Java SDK Startup Guide

TABLE OF CONTENTS

1 Getting Started 1
1.1 Introduction 1
1.2 Prerequisites 1
1.3 Additional Resources 2
2 Android Operating System Basics 2
2.1 What is Andriod? 2
2.2 Andriod Studio 2
2.3 Java 2
3 Installing Android Studio 3
4 Installing the FIRST Global SDK 7
5 Using your Robot 10
5.1 Creating an Op Mode 10
5.2 Building the Robot Controller App 15
6 Troubleshooting 19

Revision	Release Date	Document Changes
Rev 0	4/12/2017	Initial Release
Rev 1	5/24/2017	Minor Install Instruction Change, fixed numbering error, minor edits for clarity

1 Getting Started

1.1 Introduction

The *FIRST* Global Challenge uses an Android-based control system for its competition robots. This document provides basic information on how to install, configure and use the FIRST Global Software Development Kit (SDK) to program the robot using Java in the Android Studio interface. Programming is how teams customize the behavior of the robot.

Many teams will program in Java, but it is not required. Teams can program their robots using other methods such as the Block programing interface covered in the FIRST Global Block Programming Guide available on the FIRST Global resources webpage.

The robot program will need to be completely written in one language; it is not recommended to try to mix and match code.

1.2 Prerequisites

In order to complete the exercises that are contained in this document, it is assumed you have done the following:

1. Read and complete all steps in the "Control System Startup Guide" from the FIRST Global Website.
2. Be familiar with setting up a configuration via the "Configure Robot" menu selection on the Driver Station.
3. Have an understanding of the basic syntax of java and be familiar with concepts such as classes, members, methods and program structure. Oracle offers free java tutorials: http://docs.oracle.com/javase/tutorial/
4. Completed one of the following 2-motor basic robot examples.
    a. Follow the miniBot Build example (Figure 1) on the FIRST Global resources webpage.
    b. Visit the Robot Kit Instructional Videos web page, and view the Practice-Bot Build Walkthrough video.

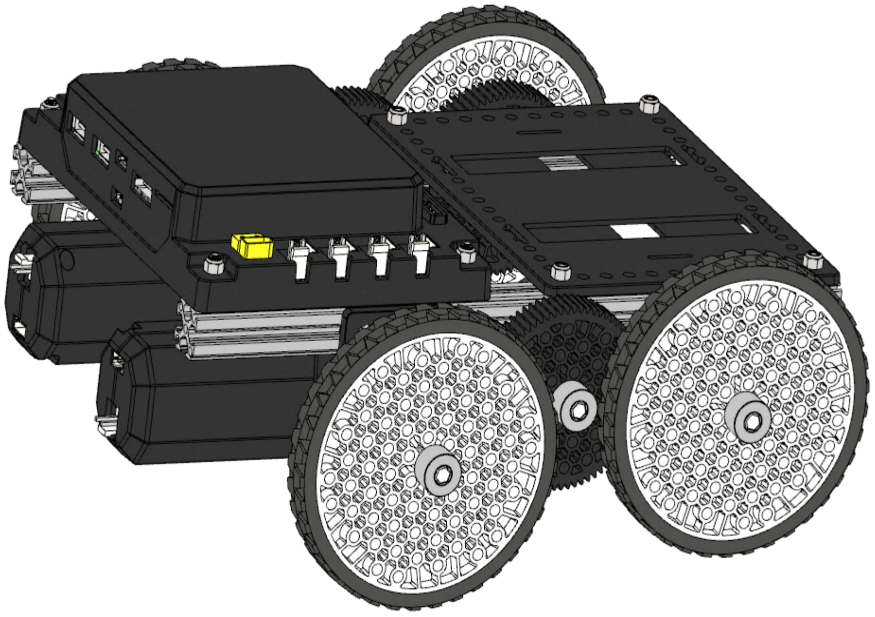


Figure 1: Completed miniBot Example

1.3 Additional Resources

This document only provides a very basic introduction to Android programming and how it relates to the FIRST Global Robot Controller platform. Participants are encouraged to learn more about Java and Android development from other resources.

The official Android Developer's website:

http://developer.android.com/index.html

The Android Developer's website has a series of Android development tutorials (including video-based interactive training):

http://developer.android.com/training/index.html

There is an excellent free tutorial from Oracle that teaches basic and advanced Java programming:

http://docs.oracle.com/javase/tutorial/

2 Android Operating System Basics

2.1 What is Andriod?

The FIRST Global Platform uses an Android tablet as the Driver station controller and a REV Robotics Control Hub as the robot brain. Android is the *operating system(OS)* that runs on both of these devices. Similar to a laptop that has Microsoft Windows or MacOS as its operating system, a tablet has its own operating system that manages the device's hardware and software components.

Google produces, develops, and maintains the Android code. This source code is available to the public under an open source license. Google provides free developer tools that can be used to write applications or "apps" for the Android OS.

2.2 Andriod Studio

The official development tool for Android development is known as *Android Studio*. The Driver Station and Control Hub run special apps for the FIRST Global competition. These apps are created using Android Studio.

Android Studio is known as an Integrated Development Environment (IDE). Android Studio is a software package that you install onto a computer or laptop. It has a suite of tools, such as a text editor, debugger, and other tools to help author, build and install apps for the Android operating system.

App development might seem intimidating at first. However, for the FIRST Global robots, the process has been simplified. The FIRST Global SDK includes a framework that makes it easier for a novice to program their robot. This framework takes care of much of the more complex programming tasks. The student or mentor can focus on programming the robot behavior and not have to worry about developing the framework of the Android app.

2.3 Java

Java is a popular text-based, object-oriented programming language. Android apps are written using the Java language. The programs that we will be using in this tutorial require a basic knowledge of Java. Unfortunately, the scope of this document does not allow for a detailed examination of the Java programming language. However, the Oracle Corporation maintains an excellent, free online Java tutorial:

http://docs.oracle.com/javase/tutorial/

Students are encouraged to review the online Java tutorial before they attempt to try the exercises in this training manual. It will be helpful if students have a basic knowledge of Java to complete the exercises in this manual. Students do not have to be Java experts, but should understand the basic syntax of java and be familiar with concepts such as classes, members, methods and program structure.

3 Installing Android Studio

***Note:** This training manual contains instructions on how to install the Android Studio software onto your PC. This information is provided to help you with the installation of the software, however, the screen shots and links in this document might be out of date.*

Android Studio Installation Procedure

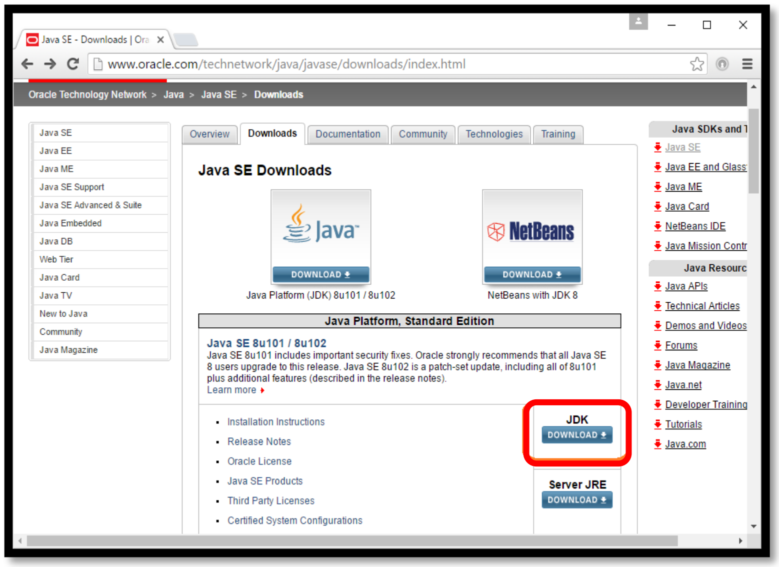
1. Before continuing you should first check the list of system requirements on the Android developer's website to verify that your system satisfies the list of minimum requirements:

   http://developer.android.com/sdk/index.html#Requirements

You will need to download and install the Java Development Kit (JDK) onto your computer, before installing the Android Studio software.

The JDK software can be downloaded from the Oracle Java Standard Edition web page:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Click on the Download button located below the text "JDK" to jump to the JDK download page.

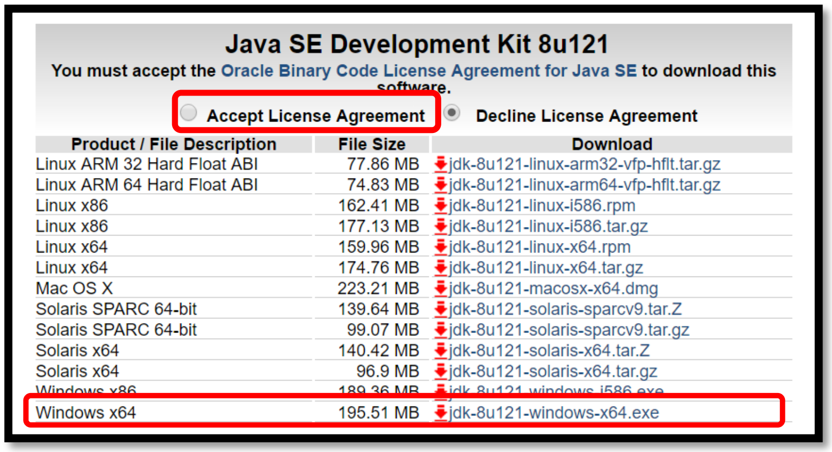


Java SE 8u101 includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Java SE 8u102 is a patch-set update, including all of 8u101 plus additional features (described in the release notes).

3. On the JDK download page, find the correct download package for your computer (Windows, Linux or Mac and x86 or x64).

   Accept the license agreement, then click on the appropriate download link for the desired package.

   Note that the JDK download page might look slightly different from the one shown here.



Java SE Development Kit 8u121

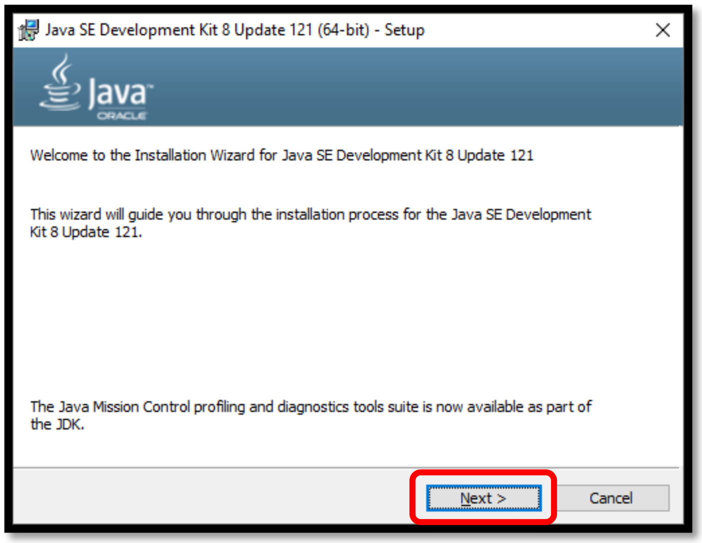
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ● Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.86 MB	jdk-8u121-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u121-linux-arm64-vfp-hflt.tar.gz
Linux x86	162.41 MB	jdk-8u121-linux-i586.rpm
Linux x86	177.13 MB	jdk-8u121-linux-i586.tar.gz
Linux x64	159.96 MB	jdk-8u121-linux-x64.rpm
Linux x64	174.76 MB	jdk-8u121-linux-x64.tar.gz
Mac OS X	223.21 MB	jdk-8u121-macosx-x64.dmg
Solaris SPARC 64-bit	139.64 MB	jdk-8u121-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.07 MB	jdk-8u121-solaris-sparcv9.tar.gz
Solaris x64	140.42 MB	jdk-8u121-solaris-x64.tar.Z
Solaris x64	96.9 MB	jdk-8u121-solaris-x64.tar.gz
Windows x86	189.36 MB	jdk-8u121-windows-i586.exe
Windows x64	195.51 MB	jdk-8u121-windows-x64.exe

4. Once you've downloaded the appropriate JDK package, run the package and follow the on-screen instructions to install the JDK software

   The default options are okay on most computers.



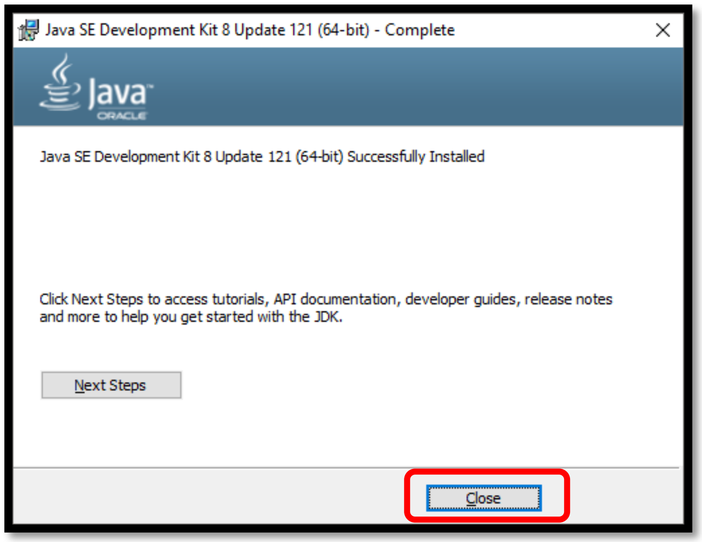
Java SE Development Kit 8 Update 121 (64-bit) - Setup

Welcome to the Installation Wizard for Java SE Development Kit 8 Update 121

This wizard will guide you through the installation process for the Java SE Development Kit 8 Update 121.

The Java Mission Control profiling and diagnostics tools suite is now available as part of the JDK.

5. After the install has completed you should get a message that it was successful. Select Close.



Java SE Development Kit 8 Update 121 (64-bit) - Complete

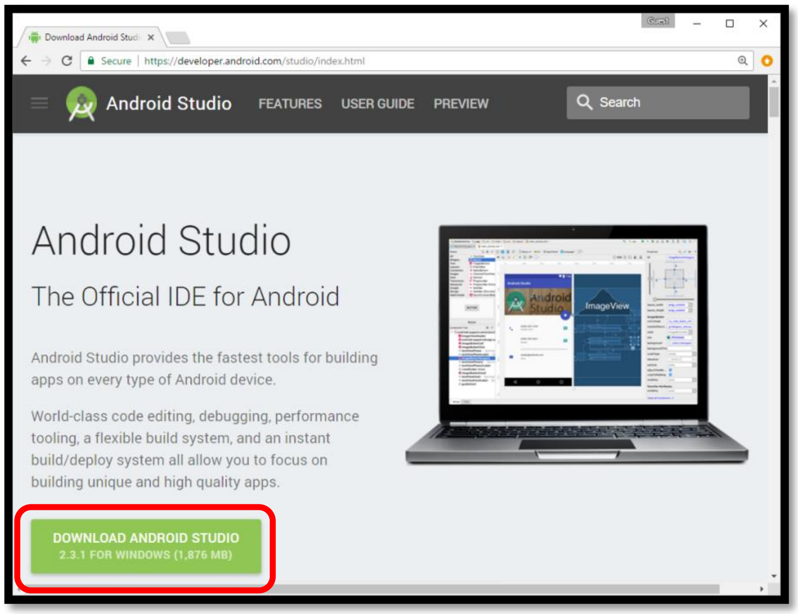
Java SE Development Kit 8 Update 121 (64-bit) Successfully Installed

Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

Once you have successfully installed the JDK software, you can go to the Android developer's website to download and install Android Studio.

https://developer.android.com/studio/index.html

6. Click on the green "DOWNLOAD ANDROID STUDIO" button to start the download process. Accept the default settings throughout the install process.



Android Studio

The Official IDE for Android

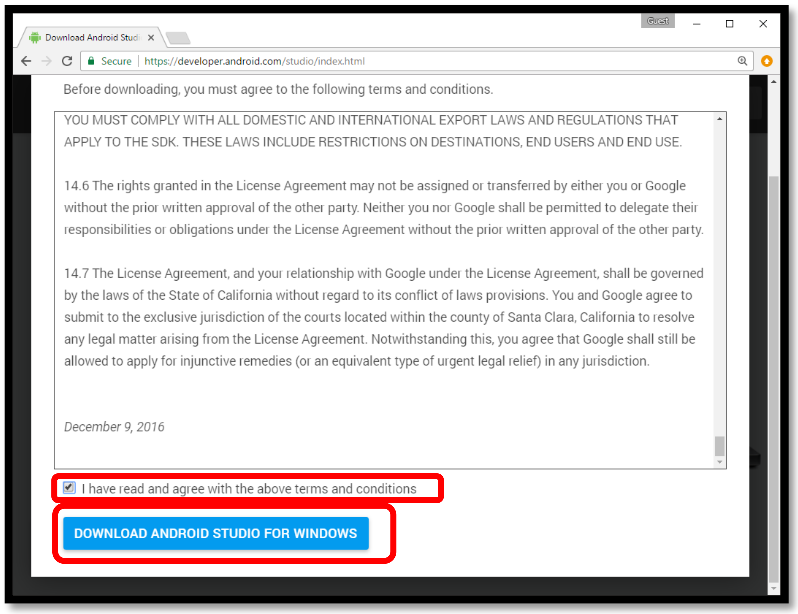
Android Studio provides the fastest tools for building apps on every type of Android device.

World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.

7. After clicking the download button, a pop-up will appear asking you to agree to the terms and conditions.

Read the terms and check the accept terms box to continue.

Click the Download Android Studio Button. Your page will redirect to provide installation instructions.


Download Android Studi
Secure | https://developer.android.com/studio/index.html
DOWNLOAD ANDROID STUDIO FOR WINDOWS

Before downloading, you must agree to the following terms and conditions.

YOU MUST COMPLY WITH ALL DOMESTIC AND INTERNATIONAL EXPORT LAWS AND REGULATIONS THAT APPLY TO THE SDK. THESE LAWS INCLUDE RESTRICTIONS ON DESTINATIONS, END USERS AND END USE.

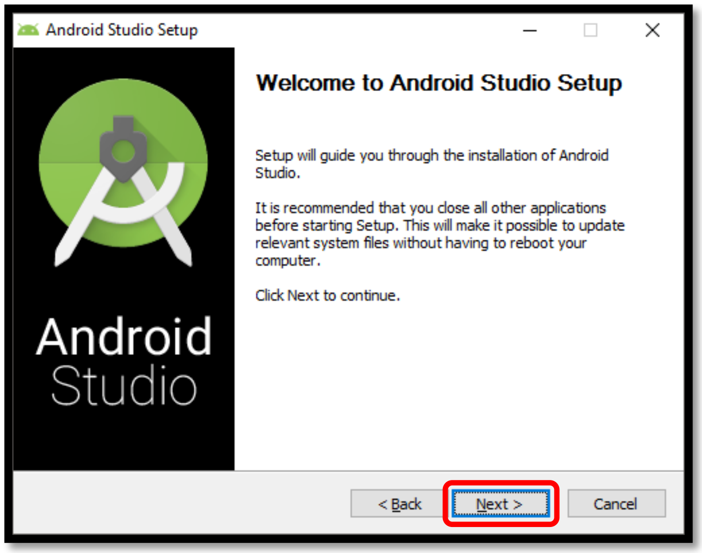
14.6 The rights granted in the License Agreement may not be assigned or transferred by either you or Google without the prior written approval of the other party. Neither you nor Google shall be permitted to delegate their responsibilities or obligations under the License Agreement without the prior written approval of the other party.

14.7 The License Agreement, and your relationship with Google under the License Agreement, shall be governed by the laws of the State of California without regard to its conflict of laws provisions. You and Google agree to submit to the exclusive jurisdiction of the courts located within the county of Santa Clara, California to resolve any legal matter arising from the License Agreement. Notwithstanding this, you agree that Google shall still be allowed to apply for injunctive remedies (or an equivalent type of urgent legal relief) in any jurisdiction.

December 9, 2016

[✓] I have read and agree with the above terms and conditions

8. Once the setup package has downloaded, launch the application and follow instructions to install Android Studio.



Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

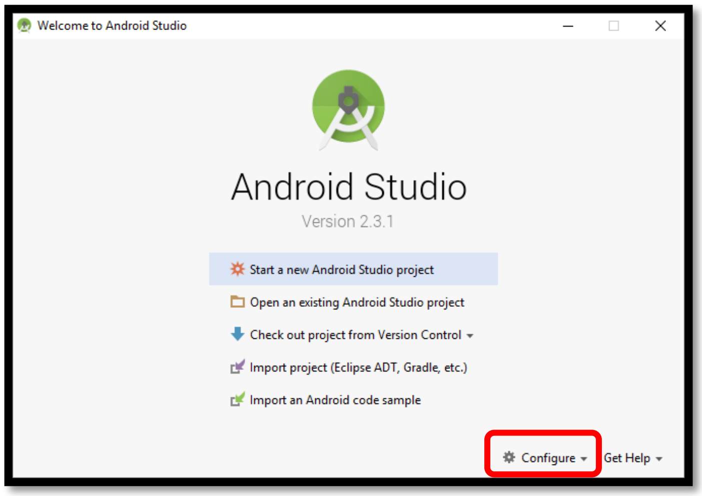
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

9. Once Android Studio has installed you will need to install the Android Marshmallow SDK.

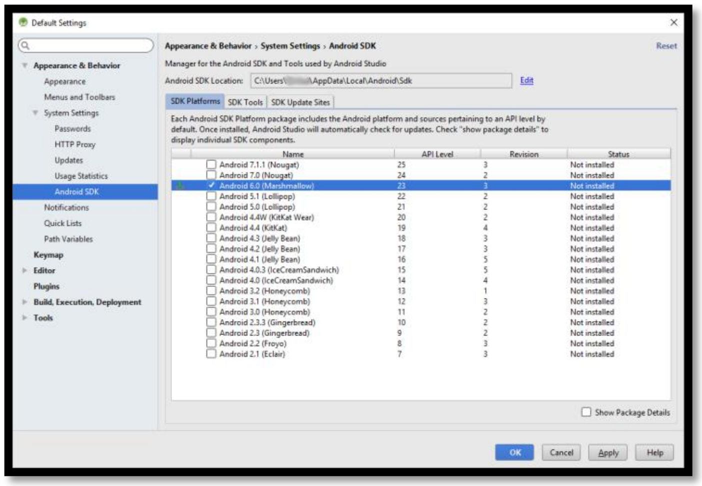
   First run Android Studio. If the "Welcome to Android Studio" dialog is displayed, drop down the configure menu and select SDK Manager

   Otherwise, select the SDK Manager from the menu:
   `Tools > Android > SDK Manager`


Welcome to Android Studio
Android Studio
Version 2.3.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

10. In the SDK Manager window, select "Android 6.0 (Marshmallow)" API level 23, Then click the "Apply" button.

    Follow the prompts to install the updates



Appearance & Behavior › System Settings › Android SDK

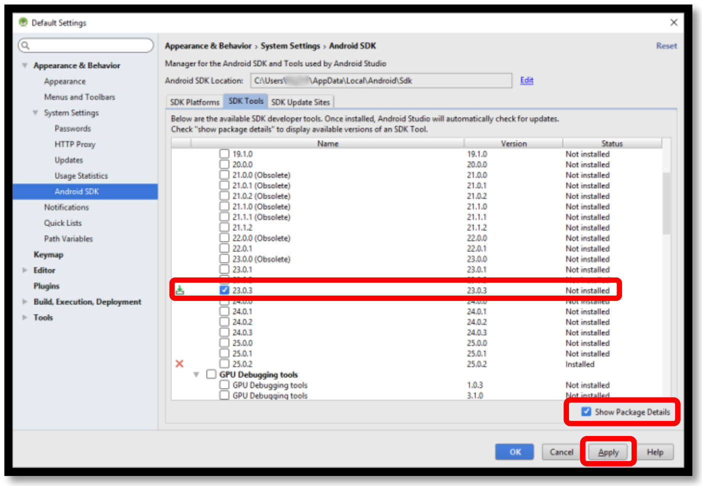
Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\[redacted]\AppData\Local\Android\Sdk

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
☐ Android 7.1.1 (Nougat)	25	3	Not installed
☐ Android 7.0 (Nougat)	24	2	Not installed
✓ Android 6.0 (Marshmallow)	23	3	Not installed
☐ Android 5.1 (Lollipop)	22	2	Not installed
☐ Android 5.0 (Lollipop)	21	2	Not installed
☐ Android 4.4W (KitKat Wear)	20	2	Not installed
☐ Android 4.4 (KitKat)	19	4	Not installed
☐ Android 4.3 (Jelly Bean)	18	3	Not installed
☐ Android 4.2 (Jelly Bean)	17	3	Not installed
☐ Android 4.1 (Jelly Bean)	16	5	Not installed
☐ Android 4.0.3 (IceCreamSandwich)	15	5	Not installed
☐ Android 4.0 (IceCreamSandwich)	14	4	Not installed
☐ Android 3.2 (Honeycomb)	13	1	Not installed
☐ Android 3.1 (Honeycomb)	12	3	Not installed
☐ Android 3.0 (Honeycomb)	11	2	Not installed
☐ Android 2.3.3 (Gingerbread)	10	2	Not installed
☐ Android 2.3 (Gingerbread)	9	2	Not installed
☐ Android 2.2 (Froyo)	8	3	Not installed
☐ Android 2.1 (Eclair)	7	3	Not installed

11. When this process completes, select the "SDK Tools" tab, and click the "Show Package Details" checkbox at the bottom right of the SDK Manager. Under the Android SDK Build-Tools section, select version 23.0.3 and click "Apply" again and allow the installation to complete



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\[redacted]\AppData\Local\Android\Sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
☐ 19.1.0	19.1.0	Not installed
☐ 20.0.0	20.0.0	Not installed
☐ 21.0.0 (Obsolete)	21.0.0	Not installed
☐ 21.0.1 (Obsolete)	21.0.1	Not installed
☐ 21.0.2 (Obsolete)	21.0.2	Not installed
☐ 21.1.0 (Obsolete)	21.1.0	Not installed
☐ 21.1.1 (Obsolete)	21.1.1	Not installed
☐ 21.1.2	21.1.2	Not installed
☐ 22.0.0 (Obsolete)	22.0.0	Not installed
☐ 22.0.1	22.0.1	Not installed
☐ 23.0.0 (Obsolete)	23.0.0	Not installed
☐ 23.0.1	23.0.1	Not installed
[illegible]	[illegible]	[illegible]
✓ 23.0.3	23.0.3	Not installed
[illegible]	[illegible]	[illegible]
☐ 24.0.1	24.0.1	Not installed
☐ 24.0.2	24.0.2	Not installed
☐ 24.0.3	24.0.3	Not installed
☐ 25.0.0	25.0.0	Not installed
☐ 25.0.1	25.0.1	Not installed
☐ 25.0.2	25.0.2	Installed
☐ **GPU Debugging tools**		
☐ GPU Debugging tools	1.0.3	Not installed
☐ GPU Debugging tools	3.1.0	Not installed

Click OK to close the settings window and close Android Studio. Android Studio is now installed and set-up.

4 Installing the FIRST Global SDK

After successfully installing Android Studio in Section 3 above, the next task is to download and import the FIRST Global Software Development Kit (SDK). The FIRST Global SDK is available as an Android Studio project folder. When you download this folder to your computer and then import it into the Android Studio IDE, you can use this project to customize, build and install the FIRST Global Robot Controller app.

An *operational mode* (aka, "Op Mode") is a code module that contains instructions that customize the behavior of a robot. Teams can use the FIRST Global SDK to create their own Op Modes to program their robots to compete in FIRST Global matches.

The FIRST Global SDK includes the libraries that are needed to communicate with the robot hardware, plus it has a host of sample Op Modes than can be copied and used to customize a robot's behavior. The SDK also includes the source code and resource files that define the look and behavior of the Robot Controller app.

The FIRST Global SDK can be downloaded from a GitHub repository. GitHub is a web-based version control company that lets individuals and organizations host content online. In order to access the FIRST Global software, you will need to have a GitHub account.

Advanced GitHub Users: This document assumes that the user is a novice with respect to using GitHub and the *git* version control software. If you are a GitHub power user, you can use *git* to create a local copy of the ftc_app repository as you normally would.

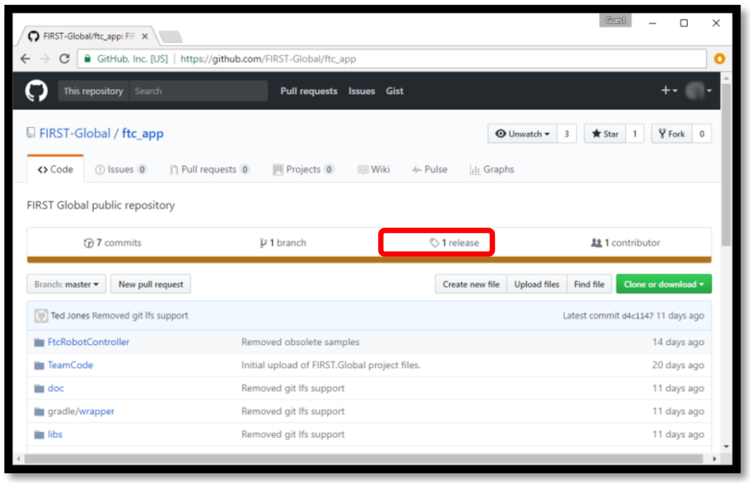
Installing the FIRST Global SDK from GitHub

1. Create a GitHub account if you don't already have one: https://github.com/

   Login and go to the FIRST Global public repository: https://github.com/FIRST-Global/ftc_app

2. From the main repository web page, click on the "releases" link to jump to the **Releases** page for the repository.

   The **Releases** page should list the available software releases for the repository. The latest release should be displayed near the top of the page.



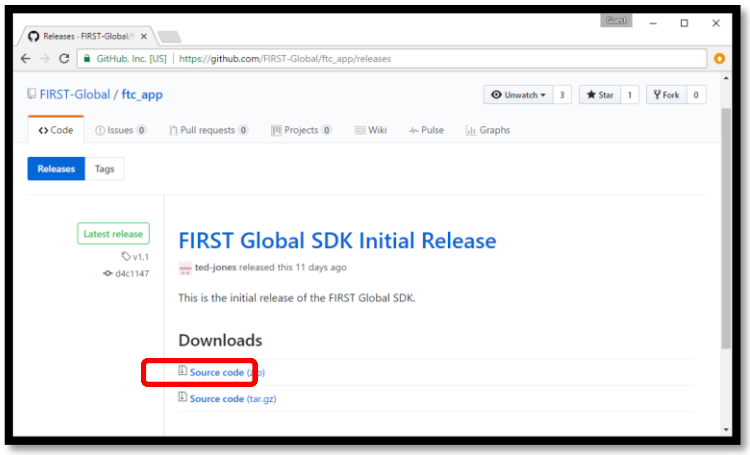
FIRST Global public repository

Ted Jones Removed git lfs support — Latest commit d4c1147 11 days ago

FtcRobotController	Removed obsolete samples	14 days ago
TeamCode	Initial upload of FIRST.Global project files.	20 days ago
doc	Removed git lfs support	11 days ago
gradle/wrapper	Removed git lfs support	11 days ago
libs	Removed git lfs support	11 days ago

3. Each software release should include a **Downloads** section that you can use to download the software that you will need to program your robot.

   Click `Source Code(zip)` link to download the compressed Android Studio Project



FIRST Global SDK Initial Release

ted-jones released this 11 days ago

This is the initial release of the FIRST Global SDK.

Downloads

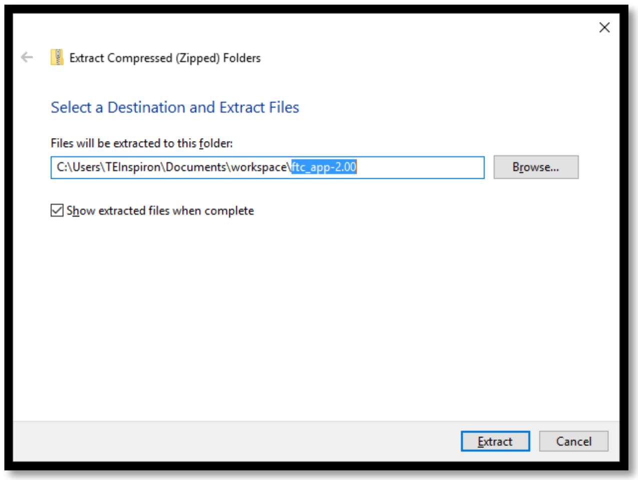
Source code (zip)

Source code (tar.gz)

4. Find the zip file you just downloaded and unzip the contents.

   You may want to move the file to a more convenient location before unzipping.

   For Window's users, right click on the file and select "extract all" from the menu. Windows should prompt you to select a destination for the extracted project folder.

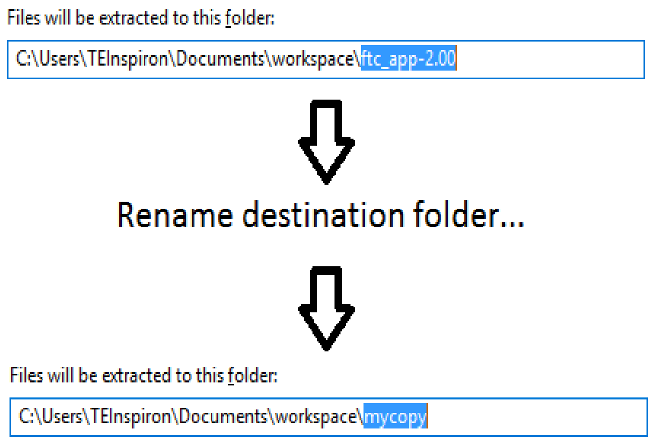


Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

Files will be extracted to this folder:

C:\Users\TEInspiron\Documents\workspace\ftc_app-2.00



Files will be extracted to this folder:

C:\Users\TEInspiron\Documents\workspace\ftc_app-2.00

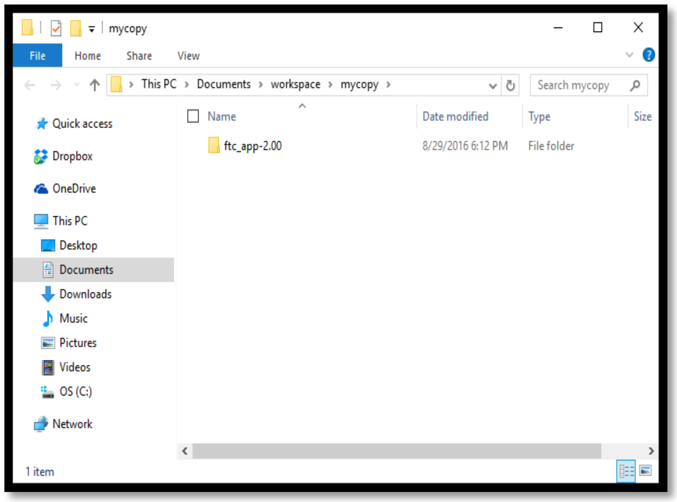
Rename destination folder...

Files will be extracted to this folder:

C:\Users\TEInspiron\Documents\workspace\mycopy

5. After the extraction process is complete, verify that the project folder was successfully extracted to its target destination.

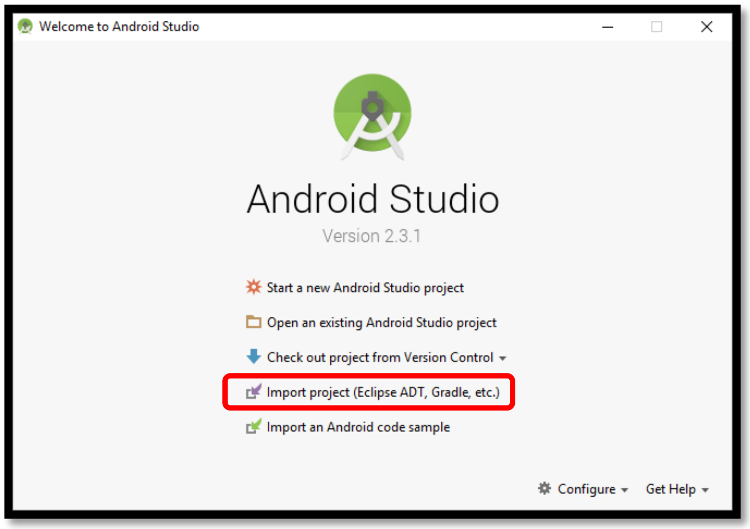
   Remember this file location. You will need it later.



Name	Date modified	Type	Size
ftc_app-2.00	8/29/2016 6:12 PM	File folder	

Now that we have successfully extracted the contents of the archived file, you are ready to import the FIRST Global project into Android Studio.

6. launch the Android Studio software on your computer. On the main Android Studio Welcome screen, select the option to "Import project (Eclipse, ADT, Gradle, etc.)" to begin the import process.

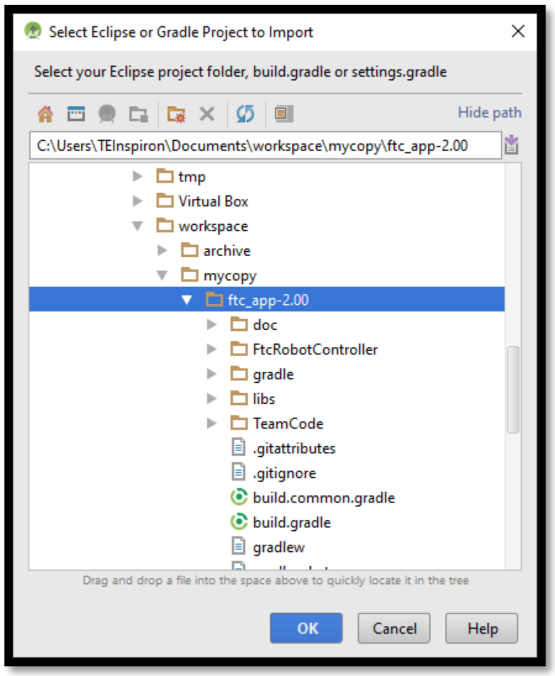

Welcome to Android Studio
Android Studio
Version 2.3.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

7. Android Studio will ask you to select the project folder that you would like to import.

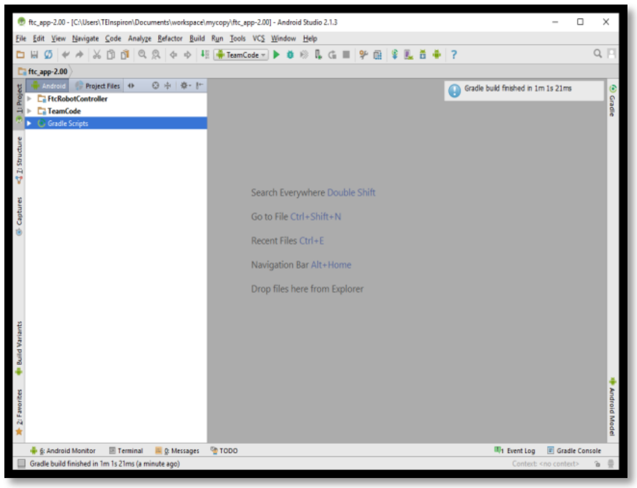
   Use the file browser in the pop up dialog box to locate and then select the folder that you extracted step 5.

   Make sure you select the extracted project folder (and not the .ZIP file which might have a similar name to the extracted folder).

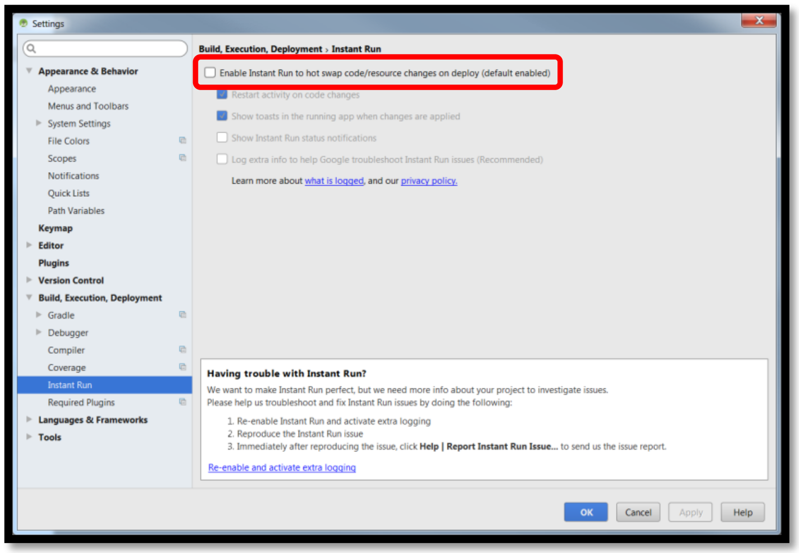
   Hit the "OK" button to import the selected project into Android Studio


Select Eclipse or Gradle Project to Import
Select your Eclipse project folder, build.gradle or settings.gradle
Hide path
C:\Users\TEInspiron\Documents\workspace\mycopy\ftc_app-2.00
tmp
Virtual Box
workspace
archive
mycopy
ftc_app-2.00
doc
FtcRobotController
gradle
libs
TeamCode
.gitattributes
.gitignore
build.common.gradle
build.gradle
gradlew
Drag and drop a file into the space above to quickly locate it in the tree
OK
Cancel
Help

8. It might take Android Studio several minutes to import the project. Once the project has been successfully imported, the screen should look similar to this.


ftc_app-2.00 - [C:\Users\TEInspiron\Documents\workspace\mycopy\ftc_app-2.00] - Android Studio 2.1.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-2.00
Android
Project Files
FtcRobotController
TeamCode
Gradle Scripts
Gradle build finished in 1m 1s 21ms
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here from Explorer
1: Project
7: Structure
Captures
Build Variants
2: Favorites
Gradle
Android Model
6: Android Monitor
Terminal
0: Messages
TODO
Event Log
Gradle Console
Gradle build finished in 1m 1s 21ms (a minute ago)
Context: <no context>

9. Select "Settings…" from the File menu. Then select "Instant Run" under the "Build, Execution, Deployment" dropdown and disable it by de-selecting the checkbox at the top of the dialog.



Build, Execution, Deployment › Instant Run

☐ Enable Instant Run to hot swap code/resource changes on deploy (default enabled)
☑ Restart activity on code changes
☑ Show toasts in the running app when changes are applied
☐ Show Instant Run status notifications
☐ Log extra info to help Google troubleshoot Instant Run issues (Recommended)

Learn more about what is logged, and our privacy policy.

Having trouble with Instant Run?
We want to make Instant Run perfect, but we need more info about your project to investigate issues.
Please help us troubleshoot and fix Instant Run issues by doing the following:
1. Re-enable Instant Run and activate extra logging
2. Reproduce the Instant Run issue
3. Immediately after reproducing the issue, click **Help | Report Instant Run Issue…** to send us the issue report.

Re-enable and activate extra logging

5 Using your Robot

Now that you've successfully configured your development environment on your computer, let's use the FIRST Global software to create an op mode that will allow us to learn the basics of how to get you working with the software.

Before you begin, you must create a configuration for your robot. The FIRST Global Control System Startup guide on the FIRST Global website (http://first.global/resources/technical-information/robot-kit-manuals) describes how to create a hardware configuration for your robot. The remainder of this document assumes that you have created a hardware configuration that consists of two motors named "`left_drive`" and "`right_drive`".

5.1 Creating an Op Mode

We will use Android Studio to create an Op Mode that can be used to drive the Practice-Bot robot. Let's take a look at the Android Studio user interface (Figure 2). In the left hand side, there should be a **Project** pane that shows the Project Browser. If you do not see the Project Browser, click on the word **Project** on the left hand border to expand the pane and display the Browser.

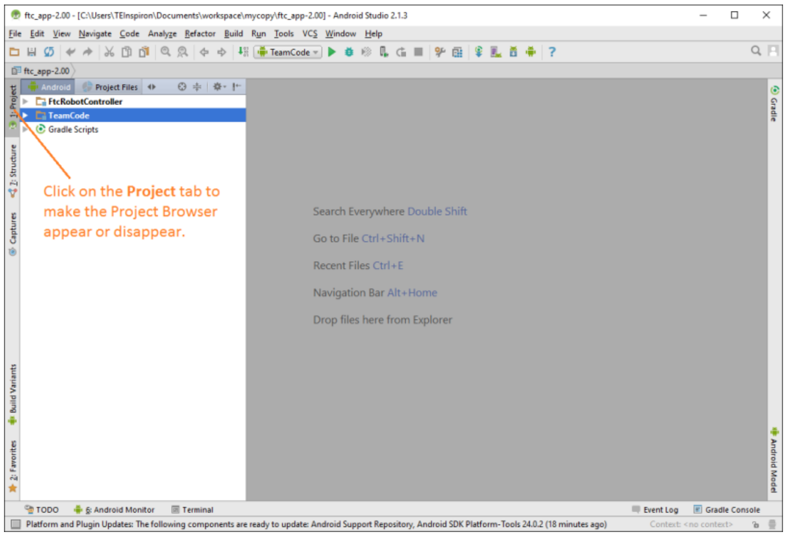
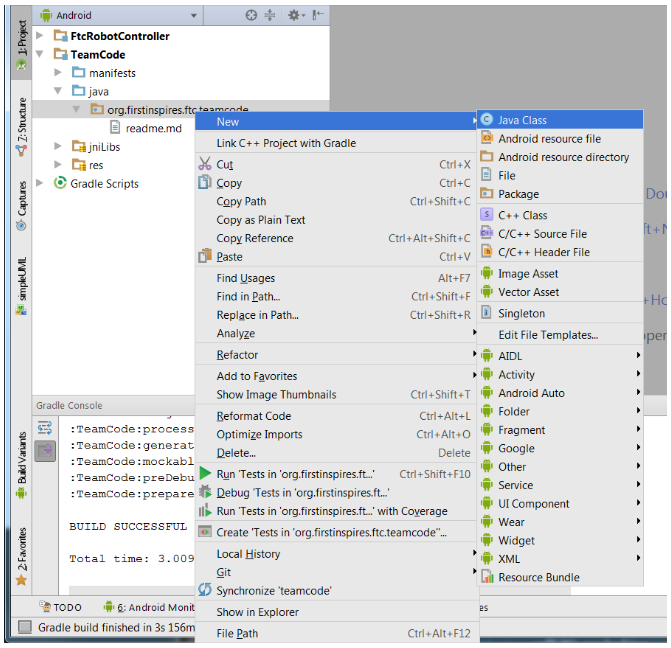

ftc_app-2.00 - [C:\Users\TEInspiron\Documents\workspace\mycopy\ftc_app-2.00] - Android Studio 2.1.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-2.00
Android
Project Files
1: Project
2: Structure
Captures
Build Variants
2: Favorites
FtcRobotController
TeamCode
Gradle Scripts
Click on the Project tab to make the Project Browser appear or disappear.
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here from Explorer
Gradle
Android Model
TODO
6: Android Monitor
Terminal
Event Log
Gradle Console
Platform and Plugin Updates: The following components are ready to update: Android Support Repository, Android SDK Platform-Tools 24.0.2 (18 minutes ago)
Context: <no context>

Figure 2: Click on the Project tab to make the Project Browser appear or disappear.

For the FIRST Global project, there should be two *packages* in the Project Browser. The **FtcRobotController** package contains example Op Modes for different types of robots. We are going to create a new Op Mode in the **TeamCode** package. The **TeamCode** package is where your team's custom op modes and other source files should reside.

Use the project browser to find the **TeamCode->java->org.firstinspires.ftc.teamcode** folder (Figure 3). Right click on that folder and select **New->Java Class**

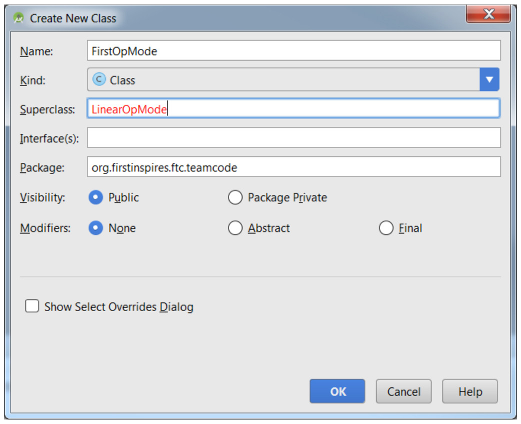


```
:TeamCode:process
:TeamCode:generat
:TeamCode:mockabl
:TeamCode:preDebu
:TeamCode:prepare

BUILD SUCCESSFUL

Total time: 3.009
```

Gradle build finished in 3s 156m

Figure 3: Create a new Java class in the org.firstinspires.ftc.teamcode folder.

In the Create New Class dialog, enter the name of the Op Mode as `FirstOpMode` and enter the Superclass as `LinearOpMode` (Figure 4).



Create New Class

Name: FirstOpMode

Kind: Class

Superclass: LinearOpMode

Interface(s):

Package: org.firstinspires.ftc.teamcode

Visibility: [✓ Public] Package Private

Modifiers: [✓ None] Abstract Final

☐ Show Select Overrides Dialog

Figure 4: Enter the Op Mode name and Superclass

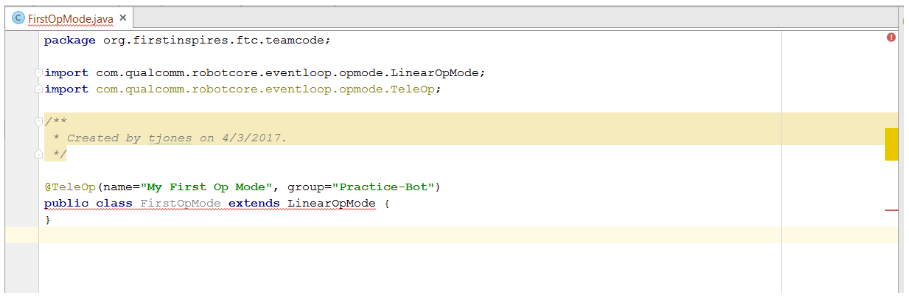
Select the "OK" button and Android Studio will create your class file. The `LinearOpMode` class will show up red since Android Studio does not know where to find this class. You must import the file containing this class definition. Android Studio created an import statement in the file "`import LinearOpMode;`". This should be changed to:

```
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
```

You will also need to import `com.qualcomm.robotcore.eventloop.opmode.TeleOp` to obtain the TeleOp Java annotation which will tell the robot controller that this Op Mode should be categorized as a TeleOp or manually controlled Op Mode. The TeleOp annotation should appear just before your class definition and contains a "name" and "group" parameter. The name parameter is the name that will be displayed on the driver station and the group parameter defines a group that this Op Mode will appear under. Enter this annotation as follows:

```
@TeleOp(name="My First Op Mode", group="Practice-Bot")
```

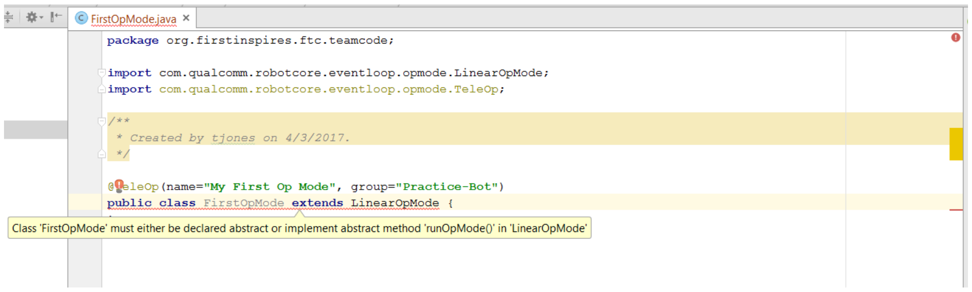
Your file should now appear as below (Figure 5).


FirstOpMode.java

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;

/**
 * Created by tjones on 4/3/2017.
 */

@TeleOp(name="My First Op Mode", group="Practice-Bot")
public class FirstOpMode extends LinearOpMode {
}
```

Figure 5 – Op Mode

Notice that Android Studio underlined your class definition in red. It is notifying you of an error in the source code. If you let the cursor hover over this line android studio will tell you what the error is (Figure 6).

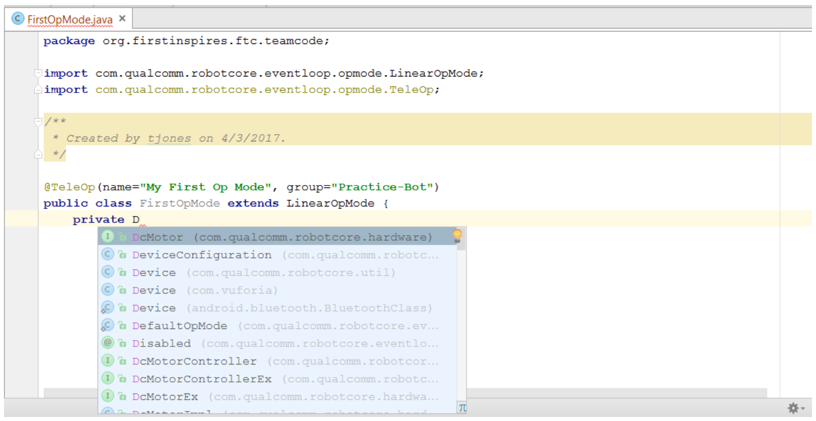

FirstOpMode.java

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;

/**
 * Created by tjones on 4/3/2017.
 */

@TeleOp(name="My First Op Mode", group="Practice-Bot")
public class FirstOpMode extends LinearOpMode {
```

Class 'FirstOpMode' must either be declared abstract or implement abstract method 'runOpMode()' in 'LinearOpMode'

Figure 6: The copied file should now appear in the destination folder.

LinearOpMode is an abstract class that declares a method called '`runOpMode()`' that must be implemented by your OpMode class. First, we will declare some class member variables to represent the hardware on our robot. The Practice-bot has two motors so we will declare one member variable to represent each motor, a `leftMotor` variable and a `rightMotor` variable both of type `DcMotor`. If we start typing '`private D`' Android Studio will make suggestions for the complete class context (Figure 7).


FirstOpMode.java

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;

/**
 * Created by tjones on 4/3/2017.
 */

@TeleOp(name="My First Op Mode", group="Practice-Bot")
public class FirstOpMode extends LinearOpMode {
    private D
```

```
DcMotor (com.qualcomm.robotcore.hardware)
DeviceConfiguration (com.qualcomm.robotc...
Device (com.qualcomm.robotcore.util)
Device (com.vuforia)
Device (android.bluetooth.BluetoothClass)
DefaultOpMode (com.qualcomm.robotcore.ev...
Disabled (com.qualcomm.robotcore.eventlo...
DcMotorController (com.qualcomm.robotcor...
DcMotorControllerEx (com.qualcomm.robotc...
DcMotorEx (com.qualcomm.robotcore.hardwa...
```

Figure 7 – Android Studio will suggest a class completion.

DcMotor should appear in the suggestion list. Select this and Android Studio will complete the class name and automatically add the appropriate import statement to your file. Name the first motor leftMotor and then repeat to add the rightMotor definition. These two variables are declared private since they will only be accessed from within this class. The declarations should look like the following:

```
    private DcMotor leftMotor;
    private DcMotor rightMotor;
```

Your runOpMode() method will run continuously in a loop and perform whatever functionality you specify. As long as your Op Mode is doing something the Android operating system will try to keep your Op Mode active in the CPU. This will tie up one of the cores of the CPU with the Op Mode which may prevent other important processing from occurring.

It is advisable to periodically give up your CPU time to allow other tasks to run. For this purpose we will create a function waitForTick() which will call a sleep() method on the current thread and tell the operating system that it can swap your process out. First create a private member variable to keep track of elapsed time as follows:

```
private ElapsedTime period = new ElapsedTime();
```

You must also import com.qualcomm.robotcore.util.ElapsedTime for the ElapsedTime definition (or use the Android Studio auto-complete function as with the motor definitions).

Then enter the following private method into your op mode:

```
/***
 *
 * waitForTick implements a periodic delay. However, this acts like a metronome
 * with a regular periodic tick.  This is used to compensate for varying
 * processing times for each cycle. The function looks at the elapsed cycle time,
 * and sleeps for the remaining time interval.
 *
 * @param periodMs Length of wait cycle in mSec.
 */

private void waitForTick(long periodMs) throws java.lang.InterruptedException
 {
    long  remaining = periodMs - (long)period.milliseconds();
    // sleep for the remaining portion of the regular cycle period.
    if (remaining > 0) {
        Thread.sleep(remaining);
    }
    // Reset the cycle clock for the next pass.
    period.reset();
}
```

This method is declared private since it is only intended to be used from within this class. You will call this function on each iteration of the loop to tell the operating system that your process can be swapped out. The parameter passed to this function indicates that your process wants to run every periodMs milliseconds. This method will adjust the sleep time to compensate for processing time spent in your loop. In other words, if you want to run every 40 milliseconds and your processing time was 15 milliseconds this method will sleep for 25 milliseconds (40 – 15).

The java.lang.interrupted exception may be thrown from the Thread.sleep() method so we must either handle it or declare it as an exception that may be thrown from this method. In this case we will allow it to be thrown out of this method since it is telling us that our thread has been interrupted and should terminate. We will allow this to be handled from our runOpMode() method.

Now we can create the `runOpMode()` method. Since this method is overriding an inherited method we will use the Java `@override` annotation. This annotation is optional but will help the compiler inform you of potential errors such as return type or parameter mismatches or misspellings. Add the following code to your op mode (this will be explained in detail below):

```
@Override
public void runOpMode() {
    double left = 0.0;
    double right = 0.0;

    leftMotor = hardwareMap.dcMotor.get("left_drive");
    rightMotor = hardwareMap.dcMotor.get("right_drive");
    rightMotor.setDirection(DcMotorSimple.Direction.REVERSE);

    // Set all motors to zero power
    leftMotor.setPower(0);
    rightMotor.setPower(0);

    // Send telemetry message to signify robot waiting;
    telemetry.addData("Say", "Hello Driver");    //
    telemetry.update();

    // Wait for the game to start (driver presses PLAY)
    waitForStart();

    try {
       // run until the end of the match (driver presses STOP)
       while (opModeIsActive()) {
            // Run wheels in tank mode (note: The joystick goes negative when
            // pushed forwards, so negate it)
            left = -gamepad1.left_stick_y;
            right = -gamepad1.right_stick_y;
            leftMotor.setPower(left);
            rightMotor.setPower(right);

            // Send telemetry message to signify robot running;
            telemetry.addData("left", "%.2f", left);
            telemetry.addData("right", "%.2f", right);
            telemetry.update();

            // Pause for metronome tick.  40 mS each cycle = update 25 times
            // a second.
            waitForTick(40);
        }
    }
    catch (java.lang.InterruptedException exc) {
        return;
    }
    finally {
        leftMotor.setPower(0);
        rightMotor.setPower(0);
    }
}
```

This Op Mode retrieves the motor definitions from the `hardwareMap` member variable. This member variable is inherited from the `OpMode` class and contains all of the hardware items that were defined in your configuration. These hardware objects are retrieved using the names you provided when the robot was configured. There are separate device map lists in the `HardwareMap` for each type of device supported. In this case we are looking for motors so we use the `hardwareMap.dcMotor` list.

We set the direction of the right motor to reverse as explained in the Software Tool Overview video on the FIRST Global website (http://first.global/resources/technical-information/robot-kit-instructional-videos). Since we are referring to an enumeration (Direction) defined in the `DcMotorSimple` class we must also add the following import directive to imports at the top of our file:

```
        import com.qualcomm.robotcore.hardware.DcMotorSimple;
```

We then set the power for both motors to '`0`' to ensure that they are not moving and we send a telemetry message to the driver station to indicate that the robot is ready to go. The `waitForStart()` method is then called to wait until the driver presses the play button.

The next two statements establish a try block and the main Op Mode loop. The loop will continue to execute until the Op Mode is stopped by the press of the stop button or an exception being thrown. Since we know that the `java.lang.InterruptedException` may be thrown by the `waitForTick` method we must implement a try block and catch this exception. For reasons explained later it is good practice to use a try block in Java to protect resource allocation or global setting changes.

The next four lines of code retrieve the left and right joystick values from the controller. This Op Mode is assuming only one driver so we take the values from the `gamepad1` member of the `OpMode` class. There is also a `gamepad2` member for when two gamepads are used. When the joysticks are pressed up they return a negative value which is not intuitive so these values are negated to make them easier to deal with. We then set the left and right motor power to the value retrieved from the joysticks. The joysticks return a value in the range of -1.0 to 1.0 which is the same range the motor controller `setPower()` function is expecting so no further conversion is required.

We then send some data to the driver station through telemetry calls to display the current motor power on the driver station. The last part of the Op Mode loop is to call the `waitForTick()` method. We pass a period of 40 ms so that our loop will execute 25 times per second.

The catch block catches the `java.lang.InterruptedException` which tells us that our thread has been interrupted and should terminate. We simply return here as there is no other specific error handling to be done. Any other unexpected exception will be thrown from our Op Mode and caught at a higher level where a message will be displayed to the Driver Station.

The finally block allows us to do any clean up that may be needed. The Java language guarantees that any code in the finally block will be executed at the end of a try block regardless of how the try block was terminated (normally, through an exception, return, break, etc.) This gives us a good opportunity to release any allocated resources and restore global settings changed in our Op Mode. We set the power on both motors to 0 to ensure that our robot is at a complete stop when the Op Mode terminates.

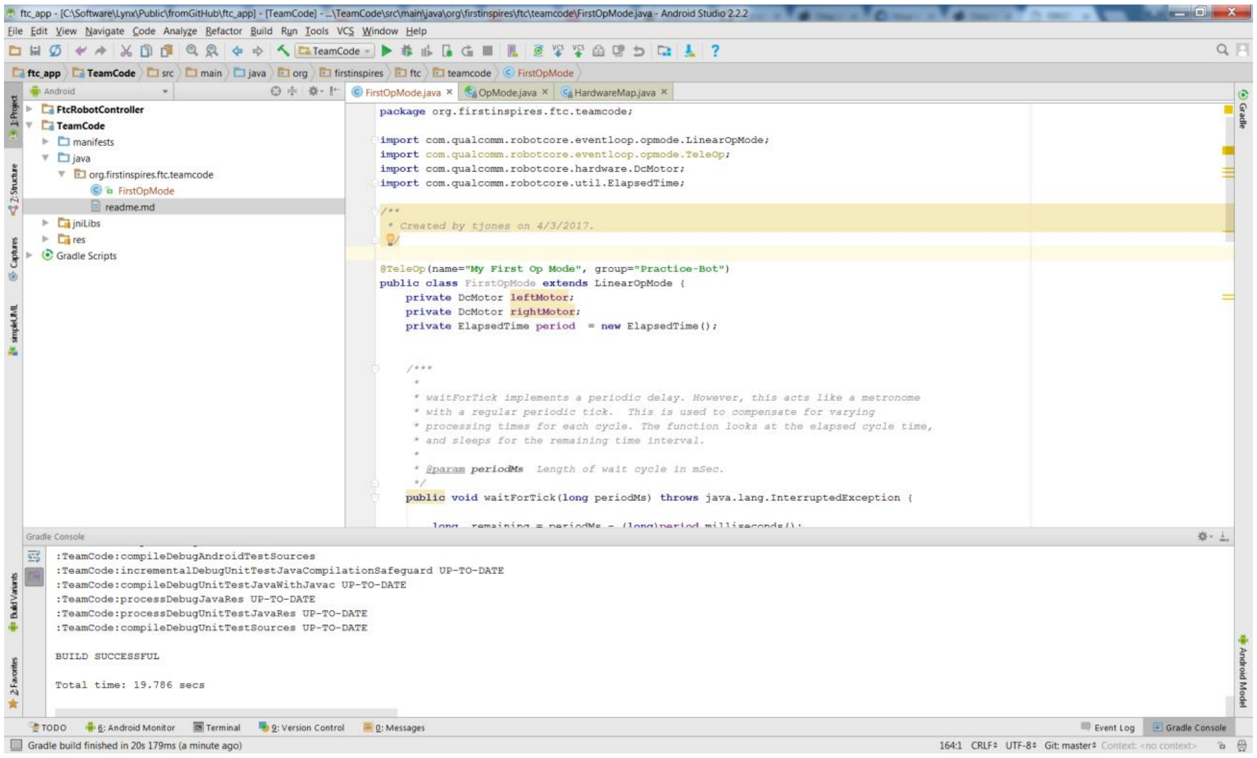
5.2 Building the Robot Controller App

We are now ready to build the Robot Controller App with our new Op Mode. The complete source code should look something like the following:

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;
import com.qualcomm.robotcore.util.ElapsedTime;

/**
 * Created by tjones on 4/3/2017.
```

```java
 */

@TeleOp(name="My First Op Mode", group="Practice-Bot")
public class FirstOpMode extends LinearOpMode {
    private DcMotor leftMotor;
    private DcMotor rightMotor;
    private ElapsedTime period  = new ElapsedTime();

    /***
     *
     * waitForTick implements a periodic delay. However, this acts like a metronome
     * with a regular periodic tick.  This is used to compensate for varying
     * processing times for each cycle. The function looks at the elapsed cycle time,
     * and sleeps for the remaining time interval.
     *
     * @param periodMs  Length of wait cycle in mSec.
     */
    private void waitForTick(long periodMs) throws java.lang.InterruptedException {

        long  remaining = periodMs - (long)period.milliseconds();

        // sleep for the remaining portion of the regular cycle period.
        if (remaining > 0) {
            Thread.sleep(remaining);
        }

        // Reset the cycle clock for the next pass.
        period.reset();
    }

    @Override
    public void runOpMode() {
        double left = 0.0;
        double right = 0.0;

        leftMotor = hardwareMap.dcMotor.get("left_drive");
        rightMotor = hardwareMap.dcMotor.get("right_drive");
        rightMotor.setDirection(DcMotorSimple.Direction.REVERSE);

        // Set all motors to zero power
        leftMotor.setPower(0);
        rightMotor.setPower(0);

        // Send telemetry message to signify robot waiting;
        telemetry.addData("Say", "Hello Driver");    //
        telemetry.update();

        // Wait for the game to start (driver presses PLAY)
        waitForStart();

        try {
            // run until the end of the match (driver presses STOP)
            while (opModeIsActive()) {
                // Run wheels in tank mode (note: The joystick goes negative when pushed forwards,
so negate it)
                left = -gamepad1.left_stick_y;
                right = -gamepad1.right_stick_y;
                leftMotor.setPower(left);
                rightMotor.setPower(right);

                // Send telemetry message to signify robot running;
                telemetry.addData("left", "%.2f", left);
                telemetry.addData("right", "%.2f", right);
                telemetry.update();

                // Pause for metronome tick.  40 mS each cycle = update 25 times a second.
                waitForTick(40);
            }
        }
        catch (java.lang.InterruptedException exc) {
            return;
        }
        finally {
            leftMotor.setPower(0);
            rightMotor.setPower(0);
        }
    }
}
```

To ensure that your op mode builds select **Build->Rebuild Project** in Android Studio. If you select the "Gradle Console" (on the bottom right of the Android Studio window) you should see something like the following:



```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.util.ElapsedTime;

/**
 * Created by tjones on 4/3/2017.
 */

@TeleOp(name="My First Op Mode", group="Practice-Bot")
public class FirstOpMode extends LinearOpMode {
    private DcMotor leftMotor;
    private DcMotor rightMotor;
    private ElapsedTime period  = new ElapsedTime();


    /***
     *
     * waitForTick implements a periodic delay. However, this acts like a metronome
     * with a regular periodic tick.  This is used to compensate for varying
     * processing times for each cycle. The function looks at the elapsed cycle time,
     * and sleeps for the remaining time interval.
     *
     * @param periodMs  Length of wait cycle in mSec.
     */
    public void waitForTick(long periodMs) throws java.lang.InterruptedException {
```

```
:TeamCode:compileDebugAndroidTestSources
:TeamCode:incrementalDebugUnitTestJavaCompilationSafeguard UP-TO-DATE
:TeamCode:compileDebugUnitTestJavaWithJavac UP-TO-DATE
:TeamCode:processDebugJavaRes UP-TO-DATE
:TeamCode:processDebugUnitTestJavaRes UP-TO-DATE
:TeamCode:compileDebugUnitTestSources UP-TO-DATE

BUILD SUCCESSFUL

Total time: 19.786 secs
```

Gradle build finished in 20s 179ms (a minute ago)

It is now time to connect your Control Hub to your computer. When you connect your Control Hub to the USB port of your computer ensure that you are connecting to the micro-USB on the top of the Control Hub, not the mini-USB on the bottom.

For Windows 7 you will need to install the ADB driver for the board. Follow the instructions on this website: https://discuss.96boards.org/t/step-by-step-instruction-to-install-adb-usb-driver-on-windows/777.

For Windows 10, you should not need to install a driver. If you have problems, see the troubleshooting section at the end of this document.

Once the driver is installed and the control hub is connected you can install the app. Select the green arrow (Run button) on the Android Studio toolbar. Android Studio will then prompt you to select a target device to install the Robot Controller app. Your screen might look something like the image shown below (Figure 8).

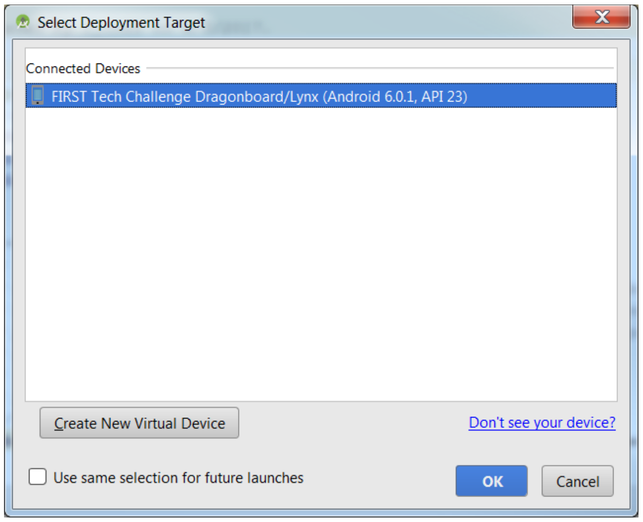

Select Deployment Target
Connected Devices
FIRST Tech Challenge Dragonboard/Lynx (Android 6.0.1, API 23)
Create New Virtual Device
Don't see your device?
Use same selection for future launches
OK
Cancel

Figure 8 - Android Studio might prompt you for the target device.

Make sure that you select the correct target device if more than one device is displayed. When you select the OK button Android Studio will first perform a build. You will then see a status bar at the bottom of Android Studio that says "Installing APK". Do not disconnect the Control Hub until this process finishes.

When the install is finished you can disconnect the Control Hub from the USB and start the Driver Station app on the (paired) tablet. Your Op Mode will now be selectable from the TeleOp modes menu on the Driver Station.

6 Troubleshooting

If you have problems building or installing the Robot Controller software Try the following

Troubleshooting

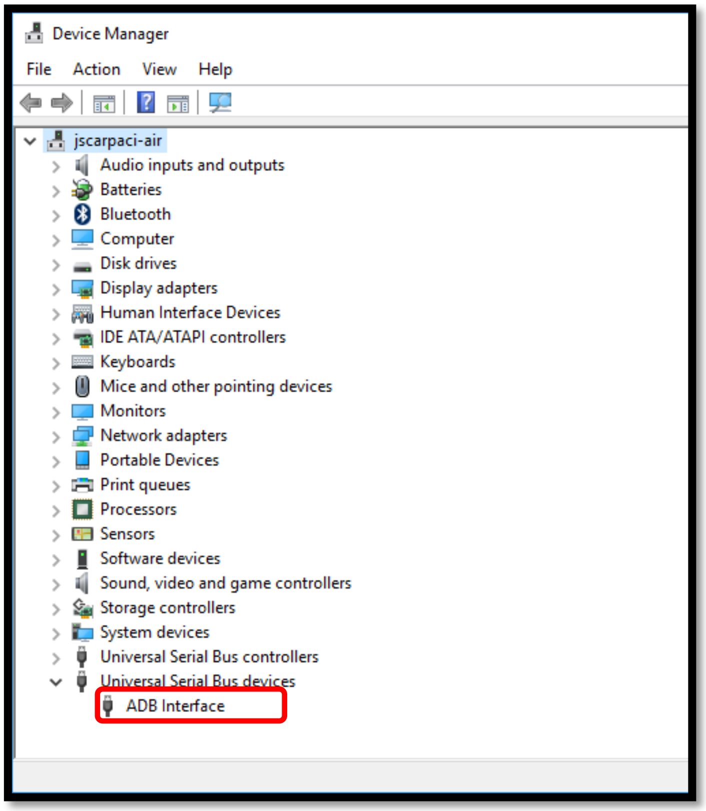
1. Make sure you have all of the latest updates to your operating system. This process is specific to the OS version, see the Windows documentation for installing updates.

2. **On Windows 10 you are not able to see the Control Hub through ADB**

   Open the Device manager.

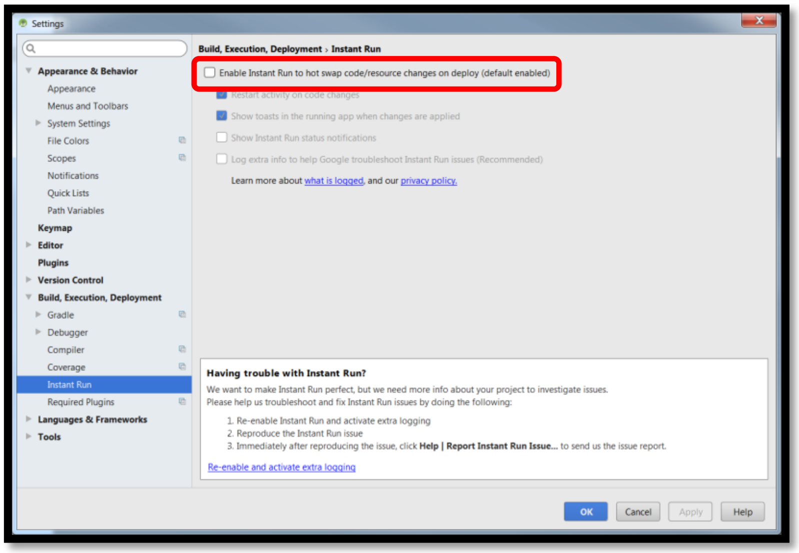
   If your device driver shows up as "ADB Interface" or "Android Device". Right click on it and select "Update Driver Software". Make sure you are connected to the internet so Windows can search for the driver online.

   Windows should find an appropriate driver. And it should show up as Dragonboard/Lynx or as Kedacom KDB Composite Interface.



3. **APK does not run and light stays blue.**

   Verify that "Instant Run is disabled" Select "Settings…" from the File menu. Then select "Instant Run" under the "Build, Execution, Deployment" dropdown and disable it by de-selecting the checkbox at the top of the dialog.



Build, Execution, Deployment › Instant Run

- [ ] Enable Instant Run to hot swap code/resource changes on deploy (default enabled)
- [✓] Restart activity on code changes
- [✓] Show toasts in the running app when changes are applied
- [ ] Show Instant Run status notifications
- [ ] Log extra info to help Google troubleshoot Instant Run issues (Recommended)

Learn more about what is logged, and our privacy policy.

Having trouble with Instant Run?

We want to make Instant Run perfect, but we need more info about your project to investigate issues.
Please help us troubleshoot and fix Instant Run issues by doing the following:

1. Re-enable Instant Run and activate extra logging
2. Reproduce the Instant Run issue
3. Immediately after reproducing the issue, click **Help | Report Instant Run Issue…** to send us the issue report.

Re-enable and activate extra logging