



FTC WI.R.E.S Software Platform for Centerstage (2023-24 season)

Instructions document

Released on 10/26/2023, Update on 11/13/2023

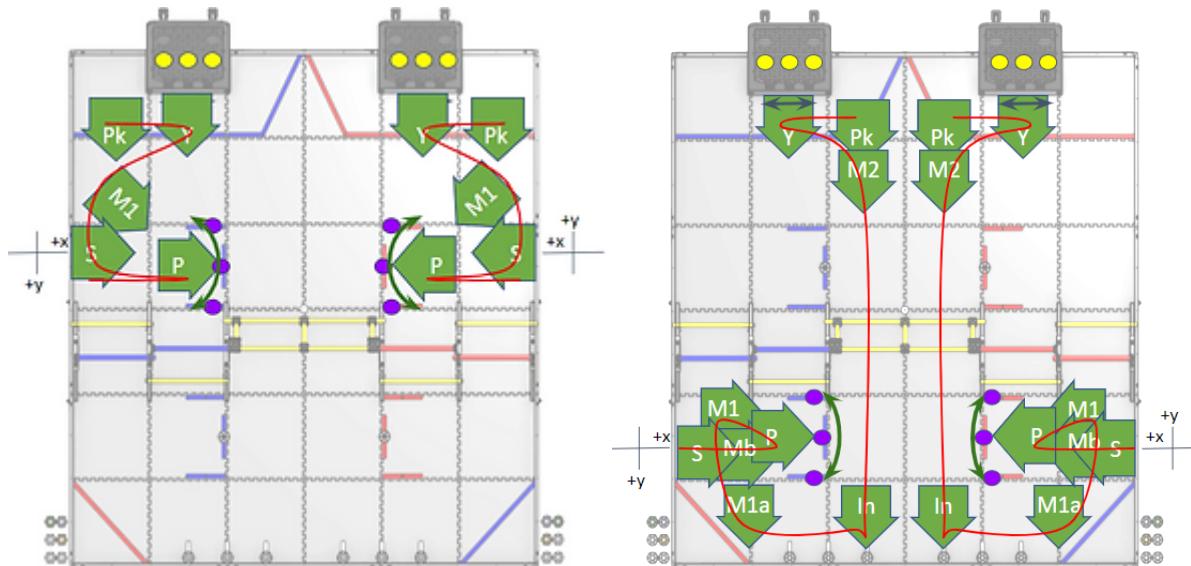
For more information, check
<https://www.ftcwires.org/softwareplatform>

or contact : ftcwires@gmail.com

ftcwiress Software Platform for easy Autonomous coding - designed for Rookie teams to have a good Autonomous mode at their first qualifier itself!

FTC WIRES Software Platform for CENTERSTAGE with 9.0.1 and RR 1.8 is now released

Integrates New Roadrunner 1.8 for motion planning , Vision library for Pixel detection, Sample code for autonomous path.



Intention: This platform is targeted to be used by rookie teams or teams who are learning autonomous programming. The aim is have all teams in Wisconsin have a basic autonomous mode working before their qualifiers.

Why: During the FTC WIRES survey in 2021-22 season, it was observed that many of the rookie teams and newer teams did not have a working autonomous mode in the early qualifiers. This was a demotivator for the teams as well as their alliance partners. This platform should ease the process of building a good and working autonomous mode in 1-2 days. We released the first FTC WIRES Software Platform in 2022-23 Powerplay season. Most Wisconsin teams had an autonomous mode in the year. The platform was subscribed by more than 70 teams worldwide too. Based on request, we are releasing the new version of FTCWIRES Software Platform for Centerstage 2023-24 Season

Acknowledgement: [Acmerobotics](#) and Ryan Brott for providing the Roadrunner library, Noah Bres for Learnroadrunner.com and [Team Hazmat 13201](#) who created the FTC Wires platform

Note: The document may be 35 pages. But most of it is pictures and explanations!. The actual code change is not more than 25 lines, and effort could be completed in 2 hrs.

Table of Contents:

Table of Contents:	3
What does the platform contain :	4
How does the FTCWires Autonomous Mode work :	6
Instructions for Setup:	8
Configure and Tune the robot	10
Step 1: Motor names in MecanumDrive Class in MecanumDrive.java	10
Step 2: IMU setup in MecanumDrive Class in MecanumDrive.java	11
Step 3: Localizer selection in MecanumDrive.java	12
Step 4 : Set the motor movement direction forward	13
Step 5 : Forward Push Test	15
Step 6 : Lateral Push Test (mecanum drive encoders only).	16
Step 7 : ForwardRampLogger (Only for dead wheel encoders)	17
Step 8: Lateral Ramp Logger (Only for dead wheel encoders)	19
Step 9 : Angular Ramp Logger for Drive Encoders.	19
Step 10 : Set Track Width	22
Step 11: Angular Ramp Logger for Dead Wheels Encoders.	23
Step 12: Manual Feedforward Tuner.	24
Step 13: Manual Feedback Tuner.	27
Step 14: LocalizationTest	30
Step 15 : SplineTest.	31
Autonomous Op Mode Code	32
TeleOp Mode Code	35

Update 11/13/2023

Step 11.1 introduced - Only for Dead Wheel Encoders, Need to update tick count for parallel and perp encoder position.

What does the platform contain :

- The platform is a fork from Road-runner-Quickstart based off of FtcRobotController SDK 9.0.1 released by FIRST and can accessed at <https://github.com/ftcwires/centerstage-road-runner>
- Roadrunner (Rev 1.8) is the newly released version of the motion planning library developed by Acmerobotics (Ryan Brott) . Designed primarily for autonomous robotic movement, it allows for complex path following and generation while maintaining control of velocity and acceleration. This enables bots to have more accurate and advanced path following capabilities. We are going to use Drive Encoder based odometry. (Detailed information on this is available at <https://rr.brott.dev/> and <https://github.com/acmerobotics/road-runner-quickstart> but the idea here is to help teams who find those pages overwhelming, so you don't need to look!)
- It also includes integration of the Vision Portal TensorFlow detection of the white pixel (default one, not of your customized team game element), to find the spikemark to drop purple pixel and the location to drop Yellow pixel on Backdrop in autonomous mode. This code is derived from the `ConceptTensorFlowObjectDetection.java` provided as an example in the FTC sdk.
- Using these, an example Autonomous mode for Centerstage is implemented. You could modify this easily to develop your own autonomous mode. This also includes a simple tuning process for Roadrunner, as well as easy way to program positions for autonomous modes based on robot centric coordinate system.
- This also includes a sample version of TeleOp with motion management based on Roadrunner.

Assumptions :

- You have a robot that uses mecanum wheels with the motor encoders connected for odometry (80% of FTC teams use this). We call this Drive Encoder based odometry
- You should also have a webcam connected and positioned in a way to see the pixel on the spike mark
- The robot design assumes pick up of pixels from the front of the robot and drop of pixels on backdrop from back of the robot. (If you have a different orientation, all you need is to change the position coordinates)
- Your robot would need to add the code for mechanism to drop purple pixel, ability to intake pixels and ability to drop pixels on backstage or backdrop.
- You have a minimum understanding of Java and using the sdk, and coding using android studio.

If you have “No” on one of the assumptions, you could still use it :

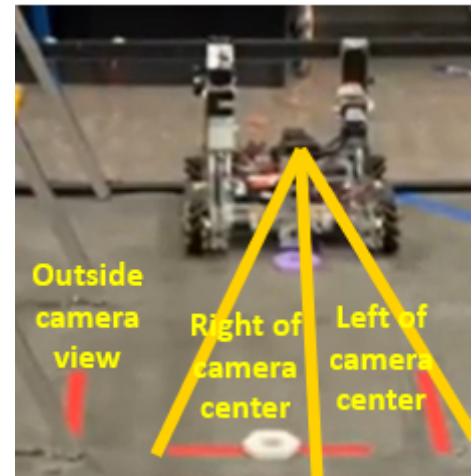
- If you are using block programming, you could always finish your teleop mode in block programming for all your non-drive systems and then transition to this platform, get to the java code and add to this.
- If you are using tank drive, you still could use this, but have to figure out how roadrunner works with tank drive from the tuning docs of Roadrunner
- If you want to use dead wheel encoders, you can still use this. Just need to make the localizer changes and do tuning (based on roadrunner docs). We have not verified if all the instructions are accurate for this.

Disclaimer :

- This is a basic autonomous mode - better than a crude one, but it is certainly not the best.
- This only uses minimal roadrunner capability in terms of motion profiles possible. Roadrunner has several additional motion profiles to use, ability to time actions in parallel, sequential, stop and start, etc. which is not used here.
- Roadrunner also provides the ability to visualize motion on a digital dashboard. This version of the program won't support it, since the coordinate system assumed (for simplicity) is based on the starting position of the robot. Dashboard requires a field centric coordinate system.
- If you feel like you will miss the fun of discovering how to code the autonomous mode the hard way - don't look, this is just to make the journey easy for those who want to start with an example.

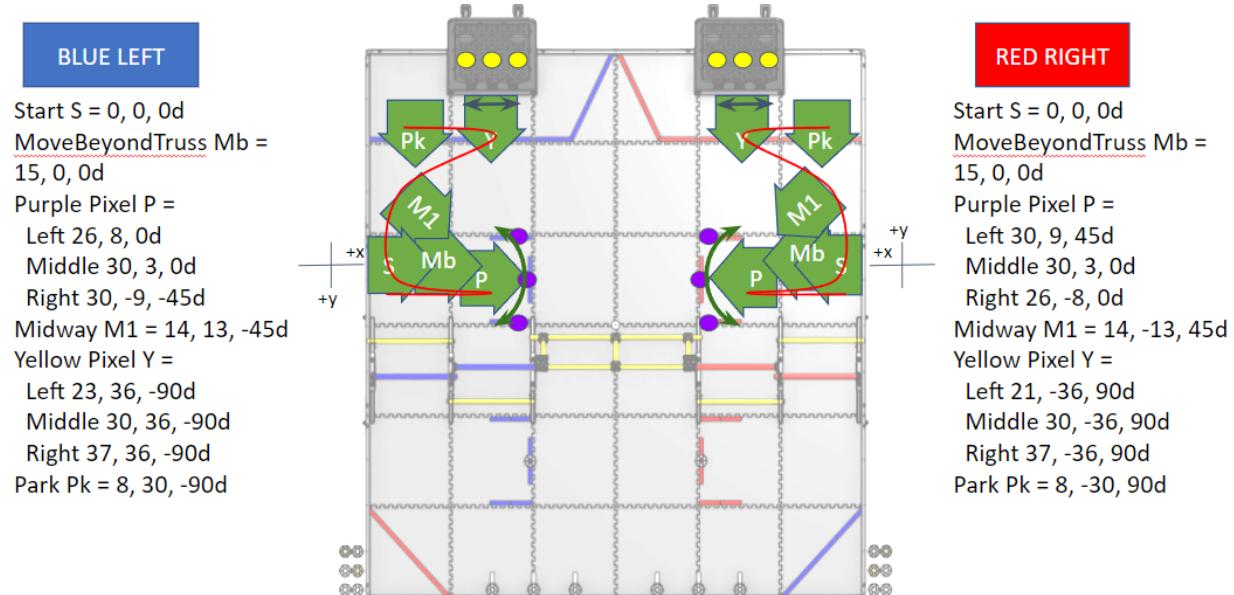
How does the FTCWires Autonomous Mode work :

The sample code provided gives an implementation of the autonomous mode based on instructions in the Game Manual 2. There are 4 modes to select - based on the starting location of the robot (Red Left, Red Right, Blue Left, Blue Right). The starting point of the robot is assumed to be on the starting tile, and along the edge farther from the truss legs. You should also have a webcam connected and positioned in a way to see the middle spike mark and the spike mark away from the truss (and ideally nothing else). We assumed the camera to be in the center of the robot.



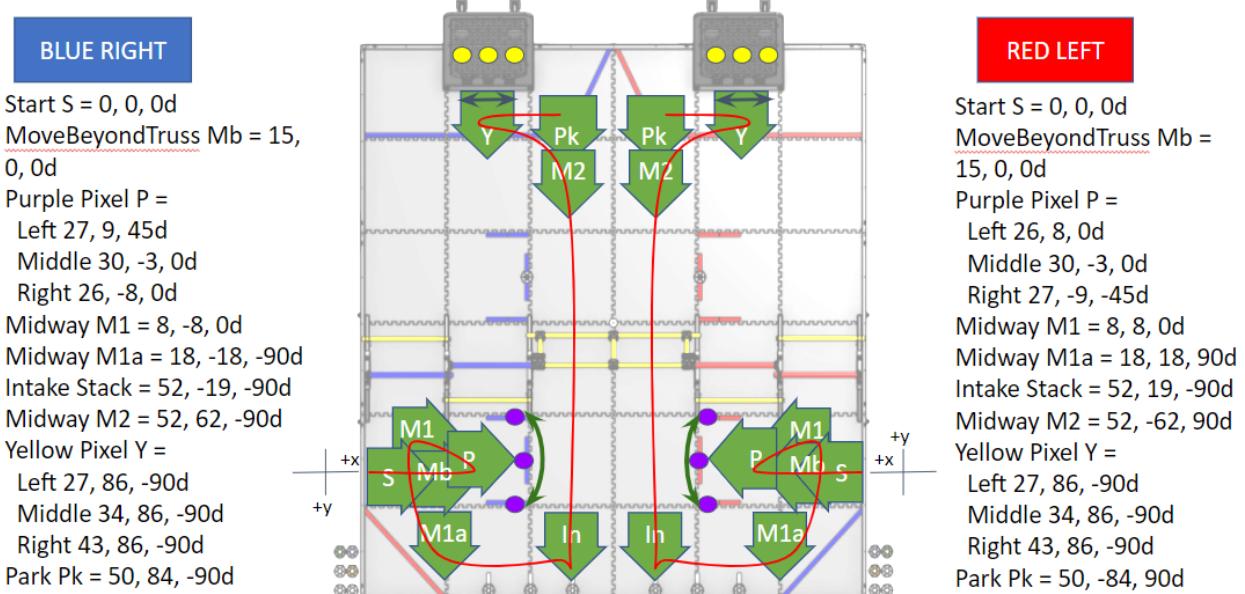
Blue Left and Red Right Autonomous mode

1. Robot starts in the position S marked in the picture (aligned to tile edge farther from the truss)
2. On Init, The robot vision is started and it identifies the white pixel in the spike mark (Left, Middle or Right).
3. On Start, the robot first moves to position Mb (to avoid hitting the truss legs) and then to the spike mark detected by vision (position P). The code for dropping purple pixel needs to be executed at this point.
4. Robot then moves back to M1 (to avoid the purple pixel) and moves to position Y (based on the spike mark detected). The code for dropping Yellow pixel on the back board needs to be executed at this point.
5. Robot then moves to parking position Pk.



Blue Right and Red Left Autonomous mode

1. Robot starts in the position S marked in the picture (aligned to tile edge farther from the truss)
2. On Init, The robot vision is started and it identifies the white pixel in the spike mark (Left, Middle or Right).
3. On Start, the robot first moves to position Mb (to avoid hitting the truss legs) and then to the spike mark detected by vision (position P). The code for dropping purple pixel needs to be executed at this point.
4. Robot then moves back to M1 and M1a (to avoid the purple pixel) and moves to position in front of the pixel stack. Code for picking a pixel needs to be added here.
5. Robot then moves to position M2 (through the central path. If the stage door needs to be opened for the robot to pass, code needs to be added for it.
6. Robot then moves to position Y (based on the spike mark detected). The code for dropping Yellow pixel on the back board needs to be executed at this point.
7. Robot then moves to parking position Pk.

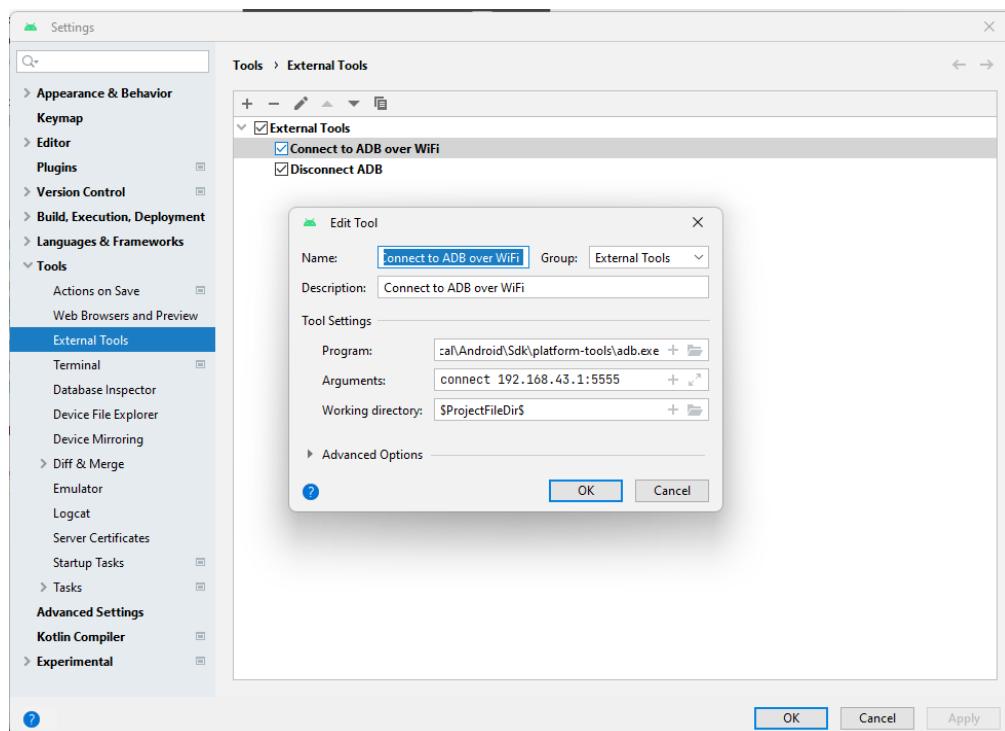


Instructions for Setup:

1. The code for FTC WIRES Software platform is at :
 - o <https://github.com/ftcwires/centerstage-road-runner>
 - o Go to the page. If you don't have a github account, create one and login to see this.
2. Click on the green Code button and Download zip file and uncompress on your computer.
 - o If you are comfortable with git, you should create a new fork of the project <https://github.com/ftcwires/centerstage-road-runner.git> . Will help in keeping sync in case FtcRobotController project or roadrunner is updated.
3. Create a new project on Android Studio with this downloaded code (or clone the git project).
 - o Ensure you are connected to the internet and click on File -> Sync Project with Gradle Files.
 - o Follow any instructions that Android suggests to upgrade Android versions.
 - o Click on Build -> Make Project and see that it completes successfully.
 - o Congratulations you have the set up ready
 - o If you have trouble setting up AndroidStudio, please refer to https://ftc-docs.firstinspires.org/en/latest/programming_resources/android_studio/java/Android-Studio-Tutorial.html. If you are still having trouble, reach out to us at ftcwires@gmail.com

Connect the Rev Control Hub over wifi :

To create a wireless connection from Android Studio to the robot, Go to File-> Settings and on the popup window. Go to Tools->External Tools. Click on + and create a tool code as in picture below:



Name : Connect to ADB over WiFi

Description : Connect to ADB over WiFi

Tool Settings:

Programs:

C:\Users\<your_user_folder>\AppData\Local\Android\Sdk\platform-tools
\adb.exe

Arguments : connect 192.168.43.1:5555

Working Directory : \$ProjectFileDir\$

Once you click Click on Tools -> External Tools and you will see a button for Connect to ADB over WiFi. Connect your laptop to the WiFi network of your Rev Control Hub. It is best to have a USB wifi adapter on your laptop. This way, your laptop's main wifi can be connected to the internet, while the USB wifi can connect to your Rev Control Hub.

Download Code to Rev Control Hub:

Once connected, You will see the Rev Controller showing up as a connected device in the top bar of Android Studio. Click on Run (Green Arrow) button to download the built code to the Rev Control Hub.

You should be able to see the Opmodes FTC Wires Autonomous Mode **and** FTC Wires TeleOp, along with a few other Opmodes.

Congratulations! Your setup is complete!

Configure and Tune the robot

We assumed a robot design that has 4 mecanum wheels, and has a mechanism to pick up pixels from the front and drop the pixels on the backdrop from the back of the robot. (If you have a different orientation, all you need is to change the position coordinates). We also assumed the robot has a webcam in the front center and at a height of 7-8 inches from the ground, pointed in a way to “see” the spike mark from the starting position of the robot.

Create a configuration on the Driver Hub with the 4 motors named as leftFront, leftBack, rightBack, rightFront. The webcam should be configured as Webcam 1.

Tuning the Robot

Each robot is different, and Road Runner needs to be calibrated to your robot. The process consists of running several op modes and either adjusting parameters manually depending on the response or automatically fitting parameters to data collected. The entire activity shouldn't take more than two hours.

Step 1: Motor names in MecanumDrive Class in MecanumDrive.java

- Open MecanumDrive.java on Android Studio under teamcode package:

```
188     }
189
190     public MecanumDrive(HardwareMap hardwareMap, Pose2d pose) {
191         this.pose = pose;
192
193         LynxFirmware.throwIfModulesAreOutdated(hardwareMap);
194
195         for (LynxModule module : hardwareMap.getAll(LynxModule.class)) {
196             module.setBulkCachingMode(LynxModule.BulkCachingMode.AUTO);
197         }
198
199         //TODO Step 1 Drive Classes : get basic hardware configured. Update motor names to what is used
200         leftFront = hardwareMap.get(DcMotorEx.class, "leftFront");
201         leftBack = hardwareMap.get(DcMotorEx.class, "LeftBack");
202         rightBack = hardwareMap.get(DcMotorEx.class, "rightBack");
203         rightFront = hardwareMap.get(DcMotorEx.class, "rightFront");
204
205         //TODO End Step 1
206
207         //TODO Step 4.1 Run MecanumDirectionDebugger Tuning OpMode to set motor direction correctly
208         //Uncomment the lines for which the motorDirection need to be reversed to ensure all motors
209         //leftFront.setDirection(DcMotorEx.Direction.REVERSE);
210         //leftBack.setDirection(DcMotorEx.Direction.REVERSE);
211         rightFront.setDirection(DcMotorEx.Direction.REVERSE);
212         rightBack.setDirection(DcMotorEx.Direction.REVERSE);
213
214         //TODO Make the same update in DriveLocalizer() function. Search for Step 4.2
215
216         leftFront.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
217         leftBack.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
```

- Search for "TODO Step 1". Make sure the motor names listed match the motor names in your robot configuration. "leftFront", "leftBack", "rightBack", and "rightFront"
 - Remove the TODO word in the comment as you complete the step.

Step 2: IMU setup in MecanumDrive Class in MecanumDrive.java

- Search for “TODO Step 2” in MecanumDrive.java.
- It is assumed that your IMU is named “imu” Update direction of IMU by updating orientation of Driver Hub below
- Update the RevHubOrientationOnRobot.LogoFacingDirection.UP parameter based on how the Rev Control Hub is oriented in your robot. Change to UP / DOWN / LEFT / RIGHT / FORWARD / BACKWARD as in robot
- Update the RevHubOrientationOnRobot.UsbFacingDirection.FORWARD parameter based on how the USB port of the Rev Control Hub is oriented in your robot. Change to UP / DOWN / LEFT / RIGHT / FORWARD / BACKWARD as in robot

```
leftFront.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
leftBack.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
rightBack.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
rightFront.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

imu = hardwareMap.get(IMU.class, deviceName: "imu");
//TODO Step 2 : Update direction of IMU by updating orientation of Driver Hub below
IMU.Parameters parameters = new IMU.Parameters(new RevHubOrientationOnRobot(
    RevHubOrientationOnRobot.LogoFacingDirection.UP, // Change to UP / DOWN / LEFT / RIGHT
    RevHubOrientationOnRobot.UsbFacingDirection.FORWARD)); //Change to UP / DOWN / LEFT / RIGHT

imu.initialize(parameters);
//TODO End Step 2

voltageSensor = hardwareMap.voltageSensor.iterator().next();

//TODO Step 3: Specify how the robot should track its position
//Comment this line if NOT using Drive Encoder localization
localizer = new DriveLocalizer();
//Uncomment next line if using Two Dead Wheel Localizer and also check TwoDeadWheelLocalizer
//localizer = new TwoDeadWheelLocalizer(hardwareMap, imu, PARAMS.inPerTick)

//Uncomment next line if using Three Dead Wheel Localizer and also check ThreeDeadWheelLocalizer
//localizer = new ThreeDeadWheelLocalizer(hardwareMap, PARAMS.inPerTick)
//TODO End Step 3

FlightRecorder.write( ch: "MECANUM_PARAMS", PARAMS);

}

public void setDrivePowers(DriveVelocity2d powers) {
}
```

Step 3: Localizer selection in MecanumDrive.java

- Search for "TODO Step 3" in MecanumDrive.java.. Specify how the robot should track its position. There are a few built-in localizers:
- Drive encoders: This is the default. The IMU will also be used on the mecanum to get better heading. (`localizer = new DriveLocalizer();`)
- Two (dead) wheel: Change the right-hand-side of `localizer = to new TwoDeadWheelLocalizer(hardwareMap, imu, PARAMS.inPerTick)`. The code expects the parallel, forward-pointing encoder to be named "par" and the perpendicular one to be named "perp".
- Three (dead) wheel: Change the right-hand-side of `localizer = to new ThreeDeadWheelLocalizer(hardwareMap, PARAMS.inPerTick)`. The code expects the two parallel encoders to be named "par0" and "par1" and the perpendicular one to be named "perp".
- If changing from default, make the change in also In tuning/TuningOpModes.java
- Download the code to the Rev Control Hub

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help centerstage-road-runner - MecanumDrive.java [centerstage-road-runner.TeamCode.main]
org firstinspires ftc teamcode MecanumDrive MecanumDrive TeamCode build.common.gradle build.dependencies.gradle build.gradle (TeamCode) ConceptTensorFlow ...
Android FtcRobotController Project ResourceManager ...
TeamCode manifests ...
java org.firstinspires.ftc.teamcode ...
tuning LocalizationTest ManualFeedbackTuner SplineTest TuningOpModes ConceptTensorFlowObjectDetection FTCWiresAutonomous FTCWiresTeleOpMode Localizer MecanumDrive PoseMessage TankDrive ThreeDeadWheelLocalizer TwoDeadWheelLocalizer ...
java (generated) jniLibs res res (generated) Gradle Scripts ...
Step 2
224 RevHubOrientationOnRobot.LogoFacingDirection.UP, // Change to UP / A 1 2 ✕ 26 ^ y
225 RevHubOrientationOnRobot.UsbFacingDirection.FORWARD); //Change to UP / DOWN / LEFT
226 imu.initialize(parameters);
227 //TODO End Step 2
228
229 voltageSensor = hardwareMap.voltageSensor.iterator().next();
230
231 //TODO Step 3: Specify how the robot should track its position
232 //Comment this line if NOT using Drive Encoder localization
233 localizer = new DriveLocalizer();
234 //Uncomment next line if using Two Dead Wheel Localizer and also check TwoDeadWheelLocalizer
235 //localizer = new TwoDeadWheelLocalizer(hardwareMap, imu, PARAMS.inPerTick)
236
237 //Uncomment next line if using Three Dead Wheel Localizer and also check ThreeDeadWheelLocalizer
238 //localizer = new ThreeDeadWheelLocalizer(hardwareMap, PARAMS.inPerTick)
239 //TODO End Step 3
240
241 FlightRecorder.write( ch: "MECANUM_PARAMS", PARAMS);
242
243 public void setDrivePowers(PoseVelocity2d powers) {
244     MecanumKinematics.WheelVelocities<Time> wheelVels = new MecanumKinematics( trackWidth: 1).invers
245         PoseVelocity2dDual.constant(powers, n: 1);
246
247     double maxPowerMag = 1;
248     for (DualNum<Time> power : wheelVels.all()) {
249         maxPowerMag = Math.max(maxPowerMag, power.value());
250     }
251 }
```

The screenshot shows the Eclipse IDE interface with the MecanumDrive.java file open. The code editor has syntax highlighting and a search bar at the top. The left sidebar shows the project structure with various FTC components like FtcRobotController, TeamCode, and Java files for localization and drive modes. The main code area contains several TODO comments and localizer selection logic. A yellow circle highlights the line `localizer = new DriveLocalizer();`, indicating it's the current point of interest. The status bar at the bottom provides build information.

Step 4 : Set the motor movement direction forward

- From now on you would be using the robot to calibrate. Ensure the battery voltage is above 12.5V through the rest of the process.
 - On the Driver Hub, Start the Opmode : `MecanumDirectionDebugger` to make sure all the directions are correct. The op mode uses the following button mappings:

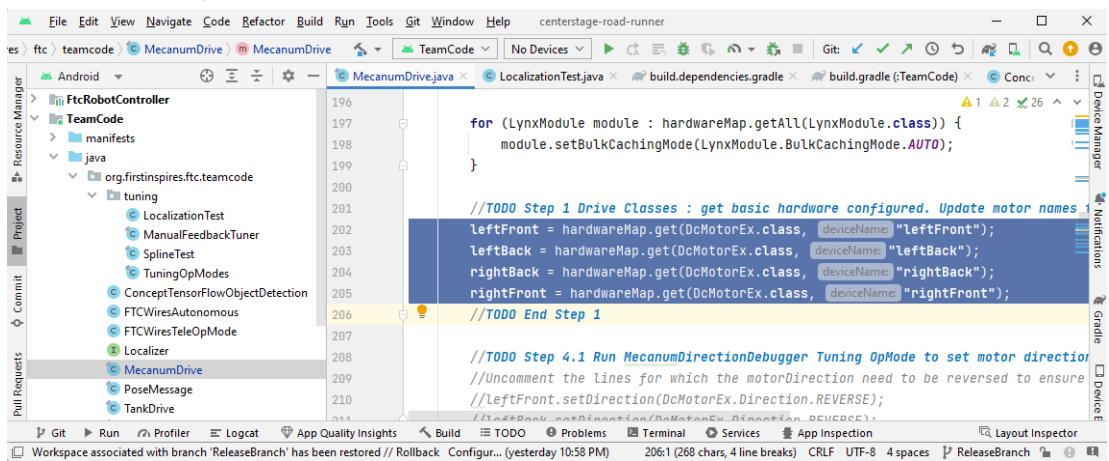
```

/***
 * Xbox/PS4 Button - Motor
 * X / □ - Front Left
 * Y / Δ - Front Right
 * B / O - Rear Right
 * A / X - Rear Left
 *
 * The buttons are mapped to match the wheels spatially if you
 * were to rotate the gamepad 45deg°. x/square is the front left
 * and each button corresponds to the wheel as you go clockwise
 *
 *          / _____ \
 *   -----.-'   _`-'..+
 *          /   ( Y )   \
 *          |   ( X )   ( B ) |
 *          .   ( A )   /|   Wheel   \   Front Right
 *   .-'   .-'-_____.-'   .'   (x/□)   \   Wheel
 *   |       |           |   Rear Left   \   Rear Right
 *   .___.'   .           |   Wheel   \   Wheel
 *   .       /           (A/X)   \   (B/O)
 *   \.     .
 *   \_____/
 */

```

- Press the button corresponding to each motor and check that the motors spin in the forward direction. And if you're using drive encoders, the ticks recorded should increase in a positive direction.
 - Search for “TODO Step 4.1” in MecanumDrive.java. The default code has the right side motors reversed lines as follows in MecanumDrive() method:

```
rightFront.setDirection(DcMotorEx.Direction.REVERSE);  
rightBack.setDirection(DcMotorEx.Direction.REVERSE);
```
 - If your robot has different motor orientation, any of the wheels are rotating backwards, comment / uncomment the corresponding line and download the code to the Rev Control Hub. And test again.



- Once all the motors are driving forward, note which motors are reversed. Search for “TODO Step 4.2” in MecanumDrive.java. And make the similar changes in the DriveLocalizer() method.

```

public class DriveLocalizer implements Localizer {
    public final Encoder leftFront, leftBack, rightBack, rightFront;

    private int lastLeftFrontPos, lastLeftBackPos, lastRightBackPos, lastRightFrontPos;
    private Rotation2d lastHeading;

    public DriveLocalizer() {
        leftFront = new OverflowEncoder(new RawEncoder(MecanumDrive.this.leftFront));
        leftBack = new OverflowEncoder(new RawEncoder(MecanumDrive.this.leftBack));
        rightBack = new OverflowEncoder(new RawEncoder(MecanumDrive.this.rightBack));
        rightFront = new OverflowEncoder(new RawEncoder(MecanumDrive.this.rightFront));

        //TODO Step 4.2 Run MecanumDirectionDebugger Tuning OpMode to set motor direction correctly
        //Uncomment the lines for which the motorDirection need to be reversed to ensure all motors
        //LeftFront.setDirection(DcMotorEx.Direction.REVERSE);
        //LeftBack.setDirection(DcMotorEx.Direction.REVERSE);
        rightBack.setDirection(DcMotorEx.Direction.REVERSE);
        rightFront.setDirection(DcMotorEx.Direction.REVERSE);

        //TODO End Step 4.2
    }

    lastLeftFrontPos = leftFront.getPositionAndVelocity().position;
    lastLeftBackPos = leftBack.getPositionAndVelocity().position;
    lastRightBackPos = rightBack.getPositionAndVelocity().position;
    lastRightFrontPos = rightFront.getPositionAndVelocity().position;

    lastHeading = Rotation2d.exp(imu.getRobotYawPitchRollAngles().getYaw(AngleUnit.RADIANS));
}

@Override
public Twist2dDual<Time> update() {
}

```

- Download the code to the Rev Control Hub

Step 5 : Forward Push Test

- The goal is to determine the distance in inches traveled per tick of the encoder `inPerTick` empirically
 - Place the robot on the tiles with plenty of room in front. Square the robot up with the grid and make note of its starting position. Run `ForwardPushTest` Opmode on the Driver Hub and then slowly push the robot straight until the end of the tiles. Record the “ticks traveled” from display on the Driver Hub before stopping the op mode.
 - Make sure the wheels don’t slip! The motors should spin freely. If the bot is too light or otherwise difficult, you can use theoretical values for ticks per revolution, gear ratio, and wheel diameter to compute `inPerTick` instead.
 - If the ticks traveled is close to zero no matter how far you push, the motors on one side are probably reversed. Make sure the previous step is finished before returning (particularly Step 4.2 motor reversal matches the one in 4.1).
 - Without moving the robot, record also the forward distance traveled on the tiles in inches. (Measure where the back of the robot was when you started to where it is now). Calculate the `inPerTick` value as the real distance traveled divided by the ticks traveled.
 - As example for expected values, in our robot with 312rpm gobilda motors, we got 5709.25 ticks for 127 inches yielding `inPerTick = 0.02224460305`
 - Search for “TODO Step 5” in `MecanumDrive.java`. Set the `inPerTick` value to the calculated value in the static PARAMS class in `Mecanumdrive` class

The screenshot shows the Android Studio interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Project Bar:** teamcode > MecanumDrive > Params > inPerTick.
- Toolbars:** TeamCode, No Devices, Git, Layout Inspector.
- Left Sidebar:** Resource Manager, Project, Commit, Full Requests, Bookmarks, Build Variants.
- Code Editor:** The file MecanumDrive.java is open. The code defines a class MecanumDrive with a static inner class Params. It includes several TODO comments related to localization parameters like inPerTick, lateralInPerTick, and trackWidthTicks. It also defines feedforward parameters kS and kV.
- Right Sidebar:** Device Manager, Notifications, Gradle, Device picker, Running Devices.
- Bottom Navigation:** Git, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, App Inspection, Layout Inspector.

```
import ...  
@Config  
public final class MecanumDrive {  
    public static class Params {  
        // drive model parameters  
        //TODO Step 5 Set value of inPerTick after running ForwardPushTest  
        //TODO Step 14 Make value of inPerTick accurate after running LocalizationTest  
        public double inPerTick = 0;  
  
        //TODO Step 6 (Only for DriveEncoder Localizer) Set value of lateralInPerTick after running LocalizationTest  
        //TODO Step 8 (Only for DeadWheel Localizer) Set value of lateralInPerTick after running LocalizationTest  
        //TODO Step 14 Make value of lateralInPerTick accurate after running LocalizationTest  
        public double lateralInPerTick = 1;  
  
        //TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after running LocalizationTest  
        //TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after running LocalizationTest  
        public double trackWidthTicks = 0;  
  
        // feedforward parameters (in tick units)  
        //TODO Step 7 (Only for DeadWheel Localizer) Set value for kS and KV after running LocalizationTest  
        //TODO Step 9 (Only for DriveEncoder Localizer) Set value for kS and KV after running LocalizationTest  
        public double kS = 0;  
        public double KV = 0;
```

- Download the code to the Rev Control Hub.

Step 6 : Lateral Push Test (mecanum drive encoders only).

- The goal is to determine `lateralInPerTick` empirically
- This routine is simply similar to `ForwardPushTest`, the difference being that it measures rightward motion instead of forward motion .
- Start the `LateralPushTest` opmode on the Driver Hub.
- Set the robot up as before with space towards the right of the robot, slowly push it right, and Record the “ticks traveled” from display on the Driver Hub before stopping the op mode.
- Measure the actual distance traveled in inches and calculate the value of `lateralInPerTick` as the measured distance divided by ticks traveled.
- Note that there would be the inevitable strafing slip that occurs, so you should expect it to be smaller than `inPerTick` by a modest amount. Dont worry about it now, this would be corrected at the end.
- As example for expected values, in our robot with 312rpm gobilda motors, we got 6101 ticks for 126 inches yielding `lateralInPerTick = 0.02065573770`
- Search for “`TODO Step 6`” in `MecanumDrive.java`. Set the `lateralInPerTick` value to the calculated value in the static `PARAMS` class in `Mecanumdrive` class

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help centerstage-road-runner
MecanumDrive > C MecanumDrive > Params > lateralInPerTick TeamCode No Devices Git ✓ ✓ ✎
Resource Manager Project Commit Pull Requests Bookmarks Build Variants
Android > FtcRobotController > TeamCode > manifests > java > org.ftcinspires.ftc.teamcode > tuning > LocalizationTest > ManualFeedbackTuner > SplineTest > TuningModes > ConceptTensorFlowObjectDetection > FTCWiresAutonomous > FTCWiresTeleOpMode > Localizer > MecanumDrive > PoseMessage > TankDrive > ThreeDeadWheelLocalizer > TwoDeadWheelLocalizer > java (generated) > jnilibs > res > res (generated) > Gradle Scripts
MecanumDrive.java
import ...
@Configuration
public final class MecanumDrive {
    public static class Params {
        // drive model parameters
        //TODO Step 5 Set value of inPerTick after running ForwardPushTest
        //TODO Step 14 Make value of inPerTick accurate after running LocalizationTest
        public double inPerTick = 0;

        //TODO Step 6 (Only for DriveEncoder Localizer) Set value of lateralInPerTick after
        //TODO Step 8 (Only for DeadWheel Localizer) Set value of lateralInPerTick after
        //TODO Step 14 Make value of lateralInPerTick accurate after running LocalizationTest
        public double lateralInPerTick = 1;

        //TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after
        //TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after
        public double trackWidthTicks = 0;

        // feedforward parameters (in tick units)
        //TODO Step 7 (Only for DeadWheel Localizer) Set value for KS and KV after running
        //TODO Step 9 (Only for DriveEncoder Localizer) Set value for KS and KV after running
        public double KS = 0;
        public double KV = 0;
    }
}
```

- Download the code to the Rev Control Hub.

Step 7 : ForwardRampLogger (Only for dead wheel encoders)

Skip to Step 9, if using Drive Encoders.

- The goal is to tune the Motor FeedForward parameters kS and kV empirically. Theory about Motor feed forward is at <https://docs.wpilib.org/en/stable/docs/software/advanced-controls/introduction/introduction-to-feedforward.html#introduction-to-dc-motor-feedforward>
- This opmode `ForwardRampLogger` slowly increases the forward power given to the robot and measures the forward velocity over time to calculate the static and velocity feedforward parameters (kS and kV, respectively). By default, the power will increase by 0.1 each second until it reaches 0.9.
- Place your robot on the tiles with as much distance in front of it as possible. Start the op mode and press stop immediately when the robot nears the edge of the tile. Don't expect anything to be displayed in telemetry. All data collected is saved to a file for further analysis.
- Once you finish, connect your computer to the RC wireless network using these instructions.
- If you have a control hub, navigate to <http://192.168.43.1:8080/tuning/forward-ramp.html> .
- If you have a RC phone, navigate to <http://192.168.49.1:8080/tuning/forward-ramp.html> .
- From now on, I'll use Control Hub URLs.
- You should see a page that looks like this:

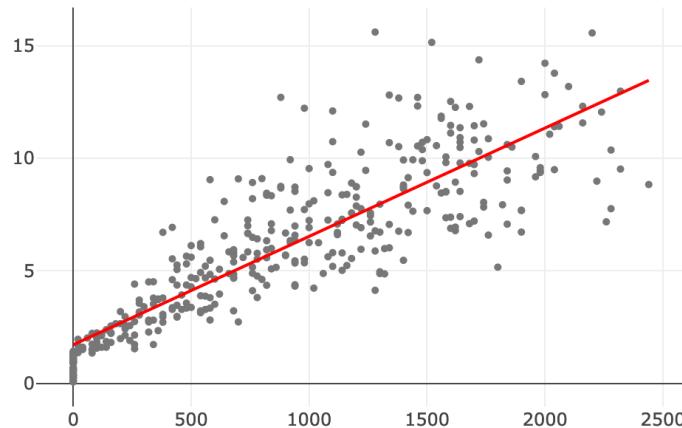
RR Drive Encoder Angular Ramp Regression

► Details

No file chosen

- Click the “Latest” button, and you should see a graph appear:

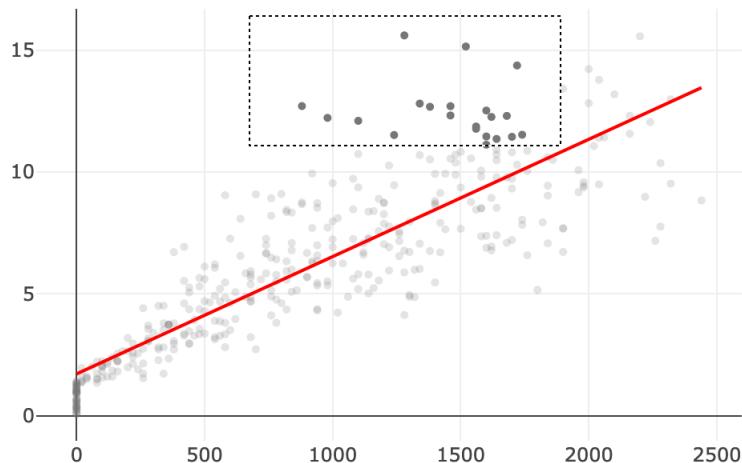
Ramp Regression



- The gray points are the measurements made by the tuner, and the red line is the line of best fit. Its from this line that the feedforward parameters will be extracted. In fact, you can see preliminary values for kS and kV already displayed above. But those estimates are limited by the presence of outliers. You can see that the red line doesn't quite fit the trend. Let's help the algorithm out by manually filtering out outlier points.

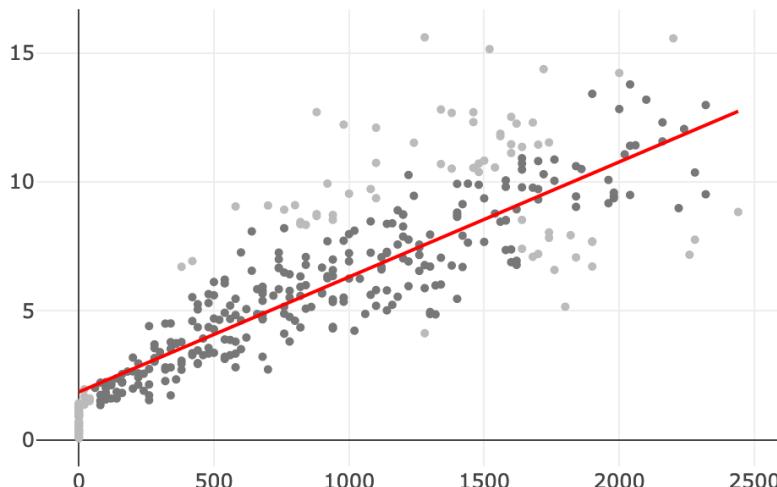
- First, click and drag to select a rectangular region of the plot:

Ramp Regression



- Then to remove those points, press the “e” key or the “exclude” button above. The selection box will disappear, and the points will turn a lighter gray. After repeating this process as many times as necessary, you should end up with a plot like this:

Ramp Regression



- Now you can copy the values above the plot into the kS and KV fields in the drive class.
- If you mess up and accidentally exclude non-outlier points, select them and press “i” or click the include button to bring them back into the calculation.
- If you have issues or the data looks weird, click the “Download” button to retrieve your data and include it when asking for help.

Step 8: Lateral Ramp Logger (Only for dead wheel encoders)

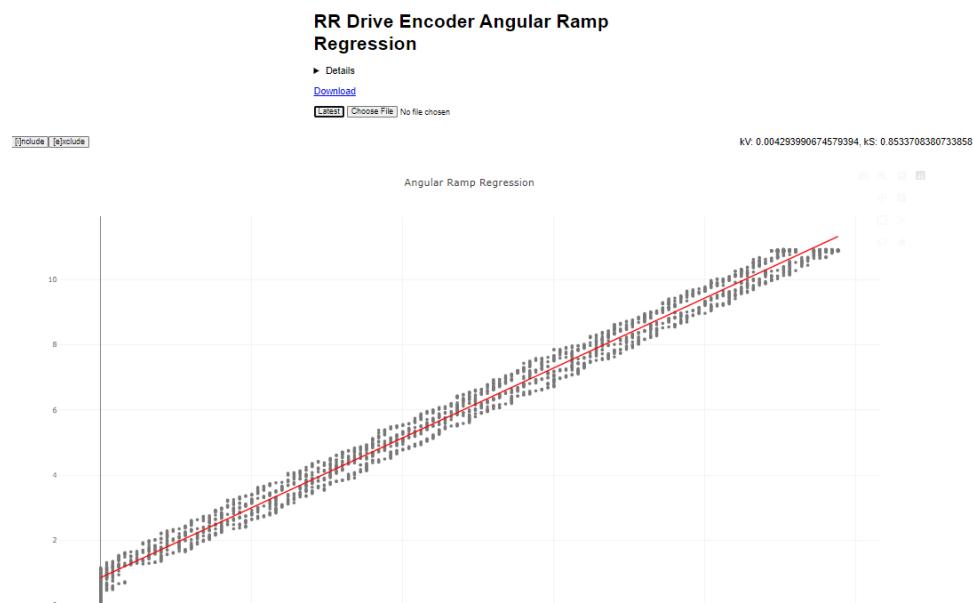
Skip to Step 9, if using Drive Encoders.

- The goal is to determine `lateralInPerTick` empirically
- If all is configured correctly, the `LateralRampLogger` OpMode will move the robot to the left, increasing in speed as it goes (similar to `ForwardRampLogger`). Stop it at any point, keeping in mind that longer runs will collect more data.
- When you're done, go to <http://192.168.43.1:8080/tuning/lateral-ramp.html> and click the "Latest" button. The data should be close to linear, and the slope reported is `lateralInPerTick`.

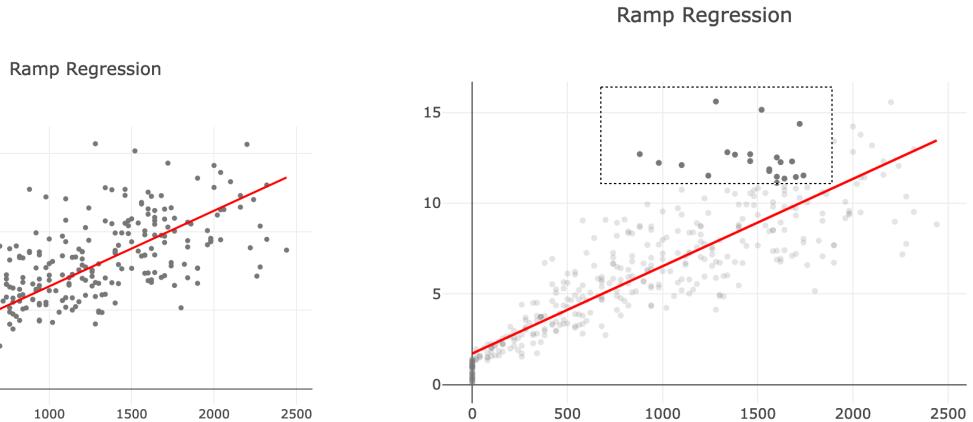
Step 9 : Angular Ramp Logger for Drive Encoders.

(Skip to Step 11 if using Dead Wheel Encoders).

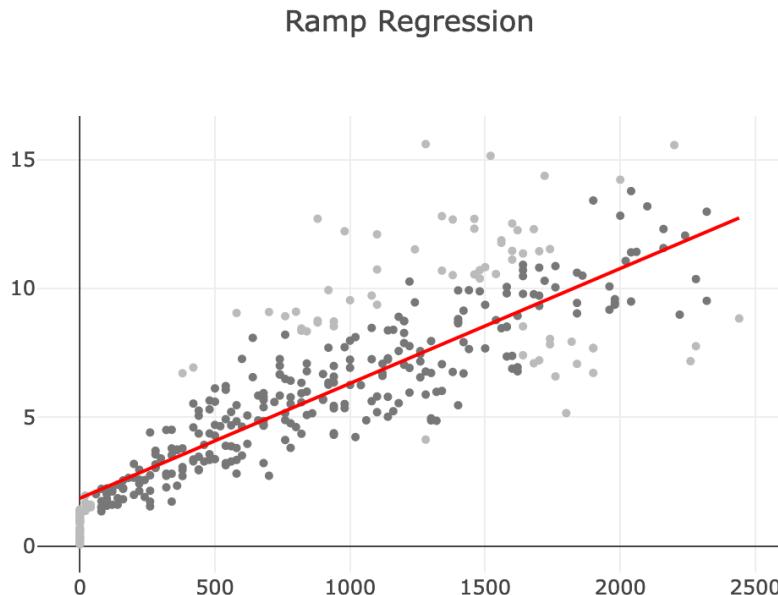
- The goal is determine `trackWidthTicks` empirically
- This opmode `AngularRampLogger` slowly increases the rotation power given to the robot and measures the angular velocity over time to calculate the static and velocity feedforward parameters (`kS` and `kV`, respectively). By default, the power will increase by 0.1 each second until it reaches 0.9.
- Place your robot on the middle of your tiles. Start the `AngularRampLogger` op mode and press stop when the robot reaches maximum rotation speed (ie when is not increasing any further). Don't expect anything to be displayed in telemetry. All data collected is saved to a file for further analysis.
- Once you finish, connect your computer to the RC wireless network using these instructions.
- If you have a control hub, navigate to <http://192.168.43.1:8080/tuning/drive-encoder-angular-ramp.html>
- If you have a RC phone, navigate to <http://192.168.49.1:8080/tuning/drive-encoder-angular-ramp.html> .
- From now on, I'll use Control Hub URLs.
- You should see a page that looks like this (Click the "Latest" button, and you should see a graph appear):



- The gray points are the measurements made by the tuner, and the red line is the line of best fit. Its from this line that the feedforward parameters will be extracted. In fact, you can see preliminary values for kS and kV already displayed above. The above graph is a example of a good fit.
- But in some cases, those estimates are limited by the presence of outliers sometimes as in the example below. You can see that the red line doesn't quite fit the trend. In such cases, we need to help the algorithm out by manually filtering out outlier points.
- First, click and drag to select a rectangular region of the plot:



- Then to remove those points, press the “e” key or the “exclude” button above. The selection box will disappear, and the points will turn a lighter gray. After repeating this process as many times as necessary, you should end up with a plot like this:



- If you mess up and accidentally exclude non-outlier points, select them and press “i” or click the include button to bring them back into the calculation.
- As an example for expected range, we got kS value of 0.85337083807 and kV of 0.00429399067

- Search for “TODO Step 9” in MecanumDrive.java. Now you can copy the values above the plot into the ks and kv fields in the static PARAMS class in Mecanumdrive class.

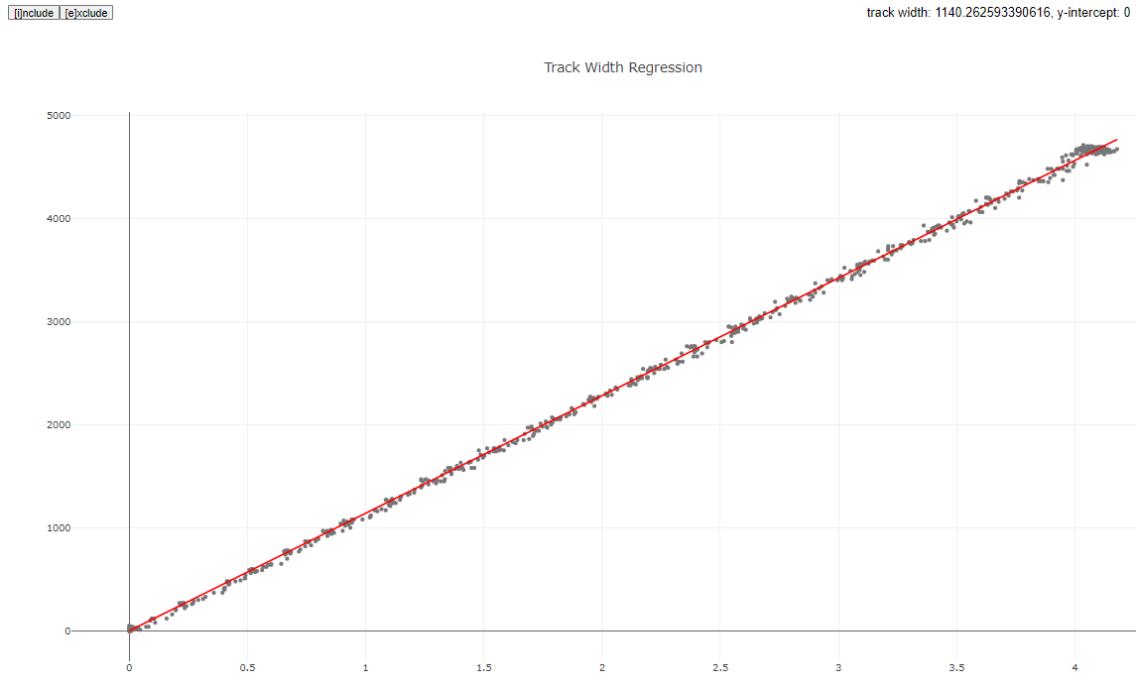
```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help centerstage-road-runner
es ftc teamcode MecanumDrive Params KS TeamCode No Devices Git: ✓ ✓ ✓ ...
Android MecanumDrive.java LocalizationTest.java build.dependencies.gradle build.gradle (TeamCode) Conc ...
Resource Manager
> FtcRobotController
  > TeamCode
    > manifests
    > java
      > org.firstinspires.ftc.teamcode
        > tuning
          LocalizationTest
          ManualFeedbackTuner
          SplineTest
          TuningOpModes
          ConceptTensorFlowObjectDetection
          FTCWiresAutonomous
          FTCWiresTeleOpMode
          Localizer
          MecanumDrive
            PoseMessage
            TankDrive
            ThreeDeadWheelLocalizer
            TwoDeadWheelLocalizer
          java (generated)
          jniLibs
          res
          res (generated)
        Gradle Scripts
MecanumDrive.java
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
public double inPerTick = 0;
//TODO Step 6 (Only for DriveEncoder Localizer) Set value of lateralInPerTick after running LocalizationTest
//TODO Step 8 (Only for DeadWheel Localizer) Set value of lateralInPerTick after running LocalizationTest
//TODO Step 14 Make value of lateralInPerTick accurate after running LocalizationTest
public double lateralInPerTick = 1;
//TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after running LocalizationTest
//TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after running LocalizationTest
public double trackWidthTicks = 0;
// feedforward parameters (in tick units)
//TODO Step 7 (Only for DeadWheel Localizer) Set value for ks and kv after running LocalizationTest
//TODO Step 9 (Only for DriveEncoder Localizer) Set value for ks and kv after running LocalizationTest
public double kS = 0;
public double kV = 0;
//TODO Step 12 Set value of kA after running ManualFeedforwardTuner. In this example, kA = 0
public double kA = 0;
// path profile parameters (in inches)
public double maxWheelVel = 50;
public double minProfileAccel = -30;
public double maxProfileAccel = 50;
// turn profile parameters (in radians)

```

Step 10 : Set Track Width

On the same Angular Ramp Logger plot as in Step 9, Scroll down to find the plot for Track Width Regression.



- In case the graph is not as clean as above, use the same outlier-exclusion technique on the “Track Width Regression” and set `trackWidthTicks` to the “track width” value displayed above when you’re finished.
- As an example of expected value, we got value of 1140.262593390616

```
import ...  
@Config  
public final class MecanumDrive {  
    public static class Params {  
        // drive model parameters  
        //TODO Step 5 Set value of inPerTick after running ForwardPushTest  
        public double inPerTick = 0;  
  
        //TODO Step 6 (Only for DriveEncoder Localizer) Set value of lateralInPerTick after running  
        //TODO Step 8 (Only for DeadWheel Localizer) Set value of lateralInPerTick after running Lnt  
        public double lateralInPerTick = 1;  
  
        //TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after running  
        //TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after running Ang  
        public double trackWidthTicks = 0;  
  
        // feedforward parameters (in tick units)  
        //TODO Step 7 (Only for DeadWheel Localizer) Set value for KS and KV after running ForwardRa  
        //TODO Step 9 (Only for DriveEncoder Localizer) Set value for KS and KV after running Angular  
        public double KS = 0;  
        public double KV = 0;  
  
        //TODO Step 12 Set value of KA after running ManualFeedforwardTuner. In this emperical proce  
        public double KA = 0;  
  
        // path profile parameters (in inches)  
        public double maxWheelVel = 50;  
        public double minProfileAccel = -30;  
    }  
}
```

- Download the code to the Rev Control Hub.

Step 11: Angular Ramp Logger for Dead Wheels Encoders. (Skip to Step 12 ff using Drive Encoders).

- This opmode `AngularRampLogger` slowly increases the rotation power given to the robot and measures the angular velocity over time to calculate the static and velocity feedforward parameters (`kS` and `kV`, respectively). By default, the power will increase by 0.1 each second until it reaches 0.9.
- Place your robot on the middle of your tiles. Start the `AngularRampLogger` op mode and press stop when the robot reaches maximum rotation speed (ie when is not increasing any further). Don't expect anything to be displayed in telemetry. All data collected is saved to a file for further analysis.
- Go to <http://192.168.43.1:8080/tuning/dead-wheel-angular-ramp.html> and click the “Latest” button. Copy the `kS` and `kV` values from `ForwardRampLogger` into the appropriate boxes and click the “update” button. Use the instructions from the `ForwardRampRegression` analysis to fill in `trackWidthTicks` in your drive class and the wheel position fields in your dead wheel class.
- To get an accurate `trackWidthTicks` value, you need accurate values for `kS`, `kV`.

```
//TODO Step 6 (Only for DriveEncoder Localizer) Set value of lateralInPerTick after running LateralPlus
//TODO Step 8 (Only for DeadWheel Localizer) Set value of lateralInPerTick after running ForwardRampLogger
//TODO Step 14 Make value of lateralInPerTick accurate after running LocalizationTest
public double lateralInPerTick = 1;

//TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after running AngularRampLogger
//TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after running AngularRampLogger
// Go to Step 11.1 in Three or Two DeadWheelLocalizer and updated values of par0Yticks, part1Yticks, perpXTicks
public double trackWidthTicks = 0;

// feedforward parameters (in tick units)
//TODO Step 7 (Only for DeadWheel Localizer) Set value for kS and KV after running ForwardRampLogger
//TODO Step 9 (Only for DriveEncoder Localizer) Set value for kS and KV after running AngularRampLogger
public double kS = 0;
public double KV = 0;

//TODO Step 12 Set value of kS after running ManualFeedbackTuner. In this empirical process update value in
// perpXTicks
```

11/13/2023 Update

- **Step 11.1** Scroll down <http://192.168.43.1:8080/tuning/dead-wheel-angular-ramp.html> and if using Three Dead Wheel configuration, find curve fits for `par0YTicks`, `par1YTicks`, `perpXTicks`. Go to `ThreeDeadWheelLocalizer.java` and update values there

```
@Config
public final class ThreeDeadWheelLocalizer implements Localizer {
    public static class Params {
        //TODO Step 11.1 : Update values of par0Yticks, part1Yticks, perpXTicks from AngularRampLogger
        public double par0Yticks = 0; // y position of the first parallel encoder (in tick units)
        public double par1Yticks = 1; // y position of the second parallel encoder (in tick units)
        public double perpXTicks = 0; // x position of the perpendicular encoder (in tick units)
    }

    public static Params PARAMS = new Params();

    public final Encoder par0, par1, perp;

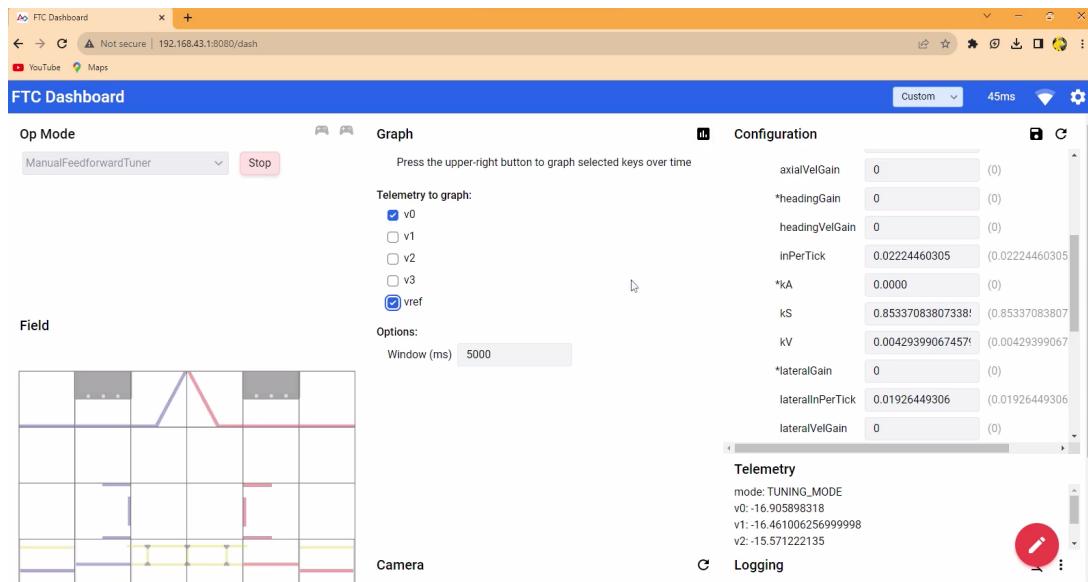
    public final double inPerTick;

    private int lastD0Pos, lastD1Pos, lastD2Pos;
```

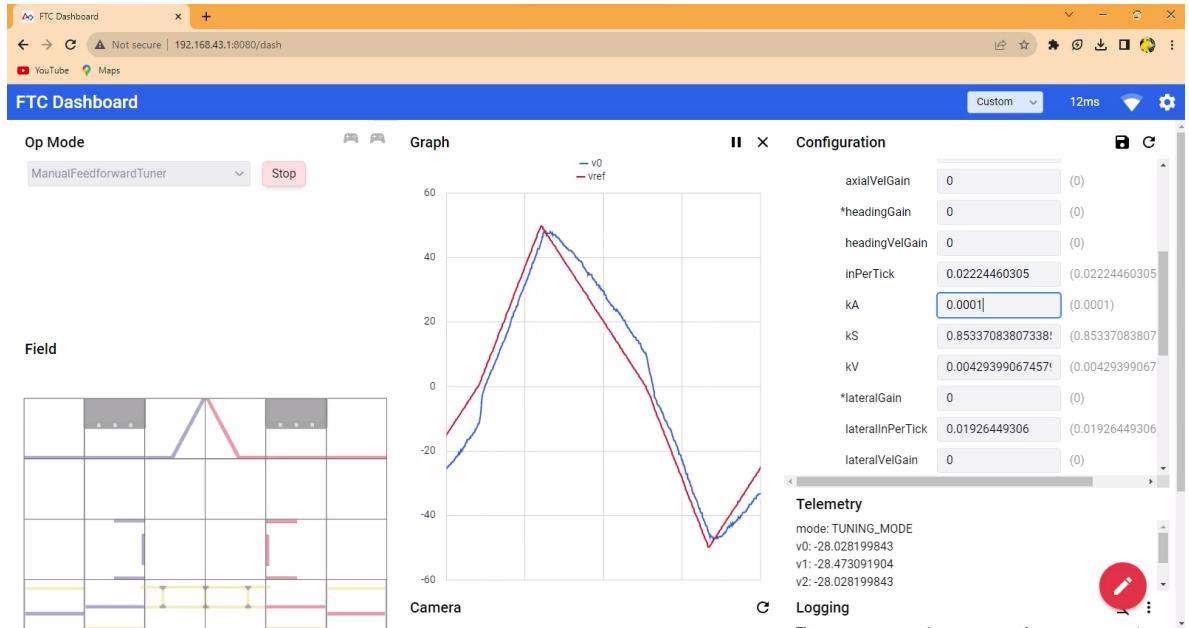
If using Thwo Dead Wheel configuration, find curve fits for `parYTicks`, `perpXTicks`. Go to `TwoDeadWheelLocalizer.java` and update values there

Step 12: Manual Feedforward Tuner.

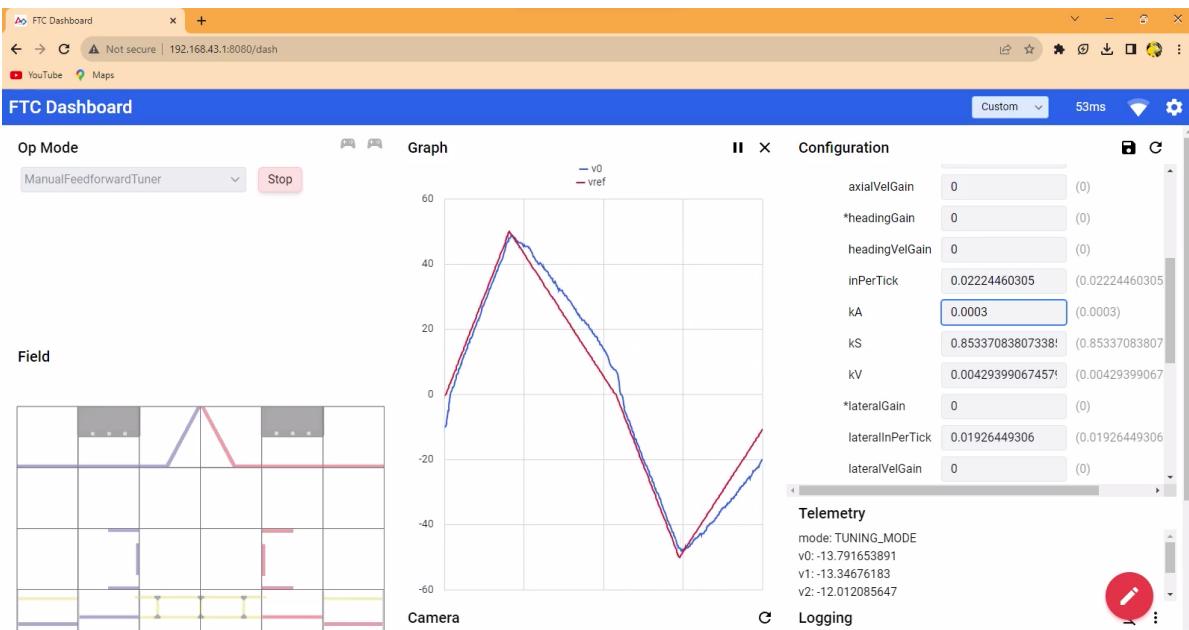
- The goal is to fine-tune kS, kV and add in kA. This routine repeatedly drives forward and back DISTANCE units, giving you an opportunity to finalize the feedforward parameters.
- For this, open FTC Dashboard on your laptop by navigating to <http://192.168.43.1:8080/dash>.
- Place your robot on the tiles with atleast 80 inches distance in front.
- Run the `ManualFeedforwardTuner` op mode from the pull down on the top left. The robot will start moving back and forth, each traveling around 64 inches.
- There may be a drift on the robot as the motion happens. This is expected. Pressing Y/Δ (Xbox/PS4) will pause the tuning process and enter driver override, allowing you to reset the position of the bot. You can use the joystick buttons on the gamepad to move the robot around. Pressing B/O (Xbox/PS4) will return to the tuning process.
- On the dashboard, In the graph section in the center, check the boxes for v0 and vref.and click on the graph button on the top.
- The graph for V0 and Vref will be plotted as below.



- On the Configuration section on the right, scroll to find kA and Enter value of kA in increments of 0.0001 (and hit Enter each time you change the value) to make the lines v0 and vref as close as possible. At some point the lines don't get any better even if you increase kA, and beyond that the robot starts to move uncontrollably.



- Pick the lowest value of kA when the lines are close to each other. As an example the value we got was 0.0003.



- Search for “TODO Step 12” in MecanumDrive.java. Now you can copy the values above the plot into the `ka` fields in the static `PARAMS` class in Mecanumdrive class

```
public double lateralInPerTick = 1;

//TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after running LocalizationTest
//TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after running SplineTest
public double trackWidthTicks = 0;

// feedforward parameters (in tick units)
//TODO Step 7 (Only for DeadWheel Localizer) Set value for KS and KV after running FuzzyTuningOpModes
//TODO Step 9 (Only for DriveEncoder Localizer) Set value for KS and KV after running ConceptTensorFlowObjectDetection
public double KS = 0;
public double KV = 0;

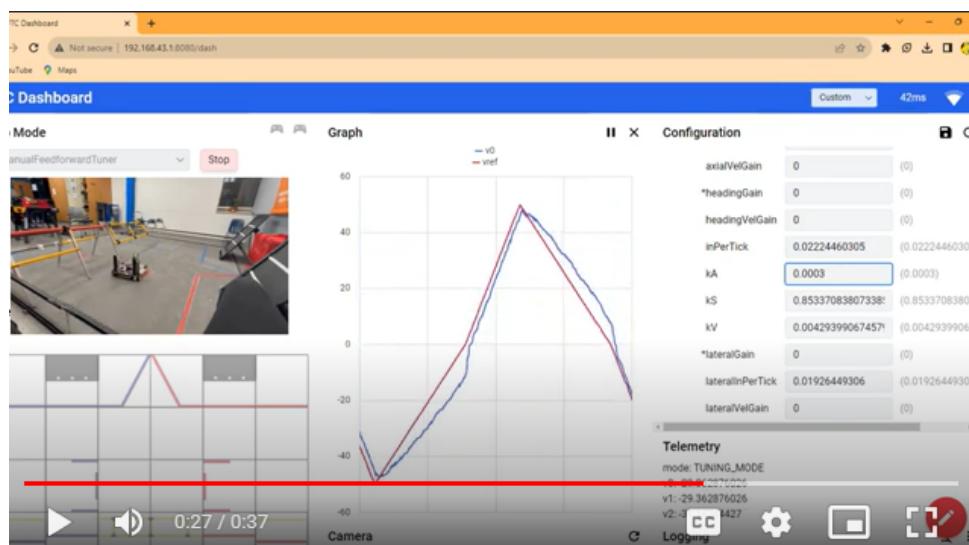
//TODO Step 12 Set value of KA after running ManualFeedforwardTuner. In this empirical case KA = 0
public double KA = 0;

// path profile parameters (in inches)
public double maxWheelVel = 50;
public double minProfileAccel = -30;
public double maxProfileAccel = 50;

// turn profile parameters (in radians)
public double maxAngVel = Math.PI; // shared with path
public double maxAngAccel = Math.PI;

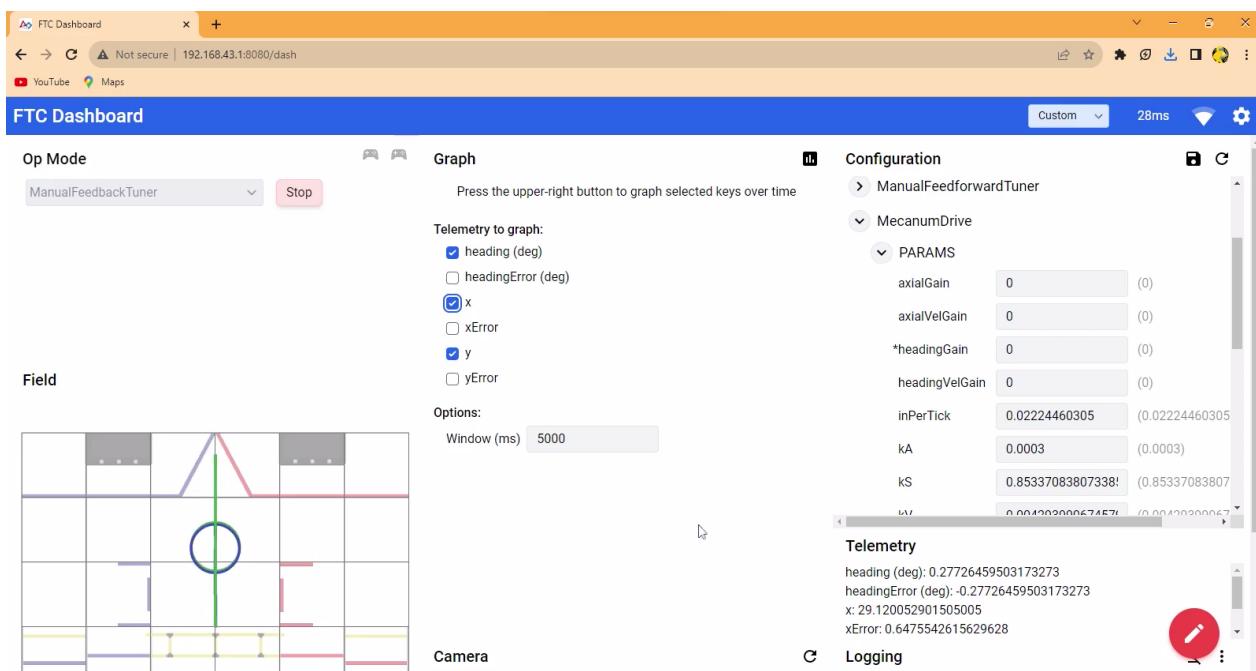
// path controller gains
//TODO Step 12 Set value of Gains after running ManualFeedbackTuner
public double axialGain = 0.0;
public double lateralGain = 0.0;
public double headingGain = 0.0; // shared with turn
```

- Download the code to the Rev Control Hub
 - [Click here for video of Manual Feed Forward Tuning for reference](#)



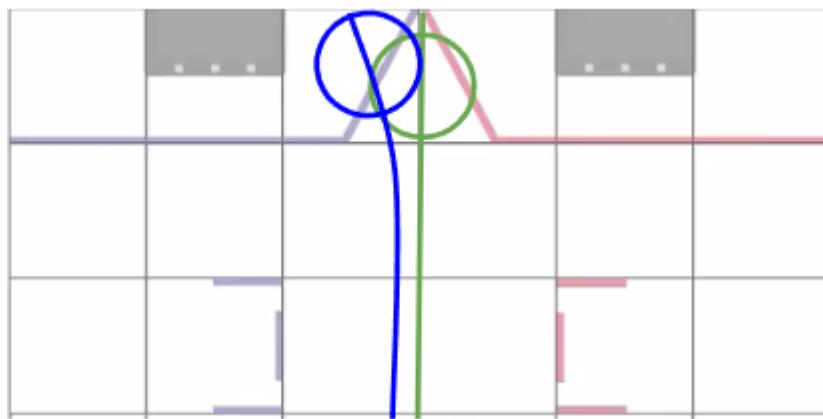
Step 13: Manual Feedback Tuner.

- In the previous steps, you would be experiencing drift in the robot over the many cycles. This is since the robot still has no feedback. The goal is now to tune the feedback parameters.
- Place your robot on the tiles with atleast 80 inches distance in front.
- For this test , open FTC Dashboard on your laptop by navigating to <http://192.168.43.1:8080/dash>
- Run the `ManualFeedbackTuner` op mode from the pull down on the top left. The robot will start moving back and forth, each traveling around 64 inches. The motion would be plotted on the Field view in the dashboard, The Green Circle denotes the input or target motion, and the Blue Circle denotes the robot's actual motion.
- This routine also goes back and forth DISTANCE (64 inches)units but using combined feedforward and feedback. The theory is that we are using this opportunity to tune the feedforward parameters of your trajectory following controller.

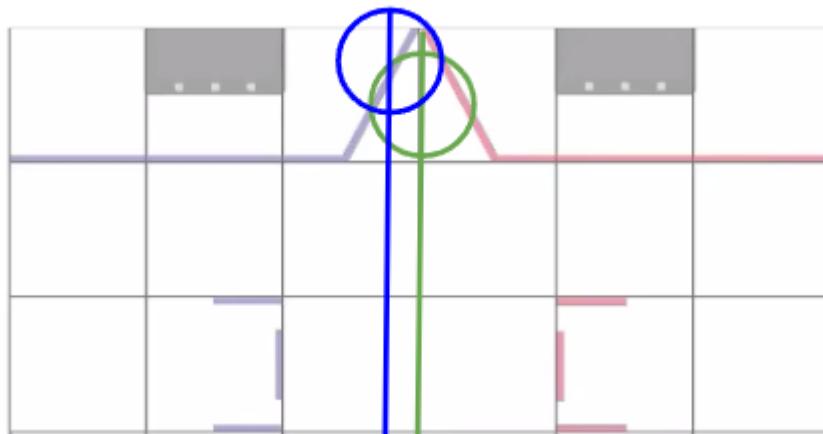


- On the dashboard, In the graph section in the center, check the boxes for heading, x and y and click on the graph button on the top.
- There are two gains for each travel direction (forward, strafe, rotate). The first (“position gain”) is like the proportional term of a position PID, and the second (“velocity gain”) is like the derivative term of a position PID. We would need to only tune the proportional gains mostly.
- The configuration values we are going to tune are the headingGain, axialGain and lateralGain.

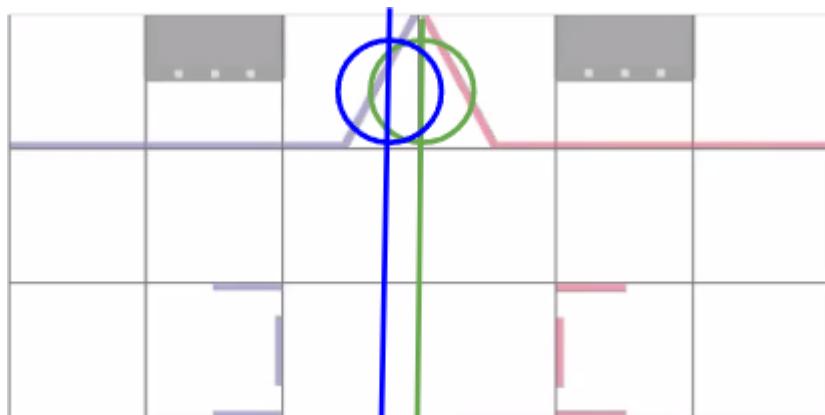
- In the worst case, the starting point would be where the blue circle is drifting away from the green circle in each cycle. The blue circle moves forward and backward more than the green circle and the blue circle motion does not overlap the green circle line, and goes in parallel.



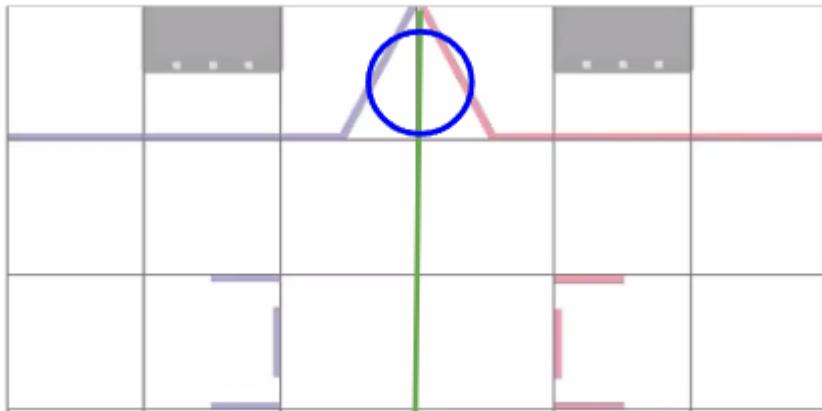
- Start with tuning the headingGain. Increment value of headingGain in steps of 0.5. (Hit Enter each time you change the value). The drifting of the blue circle would start reducing and become parallel to the green circle motion.



- Then tune the axialGain. Increment value of axialGain in steps of 0.5. The overshoot of the blue circle will reduce and it would match the extent of the green circle.



- Lastly, tune the lateralGain. Increment value of lateralGain in steps of 0.5. The blue line and green line would overlap at the right value. At this time, you can push the robot on the side during motion, and it should still track back to the right position.



- As an example, the value we got for headingGain = 3.0, axialGain = 2.0, lateralGain = 2.0.
- Even if there is no variance, suggest keeping the minimum values of the Gains as 1.0.
- Save the values of gains and update the code.
- Search for “TODO Step 13” in MecanumDrive.java. Now you can copy the values above the plot into the `KA` fields in the static `PARAMS` class in Mecanumdrive class
- Download the code to the Rev Control Hub

```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help centerstage-road-runner - MecanumDrive.java [centerstage-road-runner.TeamCode.main]
org firstspires ftc teamcode MecanumDrive Params TeamCode No Devices
Android FtcRobotController TeamCode
> manifests
> java org.firstspires.ftc.teamcode
  > tuning
    LocalizationTest
    ManualFeedbackTuner
    SplineTest
    TuningOpModes
    ConceptTensorFlowObjectDetection
    FTCWiresAutonomous
    FTCWiresTeleOpMode
    Localizer
    MecanumDrive
    PoseMessage
    TankDrive
    ThreeDeadWheelLocalizer
    TwoDeadWheelLocalizer
> java (generated)
> jniLibs
> res
  > res (generated)
> Gradle Scripts

MecanumDrive.java
69 //TODO Step 12 Set value of KA after running ManualFeedforwardTuner.
70 public double KA = 0;
71
72 // path profile parameters (in inches)
73 public double maxWheelVel = 50;
74 public double minProfileAccel = -30;
75 public double maxProfileAccel = 50;
76
77 // turn profile parameters (in radians)
78 public double maxAngVel = Math.PI; // shared with path
79 public double maxAngAccel = Math.PI;
80
81 // path controller gains
82 //TODO Step 12 Set value of Gains after running ManualFeedbackTuner
83 public double axialGain = 0.0;
84 public double lateralGain = 0.0;
85 public double headingGain = 0.0; // shared with turn
86
87 public double axialVelGain = 0.0;
88 public double lateralVelGain = 0.0;
89 public double headingVelGain = 0.0; // shared with turn
90 //TODO End Step 12
91
92
93
94
95
96
97

public static Params PARAMS = new Params();

public final MecanumKinematics kinematics = new MecanumKinematics(
    trackWidth: PARAMS.inPerTick * PARAMS.trackWidthTicks,
    lateralMultiplier: PARAMS.inPerTick
)

```

The screenshot shows the Android Studio interface with the project navigation bar at the top. The left sidebar shows the project structure with a tree view of files and folders. The main editor window displays the `MecanumDrive.java` file. The code contains several TODO comments and annotations. Lines 69, 72, 78, 82, 83, 86, and 90 are highlighted in blue, indicating they are part of a TODO step. Lines 87 through 95 are highlighted in yellow, indicating they are part of another TODO step. The code defines parameters like `KA`, `maxWheelVel`, `minProfileAccel`, `maxProfileAccel`, `maxAngVel`, `maxAngAccel`, and various gains for path and turn controllers. It also initializes a `Params` object and creates a `MecanumKinematics` instance.

Step 14: LocalizationTest

- In the Localization Test, you could drive the robot around with the joystick and the robot would be moving without drifts. However, there may be variation in the physical distance traveled vs what is shown as x, y and heading in the drive station.
- The goal of this test is to fix the motion such that the physical distances match the displayed values of the coordinates
- Straight Test : Fix forward motion measurement (`inPerTick`):
 - Place the robot on the tiles with plenty of room in front. Square the robot up with the grid and make note of its starting position.
 - Run LocalizationTest OpMode on the Driver Hub and then use the joystick to move the robot straight until the end of the tiles.
 - Record the x value from display on the Driver Hub before stopping the op mode. And measure the actual distance traveled with a measuring tape.
 - Calculate the product of the current `inPerTick` and the ratio of actual distance traveled over the displayed value of x.
 - Assign the calculated value as the new `inPerTick`.
 - Download code and repeat the same, till the actual and measured distance are almost equal.
- Strafe Test: Fix lateral motion measurement (`lateralInPerTick`):
 - Set the robot up as before with space towards the right of the robot, Square the robot up with the grid and make note of its starting position.
 - Run LocalizationTest OpMode on the Driver Hub and then use the joystick to strafe the robot right until the end of the tiles.
 - Record the y value from display on the Driver Hub before stopping the op mode. And measure the actual distance traveled with a measuring tape.
 - Calculate the product of the current `lateralInPerTick` and the ratio of actual distance traveled over the displayed value of y.
 - Assign the calculated value as the new `lateralInPerTick` .
 - Download code and repeat the same, till the actual and measured distance are almost equal.

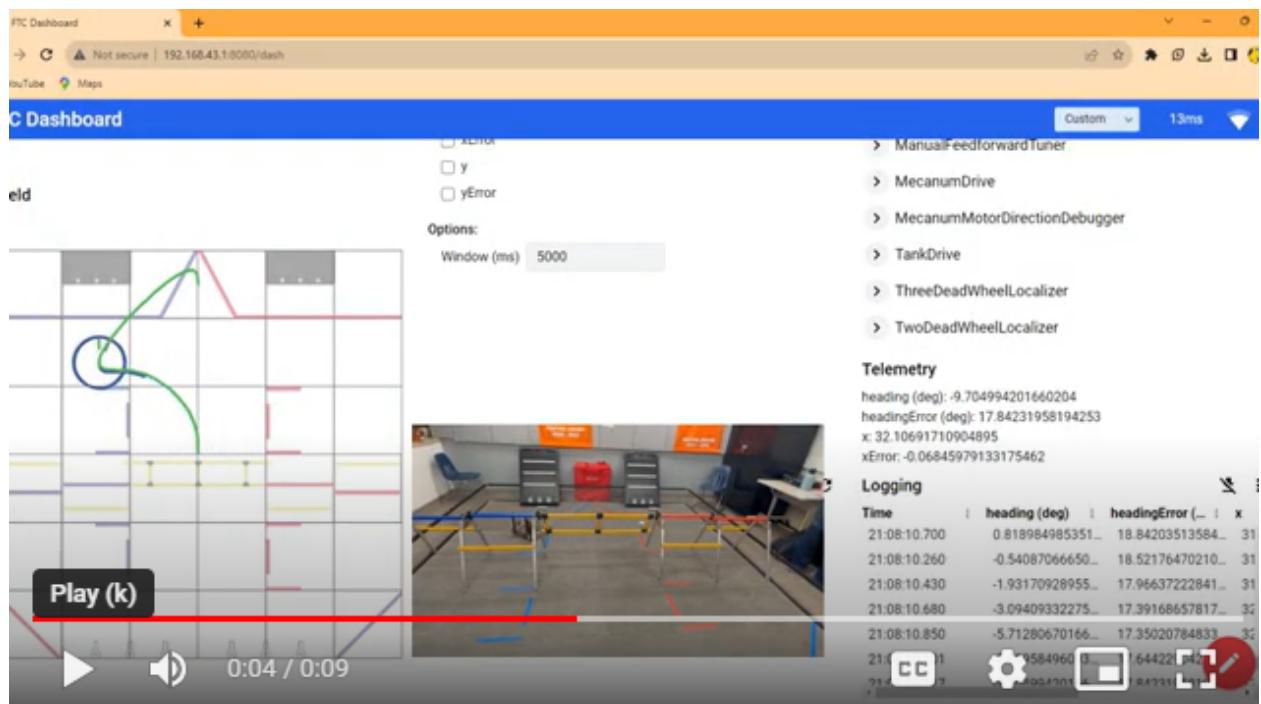
The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project tree under "teamcode". It includes packages like FtcRobotController, TeamCode, manifests, java, org.firstinspires.ftc.teamcode, tuning, LocalizationTest, ManualFeedbackTuner, SplineTest, TuningOpModes, ConceptTensorFlowObjectDetection, FTCWiresAutonomous, FTCWiresTeleOpMode, Localizer, MecanumDrive, PoseMessage, and TankDrive.
- MecanumDrive.java Code:** The main code editor displays the following Java code:

```
public final class MecanumDrive {  
    public static final class Params {  
        // drive model parameters  
        // TODO Step 5 Set value of inPerTick after running ForwardPushTest  
        // TODO Step 14 Make value of inPerTick accurate after running LocalizationTest  
        public double inPerTick = 0;  
  
        // TODO Step 6 (Only for DriveEncoder Localizer) Set value of lateralInPerTick  
        // TODO Step 8 (Only for DeadWheel Localizer) Set value of lateralInPerTick after  
        // TODO Step 14 Make value of lateralInPerTick accurate after running LocalizationTest  
        public double lateralInPerTick = 1;  
  
        // TODO Step 10 (Only for DriveEncoder Localizer) Set value of trackWidthTicks after  
        // TODO Step 11 (Only for DeadWheel Localizer) Set value of trackWidthTicks after  
        public double trackWidthTicks = 0;  
    }  
}
```
- Toolbars and Status Bar:** The top has standard Android Studio toolbars. The bottom status bar shows "67:48 CRLF UTF-8 4 spaces ReleaseBranch".

Step 15 : SplineTest.

- The goal is to validate the calibration done by running a spline motion.
- Set the robot in the center of the field.
- For this test , open FTC Dashboard on your laptop by navigating to <http://192.168.43.1:8080/dash>
- Run the SplineTest op mode from the pull down on the top left. The robot will start moving in a spline motion denoted in the screenshot below. If tuned well, the Blue circle and the robot will trace the green circle accurately.
- [Click here for video of how it is done](#)



If you see variance in the physical motion, or on the display, retrace your steps and fix any errors in the process.

Once you are satisfied with calibration, remove the TODO comments from the code, to mark them as completed.

The calibration should be repeated occasionally, and particularly where there is a significant change physically on the robot, that affects its weight, center of gravity, motors or wheels.

This concludes the calibration process!

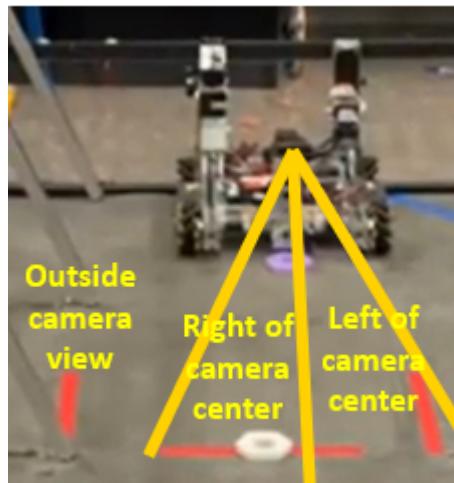
Autonomous Op Mode Code

Now that the setup is done, let's get to the real deal - Autonomous mode. The set up and operation is explained in the "How will the FTCWires Autonomous Mode work" section earlier in this document.

Open the Autonomous mode code in `teamcode/FTCWiresAutonomous.java`

Edit the team name and team number to reflect your credentials.

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help centerstage-road-runner - FTCWiresAutonomous.java [centerstage-road-runner.TeamCode.main]
n java org firstinspires ftc teamcode FTCWiresAutonomous TeamCode No Devices Git
Resource Manager Project Commit Pull Requests Bookmarks Variants
Android > FtcRobotController > TeamCode > manifests > java > org.firstinspires.ftc.teamcode > tuning > LocalizationTest > ManualFeedbackTuner > SplineTest > TuningOpModes > ConceptTensorFlowObjectDetection > FTCWiresAutonomous > FTCWiresTeleOpMode > Localizer > MecanumDrive > PoseMessage > TankDrive > ThreeDeadWheelLocalizer > TwoDeadWheelLocalizer > java (generated) > jniLibs > res > res (generated) > Gradle Scripts
1 //...
29
30 package org.firstinspires.ftc.teamcode;
31
32 import ...
33
34 /**
35  * FTC WIRES Autonomous Example for only vision detection using tensorflow and park
36 */
37
38 @Autonomous(name = "FTC Wires Autonomous Mode", group = "00-Autonomous", preselectTeleOp = "FTC WIRE")
39 public class FTCWiresAutonomous extends LinearOpMode {
40
41     public static String TEAM_NAME = "EDIT TEAM NAME"; //TODO: Enter team Name
42     public static int TEAM_NUMBER = 0; //TODO: Enter team Number
43
44     private static final boolean USE_WEBCAM = true; // true for webcam, false for phone camera
45
46     //Vision parameters
47     private TfodProcessor tfod;
48     private VisionPortal visionPortal;
49
50     //Define and declare Robot Starting Locations
51     public enum START_POSITION{
52
53     }
54
55     public void init() {
56         super.init();
57         tfod = new TfodProcessor(this);
58         tfod.loadModel("res/raw/tfod.tflite");
59         tfod.setMinResultConfidence(0.095f);
60         tfod.activate();
61         tfod.setSearchSpace(SearchSpace.FRONT);
62
63         tfod.setMinResultConfidence(0.095f);
64     }
65
66     public void start() {
67         super.start();
68         tfod.setSearchSpace(SearchSpace.FRONT);
69
70         tfod.setMinResultConfidence(0.095f);
71     }
72
73     public void loop() {
74         super.loop();
75         tfod.setSearchSpace(SearchSpace.FRONT);
76
77         tfod.setMinResultConfidence(0.095f);
78     }
79
80     public void stop() {
81         super.stop();
82         tfod.setSearchSpace(SearchSpace.FRONT);
83
84         tfod.setMinResultConfidence(0.095f);
85     }
86
87     public void end() {
88         super.end();
89         tfod.setSearchSpace(SearchSpace.FRONT);
90
91         tfod.setMinResultConfidence(0.095f);
92     }
93
94     public void autonomous() {
95         super.autonomous();
96         tfod.setSearchSpace(SearchSpace.FRONT);
97
98         tfod.setMinResultConfidence(0.095f);
99     }
100
101     public void teleop() {
102         super.teleop();
103         tfod.setSearchSpace(SearchSpace.FRONT);
104
105         tfod.setMinResultConfidence(0.095f);
106     }
107
108     public void test() {
109         super.test();
110         tfod.setSearchSpace(SearchSpace.FRONT);
111
112         tfod.setMinResultConfidence(0.095f);
113     }
114
115     public void reset() {
116         super.reset();
117         tfod.setSearchSpace(SearchSpace.FRONT);
118
119         tfod.setMinResultConfidence(0.095f);
120     }
121
122     public void error() {
123         super.error();
124         tfod.setSearchSpace(SearchSpace.FRONT);
125
126         tfod.setMinResultConfidence(0.095f);
127     }
128
129     public void initOpMode() {
130         super.initOpMode();
131         tfod.setSearchSpace(SearchSpace.FRONT);
132
133         tfod.setMinResultConfidence(0.095f);
134     }
135
136     public void startOpMode() {
137         super.startOpMode();
138         tfod.setSearchSpace(SearchSpace.FRONT);
139
140         tfod.setMinResultConfidence(0.095f);
141     }
142
143     public void loopOpMode() {
144         super.loopOpMode();
145         tfod.setSearchSpace(SearchSpace.FRONT);
146
147         tfod.setMinResultConfidence(0.095f);
148     }
149
150     public void stopOpMode() {
151         super.stopOpMode();
152         tfod.setSearchSpace(SearchSpace.FRONT);
153
154         tfod.setMinResultConfidence(0.095f);
155     }
156
157     public void errorOpMode() {
158         super.errorOpMode();
159         tfod.setSearchSpace(SearchSpace.FRONT);
160
161         tfod.setMinResultConfidence(0.095f);
162     }
163
164     public void resetOpMode() {
165         super.resetOpMode();
166         tfod.setSearchSpace(SearchSpace.FRONT);
167
168         tfod.setMinResultConfidence(0.095f);
169     }
170
171     public void initTfod() {
172         super.init();
173         tfod = new TfodProcessor(this);
174         tfod.loadModel("res/raw/tfod.tflite");
175         tfod.setMinResultConfidence(0.095f);
176         tfod.activate();
177
178         tfod.setSearchSpace(SearchSpace.FRONT);
179
180         tfod.setMinResultConfidence(0.095f);
181     }
182
183     public void runTfodTensorFlow() {
184         super.loop();
185         tfod.setSearchSpace(SearchSpace.FRONT);
186
187         tfod.setMinResultConfidence(0.095f);
188     }
189
190     public void runTfod() {
191         super.loop();
192         tfod.setSearchSpace(SearchSpace.FRONT);
193
194         tfod.setMinResultConfidence(0.095f);
195     }
196
197     public void runTfodTensorFlow() {
198         super.loop();
199         tfod.setSearchSpace(SearchSpace.FRONT);
200
201         tfod.setMinResultConfidence(0.095f);
202     }
203
204     public void runTfod() {
205         super.loop();
206         tfod.setSearchSpace(SearchSpace.FRONT);
207
208         tfod.setMinResultConfidence(0.095f);
209     }
210
211     public void runTfodTensorFlow() {
212         super.loop();
213         tfod.setSearchSpace(SearchSpace.FRONT);
214
215         tfod.setMinResultConfidence(0.095f);
216     }
217
218     public void runTfod() {
219         super.loop();
220         tfod.setSearchSpace(SearchSpace.FRONT);
221
222         tfod.setMinResultConfidence(0.095f);
223     }
224
225     public void runTfodTensorFlow() {
226         super.loop();
227         tfod.setSearchSpace(SearchSpace.FRONT);
228
229         tfod.setMinResultConfidence(0.095f);
230     }
231
232     public void runTfod() {
233         super.loop();
234         tfod.setSearchSpace(SearchSpace.FRONT);
235
236         tfod.setMinResultConfidence(0.095f);
237     }
238
239     public void runTfodTensorFlow() {
240         super.loop();
241         tfod.setSearchSpace(SearchSpace.FRONT);
242
243         tfod.setMinResultConfidence(0.095f);
244     }
245
246     public void runTfod() {
247         super.loop();
248         tfod.setSearchSpace(SearchSpace.FRONT);
249
250         tfod.setMinResultConfidence(0.095f);
251     }
252
253     public void runTfodTensorFlow() {
254         super.loop();
255         tfod.setSearchSpace(SearchSpace.FRONT);
256
257         tfod.setMinResultConfidence(0.095f);
258     }
259
260     public void runTfod() {
261         super.loop();
262         tfod.setSearchSpace(SearchSpace.FRONT);
263
264         tfod.setMinResultConfidence(0.095f);
265     }
266
267     public void runTfodTensorFlow() {
268         super.loop();
269         tfod.setSearchSpace(SearchSpace.FRONT);
270
271         tfod.setMinResultConfidence(0.095f);
272     }
273
274     public void runTfod() {
275         super.loop();
276         tfod.setSearchSpace(SearchSpace.FRONT);
277
278         tfod.setMinResultConfidence(0.095f);
279     }
280
281     public void runTfodTensorFlow() {
282         super.loop();
283         tfod.setSearchSpace(SearchSpace.FRONT);
284
285         tfod.setMinResultConfidence(0.095f);
286     }
287
288     public void runTfod() {
289         super.loop();
290         tfod.setSearchSpace(SearchSpace.FRONT);
291
292         tfod.setMinResultConfidence(0.095f);
293     }
294
295     public void runTfodTensorFlow() {
296         super.loop();
297         tfod.setSearchSpace(SearchSpace.FRONT);
298
299         tfod.setMinResultConfidence(0.095f);
300     }
301
302     public void runTfod() {
303         super.loop();
304         tfod.setSearchSpace(SearchSpace.FRONT);
305
306         tfod.setMinResultConfidence(0.095f);
307     }
308
309     public void runTfodTensorFlow() {
310         super.loop();
311         tfod.setSearchSpace(SearchSpace.FRONT);
312
313         tfod.setMinResultConfidence(0.095f);
314     }
315
316     public void runTfod() {
317         super.loop();
318         tfod.setSearchSpace(SearchSpace.FRONT);
319
320         tfod.setMinResultConfidence(0.095f);
321     }
322
323     public void runTfodTensorFlow() {
324         super.loop();
325         tfod.setSearchSpace(SearchSpace.FRONT);
326
327         tfod.setMinResultConfidence(0.095f);
328     }
329
330     public void runTfod() {
331         super.loop();
332         tfod.setSearchSpace(SearchSpace.FRONT);
333
334         tfod.setMinResultConfidence(0.095f);
335     }
336
337     public void runTfodTensorFlow() {
338         super.loop();
339         tfod.setSearchSpace(SearchSpace.FRONT);
340
341         tfod.setMinResultConfidence(0.095f);
342     }
343
344     public void runTfod() {
345         super.loop();
346         tfod.setSearchSpace(SearchSpace.FRONT);
347
348         tfod.setMinResultConfidence(0.095f);
349     }
350
351     public void runTfodTensorFlow() {
352         super.loop();
353         tfod.setSearchSpace(SearchSpace.FRONT);
354
355         tfod.setMinResultConfidence(0.095f);
356     }
357
358     public void runTfod() {
359         super.loop();
360         tfod.setSearchSpace(SearchSpace.FRONT);
361
362         tfod.setMinResultConfidence(0.095f);
363     }
364
365     public void runTfodTensorFlow() {
366         super.loop();
367         tfod.setSearchSpace(SearchSpace.FRONT);
368
369         tfod.setMinResultConfidence(0.095f);
370     }
371
372     public void runTfod() {
373         super.loop();
374         tfod.setSearchSpace(SearchSpace.FRONT);
375
376         tfod.setMinResultConfidence(0.095f);
377     }
378
379     public void runTfodTensorFlow() {
380         super.loop();
381         tfod.setSearchSpace(SearchSpace.FRONT);
382
383         tfod.setMinResultConfidence(0.095f);
384     }
385
386     public void runTfod() {
387         super.loop();
388         tfod.setSearchSpace(SearchSpace.FRONT);
389
390         tfod.setMinResultConfidence(0.095f);
391     }
392
393     public void runTfodTensorFlow() {
394         super.loop();
395         tfod.setSearchSpace(SearchSpace.FRONT);
396
397         tfod.setMinResultConfidence(0.095f);
398     }
399
400     public void runTfod() {
401         super.loop();
402         tfod.setSearchSpace(SearchSpace.FRONT);
403
404         tfod.setMinResultConfidence(0.095f);
405     }
406
407     public void runTfodTensorFlow() {
408         super.loop();
409         tfod.setSearchSpace(SearchSpace.FRONT);
410
411         tfod.setMinResultConfidence(0.095f);
412     }
413
414     public void runTfod() {
415         super.loop();
416         tfod.setSearchSpace(SearchSpace.FRONT);
417
418         tfod.setMinResultConfidence(0.095f);
419     }
420
421     public void runTfodTensorFlow() {
422         super.loop();
423         tfod.setSearchSpace(SearchSpace.FRONT);
424
425         tfod.setMinResultConfidence(0.095f);
426     }
427
428     public void runTfod() {
429         super.loop();
430         tfod.setSearchSpace(SearchSpace.FRONT);
431
432         tfod.setMinResultConfidence(0.095f);
433     }
434
435     public void runTfodTensorFlow() {
436         super.loop();
437         tfod.setSearchSpace(SearchSpace.FRONT);
438
439         tfod.setMinResultConfidence(0.095f);
440     }
441
442     public void runTfod() {
443         super.loop();
444         tfod.setSearchSpace(SearchSpace.FRONT);
445
446         tfod.setMinResultConfidence(0.095f);
447     }
448
449     public void runTfodTensorFlow() {
450         super.loop();
451         tfod.setSearchSpace(SearchSpace.FRONT);
452
453         tfod.setMinResultConfidence(0.095f);
454     }
455
456     public void runTfod() {
457         super.loop();
458         tfod.setSearchSpace(SearchSpace.FRONT);
459
460         tfod.setMinResultConfidence(0.095f);
461     }
462
463     public void runTfodTensorFlow() {
464         super.loop();
465         tfod.setSearchSpace(SearchSpace.FRONT);
466
467         tfod.setMinResultConfidence(0.095f);
468     }
469
470     public void runTfod() {
471         super.loop();
472         tfod.setSearchSpace(SearchSpace.FRONT);
473
474         tfod.setMinResultConfidence(0.095f);
475     }
476
477     public void runTfodTensorFlow() {
478         super.loop();
479         tfod.setSearchSpace(SearchSpace.FRONT);
480
481         tfod.setMinResultConfidence(0.095f);
482     }
483
484     public void runTfod() {
485         super.loop();
486         tfod.setSearchSpace(SearchSpace.FRONT);
487
488         tfod.setMinResultConfidence(0.095f);
489     }
490
491     public void runTfodTensorFlow() {
492         super.loop();
493         tfod.setSearchSpace(SearchSpace.FRONT);
494
495         tfod.setMinResultConfidence(0.095f);
496     }
497
498     public void runTfod() {
499         super.loop();
500         tfod.setSearchSpace(SearchSpace.FRONT);
501
502         tfod.setMinResultConfidence(0.095f);
503     }
504
505     public void runTfodTensorFlow() {
506         super.loop();
507         tfod.setSearchSpace(SearchSpace.FRONT);
508
509         tfod.setMinResultConfidence(0.095f);
510     }
511
512     public void runTfod() {
513         super.loop();
514         tfod.setSearchSpace(SearchSpace.FRONT);
515
516         tfod.setMinResultConfidence(0.095f);
517     }
518
519     public void runTfodTensorFlow() {
520         super.loop();
521         tfod.setSearchSpace(SearchSpace.FRONT);
522
523         tfod.setMinResultConfidence(0.095f);
524     }
525
526     public void runTfod() {
527         super.loop();
528         tfod.setSearchSpace(SearchSpace.FRONT);
529
530         tfod.setMinResultConfidence(0.095f);
531     }
532
533     public void runTfodTensorFlow() {
534         super.loop();
535         tfod.setSearchSpace(SearchSpace.FRONT);
536
537         tfod.setMinResultConfidence(0.095f);
538     }
539
540     public void runTfod() {
541         super.loop();
542         tfod.setSearchSpace(SearchSpace.FRONT);
543
544         tfod.setMinResultConfidence(0.095f);
545     }
546
547     public void runTfodTensorFlow() {
548         super.loop();
549         tfod.setSearchSpace(SearchSpace.FRONT);
550
551         tfod.setMinResultConfidence(0.095f);
552     }
553
554     public void runTfod() {
555         super.loop();
556         tfod.setSearchSpace(SearchSpace.FRONT);
557
558         tfod.setMinResultConfidence(0.095f);
559     }
560
561     public void runTfodTensorFlow() {
562         super.loop();
563         tfod.setSearchSpace(SearchSpace.FRONT);
564
565         tfod.setMinResultConfidence(0.095f);
566     }
567
568     public void runTfod() {
569         super.loop();
570         tfod.setSearchSpace(SearchSpace.FRONT);
571
572         tfod.setMinResultConfidence(0.095f);
573     }
574
575     public void runTfodTensorFlow() {
576         super.loop();
577         tfod.setSearchSpace(SearchSpace.FRONT);
578
579         tfod.setMinResultConfidence(0.095f);
580     }
581
582     public void runTfod() {
583         super.loop();
584         tfod.setSearchSpace(SearchSpace.FRONT);
585
586         tfod.setMinResultConfidence(0.095f);
587     }
588
589     public void runTfodTensorFlow() {
590         super.loop();
591         tfod.setSearchSpace(SearchSpace.FRONT);
592
593         tfod.setMinResultConfidence(0.095f);
594     }
595
596     public void runTfod() {
597         super.loop();
598         tfod.setSearchSpace(SearchSpace.FRONT);
599
600         tfod.setMinResultConfidence(0.095f);
601     }
602
603     public void runTfodTensorFlow() {
604         super.loop();
605         tfod.setSearchSpace(SearchSpace.FRONT);
606
607         tfod.setMinResultConfidence(0.095f);
608     }
609
610     public void runTfod() {
611         super.loop();
612         tfod.setSearchSpace(SearchSpace.FRONT);
613
614         tfod.setMinResultConfidence(0.095f);
615     }
616
617     public void runTfodTensorFlow() {
618         super.loop();
619         tfod.setSearchSpace(SearchSpace.FRONT);
620
621         tfod.setMinResultConfidence(0.095f);
622     }
623
624     public void runTfod() {
625         super.loop();
626         tfod.setSearchSpace(SearchSpace.FRONT);
627
628         tfod.setMinResultConfidence(0.095f);
629     }
630
631     public void runTfodTensorFlow() {
632         super.loop();
633         tfod.setSearchSpace(SearchSpace.FRONT);
634
635         tfod.setMinResultConfidence(0.095f);
636     }
637
638     public void runTfod() {
639         super.loop();
640         tfod.setSearchSpace(SearchSpace.FRONT);
641
642         tfod.setMinResultConfidence(0.095f);
643     }
644
645     public void runTfodTensorFlow() {
646         super.loop();
647         tfod.setSearchSpace(SearchSpace.FRONT);
648
649         tfod.setMinResultConfidence(0.095f);
650     }
651
652     public void runTfod() {
653         super.loop();
654         tfod.setSearchSpace(SearchSpace.FRONT);
655
656         tfod.setMinResultConfidence(0.095f);
657     }
658
659     public void runTfodTensorFlow() {
660         super.loop();
661         tfod.setSearchSpace(SearchSpace.FRONT);
662
663         tfod.setMinResultConfidence(0.095f);
664     }
665
666     public void runTfod() {
667         super.loop();
668         tfod.setSearchSpace(SearchSpace.FRONT);
669
670         tfod.setMinResultConfidence(0.095f);
671     }
672
673     public void runTfodTensorFlow() {
674         super.loop();
675         tfod.setSearchSpace(SearchSpace.FRONT);
676
677         tfod.setMinResultConfidence(0.095f);
678     }
679
680     public void runTfod() {
681         super.loop();
682         tfod.setSearchSpace(SearchSpace.FRONT);
683
684         tfod.setMinResultConfidence(0.095f);
685     }
686
687     public void runTfodTensorFlow() {
688         super.loop();
689         tfod.setSearchSpace(SearchSpace.FRONT);
690
691         tfod.setMinResultConfidence(0.095f);
692     }
693
694     public void runTfod() {
695         super.loop();
696         tfod.setSearchSpace(SearchSpace.FRONT);
697
698         tfod.setMinResultConfidence(0.095f);
699     }
699
700     public void runTfodTensorFlow() {
701         super.loop();
702         tfod.setSearchSpace(SearchSpace.FRONT);
703
704         tfod.setMinResultConfidence(0.095f);
705     }
705
706     public void runTfod() {
707         super.loop();
708         tfod.setSearchSpace(SearchSpace.FRONT);
709
709         tfod.setMinResultConfidence(0.095f);
710     }
710
711     public void runTfodTensorFlow() {
712         super.loop();
713         tfod.setSearchSpace(SearchSpace.FRONT);
714
715         tfod.setMinResultConfidence(0.095f);
716     }
716
717     public void runTfod() {
718         super.loop();
719         tfod.setSearchSpace(SearchSpace.FRONT);
720
720         tfod.setMinResultConfidence(0.095f);
721     }
721
722     public void runTfodTensorFlow() {
723         super.loop();
724         tfod.setSearchSpace(SearchSpace.FRONT);
725
726         tfod.setMinResultConfidence(0.095f);
727     }
727
728     public void runTfod() {
729         super.loop();
730         tfod.setSearchSpace(SearchSpace.FRONT);
731
731         tfod.setMinResultConfidence(0.095f);
732     }
732
733     public void runTfodTensorFlow() {
734         super.loop();
735         tfod.setSearchSpace(SearchSpace.FRONT);
736
737         tfod.setMinResultConfidence(0.095f);
738     }
738
739     public void runTfod() {
740         super.loop();
741         tfod.setSearchSpace(SearchSpace.FRONT);
742
742         tfod.setMinResultConfidence(0.095f);
743     }
743
744     public void runTfodTensorFlow() {
745         super.loop();
746         tfod.setSearchSpace(SearchSpace.FRONT);
747
748         tfod.setMinResultConfidence(0.095f);
749     }
749
750     public void runTfod() {
751         super.loop();
752         tfod.setSearchSpace(SearchSpace.FRONT);
753
753         tfod.setMinResultConfidence(0.095f);
754     }
754
755     public void runTfodTensorFlow() {
756         super.loop();
757         tfod.setSearchSpace(SearchSpace.FRONT);
758
759         tfod.setMinResultConfidence(0.095f);
760     }
760
761     public void runTfod() {
762         super.loop();
763         tfod.setSearchSpace(SearchSpace.FRONT);
764
764         tfod.setMinResultConfidence(0.095f);
765     }
765
766     public void runTfodTensorFlow() {
767         super.loop();
768         tfod.setSearchSpace(SearchSpace.FRONT);
769
770         tfod.setMinResultConfidence(0.095f);
771     }
771
772     public void runTfod() {
773         super.loop();
774         tfod.setSearchSpace(SearchSpace.FRONT);
775
775         tfod.setMinResultConfidence(0.095f);
776     }
776
777     public void runTfodTensorFlow() {
778         super.loop();
779         tfod.setSearchSpace(SearchSpace.FRONT);
780
781         tfod.setMinResultConfidence(0.095f);
782     }
782
783     public void runTfod() {
784         super.loop();
785         tfod.setSearchSpace(SearchSpace.FRONT);
786
786         tfod.setMinResultConfidence(0.095f);
787     }
787
788     public void runTfodTensorFlow() {
789         super.loop();
790         tfod.setSearchSpace(SearchSpace.FRONT);
791
792         tfod.setMinResultConfidence(0.095f);
793     }
793
794     public void runTfod() {
795         super.loop();
796         tfod.setSearchSpace(SearchSpace.FRONT);
797
797         tfod.setMinResultConfidence(0.095f);
798     }
798
799     public void runTfodTensorFlow() {
800         super.loop();
801         tfod.setSearchSpace(SearchSpace.FRONT);
802
803         tfod.setMinResultConfidence(0.095f);
804     }
804
805     public void runTfod() {
806         super.loop();
807         tfod.setSearchSpace(SearchSpace.FRONT);
808
808         tfod.setMinResultConfidence(0.095f);
809     }
809
810     public void runTfodTensorFlow() {
811         super.loop();
812         tfod.setSearchSpace(SearchSpace.FRONT);
813
814         tfod.setMinResultConfidence(0.095f);
815     }
815
816     public void runTfod() {
817         super.loop();
818         tfod.setSearchSpace(SearchSpace.FRONT);
819
819         tfod.setMinResultConfidence(0.095f);
820     }
820
821     public void runTfodTensorFlow() {
822         super.loop();
823         tfod.setSearchSpace(SearchSpace.FRONT);
824
825         tfod.setMinResultConfidence(0.095f);
826     }
826
827     public void runTfod() {
828         super.loop();
829         tfod.setSearchSpace(SearchSpace.FRONT);
830
830         tfod.setMinResultConfidence(0.095f);
831     }
831
832     public void runTfodTensorFlow() {
833         super.loop();
834         tfod.setSearchSpace(SearchSpace.FRONT);
835
836         tfod.setMinResultConfidence(0.095f);
837     }
837
838     public void runTfod() {
839         super.loop();
840         tfod.setSearchSpace(SearchSpace.FRONT);
841
841         tfod.setMinResultConfidence(0.095f);
842     }
842
843     public void runTfodTensorFlow() {
844         super.loop();
845         tfod.setSearchSpace(SearchSpace.FRONT);
846
847         tfod.setMinResultConfidence(0.095f);
848     }
848
849     public void runTfod() {
850         super.loop();
851         tfod.setSearchSpace(SearchSpace.FRONT);
852
852         tfod.setMinResultConfidence(0.095f);
853     }
853
854     public void runTfodTensorFlow() {
855         super.loop();
856         tfod.setSearchSpace(SearchSpace.FRONT);
857
858         tfod.setMinResultConfidence(0.095f);
859     }
859
860     public void runTfod() {
861         super.loop();
862         tfod.setSearchSpace(SearchSpace.FRONT);
863
863         tfod.setMinResultConfidence(0.095f);
864     }
864
865     public void runTfodTensorFlow() {
866         super.loop();
867         tfod.setSearchSpace(SearchSpace.FRONT);
868
869         tfod.setMinResultConfidence(0.095f);
870     }
870
871     public void runTfod() {
872         super.loop();
873         tfod.setSearchSpace(SearchSpace.FRONT);
874
874         tfod.setMinResultConfidence(0.095f);
875     }
875
876     public void runTfodTensorFlow() {
877         super.loop();
878         tfod.setSearchSpace(SearchSpace.FRONT);
879
880         tfod.setMinResultConfidence(0.095f);
881     }
881
882     public void runTfod() {
883         super.loop();
884         tfod.setSearchSpace(SearchSpace.FRONT);
885
885         tfod.setMinResultConfidence(0.095f);
886     }
886
887     public void runTfodTensorFlow() {
888         super.loop();
889         tfod.setSearchSpace(SearchSpace.FRONT);
890
891         tfod.setMinResultConfidence(0.095f);
892     }
892
893     public void runTfod() {
894         super.loop();
895         tfod.setSearchSpace(SearchSpace.FRONT);
896
896         tfod.setMinResultConfidence(0.095f);
897     }
897
898     public void runTfodTensorFlow() {
899         super.loop();
900         tfod.setSearchSpace(SearchSpace.FRONT);
901
902         tfod.setMinResultConfidence(0.095f);
903     }
903
904     public void runTfod() {
905         super.loop();
906         tfod.setSearchSpace(SearchSpace.FRONT);
907
907         tfod.setMinResultConfidence(0.095f);
908     }
908
909     public void runTfodTensorFlow() {
910         super.loop();
911         tfod.setSearchSpace(SearchSpace.FRONT);
912
913         tfod.setMinResultConfidence(0.095f);
914     }
914
915     public void runTfod() {
916         super.loop();
917         tfod.setSearchSpace(SearchSpace.FRONT);
918
918         tfod.setMinResultConfidence(0.095f);
919     }
919
920     public void runTfodTensorFlow() {
921         super.loop();
922         tfod.setSearchSpace(SearchSpace.FRONT);
923
924         tfod.setMinResultConfidence(0.095f);
925     }
925
926     public void runTfod() {
927         super.loop();
928         tfod.setSearchSpace(SearchSpace.FRONT);
929
929         tfod.setMinResultConfidence(0.095f);
930     }
930
931     public void runTfodTensorFlow() {
932         super.loop();
933         tfod.setSearchSpace(SearchSpace.FRONT);
934
935         tfod.setMinResultConfidence(0.095f);
936     }
936
937     public void runTfod() {
938         super.loop();
939         tfod.setSearchSpace(SearchSpace.FRONT);
940
940         tfod.setMinResultConfidence(0.095f);
941     }
941
942     public void runTfodTensorFlow() {
943         super.loop();
944         tfod.setSearchSpace(SearchSpace.FRONT);
945
946         tfod.setMinResultConfidence(0.095f);
947     }
947
948     public void runTfod() {
949         super.loop();
950         tfod.setSearchSpace(SearchSpace.FRONT);
951
951         tfod.setMinResultConfidence(0.095f);
952     }
952
953     public void runTfodTensorFlow() {
954         super.loop();
955         tfod.setSearchSpace(SearchSpace.FRONT);
956
957         tfod.setMinResultConfidence(0.095f);
958     }
958
959     public void runTfod() {
960         super.loop();
961         tfod.setSearchSpace(SearchSpace.FRONT);
962
962         tfod.setMinResultConfidence(0.095f);
963     }
963
964     public void runTfodTensorFlow() {
965         super.loop();
966         tfod.setSearchSpace(SearchSpace.FRONT);
967
968         tfod.setMinResultConfidence(0.095f);
969     }
969
970     public void runTfod() {
971         super.loop();
972         tfod.setSearchSpace(SearchSpace.FRONT);
973
973         tfod.setMinResultConfidence(0.095f);
974     }
974
975     public void runTfodTensorFlow() {
976         super.loop();
977         tfod.setSearchSpace(SearchSpace.FRONT);
978
979         tfod.setMinResultConfidence(0.095f);
980     }
980
981     public void runTfod() {
982         super.loop();
983         tfod.setSearchSpace(SearchSpace.FRONT);
984
984         tfod.setMinResultConfidence(0.095f);
985     }
985
986     public void runTfodTensorFlow() {
987         super.loop();
988         tfod.setSearchSpace(SearchSpace.FRONT);
989
990         tfod.setMinResultConfidence(0.095f);
991     }
991
992     public void runTfod() {
993         super.loop();
994         tfod.setSearchSpace(SearchSpace.FRONT);
995
995         tfod.setMinResultConfidence(0.095f);
996     }
996
997     public void runTfodTensorFlow() {
998         super.loop();
999         tfod.setSearchSpace(SearchSpace.FRONT);
1000
1001         tfod.setMinResultConfidence(0.095f);
1002     }
1002
1003     public void runTfod() {
1004         super.loop();
1005         tfod.setSearchSpace(SearchSpace.FRONT);
1006
1006         tfod.setMinResultConfidence(0.095f);
1007     }
1007
1008     public void runTfodTensorFlow() {
1009         super.loop();
1010         tfod.setSearchSpace(SearchSpace.FRONT);
1011
1012         tfod.setMinResultConfidence(0.095f);
1013     }
1013
1014     public void runTfod() {
1015         super.loop();
1016         tfod.setSearchSpace(SearchSpace.FRONT);
1017
1017         tfod.setMinResultConfidence(0.095f);
1018     }
1018
1019     public void runTfodTensorFlow() {
1020         super.loop();
1021         tfod.setSearchSpace(SearchSpace.FRONT);
1022
1023         tfod.setMinResultConfidence(0.095f);
1024     }
1024
1025     public void runTfod() {
1026         super.loop();
1027         tfod.setSearchSpace(SearchSpace.FRONT);
1028
1028         tfod.setMinResultConfidence(0.095f);
1029     }
1029
1030     public void runTfodTensorFlow() {
1031         super.loop();
1032         tfod.setSearchSpace(SearchSpace.FRONT);
1033
1034         tfod.setMinResultConfidence(0.095f);
1035     }
1035
1036     public void runTfod() {
1037         super.loop();
1038         tfod.setSearchSpace(SearchSpace.FRONT);
1039
1039         tfod.setMinResultConfidence(0.095f);
1040     }
1040
1041     public void runTfodTensorFlow() {
1042         super.loop();
1043         tfod.setSearchSpace(SearchSpace.FRONT);
1044
1045         tfod.setMinResultConfidence(0.095f);
1046     }
1046
1047     public void runTfod() {
1048         super.loop();
1049         tfod.setSearchSpace(SearchSpace.FRONT);
1050
1050         tfod.setMinResultConfidence(0.095f);
1051     }
1051
1052     public void runTfodTensorFlow() {
1053         super.loop();
1054         tfod.setSearchSpace(SearchSpace.FRONT);
1055
1056         tfod.setMinResultConfidence(0.095f);
1057     }
1057
1058     public void runTfod() {
1059         super.loop();
1060         tfod.setSearchSpace(SearchSpace.FRONT);
1061
1061         tfod.setMinResultConfidence(0.095f);
1062     }
1062
1063     public void runTfodTensorFlow() {
1064         super.loop();
1065         tfod.setSearchSpace(SearchSpace.FRONT);
1066
1067         tfod.setMinResultConfidence(0.095f);
1068     }
1068
1069     public void runTfod() {
1070         super.loop();
1071         tfod.setSearchSpace(SearchSpace.FRONT);
1072
1072         tfod.setMinResultConfidence(0.095f);
1073     }
1073
1074     public void runTfodTensorFlow() {
1075         super.loop();
1076         tfod.setSearchSpace(SearchSpace.FRONT);
1077
1078         tfod.setMinResultConfidence(0.095f);
1079     }
1079
1080     public void runTfod() {
1081         super.loop();
1082         tfod.setSearchSpace(SearchSpace.FRONT);
1083
1083         tfod.setMinResultConfidence(0.095f);
1084     }
1084
1085     public void runTfodTensorFlow() {
1086         super.loop();
1087         tfod.setSearchSpace(SearchSpace.FRONT);
1088
1089         tfod.setMinResultConfidence(0.095f);
1090     }
1090
1091     public void runTfod() {
1092         super.loop();
1093         tfod.setSearchSpace(SearchSpace.FRONT);
1094
1094         tfod.setMinResultConfidence(0.095f);
1095     }
1095
1096     public void runTfodTensorFlow() {
1097         super.loop();
1098         tfod.setSearchSpace(SearchSpace.FRONT);
1099
1100         tfod.setMinResultConfidence(0.095f);
1101     }
1101
1102     public void runTfod() {
1103         super.loop();
1104         tfod.setSearchSpace(SearchSpace.FRONT);
1105
1105         tfod.setMinResultConfidence(0.095f);
1106     }
1106
1107     public void runTfodTensorFlow() {
1108         super.loop();
1109         tfod.setSearchSpace(SearchSpace.FRONT);
1110
1111         tfod.setMinResultConfidence(0.095f);
1112     }
1112
1113     public void runTfod() {
1114         super.loop();
1115         tfod.setSearchSpace(SearchSpace.FRONT);
1116
1116         tfod.setMinResultConfidence(0.095f);
1117     }
1117
1118     public void runTfodTensorFlow() {
1119         super.loop();
1120         tfod.setSearchSpace(SearchSpace.FRONT);
1121
1122         tfod.setMinResultConfidence(0.095f);
1123     }
1123
1124     public void runTfod() {
1125         super.loop();
1126         tfod.setSearchSpace(SearchSpace.FRONT);
1127
1127         tfod.setMinResultConfidence(0.095f);
1128     }
1128
1129     public void runTfodTensorFlow() {
1130         super.loop();
1131         tfod.setSearchSpace(SearchSpace.FRONT);
1132
1133         tfod.setMinResultConfidence(0.095f);
1134     }
1134
1135     public void runTfod() {
1136         super.loop();
1137         tfod.setSearchSpace(SearchSpace.FRONT);
1138
1139         tfod.setMinResultConfidence(0.095f);
1140     }
1140
1141     public void runTfodTensorFlow() {
1142         super.loop();
1143         tfod.setSearchSpace(SearchSpace.FRONT);
1144
1145         tfod.setMinResultConfidence(0.095f);
1146     }
1146
1147     public void runTfod() {
1148         super.loop();
1149         tfod.setSearchSpace(SearchSpace.FRONT);
1
```



- The `runTfodTensorFlow()` is run in the while loop till the Start button is pressed, the last value it records determines the last identified spike mark with the white pixel
 - More details on TensorFlow and how you could use your own team element is at https://ftc-docs.firstinspires.org/en/latest/programming_resources/vision/tensorflow_cs_2023/tensorflow-CS-2023.html
 - Once the Start button is pressed `opModelsActive()` becomes true and the autonomous code in `runAutonomousMode()` is executed. This method creates the trajectories and runs the automated motion of the robot
 - The method starts with setting coordinates for the different positions of the robot based on the start location selected and the identified spike mark. The coordinate system used here is based on the starting position of the robot. The front of the robot is $+x$ and to the right is $+y$. The robot starts at $x=0$, $y=0$, heading = 0 degrees (Heading is the angle the robot is rotated). The position of robot is indicated by the `Pose2d` object

- The trajectories are build and run in the next section in the `Actions.runBlocking(drive.actionBuilder(drive.pose), [motion].build())` methods

```

    //Move robot to dropPurplePixel based on identified Spike Mark Location
    Actions.runBlocking(
        drive.actionBuilder(drive.pose)
            .strafeToLinearHeading(moveBeyondTrussPose.position, moveBeyondTrussPose.heading)
            .strafeToLinearHeading(dropPurplePixelPose.position, dropPurplePixelPose.heading)
            .build());

    //TODO : Code to drop Purple Pixel on Spike Mark
    safeWaitSeconds( time: 1);

    //Move robot to midwayPose1
    Actions.runBlocking(
        drive.actionBuilder(drive.pose)
            .strafeToLinearHeading(midwayPose1.position, midwayPose1.heading)
            .build());

    //For Blue Right and Red Left, intake pixel from stack
    if (startPosition == START_POSITION.BLUE_RIGHT ||
        startPosition == START_POSITION.RED_LEFT) {
        Actions.runBlocking(
            drive.actionBuilder(drive.pose)
                .strafeToLinearHeading(midwayPose1a.position, midwayPose1a.heading)
                .strafeToLinearHeading(intakeStack.position, intakeStack.heading)
                .build());

        //TODO : Code to intake pixel from stack
        safeWaitSeconds( time: 1);

        //Move robot to midwayPose2 and to dropYellowPixelPose
        Actions.runBlocking(
    
```

- There are sections marked to add code for the other subsystem actions of the robot.
- Also we have added a `safeWaitSeconds()` to add waits in the motion. Use this instead of the blocking `sleep()` call.

The autonomous mode program provided does accomplish moving to the basic scoring operations required in the competition. However, you should modify it as needed to suit your strategy of scoring and how you plan to avoid running to your alliance partner's robot.

To program your own autonomous mode, use this provided code as reference and try out other motion primitives in <https://rr.brott.dev/docs/v1-0/builder-ref/>

Note: The autonomous mode example programmed here would not display correctly on the FTC Dashboard. This is since the coordinate system assumed here is robot centric, while the dashboard assumes the field centric coordinate system with Red to Blue direction as +x.

TeleOp Mode Code

`FTCWiresTeleOpMode.java` provides basic TeleOp to run the robot using roadrunner functionality. This is adapted from the `localizerTest.java`. You could add your TeleOp code for other subsystems here to utilize it.

Also, use this teleOpMode to determine the “Poses” of the robot to be used in autonomous mode. To do this, start the robot in the start location for the autonomous mode you are trying to program. Note the x, y, heading coordinates on the driver hub. It would be 0,0,0deg. Drive to the design position using your joystick. Note the x, y, heading coordinates from the driver hub and add in your autonomous program. Quite simple!

That's it : Enjoy programming Autonomous mode and score well!

Incase of any comments, questions or errors you found, or just need help - send a mail to
ftcwires@gmail.com or DM on [Instagram @ftcwires](#)

Or raise issues at <https://github.com/ftcwires/centerstage-road-runner/issues>

Acknowledgement :

Creators of Roadrunner: Acmerobotics, Ryan Brott

Learnroadrunner.com, Noah Bres

and

Team Hazmat 13201 (www.13201hazmat.org)

who created the FTCWires software platform.