Orientação ao campo

Com a condução Mecanum centrada no campo, o joystick de translação controla a direção do robô em relação ao campo, em vez do quadro do robô. Isso é preferido por alguns *drivers* e torna algumas ações evasivas mais fáceis, pois é possível girar enquanto se translaciona em uma direção específica. Para fazer isso, os componentes x/y dos joysticks são rotacionados no sentido oposto ao ângulo do robô, que é fornecido pelo IMU.

Há um IMU dentro dos Control Hubs (e modelos mais antigos de Expansion Hubs). Diferente de outros hardwares, é recomendado fazer mais do que apenas usar hardwareMap.get() para começar a usá-lo. Vale notar que, por padrão, ele é configurado como "imu" ao criar uma nova configuração.

Consulte a página de documentação do FTC sobre a interface IMU para mais informações. A maneira como o IMU será inicializado aqui é:

```
// Retrieve the IMU from the hardware map
imu = hardwareMap.get(IMU.class, "imu");
// Adjust the orientation parameters to match your robot
IMU.Parameters parameters = new IMU.Parameters(new RevHubOrientationOnRobot(
        RevHubOrientationOnRobot.LogoFacingDirection.UP,
        RevHubOrientationOnRobot.UsbFacingDirection.FORWARD));
// Without this, the REV Hub's orientation is assumed to be logo up / USB forward
imu.initialize(parameters);
```

O ângulo precisa ser lido a cada loop. Além disso, embora o IMU mantenha uma posição zero consistente entre os OpModes (incluindo entre autônomo e teleop), adicionar uma função para redefinir o ângulo é importante para corrigir o erro e porque o zero pode mudar devido a alguns tipos de desconexões.

> Objetos BNO055 redefinirão o zero do IMU quando o método initialize for chamado. A classe BNO055 não é recomendada para novos desenvolvimentos. A classe IMU não possui esse comportamento e é a substituição adequada a partir da versão SDK v8.1.

```
// This button choice was made so that it is hard to hit on accident,
// it can be freely changed based on preference.
// The equivalent button is start on Xbox-style controllers.
if (gamepad1.options) {
    imu.resetYaw();
}
```

```
double botHeading = imu.getRobotYawPitchRollAngles().getYaw(AngleUnit.RADIANS);
```

Em seguida, os valores do joystick de translação precisam ser rotacionados no sentido oposto à rotação do robô. O IMU retorna o heading (ângulo de orientação), mas precisamos rotacionar o movimento no sentido oposto à rotação do robô, então utilizamos o valor negativo do ângulo. Os valores do joystick formam um vetor, e rotacionar um vetor em 2D requer a seguinte fórmula (provada aqui), onde:

- x1 e y1 são os componentes do vetor original;
- θ é o ângulo de rotação;
- x2 e y2 são os componentes do vetor resultante.

```
// Rotate the movement direction counter to the bot's rotation
double rotX = x * Math.cos(-botHeading) - y * Math.sin(-botHeading);
double rotY = x * Math.sin(-botHeading) + y * Math.cos(-botHeading);
```

Então, esses valores rotacionados podem ser inseridos na cinemática da Mecanum mostrada anteriormente.

```
double denominator = Math.max(Math.abs(rotY) + Math.abs(rotX) + Math.abs(rx), 1);
double frontLeftPower = (rotY + rotX + rx) / denominator;
double backLeftPower = (rotY - rotX + rx) / denominator;
double frontRightPower = (rotY - rotX - rx) / denominator;
double backRightPower = (rotY + rotX - rx) / denominator;
```

Código de exemplo

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

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.hardware.IMU;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;
import com.qualcomm.hardware.rev.RevHubOrientationOnRobot;
import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;

@TeleOp
public class FieldCentricMecanumTeleOp extends LinearOpMode {
```

```
@Override
public void runOpMode() throws InterruptedException {
    // Declare our motors
    // Make sure your ID's match your configuration
    DcMotor frontLeftMotor = hardwareMap.dcMotor.get("frontLeftMotor");
    DcMotor backLeftMotor = hardwareMap.dcMotor.get("backLeftMotor");
    DcMotor frontRightMotor = hardwareMap.dcMotor.get("frontRightMotor");
    DcMotor backRightMotor = hardwareMap.dcMotor.get("backRightMotor");

    // Reverse the right side motors. This may be wrong for your setup.
    // If your robot moves backwards when commanded to go forwards,
    // reverse the left side instead.
    // See the note about this earlier on this page.
    frontRightMotor.setDirection(DcMotorSimple.Direction.REVERSE);
    backRightMotor.setDirection(DcMotorSimple.Direction.REVERSE);

    // Retrieve the IMU from the hardware map
    IMU imu = hardwareMap.get(IMU.class, "imu");
    // Adjust the orientation parameters to match your robot
    IMU.Parameters parameters = new IMU.Parameters(new RevHubOrientationOnRobot(
            RevHubOrientationOnRobot.LogoFacingDirection.UP,
            RevHubOrientationOnRobot.UsbFacingDirection.FORWARD));
    // Without this, the REV Hub's orientation is assumed to be logo up / USB forward
    imu.initialize(parameters);

    waitForStart();

    if (isStopRequested()) return;

    while (opModeIsActive()) {
        double y = -gamepad1.left_stick_y; // Remember, Y stick value is reversed
        double x = gamepad1.left_stick_x;
        double rx = gamepad1.right_stick_x;

        // This button choice was made so that it is hard to hit on accident,
        // it can be freely changed based on preference.
        // The equivalent button is start on Xbox-style controllers.
        if (gamepad1.options) {
            imu.resetYaw();
        }
```

```
            double botHeading = imu.getRobotYawPitchRollAngles().getYaw(AngleUnit.RADIANS);

            // Rotate the movement direction counter to the bot's rotation
            double rotX = x * Math.cos(-botHeading) - y * Math.sin(-botHeading);
            double rotY = x * Math.sin(-botHeading) + y * Math.cos(-botHeading);

            rotX = rotX * 1.1;  // Counteract imperfect strafing

            // Denominator is the largest motor power (absolute value) or 1
            // This ensures all the powers maintain the same ratio,
            // but only if at least one is out of the range [-1, 1]
            double denominator = Math.max(Math.abs(rotY) + Math.abs(rotX) + Math.abs(rx), 1);
            double frontLeftPower = (rotY + rotX + rx) / denominator;
            double backLeftPower = (rotY - rotX + rx) / denominator;
            double frontRightPower = (rotY - rotX - rx) / denominator;
            double backRightPower = (rotY + rotX - rx) / denominator;

            frontLeftMotor.setPower(frontLeftPower);
            backLeftMotor.setPower(backLeftPower);
            frontRightMotor.setPower(frontRightPower);
            backRightMotor.setPower(backRightPower);
        }
    }
}
```

Revisão #5
Criado 11 dezembro 2024 19:29:35 por Enzo
Atualizado 11 dezembro 2024 19:47:01 por Enzo