Simple Tele-Op

Joshua Phillips, Error 404 Robotics

July 2017

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.util.Range;


import static com.qualcomm.robotcore.hardware.DcMotor.RunMode.RUN_USING_ENCODERS;

@TeleOp(name="Wannabe Teleop", group="Teleop")

public class wannabeTeleop extends OpMode {
    ////////////////////////////////////////////
    // This is the Teleop program for driver control.
    ////////////////////////////////////////////
  DcMotor rightFront;
  DcMotor leftFront;
  Servo flashlightServo;
  Servo flipperServo;
  double powerval;
  double flashlight=0;
  double flipper=0;
  public wannabeTeleop() {}
  @Override
```

Make sure these lines are added at the very top. These lines will enable you to use libraries easily while programming.

Must be added before public class. This enables you to actually use the program when using the driver station.

```
public void init() {
  telemetry.addData ("0", "I AM HERE");
  flipperServo = hardwareMap.servo.get("flipper");
   flashlightServo= hardwareMap.servo.get("flashlight");

  leftFront = hardwareMap.dcMotor.get("left");
  rightFront = hardwareMap.dcMotor.get("right");

  leftFront.setMode(RUN_WITHOUT_ENCODER);
  rightFront.setMode(RUN_WITHOUT_ENCODER);
  leftFront.setDirection(DcMotor.Direction.REVERSE);
  rightFront.setDirection(DcMotor.Direction.FORWARD);
  telemetry.addData("","V 2");

    flipper=0.5;
    flashlight=0.5;
}
@Override
```

These allow the robot controller to find the phone in the configuration file. Otherwise, the phone will complain that it can't access the motor, servo, etc.

Tells motors what they should rely on when being used. Also tells motor which direction to turn.

```
public void loop() {
////////////////////////////////////////
/////Drive Train//////////////////////////
////////////////////////////////////////

  float leftStick = gamepad1.left_stick_y;          //reading raw values from the joysticks
  float rightStick = gamepad1.right_stick_y;         //reading raw values from the joysticks

  //clip the right/left values so that the values never exceed +/- 1.
  leftStick = (float) scaleInput(leftStick);
  rightStick = (float) scaleInput(rightStick);

//////////////////////////////////////////////////////////
///////////flashlight/////////////////
////////////////////////////////////////

 if(gamepad1.y){
     flipper+=0.01;
  }
  else if(gamepad1.a){
     flipper-=0.01;
  }
  else if(gamepad1.x){
     flashlight-=0.01;
  }
  else if(gamepad1.b){
    flashlight+=0.01;
  }
  else{
   // leftVal = 0.0;
   // rightVal=0.0;
  }
    flipper = Range.clip(flipper, 0, 1);
    flashlight = Range.clip(flashlight, 0, 1);

    flipperServo.setPosition(flipper);
  flashlightServo.setPosition(flashlight);
```

Sets a small variable to the input gotten from the joysticks. Scales the input, though as fast as the loop method refreshes, the scaling is barely noticeable.

Programs two servos for a flashlight, turning the light and turning it “off/on” with a small flipper.

Makes sure the number the servos are set to never exceed the readable range of servos.

Sets servo position.

```
    rightStick=(float)scaleInput(rightStick);
    leftStick=(float)scaleInput(leftStick);
    rightFront.setPower(rightStick);
  leftFront.setPower(leftStick);

}
@Override
public void stop() {
}
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);

  // index should be positive.
  if (index < 0) {
    index = -index;
  }

  // index cannot exceed size of array minus 1.
  if (index > 16) {
    index = 16;
  }

  // get value from the array.
  double dScale;
  if (dVal < 0) {
    dScale = -scaleArray[index];
  } else {
    dScale = scaleArray[index];
  }

  // return scaled value.
  return dScale;
}
```

Sets power to motors.

scaleInput method that scales the power set to the motors sixteen times, and if the value input is negative, it then negates that input to be used correctly in the method.