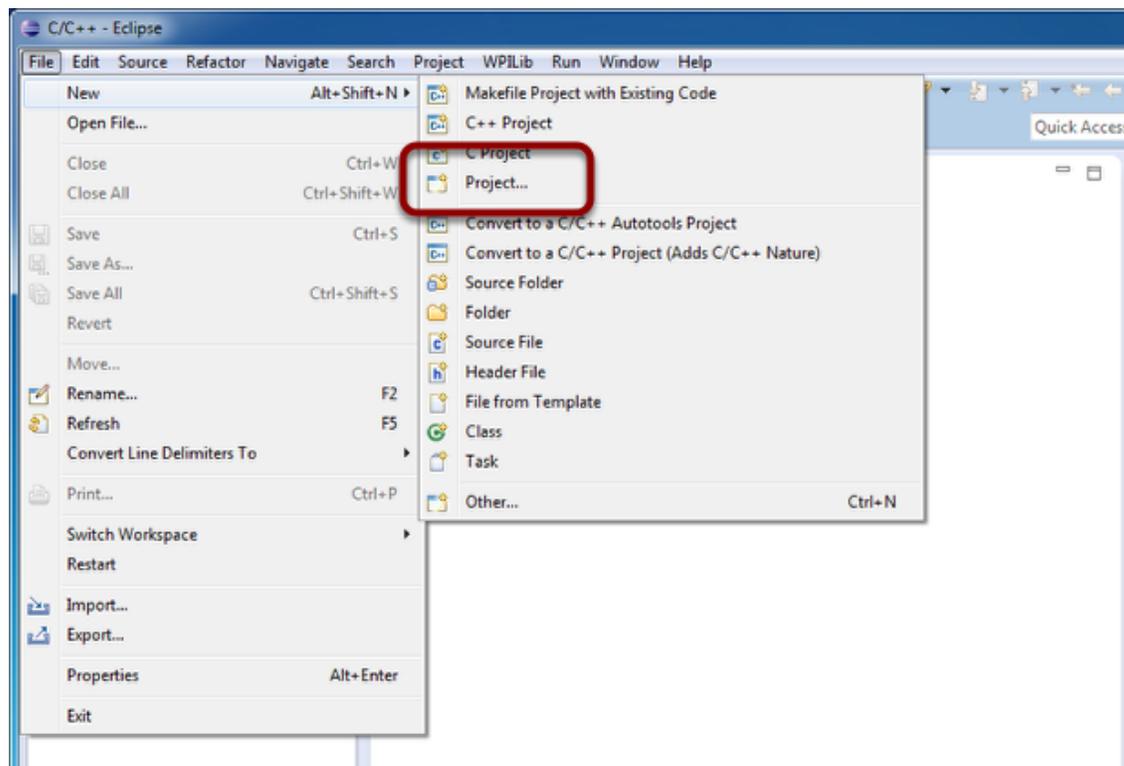# Creating your Benchtop Test Program

The simplest way to create a robot program, is to start from one of the three supplied templates (Sample, Iterative or Command). Sample is best used for very small sample programs or advanced programs that require total control over program flow. Iterative Robot is a template which provides better structure for robot programs while maintaining a minimal learning curve. Command-Based robot is a template that provides a modular, extensible structure with a moderate learning curve.

The templates will get you the basis of a robot program, organizing a larger project can often be a complex task. RobotBuilder is recommended for creating and organizing Command-Based robot programs. You can learn more about RobotBuilder here. To create a command-based robot program that takes advantage of all the newer tools look at Creating a Robot Project in the Command Based Programming Chapter.
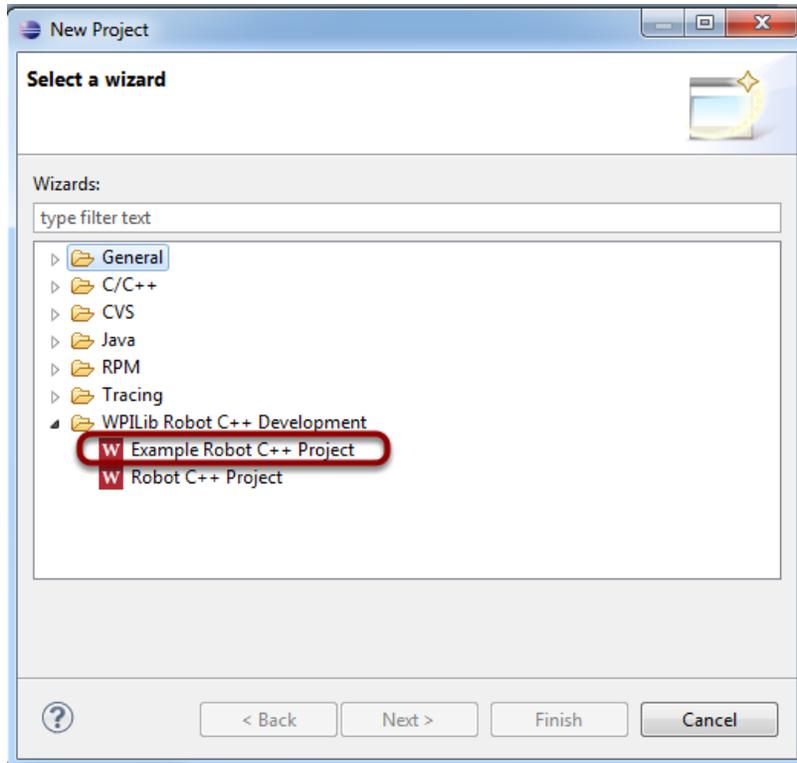
## Creating a project



To create a project click "File" then "New" then click "Project...".
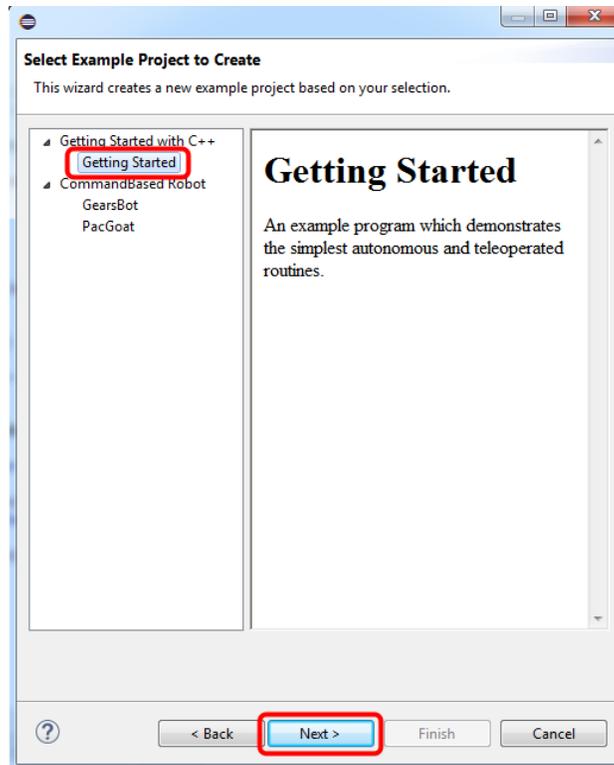
---

# Creating your Benchtop Test Program

## Selecting the project type (Example Robot C++ Project)



Choose **Example Robot C++ Project** as the project type. For creating future robot programs you will likely select **Robot C++ Project.**
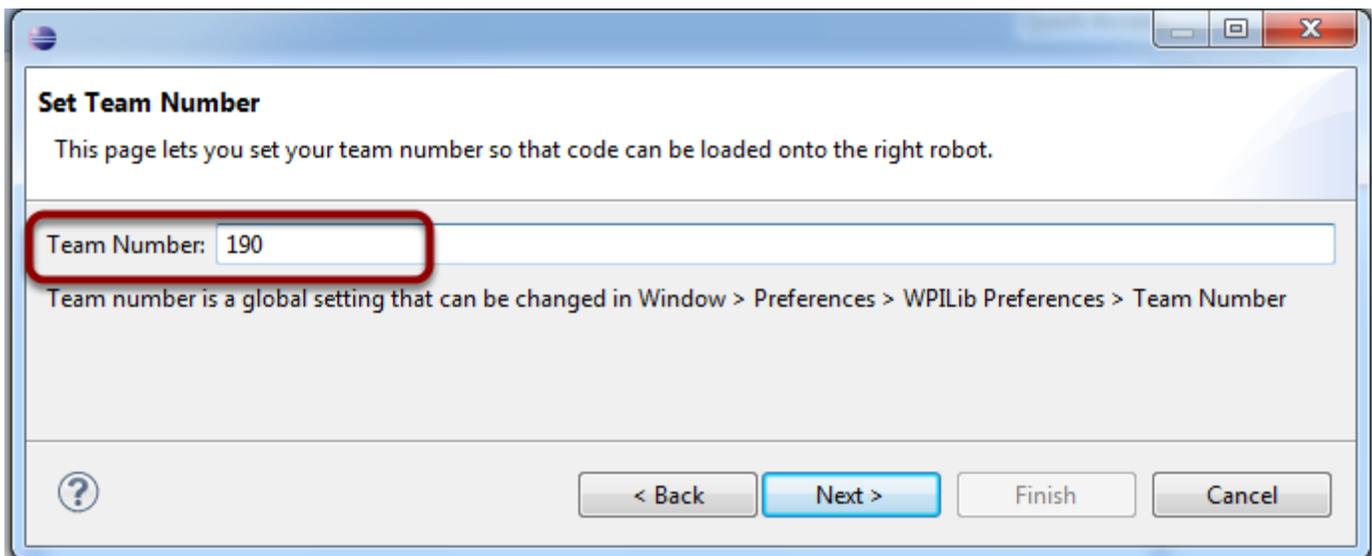
# Creating your Benchtop Test Program

## Selecting an Example



Select the **Getting Started** example, then click **Next**.

## Entering the team number

# Creating your Benchtop Test Program

Enter your team number. This is used when the Eclipse tools download programs onto your RoboRIO. As described in the dialog this is a global setting that can also be accessed from Window->Preferences->WPILib Preferences (if you need to change what team number you are targeting). This dialog will only be shown if there is no team number currently set in Eclipse global settings.
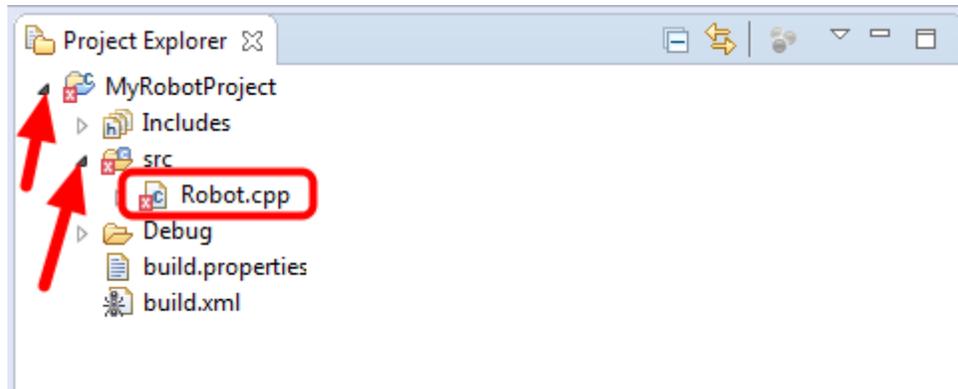
## Project Name
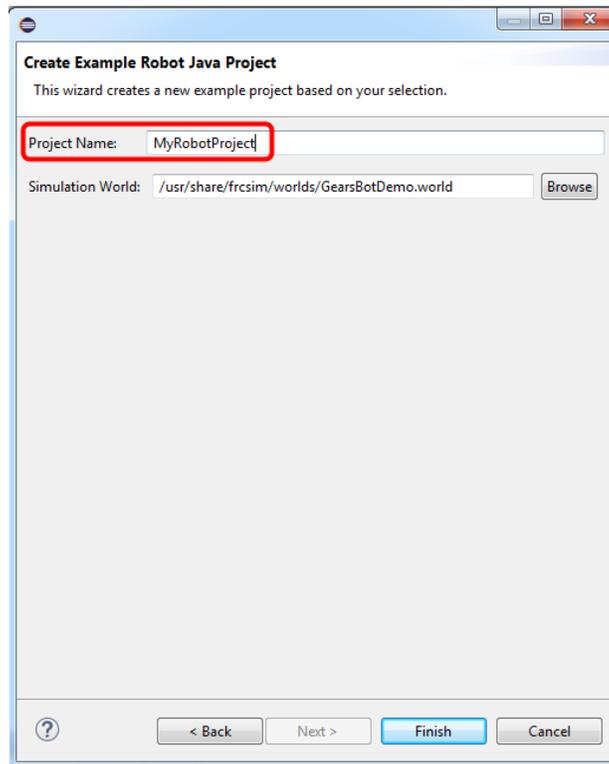


Enter a name for your robot project. The simulation world is used for simulation and can be left at the default option. Then click Finish.

---

# Creating your Benchtop Test Program

## Eclipse Project Explorer



The project will be created and appear in the Eclipse Project Explorer. Click the arrows to expand the project, then the **src** folder. Double click on **Robot.cpp** to open it.

## Rebuilding the Index



It is currently necessary to re-build the index after creating a new template or example for the Eclipse syntax checker to work properly. To do this, right click on the project in the Project Explorer and select Index->Rebuild.

# Defining the variables for our sample robot

```cpp
C++

class Robot: public IterativeRobot
{
        /*
         * Defines the variables as members of our Robot class
         */
        RobotDrive myRobot; // robot drive system
        Joystick stick; // only joystick
        LiveWindow *lw;
        int autoLoopCounter;
```

# Creating your Benchtop Test Program

```
public:
        Robot() :
                /* Initializes the variables as part of the constructor using an
Initialization
                 * list.
                 */
                myRobot(0,1), //these must be initialized in the same order
                stick(1),         //as they are declared above.
                lw(NULL),
                autoLoopCounter(0)
                {
                        myRobot.SetExpiration(0.1);
                        /*
                         * Performs robot initialization
                         * (in this case sets the safety timer expiration for
                         * the myRobot object to .1 seconds, see the next step for an
explanation of
                         * the motor safety timers).
                         */
                }
```

The sample robot in our examples will have a joystick on USB port 1 for arcade drive and two motors on PWM ports 0 and 1. Here we create objects of type RobotDrive (myRobot), Joystick (stick), LiveWindow (*lw), and integer (autoLoopCounter). This section of the code does three things:

1. Defines the variables as members of our Robot class.
2. Initializes the variables as part of the constructor using an Initialization List.
3. Performs robot initialization (in this case sets the safety timer expiration for the myRobot object to .1 seconds, see the next step for an explanation of motor safety timers).

# Creating your Benchtop Test Program

## Robot Initialization

```cpp
C++

      void RobotInit()
      {
             lw = LiveWindow::GetInstance();
      }
```

The RobotInit method is run when the robot program is starting up, but after the constructor. The RobotInit for our sample program gets a pointer to the LiveWindow instance (this is used in the test method discussed below)

## Simple autonomous sample

```cpp
C++

      void AutonomousInit() //This method is called once each time the robot enters
Autonomous
      {
             autoLoopCounter = 0;
      }

      void AutonomousPeriodic() /* This method is called each time the robot receives a
      * packet instructing the robot to be in Autonomous Enabled mode (approximately
every 20ms)
      * This means that code in this method should return in 20 ms or less (no delays or
loops)
      */
      {
             /* This example checks if the loop counter has reached 100
             * (approximately 2 seconds) THis is a fairly simple example and is not
recommended for actual
             * autonomous routines, because delayed or missed packets will
significantly affect the timing
```

# Creating your Benchtop Test Program

```
          * of this program
          */
          if(autoLoopCounter
```

# Joystick Control for teleoperation

**C++**

```cpp
    void TeleopInit()
    {
          /* The TeleopInit method is called once each time the robot enters
teleoperated mode
          *   This is where you would put presets for the teleoperated mode.
          */
    }

    void TeleopPeriodic()
    {
          /* The TeleopPeriodic method is called each time the robot recieves a
packet instructing it
          * to be in teleoperated mode
          */
          myRobot.ArcadeDrive(stick); // This line tells the drivetrain to use the
joystick to control the
          // robot with the Arcade Drive scheme (Y-Axis throttle, X-Axis turn).
    }
```

# Test Mode

**C++**

```cpp
    void TestPeriodic()
    {
          lw->Run();
```

```
    }
```

Test Mode is used for testing robot functionality. The test mode of our sample program runs LiveWindow. LiveWindow is a part of the SmartDashboard that allows you to see inputs and control outputs on the robot from the dashboard when the robot is in Test Mode. You can read more about LiveWindow in the [SmartDashboard section](#) of the Driver Station Manual.

## Final Details

Some final details:

- In this example myRobot, and stick are member objects of the RobotDemo class. They're accessed using references, one of the ways of accessing objects in C++. See the section on [pointers](#) as an alternative method of using WPILib objects.
- The myRobot.Drive() method takes two parameters: a speed and a turn rate. See the documentation about the RobotDrive object for details on how the speed and direction parameters work.

To learn about running this program on the roboRIO see the next article.