/\* 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.CRServo;

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="Teleop bun", group="Linear Opmode")

public class MyTeleOp extends LinearOpMode {

// Declare OpMode members.

private ElapsedTime runtime = new ElapsedTime();

private Servo rt\_servo = null;

private final static double lts\_min\_range = 0.9;

private final static double lts\_max\_range = 0.7;

HardwarePushbot robot = new HardwarePushbot(); // Use a Pushbot's hardware

@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).

robot.leftDrive = hardwareMap.get(DcMotor.class, "left\_drive");

robot.rightDrive = hardwareMap.get(DcMotor.class, "right\_drive");

robot.liftMotor = hardwareMap.get(DcMotor.class, "lift\_motor");

robot.armMotor1 = hardwareMap.get(DcMotor.class, "arm\_motor1");

robot.armMotor2 = hardwareMap.get(DcMotor.class, "arm\_motor2");

robot.extendMotor = hardwareMap.get(DcMotor.class, "extend\_motor");

robot.clServo = hardwareMap.get(CRServo.class, "cl\_servo");

robot.rtServo = hardwareMap.get(Servo.class, "rt\_servo");

// 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

robot.leftDrive.setDirection(DcMotor.Direction.FORWARD);

robot.rightDrive.setDirection(DcMotor.Direction.REVERSE);

//robot.liftMotor.setMode(DcMotor.RunMode.RUN\_USING\_ENCODERS);

robot.armMotor1.setMode(DcMotor.RunMode.RUN\_USING\_ENCODERS);

robot.armMotor2.setMode(DcMotor.RunMode.RUN\_USING\_ENCODERS);

robot.armMotor1.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

robot.armMotor2.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

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

waitForStart();

runtime.reset();

double motorPower = 0.3;

double standard = 0.3;

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

while (opModeIsActive()) {

// Setup a variable for each drive wheel to save power level for telemetry

double leftPower;

double rightPower;

double armPower;

double liftPower;

double extendPower;

// 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.left\_stick\_x;

double lift = gamepad1.right\_stick\_y;

double arm = gamepad2.left\_stick\_y;

double extend = -gamepad2.right\_stick\_y;

if (gamepad1.y) {

standard = 0.6;

}

if (gamepad1.a) {

standard = 0.15;

}

if (gamepad1.x) {

standard = 0.4;

}

if(gamepad1.right\_trigger != 0){

motorPower = 0.8;

}

else

if(gamepad1.left\_trigger != 0 ){

motorPower = 0.25;

}

else

motorPower = standard;

if(gamepad2.right\_trigger !=0 ){

robot.rtServo.setPosition(lts\_min\_range);

}

if(gamepad2.left\_trigger !=0 ){

robot.rtServo.setPosition(lts\_max\_range);

}

if(gamepad2.y){

robot.clServo.setPower(1);

}

if(gamepad2.b){

robot.clServo.setPower(0);

}

if(gamepad2.a){

robot.clServo.setPower(-1);

}

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

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

armPower = Range.clip(arm, -0.5, 0.5);

liftPower = Range.clip(lift, -1,1);

extendPower = Range.clip(extend, -1,1);

// Send calculated power to wheels

robot.leftDrive.setPower(leftPower);

robot.rightDrive.setPower(rightPower);

robot.armMotor1.setPower(armPower);

robot.armMotor2.setPower(armPower);

robot.liftMotor.setPower(liftPower);

robot.extendMotor.setPower(extendPower);

// Show the elapsed game time and wheel power.

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

telemetry.update();

}

}

}