Using Omni Wheels to make Holonomic Drive

Version 1.0 June 14, 2018

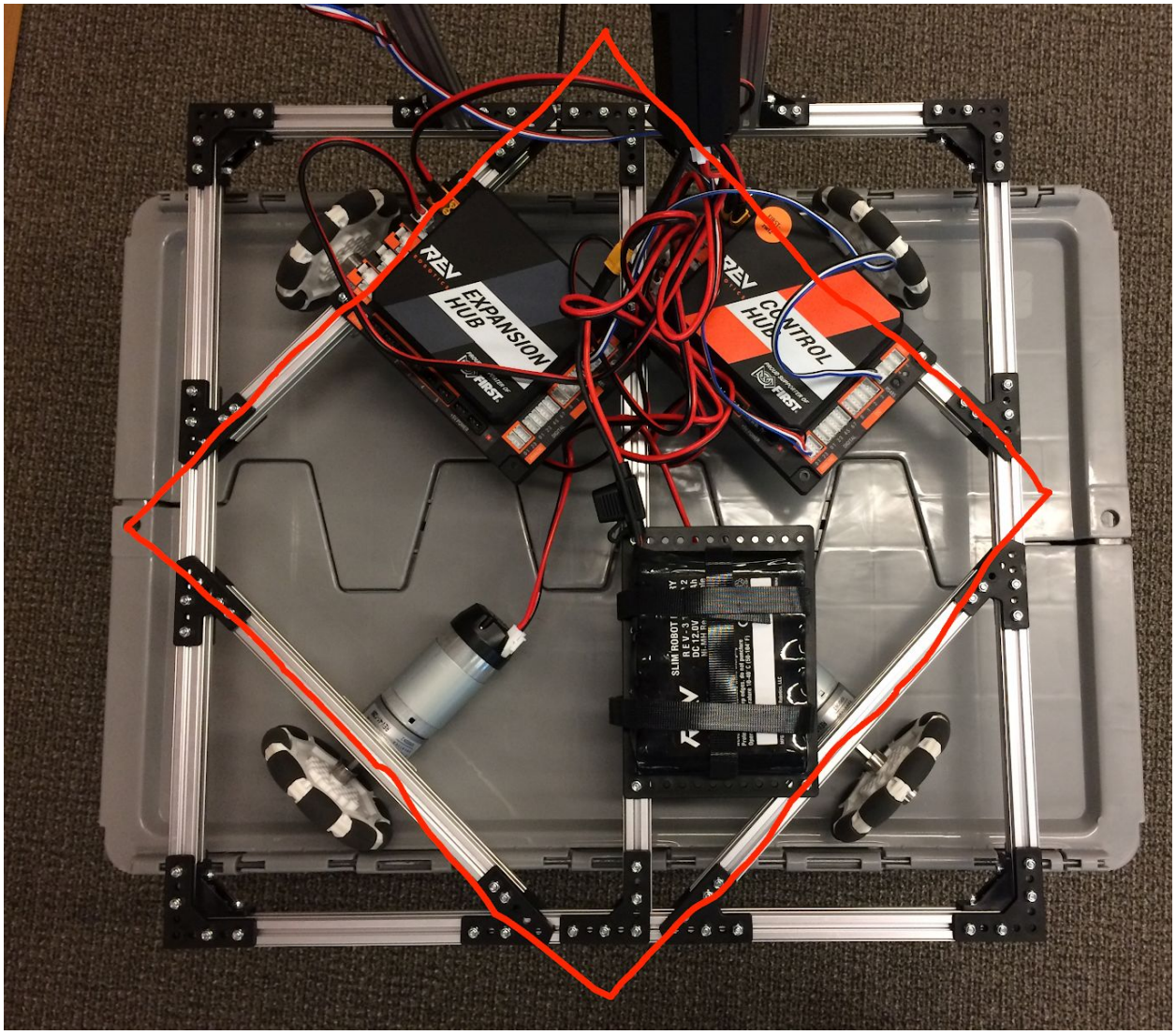
Introduction

I'll preface this guide by saying omni wheels don't work perfectly with the limited number of wheels in the 2018 FIRST Global Kit. Most designs would double layer the omni wheels to increase the contact between the rollers and the ground. Despite this limitation, the drive works well to increase control over your robot's movement. With the code in this guide, a robot can drive in any direction, rotate in place, or rotate while driving linearly.

Table of Contents

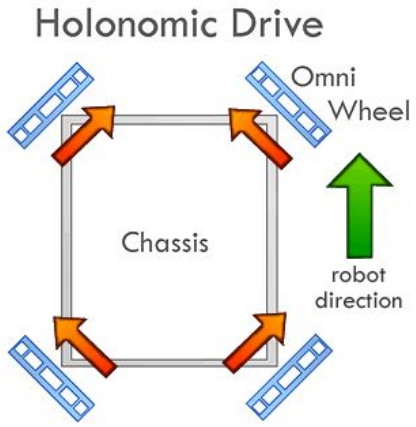
Hardware

I recommend using four omni wheels mounted at 45°. The robot in the picture below has an outer square frame with 90° brackets. The inside uses 45° brackets to mount extrusion on, which has the wheel motors attached below. 135° brackets work as well, and the decision on which to use is based on space constraints. Other builds could easily work for holonomic drive as long as the wheels form a square with each other and are at the center of each side (see highlighted square in red).



Basic Movement

Holonomic drive utilizes the rollers on the omni wheels to allow the robot to move in directions non-parallel to the wheels.



For easier coding, I've defined the motors so that setting all of their powers to positive 1 causes the robot to move in one direction. This can be done with the following code:

```
front_left_wheel.setDirection(DcMotor.Direction.REVERSE);
back_left_wheel.setDirection(DcMotor.Direction.REVERSE);
front_right_wheel.setDirection(DcMotor.Direction.FORWARD);
back_right_wheel.setDirection(DcMotor.Direction.FORWARD);
```

Setting the motors' powers to negative 1 causes the robot to move in the opposite direction. Going left and right is basically the same as forward and back, but you have to shift your perspective and figure out the directions of the motors.

Rotation requires all 4 motors running in the same direction. Since we defined 2 of our motors (front left and back left) as reverse, we have to set those 2 motors to the opposite power of the other 2 motors. For example, we would set the powers of front_left_wheel and back_left_wheel to -1 and front_right_wheel and back_right_wheel to +1.

```
public void driveSimple(){
    double power = .5;
    if(gamepad1.dpad_up){ //Forward
        front_left_wheel.setPower(-power);
        back_left_wheel.setPower(-power);
        back_right_wheel.setPower(-power);
        front_right_wheel.setPower(-power);
    }
    else if(gamepad1.dpad_left){ //Left
        front_left_wheel.setPower(power);
        back_left_wheel.setPower(-power);
        back_right_wheel.setPower(power);
        front_right_wheel.setPower(-power);
    }
    else if(gamepad1.dpad_down){ //Back
        front_left_wheel.setPower(power);
        back_left_wheel.setPower(power);
        back_right_wheel.setPower(power);
        front_right_wheel.setPower(power);
    }
    else if(gamepad1.dpad_right){ //Right
        front_left_wheel.setPower(-power);
        back_left_wheel.setPower(power);
        back_right_wheel.setPower(-power);
        front_right_wheel.setPower(power);
    }
    else if(Math.abs(gamepad1.right_stick_x) > 0){ //Rotation
        front_left_wheel.setPower(-gamepad1.right_stick_x);
        back_left_wheel.setPower(-gamepad1.right_stick_x);
        back_right_wheel.setPower(gamepad1.right_stick_x);
        front_right_wheel.setPower(gamepad1.right_stick_x);
    }
    else{
        front_left_wheel.setPower(0);
        back_left_wheel.setPower(0);
        back_right_wheel.setPower(0);
        front_right_wheel.setPower(0);
    }
}
```

Movement in Any Direction

Movement in any direction looks at the direction of the left joystick and turns it into powers for the robot to use. First, the angle of the joystick is calculated. Note: I'm using *stick_y* and *stick_x* instead of *gamepad1.left_stick_y* and *gamepad1.left_stick_x* because the variables are changed from the raw stick input. More explanation is in the Rotation section.

```
theta = (Math.atan2(stick_y, stick_x)) - (Math.PI / 2);
```

The first part of that line takens the inverse tangent of stick_y/stick_x. Note the returned angle is in radians, so we'll use radians for the rest of our calculations. Subtracting PI/2 essentially rotates the controller 90°. You could achieve the same effect by holding the controller sideways, but that's inconvenient.

Next we convert the angle into usable powers. Wheels diagonal to each other can be set to the same power, so we only need to calculate to powers: Px and Py.

```
Px = Math.sqrt(Math.pow(stick_x, 2) + Math.pow(stick_y, 2)) * (Math.sin(theta + Math.PI / 4));
Py = Math.sqrt(Math.pow(stick_x, 2) + Math.pow(stick_y, 2)) * (Math.sin(theta - Math.PI / 4));
```

We rotate each angle by 45° in opposite directions to create a 90° difference between the angles used for Px and Py. We turn those angles into a ratio using sine, and then multiply that by the total magnitude of the left stick. No matter which direction the stick is pointing, as long as the stick is pushed out as far as possible, the ratio from sine will be multiplied by the same number. The magnitude calculation gives the driver control over the robot's speed.

Then the motors are set to run at their assigned powers:

```
front_left_wheel.setPower(Py);
back_left_wheel.setPower(Px);
back_right_wheel.setPower(Py);
front_right_wheel.setPower(Px);
```

Using the Built-In IMU

The expansion hub should already have defined its IMU in the configuration under I2C Bus 0.

Declaring the IMU is a bit complicated because it has multiple parameters that come with it. Here's an example of its declaration with the important bits highlighted:

```
@TeleOp(name="driveJava", group="Pushbot")
public class driveJava extends LinearOpMode {
    BNO055IMU imu;
    @Override
```

```
    public void runOpMode() {
        imu = hardwareMap.get(BNO055IMU.class, "imu");

        BNO055IMU.Parameters parameters = new BNO055IMU.Parameters();

        parameters.mode                = BNO055IMU.SensorMode.IMU;
        parameters.angleUnit           = BNO055IMU.AngleUnit.DEGREES;
        parameters.accelUnit           = BNO055IMU.AccelUnit.METERS_PERSEC_PERSEC;
        parameters.loggingEnabled      = false;
        imu.initialize(parameters);

        while(!opModeIsActive()){}

        while(opModeIsActive()){
            //Teleop loop here
        }
    }
```

To access the gyro angle, you can use this function:

```
    public double getHeading(){
        Orientation angles = imu.getAngularOrientation(AxesReference.INTRINSIC,
AxesOrder.ZYX, AngleUnit.DEGREES);
        double heading = angles.firstAngle;
        return heading;
    }
```

Which is called by writing:

```
double gyroAngle = getHeading();
```

Since getHeading() returns a double, you can treat the function like a number and use it in calculations.

gyroAngle is defined based on the robot's starting angle. The direction the robot is facing at initialization of the teleop is 0. Rotating the robot clockwise causes *gyroAngle* to decrease, and counterclockwise causes *gyroAngle* to increase. Also, the returned angle is in degrees, not radians.

Gyro Drive

Next we can incorporate the direction the robot is facing when driving. Using this, the robot will always go away from you when you push the joystick away from you. No matter the direction the robot faces, it'll go in the direction the joystick pushes (assuming the gyro was defined correctly).

This part is a lot easier than it seems. Since we already have our joysticks as angles, we can subtract gyroAngle from theta. That turns the theta calculation from above into this:

```
theta = Math.atan2(stick_y, stick_x) - gyroAngle - (Math.PI / 2);
```

But first we have to make a few changes to *gyroAngle*. Inverse tangent outputs radians, so we have to convert *gyroAngle* from degrees into radians. Another issue with *gyroAngle* is that it flips from 180 to -180 (degrees) once the robot makes a half circle. We have to make a few changes to *gyroAngle* to make it work with our *theta* calculation.

```
double gyroAngle = getHeading() * Math.PI / 180; //Converts gyroAngle into radians
if (gyroAngle <= 0) {
    gyroAngle = gyroAngle + (Math.PI / 2);
} else if (0 < gyroAngle && gyroAngle < Math.PI / 2) {
    gyroAngle = gyroAngle + (Math.PI / 2);
} else if (Math.PI / 2 <= gyroAngle) {
    gyroAngle = gyroAngle - (3 * Math.PI / 2);
}
gyroAngle = -1 * gyroAngle;
```

And also in getHeading():

```
if(heading < -180) {
    heading = heading + 360;
}
else if(heading > 180){
    heading = heading - 360;
}
```

There's also two gyro-related quality of life improvements we can make.

The first is the ability to disable the gyroscopic drive by holding right bumper. Once you do this, pushing up on the joystick will cause the robot to move forwards relative to the robot.

```
if(gamepad1.right_bumper){ //Disables gyro, sets to -Math.PI/2 so front is
defined correctly.
    gyroAngle = -Math.PI/2;
}
```

gyroAngle is set to a constant instead of changing based on getHeading(). I'm using -Math.PI/2 so the controls make sense. You could set it to 0 and the robot will drive sideways.

The second fix is the ability to reset the gyro's initially defined angle during teleop without having to restart your opmode. Since it takes too long to reinitialize the IMU, we can create a variable which records the offset from the original 0. We'll call this variable *reset_angle*. First we have the resetAngle() function which goes into the teleop loop. Then we subtract *reset_angle* from the recorded angle from the IMU in getHeading().

```
    public void resetAngle(){
        if(gamepad1.a){
            reset_angle = getHeading() + reset_angle;
        }
    }
    public double getHeading(){
        Orientation angles = imu.getAngularOrientation(AxesReference.INTRINSIC,
AxesOrder.ZYX, AngleUnit.DEGREES);
        double heading = angles.firstAngle;
        if(heading < -180) {
            heading = heading + 360;
        }
        else if(heading > 180){
            heading = heading - 360;
        }
        heading = heading - reset_angle;
        return heading;
    }
```

In resetAngle() we add *reset_angle* to getHeading() in order to get the raw angle from the IMU without *reset_angle* modifying it.

Imagine you rotate the robot 90° counterclockwise from its starting position (0). getHeading() will display 90. Now you press A, and *reset_angle* is set to 90°. This direction is the new 0, and getHeading() returns 0.

Rotation

The rotation code takes *gamepad1.right_stick_x* and applies that power to the motors. That power is Protate.

First we'll look at rotation without linear motion (left joystick).

```
double Protate = gamepad1.right_stick_x/4;
```

You can divide *gamepad1.right_stick_x* by whatever you want, I'm using 4 because it felt fast enough, but not too fast for rotation. We can do basically the same thing as in the driveSimple() rotation code.

```
front_left_wheel.setPower(-Protate);
back_left_wheel.setPower(-Protate);
back_right_wheel.setPower(Protate);
front_right_wheel.setPower(Protate);
```

Now we can add Px and Py to their corresponding motors:

```
front_left_wheel.setPower(Py - Protate);
```

```
back_left_wheel.setPower(Px - Protate);
back_right_wheel.setPower(Py + Protate);
front_right_wheel.setPower(Px + Protate);
```

You can drive linearly with the left joystick and rotate with the right joystick now! There is a slight issue of the sum of linear power and rotational power being greater than 1 which is the power limit of the motor. When this happens, we lose the smooth rotating while driving we had with lesser powers. To fix this issue, we have to decrease Px and Py based on Protate.

The total power cap should be 1. The power cap of linear motion is 1 - *Protate*. This is

```
Math.sqrt(Math.pow(stick_x, 2) + Math.pow(stick_y, 2)) = 1 - Protate
```

The maximum magnitude will happen when *stick_x* = *stick_y* = 1. To calculate the individual cap for *stick_x* and *stick_y* we can divide the magnitude cap by 2.

$$1 - protate = \sqrt{stickx^2 + sticky^2} = \sqrt{2 * stickx^2} = \sqrt{2 * sticky^2}$$

$$(1 - protate)^2 = 2 * stickx^2$$

$$stickx = \sqrt{\frac{(1-protate)^2}{2}}$$

Now we know our cap, so we can multiply *gamepad1.left_stick_x* by that number. Since *gamepad1.left_stick_x* will never be greater than 1, *stick_x* will never exceed the cap. The same can be done for *stick_y*.

```
double stick_x = gamepad1.left_stick_x * Math.sqrt(Math.pow(1-Math.abs(Protate), 2)/2);
double stick_y = gamepad1.left_stick_y * Math.sqrt(Math.pow(1-Math.abs(Protate), 2)/2);
```

Now the combined power of *Px* ± *Protate* or *Py* ± *Protate* will never be greater than 1, and driving while rotating will work no matter the power.

Full Teleop

[Github link to this teleop](#).

```
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.util.ElapsedTime;

@TeleOp(name="driveJava", group="Pushbot")
public class driveJava extends LinearOpMode {

    float rotate_angle = 0;
    double reset_angle = 0;
```

```
private DcMotor front_left_wheel = null;
private DcMotor back_left_wheel = null;
private DcMotor back_right_wheel = null;
private DcMotor front_right_wheel = null;

BNO055IMU imu;
@Override
public void runOpMode() {
    front_left_wheel = hardwareMap.dcMotor.get("front_left_wheel");
    back_left_wheel = hardwareMap.dcMotor.get("back_left_wheel");
    back_right_wheel = hardwareMap.dcMotor.get("back_right_wheel");
    front_right_wheel = hardwareMap.dcMotor.get("front_right_wheel");

    front_left_wheel.setDirection(DcMotor.Direction.REVERSE);
    back_left_wheel.setDirection(DcMotor.Direction.REVERSE);
    front_right_wheel.setDirection(DcMotor.Direction.FORWARD);
    back_right_wheel.setDirection(DcMotor.Direction.FORWARD);

    front_left_wheel.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
    back_left_wheel.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
    front_right_wheel.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
    back_right_wheel.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);


    imu = hardwareMap.get(BNO055IMU.class, "imu");

    BNO055IMU.Parameters parameters = new BNO055IMU.Parameters();

    parameters.mode                = BNO055IMU.SensorMode.IMU;
    parameters.angleUnit           = BNO055IMU.AngleUnit.DEGREES;
    parameters.accelUnit           = BNO055IMU.AccelUnit.METERS_PERSEC_PERSEC;
    parameters.loggingEnabled      = false;
    imu.initialize(parameters);

    while(!opModeIsActive()){}

    while(opModeIsActive()){
        drive();
        resetAngle();
        //driveSimple();
        telemetry.update();
```

```
    }
}
public void driveSimple(){
    double power = .5;
    if(gamepad1.dpad_up){ //Forward
        front_left_wheel.setPower(-power);
        back_left_wheel.setPower(-power);
        back_right_wheel.setPower(-power);
        front_right_wheel.setPower(-power);
    }
    else if(gamepad1.dpad_left){ //Left
        front_left_wheel.setPower(power);
        back_left_wheel.setPower(-power);
        back_right_wheel.setPower(power);
        front_right_wheel.setPower(-power);
    }
    else if(gamepad1.dpad_down){ //Back
        front_left_wheel.setPower(power);
        back_left_wheel.setPower(power);
        back_right_wheel.setPower(power);
        front_right_wheel.setPower(power);
    }
    else if(gamepad1.dpad_right){ //Right
        front_left_wheel.setPower(-power);
        back_left_wheel.setPower(power);
        back_right_wheel.setPower(-power);
        front_right_wheel.setPower(power);
    }
    else if(Math.abs(gamepad1.right_stick_x) > 0){ //Rotation
        front_left_wheel.setPower(-gamepad1.right_stick_x);
        back_left_wheel.setPower(-gamepad1.right_stick_x);
        back_right_wheel.setPower(gamepad1.right_stick_x);
        front_right_wheel.setPower(gamepad1.right_stick_x);
    }
    else{
        front_left_wheel.setPower(0);
        back_left_wheel.setPower(0);
        back_right_wheel.setPower(0);
        front_right_wheel.setPower(0);
    }
}
public void drive() {
```

```
        double Protate = gamepad1.right_stick_x/4;
        double stick_x = gamepad1.left_stick_x * Math.sqrt(Math.pow(1-Math.abs(Protate), 2)/2);
//Accounts for Protate when limiting magnitude to be less than 1
        double stick_y = gamepad1.left_stick_y * Math.sqrt(Math.pow(1-Math.abs(Protate), 2)/2);
        double theta = 0;
        double Px = 0;
        double Py = 0;

        double gyroAngle = getHeading() * Math.PI / 180; //Converts gyroAngle into radians
        if (gyroAngle <= 0) {
            gyroAngle = gyroAngle + (Math.PI / 2);
        } else if (0 < gyroAngle && gyroAngle < Math.PI / 2) {
            gyroAngle = gyroAngle + (Math.PI / 2);
        } else if (Math.PI / 2 <= gyroAngle) {
            gyroAngle = gyroAngle - (3 * Math.PI / 2);
        }
        gyroAngle = -1 * gyroAngle;

        if(gamepad1.right_bumper){ //Disables gyro, sets to -Math.PI/2 so front is defined
correctly.
            gyroAngle = -Math.PI/2;
        }

        //Linear directions in case you want to do straight lines.
        if(gamepad1.dpad_right){
            stick_x = 0.5;
        }
        else if(gamepad1.dpad_left){
            stick_x = -0.5;
        }
        if(gamepad1.dpad_up){
            stick_y = -0.5;
        }
        else if(gamepad1.dpad_down){
            stick_y = 0.5;
        }


        //MOVEMENT
        theta = Math.atan2(stick_y, stick_x) - gyroAngle - (Math.PI / 2);
        Px = Math.sqrt(Math.pow(stick_x, 2) + Math.pow(stick_y, 2)) * (Math.sin(theta + Math.PI /
4));
```

```
        Py = Math.sqrt(Math.pow(stick_x, 2) + Math.pow(stick_y, 2)) * (Math.sin(theta - Math.PI /
4));

        telemetry.addData("Stick_X", stick_x);
        telemetry.addData("Stick_Y", stick_y);
        telemetry.addData("Magnitude",  Math.sqrt(Math.pow(stick_x, 2) + Math.pow(stick_y, 2)));
        telemetry.addData("Front Left", Py - Protate);
        telemetry.addData("Back Left", Px - Protate);
        telemetry.addData("Back Right", Py + Protate);
        telemetry.addData("Front Right", Px + Protate);

        front_left_wheel.setPower(Py - Protate);
        back_left_wheel.setPower(Px - Protate);
        back_right_wheel.setPower(Py + Protate);
        front_right_wheel.setPower(Px + Protate);
    }
    public void resetAngle(){
        if(gamepad1.a){
            reset_angle = getHeading() + reset_angle;
        }
    }
    public double getHeading(){
        Orientation angles = imu.getAngularOrientation(AxesReference.INTRINSIC, AxesOrder.ZYX,
AngleUnit.DEGREES);
        double heading = angles.firstAngle;
        if(heading < -180) {
            heading = heading + 360;
        }
        else if(heading > 180){
            heading = heading - 360;
        }
        heading = heading - reset_angle;
        return heading;
    }
}
```

Contact Information

Feel free to ask me anything about this guide or any other questions you have about robotics! Send emails to engineering@yale.edu or you can message me on Discord at nahn20#5234. I use Github to upload the example opmodes you'll see in these guides.