/\* 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.Disabled;

import java.util.concurrent.TimeUnit;

import java.util.Timer;

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

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

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

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

import com.qualcomm.robotcore.hardware.DcMotor;

import com.qualcomm.robotcore.util.ElapsedTime;

import com.qualcomm.robotcore.util.Range;

/\*\*

\* This file contains an example of an iterative (Non-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 iterative 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

\*/

@Autonomous(name="Auto", group="Auto")

public class Team10218auto extends OpMode{

private ElapsedTime runtime = new ElapsedTime();

DcMotor backleftDrive;

DcMotor backrightDrive;

DcMotor frontleftDrive;

DcMotor frontrightDrive;

@Override

public void init() {

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

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

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

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

// 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);

// Tell the driver that initialization is complete.

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

}

/\*

\* Code to run REPEATEDLY after the driver hits INIT, but before they hit PLAY

\*/

@Override

public void init\_loop() {

}

/\*

\* Code to run ONCE when the driver hits PLAY

\*/

@Override

public void start() {

runtime.reset();

//Move forward past platform

while (runtime.seconds() < 2) {

frontleftDrive.setPower(1.0);

frontrightDrive.setPower(1.0);

backleftDrive.setPower(1.0);

backrightDrive.setPower(1.0);

}

runtime.reset();

while (runtime.seconds() < 0.75) {

//Make a right turn

frontleftDrive.setPower(1.0);

frontrightDrive.setPower(0.0);

backleftDrive.setPower(1.0);

backrightDrive.setPower(0.0);

}

runtime.reset();

while (runtime.seconds() < .6){

//Move forward

frontleftDrive.setPower(1.0);

frontrightDrive.setPower(1.0);

backleftDrive.setPower(1.0);

backrightDrive.setPower(1.0);

}

runtime.reset();

while (runtime.seconds() < 0.75){

//Make a right turn

frontleftDrive.setPower(1.0);

frontrightDrive.setPower(0.0);

backleftDrive.setPower(1.0);

backrightDrive.setPower(0.0);

}

runtime.reset();

//Move forward

while (runtime.seconds() < 3){

frontleftDrive.setPower(1.0);

frontrightDrive.setPower(1.0);

backleftDrive.setPower(1.0);

backrightDrive.setPower(1.0);

}

runtime.reset();

}

/\*

\* Code to run REPEATEDLY after the driver hits PLAY but before they hit STOP

\*/

@Override

public void loop() {

}

/\*

\* Code to run ONCE after the driver hits STOP

\*/

@Override

public void stop() {

frontleftDrive.setPower(0.0);

frontrightDrive.setPower(0.0);

backleftDrive.setPower(0.0);

backrightDrive.setPower(0.0);

}

}