```
/*
Copyright (c) 2016 Robert Atkinson

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 Robert Atkinson nor the names of his 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 FITNESSFOR 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.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.util.Range;

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

/**
 * This OpMode uses the common Babybot hardware class to define the devices on the robot.
 * All device access is managed through the HardwareBabybot class.
 * The code is structured as a LinearOpMode
 *
 Game Pad 1 Controls:
 gamepad1.a
 gamepad1.b
 gamepad1.y
 gamepad1.x
 gamepad1.left_bumper
 gamepad1.left_trigger > 0.25
 gamepad1.left_stick_y
 gamepad1.right_bumper
 gamepad1.right_trigger > 0.25
 gamepad1.right_stick_y

 Game Pad 2 Controls:
 gamepad2.a
 gamepad2.b
 gamepad2.y
 gamepad2.x
 gamepad2.left_bumper
 gamepad2.left_trigger > 0.25
 gamepad2.left_stick_y
 gamepad2.right_bumper
 gamepad2.right_trigger > 0.25
 gamepad2.right_stick_y

 Robot Component Configuration:
 Moror Controller 1
 "motor_1" is on the right side of the bot.
 "motor_2" is on the left side of the bot and reversed.

 *
 * 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="BabyBot Teleop POV", group="Abdul")
///@Disabled
public class BabyTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwareBabybot robot           = new HardwareBabybot();   // Use a Babybot's hardware
                                                               // could also use HardwareBabyBot class.

    final static double ARM_MIN_RANGE  = 0.10; //turning the obj it catches
    final static double ARM_MAX_RANGE  = 0.95;

    final static double CLAW_MIN_RANGE  = 0.10; //catching an obj
    final static double CLAW_MAX_RANGE  = 0.90;


    // position of the claw servo
    double armPosition;
    double clawPosition;

    // amount to change the claw servo position by
    double clawDelta = 0.1;
    double armDelta = 0.1;



    @Override
    public void runOpMode() throws InterruptedException {


        /* Initialize the hardware variables.
         * The init() method of the hardware class does all the work here
         */
        robot.init(hardwareMap);

        // Send telemetry message to signify robot waiting;
        telemetry.addData("Say", "Hello I am BabyBot");    //
        telemetry.update();

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

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

            // Run wheels in POV mode (note: The joystick goes negative when pushed forwards, so negate it)
            // In this mode the Left stick moves the robot fwd and back, the Right stick turns left and right.
            // tank drive
            // note that if y equal -1 then joystick is pushed all of the way forward.
            float left = -gamepad1.left_stick_y;
            float right = -gamepad1.right_stick_y;

            // clip the right/left values so that the values never exceed +/- 1
            right = Range.clip(right, -1, 1);
            left = Range.clip(left, -1, 1);

            // scale the joystick value to make it easier to control
            // the robot more precisely at slower speeds.
            right = (float)scaleInput(right);
            left =  (float)scaleInput(left);

            // write the values to the motors
            robot.leftMotor.setPower(left);
            robot.rightMotor.setPower(right);



//////////////////////
            /// New Claw Positions the turner
            if (gamepad1.x) {

                clawPosition = 0.20;

            }
            else{
                clawPosition = 0.9;

            }
```

```
/// New Arm Positions the turner
            if (gamepad1.a) {

                armPosition = armDelta + armPosition;
                if (armPosition > ARM_MAX_RANGE){armPosition = ARM_MAX_RANGE;}
            }
            if (gamepad1.b) {

                armPosition = armPosition - armDelta ;
                if (armPosition < ARM_MIN_RANGE){armPosition = ARM_MIN_RANGE;}
            }
            else{


            }

            // write the values to the motors
            robot.claw.setPosition(clawPosition);
            robot.arm.setPosition(armPosition);



            // Send telemetry message to signify robot running;
            telemetry.addData("claw",  "Offset = %.2f", clawPosition);
            telemetry.addData("arm",  "Offset = %.2f", armPosition);
            telemetry.addData("left",  "%.2f", left);
            telemetry.addData("right", "%.2f", right);
            telemetry.update();

            // Pause for metronome tick.  40 mS each cycle = update 25 times a second.
            robot.waitForTick(40);

            idle(); // Always call idle() at the bottom of your while(opModeIsActive()) loop
        }


    }
    /*
         * This method scales the joystick input so for low joystick values, the
         * scaled value is less than linear.  This is to make it easier to drive
         * the robot more precisely at slower speeds.
         */
    double scaleInput(double dVal)  {
        double[] scaleArray = { 0.0, 0.05, 0.09, 0.10, 0.12, 0.15, 0.18, 0.24,
                0.30, 0.36, 0.43, 0.50, 0.60, 0.72, 0.85, 1.00, 1.00 };

        // get the corresponding index for the scaleInput array.
        int index = (int) (dVal * 16.0);
        if (index < 0) {
            index = -index;
        } else if (index > 16) {
            index = 16;
        }

        double dScale = 0.0;
        if (dVal < 0) {
            dScale = -scaleArray[index];
        } else {
            dScale = scaleArray[index];
        }

        return dScale;
    }

}
```