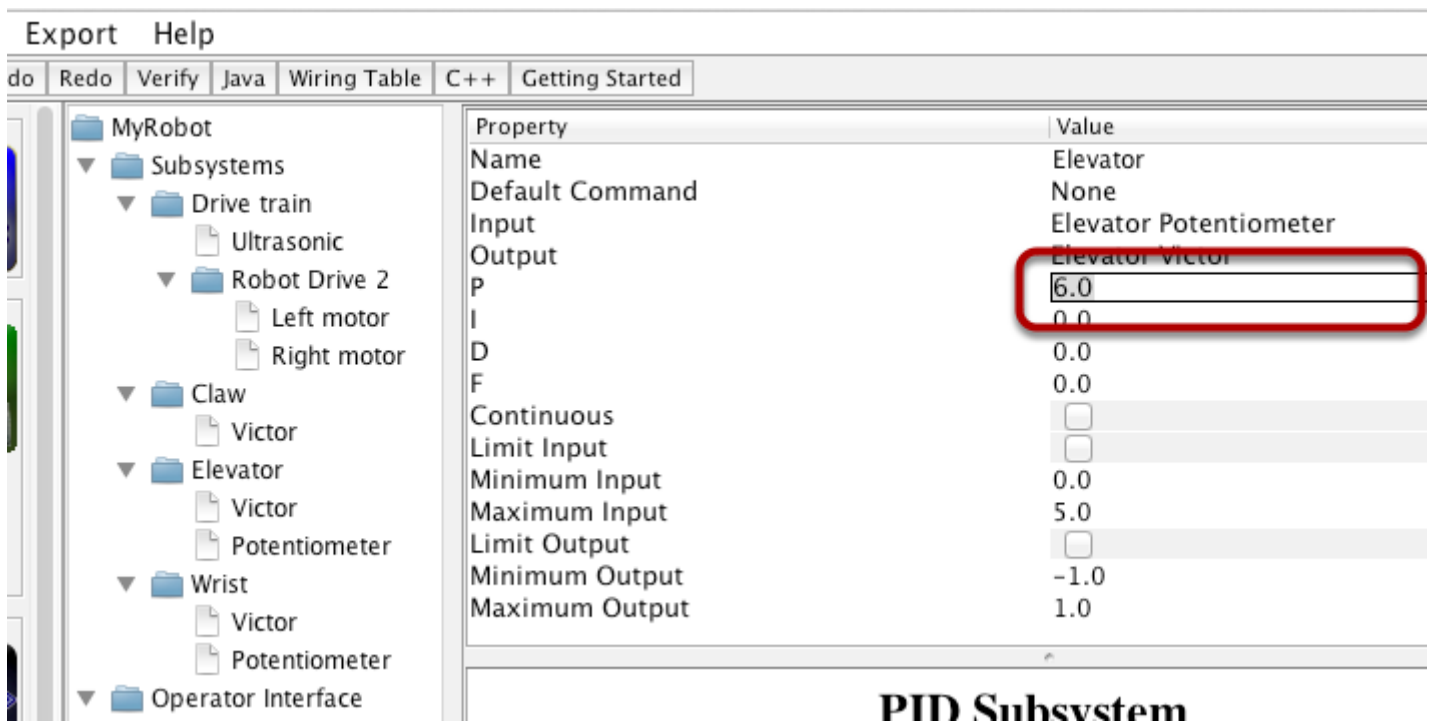


Writing the code for a PIDSubsystem in Java

PIDSubsystems use feedback to control the actuator and drive it to a particular position. In this example we use an elevator with a 10-turn potentiometer connected to it to give feedback on the height. The skeleton of the PIDSubsystem is generated by the RobotBuilder and we have to fill in the rest of the code to provide the potentiometer value and drive the motor with the output of the imbedded PIDController.

Setting the PID constants



Make sure the Elevator PID subsystem has been created in the RobotBuilder. In the case of our elevator we use a proportional constant of 6.0 and 0 for the I and D terms. Once it's all set, generate Java code for the project using the Export menu or the Java toolbar menu.

Writing the code for a PIDSubsystem in Java

Add constants for the Elevator preset positions and enable the PID controller

```
11  L  */
12  public class Elevator extends PIDSubsystem {
13      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
14      AnalogChannel potentiometer = RobotMap.ELEVATOR_POTENTIOMETER;
15      Victor victor = RobotMap.ELEVATOR_VICTOR;
16      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
17
18      public static final double BOTTOM = 4.6,
19          STOW = 1.65,
20          TABLE_HEIGHT = 1.58;
21
22      public Elevator() {
23          // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=PID
24          super("Elevator", 1.0, 0.0, 0.0);
25          getPIDController().setContinuous(false);
26          // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=PID
27
28          setSetpoint(STOW);
29          enable();
30      }
31
32      public void initDefaultCommand() {
33          // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT COMMAND
```

To make it easier to drive the elevator to preset positions, we added preset positions for the bottom, stow, and table height. Then the elevator is set to the STOW position by setting the PID setpoint and the PID controller is enabled. This will cause the elevator to move to the stowed position when the robot is enabled.

Look at the autogenerated code from RobotBuilder for returnPIDInput

```
35  L  }
36
37  protected double returnPIDInput() {
38      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=SOURCE
39      return potentiometer.pidGet();
40      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=SOURCE
41  }
42
43  protected void usePIDOutput(double output) {
```

The returnPIDInput() method is used to set the value of the sensor that is providing the feedback for the PID controller. In this case, the code is automatically generated and returns the potentiometer raw analog input value (a number that ranges from 0-1023). In our case we would like the PID controller to be based on the average voltage read by the analog input for the potentiometer, not the raw value.

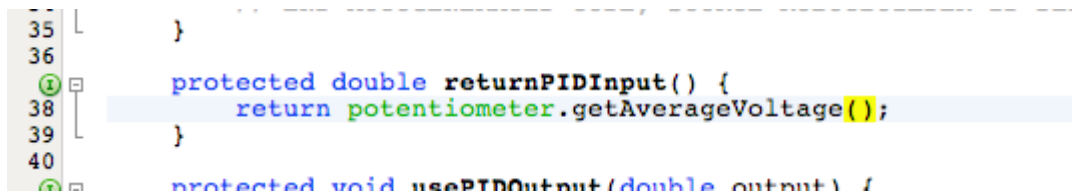
Writing the code for a PIDSubsystem in Java

If we just change the line:

```
return potentiometer.pidGet();
```

it will be overwritten by RobotBuilder next time we export to Java. You can tell which lines are automatically generated by looking at the "//BEGIN AUTOGENERATED CODE" and "//END AUTOGENERATED CODE" comments. Any code inbetween those markers will be overwritten next time RobotBuilder is run. You're free to change anything outside of those blocks.

Use the average voltage for the PID input

A screenshot of a code editor showing a Java class. Lines 35 and 36 show a closing curly brace for a previous method. Line 38 shows the start of a new method: `protected double returnPIDInput() {`. Line 39 shows the return statement: `return potentiometer.getAverageVoltage();`. Line 40 shows the closing curly brace: `}`. The line `return potentiometer.getAverageVoltage();` is highlighted in blue. Below line 40, the start of another method is visible: `protected void usePIDOutput(double output) {`.

To get around the problem from the last step, the comment blocks can be removed. Then if the line is changed as shown, it will no longer be overwritten by RobotBuilder.

Remember, if we just wanted to add code to a method it could be added safely outside of the comment blocks.

That's all that is required to create the Elevator PIDSubsystem in Java. To operate it with commands to actually control the motion see: [Operating a PIDSubsystem from a command](#)