



FIRST
TECH
CHALLENGE

Control Award Content Sheet

Team # 9789

Team Name: TOXIC

Autonomous Objectives

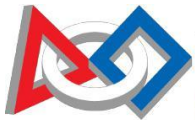
Overview - Our team has designed and implemented a total of 8 unique and highly reliable autonomous programs and strategies, each configured to maximize scoring in any given scenario in the match setting. This autonomous flexibility has yielded efficient scoring when paired with robots that present different strengths. In this content sheet, we will thoroughly highlight each of the sensors that we use, along with the performance goals and objectives associated with our most advanced 100 point program.

Performance Goals - Score 2 Particles into the Center Vortex (30 points), claim both Beacons (60 points), partial park on the Center Vortex Base (5 points), and knock the Cap Ball onto the Playing Field (5 points), totalling to 100 points.

Chronological Objective Routine

- 1.) Utilizing encoder technology, drive straight and stop at the point in which the robot shoots both Particles into the Center Vortex.
- 2.) Swiftly turn to the appropriate angle for shooting, measuring encoder counts, and then call forth our gyro sensor algorithm to ensure that this optimal alignment is accurate.
- 3.) Shoot both Particles into the Center Vortex, harnessing encoder capabilities to control the shot cycle.
- 4.) Exploiting the function of our encoder algorithm, drive toward our respective alliance's wall.
- 5.) Turn parallel with the white tape in front of the first beacon, using the motor encoders, then employ the output of the gyro sensor as a safety check to ensure that the robot heading is parallel with the wall.
- 6.) Square up on the white tape in front of the beacon using both color sensors embedded in the drivetrain, approaching such with our innovative square up algorithm and PID implementation.
- 7.) Drive toward the beacon using the optical distance sensors that command the execution of an inequality driven while statement, in which we approach the beacon until we are a set distance away.
- 8.) Utilize a color sensor to correctly read the value emitted by the beacon, which sets a boolean value in our color analysis algorithm, which determines the position of the servo to trigger the buttons.
- 9.) Return the robot to a heading that is parallel with the perimeter wall by executing our efficient gyro sensor control algorithm.
- 10.) Drive straight and approach the next beacon while integrating the heading value of the gyro sensor as a variable input to output a speed value with each of the motors to maintain a straight path.
- 11.) Repeat steps 6-8 in an effort to claim the second Beacon in a highly reliable manner.
- 12.) Pivot, drive, and park on the Center Vortex Base while knocking off the Cap Ball using the motor encoder function.

Particle Prevent Mechanism - One of the most innovative aspects of our autonomous mode is the modular servo sub-assembly that we have integrated into the chassis of our robot that extends a Lexan form outward. We use this in conjunction with the gyro sensor as the robot traverses alongside the wall, preventing particles from affecting the functionality of our robot when triggering the beacons.



**FIRST
TECH
CHALLENGE**

Control Award Content Sheet

Sensors Used

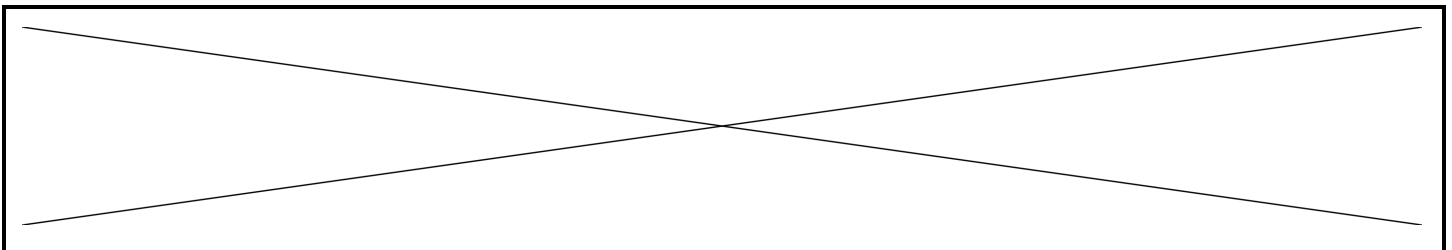
Overview - We have a total of 7 sensors on our robot, each used to enhance the reliability of each of our autonomous programs. By incorporating the logical and efficient function of their associated Java algorithms that we have designed throughout the code, we have been able to produce autonomous routines that are highly consistent, logically using some of the most effective sensor based methods we created to optimize point output. This allowed us to build a high scoring and fluid approach to the game.

Gyro Sensor - We use the gyro sensor to control and ensure the accurate turning of the robot by computing and analyzing the degree heading in which it is facing. We have integrated many gyro derived functions in our code that are used for several different scenarios, including inequality based turns in which the motors move until we reach a certain point, and also where the robot maintains a perfectly straight heading no matter the starting angle where this function initiates. Designing and implementing an efficient alignment method in many spots throughout the code has acted as another layer of security, with this “software insurance” serving as a vital aspect to our high level of consistency after encoder methods.

4 Color Sensors - Our team has two color sensors that are used to read the value emitted by the beacon, one for each alliance. We have another two color sensors that are in perfect alignment with one another on the underside of the chassis. These square up on the white tape so that we approach the beacon straight on, increasing our factor of reliability in triggering the correct color.

2 Optical Distance Sensors - Implementing two distance sensors on our robot has also served as an integral component in our consistent autonomous programs. These distance sensors are used to drive toward the beacon until it is a certain amount of centimeters away. This contributes to the reliability of our programs, as our robot will be ‘intelligent’ enough to track the location of the beacon, even if the walls have been moved during match play or other slight field variations come into consideration.

Motor Encoders - The encoders that we have integrated with 7 of the 8 motors on our robot have both autonomous and driver controlled applications. Each motor on the robot has cable linkage, which allows us to track the position of the motors in autonomous, guiding the starting and stopping points for many of the movements that the robot takes. Additionally, during Teleop, we are able to make the gearbox motors for our linear lift synchronous with one another by taking advantage of the on-board PID function. Throughout the entirety of the match, we strive to maximize the functionality of the encoders by transferring the voltage readings from the battery into a direct speed quantity for the motors, instead of a power value. What this does is prevents differences in motor velocity based on battery power. Additionally, our team analyzes encoder feedback to help troubleshoot and problem solve through our logical telemetry output and evaluation process.





FIRST
TECH
CHALLENGE

Control Award Content Sheet

Key Algorithms

Overview - Designing and programming several creative and efficient algorithms, based on the assortment of sensors that our robot is equipped with, has contributed to our team constructing an innovative autonomous program that works in a highly reliable manner. Below, we will highlight our most advanced and well developed software techniques and algorithms, with references to the context of these methods in Java, found within our pasted code entry. Everything done with our programming is thoroughly commented and annotated for logic clarity and personal troubleshooting purposes.

[Refer to the '*LineSquare()*' method in our Java code.]

Dual Color Sensor Square Up Algorithm - Our team has implemented this square up method into our code which is used to square up on the white tape so that we can approach the beacon while being in direct alignment with the buttons, which contributes to the reliable success of triggering the correct color. The threshold in which the sensor takes a reading of the white line is the value that is responsible for the command of this while loop inequality statement. This loop will continue executing until both sensors have read the line, with each side of the drivetrain associated with the sensors incorporated into the underside of the chassis on the robot. The unique part about this function is that either sensor may read the white line first, allowing for a slight amount of variance when approaching the white, improving consistency.

[Refer to the '*GyroAlign()*' method in our Java code.]

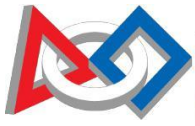
Gyro Alignment Algorithm - The majority of our gyro sensors functions are within a constant loop that reviews event-driven boolean variables constantly, with equality and inequality based conditions that associate with the power of the motors on the drivetrain. No matter the direction in which the robot is facing, a function will be called to return the robot to a heading within the threshold that we have declared. These specific quantities are visible in the parameters for the while loop that our team has set forth.

[Refer to the '*LongMovementStraight()*' & '*LongMovementStraightGyro()*' method in our Java code.]

Gyro Algorithm for Maintaining Trajectory - In this algorithm, we have programmed the robot to drive in a straight path based on the values yielded by our Modern Robotics Integrated Gyro Sensor. If the heading of this sensor value is less than our desired angle, then the right motors on the drivetrain are running at a higher velocity to compensate for this drift. Rather, if the sensor value is greater than our desired angle, then the left motors on the drivetrain run at a higher speed. Integrating this intelligent control into the functionality of our robot contributes to more consistent results in the match setting, as our robot will move in a perfectly straight path between both beacons, regardless of minor external factors.

[Refer to the '*StoreColorValue()*' & '*EngageBeacon()*' method in Java.]

Color Analysis Algorithm - In order to trigger the correct button on the beacons, we have a color sensor that directs the accomplishment of an 'if then' statement based on the values that the beacon emits. If the hue that the sensor reads is of the red threshold, then the server initiates the opposing side of our trigger mechanism, which activates the blue button. Likewise, the vice versa occurs if the former scenario is detected, but in reverse. This is accomplished through the usage of several different boolean variables applied to inequality logic. Staying simplistic with the usage of one color sensor for each side of the robot has directly correlated to a more efficient and "simple" program for the robot to execute in this high point density component of the game, a logical approach to this challenge.



FIRST
TECH
CHALLENGE

Control Award Content Sheet

[Refer to the *'DriveTowardBeacon()'* method in Java.]

Optical Distance Sensor Algorithm - This software process functions based on the drive system traversing sideways toward the beacon until a certain distance value is picked up from this sensor. Fitting this command into the while loop has ensured that our robot will be at the optimum distance away for both evaluating the color of the beacons, along with triggering such based on that corresponding decision.

[Refer to the *'FirstMovementStraight()'* method in Java.]

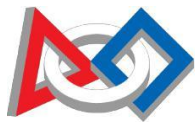
Motor Encoder Algorithms - In each movement for autonomous that does not directly rely on the information received from one of the core device sensors, we use encoders for position tracking. These function based on resetting the mode of the motors and calibrating this hardware, while then proceeding to set the position and run to achieve this target at a set power that we declare. Additionally, we have integrated a series of inequality statements to ensure that all motors stop when at least one reaches its target position that is set with a tolerance threshold, preventing motor stalling or twitching. Using encoders whenever possible has made our robot "smarter", along with being better equipped to understand some drive tendencies and other features utilizing telemetry feedback data.

Driver Controlled Enhancements

Innovative Arcade & Mecanum Drive - We have produced code to complement the innovative functionality of our mecanum based drivetrain. Since this is not a traditional tank drive system, it took a creative mindset and a detailed planning process to create the most efficient control possible. In order to maximize the maneuverable nature provided by the mecanum wheels, our team developed variables that are associated with each of the set axes on the joystick. Then, we add each of the axes' vectors together to create an ability to move in any direction at any angle, with the values of these floats, based on the plane of the joystick, being what we then set the power to on each motor. Additionally, we have enabled the on-board PID function of each motor by taking advantage of encoder technology, converting a power percentage to a fraction of speed, making driving more efficient and consistent for the driver.

Linear Lift | Motor Synchronization - Paralleling to what we were able to implement into the code for the drivetrain, the linear lift has two motors that we were able to make synchronous with one another by using encoders to enhance the overall functionality of the robot. In order to yield the best performance by doubling the total stall torque output of the gearbox, these need to be rotating at the same velocity, and using encoders and software practices lets us achieve such as we practice sound design principles.

Logical & Efficient Gamepad Layout / Telemetry - We have strategically mapped out and placed the controls for each of the sub-assemblies on our robot between the two gamepads so that we can maximize driver efficiency and synergy. Our driver controlled code includes clauses that gives our mechanisms actions based on our controller input using 'if then' statements and 'else if' extensions, maximizing the functionality of our robot. Keeping driving intuitive and relatively simplistic has been an integral part for us in continuing to refine and perfect our abilities as drivers. Additionally, we have incorporated telemetry feedback in key areas within the code so that we can problem solve through any potential concerns in a tactical manner, following the software design and development process with a professional integrity. Sensors that we evaluate through this control method include each of the color sensors, distance sensors, the gyro sensors, and the encoder counts associated with the gearbox.



FIRST
TECH
CHALLENGE

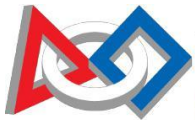
Control Award Content Sheet

Drivetrain | End Game Speed Shift - Another driver controlled enhancement that we have implemented is for when our robot has possession of the Cap Ball. When trying to maneuver our robot around the field with the ball in the air when the lift is in its extended position, we must slow down so that we do not lose possession of such. This allows use to make more precise movements on the field, something that serves as a huge advantage when attempting to maneuver around other robots. We have accomplished this in programming by creating an 'if then' statement where if the right trigger is held down and returning a value of true, then all movements with the drivetrain are scaled down to a value of 20%. This has greatly increased our consistency in placing the ball on the center vortex during this period of the competition. Then, once we need to speed up again and claim beacons before the match is over after we cap, our team will have time to accomplish such. Taking advantage of this unique, strategy based approach in developing our software helped contribute to our team capping the ball a high percentage of the time.

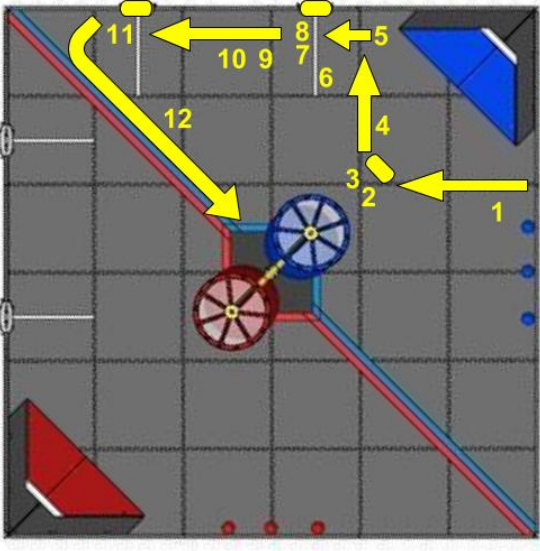
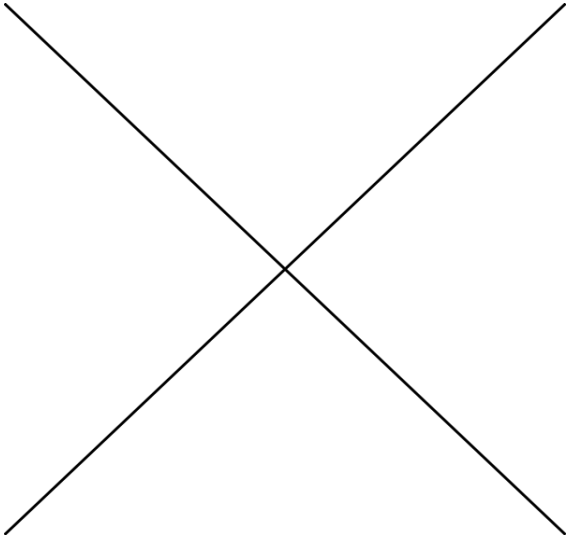
Engineering Notebook References

Overview - One of our team's most prideful values is the importance of highly detailed, thorough, and organized documentation. As a result, we have developed a programming section to thoughtfully log the design and development of the control of our robot. Below, we will provide linkages to some of our most significant entries that showcases the evolution of the software that we have created.

Feature & Entry	Notebook Page
Programming Section Highlights	Section D
Entry 06 FIRST World Championship - Programming Overview <i>*[Pasted Code]*</i>	D33
Entry 01 Mecanum Drive Code Production	D1
Entry 02 Finalized Mecanum Drive Code	D7
Entry 03 Drivetrain Code Edit & Encoder Count Control Test	D11
Entry 04 Initial Color Beacon Code Development	D19
Entry 05 Dual Color Sensor Square Up - Control Design & Implementation	D23
Preseason Section Highlights	Section A
Entry 30 Encoder Troubleshooting & Color Sensor Introduction	A227
Entry 31 Color Sensor Troubleshooting & State Machine Introduction	A237
Engineering Section Highlight	Section B
Entry 53 Autonomous Improvements & Particle Prevent Mechanism - Optimization	B485



Autonomous Program Diagrams

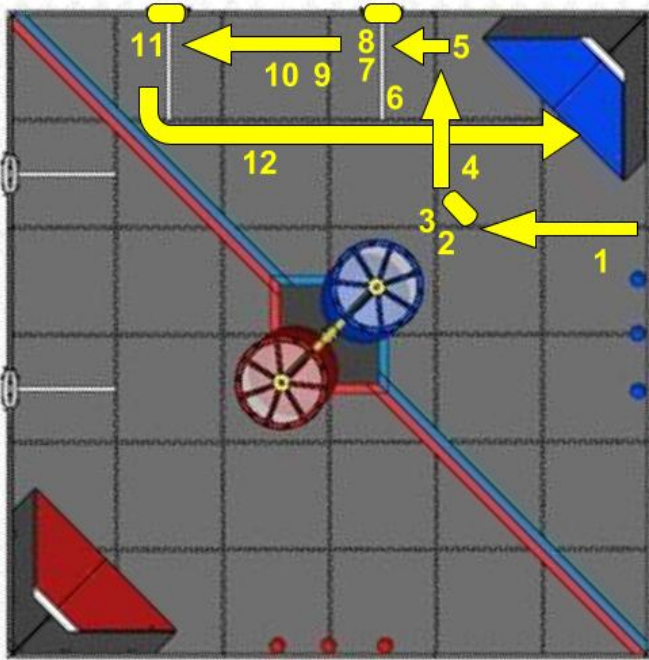
Diagram	Parallelism to Autonomous Objectives
	<ol style="list-style-type: none"> 1.) Utilizing encoder technology, drive straight and stop at the point in which the robot shoots both Particles into the Center Vortex. 2.) Swiftly turn to the appropriate angle for shooting, measuring encoder counts, and then call forth our gyro sensor algorithm to ensure that this optimal alignment is accurate. 3.) Shoot both Particles into the Center Vortex, harnessing encoder capabilities to control the shot cycle. 4.) Exploiting the function of our encoder algorithm, drive toward our respective alliance's wall. 5.) Turn parallel with the white tape in front of the first beacon, using the motor encoders, then employ the output of the gyro sensor as a safety check to ensure that the robot heading is parallel with the wall. 6.) Square up on the white tape in front of the beacon using both color sensors embedded in the drivetrain, approaching such with our innovative square up algorithm and PID implementation. 7.) Drive toward the beacon using the optical distance sensors that command the execution of an inequality driven while statement, in which we approach the beacon until we are a set distance away. 8.) Utilize a color sensor to correctly read the value emitted by the beacon, which sets a boolean value in our color analysis algorithm, which determines the position of the servo to trigger the buttons. 9.) Return the robot to a heading that is parallel with the perimeter wall by executing our efficient gyro sensor control algorithm. 10.) Drive straight and approach the next beacon while integrating the heading value of the gyro sensor as a variable input to output a speed value with each of the motors to maintain a straight path. 11.) Repeat steps 6-8 in an effort to claim the second Beacon in a highly reliable manner. 12.) Pivot, drive, and park on the Center Vortex Base while knocking off the Cap Ball using the motor encoder function.
<p>Our blue autonomous is depicted in the image above in this program diagram. The red alliance has the same coding functions, with it's path simply being a mirror image of that being currently illustrated in this pathing diagram.</p>	
	



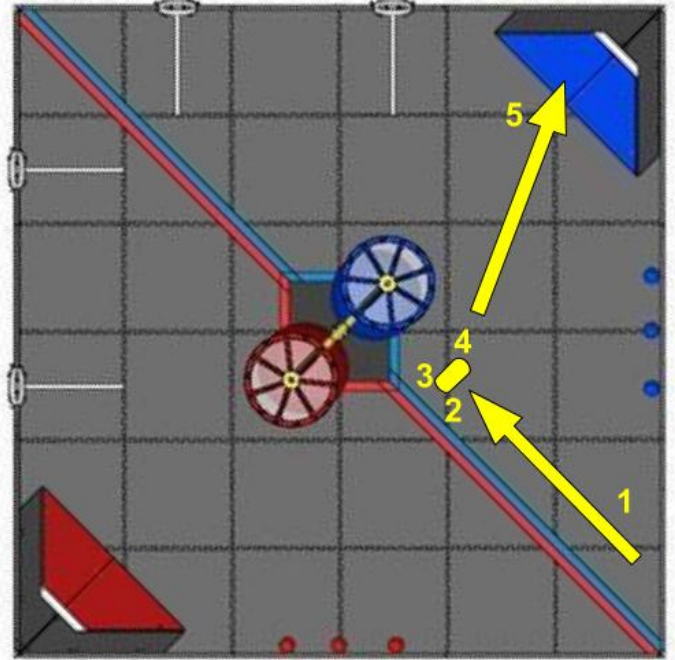
**FIRST
TECH
CHALLENGE**

Control Award Content Sheet

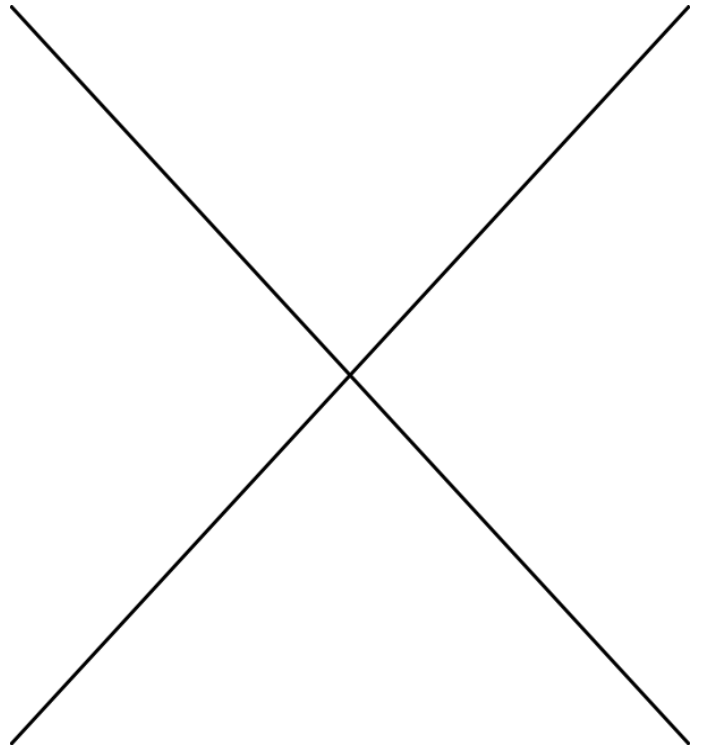
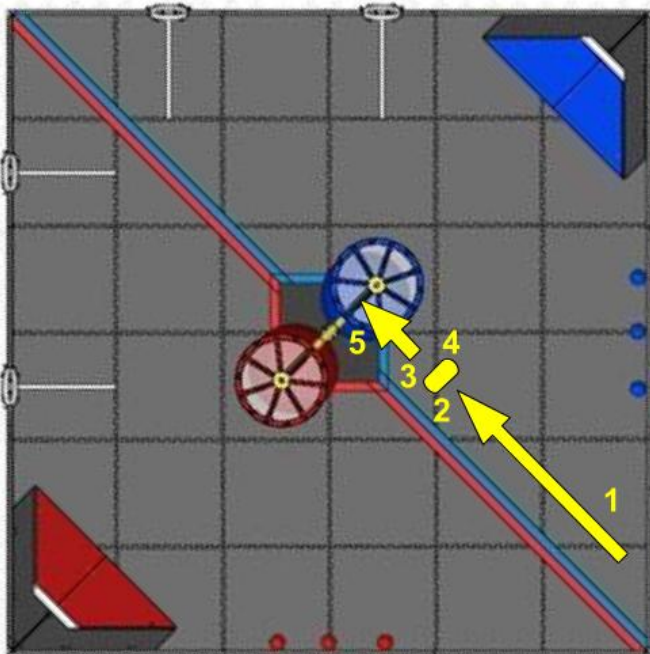
Beacon + Ramp | Autonomous Variation

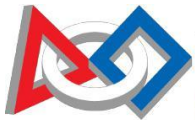


Complement + Ramp | Autonomous Variation



Complement + Center Autonomous Variation





FIRST
TECH
CHALLENGE

Control Award Content Sheet

Additional Summary Information

In conclusion, creating a reliable set of autonomous operations this season has been a goal of ours that we felt was necessary for efficient and consistent success in the match setting. This year, we have integrated a total of 7 sensors in order to enhance the intelligence of our robot to accomplish tasks in the game challenge. To maximize the full use of this hardware, we have integrated many innovative algorithms, in both the autonomous and teleoperated periods of competition, so that we can achieve efficient success in this challenge. Lastly, coming up with many innovative strategies and making this work with our code, both in terms of our many variations of programs, along with our unique teleop goals and tasks, has optimized our scoring in both major periods of the competition. We hope that this application guides you to the places in our programming where we highlight the places in which software improves our mechanical systems. If you have any questions, please feel free to visit us in the pit, and also refer to our extensive documentation of the software side of our robot, in both the programming and preseason sections of our engineering notebook.

