A Data-Driven Approach to Enhance Robotic Software Through Quality Intelligence

Havish Sripada[1], Sophia Lee[2], Jayden Mei[3], Thilan Wijeratne[2]

revampedrobotics@gmail.com

Abstract

Robotic software plays a critical role in controlling hardware and enabling robots to interact with the physical world to perform complex tasks. The software must evolve to manage real-world uncertainty: sensor malfunction, environmental variation, and hardware imperfections. This paper explores how the integration of Quality Intelligence (QI), a data-driven approach combining automated sensing, control feedback, and predictive modeling, can enhance software quality in robotics.

Drawing from the experience of FTC team RevAmped Robotics, we detail how techniques such as vision-assisted alignment, Inertial Measurement Unit (IMU) based localization, and automated tuning enhanced robot consistency and efficiency. Using encoders, IMUs, and cameras, along with data logging, the team tracked robot movement speed and measured errors. Vision processing enabled object detection and real-time adjustment. Specialized automated tuning programs helped the team tune the robot's parameters for consistency. Virtual physics-based simulations based on predictive modeling streamlined testing and design decisions. By adopting these advanced methodologies, the team significantly improved the robot's autonomous capabilities and overall reliability, demonstrating how QI can serve as a catalyst for smarter, more efficient robotics development.

Biography

The authors are dedicated high school students from Revamped Robotics, a FIRST Tech Challenge (FTC) team from Portland, Oregon, competing at state and world championship levels with distinction. Over eight years, they have advanced to the FTC World Championships five times, earning recognition for innovative designs, programming excellence, and community impact. In 2025, they won the Oregon FTC Championship and the prestigious Inspire Award, the event's highest honor. Their success comes from a refined design and build process that balances speed, efficiency, and top-tier software quality. Beyond competitions, they inspire future STEM leaders through outreach, mentoring, hands-on education, and driving innovation in their community.

[1] Jesuit High School, Portland, Oregon
[2] Sunset High School, Portland, Oregon
[3] Westview High School, Portland, Oregon

1. Introduction

1.1 FIRST Tech Challenge Background

It has long been recognized that experiential and hands-on education provides superior motivation for learning new material by providing real-world meaning to otherwise abstract knowledge. Robotics is an effective tool for hands-on learning, not only robotics itself but also of general topics in Science, Technology, Engineering, and Mathematics (STEM). Learning with robotics gives students an opportunity to engage in real-life problems that require STEM knowledge. FIRST is among the broad spectrum of avenues for pursuing robotics at the pre-university level to promote STEM worldwide. FIRST stands for "For Inspiration and Recognition of Science and Technology" and is an international youth organization focused on developing ways to inspire students in engineering and technology fields [1]. The FIRST Tech Challenge (FTC) is designed for students in grades 7–12 working in teams to compete on a playing field. Teams are responsible for designing, building, and programming their robots to compete in an alliance format against other teams. The robot kit is programmed using Java. Teams, including coaches, mentors, and volunteers, are required to develop strategies and build robots based on sound engineering principles. The goal of FTC is to reach more young people with an accessible opportunity to discover the excitement and rewards of STEM.

1.2 Match Objectives

The FTC competition field is 12' x 12'. Each match is played with four randomly selected teams, two per alliance. Four 18" x 18" robots must be able to navigate around each other without breaking when hit by another robot.

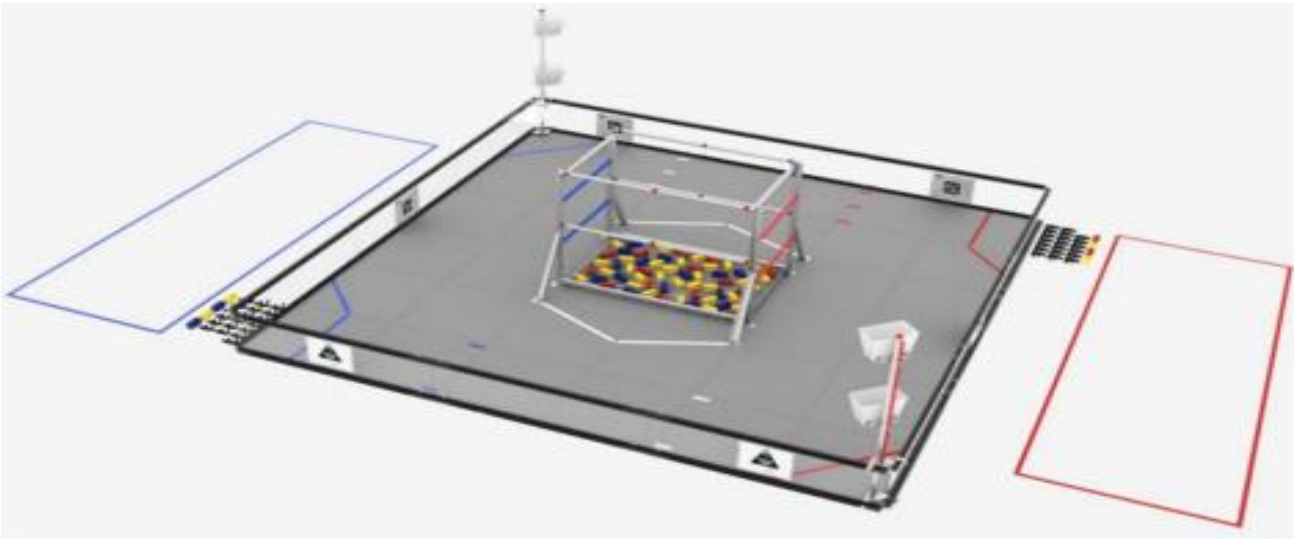


Figure 1. FTC field setup for the 2023-2024 FTC season [2]

Figure 1 shows the field setup for the 2024-25 FTC season. The FTC robot game is composed of two phases: (1) the autonomous phase and (2) the driver-controlled tele-operation phase. Throughout the two periods, the robot intakes scoring elements, called samples, from a large structure in the center of the field called the submersible. The robot can then score by either depositing these samples into baskets or attaching these samples to black clips to convert them to "specimen", which can be scored on the rungs, the colored trusses at the sides of the submersible. During the last 30 seconds of the game, the robot can gain additional points by fully suspending itself off the ground by hanging from the top of the submersible.

1.3 Autonomous Optimization

The thirty-second autonomous period has always played a crucial role in maximizing points in a match. This season, its importance increased significantly, as points scored during the autonomous period doubled.

Thus, the team focused on enhancing the reliability and precision of the robot's autonomous pathway by tackling the foundations of accurate and consistent autonomous performance: localization, robotic intelligence, and path-following.

The rest of the paper is organized as follows: we cover the enhancements done to localization methods in Section 2, application of vision processing in Section 3, a custom fastest path algorithm in Section 4, and enhancements to robotic subsystems through automated tuning in Section 5.

1.4 Quality Intelligence (QI)

The standard definition of quality intelligence refers to the integration of data-driven approaches like sensory input, data logging, and predictive modeling to analyze, monitor, and improve system software quality. In robotics, QI plays a vital role in the creation of high levels of software quality, enabling robotics systems to respond to real-world dynamic environments by enhancing consistency, adaptability, and precision.

2. Optimizing Autonomous Localization

In past seasons, the team utilized three encoder wheels with odometry [3] to track the robot's change in position on the field. One wheel would be used to measure the robot's forward motion, another its sideways motion, and the last one its heading. Even though the encoders worked relatively well, the team often experienced issues such as wheel slippage, collisions with game elements, inconsistent contact with the field surface, and other problems, which reduced the accuracy of the encoder's position measurements and caused the robot to drift from the desired autonomous path.

2.1 Implementing Pinpoint Sensor and Path Correcting Library

To remedy these issues, the team decided to implement the GoBilda Pinpoint sensor [4] in conjunction with Pedro Pathing [5], a software library for trajectory following. The pinpoint sensor features a built-in IMU that directly measures the robot's precise rotation speed and direction, resulting in significantly more reliable heading tracking. Having an accurate way to track heading is crucial, as even a 2-degree angular error can cause the robot to move diagonally and drift away from an intended straight path. Additionally, the GoBilda Pinpoint sensor measures the robot's position at a frequency of 1,500 Hz, which is approximately five times faster than encoder wheels that measure around 300 Hz. This faster update rate per second enables the robot to track its position on the field more accurately, as there is less time for errors to accumulate between updates. When the pinpoint sensor is paired with Pedro Pathing, the system demonstrates Quality Intelligence by continuously evaluating its position and heading, comparing them to the intended trajectory, and adjusting motor power to correct deviations. This enables the robot to maintain a precise path autonomously throughout the auto period.

2.2 Localization Performance Experiment

To test how effective the Pinpoint sensor is compared to the traditional three encoder wheels for tracking the robot's position, heading, and accuracy in reaching the intended destination, the team ran an experiment using both localization methods on the same drivetrain. The robot was pre-programmed to follow a path that includes straight lines, turns, speed changes, and a destination, simulating a real autonomous path. The field contained randomly placed scoring elements to account for possible collisions that could occur in a real match. The team ran the pre-programmed path 30 times, 15 times using only the Pinpoint sensor and 15 times using only the encoder wheels, and recorded the distance and orientation from the robot's final position to the intended destination after each run. The experiment was designed to test and compare the accuracy, consistency, and error correction abilities of each localization method. Figure 2 shows the results of the experiment.

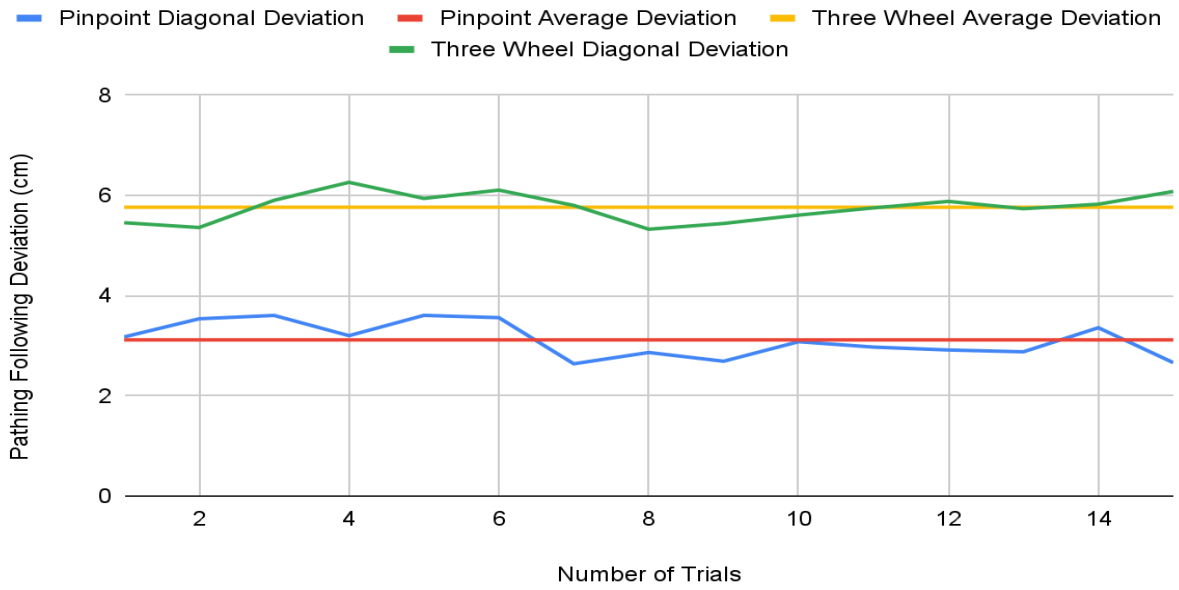


Figure 2. Pinpoint path-following Vs. Three-Wheel path-following

The graph clearly shows that the average path deviation when utilizing the three encoder wheels is approximately 5.8 cm, nearly twice that of the Pinpoint sensor, which is around 3.1 cm. There was no significant difference in heading errors of the two localization systems. This indicates that the Pinpoint sensor, when paired with Pedro Pathing, significantly improves the robot's path-following accuracy compared to the conventional three-wheel encoder system.

3. Vision Processing and Robot Intelligence

To further enhance the autonomous performance of the robot, the team implemented computer vision to facilitate the robot's intelligence in identifying and intaking game elements. One of the most significant challenges in the autonomous period of FTC is adapting in real time to field variables. Being based fully on preprogrammed instructions, the autonomous needed to be input-driven to ensure adaptability. Computer vision integrated with OpenCV [6] enables our robot to accurately detect, classify, and react to game elements, especially in the submersible. This amount of robot processing and information enables the robot to make decisions autonomously, accurately, and in real time.

3.1 Implementation of Computer Vision

The first step to our computer vision, and thus robot intelligence, is object detection. Our team wanted a fast, effective, and customizable program for object detection; the classical vision-processing algorithm OpenCV fulfilled all these requirements. The team employed OpenCV to isolate, spot, and blob specific regions on the dozen frames taken by our robot's built in camera. Through a combination of color-processing, edge detection, and block contouring, the robot can differentiate between field elements such as other robots, the submersible, and color-specific blocks. To remain consistent, we utilize a panel of LED lights on the camera to ensure similar lighting conditions. Once detection is achieved, positional data about the identified objects is calculated and passed through a filtering algorithm to help confirm confidence and accuracy. Using positional data, the robot can adjust its planned movements in real time. For example, if a scoring element was slightly misaligned from its expected position, the robot could recalculate the angle of approach or delay intake actions until the alignment was verified effectively with sensory input on board.

3.2 Computer Vision Experiment

To validate the effectiveness of our robot intelligence approach, our team designed an experiment to compare the performance of the robot's intake system with and without vision assistance. In this controlled test, the robot attempted to collect a game element using pre-programmed motion alone and then repeated the same task using visual feedback to adjust alignment in real time. By measuring the average time intake and the success rate across trials, we aimed to quantify how intelligent sensing contributes to consistent, high-quality behavior in robotic systems.

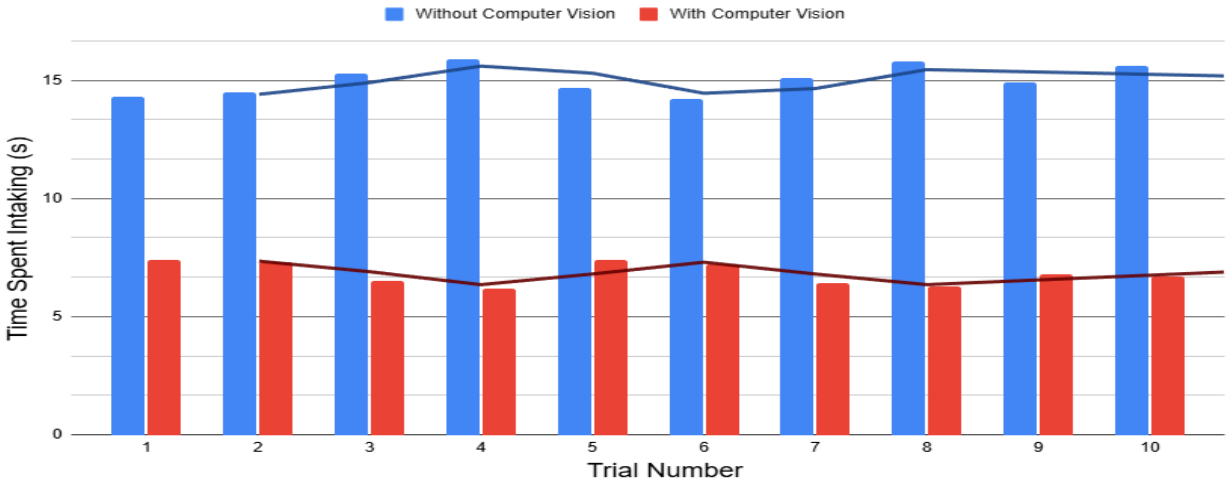


Figure 3. Time to intake with Vision Vs. without Vision

The results of the experiment were graphed (Figure 3), comparing the same robot base with and without vision, showing a great reduction in time spent intaking, nearly a 55% decrease. The experiment shows that vision-assisted control drastically improved the robot's speed and consistency in collecting game elements. By enabling the robot to react in real time to slight misalignments or variability on the field, the camera-based vision system significantly enhanced performance. This confirmed that adding computer vision contributed directly to higher-quality, intelligent robotic behavior during autonomous tasks.

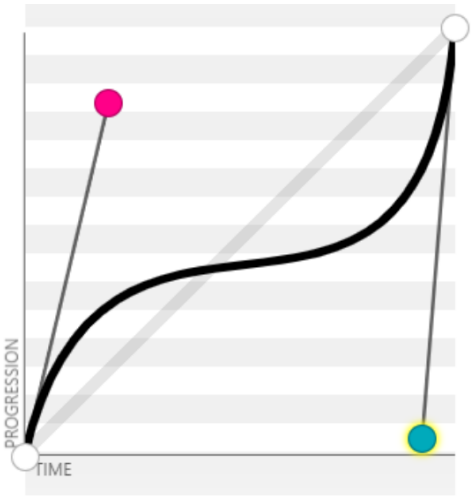
4. Simulation-Based Predictive Modeling

To ensure optimal system performance, the team applied predictive modeling to develop and validate software algorithms. Robotic software can often be enhanced by optimizing hyperparameters, allowing robots to complete actions more efficiently and effectively. The autonomous segment of an FTC match mandates the robot to drive through the field without human driver control, and therefore, optimizing the trajectory of the robot can allow faster intaking and scoring of elements. The team therefore attempted to solve the problem of finding the fastest trajectories for the robot to traverse between any two points on the field. Accounting for the obvious difficulties involved in physically running and comparing several different possible curves the robot can take between points, the team employed virtual simulations to predict the most optimal solutions.

4.1 Mecanum Kinematics

The first step to creating such a simulation was to define the task and constraints. The goal of the simulation was to determine the fastest curve the robot could take to drive between any two points on the field while avoiding any obstacles. In the FTC competition, the most used drivetrain, or mechanism the robot uses to drive across the field, is the mecanum drivetrain [7]. This drivetrain sacrifices some movement speed to produce additional utility in the form of omnidirectional movement, meaning the robot can produce complex movements like strafing (moving sideways without turning), and can move independently of its heading orientation. However, due to friction, the robot cannot achieve the same speed while strafing compared to traveling straight forward and backward. The robot's heading therefore directly impacts its driving speed,

since the robot achieves higher maximum velocities when its heading aligns more closely with the tangent to the curve, although the robot moves more slowly while trying to achieve a more ideal heading. Since the heading introduces a new variable in optimizing the path-following speed, the simulation also needed to predict the ideal rotational changes, more formally known as heading interpolations, the robot should take while following the curve. We use the term path to mean a curve and heading interpolation that the robot follows, and use the terms path-following speed and path-following time to mean the speed at which the robot drives along the path, and the time the robot takes to drive through the path, respectively. To constrain the search to accommodate the path-following limitations of a typical FTC robot, the simulation restricted paths to Bézier curves [8] of dimension 3 or smaller. Figure 4 displays the image and control points of a sample cubic Bézier curve.



Cubic (2 Control Points) Bézier Curve Example
Created using https://cubic-bezier.com/

Figure 4. Sample Cubic Bézier Curve

4.2 Path Simulation

The simulation evaluated and compared paths based on an approximation of how long the robot would take to drive through them. To quantify such a number, the simulation had to use rigorous kinematic modeling of the physical system and the outside forces that impact the robot during path-following. The team determined a function for evaluating the robot's velocity at any given time in terms of the robot's maximum velocity and then created a differential equation to find the amount of time needed to travel the length of the path, which was computed using standard integration techniques. To create a more realistic representation of the system, the team accounted for the speed reduction created while the robot was turning, as well as the, frictional force to rank the viability of different heading interpolations. Therefore, given any path, the team approximated the solution to this differential equation to derive a numerical metric for the robot's estimated path-following time. We define the time function as the function that associates with every path the time the robot would theoretically take to traverse it.

4.3 Optimization and Predictive Modeling

Using the time function, the simulation could compute quantifiable means of ranking the effectiveness of each path. The team thus used this data to perform derivative-free optimization on these curves, applying a deterministic search through this data using the Constrained Optimization by Linear Approximation (COBYLA) [9] predictive modeling approach. This algorithm created linear approximations of the time function using local simplices of points and then created trust regions given by Euclidean balls in the domain, where these linear approximations would locally perform well. The algorithm then iterated through creating new simplices, evaluating the minimum time values of each linear approximation, and expanding trust regions to eventually encompass the entire domain and converge to a global minimum for the function. We refer to this path-optimizing methodology as the Fastest Path Algorithm (FPA). For each unique section of the autonomous segment where long paths were necessary, the team applied the FPA and deployed the resulting paths to the

robot. This data-driven predictive modeling approach therefore allowed the team to minimize the robot's driving time and score the most points.

4.4 Analysis of Results

The most important path the team needed to optimize was the path between the basket and the long side of the submersible. This was the longest path the robot needed to follow autonomously, typically several times during the 30-second period, and therefore was most beneficial to optimize. The team iterated through their FPA velocity model throughout the season, producing more efficient paths to the submersible each time. Figure 5 shows the robot's average path-following time to the submersible throughout the various competitions the team participated in through the season. Note that the team's first competition was omitted since the robot didn't have autonomous capabilities that early in the season.

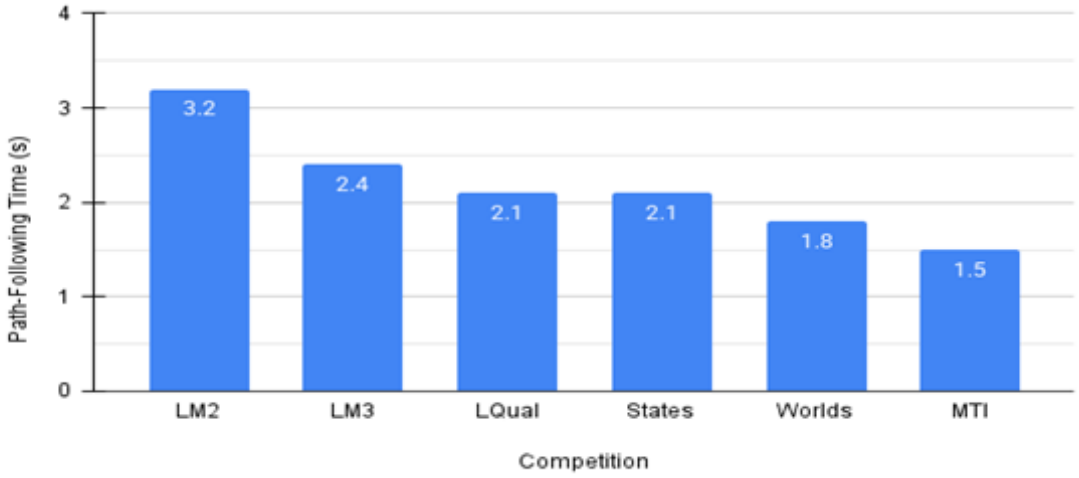


Figure 5. Path-Following Times Through the Season

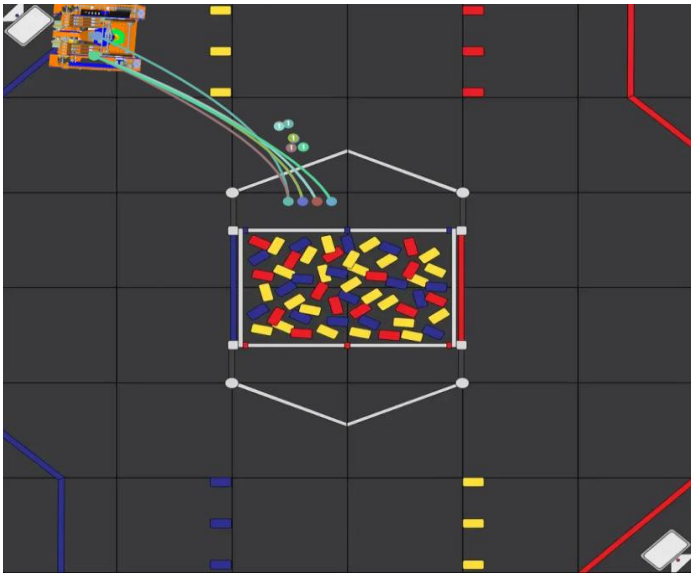


Figure 6. Final Paths to the Submersible

The lack of improvement between the league qualifying tournament and state championships is due to the team's programmers focusing on tasks other than FPA, whereas after the state championships, the team focused more on optimizing this path. The graph shows significant improvement in the robot's speed, which can be directly attributed to the team's application of Quality Intelligence through the Fastest Path Algorithm. Figure 6 displays the robot's final paths to the submersible at the Maryland Tech Invitational (MTI) competition [10], which was the team's last tournament in the season.

5. Automated Tuners

To enhance system efficiency and longevity, the team implemented multiple automated tuning processes, including SquID controller adjustments and servo-motor calibrators, supporting consistent hardware progression.

5.1 SquID Controller

To achieve precise and efficient movement of robot mechanisms, the team implemented SquID Controllers [11], which enable quick and accurate motor positioning. These controllers use rotary encoders to move motors along a trapezoidal motion profile, based on error responses. Specifically, the control signal is calculated by multiplying the square root of the positional error by a proportional factor. A SquID Controller's trapezoidal motion profile and end-effecting position reference can be seen in Figure 7. Since optimal performance depends on tuning this factor, and motor applications vary across the robot, the team developed multiple automated methods to adjust it for each use case.

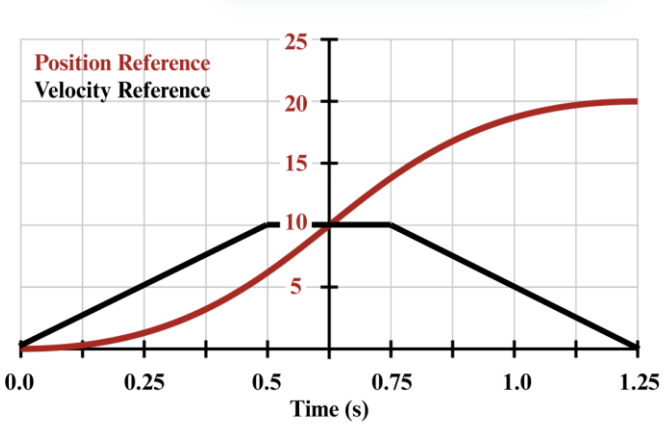


Figure 7. Trapezoidal Motion Profile

5.2 Automated SquID Tuning Methods

The first tuning method implemented is Load-Based Adaptation [12]. Utilizing live data from rotary encoders and current sensors, motor load estimations are calculated. Since higher loads dampen a motor's responses to error, logarithmic scaling is applied to the proportional factor depending on the change in load. The second method implemented is a Ziegler-Nichols-Inspired [13] heuristic process. To tune this factor, the robot routinely and automatically completes a specific motor action multiple times to estimate ultimate gain, continually increasing the proportional factor until detrimental system oscillations or overshoots are detected through rotary encoders and current sensors. To determine a stable and responsive gain, the proportional factor is set to the ultimate gain divided by two, as seen below in Figure 8, ensuring quick and smooth movement.

$$P = U/2$$

Figure 8. Responsive Gain Derived from Measured Ultimate Gain

The third method implemented is an Error-Dynamics process [14]. Utilizing pre-determined measurements of optimal error size for the specific motor application, the automatic SquID tuner adjusts its proportional factor depending on the current measurements of average error size. If the current error average is higher than the optimal error size, the proportional factor is increased through linear scaling. If detrimental overshoots are detected, the proportional factor is decreased with the same linear scaling process. By implementing these automated SquID tuners, motor-actuated mechanisms used by the robot reported a 36% improvement in average speed and a 3.4 Ampere reduction in action-induced current draw.

5.3 Servo-Motor Position Tuner

To ensure mechanism consistency and longevity, the team implemented and routinely utilized automated servo-motor calibrators. FTC servomotors generally have 255 discrete positions, which are used for precise mechanism movements. Over time, it is essential to tune these positions for servo longevity routinely. This process moves a servo to each of its implemented pre-calibrated positions and analyzes the current drawing using current sensors on each to determine possible risks of servo strain. If a servo position triggers a current draw greater than a safe threshold, the position is then decreased until the current draw noticeably decreases or increases further. If the current draw decreases, the servo position will continue to decrease until its triggered current draw is under a safe threshold. If the current draw instead increases further from the safe threshold, the position is then increased rather than decreased until the triggered current draw is within the safe threshold. A flowchart representing this automated process can be seen below in Figure 9. This process ensures optimal and consistent servo-motor positions on every robot mechanism, in turn increasing servo-motor longevity and reducing current draw by a system total of 1.3 Ampere.

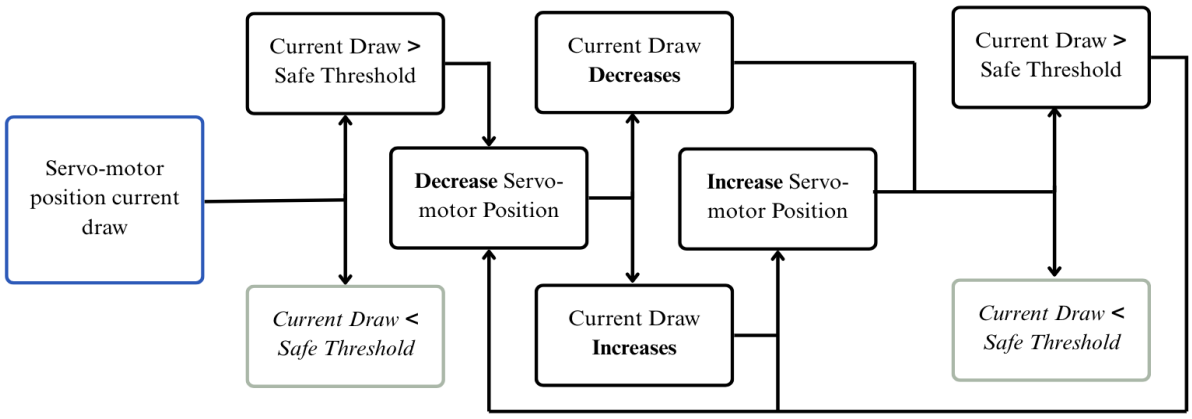


Figure 9. Flowchart Representation of the Team's Automated Servo-motor Position Tuner

6. Quality Intelligence and Software Quality

The implementation of quality intelligence greatly helped the team in improving the robot's software quality, which the team measured through three metrics: maintainability, repeatability, and debuggability. These three factors allowed the robot's software to perform more reliably and consistently through the team's various competitions.

6.1 Maintainability

An FTC robot's hardware mechanisms typically feature high-maintenance components and require frequent recalibrations to enable consistent performance. The team applied the quality intelligence methodology to create automated tuning software, using external sensor information to recalibrate servo positions and SquID controllers. These automated programs enabled the team to efficiently keep up with the schedule of regular maintenance needed for the robot's subsystems. Additionally, the team made automated testing programs that helped the robot efficiently self-diagnose any faults in its hardware components. This allowed the team to quickly adjust the robot before important events, ensuring that the robot would always be in pristine condition.

6.2 Repeatability

The autonomous period of an FTC robot requires the robot to execute precise movements and produce the same results several times without failure. The robot's ability to self-diagnose itself facilitated a reliable performance in-match, allowing the robot to make real-time adjustments to the behavior of its physical mechanisms. The robot uses data from servo current and errors in motor encoders to determine how much power it should give to these components in-match. In particular, the robot stores information from previous scoring cycles inside a match to adjust its control loops in future scoring cycles. In the driver-controlled period, the robot also uses information about the driver's actions to determine the reliability of its sensor outputs and adjust its behavior accordingly. This allows the robot to always perform well inside the competition, regardless of any faults that might occur in its calibrations. The team also recognized that changes to the robot's software could affect the repeatability of the robot's autonomous path-following, so the team utilized virtual simulations to test any changes and ensure consistency.

6.3 Debuggability

The team used logging software and telemetry to detect and debug any issues in the robot's software. Using information from sensors, the team was able to graph the states of the robot's different mechanisms and determine discrepancies in the expected and experimental states. The team also used this information to determine issues in the boolean conditions that determined when transitions between states would occur, for example, the necessary readings on the GoBilda Pinpoint Sensor to determine when the robot had finished its path-following. This process enabled the team to review information on match logs after each practice run to find and resolve any bugs in the robot's software.

7. Summary

This paper explores the usage of a data-driven approach to optimizing robotic software performance. The team integrated several software methodologies to increase the robot's intelligence and autonomously tune and optimize parameters. The team conducted experiments to show how quality intelligence enhanced various subsystems of the robot. These techniques can be applied to industrial robots to improve their precision and functionality.

References

[1] "Home." FIRST. Accessed June 21, 2024. https://www.firstinspires.org/.

[2] 2024-2025 FIRST Tech Challenge into the Deep Competition Manual. Manchester: For Inspiration and Recognition of Science and Technology, n.d. Accessed June 21, 2025.

[3] GeeksforGeeks. "Odometry in Robotics | Introduction to Machine Learning." Last modified January 18, 2023. https://www.geeksforgeeks.org/machine-learning/odometry/.

[4] goBILDA. Pinpoint™ Odometry Computer: IMU Sensor Fusion for 2-Wheel Odometry. Accessed June 18, 2025. https://www.gobilda.com/pinpoint-odometry-computer-imu-sensor-fusion-for-2-wheel-odometry/.

[5] PedroPathing. PedroPathing: Pathfinding and Odometry Resources for FTC and FRC Robotics. Accessed June 18, 2025. https://pedropathing.com/.

[6] “About.” OpenCV, 4 Nov. 2020, opencv.org/about/.

[7] Taheri, Hamid, Bing Qiao, and Nurallah Ghaeminezhad. “Kinematic Model of a Four Mecanum Wheeled Mobile Robot.” International Journal of Computer Applications 113, no. 3 (March 18, 2015): 6–9. https://doi.org/10.5120/19804-1586.

[8] Hoffman, Gernot. Bézier Curves, 2002.

https://web.archive.org/web/20061202151511/http://www.fho-emden.de/~hoffmann/bezier18122002.pdf.

[9] Powell, Michael James David. “A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation.” Springer EBooks, January 1, 1994, 51–67. https://doi.org/10.1007/978-94-015-8330-5_4.

[10] FIRST Chesapeake. “Maryland Tech Invitational 2025,” 2025. https://www.firstchesapeake.org/event-details/maryland-tech-invitational-2025.

[11] FTC Escape Velocity. 2025. “SquID Controller Explanation Video.” YouTube. March 11, 2025. https://www.youtube.com/watch?v=WA9o3e4a01Q.

[12] Gyöngy, I.J., and D.W. Clarke. 2006. “On the Automatic Tuning and Adaptation of PID Controllers.” *Control Engineering Practice* 14 (2): 149–63. https://doi.org/10.1016/j.conengprac.2005.01.007.

[13] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," Transactions of the ASME 64, no. 11 (1942): 759–768.

[14] Frantisek Kudlacak, and Tibor Krajcovic. 2016. “Error Behaviour in PID Control Systems with Dynamic Processes.” *Digital Library (University of West Bohemia)*, September, 141–44. https://doi.org/10.1109/ae.2016.7577259.