/\* This is where the code lives. If you have any questions about what the function of something is or what it does, comment your question on that section.

\*/

**package** org.firstinspires.ftc.robotcontroller.external.samples;

*/\*\**

*\**

*\* Motor channel: Left drive motor: "left\_drive"*

*\* Motor channel: Right drive motor: "right\_drive"*

*\* Motor channel: Manipulator drive motor: "left\_arm"*

*\* Servo channel: Servo to open left claw: "left\_hand"*

*\* Servo channel: Servo to open right claw: "right\_hand"*

*\*/*

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

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

**import** com.qualcomm.robotcore.hardware.DigitalChannel;

**import** com.qualcomm.robotcore.hardware.DistanceSensor;

**import** com.qualcomm.robotcore.hardware.Gyroscope;

**import** com.qualcomm.robotcore.util.Range;

**import** org.firstinspires.ftc.robotcontroller.external.samples.HardwarePushbot;

*/\*\**

*\* Created by David.Mitchell on 10/30/2017.*

*\*/*

@TeleOp(name=**"Pushbot: Articulating Claw"**, group=**"Pushbot"**)

**public class** ForReals **extends** LinearOpMode

{

**private** Gyroscope **imu**;

*//private DcMotor leftDrive;*

*//private DcMotor rightDrive;*

**private** DigitalChannel **digitalTouch**;

**private** DistanceSensor **sensorColorRange**;

*//Sprivate Servo servo1;*

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

*// could also use HardwarePushbotMatrix class.*

**public static final double *LMID\_SERVO*** = -0.5;

**public static final double *RMID\_SERVO*** = 0.5;

**public static final double *WRIST\_MID\_SERVO*** = 0.0;

**double clawOffset** = 0; *// Servo mid position*

**double wristOffset** = 0; *// Servo mid position*

**final double CLAW\_SPEED** = 0.05; *// sets rate to move servo*

@Override

**public void** runOpMode() {

*//robot.leftDrive = hardwareMap.get(DcMotor.class, "left\_Drive");*

*//robot.rightDrive = hardwareMap.get(DcMotor.class, "right\_Drive");*

*// 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.REVERSE);*

*//robot.rightDrive.setDirection(DcMotor.Direction.FORWARD);*

*/\* Initialize the hardware variables.*

*\* The init() method of the hardware class does all the work here*

*\*/*

**robot**.init(**hardwareMap**);

**robot**.**leftClaw**.setPosition(***LMID\_SERVO***);

**robot**.**rightClaw**.setPosition(***RMID\_SERVO***);

*// Send telemetry message to signify robot waiting;*

**telemetry**.addData(**"Say"**, **"Howdy Driver!"**); *//*

**telemetry**.update();

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

waitForStart();

*// 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;

*// 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 move*

*// - 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** / 2;

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

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

*// Send calculated power to wheels*

**robot**.**leftDrive**.setPower(leftPower);

**robot**.**rightDrive**.setPower(rightPower);

*// Use gamepad left & right Bumpers to open and close the claw*

**if** (**gamepad2**.**right\_bumper**)

**clawOffset** += **CLAW\_SPEED**;

**else if** (**gamepad2**.**left\_bumper**)

**clawOffset** -= **CLAW\_SPEED**;

*// Move both servos to new position. Assume servos are mirror image of each other.*

**clawOffset** = Range.*clip*(**clawOffset**, -0.75, 0.5);

**robot**.**leftClaw**.setPosition(***LMID\_SERVO*** + **clawOffset**);

**robot**.**rightClaw**.setPosition(***RMID\_SERVO*** - **clawOffset**);

**robot**.**leftArm**.setPower(**gamepad2**.**left\_stick\_y**/2);

*/\*if (gamepad1.y)*

*robot.leftArm.setPower(robot.ARM\_UP\_POWER);*

*else if (gamepad1.a)*

*robot.leftArm.setPower(robot.ARM\_DOWN\_POWER);*

*else*

*robot.leftArm.setPower(0.0);*

*\*/*

*// Articulating Claw*

**wristOffset** = (**gamepad2**.**right\_stick\_y**);

**wristOffset** = Range.*clip*(**wristOffset**, -0.5, 0.5);

**robot**.**wrist**.setPosition(***WRIST\_MID\_SERVO*** + **wristOffset**);

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

**telemetry**.addData(**"claw"**, **"Left= %.2f, right= %.2f"**

, ***LMID\_SERVO*** + **clawOffset**,***RMID\_SERVO*** - **clawOffset**);

**telemetry**.addData(**"Status"**, **"Running"**);

**telemetry**.update();

**if** (**gamepad2**.**b**) {

clawTest();

}

*// Pace this loop so jaw action is reasonable speed.*

sleep(50);

}

}

**private void** clawTest ()

{

{

**robot**.**leftClaw**.setPosition(***LMID\_SERVO***);

**robot**.**rightClaw**.setPosition(***RMID\_SERVO***);

**clawOffset**=0;

}

}

}