C/C++ and Java Installation For 2017 FRC Teams

Mike Anderson
(robot_maker12@verizon.net)

Herndon High School
FRC Team #116

What We'll Talk About

- Goals
- The development environment
- Talking to the RoboRIO
- Making it move
- Resources
- Summary

Goals

- The goal of this presentation is to help you understand how to prepare your development environment for use with C/C++ and Java
- We clearly can't explain all of the aspects because we have limited time
  - But, you should leave here with a better understanding of the process
- We will be talking about the set up rather than the languages themselves
  - The WPILib is equivalent between the environments

Why C/C++?

- C/C++ is a standard in embedded systems programming for over 30 years
  - It's still the most predominant language in embedded Linux, the IoT and the real-time operating system (RTOS) world
    - This gives your team valuable real-world experience
- It's compiled to native machine code
  - No virtual machine interpreters
    - No pausing due to garbage collection
  - It's fast
- It's the native language of the RoboRIO's Linux-based operating system
  - The environment is written in C and Assembler
  - You get easy, direct access to the underlying O/S
- C++ is object oriented
  - Full support from WPILib

Why Not C/C++?

- C/C++ is compiled
  - This adds complexity to the build
- C/C++ is textual
  - There are no cutesy GUIs with lots of obscure symbols and squiggly lines ☺
- There is no VM to catch your mistakes
  - The syntax is similar to Java
    - Java was derived from C++
    - Java VM is written in C/C++
- C/C++ has pointers
  - Objects can be referenced in many different ways
  - This concept can be troublesome for some developers

Why Java?

- Java has wide support in the industry
  - Object-oriented approach with lots of reference material
- Java is the language used on the AP exams
  - Used in many computer science classes
- Java is a byte-code interpreted language
  - The use of the Virtual Machine (VM) allows for many dynamic language features
- The VM will help catch some common memory mistakes
- The version of Java used on the RoboRIO is version 8 from Oracle
- WPILib is actually written in Java and then translated to C++

Why Not Java?

- Java is interpreted
  - Performance is lower than C/C++
- Java is also textual like C++
  - But, Java can be written using either imperative or declarative programming styles
- The version of Java on the RoboRIO is not optimized for use in control systems
  - The version is actually targeted at business applications
- Garbage collection cycle will cause the robot to hesitate during the mark-and-sweep cycle
  - For the duration of the match, this should not be a problem

Top 20 Languages – Nov 2016

Nov 2016	Nov 2015	Change	Programming Language	Ratings	Change
1	1		Java	18.755%	-1.65%
2	2		C	9.203%	-7.94%
3	3		C++	5.415%	-0.78%
4	4		C#	3.659%	-0.66%
5	5		Python	3.567%	-0.20%
6	8	^	Visual Basic .NET	3.167%	+0.94%
7	6	v	PHP	3.125%	-0.12%
8	7	v	JavaScript	2.705%	+0.23%
9	11	^	Assembly language	2.441%	+0.56%
10	10		Perl	2.361%	+0.33%
11	14	^	Objective-C	2.246%	+0.82%
12	15	^	Swift	2.039%	+0.80%
13	48	^^	Go	2.001%	+1.80%
14	9	vv	Ruby	1.978%	-0.06%
15	16	^	MATLAB	1.967%	+0.78%
16	12	vv	Delphi/Object Pascal	1.950%	+0.27%
17	13	vv	Visual Basic	1.923%	+0.24%
18	37	^^	Groovy	1.811%	+1.48%
19	19		R	1.715%	+0.70%
20	18	v	PL/SQL	1.512%	+0.48%

- LabVIEW was #34 on this list

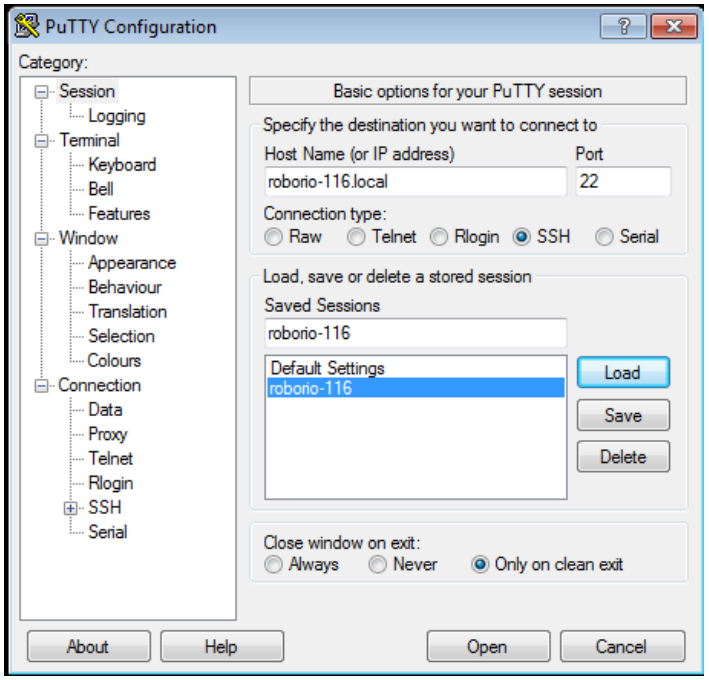
Getting Your RoboRIO Ready

- Before you can start development, you'll need to make sure that your RoboRIO has the proper operating system image on it
  - This is accomplished using the RoboRIO imaging tool or it can be done through LabVIEW

- Java developers will need to deploy the Java 8 JDK to the RoboRIO
  - We'll address this later…

Some Useful Info…

- The RoboRIO runs Linux
  - SSH server is available
    - Use Putty on Windows to get to SSH shell
  - File transfers from IDE use SCP
- Addressing is via mDNS
  - roborio-<team #>-FRC.local
- The web server requires Microsoft Silverlight plug-in
- Do not delete “admin” account
  - All program transfers require it



The Development Environment

- The FIRST-supported development platform for C/C++ and Java is the Eclipse IDE tool
  - The beta teams are using the Neon release
  - The compiler is the open-source GNU 4.9 compiler
    - Supports C++11 extensions
- The C compiler is actually a *cross-compiler*
  - We are building on an x86 for an ARM-based system
    - Again, this is a standard approach for commercial, embedded development
- There is also a way to install the Java bytecode compiler for Eclipse after you install the FRC plugins
  - RoboRIO runs Java bytecode in the Oracle JVM
  - Compiler on development platform produces the bytecode

Development Environment #2

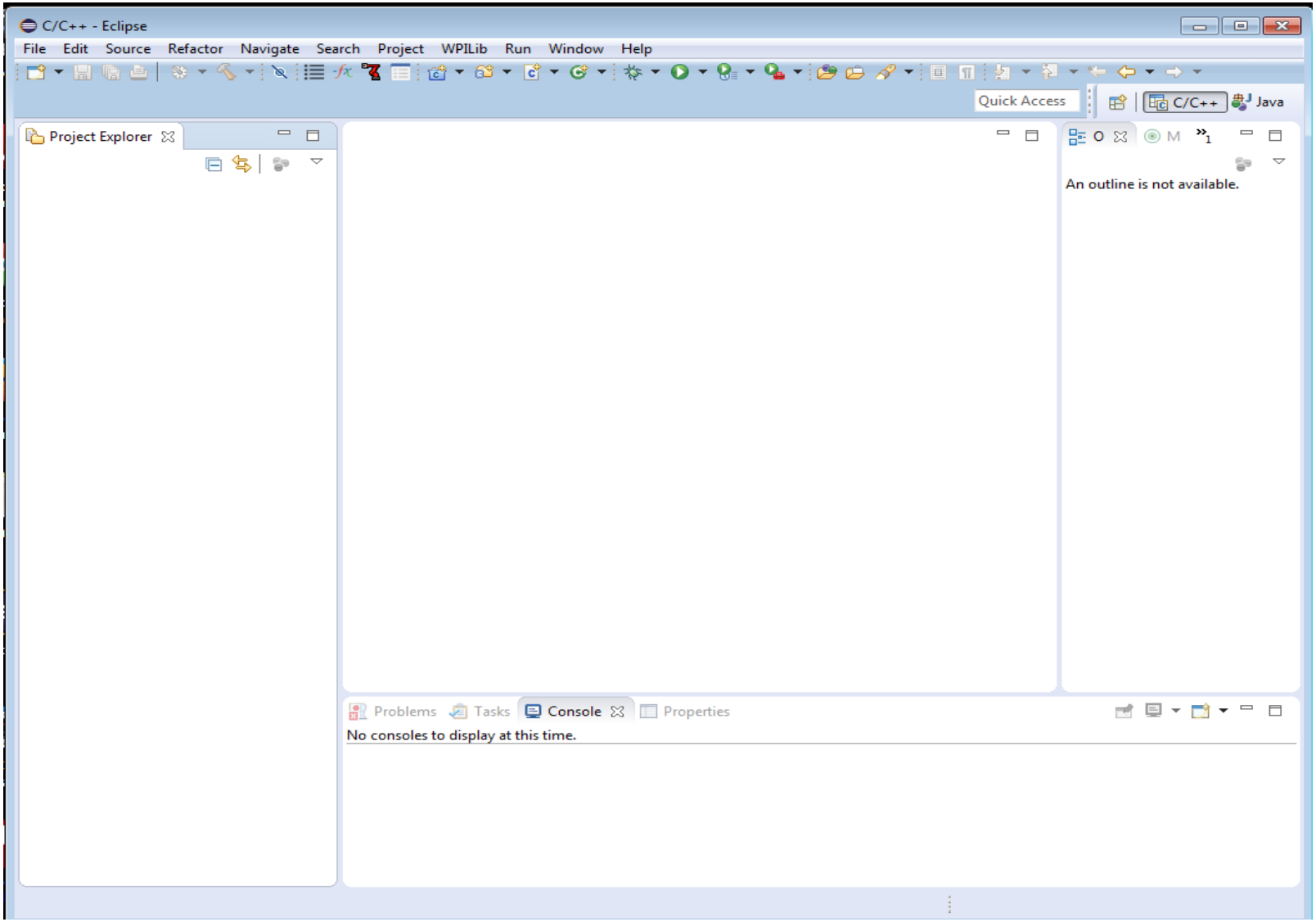
- During the beta, FIRST is only supporting Windows
  - However, libraries and compilers are available for OS/X and Linux
- The beta supports Windows 10
  - You can still run it in a virtual machine
    - For all you Mac OS/X and Linux fans
- 64-bit Windows is supported
- The Java 8 JDK needs to be installed on your platform regardless of whether you use Java or C/C++
  - Make sure you install the JDK and not just the JRE
  - http://java.com

The Eclipse Neon IDE

- To get started, first install Oracle Java 8 JDK on the Windows host
  - Either 32- or 64-bit depending on your host

- Next, go to Eclipse.org and download the Eclipse IDE for C/C++ developers
  - 32- or 64-bit version to match your Java install
  - Works for both C/C++ and Java development
- Once installed, you'll connect to FIRST to download the plug-ins for development
  - This may change for kick-off

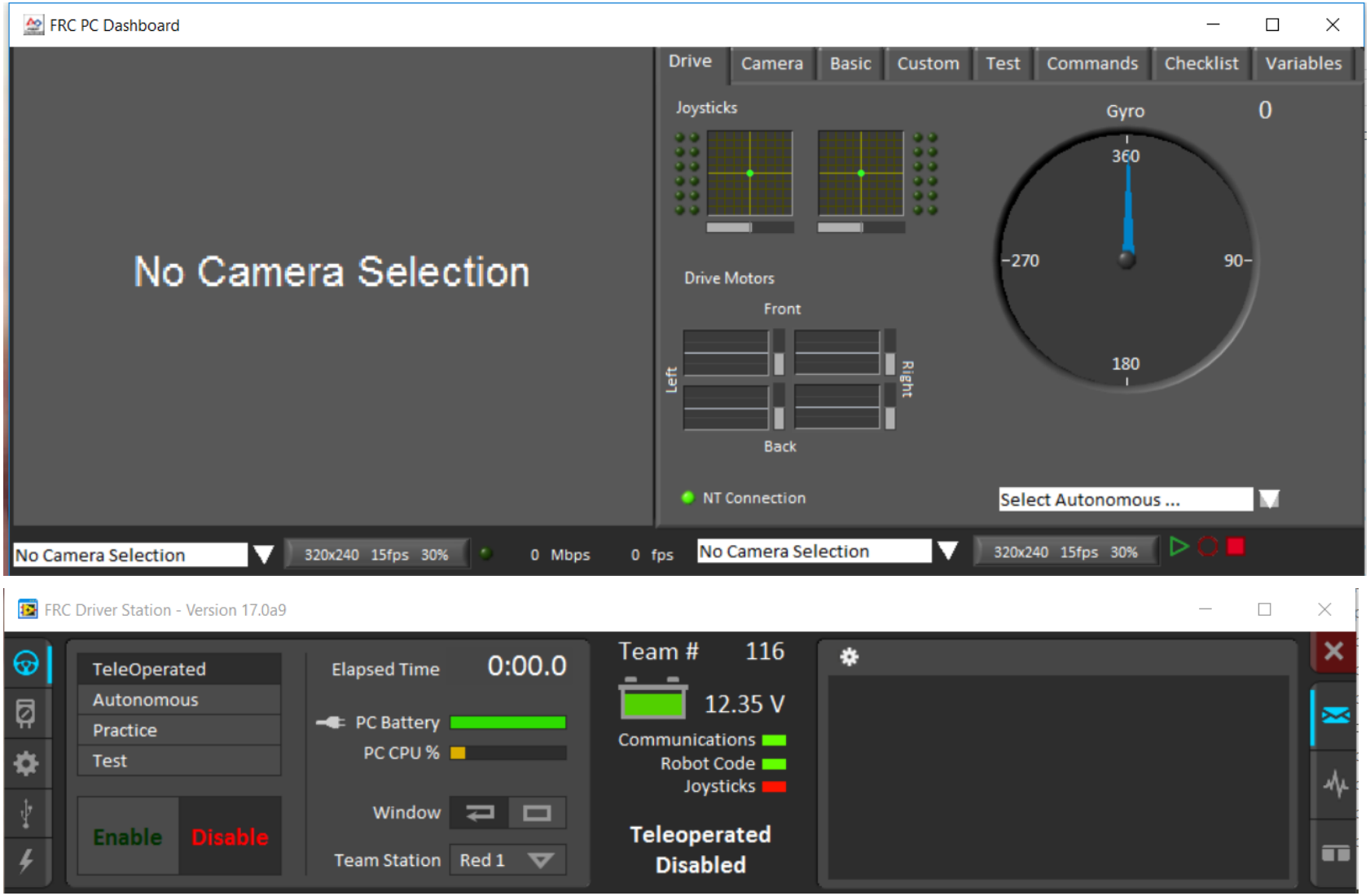
Installed Eclipse



Install FRC 2017 Update Suite

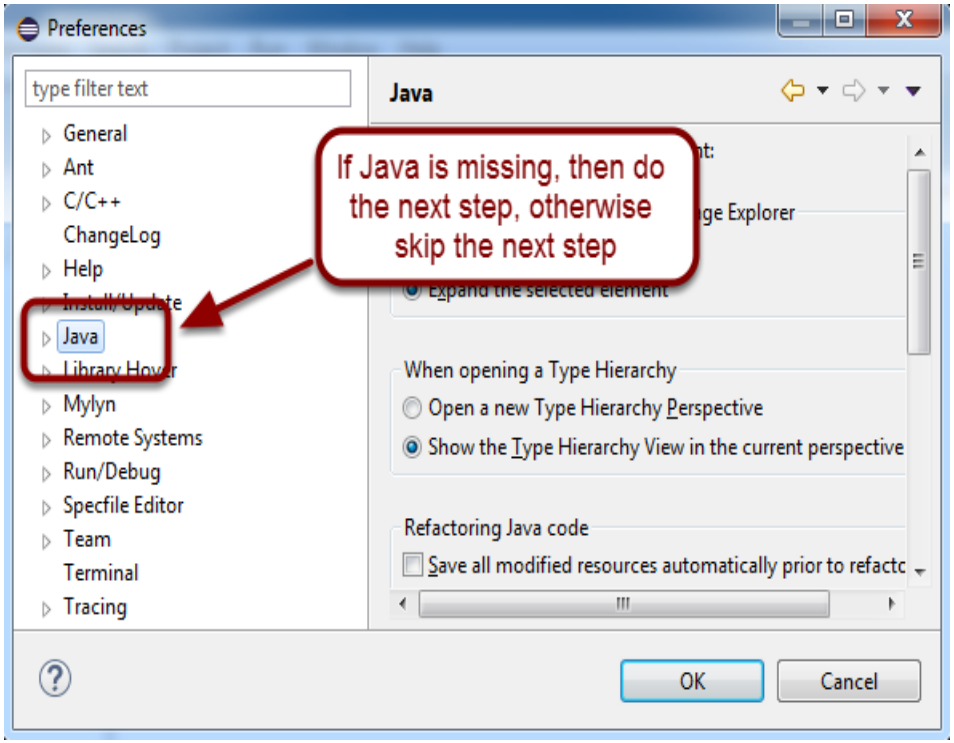
- First, make sure you remove any old LabVIEW or driver station code
- Run the update suite installer
  - This will install the RoboRIO imaging tool and the latest firmware release
  - Installs the latest driver station and smartDashboard
- The 2017 Driver Station doesn't look that different from the 2016 version
  - So far…

2017 Driver Station



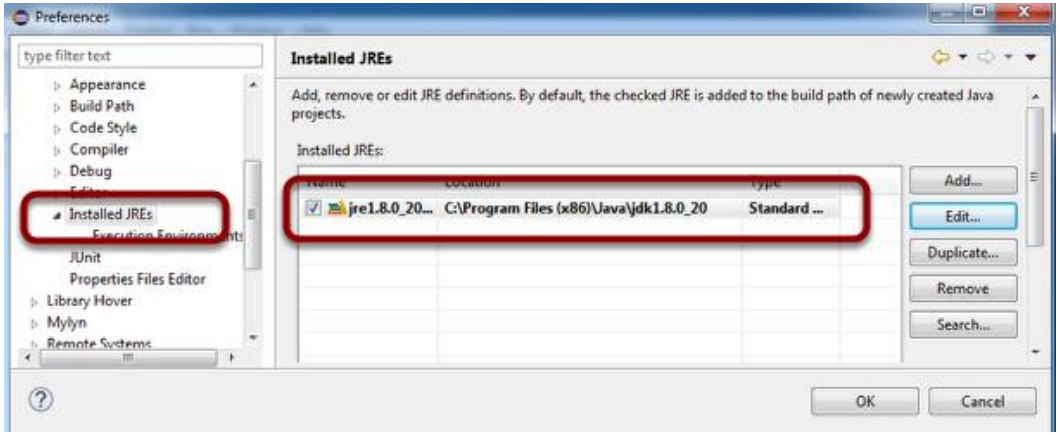
Checking for Java

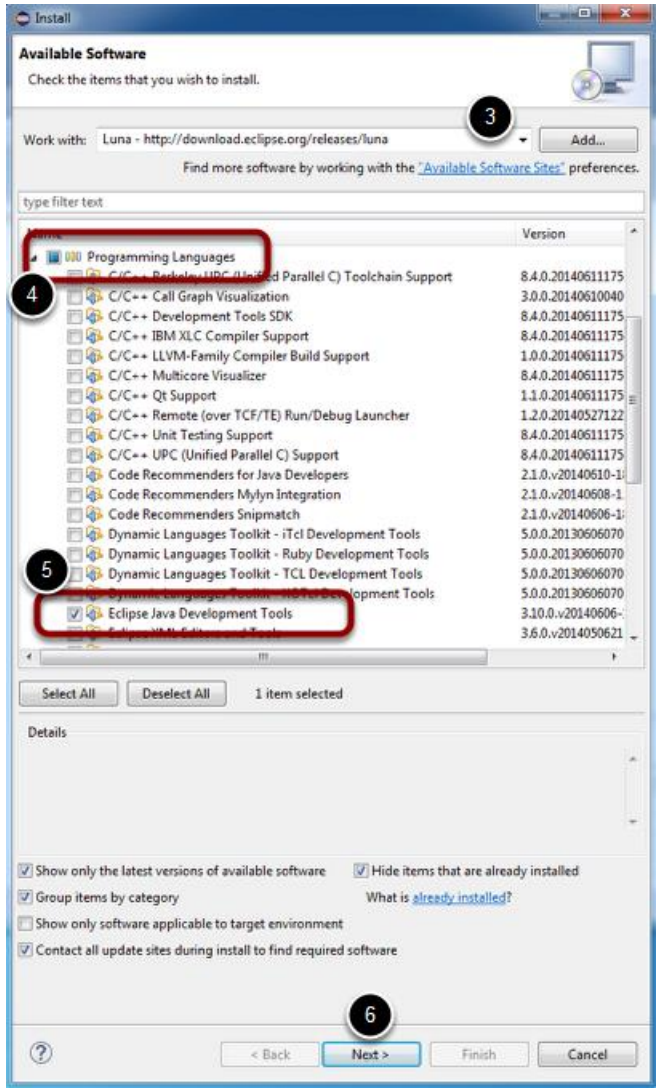
- If you're working with Java, you'll need to verify that the Java tools are installed
- Normally, you won't have to go beyond this step if you installed the Oracle Java on your development station correctly



Find the Java Development Tools

- If for some reason, Java doesn't show up in the previous tab, you'll need to install the Java tools
- Then, you'll set up the JDK in the preferences tab





Install the Third-Party Libraries

- FIRST has unbundled the libraries for motor controllers like the CTRE Talon SRX
  - This will likely include anything that has any special capabilities like the Rev Robotics SPARK as well
- You'll need to contact the manufacturer to get the instructions for integrating their product into the Eclipse environment
  - Typically an installer for Windows
    - OS/X and Linux will have to piece it together
- You'll need to add the libraries and header files to the search path of your project

Now, Java Should Work (on the Host)

- Once you've verified that the Java tools and JDK are visible to Eclipse you should be able to use the File->New->Project option to select and create a new Java-based robot project
- This process looks almost identical to the steps for C/C++
  - For the sake of time we'll show the C++ steps, but the Java steps are similar

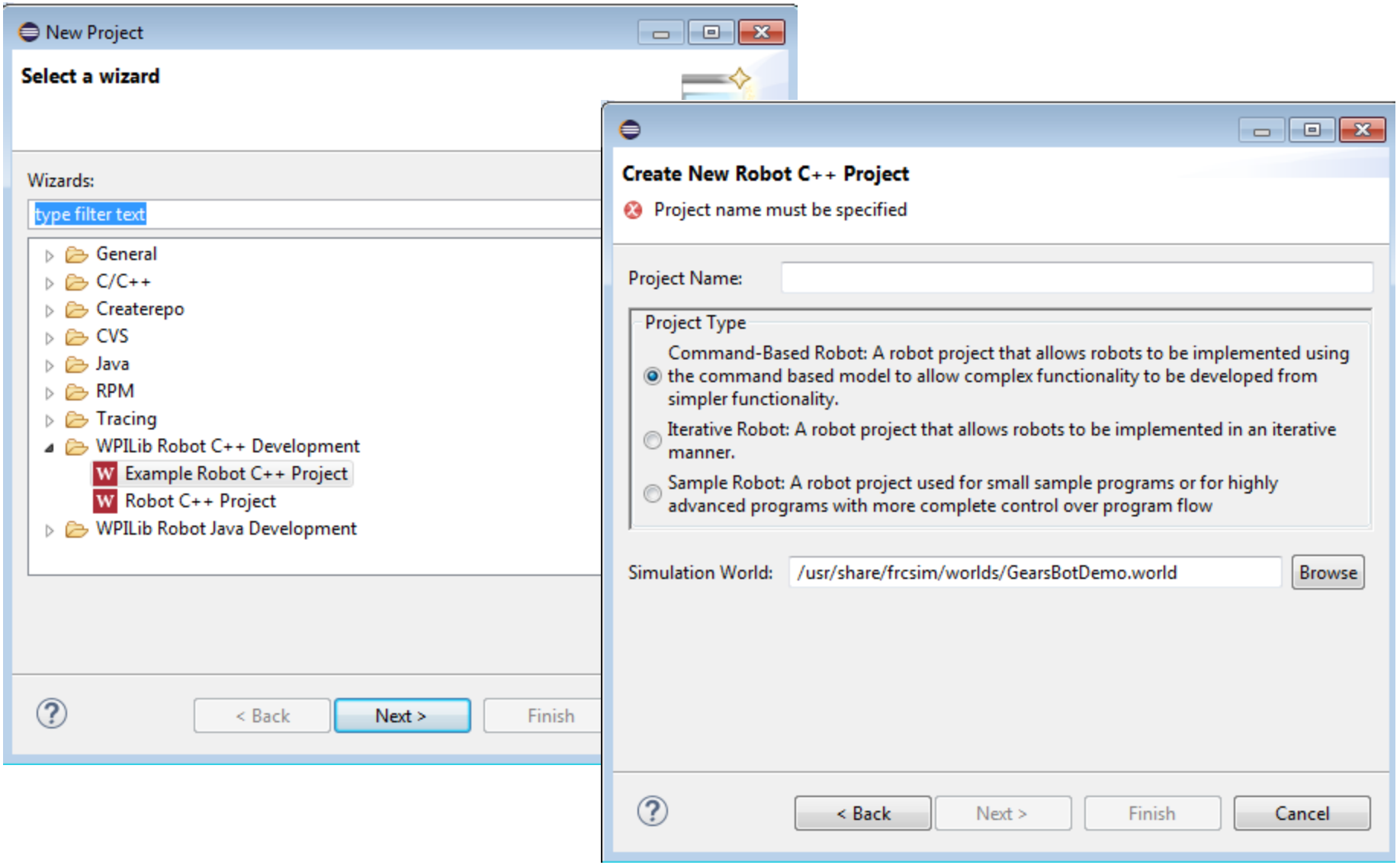
Java on the RoboRIO

- Java does not come installed on the RoboRIO
  - Issues with the license from Oracle
- You need to have your host connected to both the RoboRIO and the Internet
- cd to C:\Users\<username>\wpilib\tools and run java-installer.jar
- Follow the prompts, accept the license and let the system install Java to the RoboRIO
  - Repeat for each RoboRIO

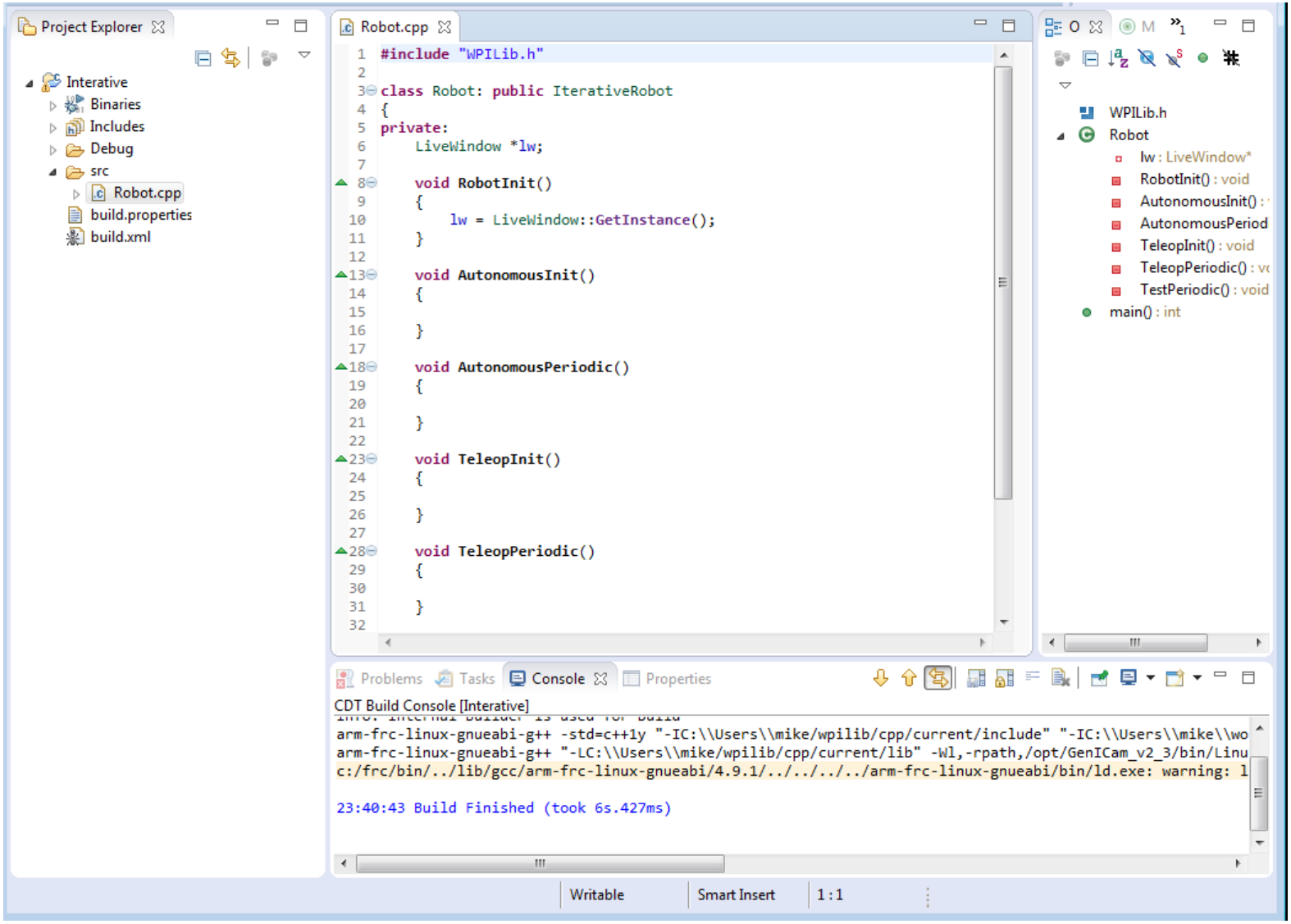
Creating A Project

- Eclipse collects all of the files related to building a piece of code into a subdirectory called a project folder
  - It's normally stored in your C:\users account
    - You can put the project elsewhere when you open Eclipse
- You can also import and export projects
  - This allows you to create a .zip of the project for archival purposes
- To create a new project use:
  - File->New->Project and select WPILib Project

New Project -- Simple Robot

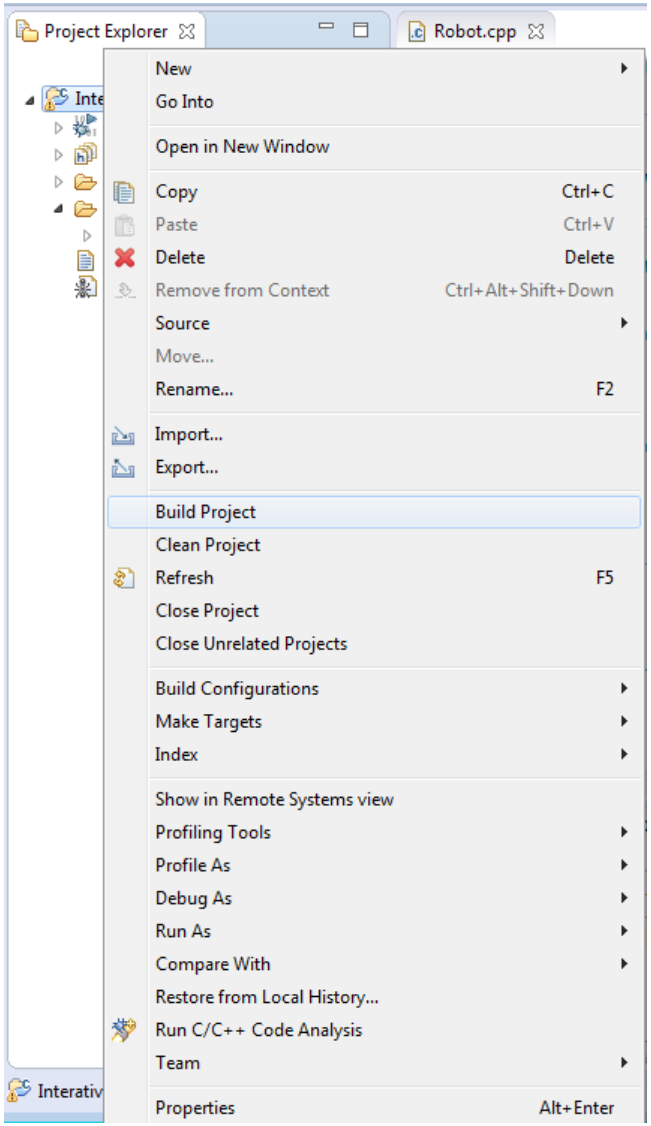


New Project Result

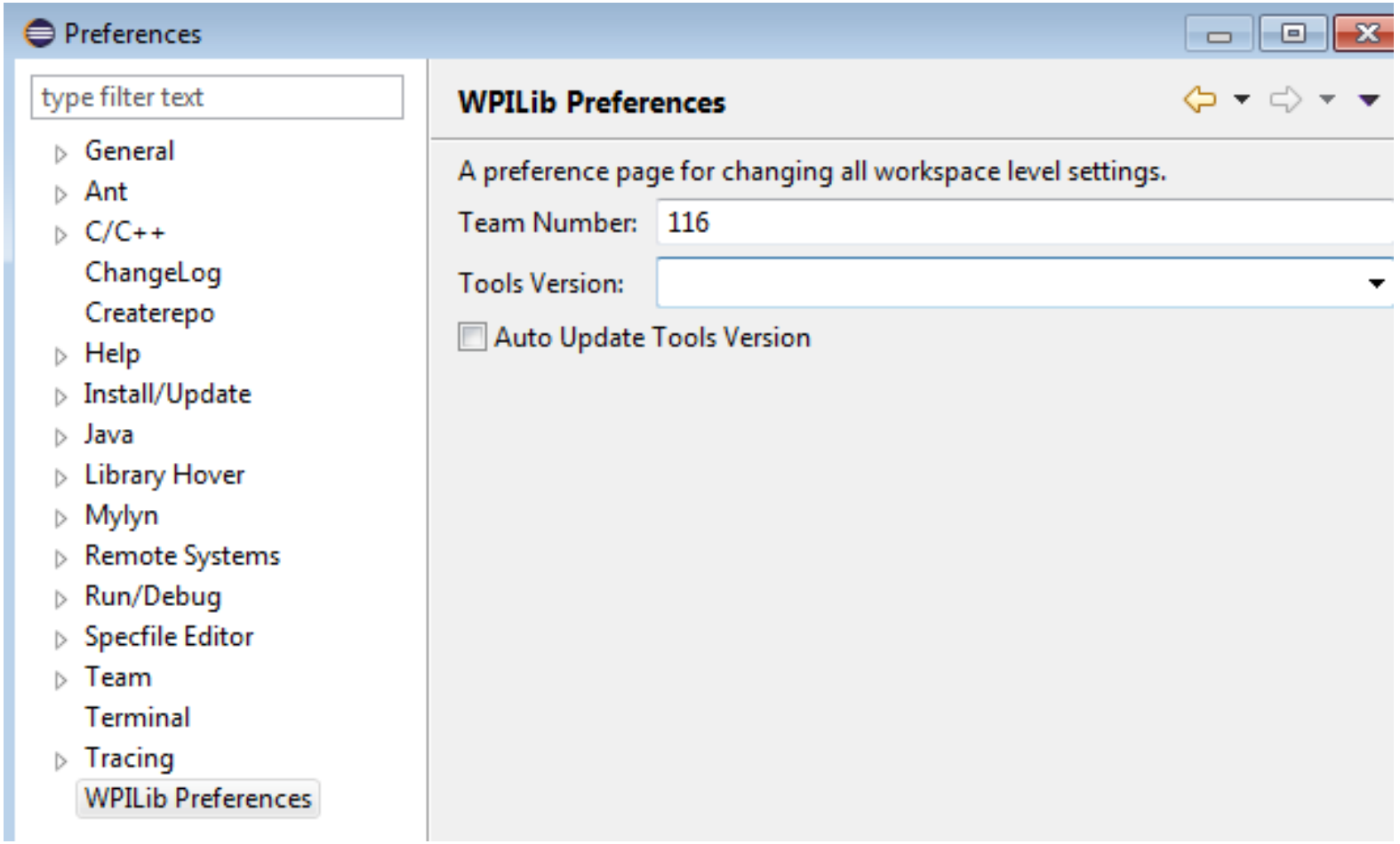


Build the Project

- Eclipse will default to building the project automatically
- However, you can clean and build the project manually
- Use the Project menu to configure the auto-build feature

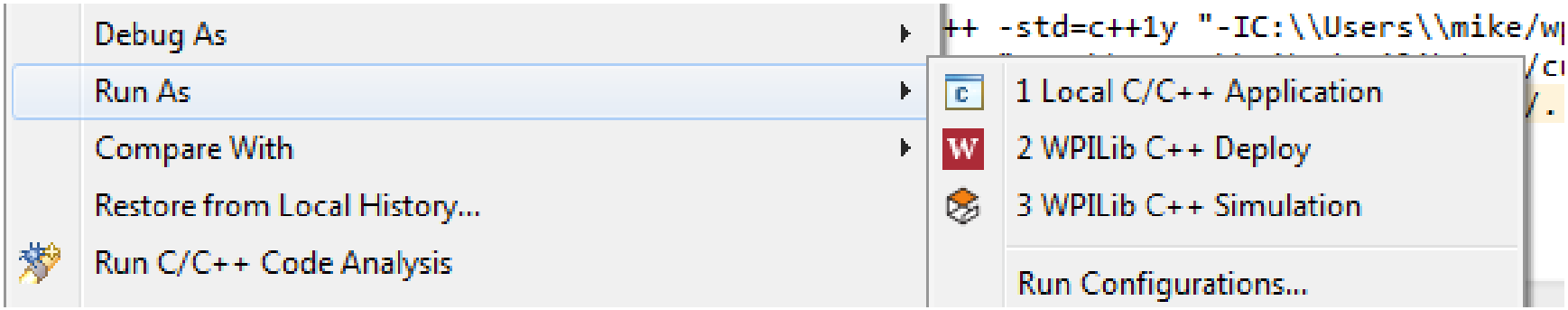


Set Team # in Preferences



Deploying to the Target

- When the code is built, you can select Run As->WPILib C++ Deploy

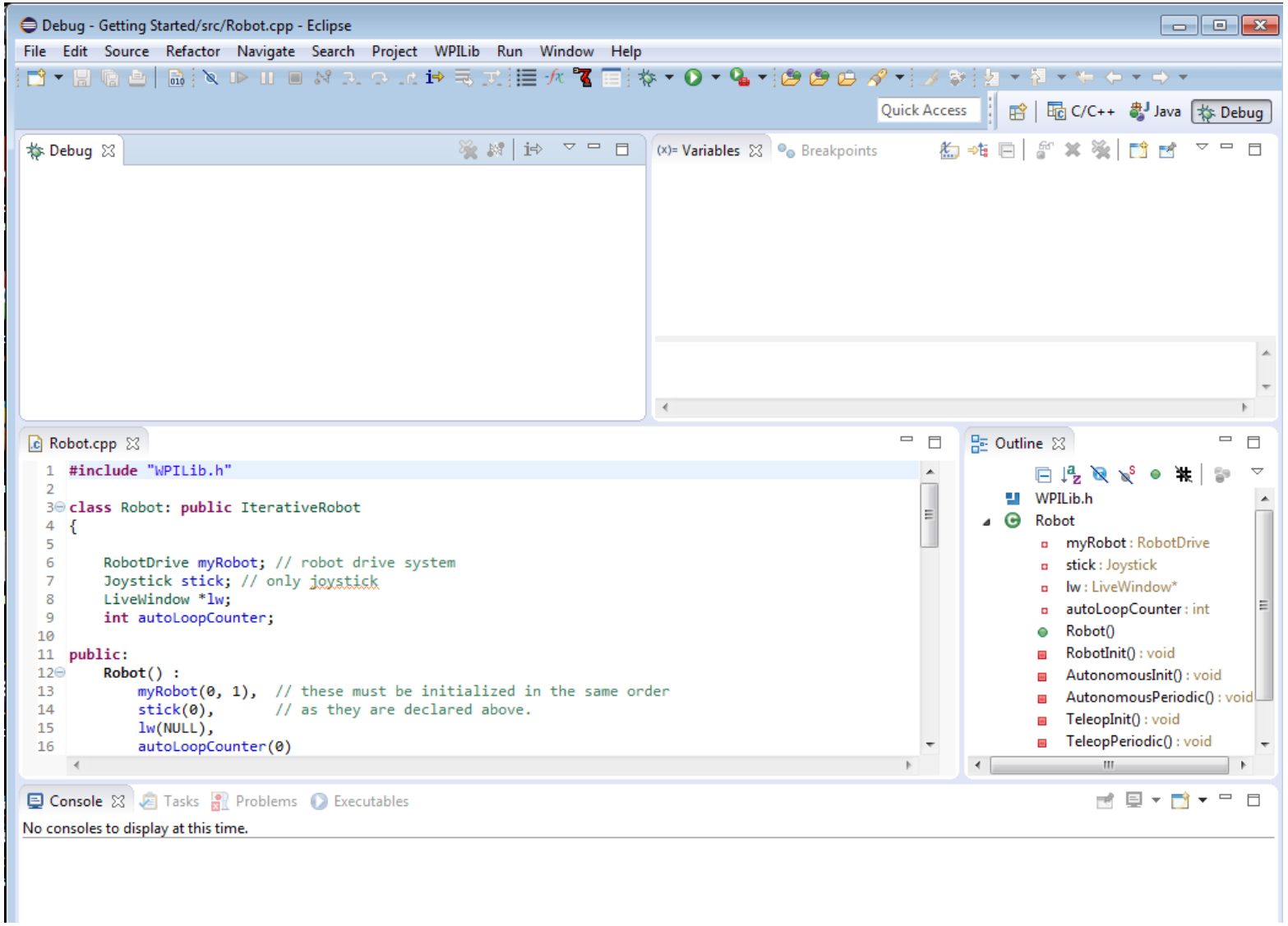


- This will open an SCP connection to the RoboRIO (as “admin”) and copy the application to the file system
- The application will then start running
  - Waiting for the driver station

Debugging Code

- The project is usually automatically created with debugging enabled
- Select Debug As->WPILib C++ Deploy
- The Eclipse will automatically switch to the Debug perspective
- You can then set breakpoints
- Once the driver station enables the application, your debug session will begin

The Eclipse Debug Perspective



Resources

- Chief Delphi
  - http://www.chiefdelphi.com
- FIRST forums
  - http://forums.usfirst.org
- NI Community Forums
  - http://ni.com/FIRST
- WPI / *FIRST* NSF Community site (ThinkTank)
- These sites are monitored members of:
  - WPI
  - NI
  - *FIRST*
- All source code available for team<->team assistance
- Phone support through NI
  - 866-511-6285 (1PM-7PM CST, M-F) ?

Summary

- C/C++ can be very challenging to new developers
  - C/C++ is similar enough to Java that Java developers can adapt to it quickly
    - However, pointers will require some explaining
  - Performance and fine-grain control are the biggest advantages to using C/C++
- Java has a lot of support within the FIRST community and many school systems
  - Being on the AP CS exam encourages schools to teach it
  - Java is also used in the new FTC development environment
    - Although the Java VM is slightly different for Android
- WPILib class libraries have equivalent capability between C++ and Java versions
- Java and C++ are syntactically very similar
  - You could start with one and then switch without too much trouble