Revamp Software Quality for Android Robots through Advanced Development and Deployment Methodologies

Theodore Guo[1], Monica Bao[2], Sophia Lee[3], Kyle Zhou[4], and Bo Li[5]

revampedrobotics@gmail.com, bo.a.li@intel.com

Abstract

Robotics software quality is crucial to control robot hardware while interacting with the physical world to accomplish a set of complex tasks. Software is evolving to tackle sensor errors, robot software/hardware failures, human errors, and uncertainties from the external environment. This paper describes challenges to revamp the software quality for an Android platform robot based on the experience of the First Tech Challenge (FTC) robotics team, Revamped Robotics.

FTC is an annual competition for pre-college students with thousands of teams across the world competing to solve real world challenges through firsthand robotics experiences. FTC competitions begins with a two-month period between when the competition challenge is released and the first competition. The FTC season is roughly eight months long consisting of multiple periods of robot performance improvements between FTC competitions. To assess the quality of suggested robot changes, the team employed the SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis methods to enable a smooth coordination between the robot hardware and software. The team utilized agile development processes, the cloud-based GitHub software management platform, and Continuous Integration/Continuous Deployment (CI/CD) methods to enable frequent software improvement cycles. In addition, the use of deep learning TensorFlow with robot embedded sensors/actuators increased the effectiveness of software development.

Biography

Authors are high school students from Revamped Robotics, a seven-year veteran FIRST Tech Challenge (FTC) team from Portland, Oregon. They advanced to the FTC World Championships three times and received recognitions for their competitive robot, software programming and contribution to the local, national and international community. The team developed a fast and streamlined design and build process while ensuring software quality throughout the competition season. They also focused on outreach activities in educating young students in communities about Science, Technology, Engineering and Mathematics (STEM) through Tech Talks, community open houses, and mentoring newer robotics teams.

Bo Li works in software development and management for Technology Development Analytics and Automation at Intel. He joined Intel in 1999 after receiving Ph.D. from Georgia Institute of Technology. Since then, he has been focusing on software development, quality, performance, and testing for a variety of software applications. Outside of work, he volunteers to mentor robotics teams in the community for high school and middle school students to foster passion and knowledge in STEM.

[1] Jesuit High School, Portland, Oregon
[2] Beaverton Academy of Science and Engineering, Beaverton, Oregon
[3] Sunset High School, Portland, Oregon
[4] Sunset High School, Portland, Oregon
[5] Technology Development Analytics and Technology Automation, Intel Corporation

1 Introduction

FIRST is an international youth organization to develop ways to inspire students in engineering and technology fields [1]. FIRST stands for "For Inspiration and Recognition of Science and Technology" and is among the broad spectrum of avenues to pursue robotics at the pre-college level and promote Science, Technology, Engineering, and Mathematics (STEM) around the world. The FIRST Tech Challenge (FTC) is designed for students in grades 7–12 to compete using a sports model. Teams are responsible for designing, building, and programming their robots to compete in a 2-team alliance format against other teams. Teams, including students, coaches, mentors and volunteers, are required to develop their strategy and build robots based on engineering principles. The ultimate goal of FTC is to reach young students with an accessible and low-cost opportunity to discover the excitement and rewards of STEM.

1.1 Background for FIRST Tech Challenge (FTC) Robot Programming

FTC game matches are played on a 12 foot by 12 foot playing field. Each match is played with 4 randomly selected teams, 2 teams per alliance. Four 18" x 18"x 18" robots must be able to navigate around each other without breaking when hit by another robot.

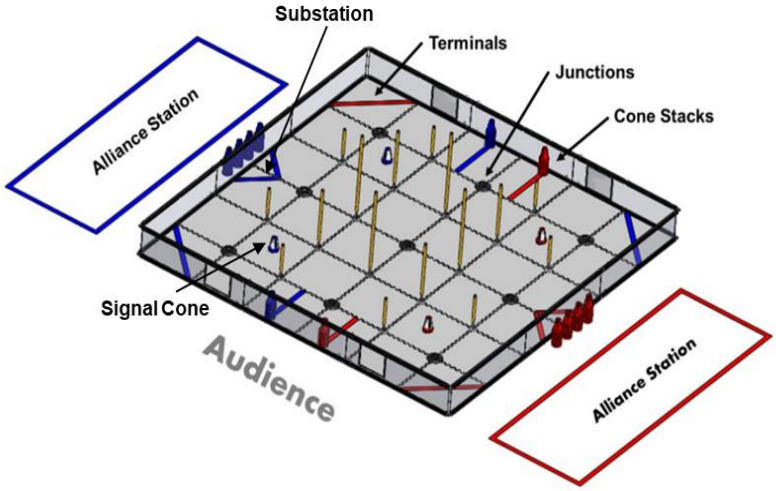


Figure 1. FTC field setup for the 2022-23 FTC season [2]

Figure 1 shows the field setup for the 2022-23 FTC season. Here is the high-level summary of 2022-2023 FTC robot game rules:

- The FTC robot game is played between two opposing alliances, Red alliance and Blue alliance.
- Each alliance is composed of two robots each built by a separate FTC team.
- The FTC robot game is composed of two phases: (1) the autonomous phase and (2) the driver controlled tele-operation phase.
- The referees at first randomize and determine the right robot parking zone # right before the match starts. The signal cone in the field will then be manually rotated so that it has the right picture on the signal cone facing toward the alliance station. Different pictures on the sides of the signal cone will match with the zone # that the Robot should park in at the end of the autonomous period.
- The goal of the game is to score the most points. Points are scored by doing the following:
    - Deliver a cone of your team's color to a junction earns 5, 4, 3, or 2 points for high, medium, low, or ground junctions, respectively.
    - 30 points for parking in the autonomous zone as determined by the signal cone.
- Teams begin in designated Stations and each robot must autonomously attempt to complete scoring tasks in the autonomous phase.

In FTC, the robot motions are controlled by a REV control hub on the robot programmed using Java. The REV control hub sends commands to different motors in order to make the robot move. The control hub also receives input from internal and external sensors. The Android device inside the control hub is programmed from a computer using Android Studio and the FTC software development kit (SDK), which adds functionality to control motors from the REV control hub. A diagram illustrating the interaction among different components of this robot system is shown in Figure 2. Both autonomous and driver control tele-operation programs are downloaded onto the robot, and then are run from the REV driver hub, connected to the control hub via a direct Wi-Fi connection. During the autonomous phase, the robot moves based solely on its autonomous program. During the driver control tele-operation phase, the REV driver hub, connected to two joysticks, transfers joystick input to the REV control hub on the robot. Thus, the drivers of the robot can use their joysticks to drive the robot. Teams use the Android software platform located both on the robot and with the driver to run their programs that control the robot. During the tele-operation period, one human player from each alliance will manually place the cone into the substation in the field one cone at a time. The robot needs to collect these cones on the field and stack them up onto the junction to score points.

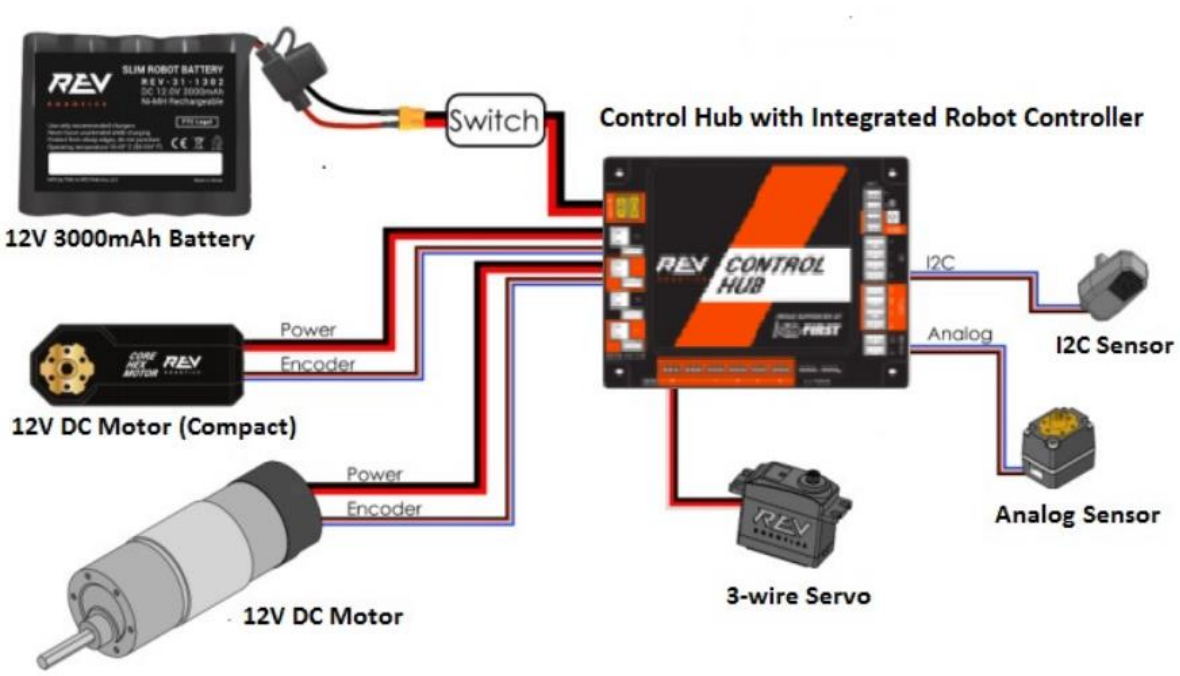


Figure 2. The control system of an FTC robot and its subsystems (e.g., external sensors, motors, power, etc.) to detect and control FTC robot [3]

1.2 Introduction to FTC Software Quality

FTC software quality is crucial due to the high demand of consistent competitive performance during FTC competitions. If the software runs into quality issues, it can significantly impact FTC competition results. The team revamped the software of the robot through advanced development and deployment methodologies leading to increased quality and high scoring capability during competitions for the FTC season.

This paper presents two main challenges in accomplishing high software quality for FTC robotics competitions. The first challenge is ensuring that the software changes can be adapted rapidly with high quality. FTC 2022-2023 season competition was released on September 10th, 2022, and the first competition was on November 12th, 2022. There was only a short two-month period to design, build, test, program and refine a competition-ready robot. To achieve this goal, the team employed the SWOT methodology, leveraged Github software storage management, and utilized CI/CD deployment methods to

meet the tight schedule. The second challenge is ensuring that the robot can utilize machine learning software to minimize the effect of the constantly changing physical world and make the right decision in different situations. The machine learning algorithms must be tested thoroughly to ensure high quality robot performance.

2 Revamp Software Quality in Android Robots

Throughout seven years of FTC experience, the team perfected a process of SWOT analysis and an industry-adapted design cycle [4] to effectively evaluate and optimize proposed solutions using the robot's software and hardware, allowing the team to achieve its maximum capability. When the complex FTC challenge is first presented to the team, each member comes up with an individual design to attempt to solve these main tasks at hand. Each design is evaluated using weighted metrics and SWOT analysis [5], based on criteria that the proposed solution must meet. Weighted metrics is a method of evaluating prototypes based on subjective criteria. Proposed solutions are assigned a numerical value based on this evaluation, and then ranked based on their relative sums. SWOT analysis evaluates the Strengths, Weaknesses, Opportunities, and Threats of a design. Strengths are internal factors that provide certain advantages to the robot's design. Weaknesses are internal factors that may limit certain aspects of the robot's performance. Opportunities are external factors that can be utilized to enhance the robot performance, and Threats are external factors that pose challenges to the capability of the robot. Utilizing weighted metrics with SWOT analysis at the beginning of the FTC season allows for preliminary robot designs to be synthesized. The result is a metaphorical uncut diamond; effective but unoptimized. Throughout the season, the team's continuous iterations with the industry-adapted design cycle allows for the robot to be optimized in order to accomplish the peak performance.

While utilizing the SWOT analysis model, the team employed the industry adapted design cycle consisting of four steps [6] — brainstorming, prototyping, fabricating, and testing (See Figure 3). During the brainstorming phase, the team focuses on creating criterion for the mechanism and laying out the initial thoughts as whiteboard drawings. The team scrutinized current bottlenecks on the robot and conceptualized more optimal solutions. Next, to test the validity of the proposed solutions, the team completed prototyping separately from the existing robot to ensure that a functioning robot is still intact in case the solution fails. The prototyping stage focuses on using PTC Creo and Fusion 360 CAD software to better visualize new or altered mechanisms, translating them into initial prototypes. The visualization then enables the team to fabricate the components on CNC and vertical milling machine tools. During the fabricating step, the team manufactured modules using industry-grade materials and machines with high precision. Lastly, to ensure the effectiveness of the solution, it is extensively tested in a variety of scenarios to ensure its efficacy in competitions. Through this process, SWOT analysis is continually applied to the robot's components to ensure that they are always at optimum performance. Similar to the weighted metrics for robot components, the robot's software and components are also assessed using identified criteria in the evaluation phase. This evaluation provides insights into the current state of the robot and helps identify areas for improvement. Then, based on the evaluation results, optimization techniques are applied to enhance the robot's performance. This evaluation-optimization process aims to maximize strengths, mitigate weaknesses, capitalize on opportunities, and address threats located in the SWOT analysis to revamp software quality.

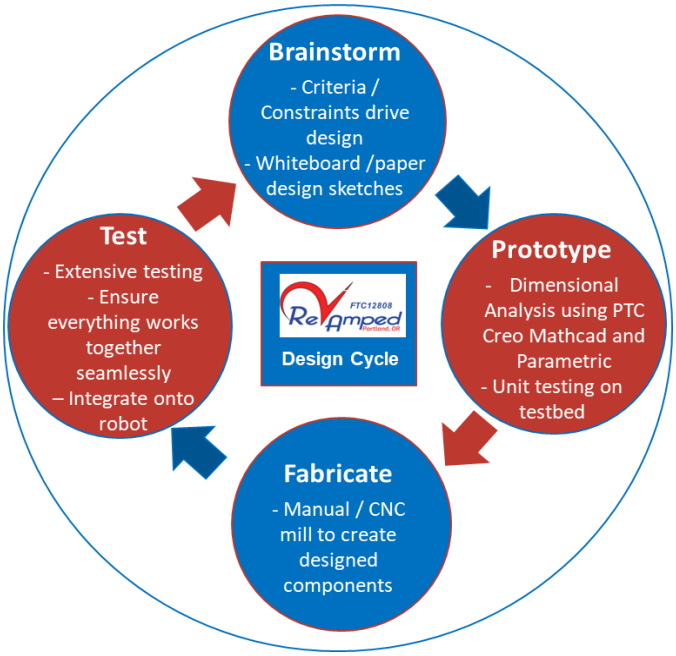


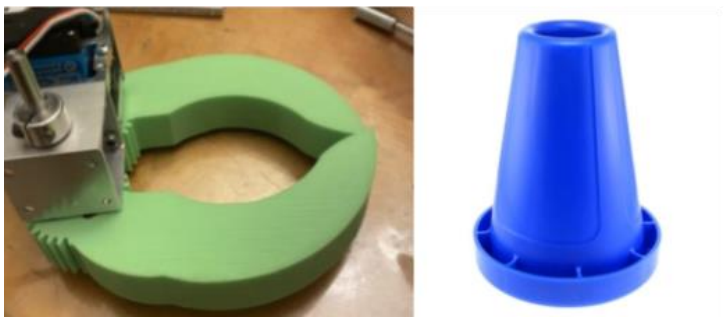
Figure 3. The Revamped industry adapted design cycle

3 Case Study: Advanced Development and Deployment Methodologies to Revamp Software Quality

The Revamped Design Cycle allows the team to make effective changes to the robot software and hardware in a fast-paced competition environment, allowing for on-field testing at the competition to ensure the high quality. After competition matches, the team further iterates upon the robot using this same cycle to produce high quality robot components while ensuring it fits with overall robot mechanisms.

3.1 Case Study: Robot Claw Optimization to Interact w/ External Physical World

During the 2022-2023 season, one critical robot component was the design of the robot claw. The claw was integral to the robot's scoring capabilities as it was the component that directly interacted with the external physical world to reliably grasp a cone and drop it off onto the junction, Thus the claw came under heavy scrutiny as the team continued to iterate and optimize the robot. At the beginning, the robot claw was a simple two-pronged grasping mechanism (Figure 4a) barely capable of retaining the scoring element, a circular plastic cone (Figure 4b).



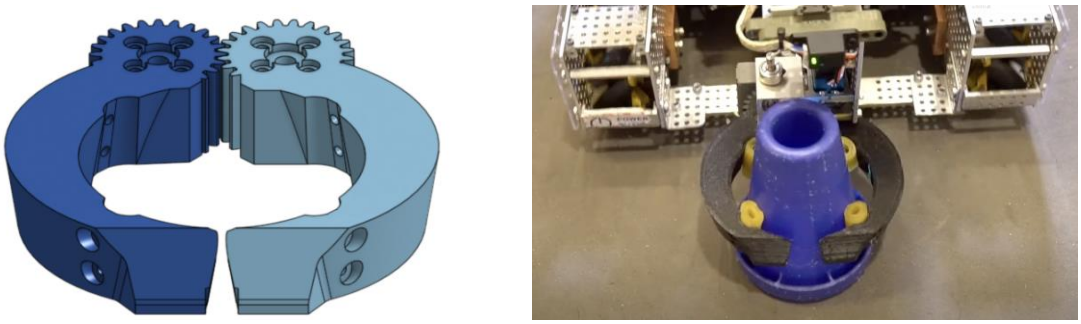
(a) Robot claw design #1 (b) Cone

Figure 4. (a) The team's preliminary claw design #1 and (b) the aforementioned cone

The ability for the robot claw to hold a cone securely for a long time was very crucial to the robot's performance. After the evaluation-optimization process, the team realized the need for a friction generator lining the inside of the claw to create sufficient friction with the cone which would allow for an easier retention of it. Design #2 quickly emerged with more force to hold the cone (Figure 5a). Thus, the team began to use SWOT analysis and weighted metrics to evaluate each design to ascertain which design would hold more promise. The criteria that each design was evaluated on was simplicity, effectiveness, and size, as well as ease of maintenance and replacement for the claw design option. The weight value is determined by the magnitude of the impact and the probability of occurrences for these SWOT analysis line items in Strengths, Weaknesses, Opportunities and Threats. Eventually, the team selected the second design based on SWOT analysis results (see Table 1 for SWOT analysis process data). This design still had flaws and need to improve to securely hold the cone. The team iterated on the claw design to finally arrive at the current design (see Figure 5a and Figure 5b). Figure 6 shows that the claw mechanism is a critical part of the robot during FTC competitions.

Table 1: SWOT Analysis for the New Claw Design (#2) with Weighted Metrics

Strengths		Weight	Weaknesses		Weight
	Software can control effectively to grip, hold and release the cone	0.25		Securely hold cone during the robot movement in matches	0.2
	Simple claw design and easy for the software to control	0.1		More possible autonomous position control for the robot claw	0.1
	Right size to fit into the overall robot	0.1		Extra steps & complexity to build the claw	0.05
Opportunities		Weight	Threats		Weight
	Easy attachment of sensors to improve the accuracy of the claw movement	0.3		Need to adjust the claw shape for more effective software control when hit by another robot	0.15
	Adjust the physical connection with the robot to simplify install	0.1		Need to change the software to cover from sensor failures	0.15



(a) The CAD of the claw design #2 (b) The current claw

Figure 5. The CAD of the claw and the current claw gripping a cone

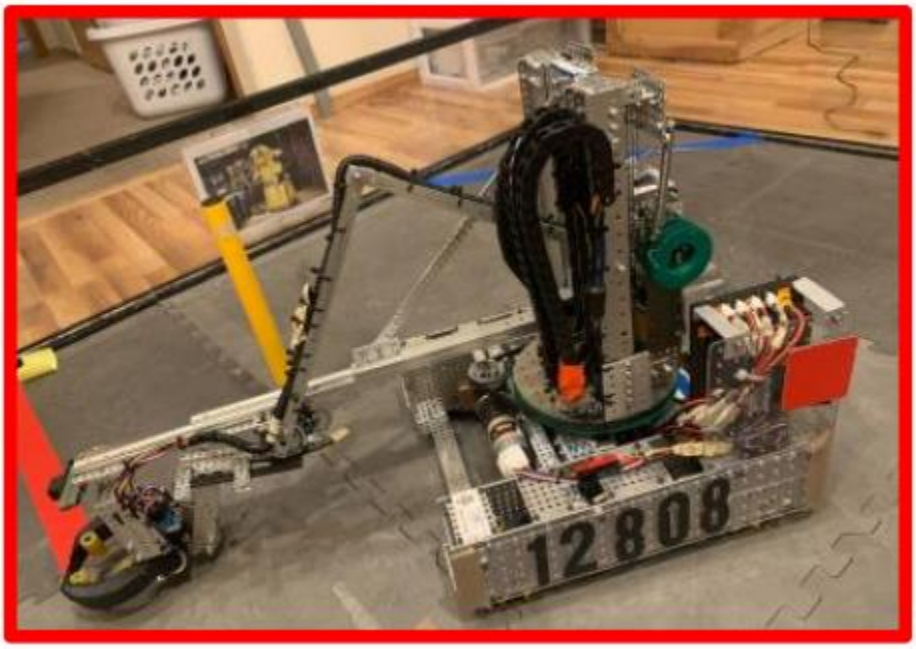
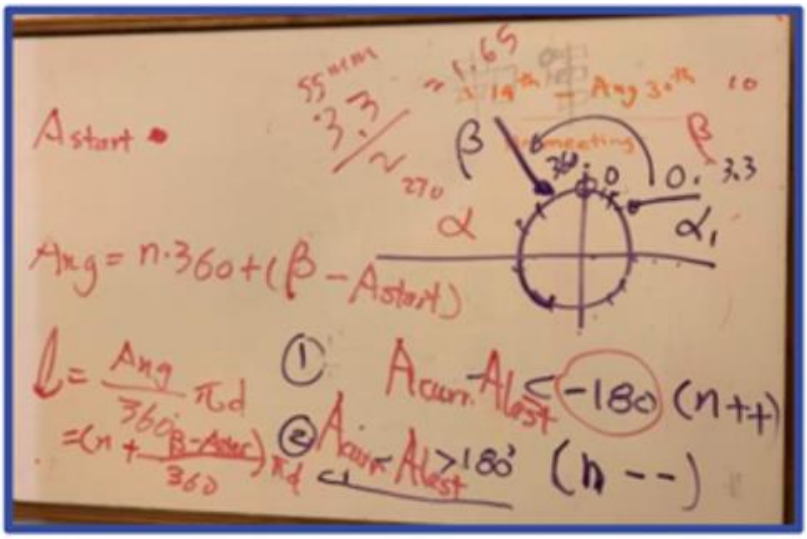


Figure 6. The Revamped Robotics FTC competition robot for FTC 2022-2023 season

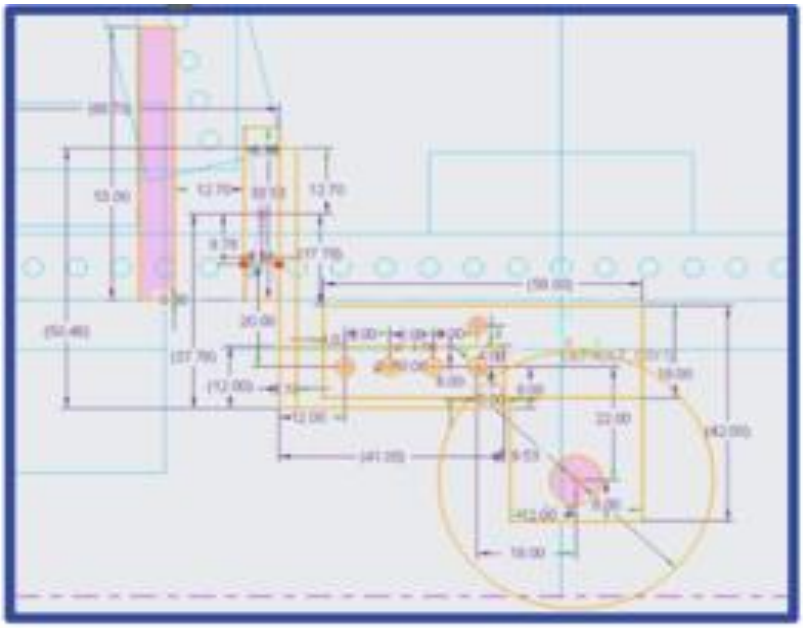
3.2 Case Study: FTC Robot Odometry Pod System Software Quality Improvement

Another case study can be constructed from the team's work on the odometry pod system, as it includes both software and hardware synthesis. The team at first identified the problem where the mecanum drive needed for accurate pathfinding in the field had reduced friction and higher drive error than was acceptable. Utilizing an external measurement of the robot position allows the robot to move with accuracy in the autonomous period so that the team can complete all autonomous goals with consistency. The team brainstormed spring-loaded carriages for constant contacts with the field and centered pods along the major axes of the robot for the highest accuracy. The system should be within 2% error on the x and y axes and must allow the sensor fusion with the external Nav X IMU for constant position tracking regardless of the robot heading direction.

The team then started the iteration of development and fabrication processes. The robot movement path and the corresponding robot wheel movement model were carefully calculated (See Figure 7a) to determine the distance and direction by converting encoder shaft outputs to the arc length traveled by a wheel. The team then CAD'ed the design (see Figure 7b) and used MA3 absolute encoders, iterating from a 1:1 to a 3:1 gear ratio and the lateral spring tensioning to prevent the revolution skipping by maintaining constant contacts with the field. At this point, the team also switched to 3 encoders. However, the testing revealed inaccuracies and complexities in fine-tuning and programming the absolute encoders, so the team switched to US Digital E4T encoders. While absolute encoders require a function to constantly convert the voltage into the distance, only a simple linear regression is needed to program the E4T encoders. Specifically, three E4T pods (2 in the y-axis and 1 in the x-axis) were used for a better spring tensioning and a more compact design. Gearing down is unnecessary because the encoder measures at ticks, resulting in an ultra-compact design. The team also switched the encoder wheel to smaller and thicker plastic wheels to address movement inconsistency issues (Figure 7c and 7d).

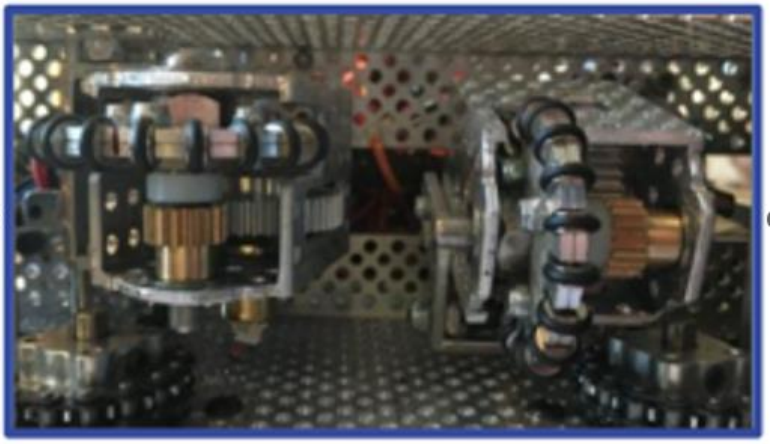


(a) Brainstorming on white board



(b) CAD Design

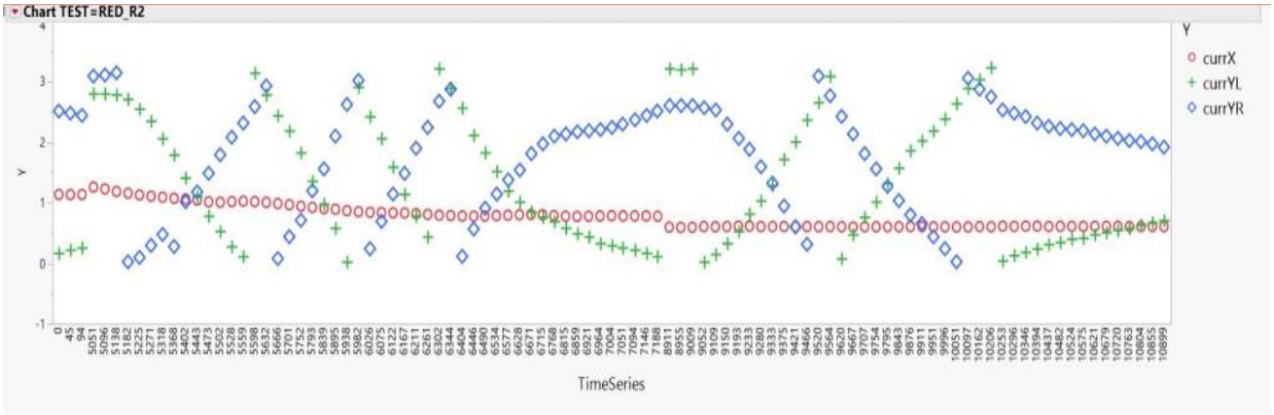
(c)Prototyping



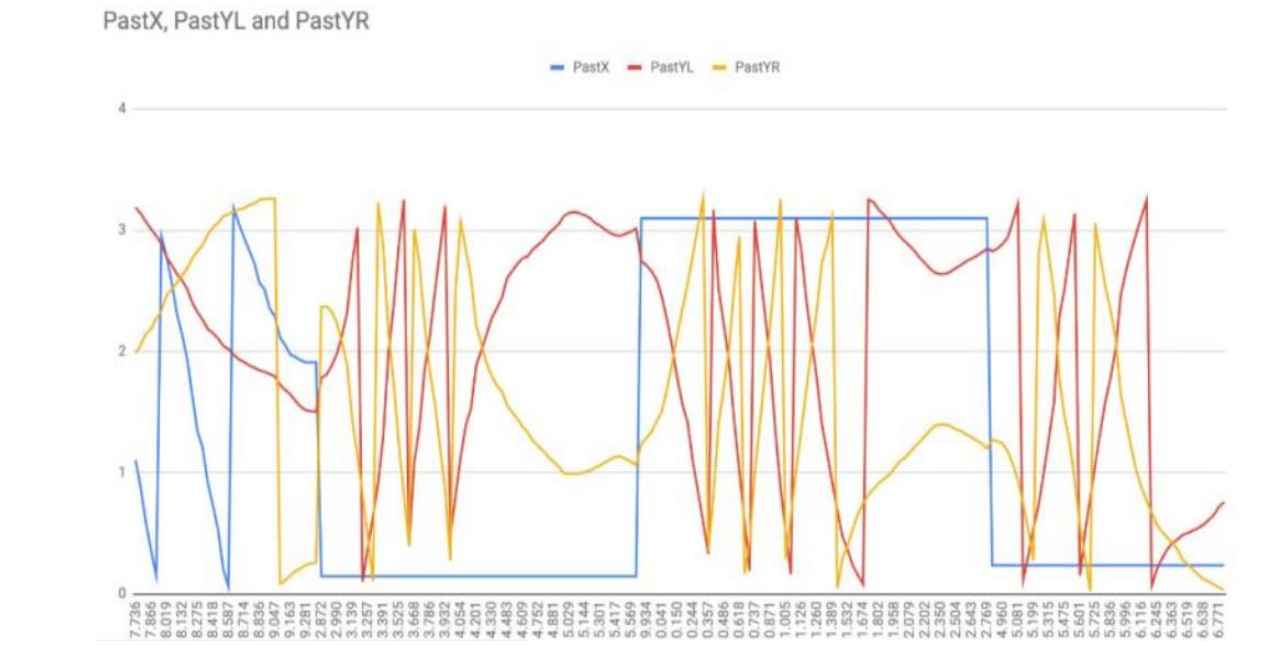
(d) Fabrication

Figure 7. Odometry (a) Brainstorming (b) CAD Design (c) Prototyping and (d) Fabrication phases

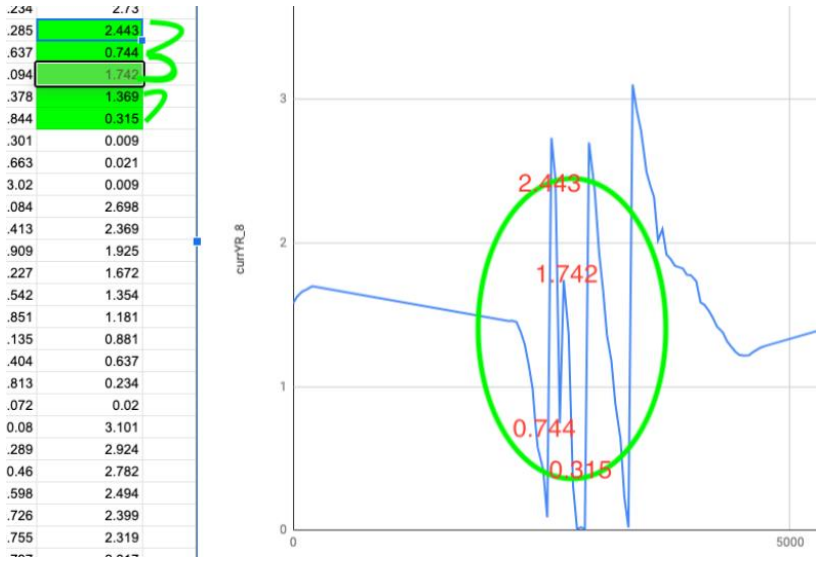
The team did thorough testing and collected data on voltage jumps to expose errors in the autonomous phase and made debugging very efficient. The data gave the team reasons to switch from absolute MA3 to E4T encoders. To visualize errors, the team logged the reading and then parsed certain values of interest to show normal trends and discrepancies (Figure 8a) where CurrX indicated the robot side move and CurrYL/ CurrYR indicated the reading on the left and the right hands side along the robot heading direction. The CurrYL and CurrYR in Figure 8a showed jumps in the reading which indicates the discrepancy issue for the robot movement. Figure 8b shows a smooth auto run with no sudden jumps for the robot side move (X) and the robot heading moving direction (YL and YR) readings, as compared to the real encoder with voltage jumps shown in Figure 8c due to the real field friction variation, etc. The example showcased the use of the Revamped Design Cycle promotes quick and iterative development process for the software quality improvement.



(a) Odometry testing data trends and discrepancies



(b) Odometry testing smooth autonomous run despite voltage jumps



(c) Odometry testing encoder voltage jumps

Figure 8. FTC robot odometry testing results

3.3 TensorFlow Machine Learning for Autonomous Phases

The team has the autonomous software program which can pick up and drop off cones to the high junctions and then park at the right zone. The diagram below shows the path that the robot takes, which consistently accomplishes high scores during the autonomous period.

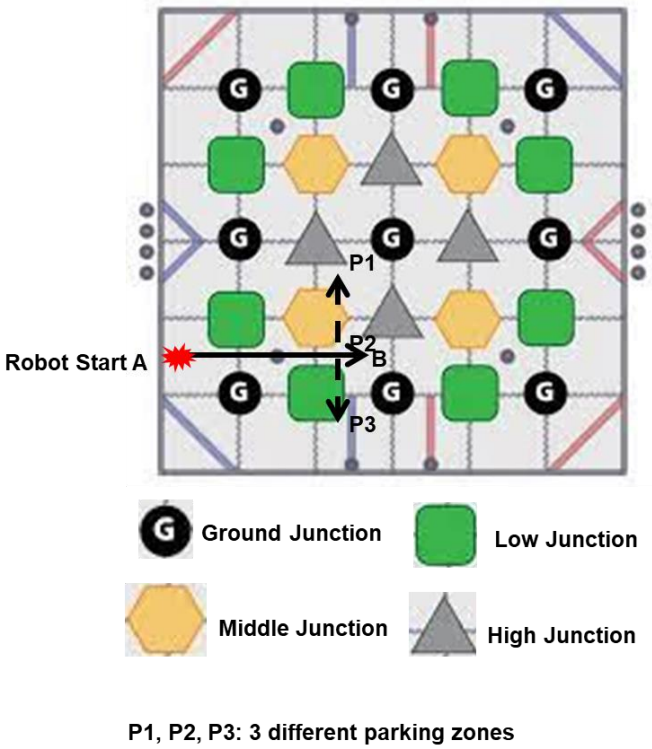


Figure 9. Illustration of autonomous robot movements

Table 2: FTC Challenge Autonomous Robot Action Sequence Example

Step	Description
1	Robot starts from 1 of 4 corner position in the field
2	Camera captures the picture on the signal cone and the robot program analyzes it
3	Determines the final stop zone using TensorFlow
4	Moves to the position near the high junction (pole)
5	Detects and picks up the cone from the pile next to the wall, and then drop off to the high junction
6	Repeat Step 5 to pick more cones with the adjusted height and drop off the cone to the high junction
7	Adjust and then move to left or right to the right parking zone determined by Step 3

One important part of the autonomous objectives is to identify the zone that the robot should park in at the end of the autonomous period accurately. Last season, the team used OpenCV to detect the team element which is a key aspect of the autonomous period. While it is simple to use, the team found that it was less reliable under different lighting conditions. This year, the team adopted the TensorFlow AI model provided by FTC and trained it on customized signal cone patterns. The TensorFlow Machine Learning process [7] utilizes image recognition (Vuforia). After the autonomous phase starts, the robot camera activates Vuforia, allowing the TensorFlow image recognition to initialize and scan the team sleeve picture on the signal cone. The Machine Learning system has been trained to recognize the pattern on the signal cone by being shown numerous pictures of this object from multiple angles, allowing for a complete perspective to be produced. Once objects are visually scanned, it compares previously learned pictures to the real-life situation. At run time, the USB web-camera on the robot captures a real-time video stream which is fed into the TensorFlow inference model to detect the randomized signal. To increase reliability, the video screen is cropped to minimize the background noise. This setup can reliably detect all three patterns, even in low lighting conditions and with a noisy background. Additionally, every iteration of the scanned objects is added to a cloud recognition database to enhance future accuracy of scanning. In some scenarios, some pictures are related to the object with different conditions, so the TensorFlow can recognize the pattern with only a part of the picture. Based on previous testing, the team was able to narrow down the expected pattern and determine the right zone to land the robot reliably.

Figure 10. Illustration of signal cone sleeve pattern detection using TensorFlow machine learning

A high scoring autonomous robot needs robust sensors and a good architecture for easy debugging and readability. To complete every autonomous task, The team structures the code following a trajectory and movement sequence that allows the team to orchestrate complex behaviors. The software runs multiple tasks concurrently allowing the team to plan more time efficient paths and dynamically generate movements

based on the robot status. For instance, if the distance and color sensors detect a junction or a stack of cones in the proximity, the software can control the turret and slides on the fly to grab a cone and score it on a junction. This prevents the autonomous failure given the trajectories are based on the sensor inputs rather on pre-programmed instructions. With the advanced troubleshooting practice and methodology, the robot can detect the right pattern accurately (See Figure 11)

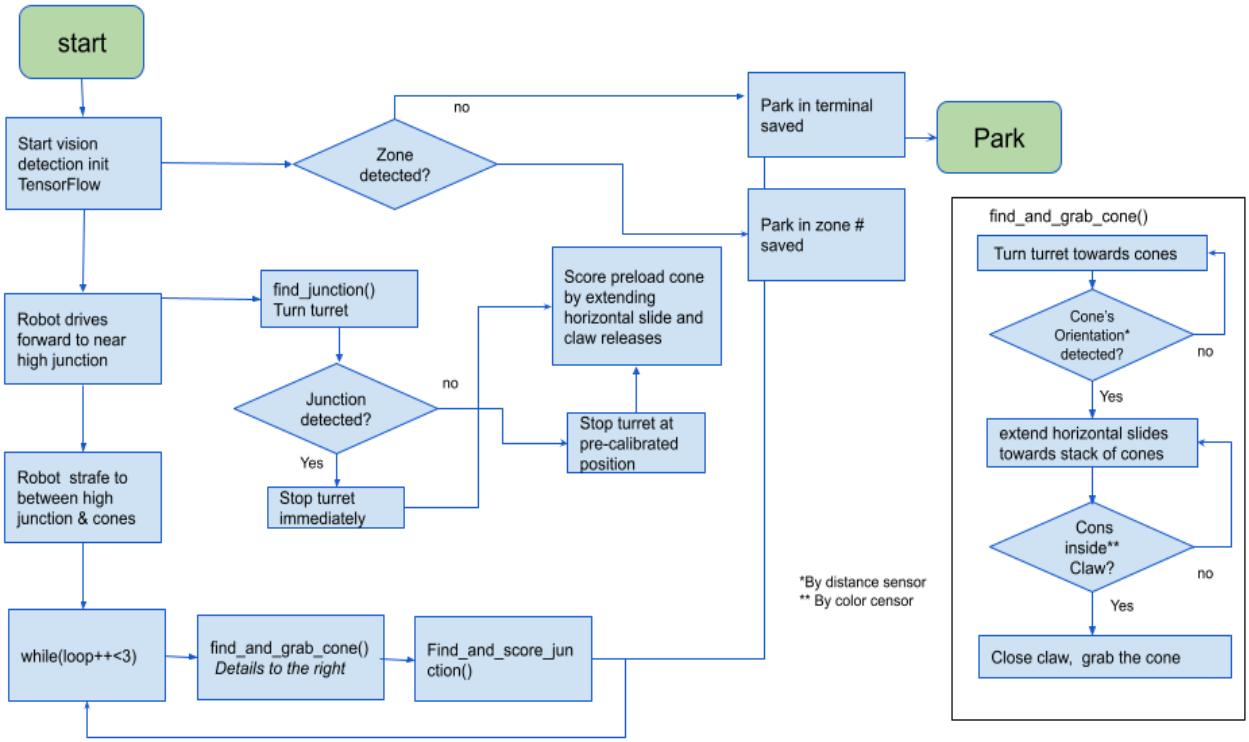


Figure 11. Autonomous program flowchart

3.4 Github and CI/CD Software Deployment Methodology to Accelerate Software Quality Improvement Process

The team has been using GitHub as a web-based version control and collaboration platform for the software development. Given the agile software development cycle in the team, Github allows team members to collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple team members. The team changes, adapts and improves software from its public repositories. The team grouped the code into “Autonomous”, “TeleOp”, “test”, and “utils” code repositories correspondingly. In addition, the team also maintained selected component codes that were tested from earlier seasons and can be reused for this season (e.g., roadrunner, etc.)

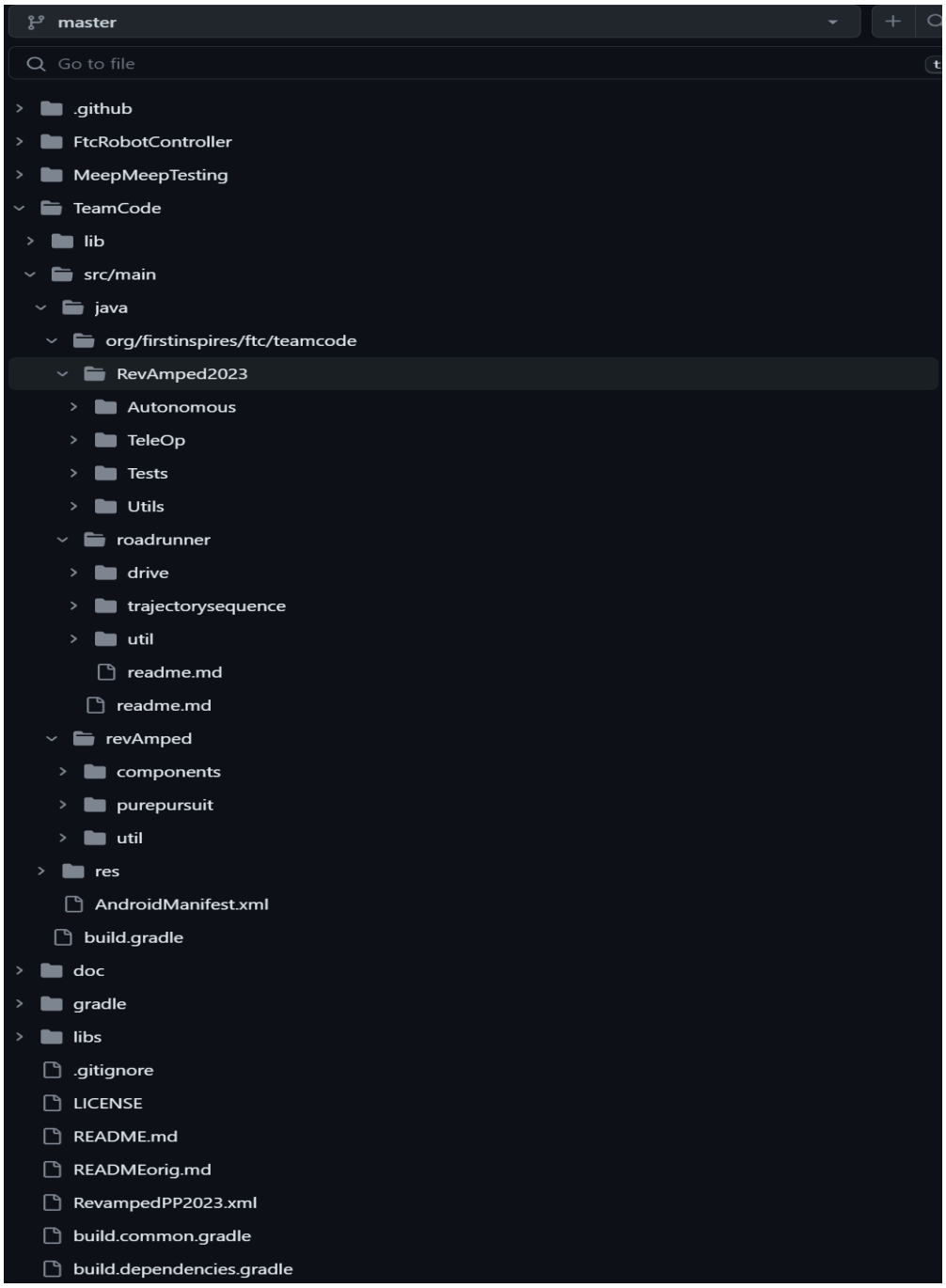


Figure 12. Revamp Robotics team GitHub code storage and management model screen shot

The team employed the Github technology below to set up different branches for different developers to ensure developers make modifications without affecting the original master code. After reviewing modifications, the original owner would “Add” the modifications into the repository on a very frequent basis. The team uses “Commit” to commit an individual change where commits are retained and interleaved onto the main project appropriately. The team member can accept the modifications and merge them with the master repository, enabling a flexible and fast revision process that spans the entire FTC season. Furthermore, the team uses “Push” to send code from a local copy to the online repository.

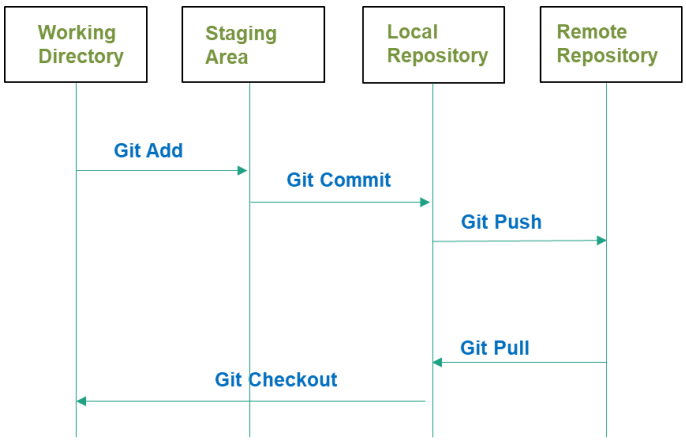


Figure 13. Flowchart for Github code storage and management

Through this journey to revamp the software quality, the team members were able to follow each other, receive updates for specific open source projects (e.g., autonomous project, etc.), and communicate learnings effectively. Github enables the team to work effectively when different team members work on the code at different times with the flawless transfer of the developed code.

The team also utilized GitHub software storage management to support integrated development environments and continuous integrated development (CI/CD) with more security and auditability [8]. By automating CI/CD throughout the entire software development lifecycle in FTC season, the team was able to develop higher quality code at a faster pace. CI/CD brought different team members together to deploy software to the robot promptly as soon as the software code change was completed and ready. As the speed of the software application to the Robot becomes a key to how FTC teams differentiate, the rate at which the code can be released has become a competitive differentiator. Revamped Robotics team was able to accomplish ~15-33% new feature implementation acceleration after employing Github software storage management, and CI/CD development and deployment methodologies (See Table 3).

Table 3 Github and CI/CD Software Development and Deployment Acceleration

Categories	Small Size Feature (Weeks)	Medium Size Feature (Weeks)	Large Size Feature (Weeks)
Legacy Deployment	0.3	1	2
Github & CI/CD	0.2	0.8	1.7
Acceleration	33%	20%	15%

4 Summary

This paper presented Revamped Robotic team's approaches in improving software quality in Android robots that integrate robot software and physical worlds together. Case studies were shared in SWOT analysis, software deep learning and Github/CI/CD in Android robots to revamp software quality and accomplish the top robot performance in FTC. The SWOT analysis is used to effectively design, prototype, fabricate and test solutions. Furthermore, the paper presented the method to improve the software quality through deep learning TensorFlow to minimize dependencies on the external physical sensor reading and the mechanism to mount the sensor. In addition, Github storage management and CI/CD process showed ~15-33% FTC feature implementation acceleration. All of these methods can be utilized in industries to improve software quality at an accelerated pace when handling physical world uncertainties and complicities.

Acknowledgments

Authors would like to thank paper reviewers, Cameron Kilgore and Nick Bonnichsen, for their invaluable assistance and discussions on software quality. Furthermore, authors would like to thank the Ramped Robotics team coach Zhunquin Wang for his excellent guidance throughout our multiple year FTC journey. In addition, authors would like to thank Oregon Robotics Tournament & Outreach Program (ORTOP) whose planning and assistance are very crucial for the success of FTC events in Oregon.

References

[1] http://www.usfirst.org/ (accessed May 31, 2023).
[2] https://firstinspiresst01.blob.core.windows.net/first-energize-ftc/game-manual-part-2-traditional.pdf (accessed May 31, 2023).
[3] https://github.com/ftctechnh/ftc_app/wiki/The-FTC-Control-System (accessed July 23, 2023).
[4] https://www.aha.io/roadmapping/guide/it-templates/swot (accessed May 31, 2023).
[5] Escalona, M.J., Nora Koch and Gustavo Rossi. 2022. “A Quantitative SWOT-TOWS Analysis for the Adoption of Model-Based Software Engineering, Journal of Object Technology (January 2022). 21(4):4:1.
[6] https://mitsloan.mit.edu/ideas-made-to-matter/design-thinking-explained (accessed May 31, 2023).
[7] https://www.tensorflow.org/tutorials (accessed May 31, 2023).
[8] https://resources.github.com/ci-cd/ (accessed May 31, 2023).