Control Award Sponsored by Arm Submission Form

Team #: 7163	Team Name: The Bazinga Project

Autonomous objectives:

Initially, the basic autonomous plan we came up with to score the greatest number of points we could was to spin the duck off, if we are near it, deliver the freight into the shipping hub, and then finally park into either the warehouse or the storage unit. Later, we implemented the use of TFOD (Tensor Flow Object Detection) and Vuforia, both are incorporated into the camera so that we can find the position of the duck on the barcode sequence to deliver freight to the correct level of the shipping hub. We use Github to store backup code just in case and it also includes information about our day-to-day progress.

Sensors used:

1. USB Camera – The USB Camera is used in the beginning of each autonomous to scan for the duck or capstone. The position is stored into a variable for later use.
2. Motor Encoders – The motor encoders are built-in encoders in each motor to help measure and determine the distance to drive in autonomous.

Key algorithms:

One of the things we wanted accomplish was to precisely dump freight into the correct level on the shipping hub. To achieve that, we added a camera onto our robot to scan for the duck during initialization – eventually replacing it with a 3D printed capstone. To program the camera to be able to see the duck, we decided use Vuforia to capture images and TFOD to detect an object from those images, which we used in a previous year. At first, we looked back at the code from the previous years to figure out how to use Vuforia and TFOD; we eventually got it to work, but we ran into a problem; the camera could not detect the duck even if it was in its view. To solve this, we simply narrowed the camera's view to certain places to detect an object.

When the camera detects an object, it runs through the program to figure out what the object is and where it is on the screen, and if it matches one of the set labels, the program then finds the position of the object and stores it in a variable which is later used to determine which level to dump the freight into.

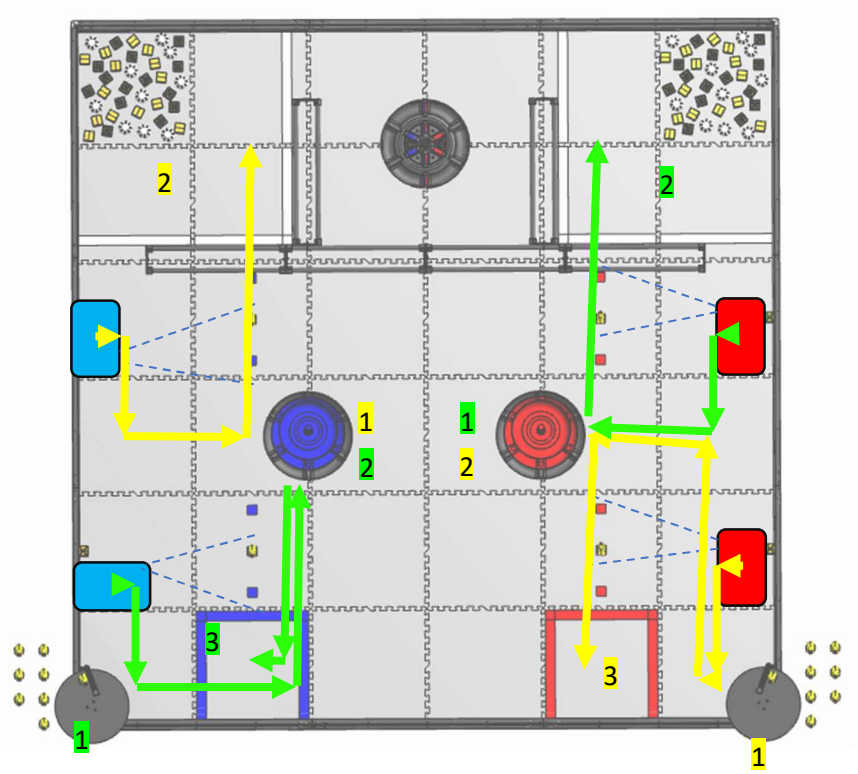
Another thing that we wanted to accomplish was the efficiency and accuracy of our autonomous programs. In previous years, we used the time method, moving the robot for a certain amount of time, we found this method to be inferior due to the way it correlates with the battery power. If the battery power was low, then the motors wouldn't have as much power as it should have, making movements inconsistent. We fixed this by optimizing the built it motor encoders. We found the tick speed and did a little math to be able to correctly use the encoders. The main benefit of this is that no matter what level the battery is at, the wheels will move the same amount no matter what, with a margin of error at approximatively one inch every 30 inches.

Driver controlled enhancements:

We like to make controlling the robot in TeleOp easy for the drivers so that we can score as many points as possible within the time limit everyone is given. Because our intake and the lift system are on opposite sides of the robot, it makes it very difficult to control if one side is backwards. To counter this, we created a reverse system where if our driver hits the right bumper, it reverses all the driving controls, and the left button makes it go back to normal. The benefit of this is to make it easy to grab a shipping element with the intake and save time by not turning 180 degrees, but rather, reverse

the controls to drive out the other way. Additionally, to make driving easier, knowing that our drivers play games, we utilized that information to set the controls like a racing video game, adding strafing as well.
We have a second driver that controls accessory mechanisms such as the lift and the tape measure for capping the capstone. To make things more efficient for this person, the tape measure mechanism moves the tape measure in and out depending on how long the joystick is pushed, this makes it easier to control and fast as well. Occasionally, while dumping a freight element into the shipping hub, the bucket that holds the element doesn't angle far enough for it to slide off. We fixed this by adding a button that slowly descends the bucket depending on how long the button is pressed.

Autonomous program diagrams:



We have 4 different Autonomous programs – two of which are mirrored to each other. Essentially, all of the programs deliver to the shipping hub and then moves to park; additionally, the two programs on the carousel side spin the duck off, as shown in the diagram above. The blue dotted lines represent the camera's range and viewpoint.

Carousel Sides – Though each side takes different paths, they both have the same objectives.

1. Our robot will strafe out a little bit and slowly approach the carousel to spin the duck off.
2. Next, the robot will move to a position to approach the shipping hub and then place the given freight into its corresponding level told by the barcode in the beginning.
3. Lastly, the robot will move hastily to the storage unit to park.

Warehouse Sides – Blue side was mirrored from the red side; thus, they are the same but opposite.

1. Our robot will strafe out a little bit, and steadily move towards the shipping hub to drop the given freight to its respectable level told by the barcode in the beginning
2. Lastly, the robot will ram itself into the warehouse to park.