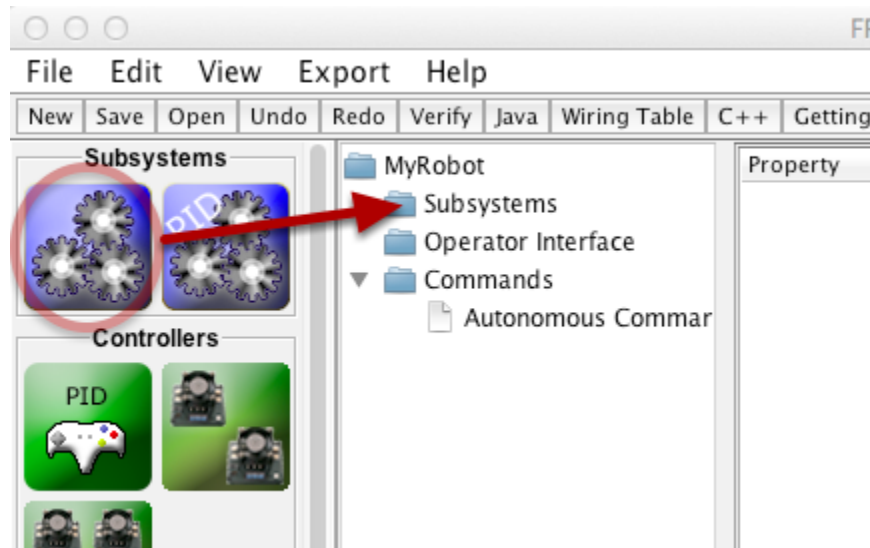


Creating a subsystem

Creating a subsystem

Subsystems are classes that encapsulate (or contain) all the data and code that make a subsystem on your robot operate. The first step in creating a robot program with the RobotBuilder is to identify and create all the subsystems on the robot. Examples of subsystems are grippers, ball collectors, the drive base, elevators, arms, etc. Each subsystem contains all the sensors and actuators that are used to make it work. For example, an elevator might have a Jaguar speed controller and a potentiometer to provide feedback of the robot position.

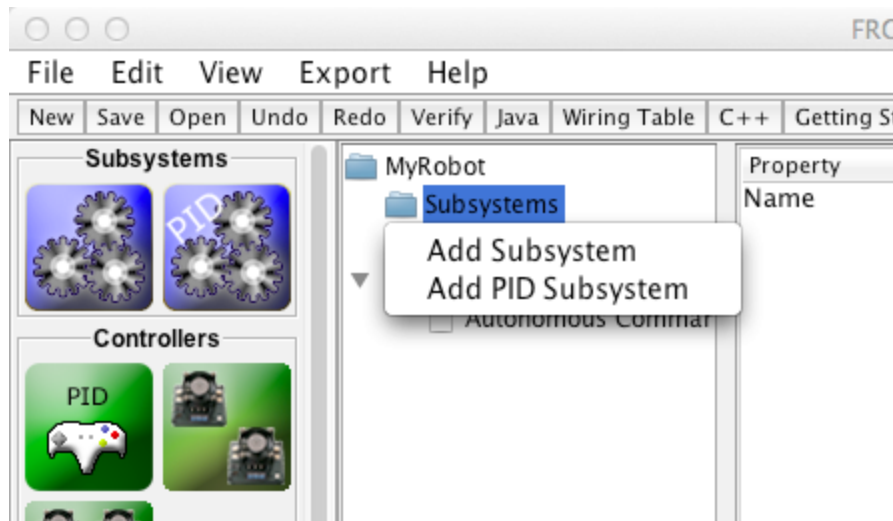
Creating a subsystem by dragging from the palette



Drag the subsystem icon from the palette to the Subsystems folder in the robot description to create a subsystem class.

Creating a subsystem

Creating a subsystem by using the context menu on the Subsystem folder



Right-click on the Subsystem folder in the robot description to add a subsystem to that folder.

Name the subsystem



After creating the subsystem by either dragging or using the context menu as described above, simply type the name you would like to give the subsystem. The name can be multiple words separated by spaces, RobotBuilder will concatenate the words to make a proper Java or C++ class name for you.

Adding constants (2016 only)

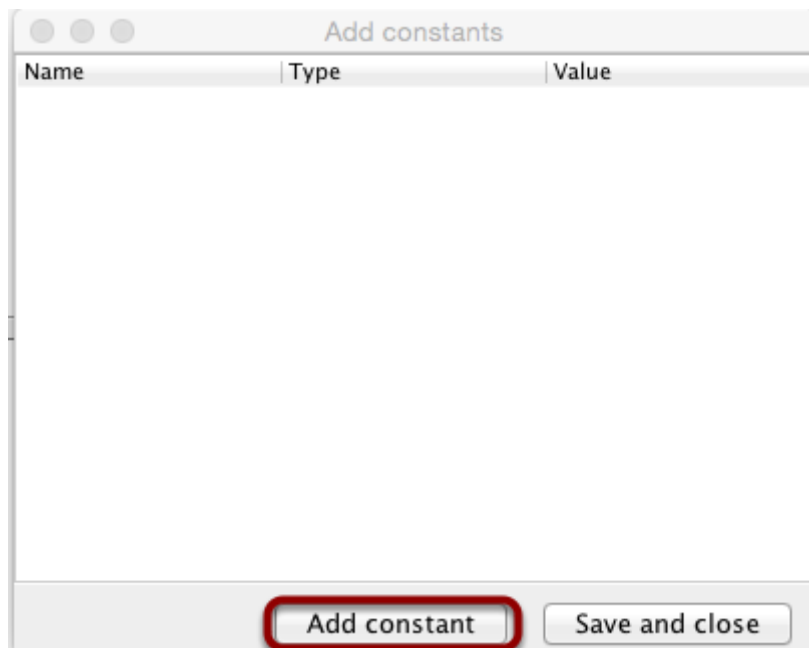


Creating a subsystem

Constants are very useful to reduce the amount of magic numbers in your code. In subsystems, they can be used to keep track of certain values, such as sensor values for specific heights of an elevator, or the speed at which to drive the robot.

By default, there will be no constants in a subsystem. Press the button next to "Constants" to open a dialog to create some.

Creating constants (2016 only)



The constants table will be empty at first. Press "Add constant" to add one.

Creating a subsystem

Add constants (2016 only)

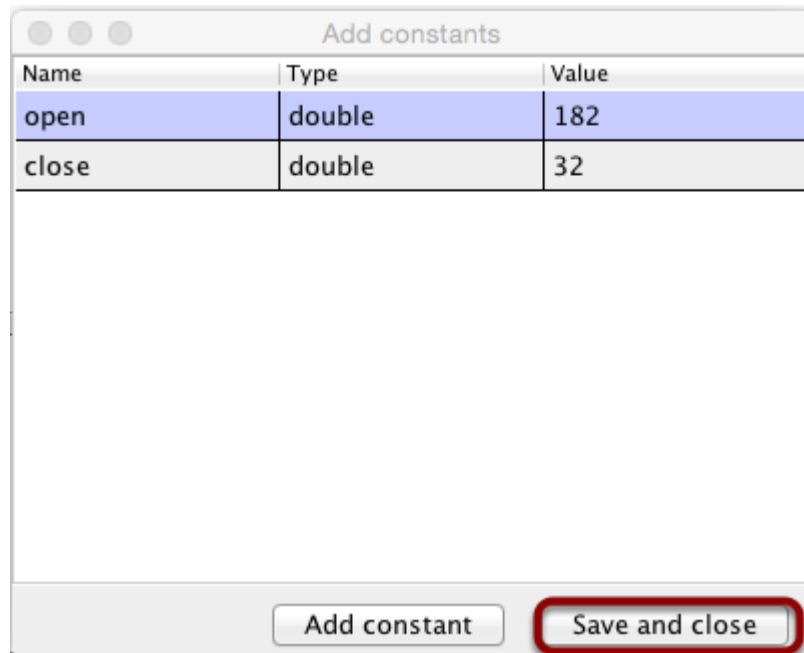
Name	Type	Value
[change me]	String	

1. The name of the constant. Change this to something descriptive. In this example of a gripper, some good constants might be "open" and "close".
2. The type of the constant. This will most likely be a double, but you can choose from one of: String, double, int, long int, boolean, or byte.
3. The value of the constant.

If a constant is valid, it will be exported to the subsystem class as a public static final/public const static variable depending on the language. If it isn't, that constant will be highlighted red to show there's something wrong with it. Common problems are an invalid Java/C++ variable name (such as having spaces or non-letter characters in the name) or having non-numeral characters in the value field of a numeric constant.

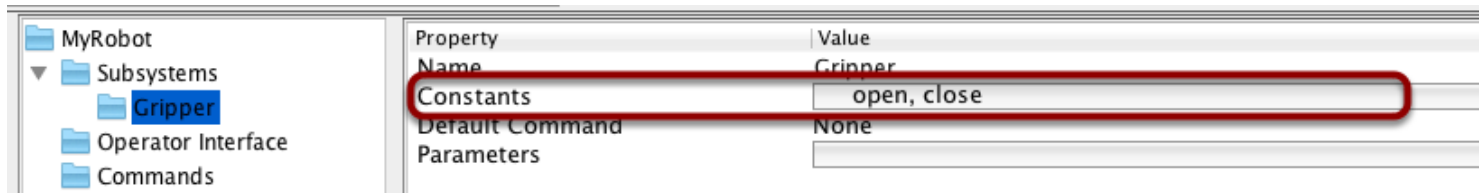
Creating a subsystem

Saving constants (2016 only)



After adding constants and setting their values, just press "Save and close" to save the constants and close the dialog. If you don't want to save, press the exit button on the top of the window.

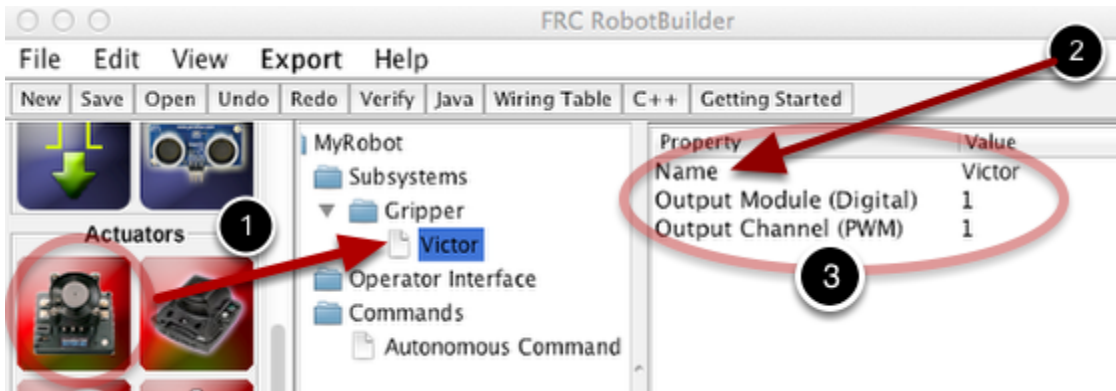
After saving (2016 only)



After saving constants, the names will appear in the "Constants" button in the subsystem properties.

Creating a subsystem

Drag actuators and sensors into the subsystem



There are two steps to adding components to a subsystem:

1. Drag actuators or sensors from the palette into the subsystem as required.
2. Give the actuator or sensor a meaningful name
3. Edit the properties such as module numbers and channel numbers for each item in the subsystem.

RobotBuilder will automatically use incrementing channel numbers for each module on the robot. If you haven't yet wired the robot you can just let RobotBuilder assign unique channel numbers for each sensor or actuator and wire the robot according to the generating wiring table.

This just creates the subsystem in RobotBuilder, and will subsequently generate skeleton code for the subsystem. To make it actually operate your robot please refer to: [Writing the code for a subsystem in Java](#) or [Writing the code for a subsystem in C++](#).