```
/*
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.Autonomous;
import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.util.ElapsedTime;

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

/**
 * This file illustrates the concept of driving a path based on time.
 * It uses the common Babybot hardware class to define the drive on the robot.
 * The code is structured as a LinearOpMode
 *
 *
 *   The desired path in this example is:
 *   - Drive forward for 3 seconds
 *   - Spin right for 1.3 seconds
 *   - Drive Backwards for 1 Second
 *   - Stop and close the claw.
 *
 *  The code is written in a simple form with no optimizations.
 *  However, there are several ways that this type of sequence could be streamlined,
 *
 * 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="Babybot: Auto By Time", group="Abdul")
//@Disabled
public class BabybotAutoDriveByTime_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwareBabybot         robot   = new HardwareBabybot();   // Use a Babybot's hardware
    private ElapsedTime     runtime = new ElapsedTime();


    static final double     FORWARD_SPEED = 0.6;
    static final double     TURN_SPEED    = 0.5;

    @Override
    public void runOpMode() throws InterruptedException {

        /*
         * Initialize the drive system 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("Status", "Ready to run");    //
        telemetry.update();

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

        // Step through each leg of the path, ensuring that the Auto mode has not been stopped along the way

        // Step 1:  Drive forward for 3 seconds
        robot.leftMotor.setPower(FORWARD_SPEED);
        robot.rightMotor.setPower(FORWARD_SPEED);
        runtime.reset();
        while (opModeIsActive() && (runtime.seconds() < 3.0)) {
            telemetry.addData("Path", "Leg 1: %2.5f S Elapsed", runtime.seconds());
            telemetry.update();
            idle();
        }

        // Step 2:  Spin right for 1.3 seconds
        robot.leftMotor.setPower(TURN_SPEED);
        robot.rightMotor.setPower(-TURN_SPEED);
        runtime.reset();
        while (opModeIsActive() && (runtime.seconds() < 1.3)) {
            telemetry.addData("Path", "Leg 2: %2.5f S Elapsed", runtime.seconds());
            telemetry.update();
            idle();
        }

        // Step 3:  Drive Backwards for 1 Second
        robot.leftMotor.setPower(-FORWARD_SPEED);
        robot.rightMotor.setPower(-FORWARD_SPEED);
        runtime.reset();
        while (opModeIsActive() && (runtime.seconds() < 1.0)) {
            telemetry.addData("Path", "Leg 3: %2.5f S Elapsed", runtime.seconds());
            telemetry.update();
            idle();
        }

        // Step 4:  Stop and close the claw.
        robot.leftMotor.setPower(0);
        robot.rightMotor.setPower(0);
        robot.claw.setPosition(1.0);
        robot.arm.setPosition(0.0);

        telemetry.addData("Path", "Complete");
        telemetry.update();
        sleep(1000);
        idle();
    }
}
```