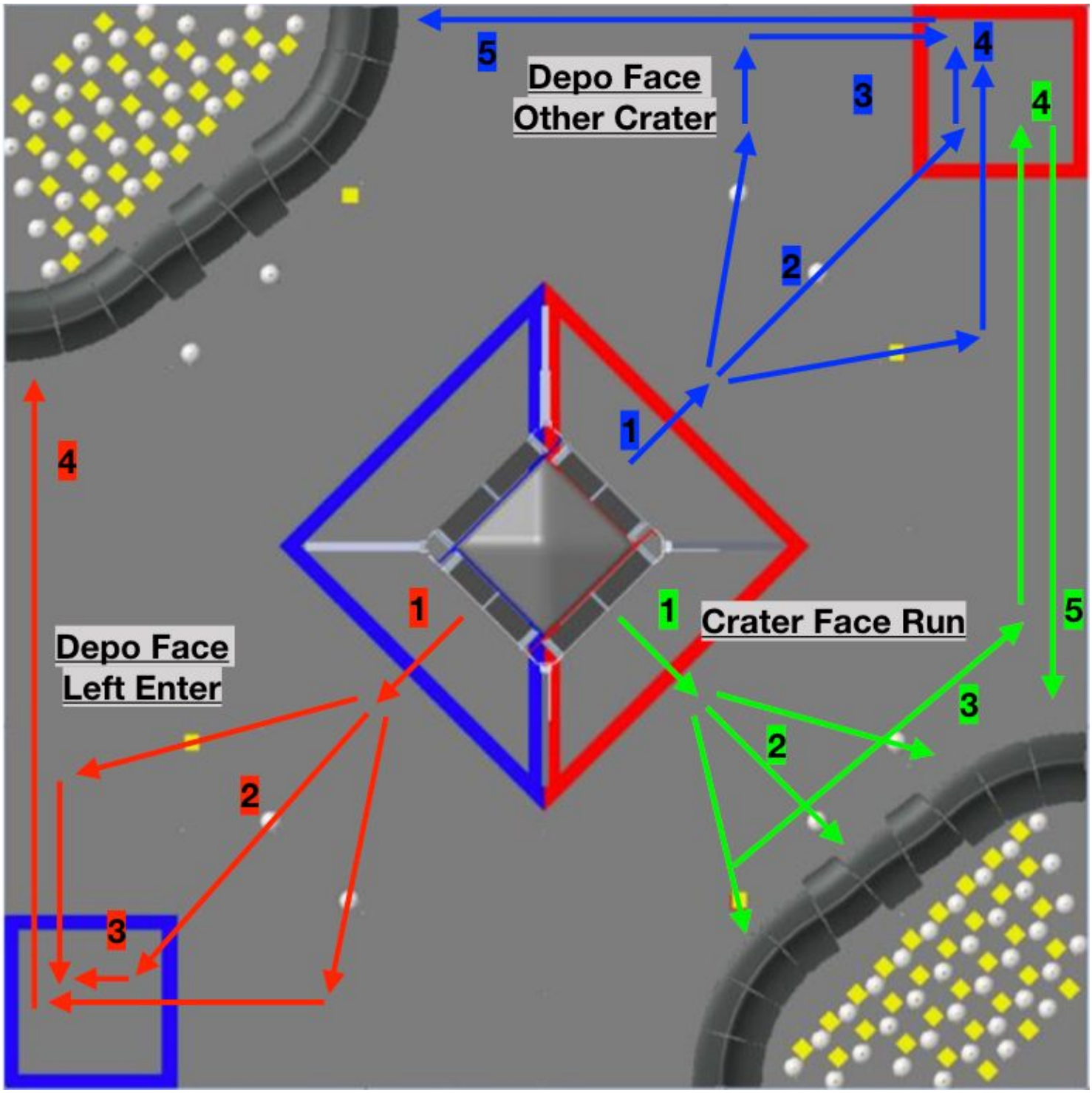
Control Award Content Sheet

Team #: 8668

Team Name: Error 404: Team Name Not Found



Autonomous:

- Our autonomous code is a state machine that uses a CSV file in the phone's internal storage to define, load, and execute robot actions. There are multiple CSV files on the phone, one for each drive path, and a specific file can be selected using child classes of the autonomous class. We have three preset drive paths (one drive path per CSV file), but these drive paths can be easily modified before queing to adapt to our alliance partner. The drive path are listed below and are depicted on the front page.

Depot Face Other Crater (option 1 for depo face - both alliance colors)

1. Land, reset gyro, and find gold mineral location.
2. Turn towards gold mineral and push gold mineral off tape.
3. Turn to 45 degrees and strafe into wall.
4. Drive to depot and drop marker.
5. Drive to crater and deploy arm.

Depot Face Left Enter (option 2 for depo face - both alliance colors)

1. Land, reset gyro, and find gold mineral location.
2. Turn towards gold mineral and push gold mineral off tape.
3. Face crater, strafe into wall, and deploy marker.
4. Drive to crater and deploy arm.

Crater Face (both blue and red alliance)

1. Land, reset gyro, and find gold mineral location.
2. Turn towards gold mineral and push gold mineral off tape.
3. Strafe left to wall.
4. Drive to depot and deploy marker.
5. Drive to crater and deploy arm.

Driver Controlled:

- Multiple remotes for driver flexibility (one remote for driving and hanging, another remote for operation of mineral arm and collector).
- Mecanum drive (on main remote) allows for quick and nimble driving of the robot. Mecanum controls include:
  - tank drive - ie: typical four wheel drive
  - strafing, also known as "crabbing" - moving sideways and/or diagonally
  - turning - point turns or curves
- Also on main remote is slow turning for minute adjustments when collecting minerals and a turbo function that brings the robot to full power for quick cross-field dashes.
- Main remote has controls for operating the hanger. There is a slow feature that allows the driver to move the hanger fast or slow depending on the need. There are also buttons that send the chassis to preset positions:
  - Mineral Deploy: Orientes the robot to deploy minerals to the lander.

  - Hang1: Orientes the robot to hang on the latch the robot started on.
  - Hang2: Orientes the robot to hang on the other latch (not the one the robot started on).
- Secondary remote has controls for operating the mineral arm and the collector.
  - The secondary remote also has buttons that quickly move the arm to preset positions. This allows for quick and accurate movement of the arm:
    - ArmHome: collapses the arm down into its storage or home position.
    - ArmCollect: extends the arm all the way into the mineral collecting position.
    - ArmDeploy: brings the arm straight up for deploying minerals into the lander.
    - ArmDrive: folds the arm into the transport position for when the robot is driving from the crater to the lander and back.

Sensors Used:

- A navX Micro sensor from Kauai Labs; gives us 3 axis gyro and accelerometer. All drive and turn moves are made using the gyro.
- An external logitech webcam for mineral detection
  - We see a lot of teams using phone cameras for the mineral sampling. We use a webcam because it is easier to position on the robot and because it has higher resolution, making it easier to detect the gold mineral.
- Two limit switches on our mineral arm to prevent the robot from driving the arm into itself or the chassis during autonomous and teleop. When pressed, the motor power for the appropriate motor is reversed until the switch is released.
- Motor encoders on the dc motors. We use the encoders on the motors for calculating how far the motor has turned and where it is relative to zero. The encoders on the arm are used for the presets and the encoders on the drive motors are used for driving a set distance.

Key Algorithms:

- We use CSV files loaded on the phone's internal storage to define our autonomous drive paths. Our code takes whichever CSV file is selected and reads through it and builds a run map of all the robot actions defined in the CSV file. The code then loads each robot action into the run list and executes them sequentially, following the information provided in each action's parameters to know which action to load and execute next. After a action is executed it is removed from the run map and the next listed action is loaded in its place.
- Lander Failsafe: We use the navx's ability to measure pitch as well as heading and roll to detect when the robot is stuck on the lander during the landing sequence in autonomous. If the robot is detected to be stuck, the code enters a failsafe cycle that takes the stress off the hanger and then moves the hanger up before trying to once again exit the latch and measuring the robot's pitch. If the failsafe executes three times in a row, the robot will stop for the remainder of autonomous so prevent any damage to the robot, its mechanisms, or the field elements.
- Our code uses a single core drive method that all other drive methods call. This one method is the only method that actually uses the motors. This allows for great flexibility when coding and also eliminates a lot of

bugs, as only one peice of code is touching the motors and therefore only one piece of code needs to be debugged and changed when upgrading something. The core method is called joystickDrive().

  - The joystickDrive() method uses the gyro to move in the desired direction (forward, backward, side to side, or diagonal) and automatically converts the input inches to the required number of encoder ticks. The method can recover from being pushed off course and will correct any error. This method was designed to be both in teleop and autonomous.

- TensorFlow: We use a modified version of the example program from FIRST that uses Google's TensorFlow Object Detection algorithm and a Vuforia engine. The original code needed to see all three minerals to be able to locate the gold mineral and would sometimes mistake red tape on the field as gold. We modified the code to only require one gold mineral. We look for gold and then check its height to weed out any tape and we also check its x-axis position on the screen. If both conditions pass than we know we've found the gold mineral.
- PointTurn: Our point turn method has two pretty neat features built into it:
  - When the code is given a target value and it automatically calculates the shortest turn distance to that target and turns that direction. This is hard because our gyro is a +/- 0 to +/- 180 degrees gyro meaning that at 180 degrees, the gyro flips 360 degrees to -180 and continues on. This does weird things to an algorithm. Our code has to be able to bridge that gap without loosing where it is and has to be able to find the shortest turn distance, ignoring the gap, and then set the motors to turn that direction.
  - Also, when turning to the target angle, the code has a PID controller that slows the robot down once it enters a plus or minus value that is close to the target angle. This is so that the robot can turn at a very high power and still turn to the correct heading. This is very useful in teleop especially because the chassis presets that the drivers have turn the chassis quickly and before the PID controller was inserted, the chassis would often severely overshoot or even completely miss the target heading.

Engineering Notebook References:

- Software Design Document
- Class Diagrams
- State Machine Diagrams
- Wiring Diagrams
- Engineering Log References:
  - Ch. 1, pg. 10 - Motor types discussed
  - Ch. 1, pg. 16 - Vuforia testing
  - Ch. 2, pg. 4 - Code structure
  - Ch. 2, pg. 16 - PixyCam testing
  - Ch. 2, pg. 24 - Autonomous code structure
  - Ch. 3A, pg. 2 - Teleop code
  - Ch. 3A, pg. 12 - Gyro in all drive methods
  - Ch. 3A, pg. 13 - Gyro in point turn
  - Ch. 3B, pg. 4 - TensorFLow testing
  - Ch. 4, pg. 7 - TensorFlow in autonomous
  - Ch. 4, pg. 9 - Modifying TensorFlow
  - Ch. 5, pg. 11-12 - Changing autonomous drive paths
  - Ch. 5, pg. 14 - mineral sampling
  - Ch. 6, pg. 5 - Software upgrades
  - Ch. 6, pg. 12-13 - Testing and modifying TensorFLow to ignore red tape
  - Ch. 6, pg. 25 - Mineral arm presets
  - Ch. 7, pg. 5 - Autonomous Opportunities
  - Ch. 78, pg. 13 - TensorFlow improvements and lander failsafe
  - Ch. 7, pg. 17 - New autonomous structure

- Ch. 4, pg. 17 - Using the arm and gyro in autonomous
- Ch. 7, pg. 29 & 30 - Implementing CSV file reader