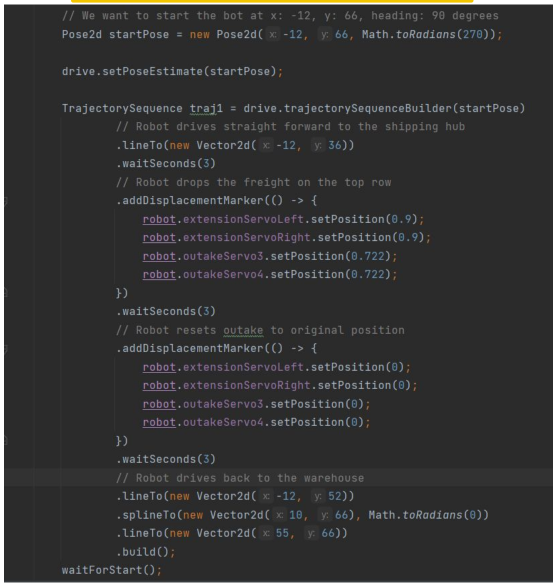
Autonomous

Our Autonomous Paths

Path One Code



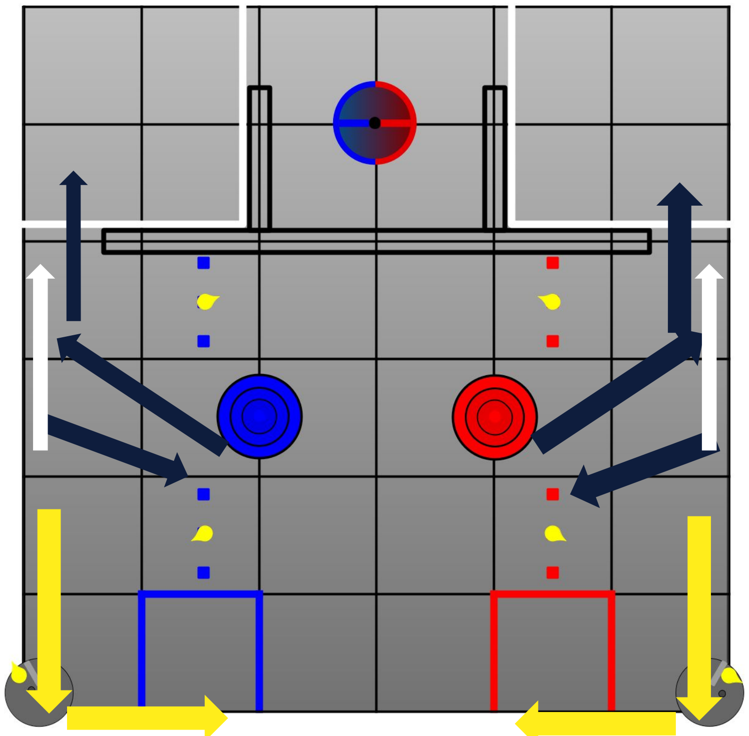
```
// We want to start the bot at x: -12, y: 66, heading: 90 degrees
Pose2d startPose = new Pose2d( x: -12,  y: 66, Math.toRadians(270));

drive.setPoseEstimate(startPose);

TrajectorySequence traj1 = drive.trajectorySequenceBuilder(startPose)
        // Robot drives straight forward to the shipping hub
        .lineTo(new Vector2d( x: -12,  y: 36))
        .waitSeconds(3)
        // Robot drops the freight on the top row
        .addDisplacementMarker(() -> {
            robot.extensionServoLeft.setPosition(0.9);
            robot.extensionServoRight.setPosition(0.9);
            robot.outakeServo3.setPosition(0.722);
            robot.outakeServo4.setPosition(0.722);
        })
        .waitSeconds(3)
        // Robot resets outake to original position
        .addDisplacementMarker(() -> {
            robot.extensionServoLeft.setPosition(0);
            robot.extensionServoRight.setPosition(0);
            robot.outakeServo3.setPosition(0);
            robot.outakeServo4.setPosition(0);
        })
        .waitSeconds(3)
        // Robot drives back to the warehouse
        .lineTo(new Vector2d( x: -12,  y: 52))
        .splineTo(new Vector2d( x: 10,  y: 66), Math.toRadians(0))
        .lineTo(new Vector2d( x: 55,  y: 66))
        .build();
waitForStart();
```

We programmed **three** main autonomous paths. These paths utilize **displacement**, line to, **trajectories**, and **waitsecond methods**. Our first path goes to the shipping hub which **drops the freight off** on the top level. The path also **returns** to the shipping container to maximize points in this autonomous path. Our second path **goes straight** to the duck wheel, **spins** the duck wheel, and then **parks** in the alliance storage unit

Our last path drives directly into the warehouse, which guarantees us some points while risking **no penalties**. It was important for us to program **multiple paths**. If one of our paths interfered with an alliances shipping hub, we could use a **separate path**. This allows us to **maximize** our potential points in the competitions.



■ First Path

□ Third Path

🟨 Second Path

Controls

To keep code organized, we used **GitHub**, a version-control repository, for programmers to share and edit code. It was especially useful in helping us work **virtually** during the pandemic and keeping others **safe**

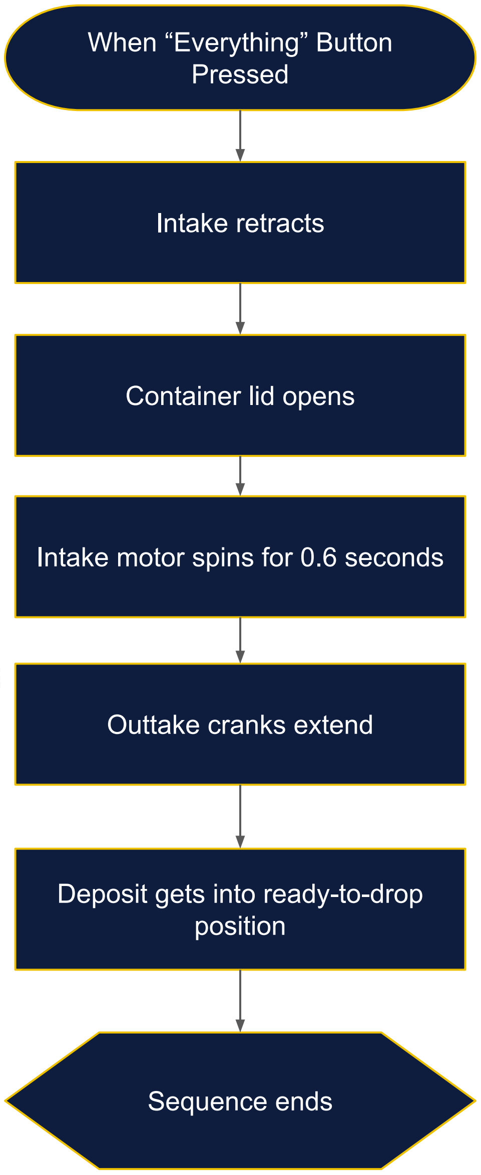
Automation in TeleOP

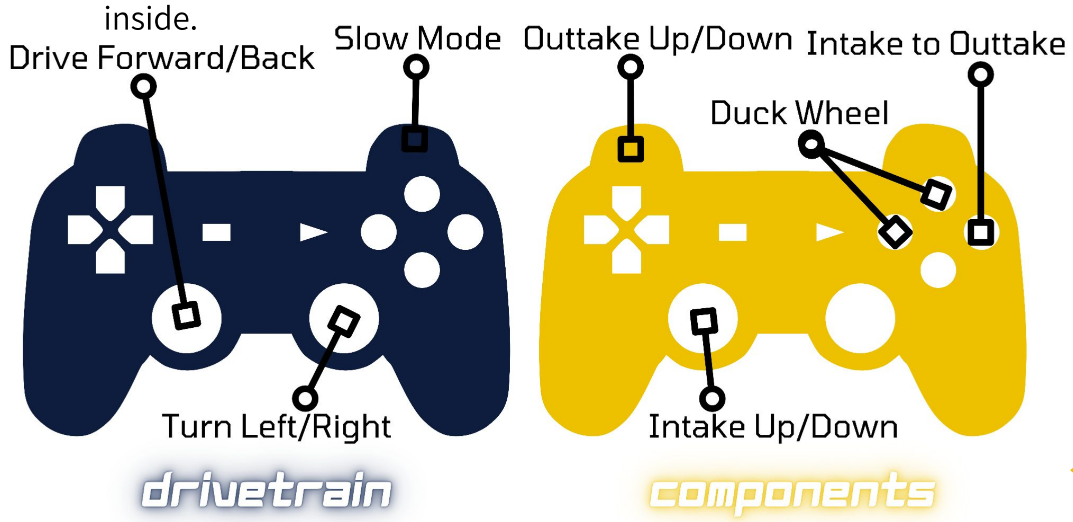
Freight Transport Sequence - *One Button to Rule Them All*

We implemented an **"*everything button*"** that when pressed, it performs a series of tasks that **automatically** transports the freight into the ready-to-outtake position so that our driver doesn't have to laboriously initiate the movements **manually**. This also eliminates **90%** of the human error.

Passive Freight Retaining System

When going over the barriers, the freight would always **fall** out due to the shake, so we implemented a **passive** method that activates whenever the robot goes over the barriers. When **IMU,** a gyroid, detects large **fluctuations**, it knows that it is going over the barrier, and **automatically** closes the container lid and slowly rotates intake motor to push the freight back inside.


When "Everything" Button Pressed
Intake retracts
Container lid opens
Intake motor spins for 0.6 seconds
Outtake cranks extend
Deposit gets into ready-to-drop position
Sequence ends



Our Sponsors

RELI Group

MSBR

Maryland Space Business Roundtable

DEKA

FEI Systems

SOLIDWORKS

FIRST®

SendCutSend

Qualcomm

Thank You Sponsors!