Control Award Content Sheet

****Please turn in this sheet during your Judge Interview along with your Engineering Notebook****

Team #3543	**Team Name: Titan Robotics Club**

Autonomous objectives:

Our objectives are flexibility, fault tolerance and accuracy. We achieved flexibility and fault tolerance by providing hundreds of autonomous variations. We are using a set of menus that prompt the driver for different choices. For example, the driver can choose what color alliance we are in (2 variations: red, blue), whether we should delay starting our autonomous so we don't bump into the robot of our alliance partner, how many particles we are shooting (3 variations: 0, 1, 2), where we should park the robot at the end of autonomous (3 variations: stay, center or corner vortex), how many beacon buttons we are pushing (3 variations: 0, 1, 2) etc. All these choices became parameters feeding into 8 strategies. The total combinations are approximately 2x2x2x3x3x3 + 2x2x2x3x3 + 2x2x3x2 + 2 + 2 + 1 = 317. Detail description with diagrams of each strategy can be found in a later section. The 8 autonomous strategies are listed below. Not only do they provide flexibility that accommodates any of our alliance partners, they also provide fault tolerance so that even if part of our robot is malfunctioning, we will have an autonomous that will still work.

1. **Do Nothing –** No autonomous. This is useful when our robot is malfunctioning and cannot perform any autonomous.
2. **Auto 100 path 1** – 100 maximum autonomous point. This strategy has many variations but will score a maximum of 100 points including shoot 2 particles (30-point), displace Cap Ball (5-point), press buttons of two beacons (60-point) and finally park at either the center or corner vortex (5-point).
3. **Auto 100 path 2** – Similar to **Auto 100 path 1** except going for a slightly different path. This is useful to confuse our opponents on which way we may go so they may not deploy an effective defense strategy against our autonomous.
4. **Auto 40 Near –** 40 maximum autonomous point. This strategy also has many variations but will score a maximum of 40 points including shoot 2 particles (30-point), displace Cap Ball (5-point) and finally park at either the center or corner vortex (5-point). We use this strategy if our robot is malfunctioning on the subsystems that are required to push beacon buttons (e.g. button pushers, color sensor, line detection light sensor, ultrasonic range sensor etc.). We only require encoders and gyro for this strategy. We may also use this strategy if our alliance partner can push the beacon buttons reliably and nothing else so we can complement them by scoring a maximum of 40 additional points.
5. **Auto 40 Far –** Similar to **Auto 40 Near** except for a different starting location of the robot. In **Auto 40 Near**, the robot is placed near the corner vortex as a starting location. In **Auto 40 Far**, the robot is placed far from the corner vortex. This is useful if our alliance partner must start close to the corner vortex.
6. **Auto 40 Far Defense –** This strategy extends the Auto 40 Far strategy. Not only will it score a maximum of 40 points, it will go across the alliance line to the opponent's side after 10 seconds and will park itself in between the two opponent beacons thus disrupting our opponent's autonomous run. It has a potential of depriving our opponents up to 35 points (one beacon and park at one of the vortexes). At the end, our robot will attempt to park on the opponent's side of the center vortex and may potentially displace their robot off the center vortex thus depriving another 5 points from their autonomous score. As an additional bonus, it also deprives one particle from the opposing alliance in TeleOp.
7. **Distance Drive –** Drive the robot for a specified distance. This is a general-purpose strategy. Again, if our robot is malfunctioning, as long as drive base encoders are still working, we can place the robot

and aim at a target (e.g. Cap Ball and center vortex) and just run it. We can also use it as a defense strategy, delaying the autonomous for 10 seconds and run across to the opponent side blocking their autonomous.

8. **Timed Drive** – Drive the robot for a specified amount of time with specified power. This is also a general-purpose strategy. Again, if our robot is malfunctioning, as long as the drive base motors still run (doesn't even need encoders), we can place the robot and aim at a target (e.g. Cap Ball and center vortex) and just run it.

Here is a link to a video showing one of our earlier versions of the 100-point autonomous strategies: https://www.youtube.com/watch?v=QHcIzLSJXnw

Sensors used:

One of our objectives is to make our autonomous very accurate. We achieved this by using different sensors.

- **Motor encoders**: Our robot uses encoders on the mecanum drive base as well as the particle accelerator (shooter). On the drive base, we use them to measure the distance traveled by the robot in both X and Y directions. On the particle accelerator, it measures the position of the shooter mechanism.
- **Touch sensor**: Although we have an encoder on the particle shooter motor to tell us how much it has turned, it does not tell us the absolute position of the shooter and when it has fired. The touch sensor activated by a cam on the sprocket axle will tell us that.
- **Gyro sensor**: Keeps track of robot heading for precision turn and running straight. Our turns are very precise because we use absolute heading to eliminate cumulative error. At the beginning of the season, we had problem tuning the autonomous turn of our robot. After intensive investigation (http://ftcforum.usfirst.org/showthread.php?7724-Gyro-Read-Latency-Development-team-update), we found out the number of devices on the I2C bus adversely affects the sampling interval of sensor readings. Our Modern Robotics gyro gave us a different readings every 300 msec. There is no way to do close-loop control with such sensor readings so we replaced the I2C Modern Robotics Gyro with an old analog gyro from our FRC team. But analog gyro only gives us rotation rate, not heading. Fortunately, our library supports software integration for analog gyro so it works for us.
- **Color sensor**: Tells us the left side color of the beacon.
- **Optical Distance sensor**: Mounted at the bottom of the robot to detect the white line.
- **Range sensor**: Tells us the distance of the robot from the beacon wall.
- **Android phone camera**: In our *Auto 40 Far Defense* strategy, we park our robot in between our opponent's two beacons. This is a high-risk strategy because it will incur a 40-point penalty if our robot parks partly on the floor tile in front of the beacon. In order to minimize this possibility, we use Vuforia to look for one of the vision targets as a reference point so that the robot can accurately navigate itself to the safe floor tile in between the two beacons.

Key algorithms:

PID (**P**roportional/**I**ntegral/**D**ifferential) control:

- Instead of using the Modern Robotics built-in PID control which is doing individual motor control, we used overall PID control across all wheel motors which maintains power distribution between them; combined with motor encoders, this allows our robot to shift power across motors if one is moving faster than the others, preventing the robot from unintentional turning.
- Our PID controlled drive algorithm consists of three software PID controllers: one to calculate net power output for the X direction, another for the Y direction and one for maintaining the robot heading. By combining these three PID output, we are able to distribute power appropriately to each wheel and accurately navigate the robot to the desired target position. This cannot be achieved by using Modern Robotics built-in PID control because individual motor PID control does not take into account of other motors.
- Our PID controlled drive can be used not only with encoders and gyro, it can be used with any sensors for different purposes. For example, if the X PID controller uses an ultrasonic range sensor to control the distance from the wall, the Y PID controller uses the encoders to move the robot parallel to the wall in the Y direction and the turn PID controller uses the gyro to maintain the heading, then we have a wall follower. Another example: if the Y PID controller uses the ultrasonic range sensor to stop the robot when it's too close to the wall and the turn PID controller uses the light sensor following the edge of the line by adjusting the turn of the robot, then we have a line follower.

Driver controlled enhancements:

One of the unique features that we have developed is a menu driven autonomous selection. As we have mentioned, we have 317 variations of autonomous strategies. Without this menu selection system, we would have to create 317 OpMode programs. Not only making it impossible to write, it also makes it very difficult to maintain. Most of the OpModes would have very similar code with only minor differences. Tuning all OpModes will be a nightmare. Instead, we have only three OpMode programs: Autonomous, TeleOp and Test. When running the Autonomous OpMode, the driver is presented with various menus displayed on the Driver Station forming a decision tree. The driver will make choices using the gamepad controls to scroll through options. Once all the choices are made, the selected strategy will be run with other selected choices as parameters. The diagram in the next section shows the organization of this menu tree.

Another unique feature is our Test OpMode. It serves two purposes: diagnostic and calibration. If the robot is not functioning properly, it allows the driver to diagnose the root cause of the problem. When run, the Test OpMode also presents a menu of diagnostic tests to the driver:

- **Sensors Test**: This test reads every sensor on the robot and display their values on the Driver Station in real time. While this test is running, the driver has full TeleOp control so he/she can drive the robot around as well as test every subsystem.
- **Drive Motors Test**: This test runs each of the drive motors on the drive base sequentially for 5 seconds each with the same power and displays the encoder count of each motor at the end so that if the encoder values have major differences, it could mean some of the wheels have mechanical problem such as friction, chain tension or weak motor etc. We can also verify if the motors have correct directions with this test.
- **X Timed Drive Test**: This test runs the drive base in the X direction for the selected number of seconds. It is primarily used for calibrating the INCHES_PER_ENCODER_COUNT scaling

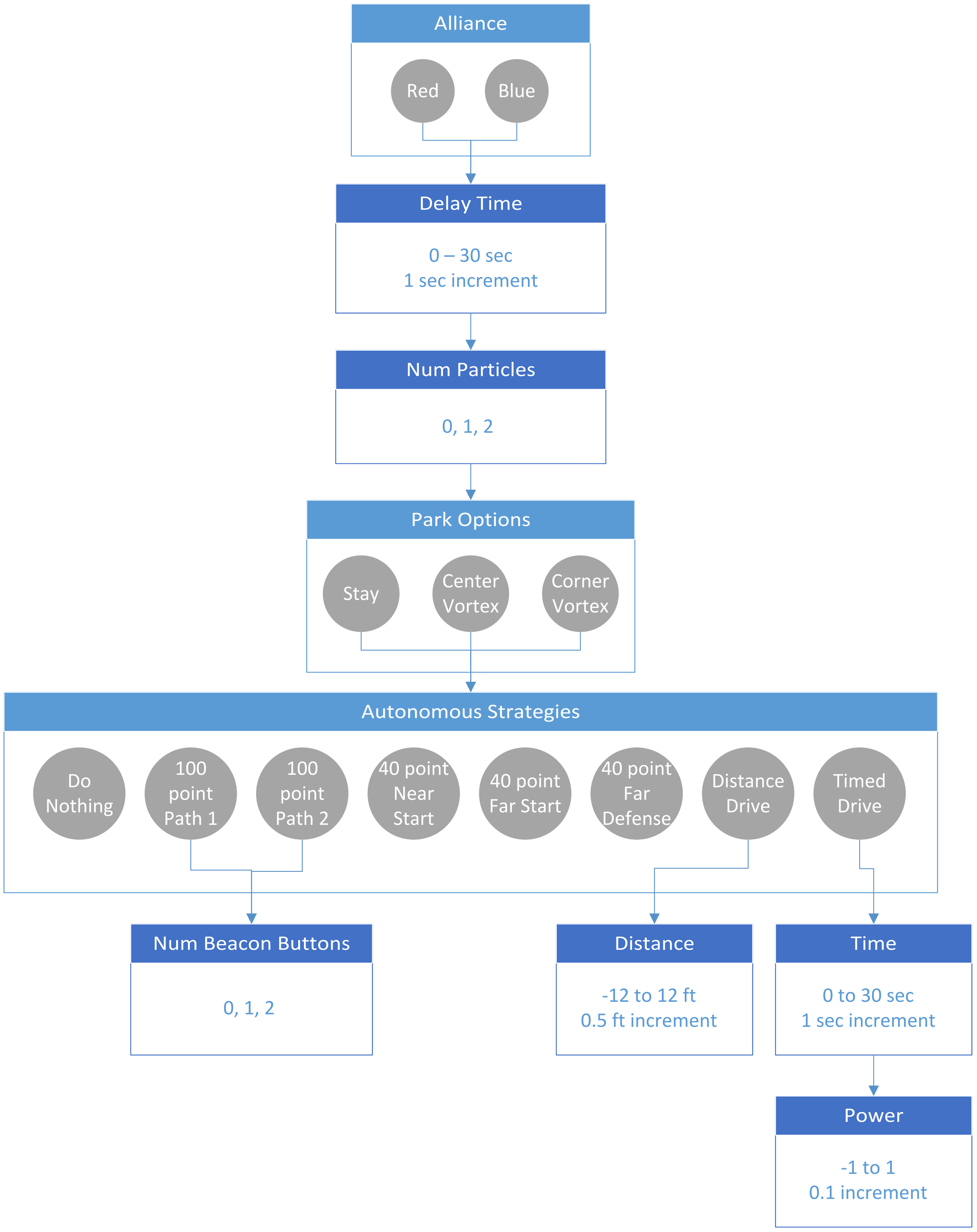
factor for the X direction. At the end of the run, encoder values are displayed on the Driver Station and one can measure the distance traveled using a tape measure. The scaling factor is determined by dividing the measured distance with the encoder count.

- **Y Timed Drive Test**: This test is similar to the **X Timed Drive Test** but for the Y direction.
- **X Distance Drive Test**: This test runs the drive base in the X direction for the selected distance. It is primarily used for calibrating the PID constants for the X PID controller. One can adjust the PID constants to make the robot go as close to the target distance as possible without oscillation and within tolerance.
- **Y Distance Drive Test**: This test is similar to the **X Distance Drive Test** but for the Y direction.
- **Gyro Turn Test**: This test rotates the drive base for the selected number of degrees. It is primarily used for calibrating the PID constants for the Turn PID controller. One can adjust the PID constants to make the robot rotate as close to the target heading as possible without oscillation and within tolerance.
- **Range Drive Test**: This test runs the robot towards the wall for the selected stopping distance. It is primarily used for calibrating the PID constants for the Range PID controller. One can adjust the PID constants to make the robot go as close to the target wall distance as possible without oscillation and within tolerance.

All these features are supported by our library and is available on GitHub as part of our Open Source Software. (https://github.com/trc492/Ftc2017VelocityVortex)

Control Award Content Sheet

Autonomous Choice Menus



Control Award Content Sheet

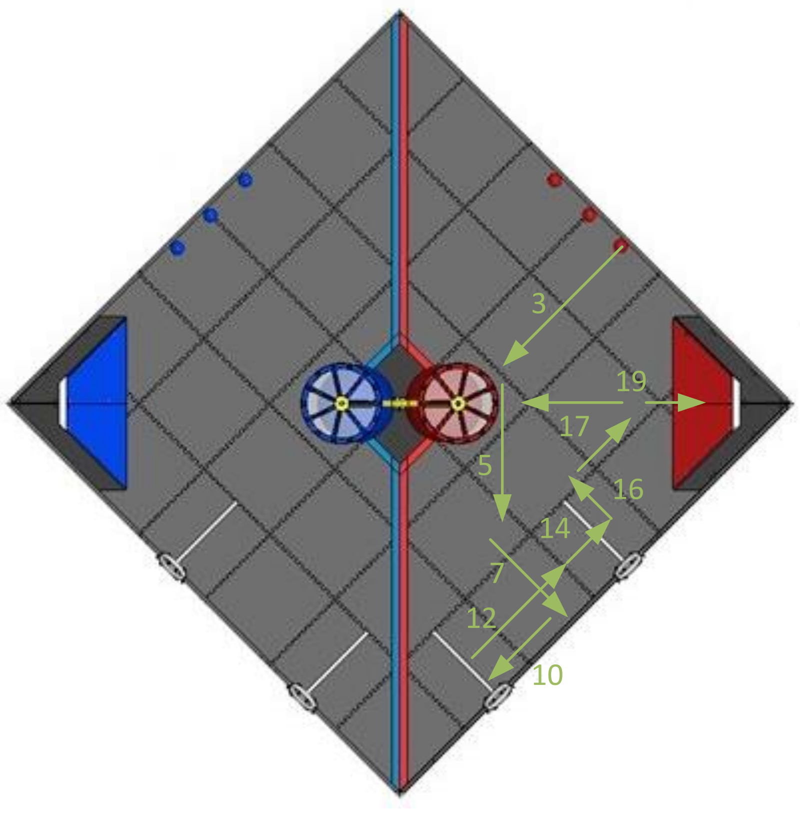
Engineering notebook references:

Feature	Notebook Pages
Autonomous Strategies	147-155
Sensors	130-136
PID Control	145
Choice Menu System	146
Test OpMode	143

Autonomous program diagrams:

The following autonomous diagrams show our robot on the red alliance. When we are on the blue alliance, our robot's paths and functions are mirrored.

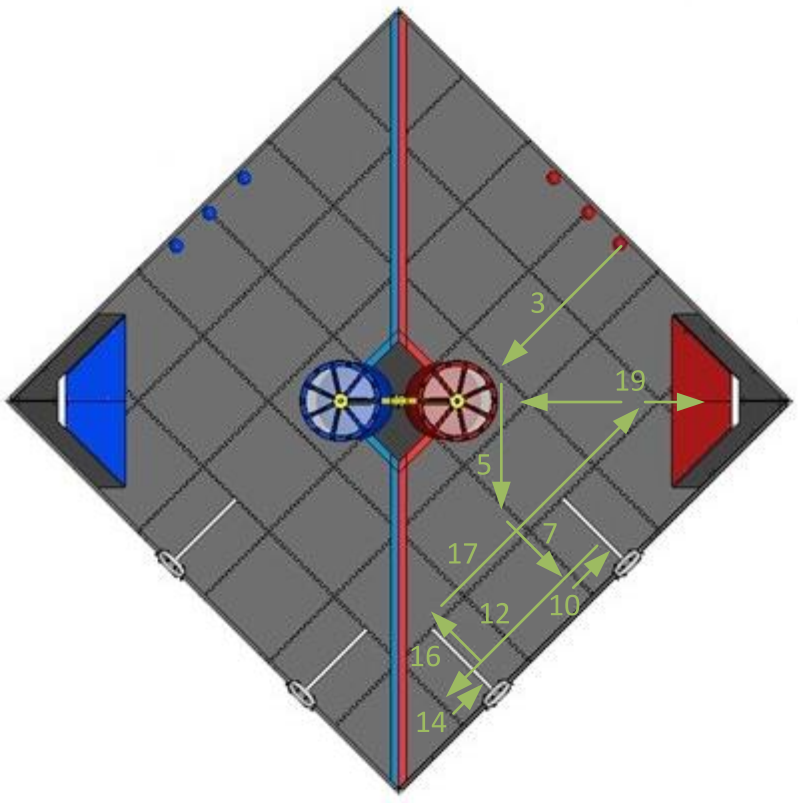
Auto 100 Path 1:



1. Shoot the selected number of particles.
2. Delay the selected number of seconds if necessary.
3. Move towards the center vortex.
4. Turn left so the robot can graze the Cap ball on the right.
5. Move forward to displace the Cap ball.
6. Turn towards the beacon wall.
7. Move towards the wall.
8. Turn right parallel to the wall.
9. Strafe left and hit the wall gently to realign the robot to the wall.

10. Move forward slowly along the wall looking for the white line, stop when found.
11. Detect the left side color of the beacon and press the appropriate button.
12. Move backward before passing the white line of the second beacon.
13. Strafe left and hit the wall gently to realign the robot to the wall.
14. Move backward slowly along the wall looking for the white line, stop when found.
15. Detect the left side color of the beacon and press the appropriate button.
16. Strafe right away from the wall.
17. Move backward towards the corner vortex.
18. Turn right and have the back towards the corner vortex.
19. Move forward to park on the center vortex or backward to park on the corner vortex.

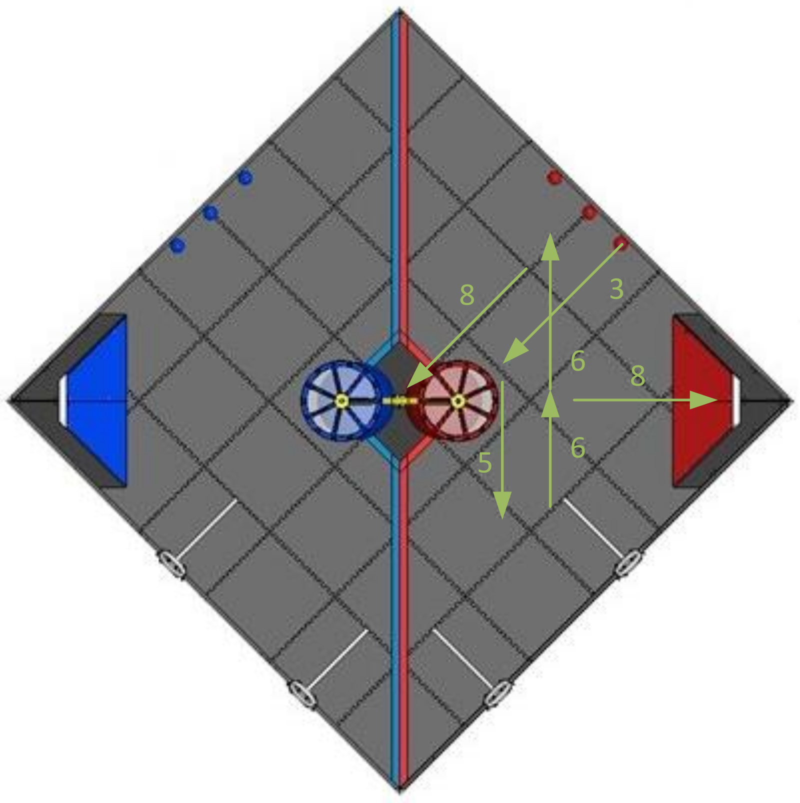
Auto 100 Path 2:



1. Shoot the selected number of particles.
2. Delay the selected number of seconds if necessary.
3. Move towards the center vortex.
4. Turn left so the robot can graze the Cap ball on the right.
5. Move forward to displace the Cap ball.
6. Turn towards the beacon wall.
7. Move towards the wall.
8. Turn right parallel to the wall.
9. Strafe left and hit the wall gently to realign the robot to the wall.

10. Move backward slowly along the wall looking for the white line, stop when found.
11. Detect the left side color of the beacon and press the appropriate button.
12. Move forward pass the white line of the second beacon.
13. Strafe left and hit the wall gently to realign the robot to the wall.
14. Move backward slowly along the wall looking for the white line, stop when found.
15. Detect the left side color of the beacon and press the appropriate button.
16. Strafe right away from the wall.
17. Move backward towards the corner vortex.
18. Turn right and have the back towards the corner vortex.
19. Move forward to park on the center vortex or backward to park on the corner vortex.

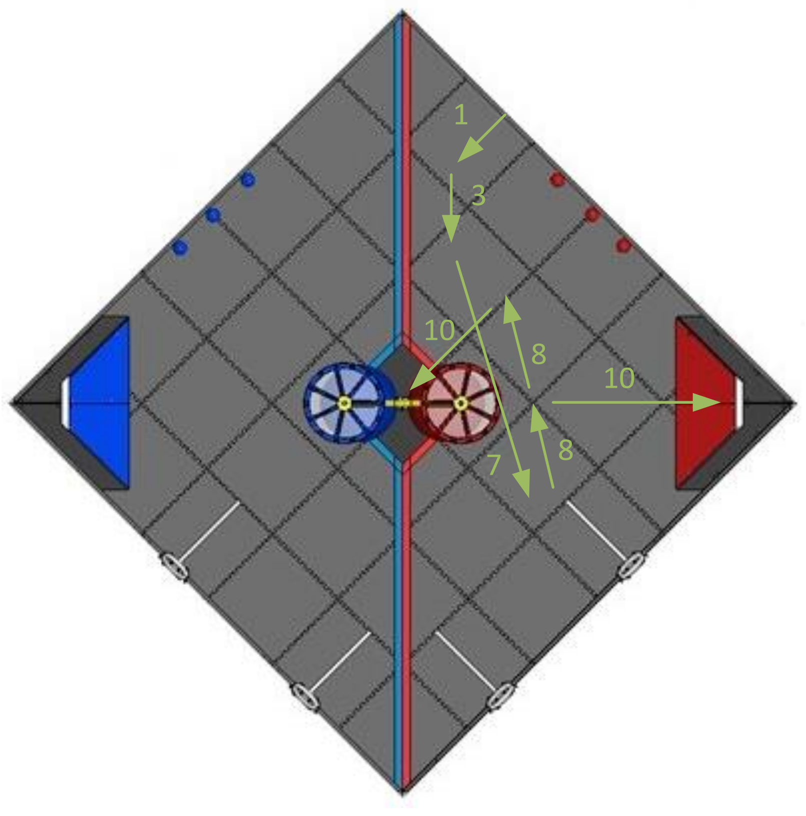
Auto 40 Near:



1. Shoot the selected number of particles.
2. Delay the selected number of seconds if necessary.
3. Move towards the center vortex.
4. Turn left so the robot can graze the Cap ball on the right.
5. Move forward to displace the Cap ball.
6. If park on the corner vortex, move backward to align to the center of the corner vortex. If park on the center vortex move backward back to the starting wall.
7. If park on the corner vortex, turn right so the back is facing the corner vortex. If park on the center vortex turn right so the front is facing the center vortex.

8. If park on the corner vortex, move backward onto the corner vortex. If park on the center vortex, move forward onto the center vortex.

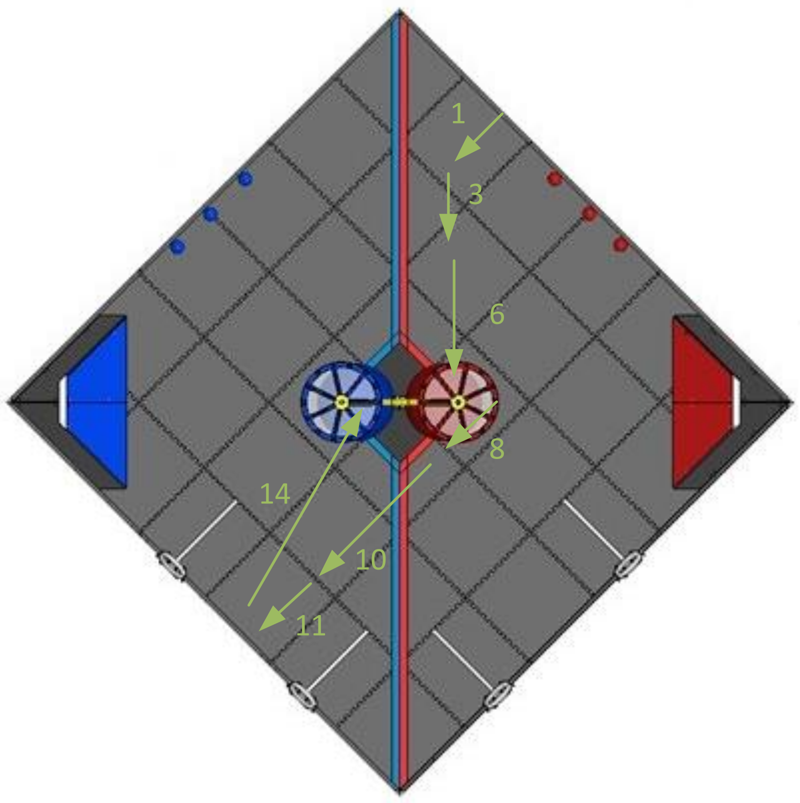
Auto 40 Far:



1. Move forward slightly towards the center.
2. Turn left to aim at the center vortex.
3. Move forward closer to the center vortex.
4. Shoot the selected number of particles.
5. Delay the selected number of seconds if necessary.
6. Turn left so the robot can graze the Cap ball on the right.
7. Move forward to displace the Cap ball.
8. If park on the corner vortex, move backward to align to the center of the corner vortex. If park on the center vortex move backward back to the starting wall.
9. If park on the corner vortex, turn right so the back is facing the corner vortex. If park on the center vortex turn right so the front is facing the center vortex.
10. If park on the corner vortex, move backward onto the corner vortex. If park on the center vortex, move forward onto the center vortex.

Control Award Content Sheet

Auto 40 Far Defense:



1. Move forward slightly towards the center.
2. Turn left to aim at the center vortex.
3. Move forward closer to the center vortex.
4. Shoot the selected number of particles.
5. Turn left so the robot can graze the Cap ball on the right.
6. Move forward to displace the Cap ball with the aid of the vision target to the left.
7. Turn towards the opponent beacon wall and look for the Gear vision target on the left.
8. Using the vision target as the reference point and move to the center vortex.
9. Wait until 10 second past autonomous start.
10. Move forward into opponent's territory and stop and look for the Tools vision target on the left.
11. Using the vison target as the reference point and move to the floor tile in between the two opponent's beacons.
12. Wait until 5 seconds before autonomous ends.
13. Turn slightly towards the opposing alliance's side of the center vortex.
14. Move backward and park at the center vortex potentially displacing the opponent's robot off the center vortex.