

Getting your robot to drive with the RobotDrive class

Getting your robot to drive with the RobotDrive class

WPILib provides a RobotDrive object that handles most cases of driving the robot either in autonomous or teleop modes. It is created with either two or four speed controller objects. There are methods to drive with either Tank, Arcade, or Mecanum modes either programmatically or directly from Joysticks.

Note: the examples illustrated in this section are generally correct but have not all been tested on actual robots. But should serve as a starting point for your projects.

Creating a RobotDrive object with Jaguar speed controllers

```
RobotDrive drive(1, 2, 3, 4); // four motor drive configuration
```

```
RobotDrive drive(1, 2); // left, right motors on ports 1,2
```

Create the RobotDrive object specifying either two or four motor ports. By default the RobotDrive constructor will create Jaguar class instances attached to each of those ports.

Using other types of speed controllers

```
0
7 public class RobotTemplate extends SimpleRobot {
8
9     RobotDrive myDrive;
10    Vector frontLeft, frontRight, rearLeft, rearRight;
11
12    public void robotInit() {
13        frontLeft = new Victor(1);
14        frontRight = new Victor(2);
15        rearLeft = new Victor(3);
16        rearRight = new Victor(4);
17        myDrive = new RobotDrive(frontLeft, rearLeft, frontRight, rearRight);
18    }
19
20    public void autonomous() {
21
22    }
```

Getting your robot to drive with the RobotDrive class

You can use RobotDrive with other types of speed controllers as well. In this case you must create the speed controller objects manually and pass the references or pointers to the RobotDrive constructor.

These are Java programs but the C++ program is very similar.

Tank driving with two joysticks

```
8  public class RobotTemplate extends SimpleRobot {
9
10     RobotDrive myDrive;
11     Joystick left, right;
12
13     public void robotInit() {
14         myDrive = new RobotDrive(1, 2, 3, 4);
15         left = new Joystick(1);
16         right = new Joystick(2);
17     }
18
19     public void autonomous() {
20     }
21
22     public void operatorControl() {
23         while (isOperatorControl() && isEnabled()) {
24             myDrive.tankDrive(left, right);
25             Timer.delay(0.01);
26         }
27     }
28 }
```

In this example a RobotDrive object is created with 4 default Jaguar speed controllers. In the operatorControl method the RobotDrive instance tankDrive method is called and it will select the Y-axis of each of the joysticks by default. There are other versions of the tankDrive method that can be used to use alternate axis or just numeric values.

Getting your robot to drive with the RobotDrive class

Arcade driving with a single joystick

```
8 public class RobotTemplate extends SimpleRobot {
9
10     RobotDrive myDrive;
11     Joystick driveStick;
12
13     public void robotInit() {
14         myDrive = new RobotDrive(1, 2, 3, 4);
15         driveStick = new Joystick(1);
16     }
17
18     public void autonomous() {
19     }
20
21     public void operatorControl() {
22         while (isOperatorControl() && isEnabled()) {
23             myDrive.arcadeDrive(driveStick);
24             Timer.delay(0.01);
25         }
26     }
27 }
```

Similar to the example above a single joystick can be used to do single-joystick driving (called arcade). In this case, the X-axis is selected by default for the turn axis and the Y-axis is selected for the speed axis.

Getting your robot to drive with the RobotDrive class

Autonomous driving using the RobotDrive object

```
8
9  public class RobotTemplate extends SimpleRobot {
10
11      RobotDrive myDrive;
12      Joystick driveStick;
13      Gyro gyro;
14      static final double Kp = 0.03;
15
16      public void robotInit() {
17          myDrive = new RobotDrive(1, 2, 3, 4);
18          gyro = new Gyro(1);
19      }
20
21      public void autonomous() {
22          while (isAutonomous() && isEnabled()) {
23              double angle = gyro.getAngle();
24              myDrive.arcadeDrive(-1.0, -angle * Kp);
25              Timer.delay(0.01);
26          }
27      }
28
```

The RobotDrive object also has a number of features that makes it ideally suited for autonomous control. This example illustrates using a gyro for driving in a straight line (the current heading) using the arcade method for steering. While the robot continues to drive in a straight line the gyro headings are roughly zero. As the robot veers off in one direction or the other, the gyro headings vary either positive or negative. This is very convenient since the arcade method turn parameter is also either positive or negative. The magnitude of the gyro headings sets the rate of the turn, that is more off zero the gyro heading is, the faster the robot should turn to correct.

This is a perfect use of proportional control where the rate of turn is proportional to the gyro heading (being off zero). The heading is in values of degrees and can easily get pretty far off, possibly as much as 10 degrees as the robot drives. But the values for the turn in the arcade method are from zero to one. To solve this problem, the heading is scaled by a constant to get it in the range required by the turn parameter of the arcade method. This parameter is called the proportional gain, often written as kP.

In this particular example the robot is designed such that negative speed values go forward. Also, not that the the angle from the gyro is written as "-angle". This is because this particular robot turns in the opposite direction from the gyro corrections and has to be negated to correct that. Your robot might be different in both cases depending on gearing and motor connections.

Getting your robot to drive with the RobotDrive class

Mecanum driving

```
public class RobotTemplate extends SimpleRobot {  
  
    RobotDrive myDrive;  
    Joystick moveStick, rotateStick;  
  
    public void robotInit() {  
        myDrive = new RobotDrive(1, 2, 3, 4);  
        moveStick = new Joystick(1);  
        rotateStick = new Joystick(2);  
    }  
  
    public void autonomous() {  
    }  
  
    public void operatorControl() {  
        while (isAutonomous() && isEnabled()) {  
            myDrive.mecanumDrive_Polar(moveStick.getY(), moveStick.getX(), rotateStick.getTwist());  
            Timer.delay(0.01);  
        }  
    }  
}
```

The RobotDrive can also handle Mecanum driving. That is using Mecanum wheels on the chassis to enable the robot to drive in any direction without first turning. This is sometimes called Holonomic driving.

In this example there are two joysticks controlling the robot. moveStick supplies the direction vector for the robot, that is which way it should move irrespective of the heading. rotateStick supplies the rate of rotation in the twist (rudder) axis on the joystick. If you push the moveStick full forward the robot will move forward, even if it's facing to the left. At the same time, if you rotate the rotateStick, the robot will spin in the rotation direction with the rotation rate from the amount of twist, while the robot continues to move forward.