Navegação do robô - OnBot Java

Introdução a navegação do robô

Como indicado na seção Olá Robô - Controle do Robô, o controle de robôs assume muitas formas diferentes. Um dos tipos de controle a ser considerado para robôs com drivetrains é a navegação do robô.

A navegação do robô como conceito depende do tipo de drivetrain e do tipo de modo de operação. Por exemplo, o código para controlar um drivetrain mecanum difere do código usado para controlar um drivetrain diferencial. Também há diferença entre codificar para direção teleoperada, com um gamepad, e codificar para autonomia, onde cada movimento do robô deve ser definido dentro do código.

A próxima seção passa pelos fundamentos da programação para um drivetrain diferencial, bem como como configurar um código de drivetrain teleoperado no estilo arcade. Os conceitos e a lógica destacados nesta seção serão aplicáveis à seção de controle autônomo Elapsed Time.

Seções	Objetivos da seção
Noções básicas de programação de transmissão	O que considerar quando estiver programando uma transmissão e como aplicar.

Noções básicas de programação de transmissão

Programação de motores de transmissão

Comece criando um Op Mode chamado de DualDrive.

> Visite a seção OnBot Java para obter mais informações sobre como criar um modo operacional (op mode). O op mode abaixo concentra-se apenas no mapeamento de hardware dos motores relevantes do sistema de transmissão.

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

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;

@TeleOp
public class DualDrive extends LinearOpMode {
  private DcMotor rightmotor;
  private DcMotor leftmotor;

  @Override
  public void runOpMode() {

        rightmotor = hardwareMap.get(DcMotor.class, "rightmotor");
        leftmotor = hardwareMap.get(DcMotor.class, "leftmotor");

        waitForStart();

        while (opModeIsActive()) {

        }
    }
}
```

Dado que o foco desta seção é criar um sistema de transmissão funcional no código, vamos começar adicionando rightmotor.setPower(1); e leftmotor.setPower(1); ao loop while do op mode.

```
while (opModeIsActive()) {
    rightmotor.setPower(1);
    leftmotor.setPower(1);
    }
```

> Antes de prosseguir, tente executar o código conforme está e considere as seguintes perguntas:
>
> - Qual comportamento o robô está exibindo?
> - Em que direção o robô está girando?

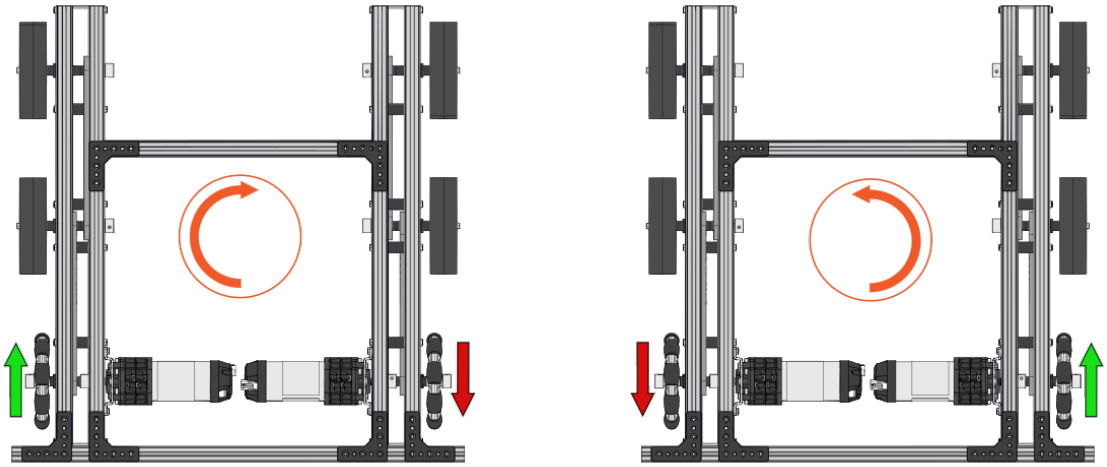
> Quando os motores funcionam em velocidades diferentes, eles giram em torno do ponto central de pivô. No entanto, os motores estão ambos configurados com uma potência (ou ciclo de trabalho) de 1?

Os motores de corrente contínua são capazes de girar em duas direções diferentes, dependendo do fluxo de corrente: no sentido horário e no sentido anti-horário. Ao usar um valor de potência positivo, o Control Hub envia corrente para o motor para fazê-lo girar no sentido horário. Com a classe Bot e o código atual, ambos os motores estão atualmente configurados para girar no sentido horário. No entanto, se você colocar o robô em blocos e executar o código novamente, verá que os motores giram em direções opostas. Com a maneira espelhada como os motores são montados no sistema de transmissão, um motor é naturalmente o inverso do outro. Por que o motor inverso faz o robô girar em círculos? Tanto a velocidade quanto a direção de rotação das rodas afetam a direção geral em que o robô se move. Neste caso, ambos os motores foram designados para ter a mesma potência e direção, mas a forma como os motores transferem o movimento para as rodas faz com que o robô gire em vez de avançar.

> Consulte a seção de Introdução ao Movimento da REV Robotics para obter mais informações sobre a mecânica de transferência de movimento e potência.

Na caixa de informações anterior, foi solicitado que você determinasse em que direção o robô girava. O robô pivota na direção do motor invertido. Por exemplo, quando o motor direito é o motor invertido, o robô irá pivotar para a direita. Se o motor esquerdo for o motor invertido, o robô pivotará para a esquerda.

O efeito dos motores do sistema de transmissão no movimento]



Para a classe Bot, em que o robô pivota para a direita, o motor direito será invertido. Adicione a linha rightmotor.setDirection(DcMotorSimple.Direction.REVERSE); ao op mode, logo abaixo das declarações de variáveis.

```
public void runOpMode() {
        float x;
        double y;

        rightmotor = hardwareMap.get(DcMotor.class, "rightmotor");
        leftmotor = hardwareMap.get(DcMotor.class, "leftmotor");

        rightmotor.setDirection(DcMotorSimple.Direction.REVERSE);

        waitForStart();

        while (opModeIsActive()) {
            rightmotor.setPower(1);
            leftmotor.setPower(1);
          }
    }
```

Adicionar a linha de código rightmotor.setDirection(DcMotorSimple.Direction.REVERSE); inverte a direção do motor direito. Agora, ambos os motores consideram a mesma direção como a direção para a frente.

Estilo Arcade de condução

Lembre-se de que quando os motores estavam girando em direções opostas, o robô girava em círculos. Essa mesma lógica será usada para controlar o robô usando o estilo arcade de controle mencionado na seção Hello Robot - Autonomous Robot.

Programando com controle

Para começar, crie duas variáveis double chamadas drive e turn.

```
public void runOpMode() {
        double x;
        double y;

        rightmotor = hardwareMap.get(DcMotor.class, "rightmotor");
        leftmotor = hardwareMap.get(DcMotor.class, "leftmotor");

        rightmotor.setDirection(DcMotorSimple.Direction.REVERSE);

        waitForStart();
```

Atribua a y o valor -gamepad1.right_stick_y;, que corresponde ao eixo y do joystick direito.

> Lembre-se de que os valores positivos/negativos inseridos pelo eixo y do gamepad são inversos dos valores positivos/negativos do motor.

Atribua x como x = gamepad1.right_stick_x;, que é o eixo x do joystick direito do gamepad. O eixo x do joystick não precisa ser invertido.

```
while (opModeIsActive()) {
        x = gamepad1.right_stick_x;
        y = -gamepad1.right_stick_y;

        rightmotor.setPower(1);
        leftmotor.setPower(1);
        }
```

Definir x = gamepad1.right_stick_x; e y = -gamepad1.right_stick_y; atribui valores do joystick do gamepad a x e y. Como mencionado anteriormente, o joystick fornece valores ao longo de um sistema de coordenadas bidimensional. y recebe o valor do eixo y e x recebe o valor do eixo x. Ambos os eixos geram valores entre -1 e 1.

Para entender melhor, considere a seguinte tabela. A tabela mostra o valor esperado gerado ao mover o joystick completamente em uma direção, ao longo do eixo. Por exemplo, quando o joystick é empurrado completamente na direção para cima, os valores das coordenadas são (0,1).

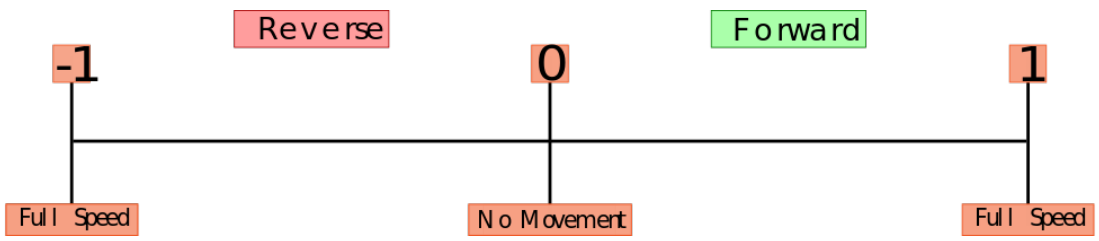
> A tabela abaixo assume que o valor de y foi invertido no código

Joystick Direction	x	y
	0	1
	0	-1
	-1	0
	1	0

Agora que você tem uma compreensão melhor de como o movimento físico do gamepad afeta as entradas numéricas fornecidas ao seu sistema de controle, é hora de considerar como controlar o sistema de propulsão usando o joystick.

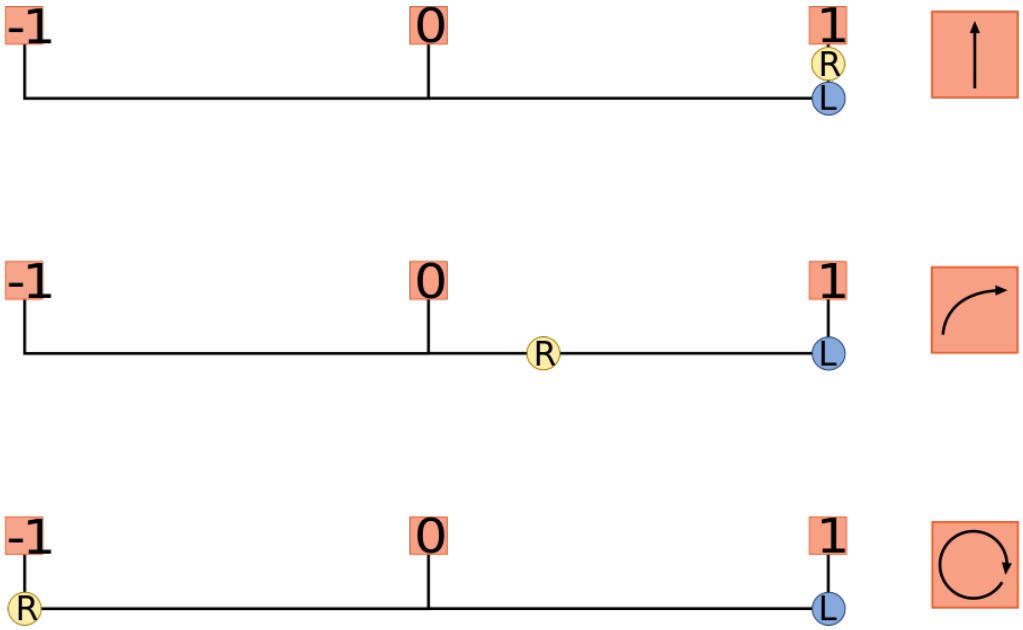
> Lembre-se, da seção de Programação dos Motores da Transmissão, que a velocidade e a direção de um motor desempenham um papel importante na forma como a transmissão se move.

Os resultados numéricos para setPower determinam a velocidade e direção dos motores. Por exemplo, quando ambos os motores são ajustados para 1, eles se movem na direção para frente a toda velocidade (ou 100% do ciclo de trabalho). Assim como nos gamepads, o valor numérico para setPower está em uma faixa de -1 a 1. O valor absoluto do número atribuído determina a porcentagem do ciclo de trabalho. Como exemplo, 0,3 e -0,3 indicam ambos que o motor está operando com um ciclo de trabalho de 30%. O sinal do número indica a direção em que o motor está girando. Para entender melhor, considere o seguinte gráfico.



Quando um motor recebe um valor de setPower entre -1 e 0, o motor girará na direção que considera ser reversa. Quando um motor recebe um valor entre 0 e 1, ele girará para frente.

Na seção de Programação dos Motores da Transmissão, discutiu-se que um robô gira quando os motores estão se movendo em direções opostas. No entanto, isso tem mais a ver com a velocidade e direção. Numericamente, uma transmissão diferencial vai girar para a direita quando o valor de setPower para o motor direito for menor do que o do motor esquerdo. Isso é exemplificado no seguinte exemplo.



Quando ambos os motores estão configurados como rightmotor.setPower(1); leftmotor.setPower(1);, o robô irá se mover a toda velocidade em linha reta. No entanto, quando o rightmotor está se movendo na mesma direção, mas a uma velocidade menor, como rightmotor.setPower(0.3); leftmotor.setPower(1);, o robô irá virar ou girar para a direita. Isso provavelmente resultará em um movimento de arco que não é tão acentuado quanto um pivô completo. Em contraste, quando o rightmotor está configurado para velocidade máxima, mas na direção oposta à do leftmotor, o robô gira para a direita. Portanto, matematicamente, o seguinte é considerado verdadeiro:

Expressão	Resultado
rightmotor.setPower = leftmotor.setPower	Frente ou trás
rightmotor.setPower > leftmotor.setPower	Rotação para esquerda
rightmotor.setPower < leftmotor.setPower	Rotação para direita

Como mencionado anteriormente, gamepad1.right_stick_y e gamepad1.right_stick_x enviam valores para o sistema de controle a partir do joystick do gamepad. Em contraste, a função setPower interpreta informações numéricas definidas no código e envia a corrente apropriada para os motores para ditar o comportamento dos motores.

Em um sistema de direção de arcade, as seguintes entradas (direções) do joystick precisam corresponder às seguintes saídas (valores de potência do motor).

Joystick Direction	(X,Y)	rightmotor	leftmotor
	(0,1)	1	1
	(0,-1)	-1	-1
	(-1,0)	1	-1
	(1,0)	-1	1

Para obter as saídas expressas na tabela acima, os valores do gamepad devem ser atribuídos a cada motor de maneira significativa, onde princípios algébricos podem ser usados para determinar as duas fórmulas necessárias para obter os valores. No entanto, as fórmulas estão fornecidas abaixo.

leftMotor = y - x

rightMotor = y + x

Em vez de setPower(1);, ambos os motores podem ser configurados com as fórmulas mencionadas anteriormente. Por exemplo, o motor direito pode ser configurado como rightmotor.setPower(y - x);

```
while (opModeIsActive()) {
        x = gamepad1.right_stick_x;
        y = -gamepad1.right_stick_y;

        rightmotor.setPower(y-x);
        leftmotor.setPower(y+x);
        }
```

Com isso, você agora possui um controle remoto funcional com sistema de direção arcade. A partir daqui, você pode começar a adicionar o mapeamento de hardware para outras peças do hardware

do robô. Abaixo está um esboço do código esperado para a classe Bot com um mapeamento completo de hardware.

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

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.hardware.Blinker;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.hardware.Gyroscope;
import com.qualcomm.robotcore.hardware.DigitalChannel;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;
import com.qualcomm.robotcore.util.ElapsedTime;

@TeleOp

public class DualDrive extends LinearOpMode {
    private Blinker control_Hub;
    private DcMotor arm;
    private Servo claw;
    private Gyroscope imu;
    private DcMotor leftmotor;
    private DcMotor rightmotor;
    private DigitalChannel touch;


    @Override
    public void runOpMode() {
        double x;
        double y;

        control_Hub = hardwareMap.get(Blinker.class, "Control Hub");
        arm = hardwareMap.get(DcMotor.class, "arm");
        claw = hardwareMap.get(Servo.class, "claw");
        imu = hardwareMap.get(Gyroscope.class, "imu");
        leftmotor = hardwareMap.get(DcMotor.class, "leftmotor");
        rightmotor = hardwareMap.get(DcMotor.class, "rightmotor");
        touch = hardwareMap.get(DigitalChannel.class, "touch");
```

```
        rightmotor.setDirection(DcMotorSimple.Direction.REVERSE);

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

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

        // run until the end of the match (driver presses STOP)
        while (opModeIsActive()) {
            x = gamepad1.right_stick_x;
            y = -gamepad1.right_stick_y;

            rightmotor.setPower(y-x);
            leftmotor.setPower(y+x);

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

        }
    }
```

Revisão #1
Criado 18 dezembro 2023 18:33:40 por Enzo
Atualizado 18 dezembro 2023 19:14:21 por Enzo