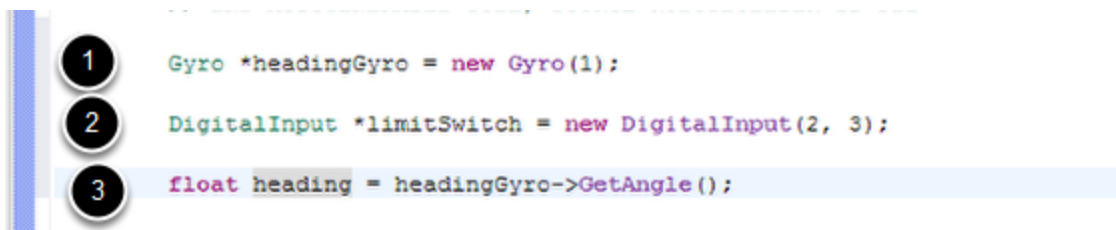


# C++ conventions for objects, methods, and variables

Every sensor, actuator and operator interface component is an object in either C++ or Java programs. To use one of these you must create an instance of it using the class name. Any references to the objects such as reading values, setting values, or setting parameters is done through the reference. There are a number of utility objects in WPILib such as the RobotDrive and Compressor that don't represent a single sensor or actuator, but a larger subsystem.

Another convention used throughout the library is the case of the method names. In **C++ all methods start with an upper case letter**, then are camel case (intermediate words capitalized). In **Java all methods start with lower case letters** then camel case for the remainder of the name.

## Creating objects that are connected to the cRIO in C++



```
1 Gyro *headingGyro = new Gyro(1);  
2 DigitalInput *limitSwitch = new DigitalInput(2, 3);  
3 float heading = headingGyro->GetAngle();
```

Generally all the objects in WPILib that connect to one of the cRIO breakout boards have one or two arguments in the constructor when created where you specify the channel or port number it is connected to. The above example illustrate the conventions used in WPILib for both C++ and Java.

1. Creates a Gyro object connected to analog module 1 channel 1 and stores its address in "headingGyro". For convenience if only a single number is specified it is assumed to be for the first module of a given type (in this case an analog module) and the number is the channel or port number.
2. Creates a DigitalInput object connected to the 2nd installed digital module using channel 3 and stores the address in the variable "limitSwitch".

# C++ conventions for objects, methods, and variables

3. Gets the current heading from the Gyro in degrees and stores it in the variable "heading". In this case, since the Gyro object is referenced through a pointer (dereferenced), then you must use the "->" operator to call methods on the Gyro.

## Creating operator interface objects

```
Joystick *stick = new Joystick(1);  
  
double speed = stick->GetX();|
```

Generally objects connected to the Driver station PC via USB (with the exception of the Cypress FIRST Touch board and Microsoft Kinect) take a single argument indicating the USB port they are connected to. A single Joystick class is provided which should provide the functionality needed to interface with any joystick or gamepad which works with the FRC Driver Station.

1. Creates a Joystick object connected to USB port 1 on the DS (listed first in the Setup tab of the DS).
2. Gets the current X axis value of the joystick and stores it in the variable "speed".

## Module ordering and numbering

Physical Slot Number	Module number In your program	8-slot cRIO	4-slot cRIO
1	1	Analog Module 9201	Analog Module 9201
2	1	Digital Module 9403	Digital Module 9403
3	1	Solenoid Module 9472	Solenoid Module 9472
4	2	<u>empty</u>	Any module type
5	2	Analog Module 9201	NA
6	2	Digital Module 9403	NA
7	2	Solenoid Module 9472	NA
8		Empty	NA

# C++ conventions for objects, methods, and variables

The device number represents the instance of the module type. For example the first digital module would be 1 and the second one would be 2. If you only had a single module of each type in your robot and you used the short form of the constructors when creating devices (where the slot number argument was left out and defaulted to the first module) then your code doesn't have to change. The library will continue to default the numbers to the first module of a given type.

## Class, method, and variable naming

Type of name	Naming rules	Examples
<b>Class name</b>	Initial upper case letter then camel case (mixed upper/lower case) except acronyms which are all upper case	<b>Victor, SimpleRobot, PWM</b>
<b>Method name</b>	Initial upper case letter then camel case	<b>StartCompetition, Autonomous, GetAngle</b>
<b>Member variable</b>	"m_" followed by the member variable name starting with a lower case letter then camel case	<b>m_deleteSpeedControllers, m_sensitivity</b>
<b>Local variable</b>	Initial lower case	<b>targetAngle</b>
<b>Macro</b>	All upper case with _ between words.  <b>Note:</b> It's better to use <u>const</u> values and inline functions than macros.	<b>DISALLOW_COPY_AND_ASSIGN</b>

Names in WPILib follow the conventions shown in the table above. It makes it very easy to determine what the scope and use of a variable is just by looking at the name and is a convention that is used throughout WPILib.