C/C++ and Java Installation For 2019 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

Warning: Beta Code…

- What you will see is the 2019 Beta software that we've been working with over the past couple of months

- Some things are likely to change, but it's pretty feature complete at this point
- There were quite a bit of head scratching while we were working with getting things running
  - The approach is quite a bit different than in years past

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 11 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
  - Given the length of our matches, this should not be a problem

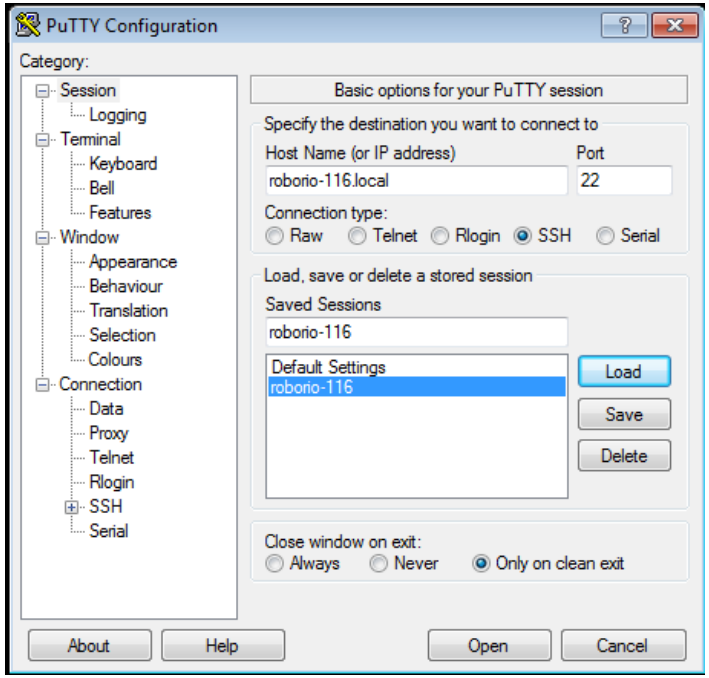
Top 20 Languages – Dec 2018

Dec 2018	Dec 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.932%	+2.66%
2	2		C	14.282%	+4.12%
3	4	^	Python	8.376%	+4.60%
4	3	v	C++	7.562%	+2.84%
5	7	^	Visual Basic .NET	7.127%	+4.66%
6	5	v	C#	3.455%	+0.63%
7	6	v	JavaScript	3.063%	+0.59%
8	9	^	PHP	2.442%	+0.85%
9	-	^^	SQL	2.184%	+2.18%
10	12	^	Objective-C	1.477%	-0.02%
11	16	^^	Delphi/Object Pascal	1.396%	+0.00%
12	13	^	Assembly language	1.371%	-0.10%
13	10	v	MATLAB	1.283%	-0.29%
14	11	v	Swift	1.220%	-0.35%
15	17	^	Go	1.189%	-0.20%
16	8	vv	R	1.111%	-0.80%
17	15	v	Ruby	1.109%	-0.32%
18	14	vv	Perl	1.013%	-0.42%
19	20	^	Visual Basic	0.979%	-0.37%
20	19	v	PL/SQL	0.844%	-0.52%

- LabVIEW was #35 on this list

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 on the RoboRIO is being redesigned at this time so we don't quite know what it will look like yet
  - However, last year's requirement for Microsoft Silverlight seems to be gone ☺
- Do not delete "admin" account
  - All program transfers require it



The Development Environment

- The FIRST-supported development platform for C/C++ and Java is Microsoft Visual Studio Code tool
  - Available for Windows, MacOS and Linux
  - The compiler is the open-source GCC 6.3 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
- For Java, the build system will run the Java source code through the Oracle JDK to produce Java bytecode

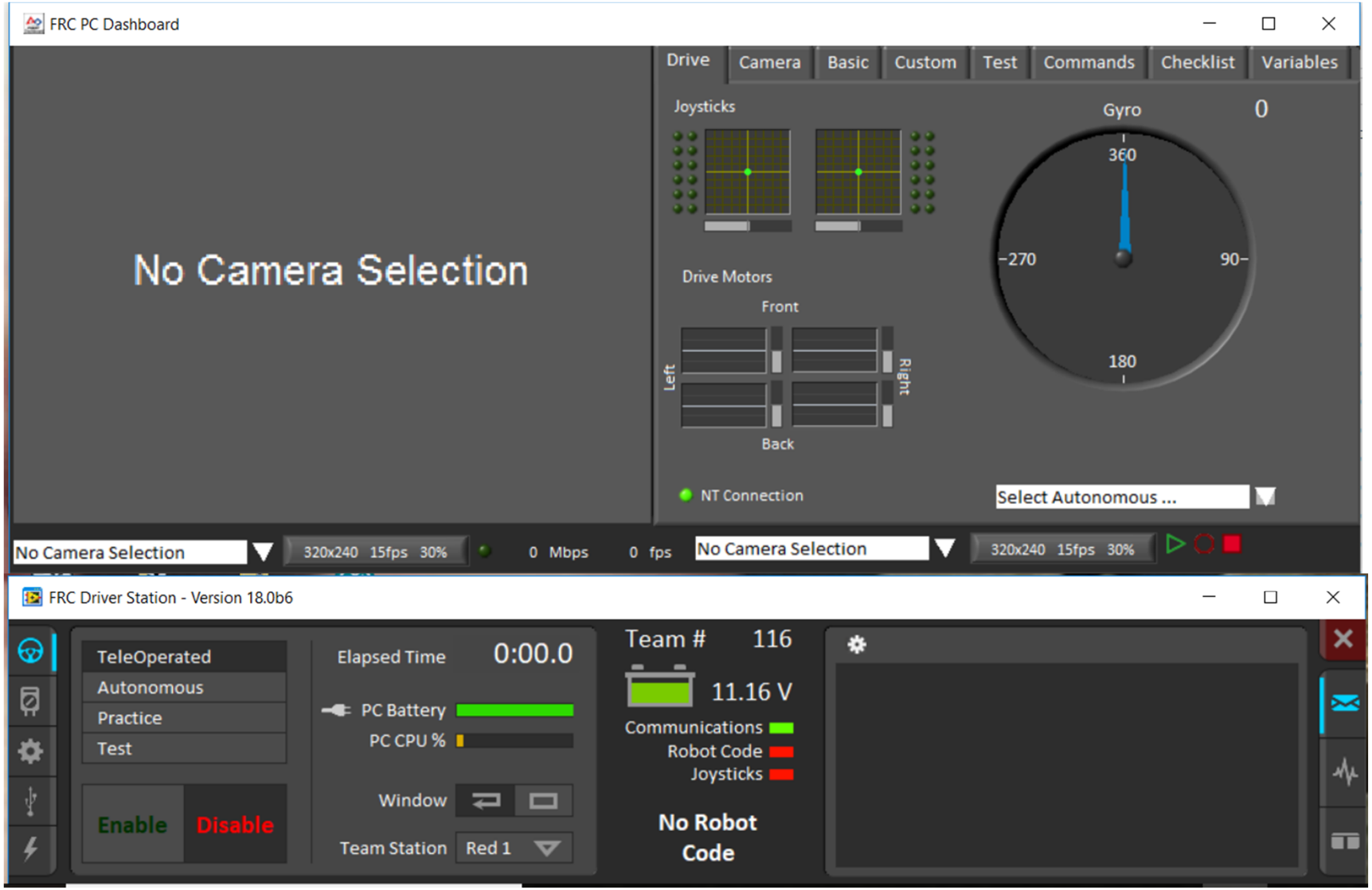
Development Environment #2

- The installation tool will install the Oracle JDK
  - And, install VSCode if you select that option
- The build environment is the GradleRIO plug-in from Github
  - https://github.com/wpilibsuite/GradleRIO
- The WPILib VSCode plug-in will have all of the tools needed to build and deploy code to the robot

Install National Instruments Update

- It's probably best if you uninstall previous versions
  - Delete the <user>/wpilib directory as well
  - It will take at least 10-20 minutes to install
    - Longer if you need to uninstall the previous version
- This will also install the FRC Driver Station application
  - This will also install the RoboRIO imaging tool and the latest firmware release
- The system will reboot after installation

2019 Driver Station

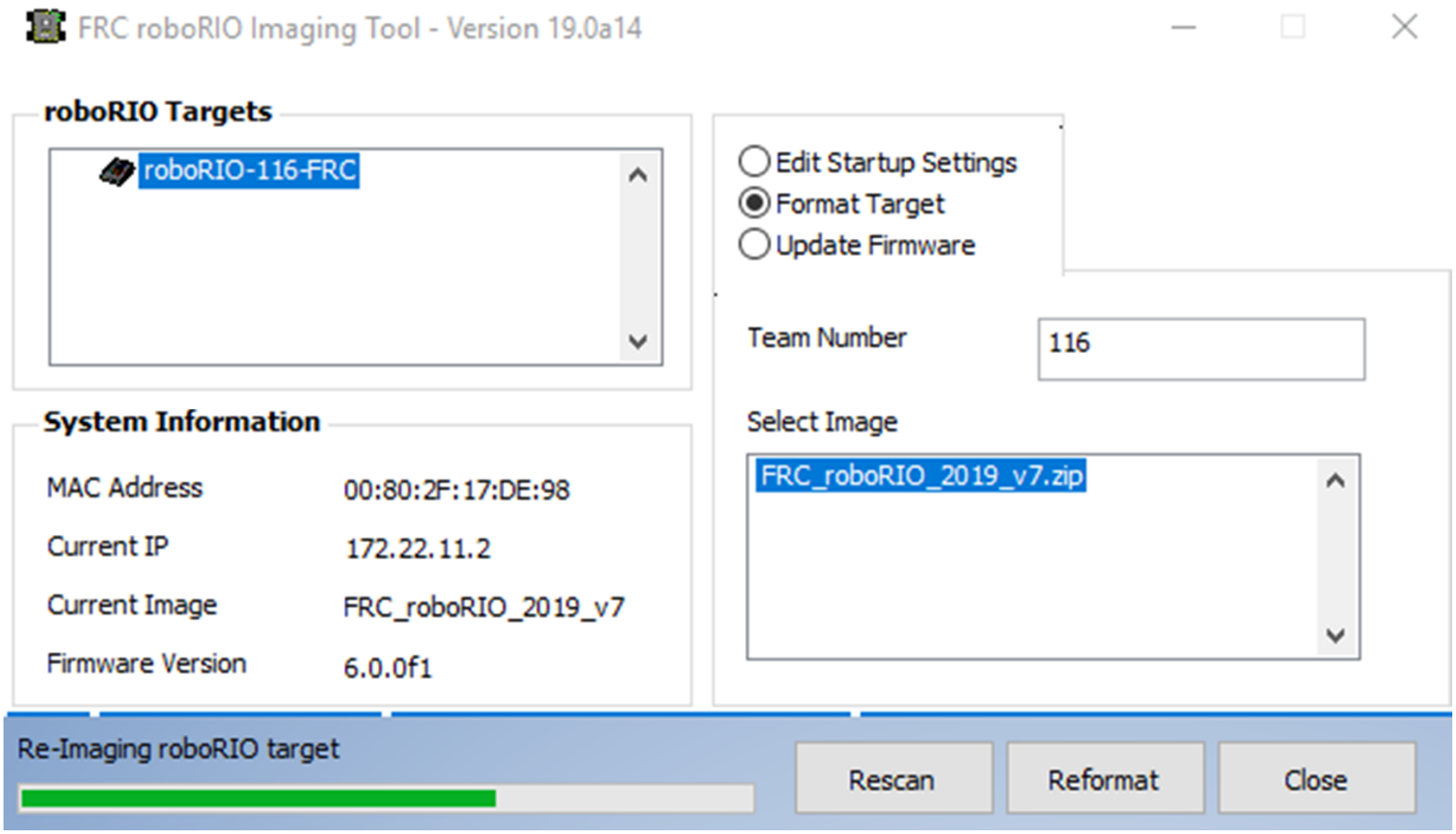


Getting Your RoboRIO Ready

- 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

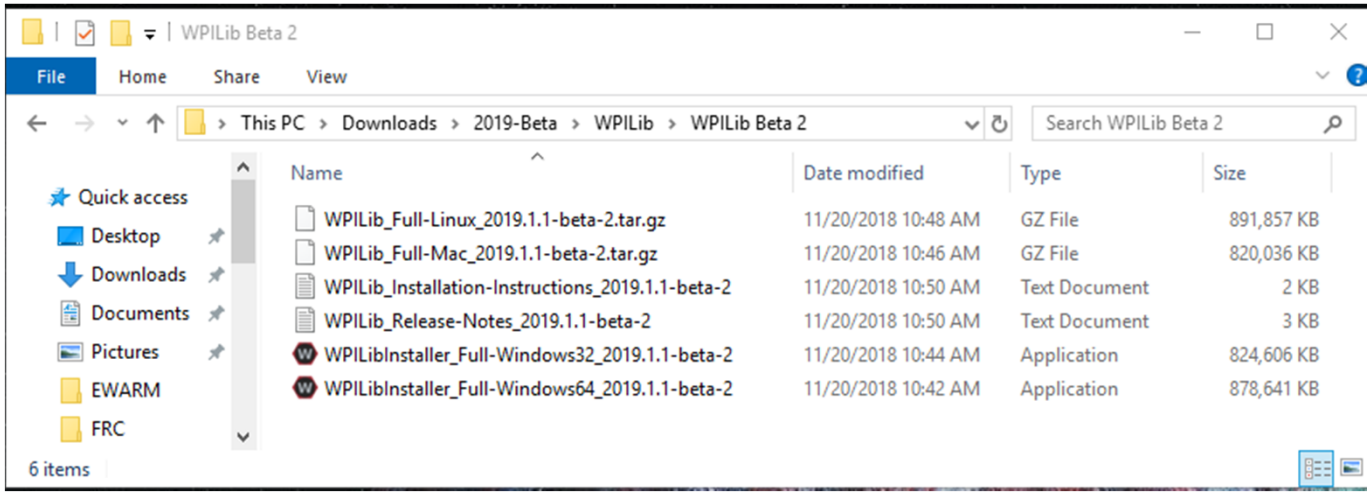
- The RoboRio imaging tool will automatically install Java on the the RoboRio

Update the RoboRIO



Launch the WPILib/tools Install

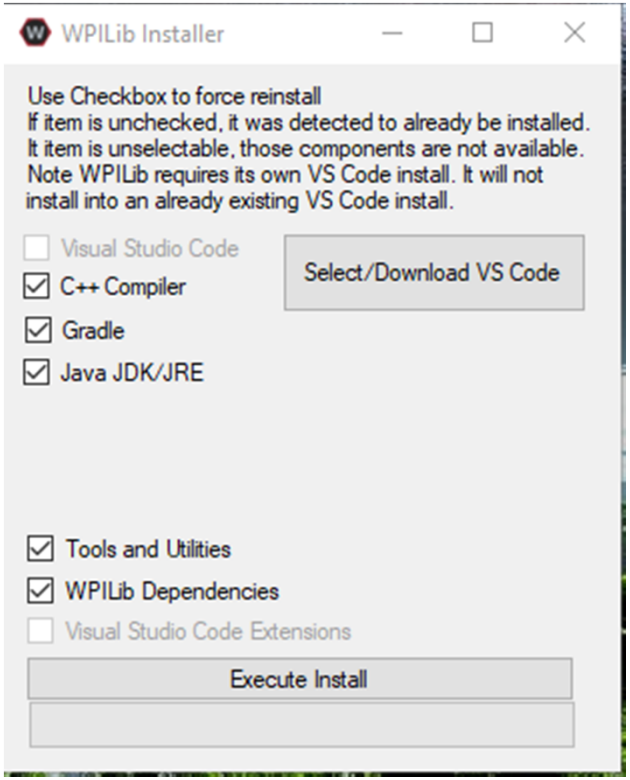
- Unlike last year, the WPILib tools are extracted from a separate archive
  - ~ 3.25 GBs for the zipped download
- We'll look at the Windows installation, but there are install steps for both MacOS and Linux as well

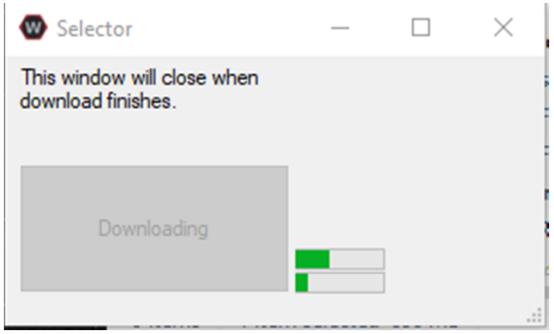


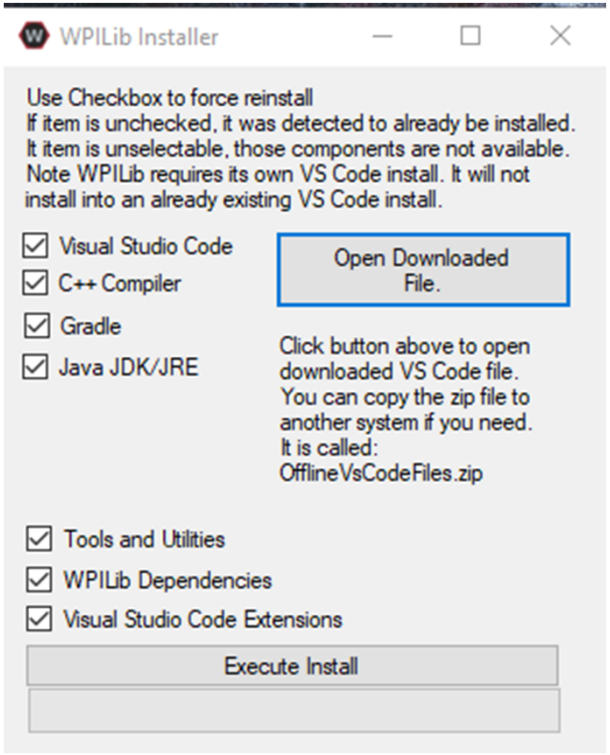
Installation of Visual Studio Code

- In theory, you should be able to use an existing VSCode installation
  - That didn't work too well in the Beta, so we opted to allow the installation tool to install VSCode for us
- The installation will take about 10 minutes
  - There are still some manual settings that you'll need to do with search paths for the JDK and the JAVA_HOME environment variable
    - Requires that you run a script to update these things
  - Presumably, these things will be taken care of by kickoff

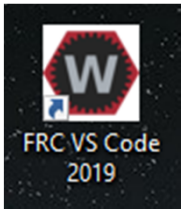
Installing WPILib/VSCode

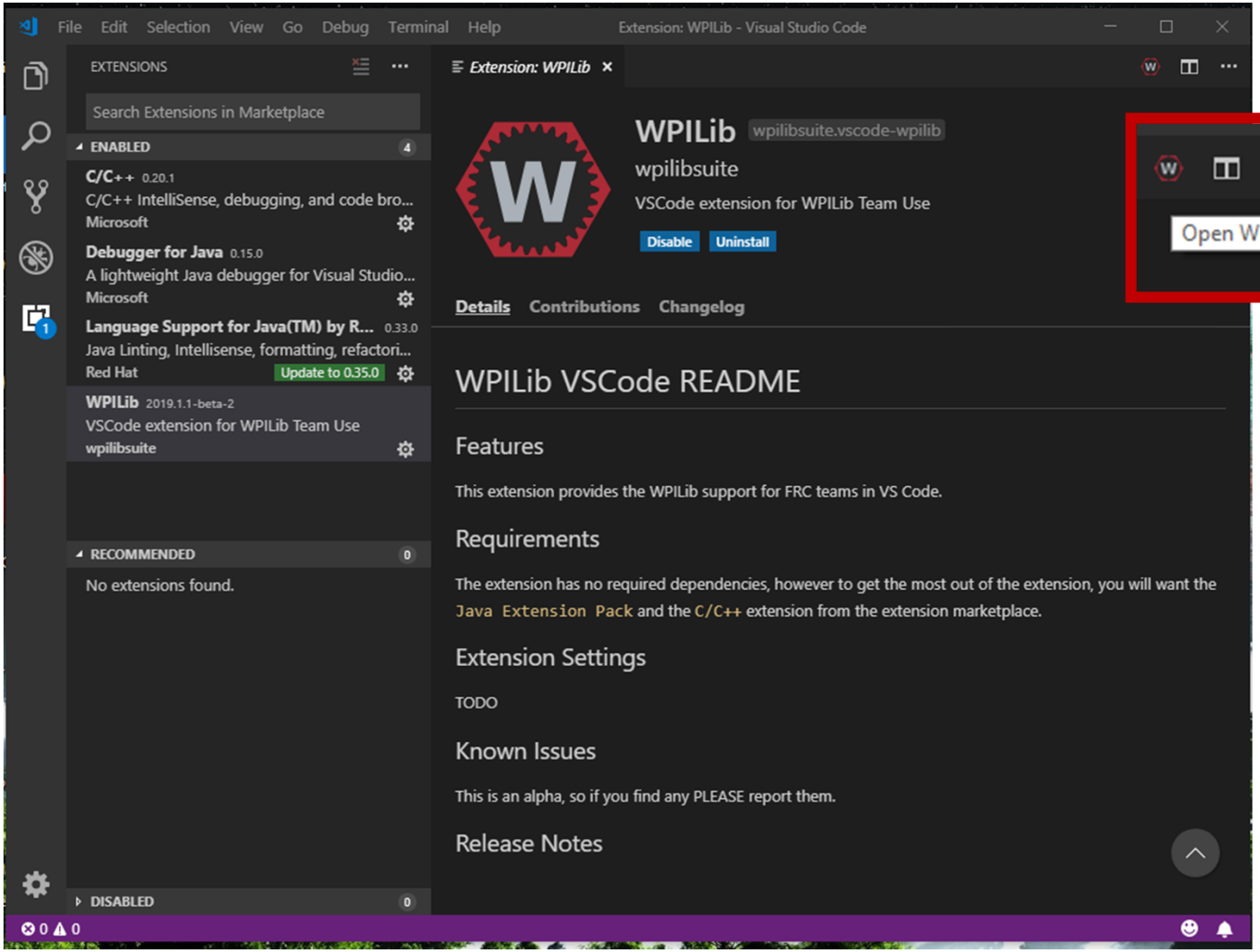




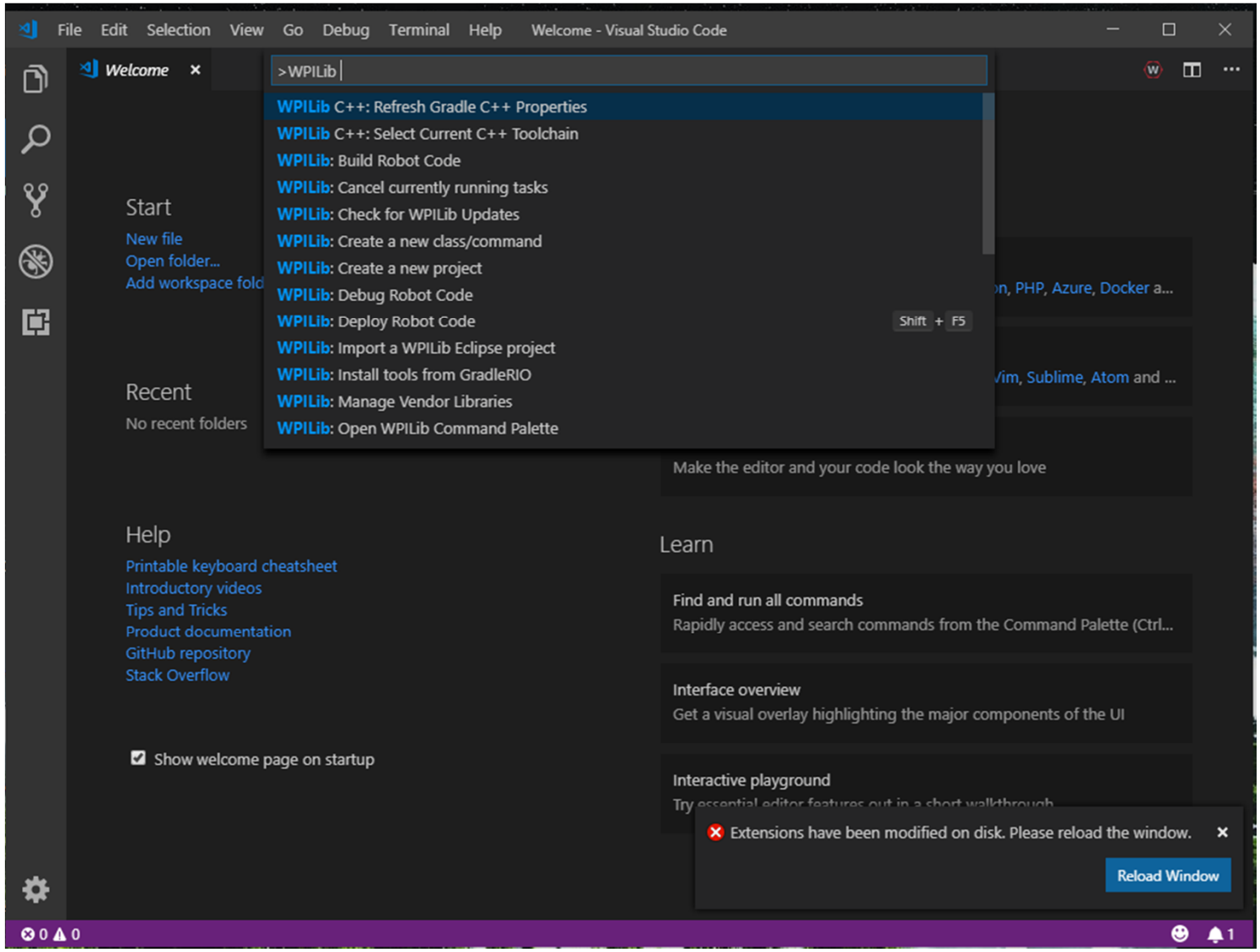


The VSCode with WPILib Extension





Creating a Project #1



Creating a Project #2

Resulting Project

EXPLORER

OPEN EDITORS

Robot.cpp src\main\cpp

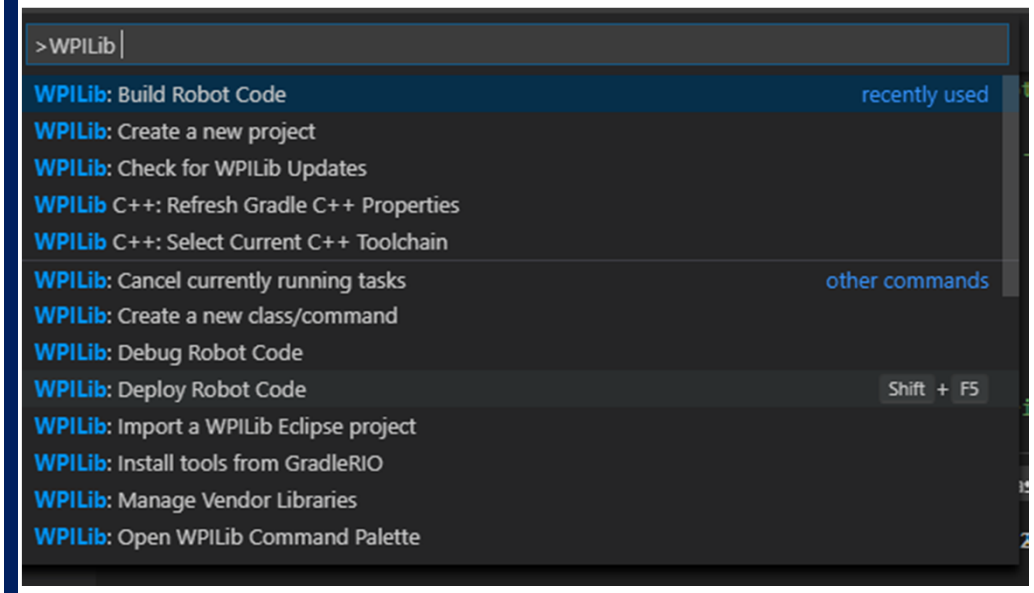
ARCADE-EXAMPLE

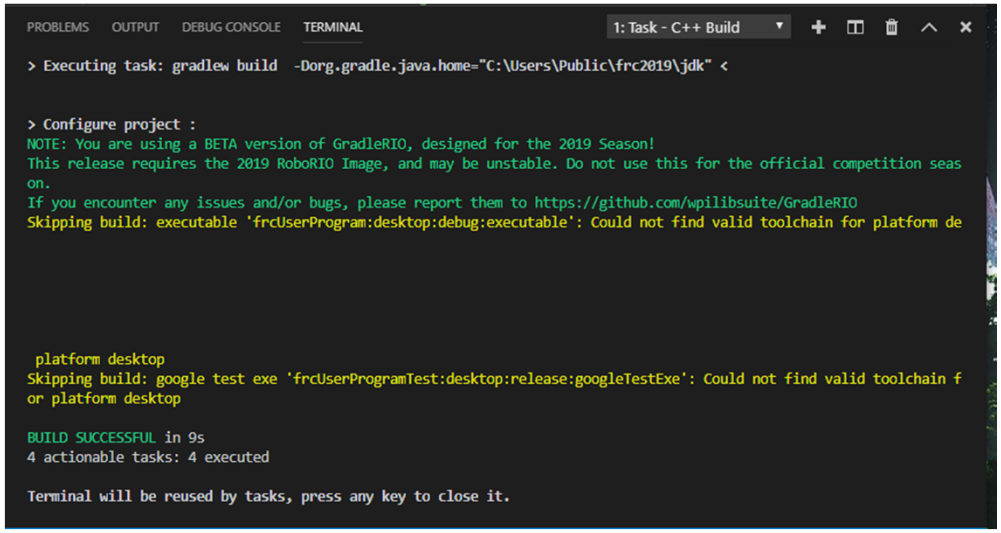
- .gradle
- .vscode
- .wpilib
- build
- gradle
- src
  - main
    - cpp
      - Robot.cpp
    - deploy
  - test
- .gitignore
- build.gradle
- gradlew
- gradlew.bat
- settings.gradle

Robot.cpp

```cpp
/*----------------------------------------------------------------------------*/
/* Copyright (c) 2017-2018 FIRST. All Rights Reserved.                        */
/* Open Source Software - may be modified and shared by FRC teams. The code   */
/* must be accompanied by the FIRST BSD license file in the root directory of */
/* the project.                                                               */
/*----------------------------------------------------------------------------*/

#include <frc/Joystick.h>
#include <frc/Spark.h>
#include <frc/TimedRobot.h>
#include <frc/drive/DifferentialDrive.h>

/**
 * This is a demo program showing the use of the DifferentialDrive class.
 * Runs the motors with arcade steering.
 */
class Robot : public frc::TimedRobot {
  frc::Spark m_leftMotor{0};
  frc::Spark m_rightMotor{1};
  frc::DifferentialDrive m_robotDrive{m_leftMotor, m_rightMotor};
  frc::Joystick m_stick{0};

 public:
  void TeleopPeriodic() {
    // Drive with arcade style
    m_robotDrive.ArcadeDrive(m_stick.GetY(), m_stick.GetX());
  }
};
```

Build and Deploy

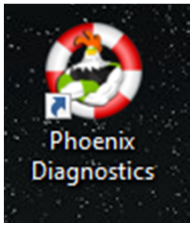




Install the Third-Party Libraries

- The CTRE and Kauaii Labs libraries are unbundled from the WPILib development environment
  - You will need to install these libraries separately into the VSCode workspace
- CAN bus is a feature now of several FRC-legal motor controllers
- For CTRE motor controllers, you will need to install the CTRE Phoenix framework onto your platform
  - The Phoenix Diagnostics application will enable you to update your CAN firmware for the PDP, PCM, Talon SRX and Victor SPX devices
- You'll need to add the libraries and header files to the search path of your project using the VSCode external library mechanism

Configure CAN Bus (CTRE)



Dashboard Version (0.4.4.0)

Options Tools

Prepare the Target Robot Controller | CAN Devices | Web Diagnostics Log

Device Name	Software Status	Hardware	ID	Firmware Version	Manufacturer Date	Bootloader Revision	Hardware Version	Vendor
PCM (Device ID 2)	Running Application.	PCM	2	1.65	June 17, 2015	3.0	1.6	Cross The Rc
PDP (Device ID 1)	Running Application.	PDP	1	1.40	July 14, 2015	3.1	Smart Module 1.1, Ba...	Cross The Rc

General Device Configuration

Change the ID: 2 — Change ID

Change the name: PCM (Device ID 2) — Change Name

Press to animate device LEDs and confirm ID is correct. — Blink

Field-Upgrade Device Firmware

Select CRF and Press "Update Firmware" to flash new firmware.

...

Update Device

Self-Test

PCM IS NOT ENABLED! If robot is enabled maybe the ID is wrong?
Close-Looping is ON, but PCM is DISABLED.
Comp Is Off
Pressure is full.

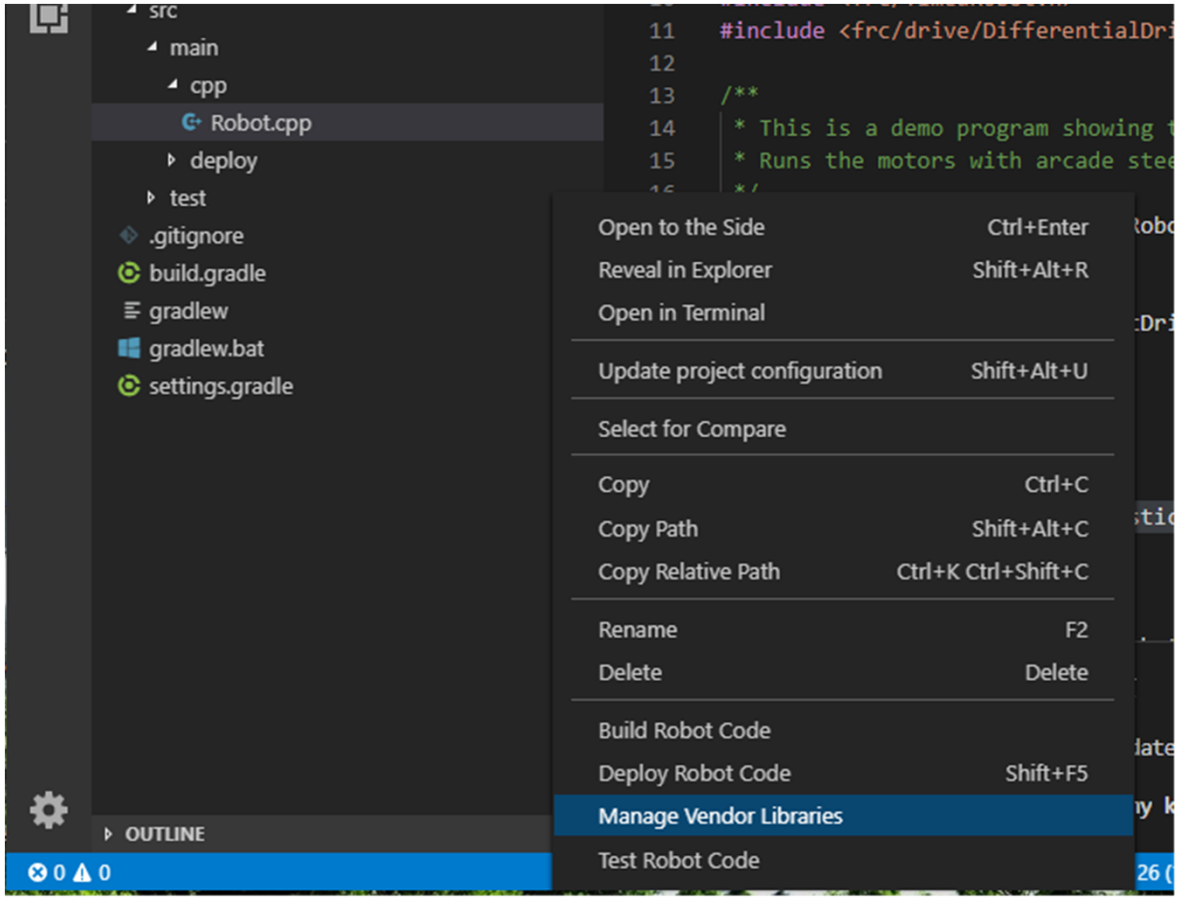
(Fault)	(Now)	(Sticky)	
Comp Curr Too High	0	0	
Comp Short Circuit	0	0	
Solenoid Fuse	0	0	
Comp Curr Too Low	0	1	Most likely the compressor is not connected.
Solen 0	0		
Solen 1	0		
Solen 2	0		
Solen 3	0		
Solen 4	0		
Solen 5	0		

Save Settings | Refresh Configs | Self-Test

Updating http://172.22.11.2:1250 CTRE Devices... Ok Server Version: 0.3.2.0 - PreRelease

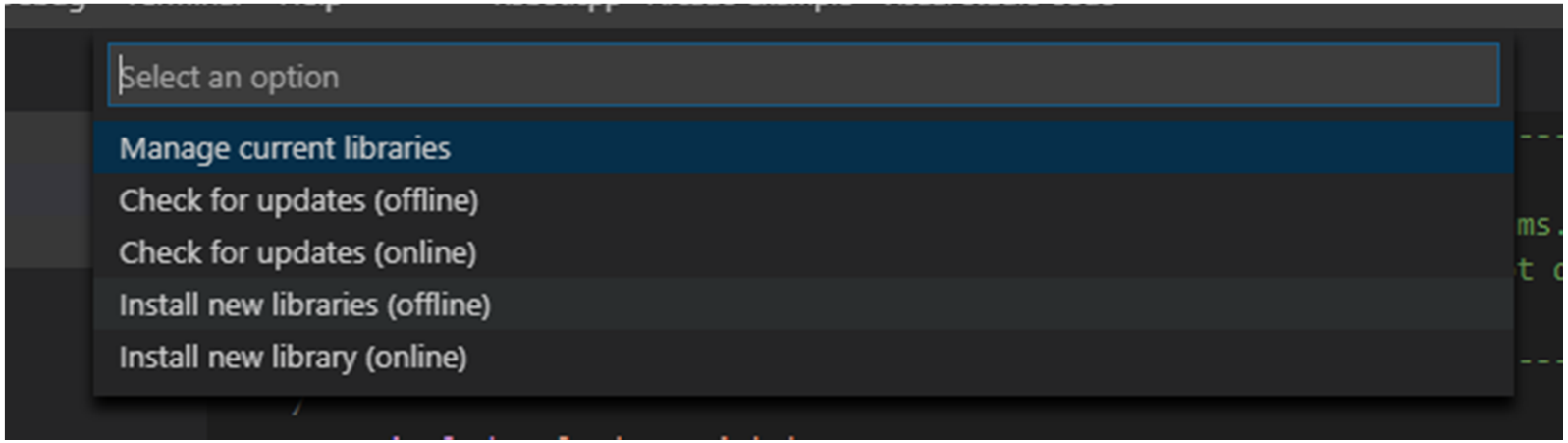
Install 3rd-Party Library into Your Project

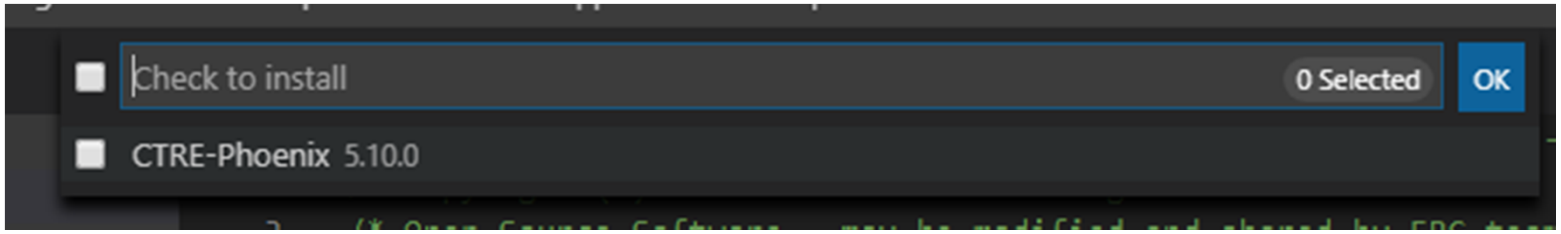
- Before you can use the 3rd-party libraries, you'll need to import them into your project



3rd-Party #2

- Select the “Install new libraries (offline)” and then select the library you want to install





3rd-Party #3

- Once the library is installed in your project, you can start using the features it provides
- You'll need to make sure you've got the header files or imports listed
  - Or, the build will fail miserably
- Once built, you can deploy the 3rd-party goodness to the robot

Example Java Robot Program

EXPLORER

OPEN EDITORS
- WPILib Help
- Main.java src\main\java\frc\robot
- Robot.java src\main\java\frc\robot

TANKDRIVE
- .gradle
- .settings
- .vscode
- .wpilib
- gradle
- src
  - main
    - deploy
    - java
      - frc
        - robot
          - Main.java
          - Robot.java
- .gitignore
- build.gradle
- gradlew
- gradlew.bat
- settings.gradle

OUTLINE

WPILib Help | Main.java | Robot.java

```java
/*----------------------------------------------------------------------------*/
/* Copyright (c) 2017-2018 FIRST. All Rights Reserved.                        */
/* Open Source Software - may be modified and shared by FRC teams. The code   */
/* must be accompanied by the FIRST BSD license file in the root directory of */
/* the project.                                                               */
/*----------------------------------------------------------------------------*/

package frc.robot;

import edu.wpi.first.wpilibj.Joystick;
import edu.wpi.first.wpilibj.Spark;
import edu.wpi.first.wpilibj.TimedRobot;
import edu.wpi.first.wpilibj.drive.DifferentialDrive;

/**
 * This is a demo program showing the use of the RobotDrive class, specifically
 * it contains the code necessary to operate a robot with tank drive.
 */
public class Robot extends TimedRobot {
  private DifferentialDrive m_myRobot;
  private Joystick m_leftStick;
  private Joystick m_rightStick;

  @Override
  public void robotInit() {
    m_myRobot = new DifferentialDrive(new Spark(0), new Spark(1));
    m_leftStick = new Joystick(0);
    m_rightStick = new Joystick(1);
  }

  @Override
  public void teleopPeriodic() {
    m_myRobot.tankDrive(m_leftStick.getY(), m_rightStick.getY());
  }
}
```

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 by 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