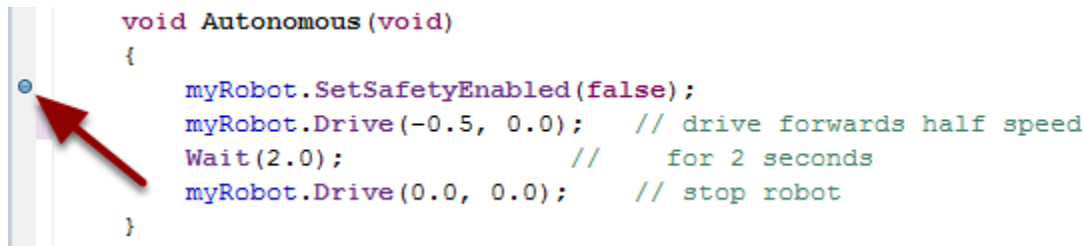


Debugging a robot program

Debugging a robot program

A debugger is used to control program flow and monitor variables in order to assist in debugging a program. This section will describe how to set up a debug session for a Java FRC robot program.

Set a Breakpoint

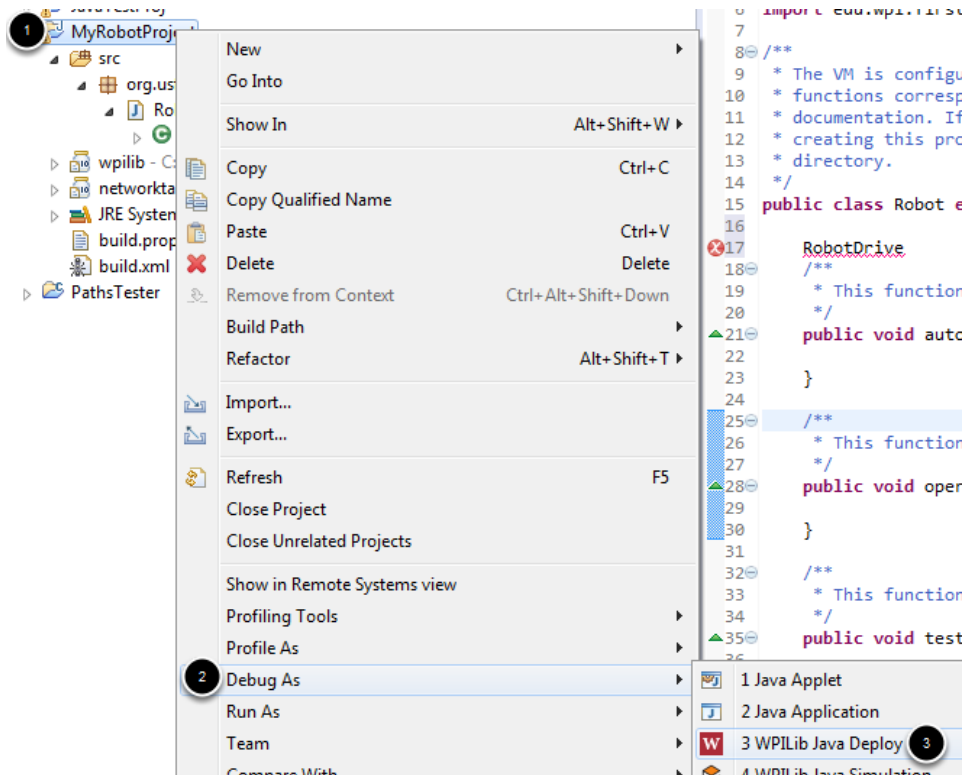


Clicking the “Debug” button does several things. It changes the Workbench to the Debug Perspective which has the views Debug, Breakpoints, and Variables along the right side of the window. It starts the robot task and connects to it with the debugger

Now double-click in the left margin of the source code window to set a breakpoint in your user program: A small blue circle indicates the breakpoint has been set on the corresponding line. (You can see all your breakpoints in the Workbench’s Breakpoints view.)

Debugging a robot program

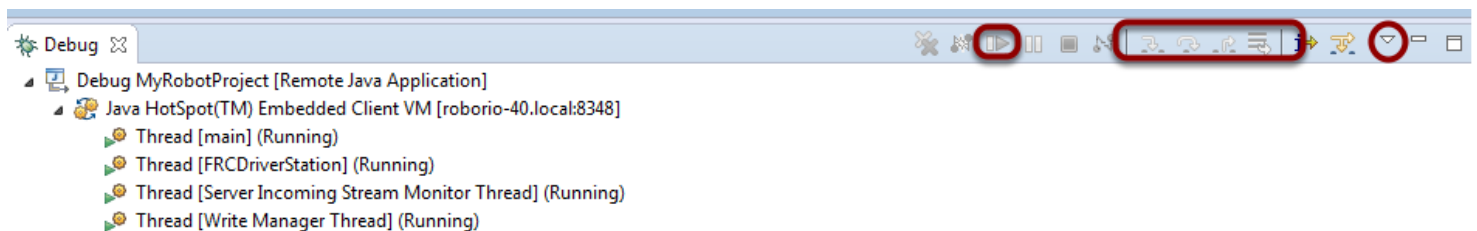
Start Debugging



1. Right click on the project
2. Select Debug As
3. Click WPILib Java Deploy

The Debug Perspective will open automatically and the debug process will begin running. Note that the Java debugger does not break at the first program instruction so if you wish to catch the program before it executes the constructor or RobotInit or a similar method you should set a breakpoint (described below) before you start debugging.

The Debug window



Debugging a robot program

The Debug window shows all processes and threads spawned by your code running on the roboRIO. When your breakpoint is reached, make sure your program is selected in the task list and your source code is displayed with a program pointer. You can continue stepping through your code using “Resume,” “Step Into,” “Step Over,” and “Step Return” buttons: If you don't see Java code displayed, it's because you've stepped down into library code where the source is not available to the debugger. “Step Return” will bring you back up a level.

If the debug toolbar (resume, terminate and step controls) is not visible, click the down arrow at the top right of the debug window and select the **Show Debug Toolbar** option

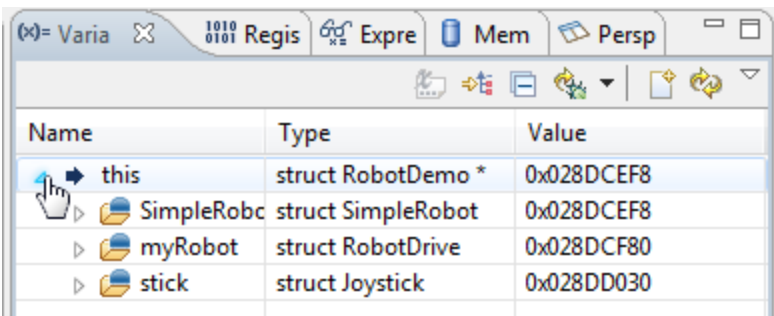
The code will automatically run until it hits a breakpoint. Note that if the breakpoint you have placed is inside one of the Teleoperated or Autonomous methods, the robot will have to be set to the appropriate mode and enabled with the Driver Station in order to reach the breakpoint.





Your Breakpoint

```
void Autonomous(void)
{
    myRobot.SetSafetyEnabled(false);
    myRobot.Drive(-0.5, 0.0); // drive forwards half speed
    Wait(2.0);                // for 2 seconds
    myRobot.Drive(0.0, 0.0);  // stop robot
}
```

The program will start running and then pause at the breakpoint. From here you can use the three panes on the left of the screen (Debug, Breakpoints and Variables) to monitor and control the execution of your program.

The Variables Tab



Name	Type	Value
 this	struct RobotDemo *	0x028DCEF8
 SimpleRobc	struct SimpleRobot	0x028DCEF8
 myRobot	struct RobotDrive	0x028DCF80
 stick	struct Joystick	0x028DD030

The Variables view shows the current values of variables. To see a variable that is not displayed, select the “Expressions” tab and enter the variable name. This will show the variable's value if it's in

Debugging a robot program

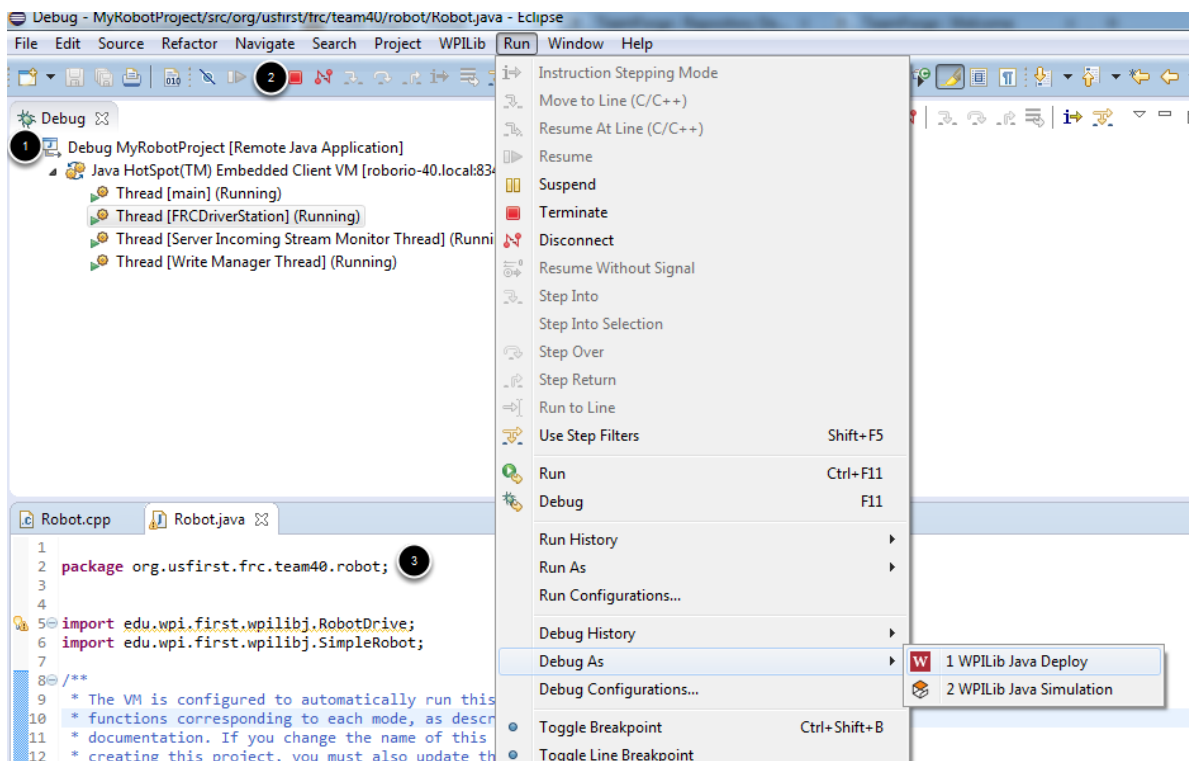
scope. You may also want to click on the arrows next to a variable to expand the tree and show it's members. For example, expanding our Robot Demo variable ("this") shows our "myRobot" and "stick" variables.

Stopping Debugging



To stop debugging, you can disconnect or terminate the process. Disconnecting detaches the debugger but leaves the process running in its current state. Terminating the process kills it on the target.

Subsequent Debugging



If you would like to change code and relaunch the debugger from the Debug perspective:

1. Click the top level of the existing debug session in the Debug pane
2. Click the Terminate button

Debugging a robot program

3. Click in the window where your project source code is located. Putting the focus on this window is necessary for Eclipse to know what project to run.
4. Click the **Run** menu
5. Select **Debug As**
6. Click **WPILib Java Deploy**

If you see grayed out menu items or don't see the WPILib Java Deploy menu option, make sure that the window focus is on source code from your project as described in Step 3.

Debugging with Console

Another way to debug your program is to use `System.out.println` statements in your code and receive them using the RioLog in Eclipse.