/\* Copyright (c) 2017 FIRST. All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without modification,

\* are permitted (subject to the limitations in the disclaimer below) provided that

\* the following conditions are met:

\*

\* Redistributions of source code must retain the above copyright notice, this list

\* of conditions and the following disclaimer.

\*

\* Redistributions in binary form must reproduce the above copyright notice, this

\* list of conditions and the following disclaimer in the documentation and/or

\* other materials provided with the distribution.

\*

\* Neither the name of FIRST nor the names of its contributors may be used to endorse or

\* promote products derived from this software without specific prior written permission.

\*

\* NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE GRANTED BY THIS

\* LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

\* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

\* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE

\* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

\* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

\* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

\* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

\* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

\* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;

import com.qualcomm.robotcore.hardware.Servo;

import com.qualcomm.robotcore.hardware.Gamepad;

import com.qualcomm.robotcore.eventloop.opmode.TeleOp;

import com.qualcomm.robotcore.eventloop.opmode.Disabled;

import com.qualcomm.robotcore.hardware.DcMotor;

import com.qualcomm.robotcore.util.ElapsedTime;

import com.qualcomm.robotcore.util.Range;

/\*\*

\* This file contains an minimal example of a Linear "OpMode". An OpMode is a 'program' that runs in either

\* the autonomous or the teleop period of an FTC match. The names of OpModes appear on the menu

\* of the FTC Driver Station. When an selection is made from the menu, the corresponding OpMode

\* class is instantiated on the Robot Controller and executed.

\*

\* This particular OpMode just executes a basic Tank Drive Teleop for a two wheeled robot

\* It includes all the skeletal structure that all linear OpModes contain.

\*

\* Use Android Studios to Copy this Class, and Paste it into your team's code folder with a new name.

\* Remove or comment out the @Disabled line to add this opmode to the Driver Station OpMode list

\*/

@TeleOp(name="Basic: drive", group="Linear Opmode")

public class Team10218Drive extends LinearOpMode {

// name\_name

// NameName

//nameName

// Declare OpMode members.

DcMotor backleftDrive;

DcMotor backrightDrive;

DcMotor frontleftDrive;

DcMotor frontrightDrive;

DcMotor RightclawMotor;

DcMotor LeftclawMotor;

DcMotor TopclawMotor;

Servo ClawServo;

Servo spinyServo;

//DcMotor PetOmniwheelDrive;

@Override

public void runOpMode() {

telemetry.addData("Status", "Initialized");

telemetry.update();

// Initialize the hardware variables. Note that the strings used here as parameters

// to 'get' must correspond to the names assigned during the robot configuration

// step (using the FTC Robot Controller app on the phone).

backleftDrive = hardwareMap.get(DcMotor.class, "backleft");

backrightDrive = hardwareMap.get(DcMotor.class, "backright");

frontrightDrive = hardwareMap.get(DcMotor.class, "frontright");

frontleftDrive = hardwareMap.get(DcMotor.class, "frontleft");

RightclawMotor = hardwareMap.get(DcMotor.class, "RightclawMotor");

LeftclawMotor = hardwareMap.get(DcMotor.class, "LeftclawMotor");

TopclawMotor = hardwareMap.get(DcMotor.class, "TopclawMotor");

ClawServo = hardwareMap.get(Servo.class, "ClawServo");

spinyServo = hardwareMap.get(Servo.class, "spinyServo");

// Most robots need the motor on one side to be reversed to drive forward

// Reverse the motor that runs backwards when connected directly to the battery

frontrightDrive.setDirection(DcMotor.Direction.REVERSE);

backrightDrive.setDirection(DcMotor.Direction.REVERSE);

// Wait for the game to start (driver presses PLAY)

waitForStart();

// runtime.reset();

// run until the end of the match (driver presses STOP)

while (opModeIsActive()) {

frontleftDrive.setPower(gamepad1.left\_stick\_y);

frontrightDrive.setPower(gamepad1.right\_stick\_y);

backleftDrive.setPower(gamepad1.left\_stick\_y);

backrightDrive.setPower(gamepad1.right\_stick\_y);

// Left stick up and down moves the claw arm

LeftclawMotor.setPower(gamepad2.left\_stick\_y\*0.5);

RightclawMotor.setPower(gamepad2.left\_stick\_y\*0.5);

// The D-pad moves the claw forward and backward

if(gamepad2.dpad\_up){

TopclawMotor.setPower(1.0);

}

else {TopclawMotor.setPower(0.0);

}

if(gamepad2.dpad\_down){

TopclawMotor.setPower(-1.0);

}

else{TopclawMotor.setPower(0.0);

}

//a moves front servo

if(gamepad2.a){

spinyServo.setPosition(1);}

//b moves front servo back

else if(gamepad2.b){

spinyServo.setPosition(0);}

//x and y open and close claw

else if(gamepad2.x)

{

ClawServo.setPosition(1.0);

}

else if(gamepad2.y){

ClawServo.setPosition(0.00);

}

//End of Active Opmode

}

// // Choose to drive using either Tank Mode, or POV Mode

// // Comment out the method that's not used. The default below is POV.

// // POV Mode uses left stick to go forward, and right stick to turn.

// // - This uses basic math to combine motions and is easier to drive straight.

// double drive = -gamepad1.left\_stick\_y;

// double turn = gamepad1.right\_stick\_x;

// leftPower = Range.clip(drive + turn, -1.0, 1.0) ;

// rightPower = Range.clip(drive - turn, -1.0, 1.0) ;

// // Tank Mode uses one stick to control each wheel.

// // - This requires no math, but it is hard to drive forward slowly and keep straight.

// // leftPower = -gamepad1.left\_stick\_y ;

// // rightPower = -gamepad1.right\_stick\_y ;

// // Send calculated power to wheels

// leftDrive.setPower(leftPower);

// rightDrive.setPower(rightPower);

// // Show the elapsed game time and wheel power.

// telemetry.addData("Status", "Run Time: " + runtime.toString());

// telemetry.addData("Motors", "left (%.2f), right (%.2f)", leftPower, rightPower);

// telemetry.update();

}

}