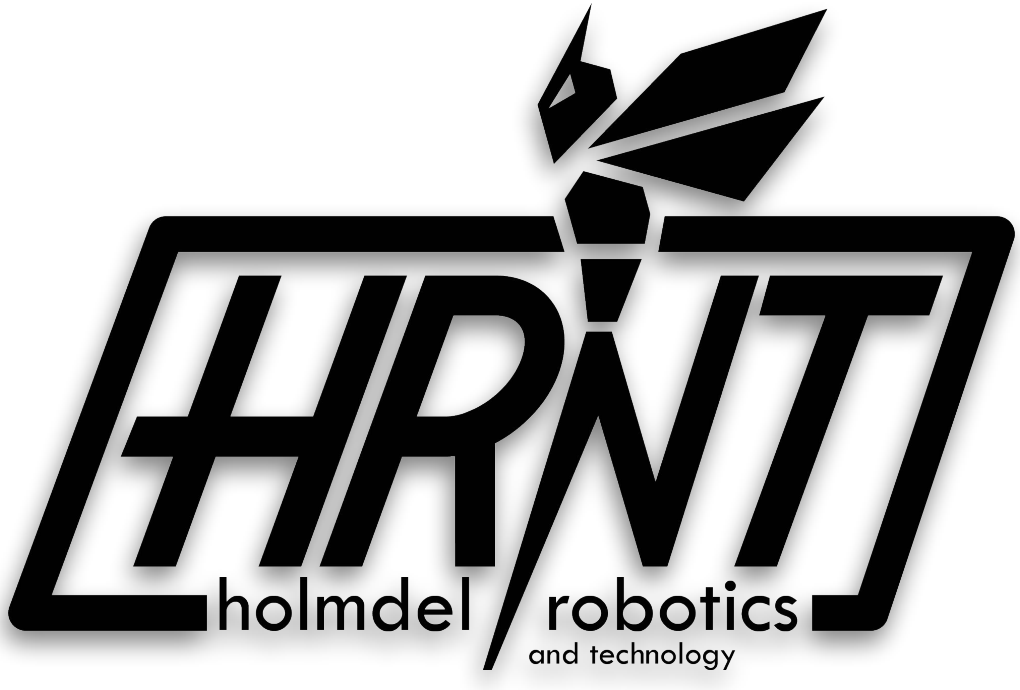
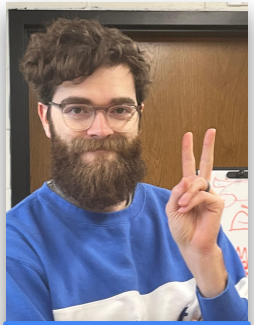


FTC Team #22261
Hornet Blue
Engineering Portfolio

Team 22261 HORNET BLUE

MEET THE TEAM

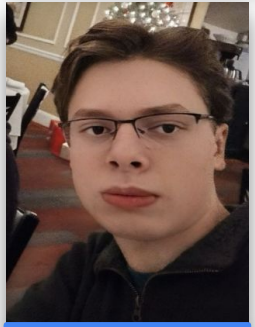
Innovate Award Winners



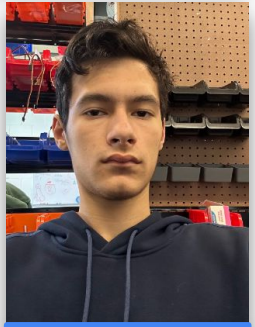
Mr. Bocchino
Coach for Hornet Blue

Finley
Engineering & Design Captain

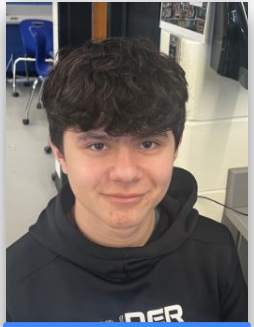
John
Programming Captain



Brian
Team Driver and chassis builder



Arthur
Building assistant



Kyle
Beacon designer

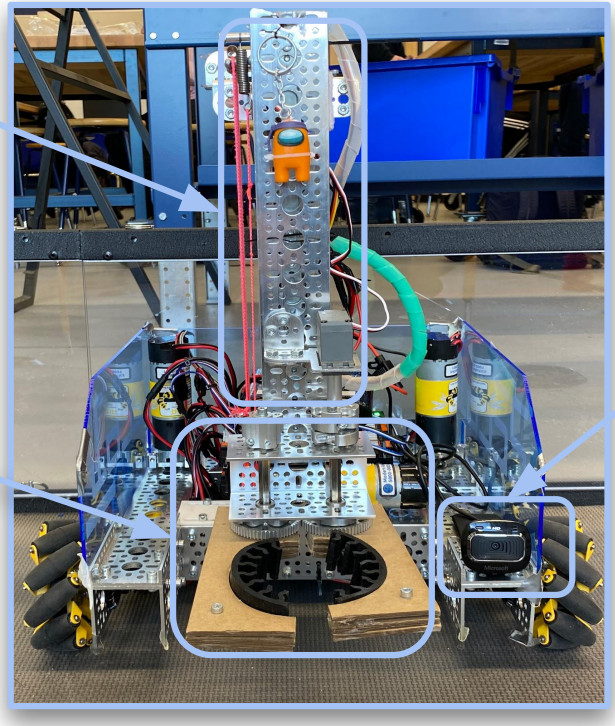
James
Strategist & CAD assistant

Aidan
Lead Strategist

FIRST ROBOT DESIGN

3 stage linear slide adapted from 2 2-stage goBilda linear slide kits.

Laser cut cardboard gripper. Uses two metal gears to rotate the two sides of the gripper in opposite directions. A rubber wheel that was cut in half is used for better grip.

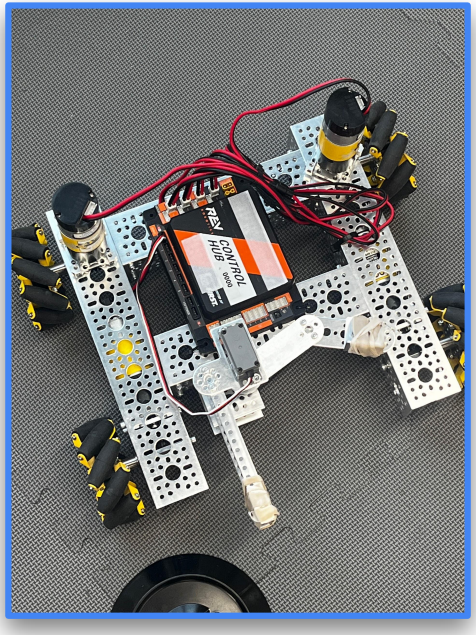


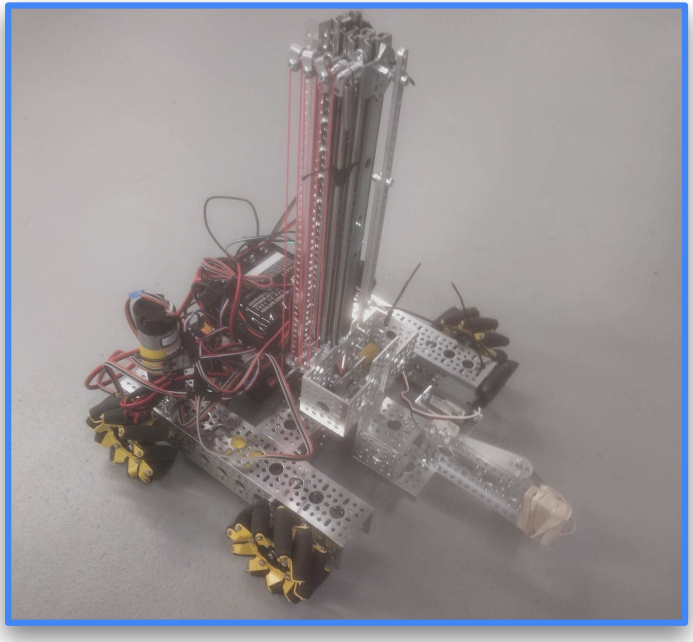
Webcam installed in order to read the custom cone sleeve for additional parking points, although code for it was not created yet.

FIRST ROBOT DESIGN - CONTINUED

Pros: Chassis is small and lightweight enabling quick, easy movement around the field.

Cons: Chassis was low enough it has issues navigating over the ground junction. The gripper was mounted using metal parts, causing the slide to struggle handling its weight. The slide also has only 3 stages, meaning that it could only reach up to the high junction. The gripper was also flimsy due to being made out of cardboard.



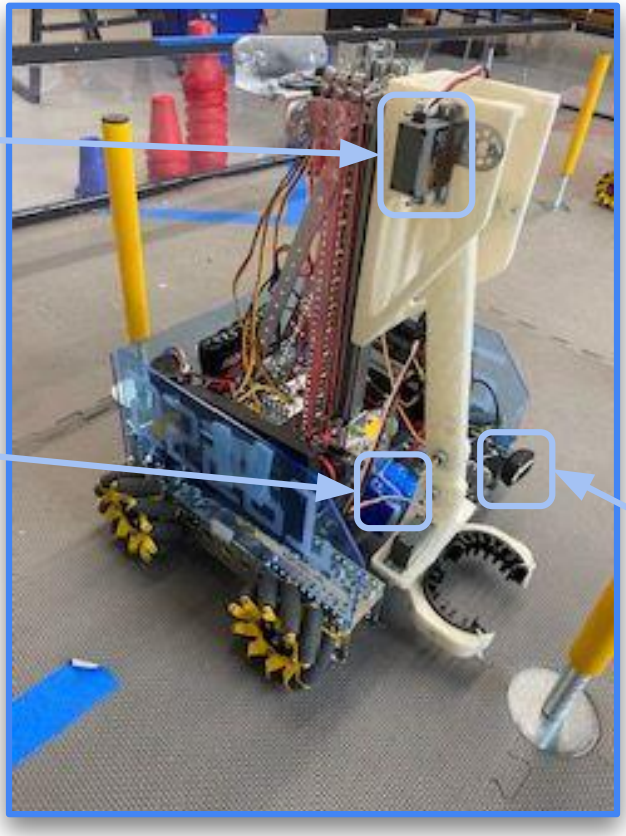


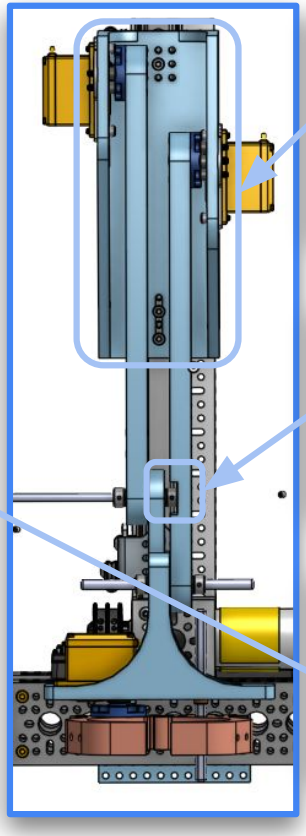
Previous versions of the robot utilized a claw with only one moving side. However this proved to be too inefficient.

SECOND ROBOT DESIGN

Only one servo in the linkage is actually connected. The other one is just there as a pivot point in lieu of an axle.

Servo “pillow” to provide support to the gripper at its resting position.





Fully 3D printed 4 bar linkage and claw designed using OnShape (CAD software).

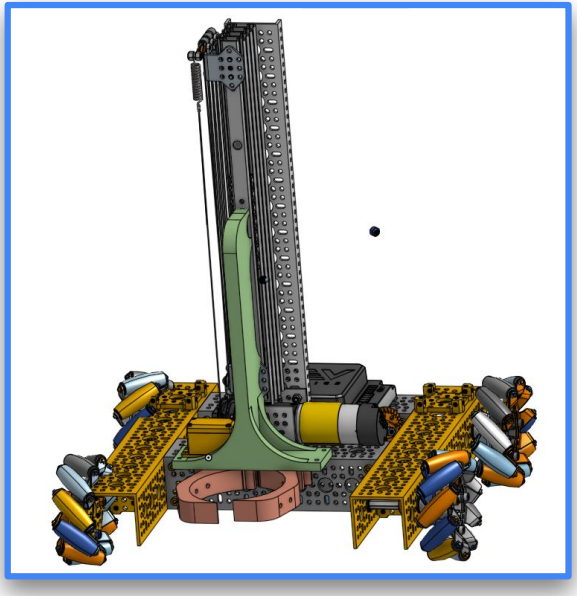
One linkage has a cutout in order to provide clearance for the servo collar.

The webcam now had code so the robot was able to properly read the signal cone.

SECOND ROBOT DESIGN - CONTINUED

Pros: The 4 bar linkage enables the gripper to have the same angle perpendicular to the robot no matter what. The 4 bar linkage and gripper being in PLA means that it weighs less than the metal while being stronger than cardboard.

Cons: The camera was not fully secured, and was prone to shifting in between matches, needing to be readjusted constantly. Due to there being only 1 linear slide, the arm would wobble, making aiming on the junctions difficult. Since the linear slide had both a extend and return string, the two strings were prone to tangling. Additionally, the battery would slip and disconnect itself occasionally.

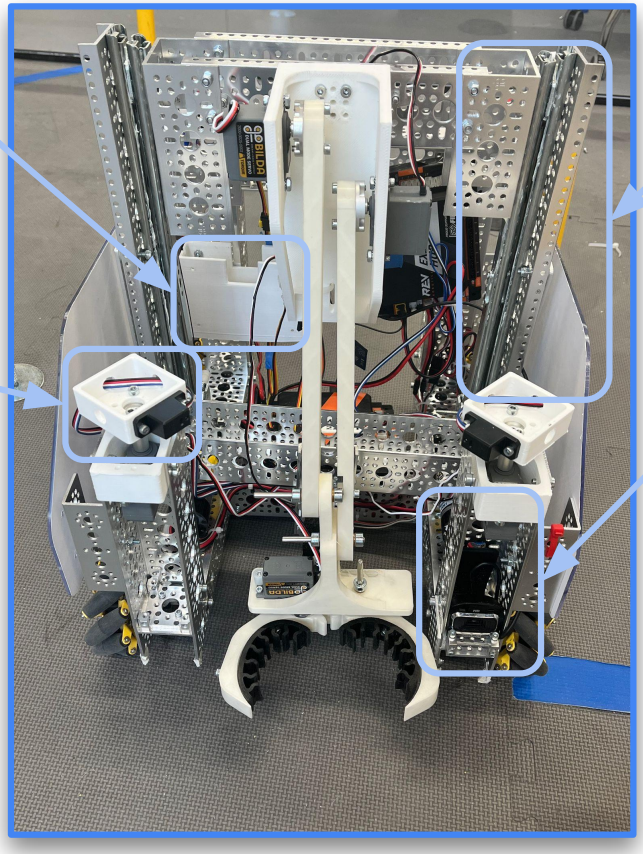


In between the first and second design a different mount for the gripper was made as a temporary solution to the weight problem.

THIRD ROBOT DESIGN

Camera mounted within a GoBilda channel for greater stability.

Two distances sensors are used to triangulate the position of a junction. The sensors are on servo to compensate for the gripper extending out different distances from the robot at certain angles.



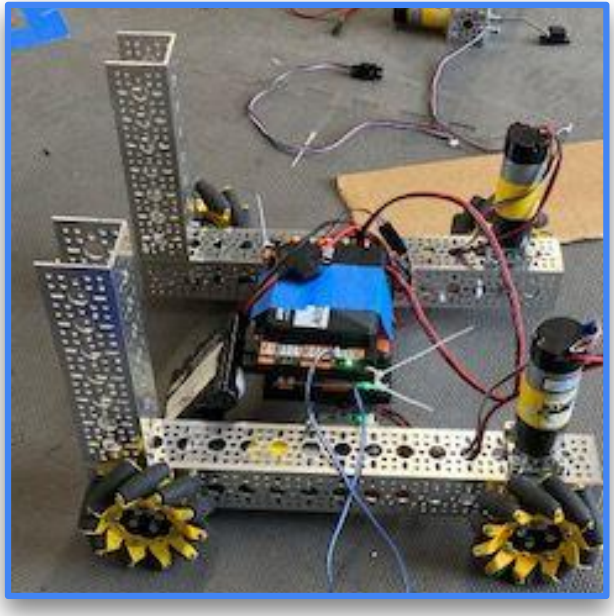
Dual 2 stage linear slides are used. However, only one linear slide is driven due to issues in the past having two motors cooperate to drive two linear slides at the same time.

Battery holder added to ensure a more stable connection.

THIRD ROBOT DESIGN - CONTINUED

Pros: The distance sensors allow a driver to align to a junction with just the push of a button. The dual linear slides minimize the linkage wobbling. Neither the camera or battery are prone to slipping anymore.

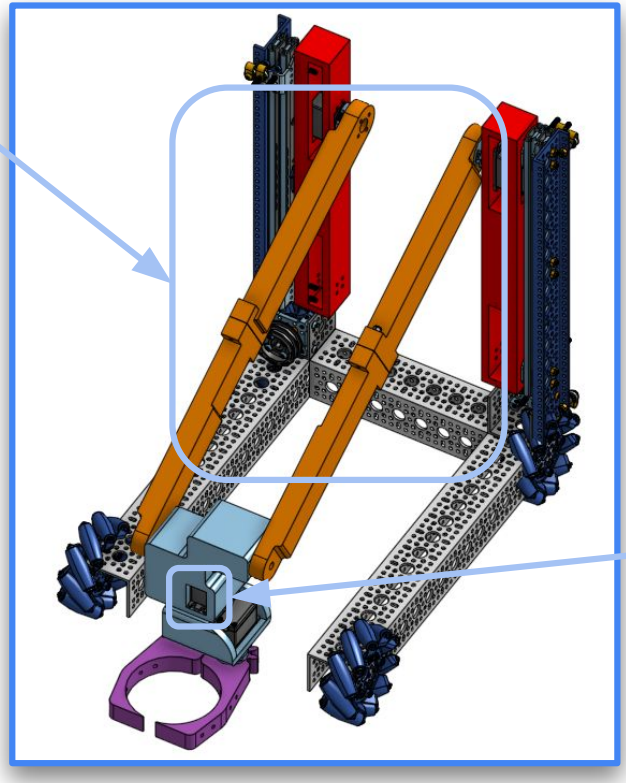
Cons: Due to only having one driven slide the other linear slide lags behind it when extended at high distances, causing the linkage and gripper to be slanted horizontally. The robot is also using 3 servos and 2 motors, which drains the battery quickly. If the robot is running on even a few volts under the max voltage of the battery, it becomes unreliable. Therefore batteries have to be swapped between every match.



The chassis had to be rebuilt in order to accommodate 2 linear slides instead of one.

FOURTH ROBOT (UNIMPLEMENTED)

New 4 bar linkage lets the gripper rotate to the other side of the robot without the robot moving itself.

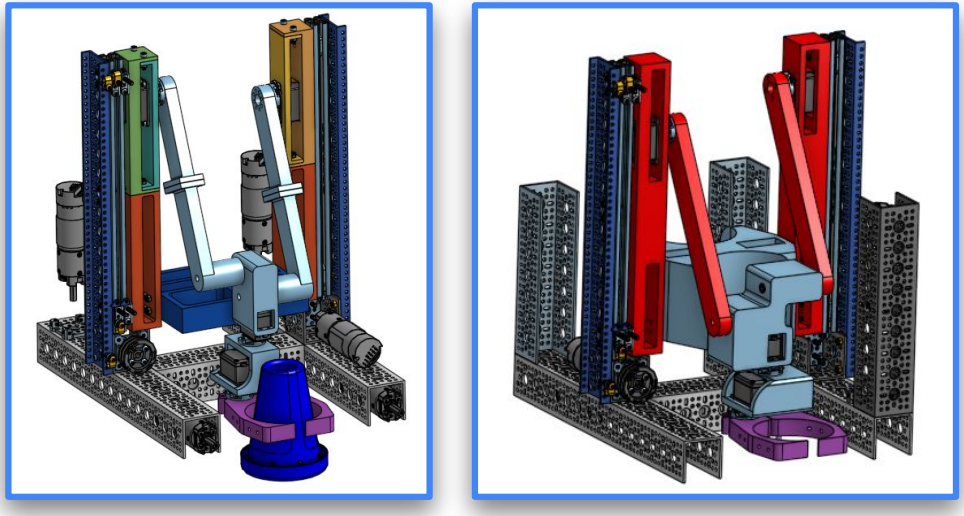


Due to being a 4 bar linkage the gripper will always be facing the same direction. In order to compensate for this a servo is attached to the gripper to allow it to rotate 180 degrees to the other direction.

FOURTH ROBOT (UNIMPLEMENTED) - CONTINUED

Pros: The robot during autonomous can drive up to the cone stack and place cones to the junction behind it constantly while the robot remains stationary, greatly increasing our points during autonomous. Additionally, this new design would make the driver's life easier as they don't have to rotate the robot as much.

Cons: Unfortunately this new design was very ambitious given how far we were into the season. As a result it was never actually implemented, although we plan to continue construction of it past the state competition.



Previous versions of the design had two major issues. The first design (left) had too many individual parts, making assembly unnecessarily difficult. While the second design (right) solved this issue by consolidating parts, but due to a miscalculation while creating the parts in OnShape neither robot extended high enough to reach the high junction.

BEACON DESIGNS

Our original beacon was a simple cardboard tube, but due to only having 2 openings if the tube were to tip over capping would be impossible.

Via the cube design the beacon had 6 openings instead of 2, allowing the driver to cap even if it's dropped. But due to there being little surface area for the gripper to hold onto, it was hard for our driver to find the right angle to grab from.

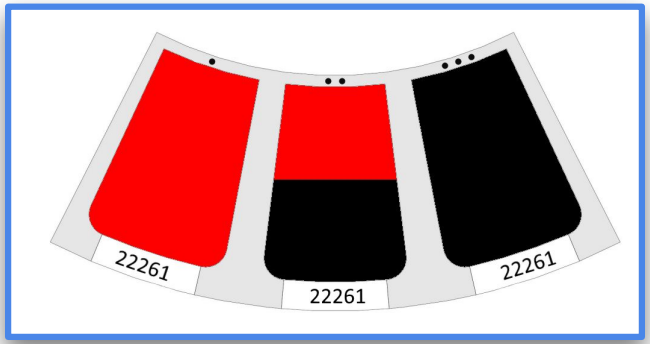
This current design fits on top of a cone, allowing for both a cone and a beacon to be scored all at once.

PROGRAMMING - AUTONOMOUS

Our Autonomous code utilizes motors with encoders and calibration systems to keep the robot constantly on-track and aware of its position on the field.

- Our movement methods that consolidate our code also contain while loops which keep the robot continuously correcting itself as it moves. It reads the current angle of the robot in relation to its starting position as provided by the IMU, then marginally speeds up and slows down certain motors to compensate for the motors' inaccuracy and stay facing the angle it's meant to face.
- We also have created a calibrate() method which further adjusts the robot to the correct angle and ensures a functional autonomous program.

We use the OpenCV Pipeline to average out the colors within two boxes in the camera view that correspond to the top and bottom half of our signal cone, then compare the two colors and determine which parking position corresponds to their layout. At the end of the program, it will park accordingly.




```
class ColorRead extends OpenCvPipeline {
    Mat mat = new Mat();
    Mat upperMat;
    Mat lowerMat;
    Mat matEnv;
    @Override
    public Mat processFrame(Mat input) {
        Mat workingMatrix = new Mat();
        input.copyTo(workingMatrix);

        if (workingMatrix.empty()) {
            return input;
        }

        Rect upperRect = new Rect(655, 450, 60, 60);
        Rect lowerRect = new Rect(645, 550, 60, 60);

        upperMat = workingMatrix.submat(upperRect);
        lowerMat = workingMatrix.submat(lowerRect);
        matEnv = workingMatrix.submat(new Rect(300, 400, 600, 300));

        Imgproc.rectangle(workingMatrix, upperRect, new Scalar(255, 0, 255));
        Imgproc.rectangle(workingMatrix, lowerRect, new Scalar(0, 255, 0));
        //Imgproc.rectangle(workingMatrix, new Rect(300, 400, 600, 300), new

        Imgproc.cvtColor(input, mat, Imgproc.COLOR_RGBA2RGB);
        Imgproc.cvtColor(mat, mat, Imgproc.COLOR_RGB2HSV);

        // -- AVERAGE --
        upperHue = Math.round(Core.mean(upperMat).val[0]);
        lowerHue = Math.round(Core.mean(lowerMat).val[0]);
```

PROGRAMMING - TELEOP

```
            } else {
                /*After setting slide to a position, continue running the
                linear slide to that same position in order to mitigate
                drooping*/
                switch (slidePos) {
                    case "HIGH":
                        slideMotor.setTargetPosition(2100);
                        slideMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
                        slideMotor.setPower(.5);
                        break;
                    case "MIDDLE":
                        slideMotor.setTargetPosition(950);
                        slideMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
                        slideMotor.setPower(.5);
                        break;
                    case "LOW":
                    case "GROUND":
                        slideMotor.setTargetPosition(0);
                        slideMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
                        slideMotor.setPower(.5);
                        break;
                }
            }
        }

    }
};
```

Our TeleOp program contains basic movement functionality as well as quality-of-life additions that improve the speed and comfort of controlling the robot.

- Our linear slide, for example, is equipped with an encoder with preset positions for it to run to, allowing for our arm to reach the perfect height with the click of a single button.
- Our program utilizes two distance sensors which triangulate the position of a junction and calibrate the robot to it. Our driver just drives up to a junction, clicks the Y button, and we're perfectly aligned to drop the cone.