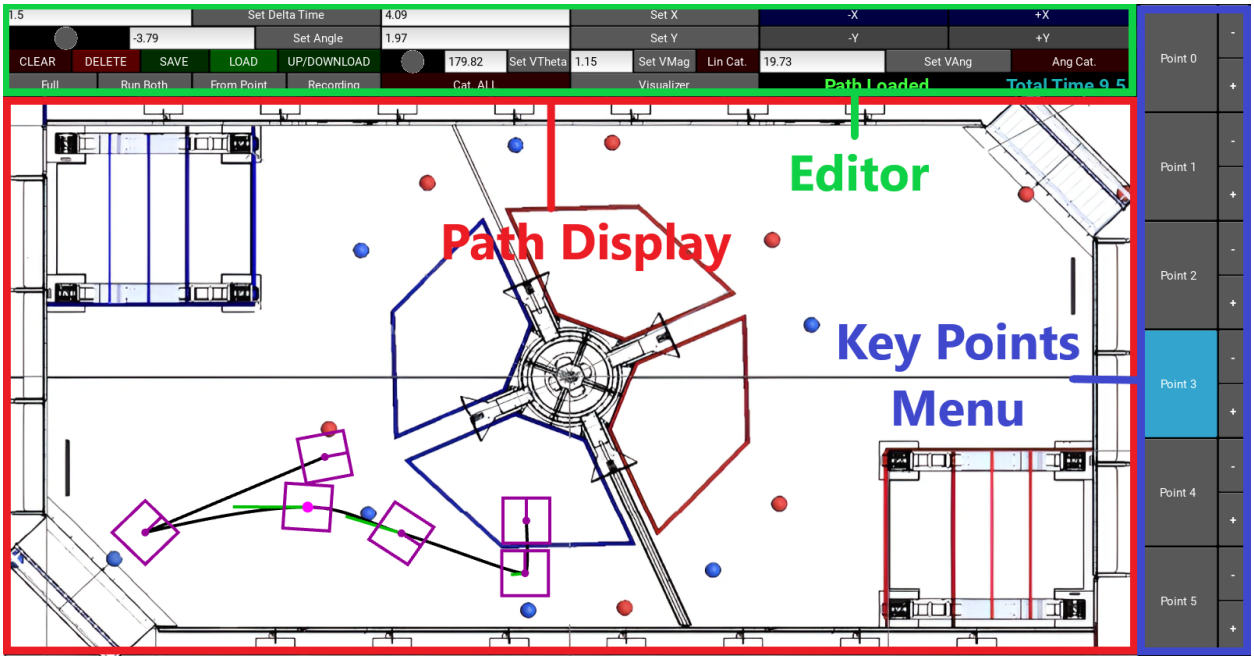
Before we started work on creating the new pathing tool, we set a couple of requirements that the tool must meet:

- Be user friendly (intuitive controls/displays)
- Generate path equations with control over linear and angular position, velocity, and acceleration at each key point in the path
- Upload and download these paths to the RoboRio onboard the robot
- Display useful feedback information to guide the creation of paths
- Be flexible as an application in order to make adjustments/expansions of the path tool easier

To meet these requirements we implemented the following solutions:

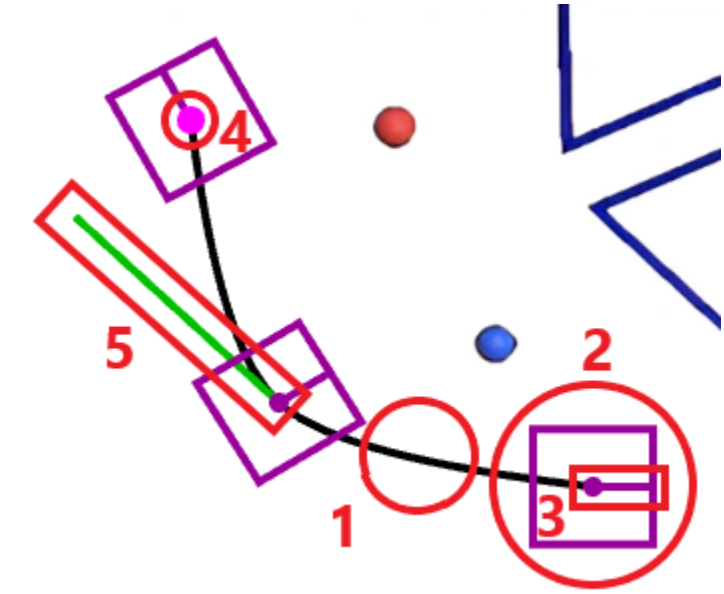
- Get feedback from using the path tool
- Use quintic Catmull-Rom hermite splines to generate path equations
- Use paramiko, a python ssh library, to handle file transfer
- Implement feedback tools such as path animations, graphs and animations of recorded odometry data, coloring the path to highlight when the path exceeds physical capabilities (linear acceleration, linear velocity) of the robot, and others
- Use kivy, an open source Python GUI framework, to create a modular interface for the path tool

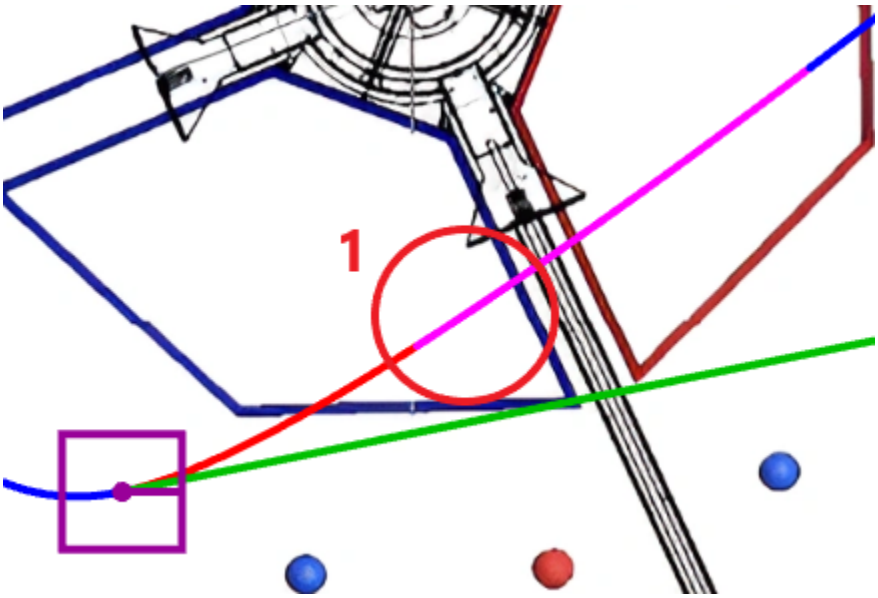
LABELED DIAGRAMS AND DESCRIPTIONS



Section	Function
Path Display	Displays the current path, allow creation of new key points, display animations and recorded odometry.
Editor	Edits parameters of individual key points, displays information (action status, total time) and provides access to popup menus for file handling and the visualizer.
Key Points Menu	Allows for easy selection of key points, and allows key points to be switched in index.

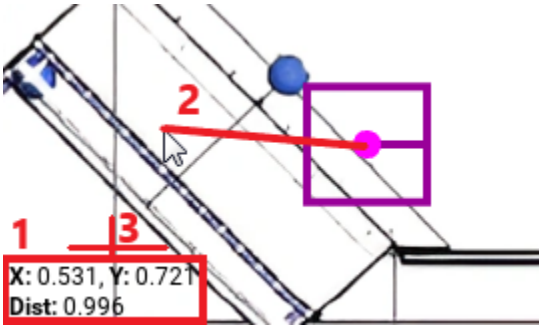
Path Display





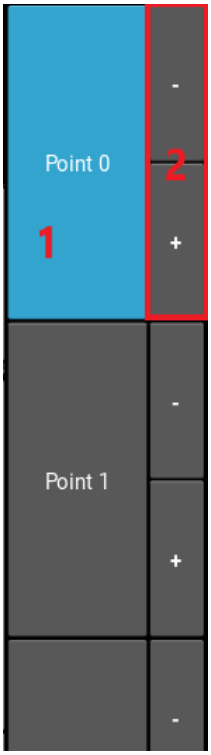
1 - Path Line	Curve that shows what path the robot should follow. Sections colored red indicate a linear velocity higher than the set physical limit. Sections colored blue indicate linear acceleration higher than the set physical limit. Sections colored magenta indicate both of the above. Sections colored black indicate neither.
2 - Robot Indicator	Shows where the edges of the robot should be at that key point.
3 - Robot Angle Indicator	Shows at what angle the robot should be facing at that key point.
4 - Selected Point Indicator	Indicates that that key point is currently selected.
5 - Linear Velocity Indicator	Indicates the magnitude and direction of the robot's linear velocity at that key point.

Cursor Position and Measuring



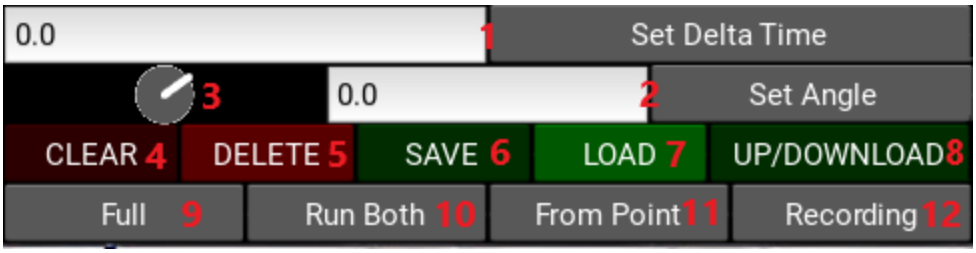
1 - Position and Measurement Text	Displays the position of the cursor (x, y) in meters as well as the distance from the cursor to the selected key point, in meters.
2 - Distance being measured	This is not shown in the actual tool, but the line drawn in the diagram indicates what distance is being measured
3 - Origin point	This is the point at (0, 0). The tool uses a right-handed coordinate system with x as the horizontal axis and y as the vertical axis, which means that x increases as points move from left to right, and y increases as points move upward.

Key Points Menu



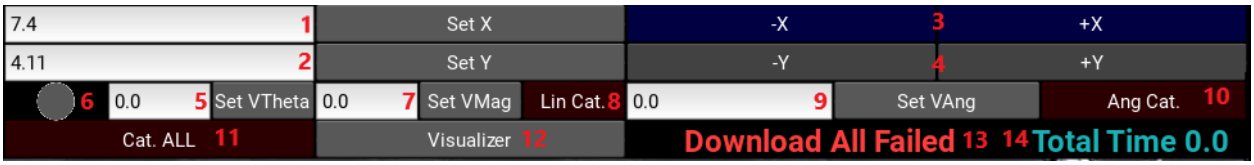
1 - Select Point Button	Selects the key point of the same index, useful for selecting key points that are very close to each other.
2 - Reindex Point Button	Switches the selected key point with the point before or after it, controlled with the buttons labeled “+” and “-”. This is useful for inserting key points into the middle of a path.

Editor



1 - Delta Time Input	Type in and press the button to set delta time. Delta time is the time in seconds since the previous key point.	**7 - Load Button**	Opens the file manager popup to load a saved path.
2 - Angle Input	Type in and press the button to set the angle. This is the angle in degrees of the selected key point.	**8 - Upload / Download Button**	Opens the file manager popup to upload the current path, upload all saved paths including the current one, or download all paths on the RoboRIO.
3 - Angle Dial	Press and drag the dial manually to set the angle of the selected point.	**9 - Full Animation Button**	Runs the entire animation of the current path.
4 - Clear Button	Deletes all key points.	**10 - Parallel Animations Button**	Runs the recording animation in parallel with the animation of the current path. A recording must be displayed on the field in order to do this.

5 - Delete Button	Deleted the selected key point.	**11 - Animation From Point Button**	Runs the animation of the current path starting at the selected key point. A key point must be selected in order to do this.
6 - Save Button	Opens the file manager popup to save the current path.	**12 - Recording Animation Button**	Runs the animation of the recording currently displayed on the field. A recording must be displayed on the field in order to do this.



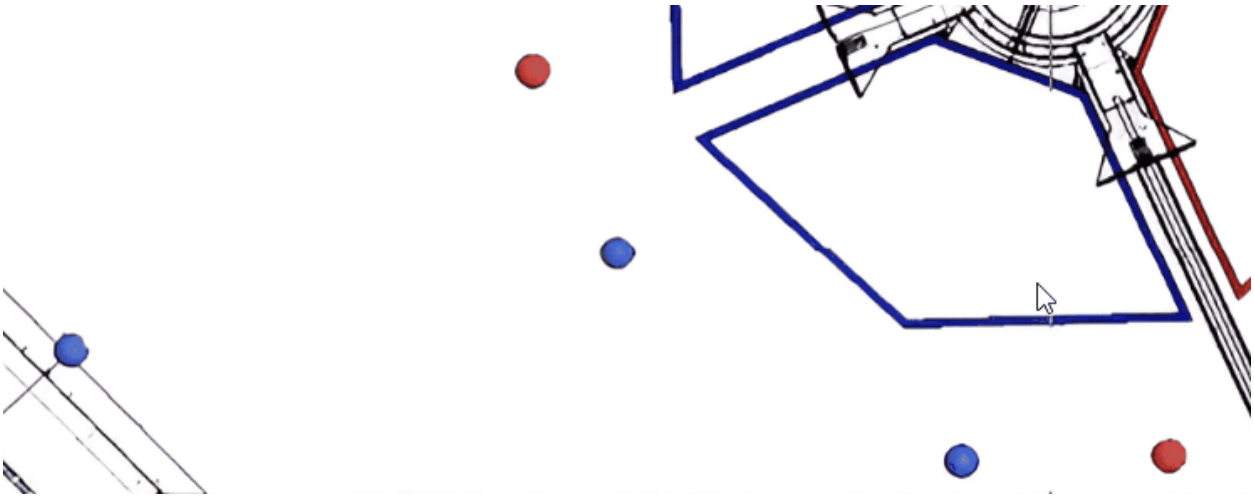
1 - X Input	Type in and press the button to set the x coordinate of the selected key point in meters.	**8 - Linear Velocity Catmull-Rom Button**	Convert the linear velocity equations of the path into Catmull-Rom splines.

2 - X Adjust	Pressing the **-X** button subtracts 0.025 meters from the selected key point's x coordinate, and pressing **+X** adds 0.025 meters.	**9 - Angular Velocity Input**	Type in and press the button to set the angular velocity of the selected key point.
3 - Y Input	Type in and press the button to set the y coordinate of the selected key point in meters.	**10 - Angular Velocity Catmull-Rom Button**	Convert the angular velocity equations of the path into Catmull-Rom splines.
4 - Y Adjust	Pressing the **-Y** button subtracts 0.025 meters from the selected key point's y coordinate, and pressing **+Y** adds 0.025 meters.	**11 - Catmull-Rom All Button**	Convert all linear and angular equations of the path into Catmull-Rom splines.
5 - Velocity Theta Input	Type in and press the button to set the velocity theta of the selected key point. The velocity theta is the angle at which the robot will be moving at the selected key point.	**12 - Visualizer Button**	Opens the visualizer popup menu.

6 - Velocity Theta Dial	Press and drag to manually set the velocity theta.	**13 - Status Label**	Displays the status of the last major path operation. This includes information about uploading, downloading, saving, loading, displaying recording on field, updating recordings, clearing recordings, and clearing RoboRIO recordings.
7 - Velocity Magnitude Input	Type in and press the button to set the velocity magnitude of the selected key point. The velocity magnitude is the linear velocity that the robot will have at the selected point.	**14 - Total Path Time Label**	Displays the total time duration of the path in seconds.

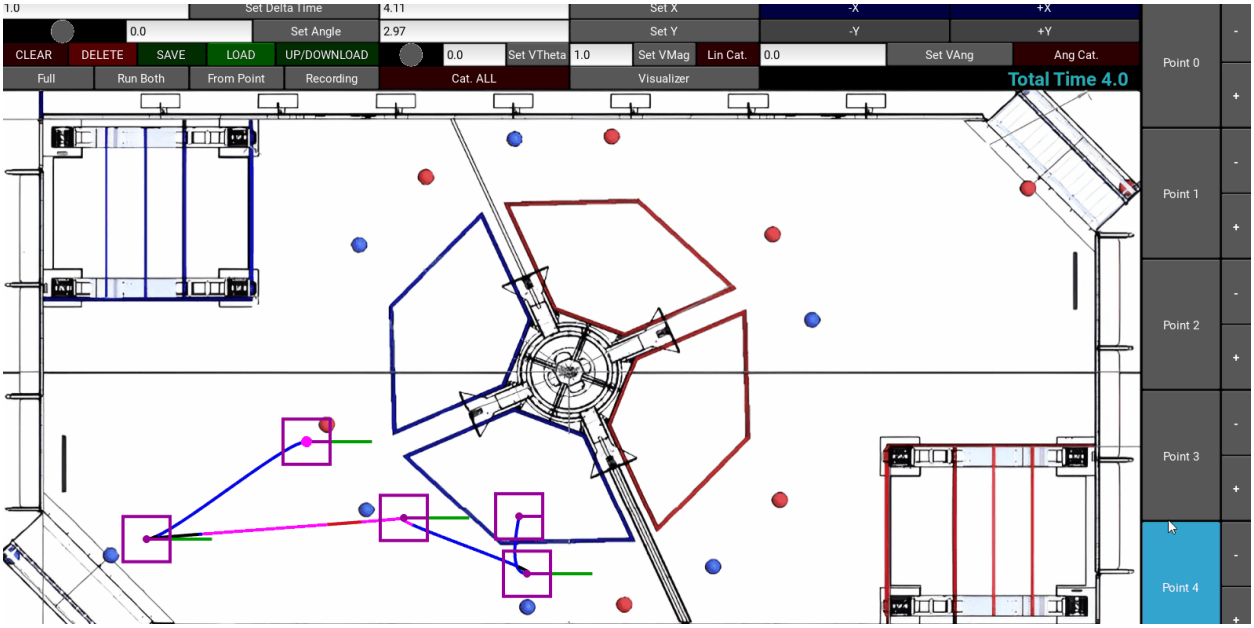
The 2023 Path Tool is capable of creating, refining, debugging, and transferring autonomous path files. Here is a guide that will explain how best to take advantage of the features it has to offer, shown with an example path.

Create a path



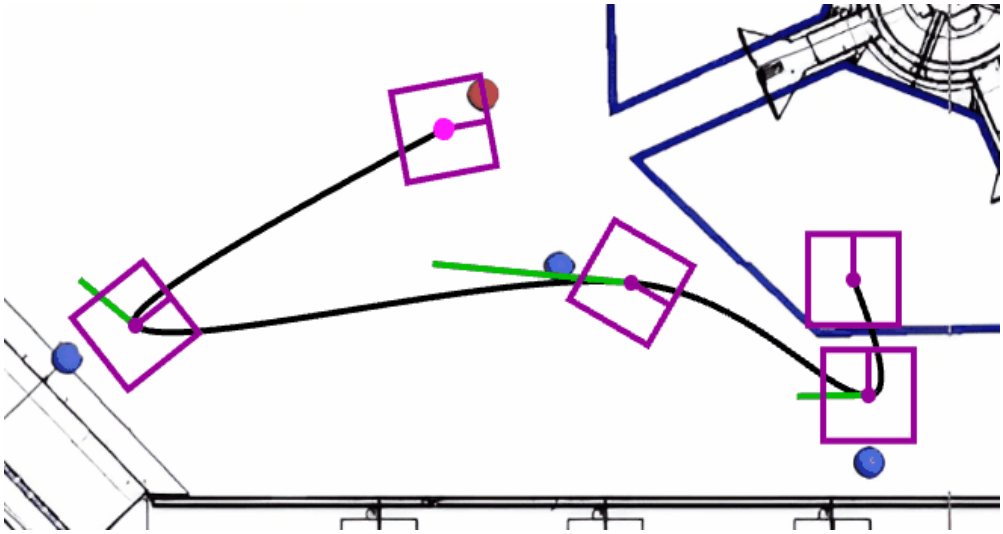
Left-click on the picture of the game field to create new key points. These key points are what define any given path.

Edit key points



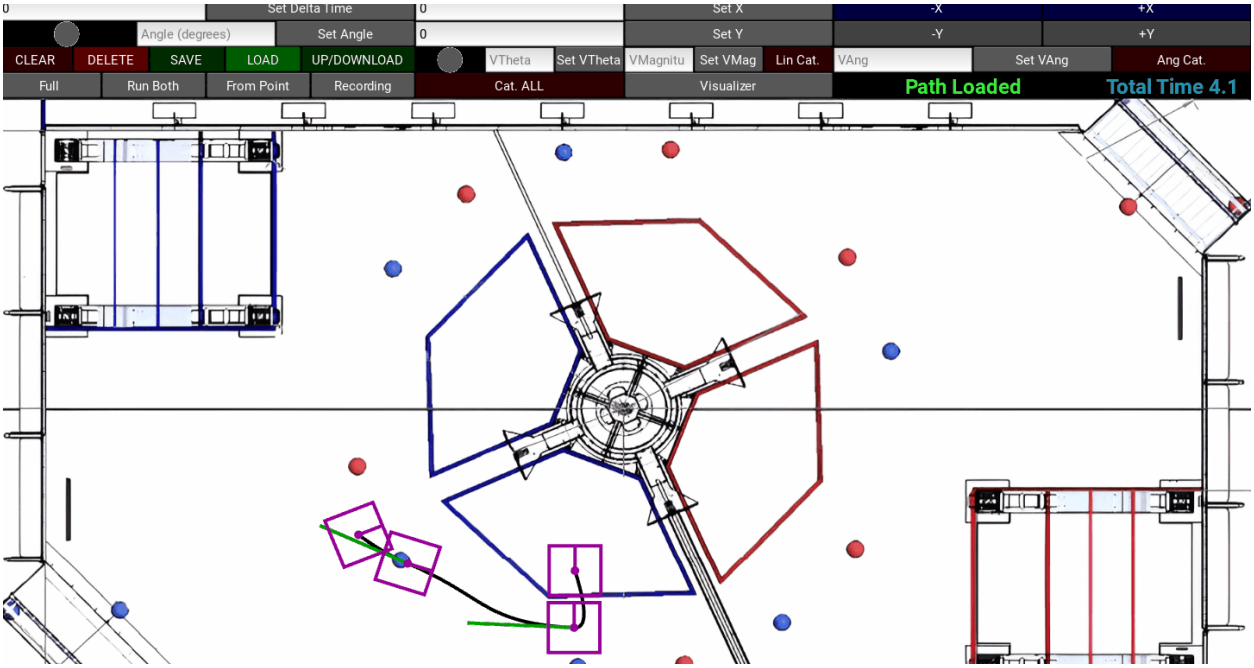
Use controls in the editor and key point menu to select each key point and edit its parameters to your liking. In the example above, each point is rotated, the timings are adjusted, and the whole path is optimized with the Catmull-Rom button.

Make sure it looks right



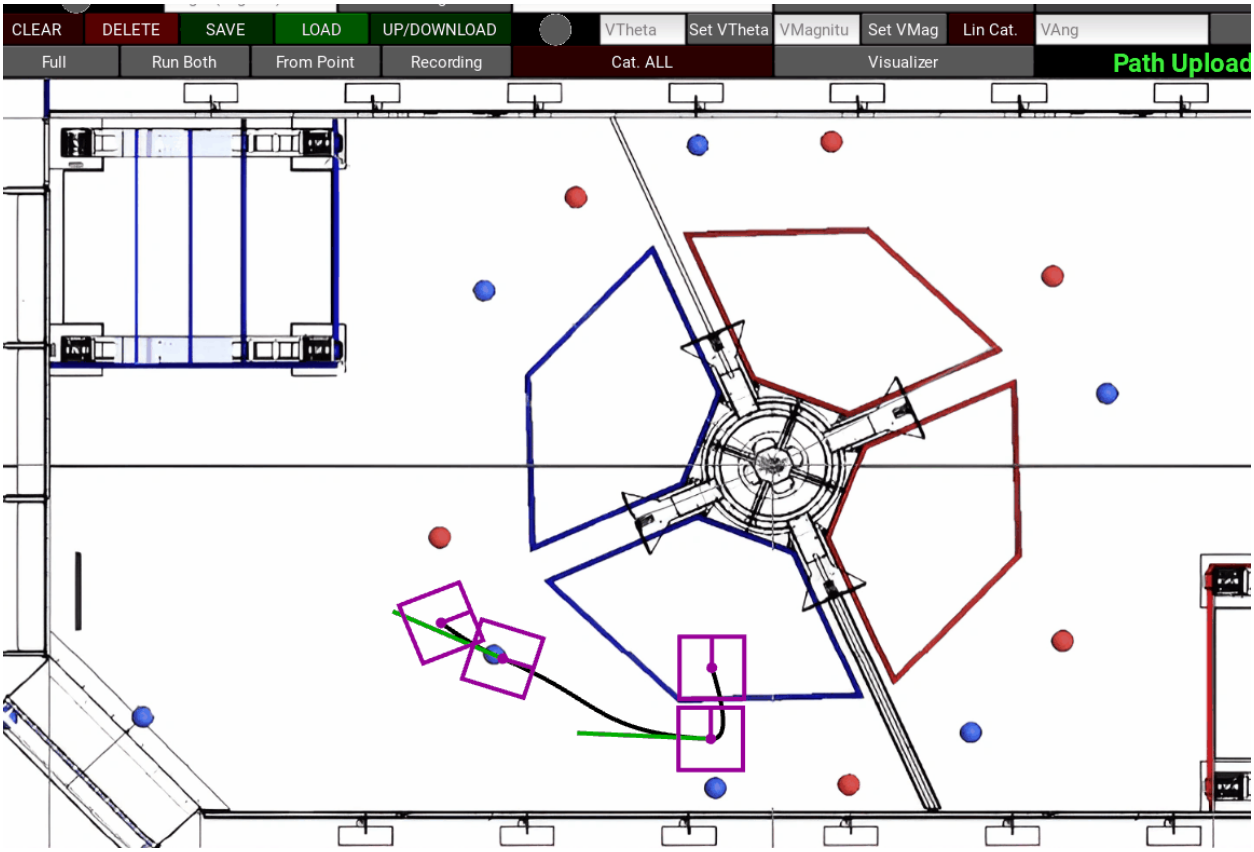
Run the full animation to double check that the path should run how you want it to.

Upload to RoboRio



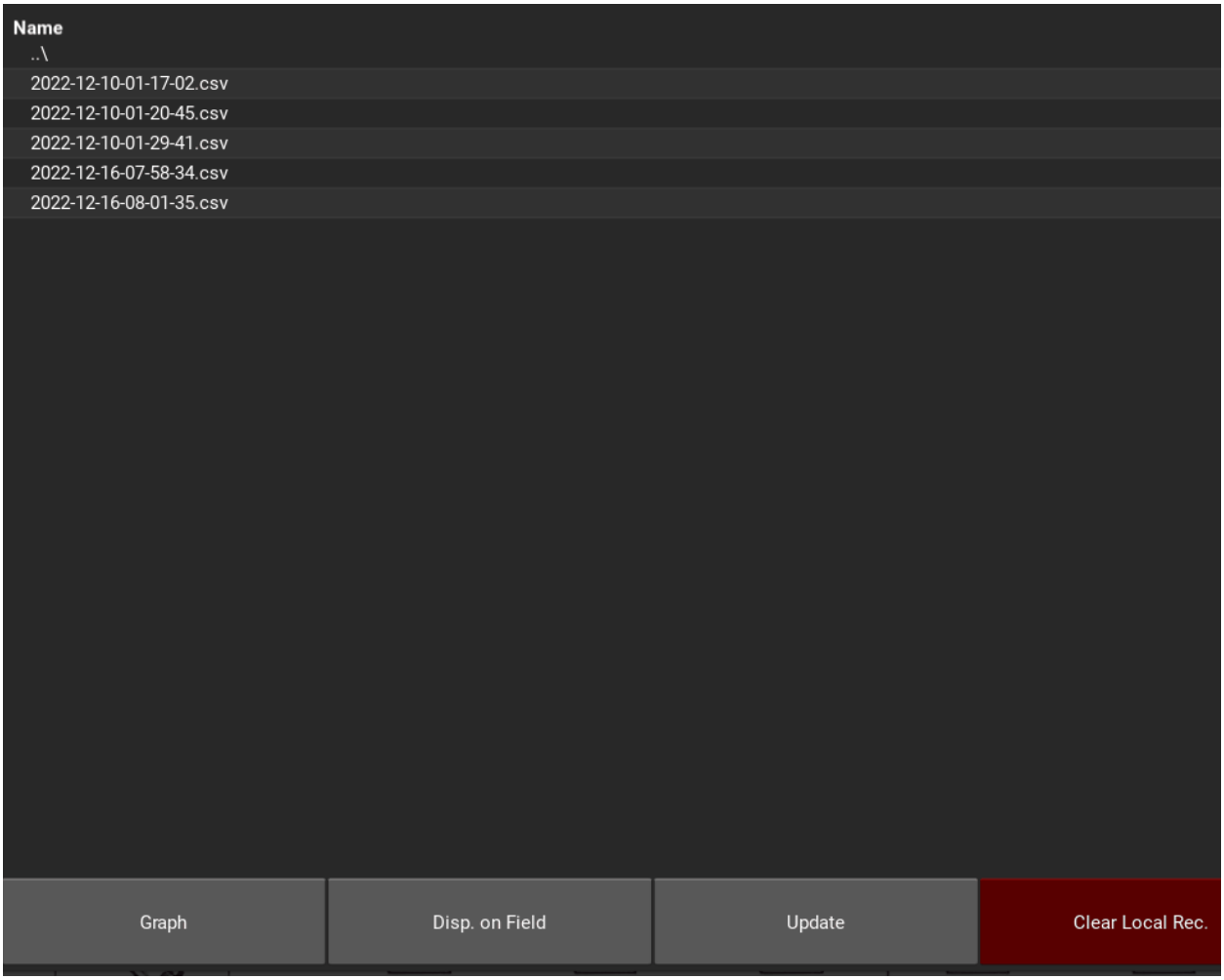
Click the Upload/Download button and upload the path to the RoboRio (must be connected to the robot).

Download recorded odometry

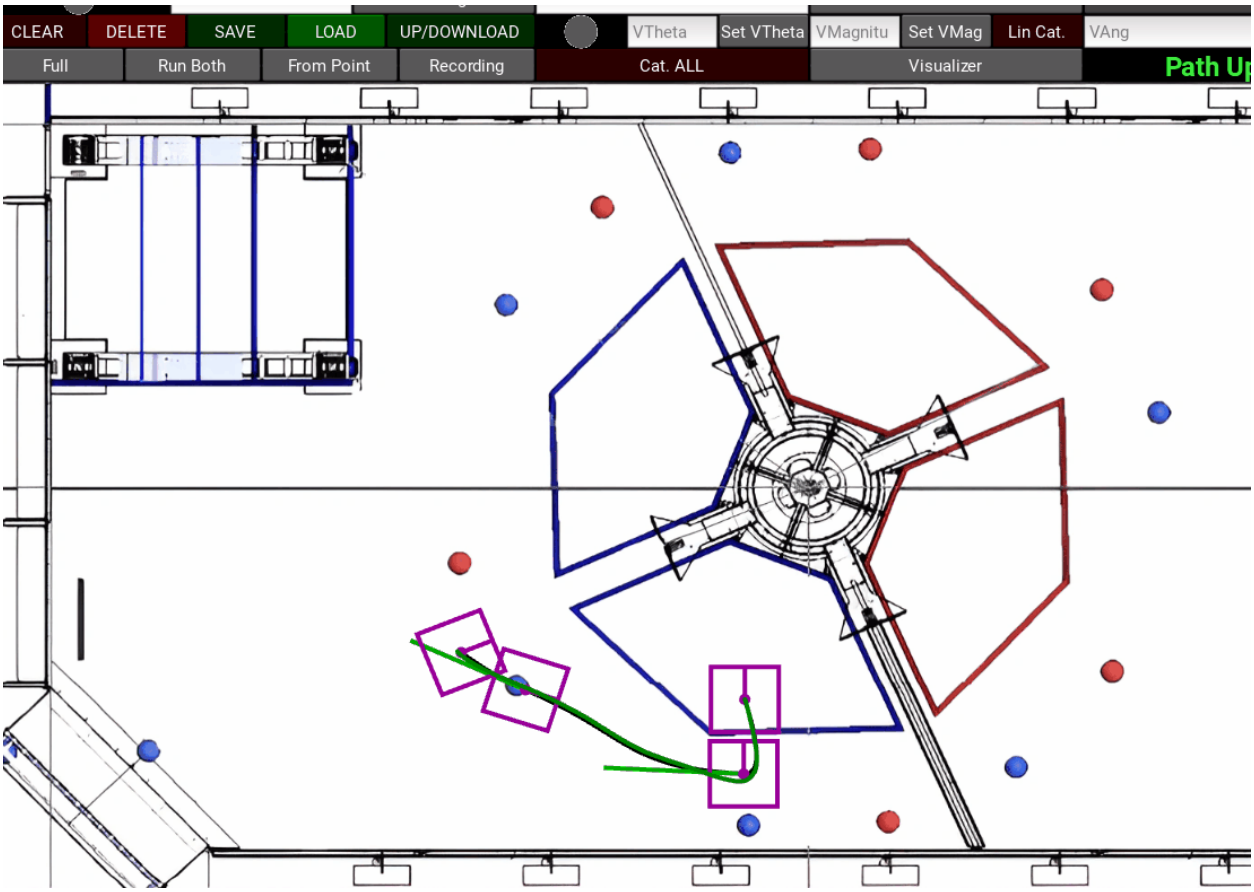


Click the Visualizer button, then click the Update button. This will download all odometry recordings from the RoboRio (must be connected to the robot).

Compare odometry to path



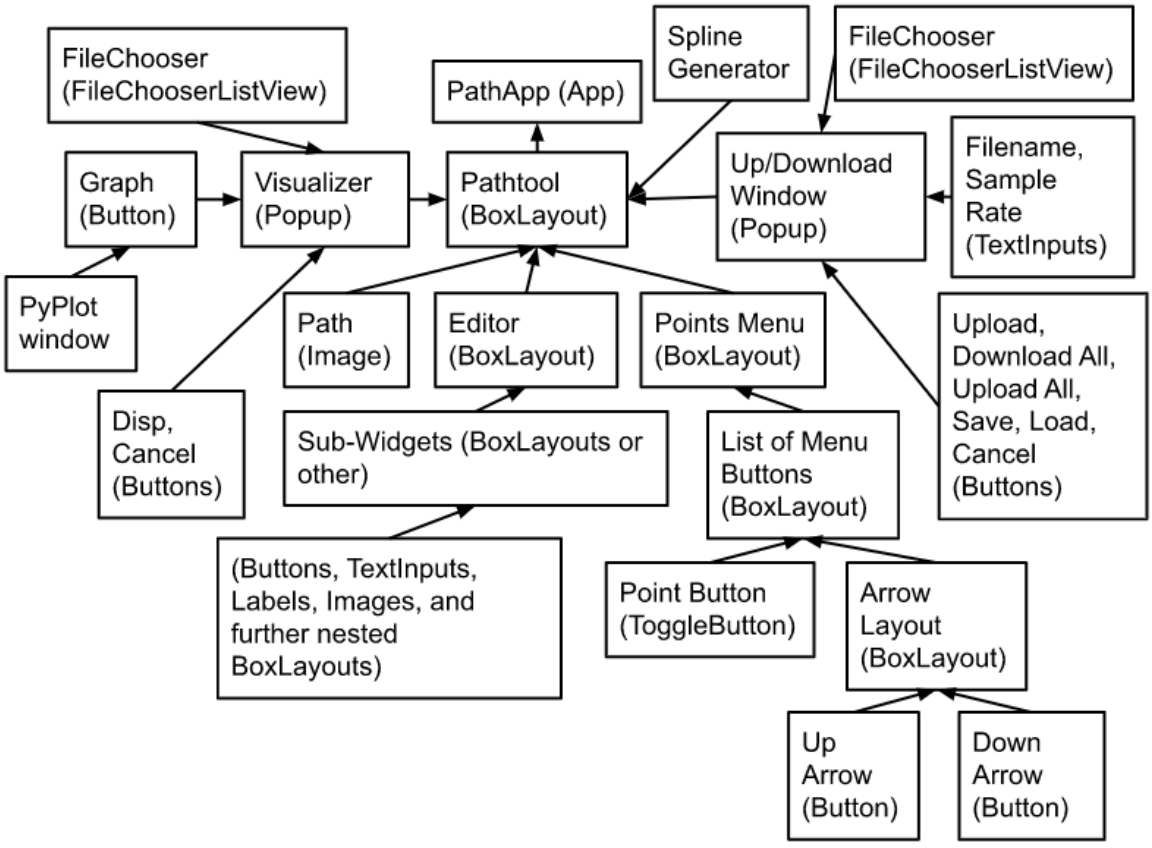
After selecting the desired odometry recording file (the recording files are named by the time the autonomous was run in the example, but they could be named anything after being generated in robot code), click the Display on Field button. This will draw the recorded odometry as a green path. With the recorded odometry and the path file loaded, click the Run Both button to run the animations of both at the same time. This is useful for visually comparing what the robot recorded that it did and what it was supposed to do. Keep in mind that the recorded odometry is not absolute truth; if the odometry accumulates error during the autonomous, the recorded odometry may look like it followed a different path than what the robot actually did.



Click the Visualizer button, then click the Graph button. A graph of the recorded x, y, and theta odometry will pop up. If a path file is loaded (as shown in the example above) , partially transparent lines representing the ideal x, y, and theta will be graphed as well. This is useful for identifying drift in the path, and other debugging such as tuning PIDs.

CODE STRUCTURE

The code for the 2023 Path Tool is written in Python 3.10.2 and uses Kivy 2.1.0 for the GUI framework. Kivy implements a system of widget for constructing GUIs, which provides flexibility and modularity.



This diagram shows the tree of ownership that organizes the different parts of the path tool. The PathApp stores an instance of the Pathtool, which in turn stores instances of the Path, Editor, and Points Menu widgets, which contain nested sub-widgets. For sub-widgets contained in the Editor to control aspects of the Path widget, call-back commands are passed from the Pathtool down to the sub-widget. The sub-widgets can then call that call-back and the Pathtool will update the appropriate widget.

There are also two static method files, Convert and File Manager, that perform extra functions for classes that import them. Convert contains methods for conversion between meters and pixels on the image of the field, calculates distances, and more. The File Manager is used to read, write, and transfer files between the RoboRIO and the local system, as well as parsing save files and recording files into usable forms.

The 2023 Path Tool uses quintic Hermite splines to interpolate between key points in a path. This has the benefit of smooth, continuous motion which reduces robot error when following the path.

Position Equations

There are 3 separate piecewise position equations, which are for x, y, and theta respectively.

Velocity and Acceleration Equations

The first and second derivative equations of the position equations, which represent velocity and acceleration as a function of time, are calculated to provide feedback on the path. This feedback is useful for optimization such as reducing peak accelerations and velocities. The line representing the path is also colored to show where the path is exceeding the kinematic limitations (maximum velocity and acceleration) of the robot. However, the maximum velocity and acceleration of the robot is not separated between x and y components. It is instead an absolute magnitude that is independent of direction.

To generate an equation that gives linear velocity and acceleration as a function of time we must take the first and second derivatives of the distance equation:

$$D(t) = \sqrt{x(t)^2 + y(t)^2}$$

$$V(t) = \frac{d}{dt}[D(t)] = \frac{x(t)x'(t) + y(t)y'(t)}{\sqrt{x(t)^2 + y(t)^2}}$$

$$A(t) = \frac{d}{dt}[V(t)] = \frac{x(t)^3x''(t)+x(t)^2y'(t)^2+x(t)^2y(t)y''(t)-2x(t)y(t)x'(t)y'(t)y(t)^2x(t)x''(t)+y(t)^2x'(t)^2+y(t)^3y''(t)}{(x(t)^2+y(t)^2)\sqrt{x(t)^2+y(t)^2}}$$

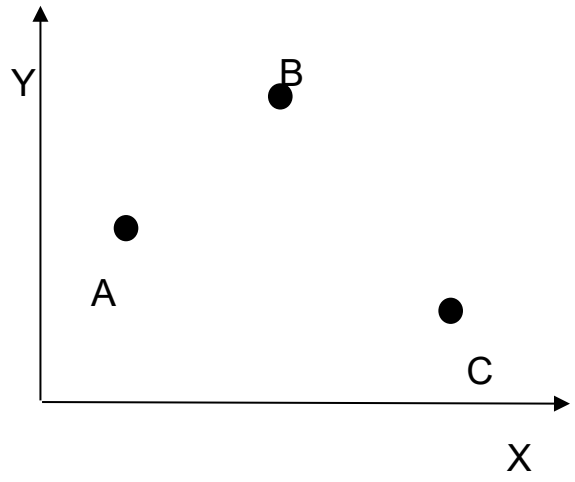
Rather than calculating these monstrous derivatives for each section of the piecewise equations by substituting in the original x and y equations along with their first and second derivatives, each of the original equations is evaluated at the given time and then substituted in.

```
x = float(np.polyval(xEquation, time))

y = float(np.polyval(yEquation, time))

vx = float(np.polyval(xVelEquation, time))

vy = float(np.polyval(yVelEquation, time))

ax = float(np.polyval(xAccelEquation, time))

ay = float(np.polyval(yAccelEquation, time))

return (x ** 3 * ax + x ** 2 * vy ** 2 + x ** 2 * y * ay + y ** 2 * x * ax - 2 * x * y * vx * vy + y ** 2 * vx ** 2 + y
** 3 * ay) / ((x ** 2 + y ** 2) * math.sqrt(x ** 2 + y ** 2))
```

Sampling these linear velocity and acceleration equations by time will give the magnitude of linear velocity and linear acceleration respectively.

Catmull-Rom Splines

The linear and/or angular components of a path can optionally be converted into Catmull-Rom splines, which helps to smooth out corners by optimizing the velocities of key points interior to the path (key points that are not the first or last point). The math to calculate these velocities is fairly simple, just set the velocity of the key point equal to the average velocity based on the key points before and after it.



In the example above, the x-velocity at key point B could be determined with the following equation:

$$x'_B = \frac{x_C - x_A}{t_C - t_A}$$

Angle Optimization

Since quintic Hermite splines interpolation angles between key points as a continuous quantity, the direction of robot rotation between key points is often not optimal because the 180/-180 degree flipping point is not taken into account.

To fix this, the angles of the key points are optimized sequentially by determining which direction requires the least amount of rotation to reach the next angle. If rotation in the positive direction is optimal, the angle of the next key point will be an increase from the current angle, and if negative rotation is optimal, the next angle will decrease. This ensures that the generated path equations will not cause the robot to rotate more than 180 degrees between key points.

```
for i in range(1, len(self.key_points)):
        p1 = self.key_points[i - 1]
        p2 = self.key_points[i]
        p2.angle %= 2 * math.pi
        if p2.angle - p1.angle > math.pi:
            p2.angle -= 2 * math.pi
        elif p2.angle - p1.angle < -math.pi:
            p2.angle += 2 * math.pi
```

Path Save Files

Path files are saved both locally and remotely on the RoboRIO (under the /home/lvuser/deploy/ folder) in JSON files in the format shown below:

```
{
   "meta_data":{
      "path_name": str,
      "sample_rate": float
   },
   "key_points": [
      {
         "index": int,
         "time": float,
         "delta_time": float,
         "x": float,
         "y": float,
         "angle": float
         "velocity_magnitude": float,
         "velocity_theta": float
      },
      ...
   ],
   "sampled_points": [
      {
         "time": float,
         "x": float,
         "y": float,
         "angle": float
      },
      ...
   ]
}
```

There are 3 components of a save file: meta-data, key points, and sampled points.

The meta-data specifies the name of the path and the sample rate (in seconds) that interpolated points are sampled at.

The key points are only included to allow paths to be downloaded and reconstructed by the path tool.

The sampled points are interpolated points in a list for the robot code to follow (in our case with a PID on the drivetrain).

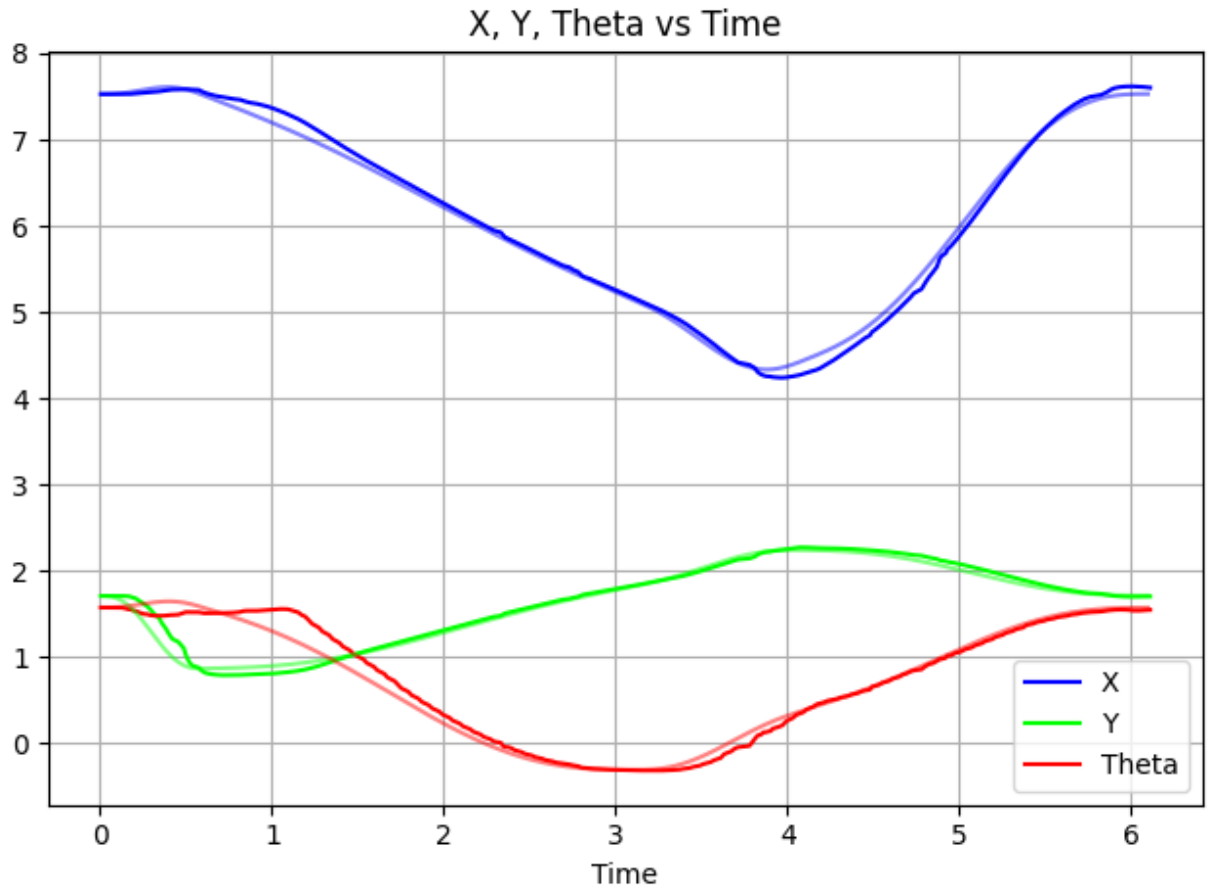
It is important to note that the permissions of the /home/lvuser/deploy/ directory must be changed to allow read and write privileges to all users. To do this, navigate to /home/lvuser/ and run chmod 777 lvuser/.

Recordings

The path tool also has the ability to download and parse CSV files containing recorded odometry information from the robot during an autonomous. Recording files are located under the /home/lvuser/deploy/recordings/ directory. The format for odometry recording files is shown below:

```
time (float), x (float), y (float), theta (float)
```

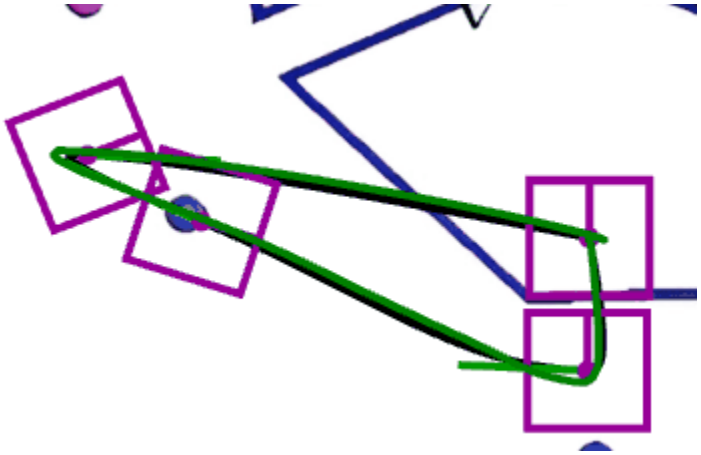
Odometry Graph



Upon selecting a recording file and pressing the graph button, a PyPlot window will pop up displaying a graph of the odometry x, y, and theta over time. The translucent lines are plots of the ideal x, y, and theta over time. These reference lines will only show up if a path is currently loaded in the path tool. Having these reference curves is very useful for tuning drivetrain PID values and identifying areas of the path prone to error.

Recording Animation

Recorded odometry can also be played back as an animation which is very nice for visualizing what the robot thinks it did during the autonomous. Both the animation of the ideal path and the recorded path animation can be played at the same time to help contrast.



The green robot marker represents the recorded odometry and the blue robot marker represents the ideal path.