DRAFT COPY: CONTENTS SUBJECT TO CHANGE

FTC Training Manual

JAVA Programming for the Next Gen Controller

Thomas Eng

5/25/2015

This document contains training material to introduce students and mentors to the next gen FIRST Tech Challenge robot controller platform.

Contents

1 Introduction

The *FIRST* Tech Challenge (FTC) will be adopting a new controller for its robot competitions. This new platform uses Android devices that are powered by QualComm Snapdragon processors. This document contains information and exercises to teach students and mentors how to use and program this new system.

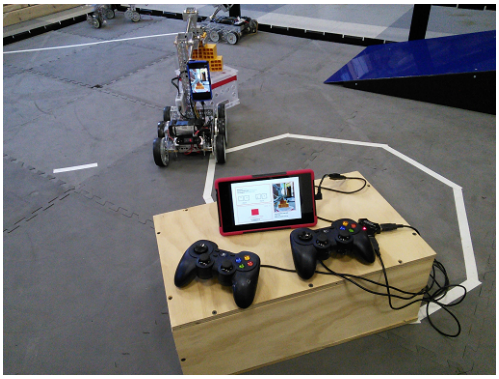

Figure 1 - FTC's new Android-based platform.[1]

2 Agenda

This training document is organized in following manner,

- Overview of the new platform
- Getting started with Android programming
- Using your Robot
- Operation Modes
- QualComm FTC Robot SDK
- Building the Robot Controller
- Writing your own Op Mode
- SDK documentation

3 Scope of Material

This document only provides a very basic introduction to Android programming and how it relates to the next gen FTC robot controller platform.

Participants are encouraged to learn more about Java and Android development for other resources, such as the official Android Developer's website:

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

[1] Note that the user interfaces that are pictured on the Android devices are simulated.

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/

4 Demo Robots

The training material in this document assumes that students will have access to a demo robot. The robot (aka, the "K9 bot") is a small-sized robot that is equipped with a DC motor controller, a servo controller, and two sensors. The DC motor controller controls two 12V DC motors. The servo controller controls two 180-degree servos. The K9 bots also are equipped with a Power Module device and a Legacy Module device. The motor and servo controllers, and the Power and Legacy Modules are made by a company called Modern Robotics.

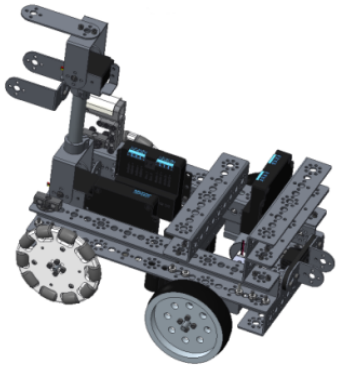

Figure 2 – This training material often makes references to a K9 bot (Matrix or Tetrix).

If you are interested in building your own version of the K9 bot, a rudimentary build guide is available and distributed with this training manual.

5 Overview of the Next Gen Platform

The next gen platform is a *point-to-point* system. This system has two main components, the *Driver Station* and the *Robot Controller*. The driver station communicates wirelessly with the robot controller.

Figure 3 - The new FTC platform is a point-to-point solution.

5.1 Driver Station

The driver station is the component that the teams use to provide input (from a pair of gamepads and/or a touch screen) to the Robot Controller. The driver station also displays messages (status information, motor speed, sensor data, etc.) from the Robot Controller.

The driver station consists of the following components,

1. ***Android Device*** – An Android device is used as the base of the driver station. This Android device communicates wirelessly with the robot controller. This device has its own internal battery.
2. ***USB Gamepad Controllers*** – The driver station uses a pair of USB-enabled gamepad controllers for driver input. For this tutorial, we will use a pair of Logitech F310 gamepads with the devices set in "X" mode (Xbox emulation mode, i.e., the switch on the bottom of the F310 should be set to the "X" position).
3. ***Non-Powered USB Hub*** – The driver station Android tablet only has a single USB port. A USB hub is used to connect both gamepads to the driver station tablet.
4. ***Micro USB Adapter*** – The driver station Android device has a type of connector port known as a *Micro USB Type B (female)* connector. An adaptor cable is needed to connect the USB hub to the Android device.
5. ***Driver Station Android App*** – The driver station needs a special Android app that receives input from the gamepads and communicates with the robot controller.

Figure 4 - Driver station hardware consists of a tablet, an adapter cable, a USB hub and two gamepads.

5.2 Robot Controller

The robot controller acts as the "brains" of the robot. It handles communications with the driver station. It processes sensor data and user commands and sends instructions to the motor and servo controllers to make the robot move.

For this tutorial, the K9 bot is equipped with the following components,

1. ***Android Device*** – Preferably a phone-sized (smaller) Android device is used as the robot controller. This device will communicate wirelessly to the driver station. This device will also communicate, through a USB cable, to the onboard robot hardware. This device has its own internal battery.
2. ***Robot Controller App*** – The Android device runs a special robot controller app that communicates with the driver station and sends/receives messages to the onboard robot hardware.
3. ***12V Rechargeable Battery*** – The battery provides power to a USB hub, which in turn provides power to other devices (such as the Legacy Module) that are connected to the USB hub. The battery also provides power to drive the DC motors and servos on the robot.
4. ***Power Module*** – The Power Module is an electronic device that has a built-in USB hub which connects the Android device to one or more USB-enabled modules (such as the Legacy Module). The Power Module draws power from the 12V battery and feeds it to the USB devices through the USB cables. The Power Module also feeds power directly to the DC motor and servo controllers.
5. ***Micro USB Adapter*** – The robot controller Android device has a type of connector port known as a *Micro USB Type B (female)* connector. An adapter cable is needed to connect the Android Device to the built-in USB.
6. ***USB Cables*** – The USB-connected modules (such as the Power Module and Legacy Module) for the next gen platform use USB Mini style cables to connect with each other and with the Android device. Note that an adapter cable is needed to connect the USB Mini style cable to the Android's Micro USB Type B port.
7. ***Legacy Module*** – This module is a special electronic device that acts as a bridge between the new Android robot controller and the older LEGO NXT-compatible devices (motor and servo

controllers, sensors, etc.). The Legacy Module communicates with the Android device through a USB connection. It receives commands and sends data from/to the Android device. It also communicates, through six NXT-style ports to existing NXT-compatible hardware (motor controller, servo controller, LEGO approved sensors, etc.). The Legacy Module draws power through the USB connection from the Power Module.

8. ***DC Motor Controller with Motors*** – The K9 bot has a single DC motor controller. The USB-enabled DC Motor Controller communicates with the Android device through a USB connection. The USB-enabled motor controller also draws 12V power from the Power Module to power the DC motors.
9. ***Servo Controller with Servos*** – The K9 bot has a single servo controller. The USB-enabled Servo Controller communicates with the Android device through a USB connection. The USB-enabled servo controller also draws 12V power from the Power Module to power the servos.
10. ***Hitechnic IR Seeker V2 Sensor*** – The K9 bot has an NXT-compatible IR seeker sensor. The sensor is connected to the Legacy Module. The Android device communicates through the Legacy Module to the sensor. The sensor draws power from the Legacy Module.
11. ***LEGO Light Sensor*** – The K9 bot has an NXT-compatible IR seeker sensor. The sensor is connected to the Legacy Module. The Android device communicates through the Legacy Module to the sensor. The sensor draws power from the Legacy Module.
12. ***NXT Cables*** – The K9 bot uses standard LEGO NXT-compatible cables to connect the Legacy Module with the sensors.

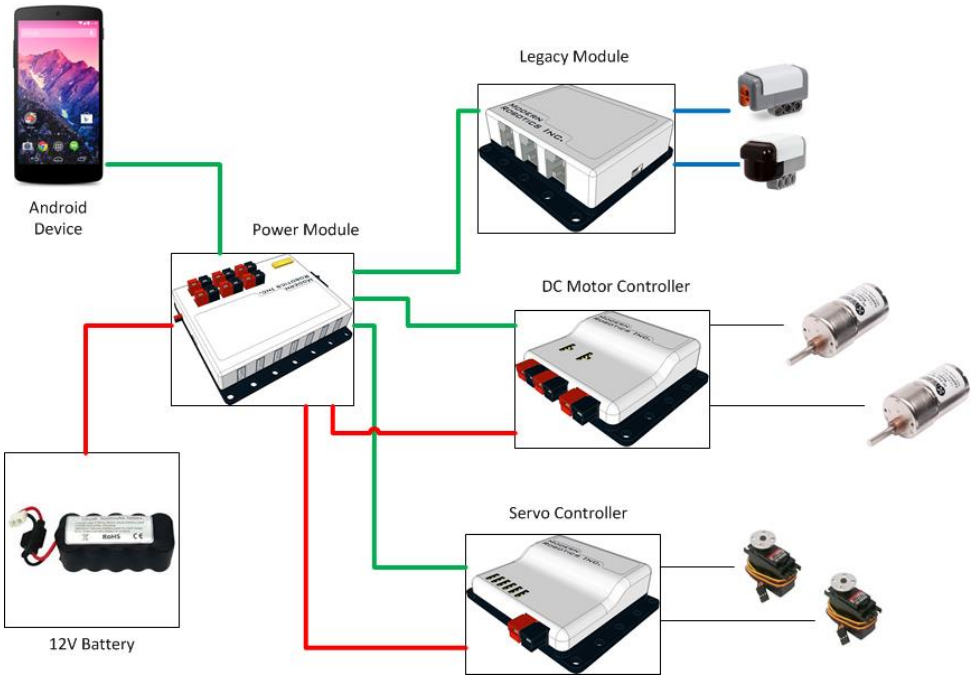

Figure 5 - K9 bot hardware with USB-enabled motor/servo controllers (cables and adapter not shown in detail).

5.3 A Note about USB Ports & Cables

The next gen platform uses USB (which stands for Universal Serial Bus) to connect the Android devices to the robot and game controller hardware. The USB cables allow the devices to communicate at a very high speed. The USB cables also provide power to the devices that are "downstream" to the Power Module.

5.3.1 USB Host & Client Modes

A USB-enabled device such as a smartphone or tablet can act as a USB host or as a USB client device. A USB host is the device that initiates all communications on the bus. The USB client responds to communications from the host. The Android devices that are used for the next gen platform have the ability to act as either a USB host or a USB client. When you plug the Android device into a laptop the Android device acts as a USB client. When you plug a gamepad into the Android device the Android device acts as a USB host. The term *USB On-the -Go* (or *USB OTG*) is used to describe a device's ability to switch between host and client modes.

Note that when an Android device is acting as a USB host, it cannot simultaneously be charged.[2] This means that when you have your Android tablet connected to the Logitech gamepads or when you have your Android phone connected to the Power Module or Legacy Module (with the phone acting as a USB host) the Android device *will be drawing power from its internal battery*. The Android device's internal battery *currently cannot be charged while it is acting as a host for the gamepad or robot modules*.

5.3.2 USB Micro Type B

The Android devices have a special type of USB port called a USB On-the-Go (OTG) port. This USB OTG port allows the Android device to automatically switch between host and client modes. The USB OTG devices typically have a *USB Micro Type B* receptacle:

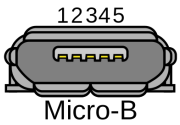

Figure 6 - USB Micro B receptacle[3].

The Android device's micro USB receptacle requires a USB Micro B cable:

[2] From http://electronics.stackexchange.com/questions/34741/can-an-android-tablet-serve-as-usb-host-and-be-charged-simultaneously-through-a accessed January 15, 2015.
[3] Image from http://en.wikipedia.org/wiki/USB#mediaviewer/File:USB_Micro_B_receptacle.svg accessed on January 15, 2015.

Figure 7 - USB Micro B cable[4].

[4] From http://www.monoprice.com/Product?c_id=103&cp_id=10303&cs_id=1030307&p_id=8639&seq=1&format=2 accessed on January 15, 2015.

5.3.3 USB Mini Type B

The electronic modules from *Modern Robotics* that are used to communicate between the Android phone and the robot hardware (motors, servos and sensors) are equipped with *USB Mini Type B* receptacles. This type of receptacle was selected by the manufacturer because they are the most durable type of connector, designed for repeated connect and disconnect cycles. These are the same style of USB receptacle found on high-end digital SLR cameras.

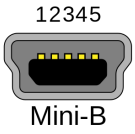

Figure 8 - USB Micro B receptacle[5].

The Legacy and Power Module's Mini B receptacle requires a *USB Mini Type B* style cable:

Figure 9 - USB Micro B cable[6].

[5] Image from http://en.wikipedia.org/wiki/USB#mediaviewer/File:USB_Mini-B_receptacle.svg accessed on January 15, 2015.
[6] From http://www.monoprice.com/Product?c_id=103&cp_id=10303&cs_id=1030302&p_id=8632&seq=1&format=2 accessed on January 15, 2015.

5.3.4 Micro USB OTG Adapter

A special adapter cable is used to connect the Android device to the Power Module or to the USB hub. This Micro USB OTG Adapter cable has a male USB Micro B connector on one end and a female standard USB Type A receptacle on the other end.

Figure 10 - USB Micro OTG Adapter Cable.[7]

When used with the driver station, the USB hub is connected to the Android tablet through the Micro OTG adapter cable.

When used with the robot controller, the Power Module is connected to the smartphone through the Micro OTG adapter cable.

[7] Image from http://www.monoprice.com/Product?c_id=108&cp_id=10833&cs_id=1083314&p_id=9724&seq=1&format=2 accessed on January 15, 2015.

6 The Android Operating System

6.1 What is Android?

The next gen platform uses a tablet and a smartphone to control a robot. The tablet and smartphone are actually sophisticated, compact computers. These handheld computers have a central processor, volatile and non-volatile memory, input/output devices, and other features commonly found on modern computers.

Android is the *operating system* that runs on these handheld devices. Similar to a laptop that has Microsoft Windows or MacOS as its operating system, a tablet or smartphone has its own operating system that manages the device's hardware and software components.

Figure 11 - Android is the world's most popular computing platform.[8]

You may have heard people refer to Android as *firmware*. The term *firmware* implies that the instructions and data that are make up the device's operating system are stored in the device's non-volatile memory. The "firm" in the word "firmware" originally implied that the instructions and data that are used to run an electronic device like a digital camera or MP3 player did not change very frequently (if at all). Typically, the firmware was installed once onto the device (at the factory) and left there for the rest of the device's life.

Android is more similar to the Windows, MacOS and Linux operating systems. It is installed onto non-volatile parts of memory of a tablet or phone. However, Android is often updated with patches or upgrades, just like Windows, MacOS and Linux. Even though Android can be updated regularly, people still often refer to it as "firmware."

Google produces Android, which is the world's most popular operating system.[9] Google develops and maintains the Android code. This source code is available to the public under an open source license.[10] Google provides free developer tools that can be used to write applications or "apps" for the Android platform. Starting with version 1.5, the different versions of Android are denoted by their version number and by the name of a dessert or candy. Android version 1.5 is known as Cupcake. Android 4.4 is known as Kit Kat. As of the January 2015, the most current version of Android (5.0) is known as Lollipop.

[8] Image from http://en.wikipedia.org/wiki/File:Android_robot.svg – Retrieved December 4, 2014.
[9] See http://developer.android.com/about/index.html for details.
[10] http://en.wikipedia.org/wiki/Android_(operating_system) – Retrieved January 15, 2015.

6.2 Android Studio

The driver station tablet and the robot controller smartphone run special apps for the FTC competition. These apps are created using tools that are provided for free by Google. The official development tool for Android development is known as *Android Studio*. In order to be able to write programs to control a robot, you will need to learn how to build apps using Android Studio.

Android Studio is a tool 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 FTC robots, the process is fairly simplified. The engineers at QualComm have developed an Android framework for the FTC robots. 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 much about developing the framework of the app.

6.3 MIT/Google App Inventor

Work is being done to integrate the new FTC platform with the MIT/Google App Inventor:

http://appinventor.mit.edu/explore/

The App Inventor is a user-friendly, server-based tool to create Android apps using a visual development environment.

The App Inventor is being modified so that it will able to support the FIRST Tech Challenge hardware. Teams will eventually be able to use the App Inventor to create apps for their competition bots.

Unfortunately, the App Inventor will not be covered in this manual. A separate App Inventor training manual will be available.

6.4 Java

Java is a popular text-based, object-oriented programming language. Android apps are written using Java syntax. 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 would 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, inheritance, and other key object oriented concepts.

7 Installing Android Studio

7.1 Important Note Regarding the Android Studio Instructions

IMPORTANT NOTE!!! This training manual contains instructions on how to install the Android Studio software onto your PC. Please note that 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.

The most up-to-date reference for installing and using the Android Studio software can be found on the Android developer website:

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

Please read and refer to the instructions on the Android developer website before you attempt to install this software onto your own PC.

7.2 Android Studio Website

In order to create programs for your next gen FTC robot, you will need to have Android Studio installed on your laptop. Android Studio is distributed freely by Google and it is available on the Android Developer's website:

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

Android Studio is available on the MacOS, Windows and Linux operating systems. This version of the manual currently only describes how to install Android Studio for the Windows environment.

7.3 System Requirements

Before you download and install the Android Studio 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

7.4 Installing the Java Development Kit

You will need to download and install the Java Development Kit (JDK) onto your computer, prior to installing the Android Studio software. At the time this document was originally written, Android Studio requires the JDK version 7 for Windows installations.

The JDK version 7 software can be downloaded from the Oracle.com website:

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

If you visit the JDK download website, you will want to select either the Windows X86 download file (if you are using a 32-bit version of Windows) or the x64 download file (if you are using a 64-bit version of Windows). At the time of this writing, JDK 7u75 is the kit you should download:

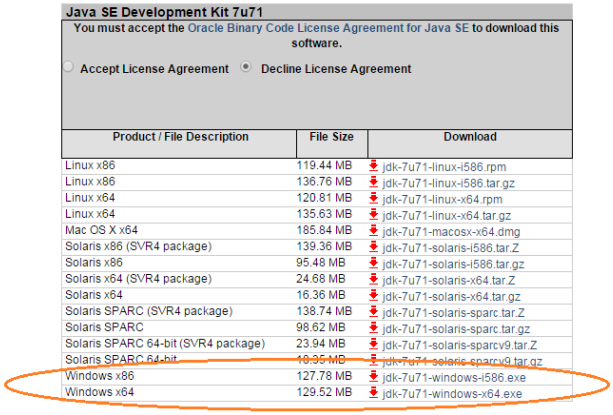

Figure 12 - Select x86 for 32-bit versions of Windows or x64 for 62-bit versions of Windows.

Note that you will only need the JDK 7 software. You do not need to download any of the extra files that are available on the Oracle website (such as the Demos & Samples download).

Once you have the executable file downloaded from the Oracle website, double click on the file to open it and start the installation. Follow the steps of the Installation Wizard to install the software onto your computer.

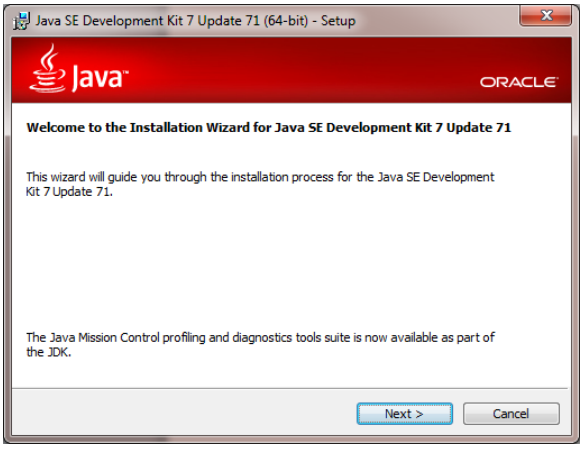

Figure 13 - Follow the steps of the Installation Wizard.

When prompted, select the default recommended settings from the Installation Wizard

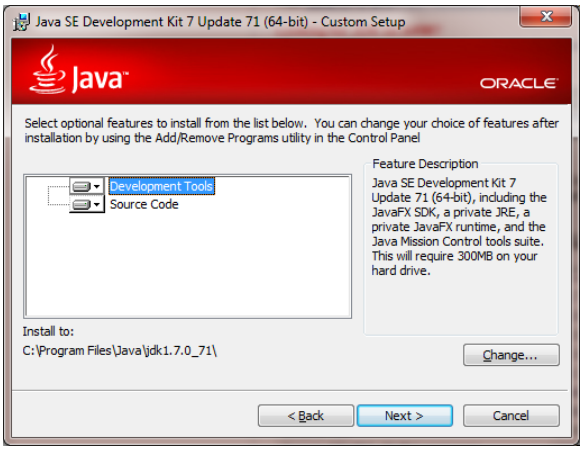

Figure 14 - Select the default recommended settings for the installation.

After the installation is complete, hit the "Close" button to close the Installation Wizard. You do not need to click on the "Next Steps" button.

Figure 15 - Hit the Close button to finish the installation.

7.5 Installing the Android Studio Software

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

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

On the Android Studio SDK website, click on the green button to take you to the download webpage:

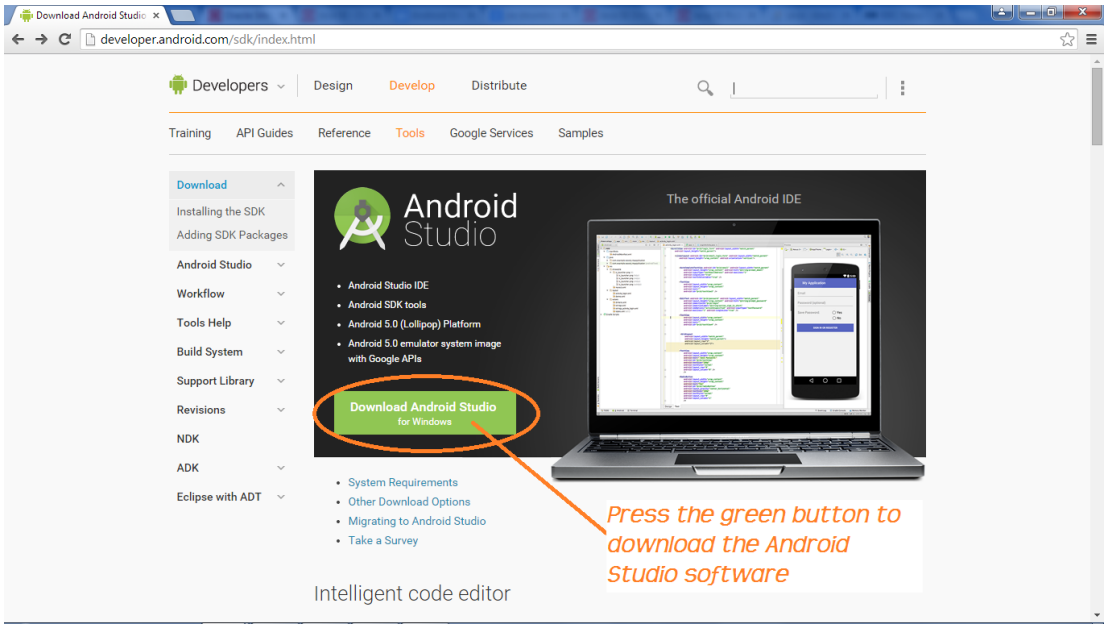

Figure 16 - Click on green button to download the software.

The Android web page should prompt you to accept or reject the license agreement. Check the checkbox to accept the agreement, then press the blue button to begin the download:

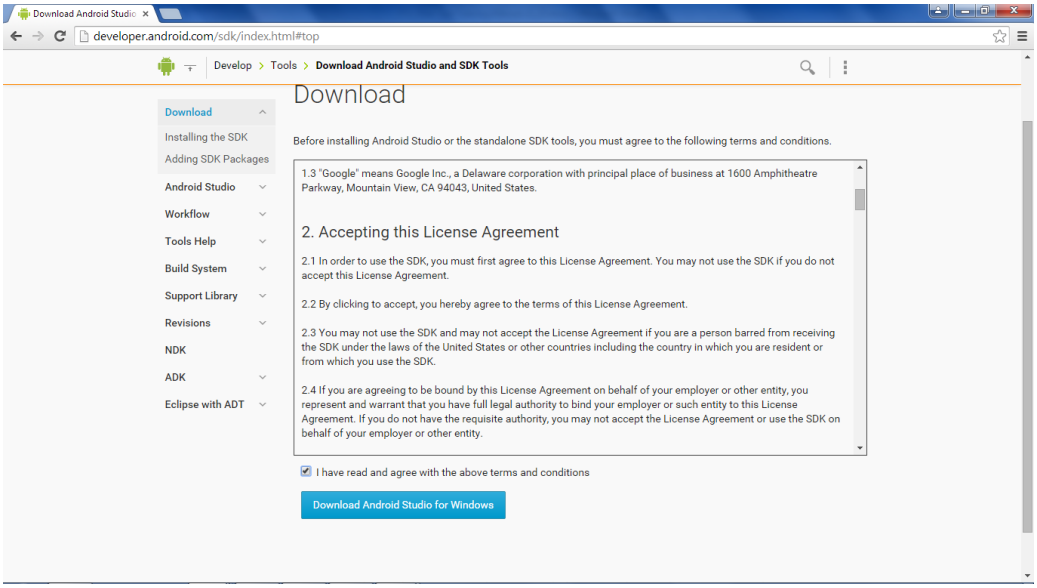

Figure 17 - Accept the license agreement and click blue button to begin download.

Once you have downloaded the installation file, double click on it to begin the install process. An Installation Wizard should appear. Follow the steps of the Installation Wizard to install the software:

Figure 18 - Follow the steps of the Installation Wizard to install the software.

When prompted by the Installation Wizard, select the default values for the settings and continue with the installation process.

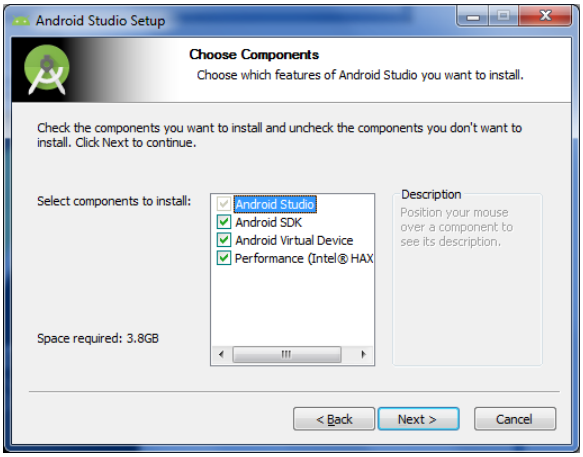

Figure 19 - Select the default values when prompted by the installation wizard.

During the installation process, the Installation Wizard will prompt you for the Install Locations for the Android Studio software, and the Google Android Software Development Kit (SDK). The Android Studio software is the tool (such as the text editor, project browser, and debugger) that you will use to create and edit your apps. The Android SDK contains the actual software components that are used to build and install the apps.

When the Installation Wizard prompts you for the Install Locations, you should note the install location of the Android SDK. You might need to know this location later on to find and run tools that are available in the SDK. In the screenshot below, the Android SDK will be installed to the directory "C:\Users\teng\AppData\Local\Android\sdk" on the Windows computer.

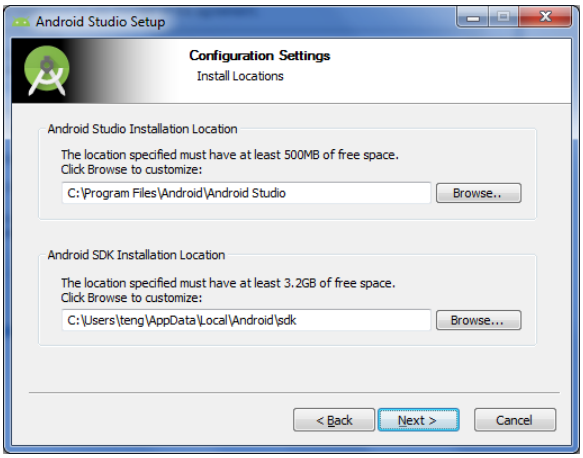

Figure 20 - Note the installation location of the Android SDK.

Continue with the installation process. Note that Android Studio is a large program and it can take a *long* time to install the required components. Please be patient and wait for the install to complete.

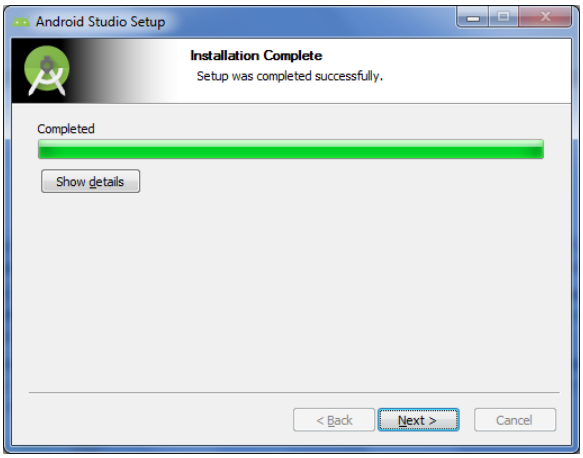

Figure 21 - Android Studio can take a long time to install the required components.

Once the installation is finished, press the "Finish" button to close the Installation Wizard and to launch Android Studio.

Figure 22 - Press "Finish" to complete the install and to launch Android Studio.

When the Android Studio software is launched, it might prompt you to import Android Studio settings from a previous installation of the software. Unless you are an advanced user who has previous Android Studio settings that you would like to preserve, select "I do not have a previous version of Android Studio or I do not want to import my settings" and click "OK" to continue.

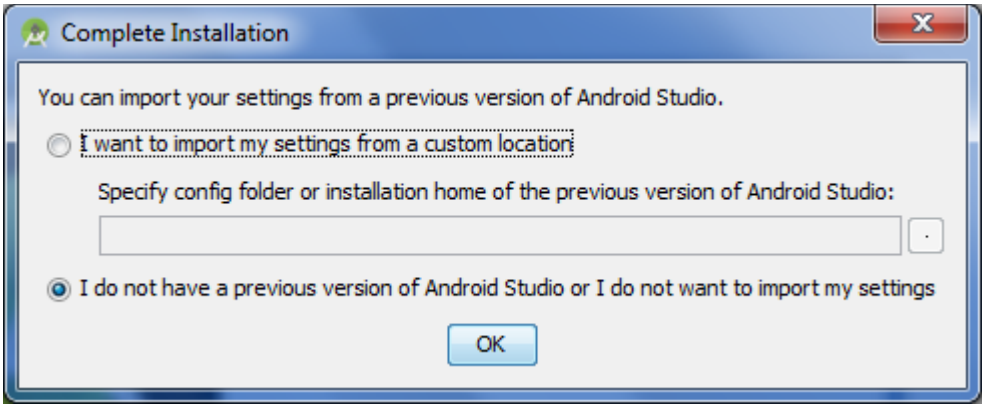

Figure 23 - Android Studio might prompt you to import previous settings.

Once Android Studio has finished downloading and installing the additional software components, click the "Finish" button to continue:

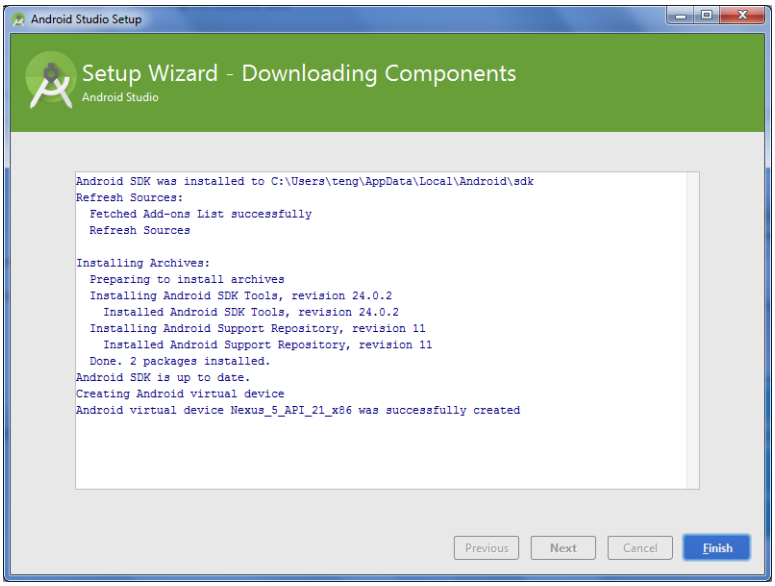

Figure 24 - Click on the "Finish" button to continue.

7.6 Android SDK Manager

The Android Studio software lets you launch an application known as the Android SDK Manager. The Android SDK Manager can be used to install SDK components and tools that you might need to build your app. You can launch the SDK Manager from the Welcome screen of Android Studio. Select "Configure" from the Welcome screen to open a configuration menu:

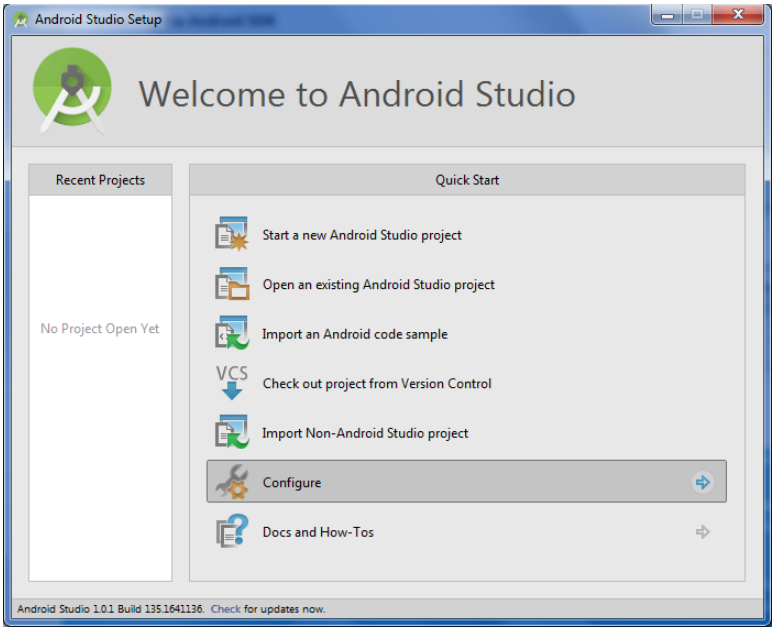

Figure 25 - Select "Configure" from the Quick Start menu of the Welcome screen.

In the Configure menu, select the "SDK Manager" option to launch the Android SDK Manager:

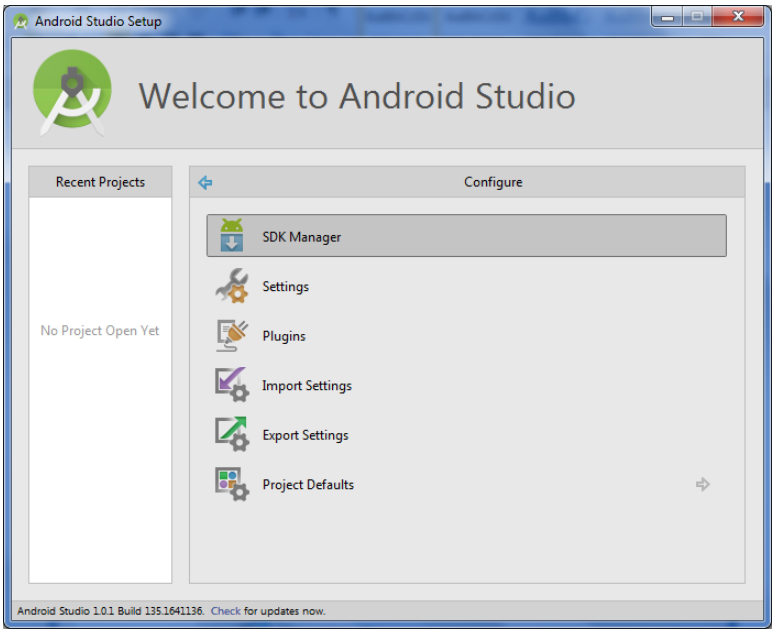

Figure 26 - Select SDK Manager from Configure menu.

The SDK Manager will list available packages of software tools that you can download if they are not already installed on your machine. The SDK Manager also lets you remove/uninstall components from your computer. In the screen grab below you can see that the "Android SDK Tools" (revision 24.0.2), the "Android SDK Platform-tools" (revision 21) and the "Android SDK Build-tools" (revision 21.1.2) are already installed on the computer.

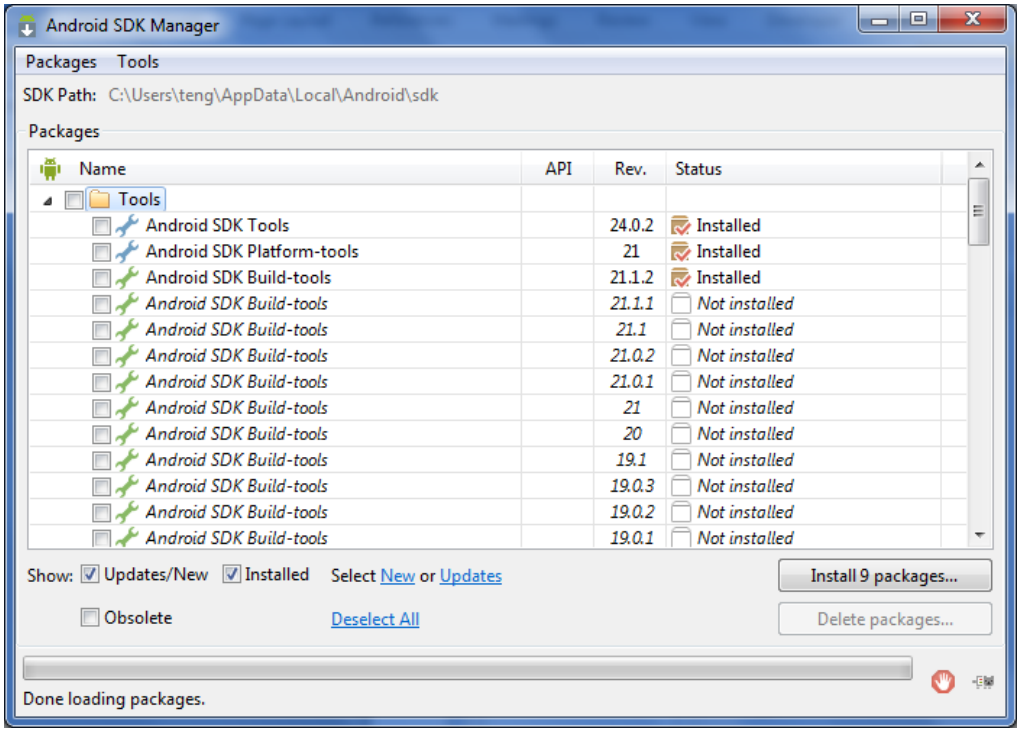

Figure 27 - The SDK Manager shows what software packages are available and what have been installed.

The Android Studio development environment will use the SDK tools to build your Android app. You can configure the Android Studio software to use different versions of the SDK tools to build your app. Sometimes, you might want to use a more recent version (such as Android SDK Tools, revision 24.0.2) of the tools because they contain bug fixes or new features that you would like to incorporate into your app. Other times, you might want to use an older version of the SDK tools, because these tools are more stable (have less bugs) or work better with your target Android hardware.

The SDK Manager also lists some extra software tools which might be useful for your development environment. If you are a Windows user, then you want to make sure a couple of "Extras" are installed onto your local machine. Towards the bottom of the scrolling view of the SDK manager, there is a package of components labeled "Extras". Navigate to this package and expand the package to view its components. Make sure the components "Google USB Driver" is installed on your machine. If they are not installed, make sure you select them by checking the boxes next to their listing, then hitting the install button.

In the screen grab below, the Google USB driver was not yet installed. The check box next to the Google USB component has been selected, indicating that the user would like to install the Google USB driver onto the machine.

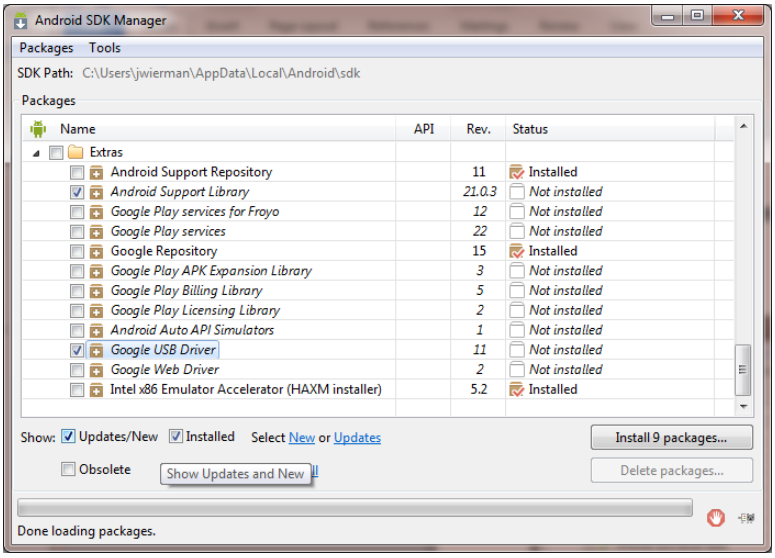

Figure 28 - In this image the USB Driver is not yet installed.

After you have selected which components you would like to install/uninstall, click on the install button then accept the user license agreements to begin the download and installation process:

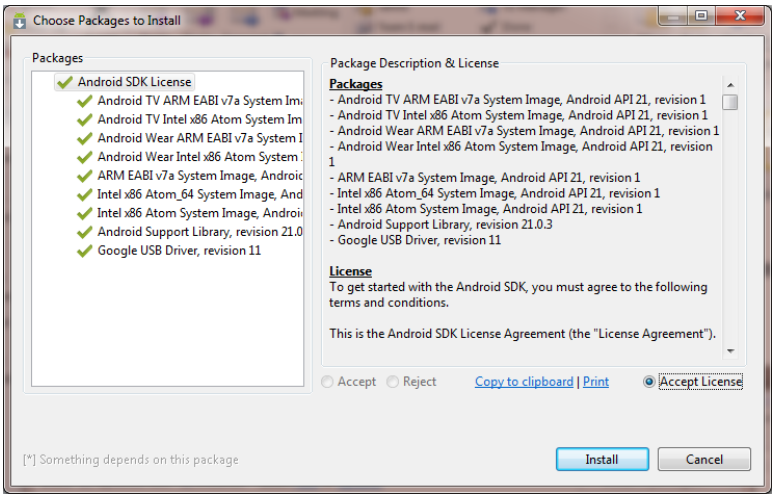

Figure 29 - Accept the license agreements to download and install the components.

Once you have completed installing the SDK components and extra tools, you can close the Android SDK Manager. Note that it might take a long time for the SDK Manager to download and install the components, so please be patient while it does so.

7.7 Building your First App

Once you have installed the SDK components that you would like for your computer, you can close the SDK manager and start programming your first app. On the Android Studio Welcome screen, you should see an option to "Start a new Android Studio Project" in the Quick Start menu. Click on this option to start a new project.

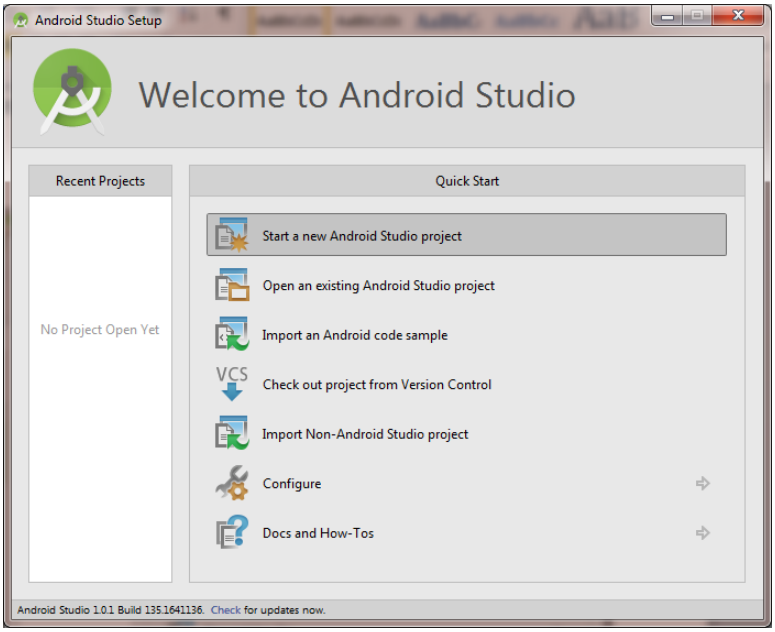

Figure 30 - Select "Start a new Android Studio project" from the Quick Start menu

Android Studio will prompt you for some information about your new project. You should provide a name for your project (the default value of "My Application" is fine – you can leave it there). Also, Android uses a naming scheme which uses the names from an Internet domain that you provide to make the Java classes that you create unique. Android Studio will store your classes using this domain name. This helps avoid conflicts with classes that you might use from other software libraries.

For this example, you can leave the default domain name ("jwierman.example.com" in the screen shot below, yours will be slightly different).

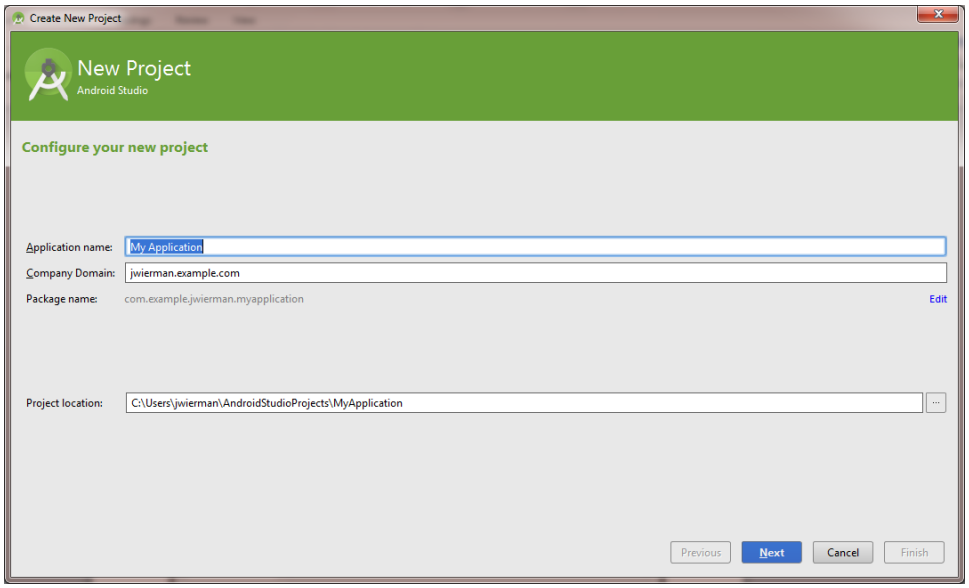

Figure 31 - You can accept the default values for the application name, domain, and project location.

The next screen prompts you for the minimum SDK version that you want to target your app to work with. Since our app is going to be a simple one, we can target an older version of the Android Operating system. Accept the default value (which in the following screen shot is Android 4.0.3, Ice Cream Sandwich):

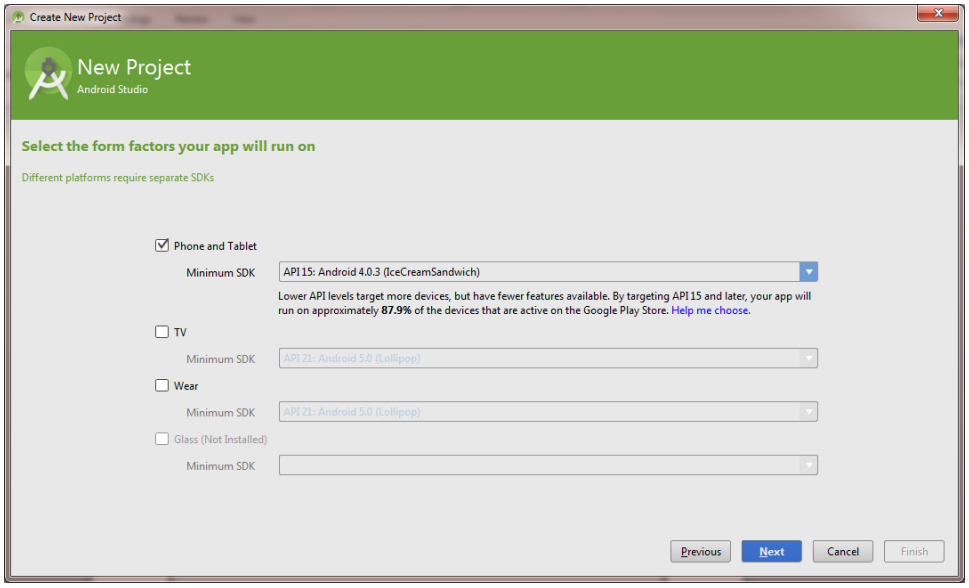

Figure 32 - Accept the default minimum SDK version that your app will work with and press Next.

Android Studio will prompt you to specify what type of *activity* that you would like to add to your project. An activity is like a window on a computer application. An activity is a container that holds *widgets* for your Android app. Make sure the default activity type (a "Blank Activity") is selected and press Next to continue.

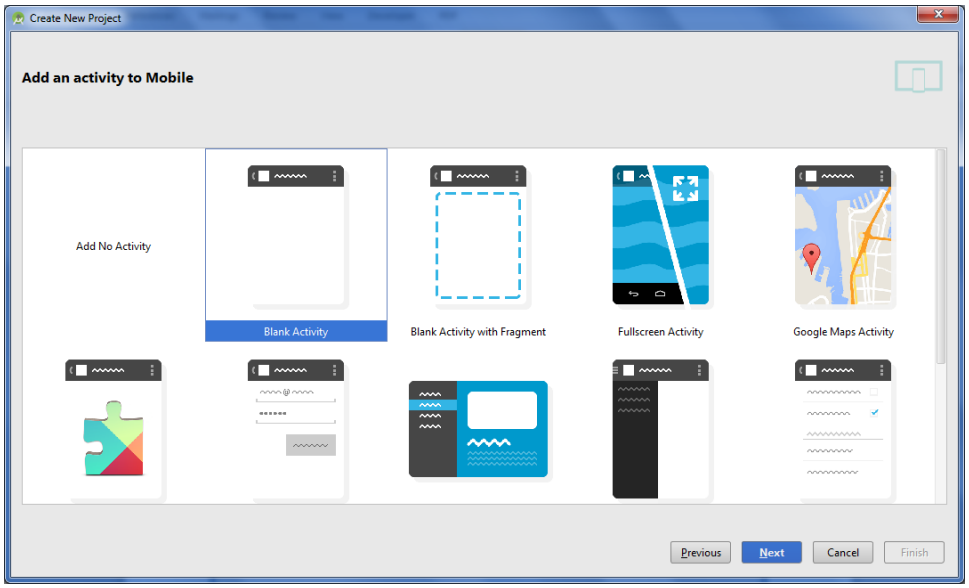

Figure 33 - Select the *Blank Activity* type and hit Next.

Android Studio will prompt you for information about the blank activity that it will create for your project. You can accept the default values and push the "Finish" button to generate the app.

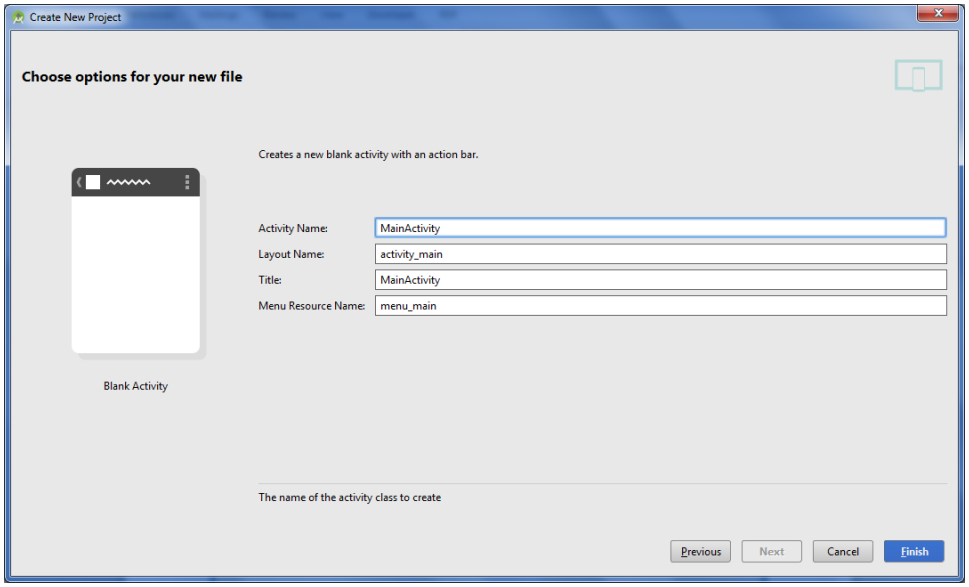

Figure 34 - Accept the default values and click Finish to generate your app.

The Android Studio software will use the parameters that you specified to generate a new project.

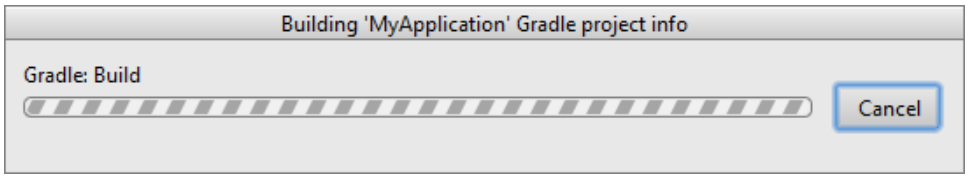

Figure 35 - Android Studio will build the project.

While it is generating the project, the Windows operating system might flash some warning messages about Java and/or Android Studio trying to access the network. You should "Allow access" for the Java and Android Studio applications.

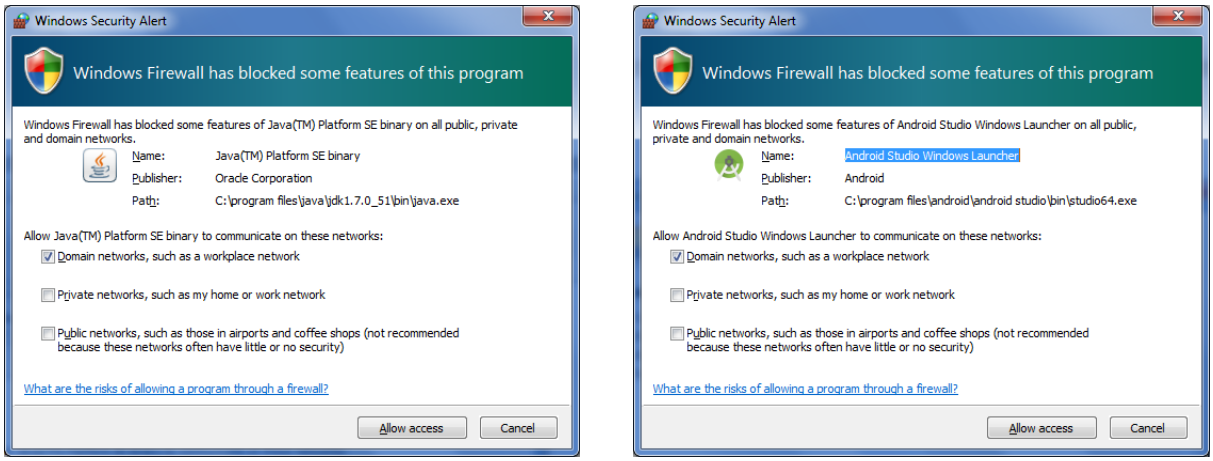

Figure 36 - You should allow access to your networks for Java and Android Studio.

Once the project has been successfully created you should see the Android Studio Tip of the Day screen. You can close this window and also uncheck the "Show Tips on Startup" option if you do not want to see any more tips.

Figure 37 - Android Studio has a Tip of the Day screen.

If you were able to install Android Studio and the JDK successfully, you should see an Android Studio screen for your newly created project.

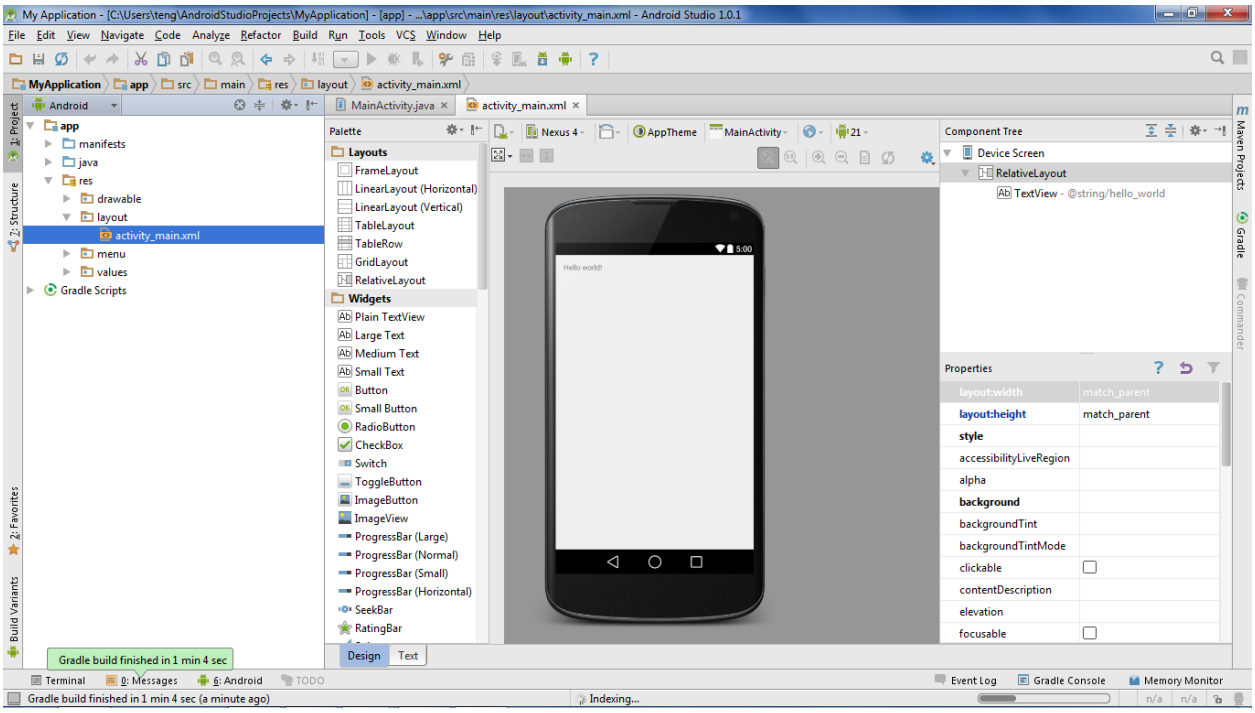

Figure 38 - If you successfully create a project, your screen should look something like this.

Take a look at the screen shot in Figure 38. At the top of the screen there is a set of menus, similar to the menus you might find on any other Windows application. You can use these menus to get information about your project or adjust settings for your project or Android Studio environment.

On the left hand side of the screen you can see a project browser. This project browser is similar to the Windows File Explorer. You can use the project browser to browse through the components of your Android project. You can click on the project components to expand or collapse them (i.e., show more or less information about that component).

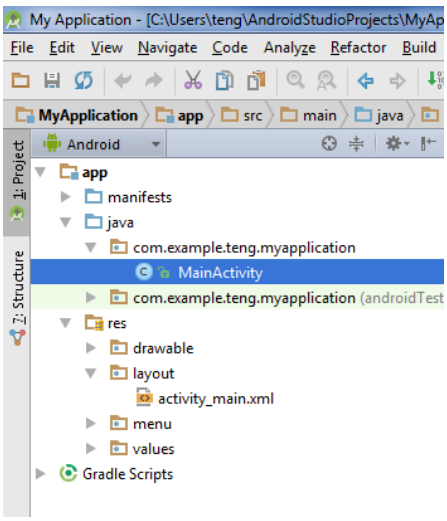

Figure 39 - The project browser can be used to browse the components that make up your project.

When you select a project item, such as a .XML file or a .java file, through the project browser, the opened file will be displayed on the right hand side of the screen. For example, let's consider the file named "activity_main.xml" which is located in the project sub directory "app->res->layout" in the screenshot below. When you click on activity_main.xml the file should open up and appear on the right hand side of your display.

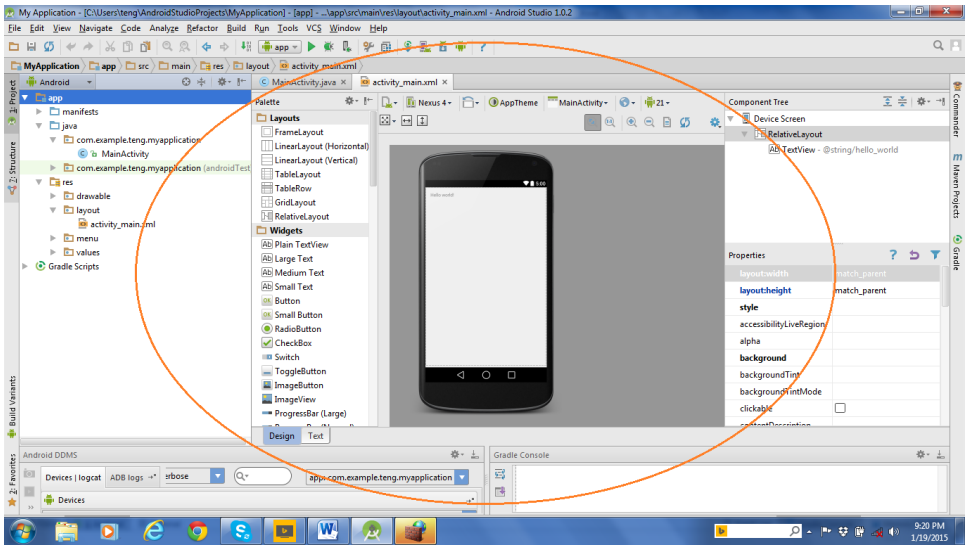

Figure 40 - Selected files will open up in the right hand side of the screen.

If you have more than one file open on the right hand side of the dispay, you can click on the tabs towards the top of the screen to select the file that you would like to view.

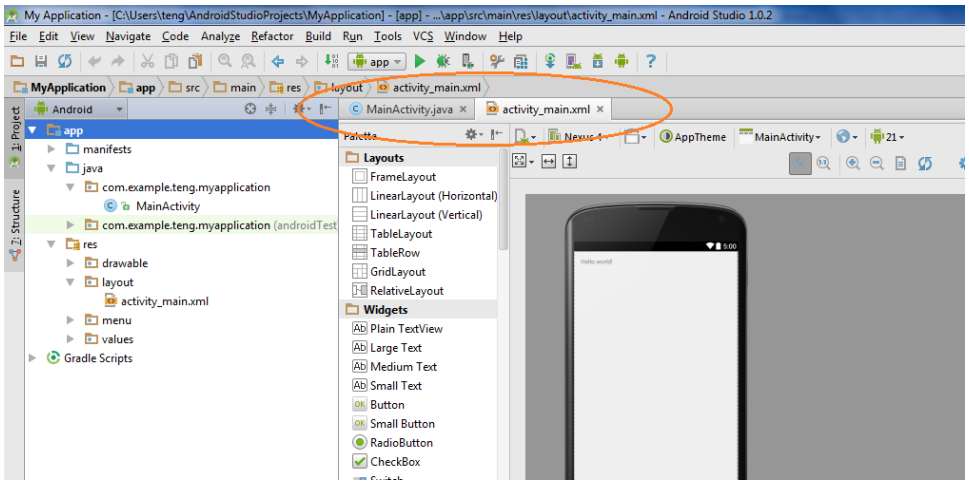

Figure 41 - You can choose which file to view by clicking on its tab near the top of the display.

In Figure 40, the activity_main.xml file contains the layout information for the main activity of the app that you are creating. Android Studio has a What-You-See-Is-What-You-Get (WYSIWYG) style graphical editor that lets you drag widgets (things like text boxes, buttons, etc.) onto your activity.

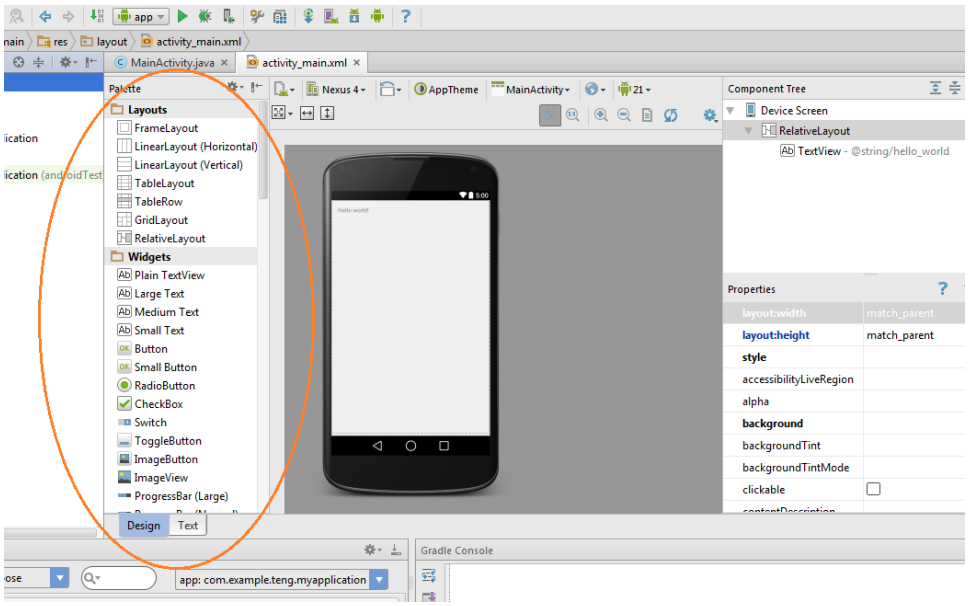

Figure 42 - Android has a WYSIWYG design editor that lets you drag widgets onto the activity screen.

By clicking on the "Text" tab towards the bottom of the screen, Android Studio lets you also edit the layout of your activity with a text editor.

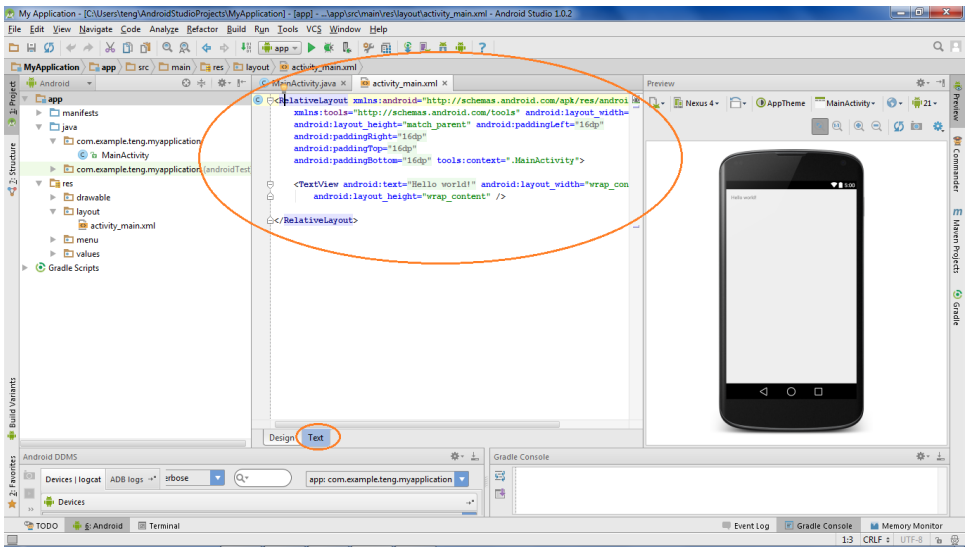

Figure 43 - Click on the "Text" tab near the bottom of the screen to see the text version of the XML file.

Believe it or not, our simple test program is just about done. For this exercise, we are only going to have a simple blank app that reads "Hello world" on it. You might have already noticed this text in the design preview in Figure 42 or Figure 43.

Before we can build and install our app, we will first need to prepare our machine to connect to a real Android device. We will learn how to do this in the next section of this manual.

7.8 Connecting Android Studio to a Real Device

We are just about ready to install our simple test app onto a real Android device. Before we can do this, we need to first make sure that our Android device has the *Developer Options* and *USB Debugging* enabled. We also have to make sure that our Windows PC has the correct USB driver installed for the Android device that you are using.

7.8.1 Removing the SIM card from the ZTE Speed

The ZTE Speed is the recommended Android device for the upcoming 2015-2016 FTC season. Before you use your phone, you should physically remove the SIM card, and then place the phone into Airplane Mode (with WiFi still enabled) in order to make sure that the Speed phone does not try to connect to the Boost Mobile network whenever it is turned on.

Figure 44 - ZTE Speed

The first step in this process is to make sure that your phone is powered off. You will need to remove the plastic back cover of the phone. With the screen facing you, if you look in the lower right hand corner of the device, you will see a gap that you can use to pry the plastic rear cover off of the phone. Pry the back cover off of the phone and then place the phone screen down onto the table.

Along the right hand side of the phone, you should see the SIM card. Push in on the SIM card – this will eject the card partially from its slot. Remove the card and store it in a safe place. Replace the back cover and then power the phone on.

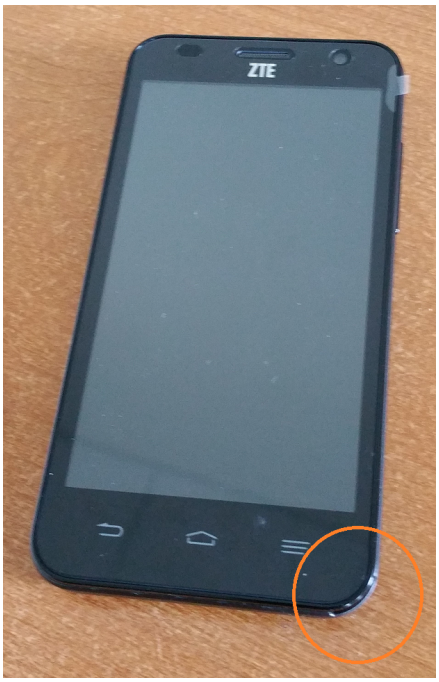

Figure 45 - There is a gap in lower right hand corner that you can use to pry the back cover off the phone.

Figure 46 - Pry the back cover off of the phone.

Figure 47 - The SIM card is installed in the right hand side.

Figure 48 - Push in on the SIM card to eject it from its slot.

Figure 49 - Remove the card from its slot and replace the back cover.

After you have removed the SIM card from the ZTE Speed phone, power the device on and step through the opening screens (see images below):

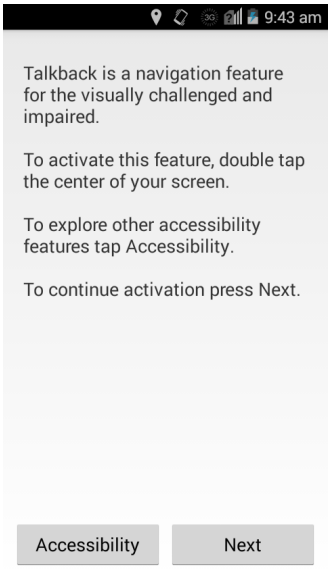

Figure 50 - The phone will ask you if you would like to set up Talkback. Hit Next to skip this step.

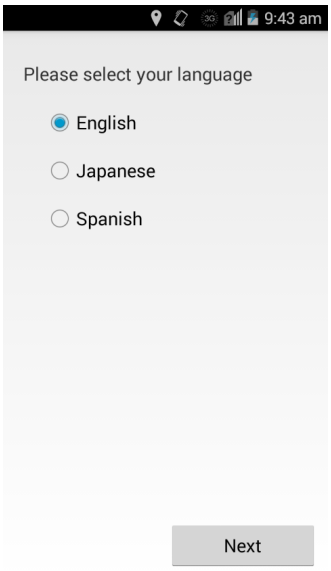

Figure 51 - Select your language and hit Next to continue.

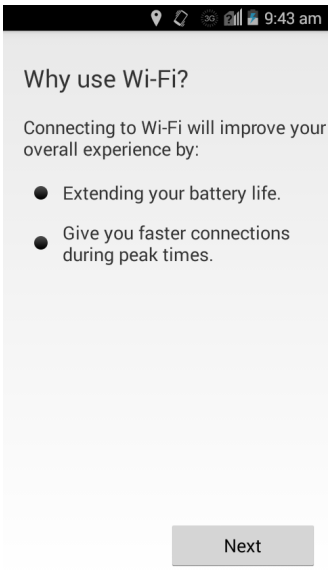

Figure 52 - The phone will prompt you to connect to a WiFi Network. Hit Next to continue.

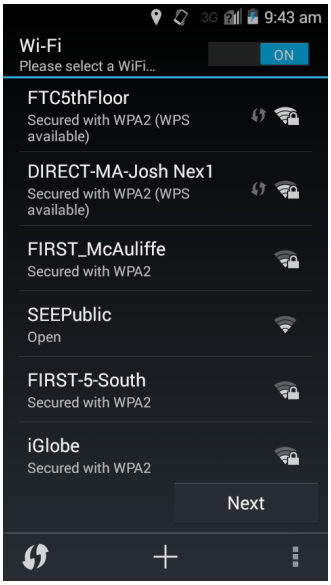

Figure 53 – You can connect to a WiFi network at this point or skip this step (hit Next).

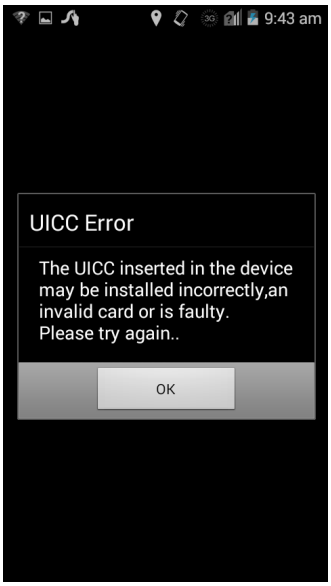

Figure 54 - You might see this message indicating that the SIM card was not detected. Hit OK to continue.

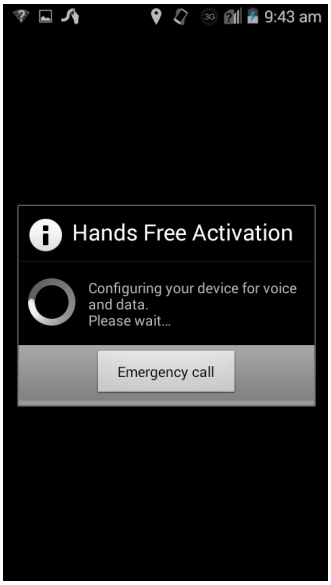

Figure 55 - The phone will display a Hands Free Activation screen.

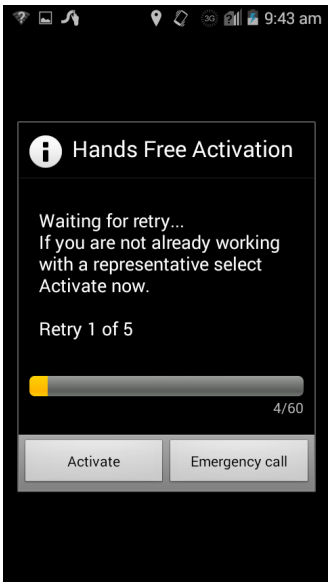

Figure 56 - The phone will try to activate itself. Hit the "Activate" button to continue.

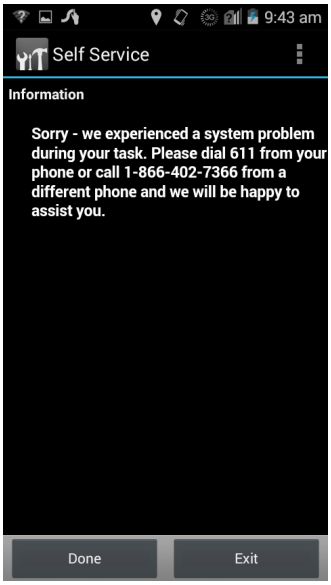

Figure 57 – The activation should fail. Hit “Done” to complete the process.

Figure 58 - You should now see your home screen. Press the Apps symbol to display the available apps on your phone.

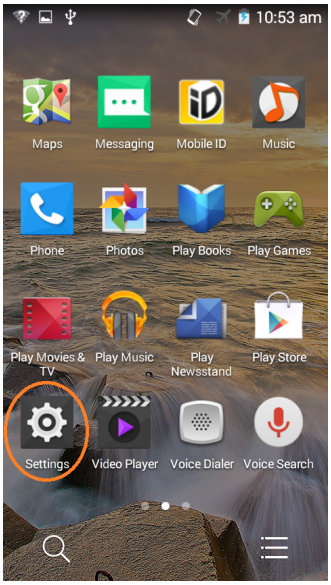

Figure 59 - Find the Settings app and press it to launch the Settings activity.

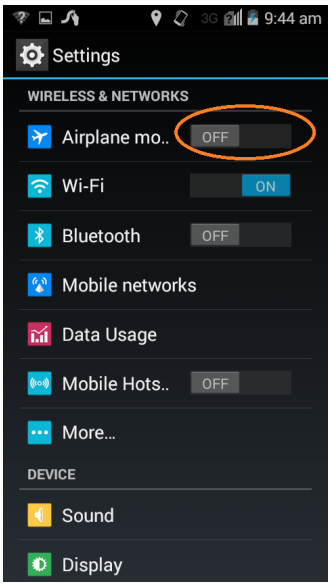

Figure 60 - Turn on Airplane Mode so the device won't try to connect to mobile network.

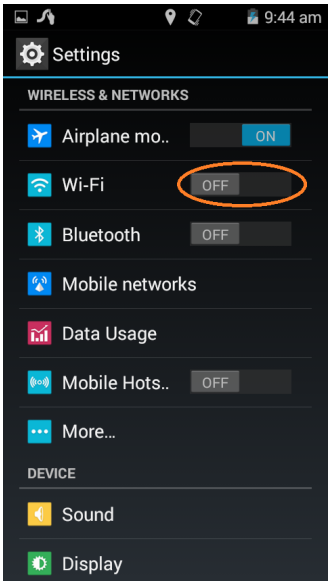

Figure 61 - Turn WiFi back on once the phone is in Airplane Mode.

Once you have removed the SIM card and switched the phone to Airplane Mode it is now ready for use.

7.8.2 Enabling Developer Options on your Android Device

For this training manual, we assume you will be using a ZTE Speed Android phone as your primary development device. In order to be able to be able to use this device with the Android Studio software, you will need to make sure the phone has “Developer Options” enabled and that the USB Debugging mode is enabled.

Detailed instructions on how to enable the Developer Options mode (and how to connect an Android device to Android Studio) are available on the Android developer website:

http://developer.android.com/tools/device.html

If your Android device is not yet in developer options mode, you can go to the settings screen of your Android device (touch the icon that looks like a cog or gear) and go to the **Settings->About** phone activity. Look for the **Build number** or in the case of the ZTE Speed look for the **SW Version** of your phone. Tap the Build number or SW Version number seven times to enable Developer Options on your device.

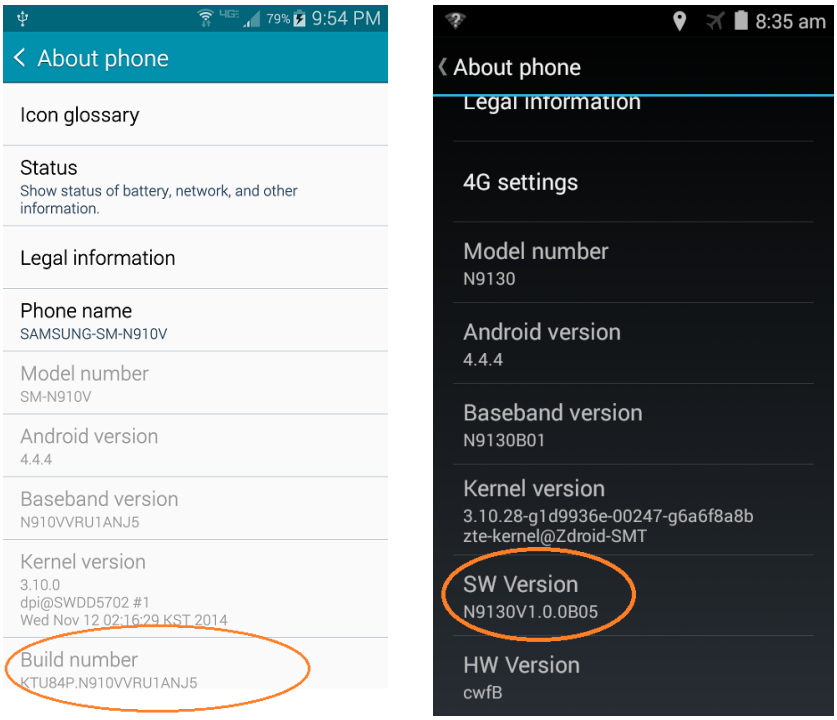

Figure 62 - Tap on the Build number (left) or SW Version number (right) 7 times to enable Developer Options.

Exit the About phone activity and look at your Settings screen. You should now see the "Developer options" item listed above "About phone" in the Settings menu.

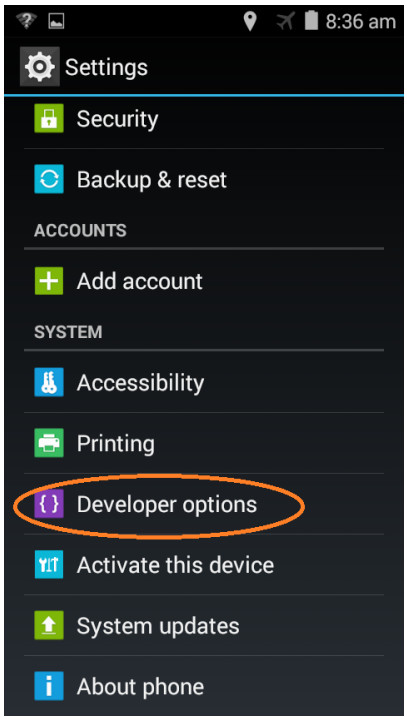

Figure 63 - Once Developer Options are enabled you should see a listing in the Settings menu.

7.8.3 Enabling USB Debugging

In order for your Android Studio computer to be able to "talk" to your Android phone properly, you will need to enable USB Debugging for your phone. Click on the "Developer options" item of the Settings menu (see Figure 63). The Developer options screen should appear on your phone. Look for the "USB debugging" option and check the checkbox to enable this option on your phone:

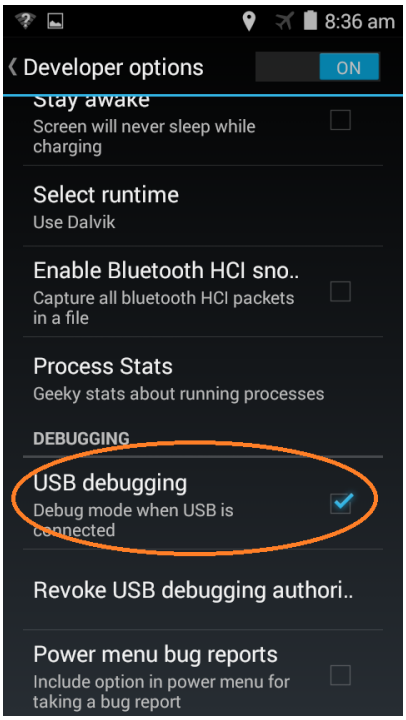

Figure 64 - Make sure USB debugging is enabled for your phone.

Once the "USB debugging" option is enabled for your phone, when you connect your phone it will connect in debug mode, which will allow Android Studio to "talk" to your phone as an Android device.

7.8.4 Installing the USB Driver for your Device

If you've enabled "USB debugging" for your Android device, you are just about ready to install your Android App onto the device. However, there is one last step that you need to do in order to connect your device successfully to Android Studio.

You will need to install the correct driver for your Android device onto your Windows computer. If your Windows computer does not have the correct driver for your phone installed, then when you connect your device to the computer via a USB cable, the Windows computer will think that the device is a media player or storage device (like a FLASH drive). You might be able to access files on your device in this mode, but your Android Studio software will not recognize the device as an available Android device that it can talk to.

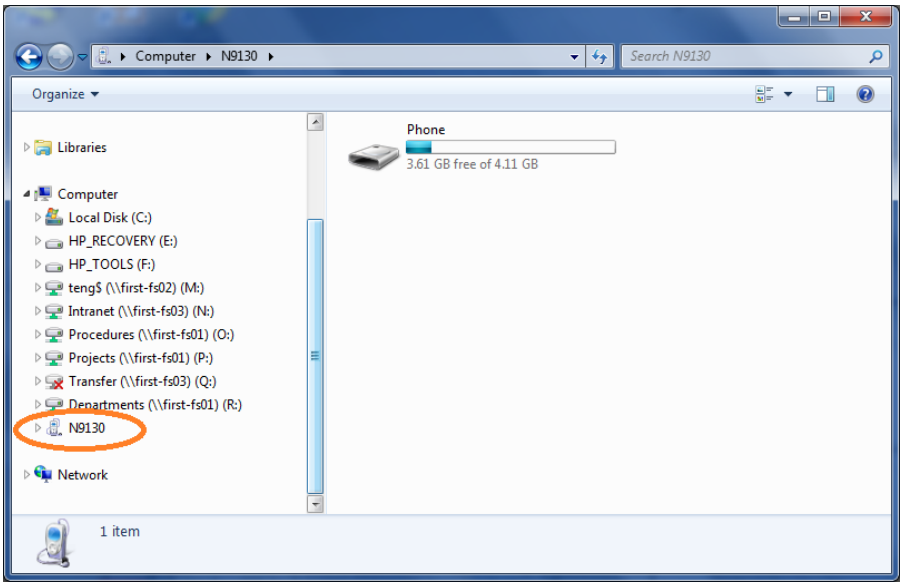

Figure 65 - If the proper driver is not installed, your PC might assume the device is a USB media player or storage device.

If you are able to install the correct USB driver onto your computer, then the next time you plug your Android device in with a USB cable, the computer and Android studio will recognize it as an Android device and Android Studio will be able to "talk" to it properly.

The ZTE Speed driver is conveniently included on the phone itself. When you first connect your ZTE speed to your computer using a USB cable, you should notice a USB symbol appear near the top of the screen (see figure below).

Figure 66 - When connected to PC, a USB icon should be visible near top of screen (in orange circle).

To display the **Connect to PC** activity, swipe downwards from the top of the touch screen and click on the “Touch for other USB options” item.

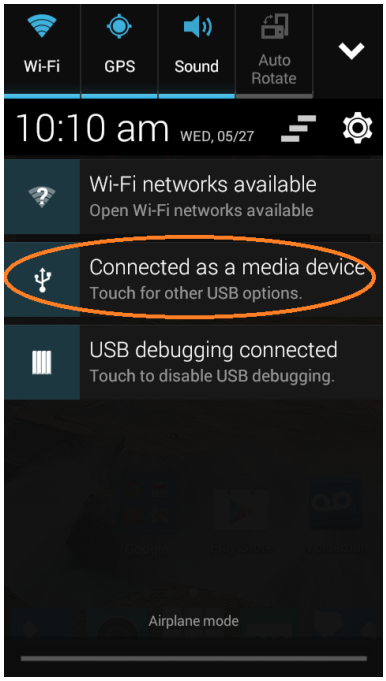

Figure 67 - Swipe downwards and select "Touch for other USB options"

To install the driver, select the “Install driver” option from the screen.

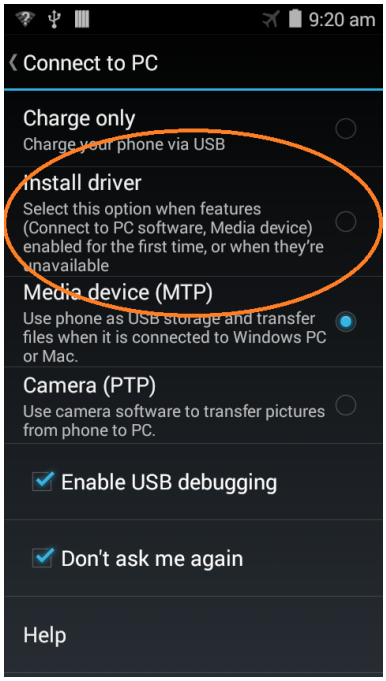

Figure 68 - Select the "Install driver" option.

Once you select the “Install driver” option on the phone your PC should prompt you if you want to run the install program (“AutoRun.exe”). Select the “Run AutoRun.exe” on the PC dialog box to start the installation.

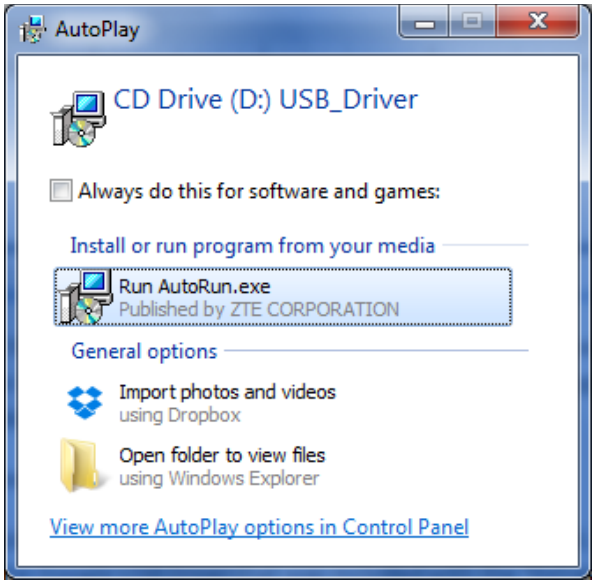

Figure 69 - Select Run AutoRun.exe option to start installation.

The install wizard should prompt you for the Setup Language. Choose your language and continue with the installation.

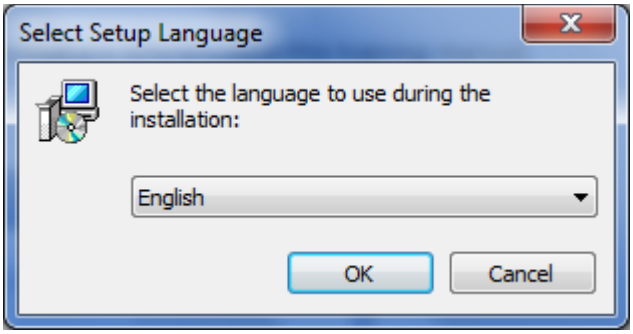

Figure 70 - Select language.

Figure 71 - Follow the on screen instructions to install the driver.

Follow the instructions on the setup wizard screens to install the ZTE handset USB driver. The setup wizard will display a progress bar showing the status of the install. When the install is complete, click on the "Finish" button to end the installation process.

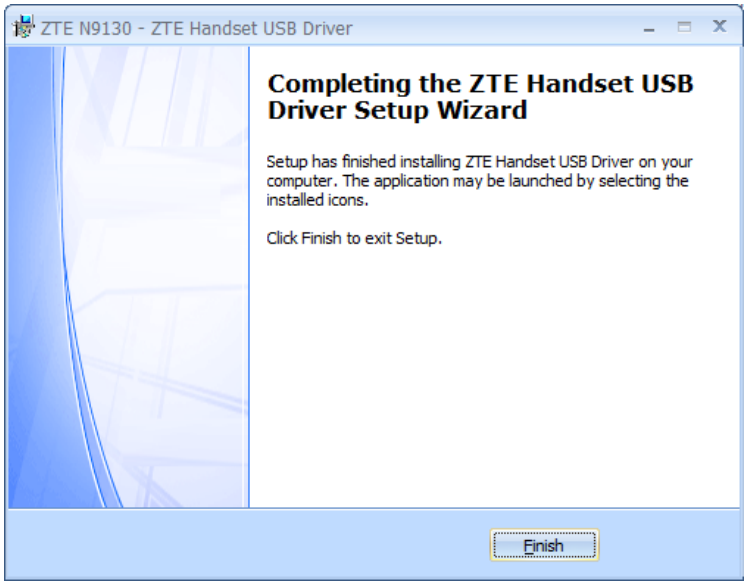

Figure 72 - Click on Finish to complete installation.

After installing the ZTE Speed driver, you want to go back to your ZTE phone and go to the **Connect to PC** activity again (swipe downwards from the top on the main screen and select the "Touch for other USB options"). Select the "Charge only" option from the list of options.

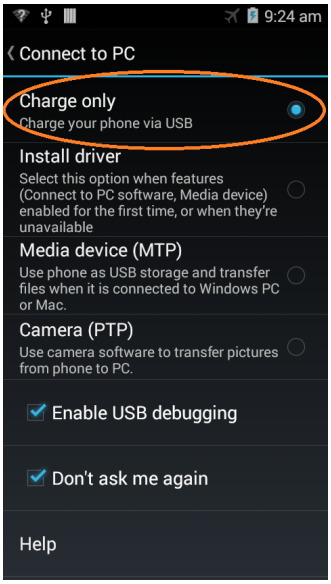

Figure 73 - Select the Charge only option.

If have successfully installed the USB driver on your PC, then your ZTE Speed phone should prompt you on whether or not to allow USB debugging by your PC.

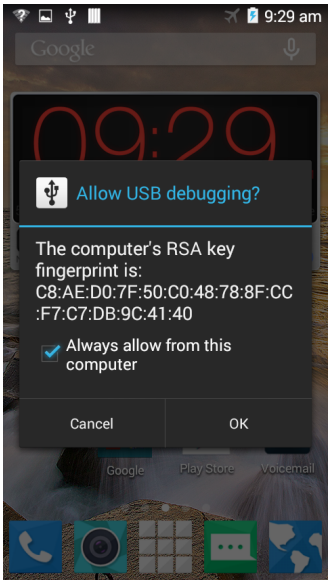

Figure 74 - Select the "Always allow" option and hit OK.

Note that if you do not see this prompt, you can unplug your phone from the USB cable, wait a few seconds, and then reconnect your phone to the cable. You should now see the prompt. Select "Always allow from this computer" option and hit "OK" to allow USB debugging access for your PC.

Once you have successfully installed the driver and configured your ZTE Speed phone, you should see the phone appear as an available Android device in the Android panel of Android Studio.

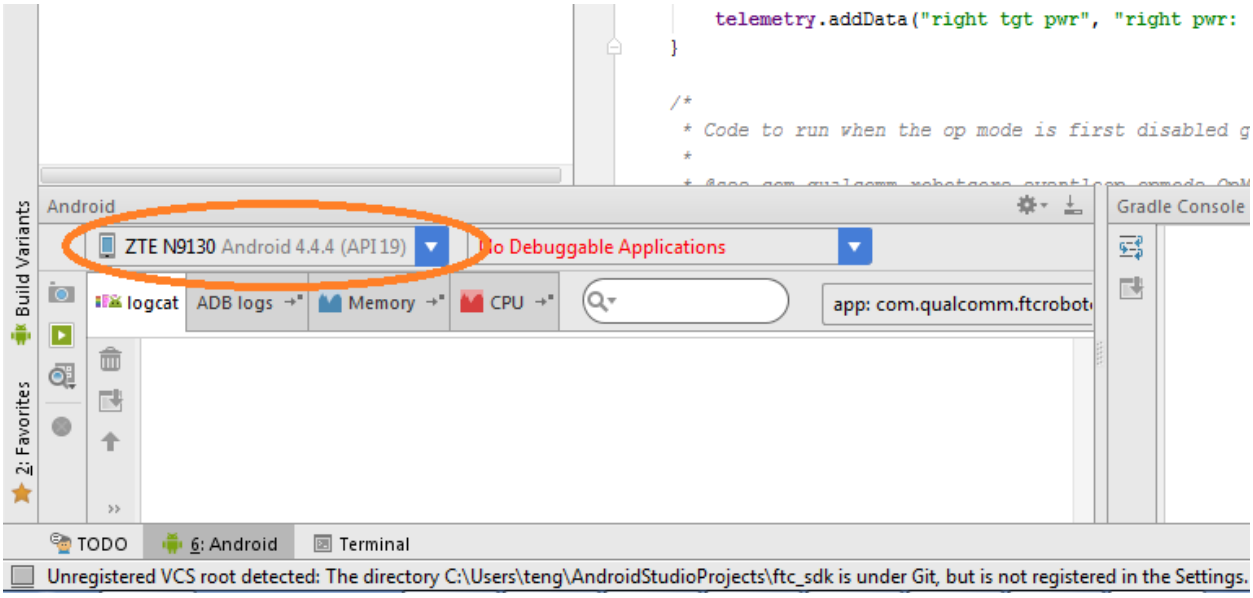

Figure 75 - Device should be visible in the Android panel of Android Studio.

Once you have installed your driver successfully, you are ready to build and install your first app!

7.8.5 A Note about Emulators

If you do not have access to an Android app, you can run apps onto a virtual Android device that behaves like a real Android device, but runs on your computer instead. For the *FIRST* Tech Challenge programming, we need to run our apps on real Android devices because the device must interface with other hardware devices (Legacy Module, motor and servo controllers, etc.). We will not cover the use of Emulators in this manual, but if you would like to use an emulator to learn how to develop apps, you can consult the Google developer website:

http://developer.android.com/tools/devices/emulator.html

Please note that using an Android emulator typically requires special graphics hardware accelerators to run properly. Your computer might not be equipped with the correct hardware to be able to run an emulator properly. For all of the exercises in this manual, we recommend using a real Android device (preferably the ZTE Speed smartphone) instead of an emulator.

7.9 Compiling and Installing an App

If you have successfully installed the USB driver and enabled USB Debugging mode on your phone, then you are ready to reconnect the phone and compile and install an app onto the device.

The first thing to do is to unplug your phone, wait for a few seconds (5), then reconnect your phone to the computer with the USB cable. When you do this, the Android device might ask you if it is OK to

allow USB debugging with the attached computer. Select the "OK" button to allow USB debugging. Note that if you would like to disable this prompt, you can check the "Always allow from this computer" option before you select "OK" on the screen prompt.

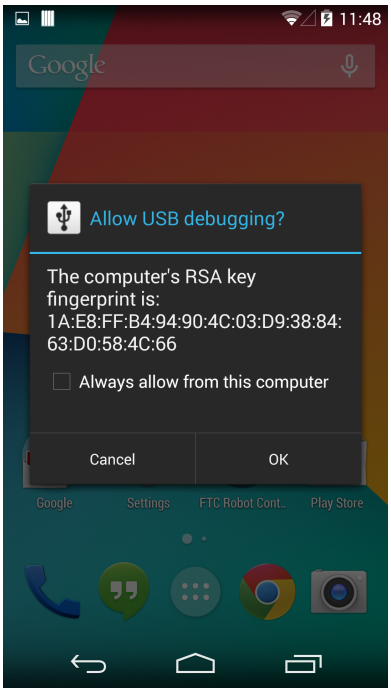

Figure 76 - Your Android device might ask you if it is OK to allow USB debugging.

Afer you have reconnected your phone, return to the Android Studio screen. Go to the **Run** menu and select **Run->Run 'app'** (where 'app' is the name of the application folder that you would like to build). Android Studio should start to build the app for you.

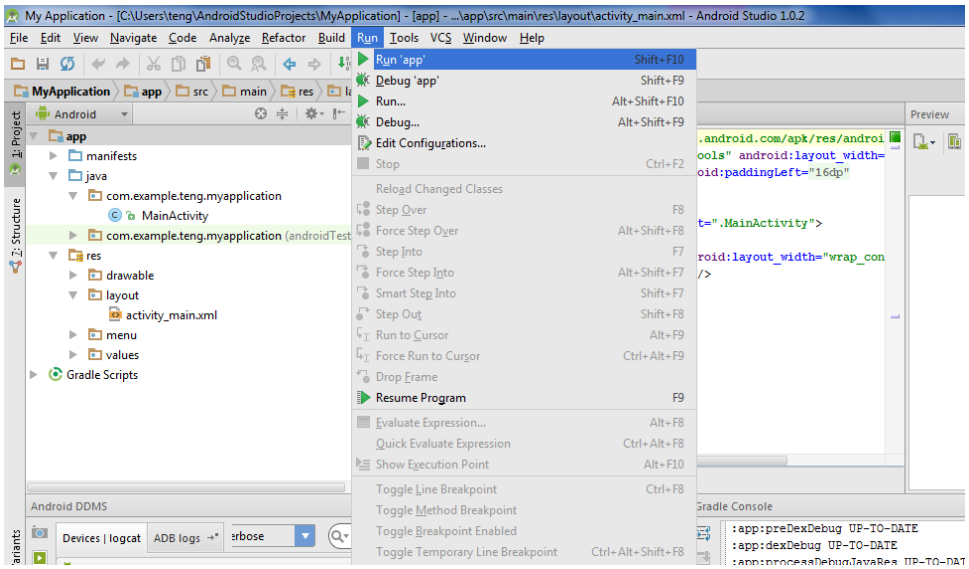

Figure 77 - Select Run->Run 'app' to build your app.

You can monitor the progress of the build by opening the Gradle window. Gradle is a tool that is used to help automate the build process for a complex piece of software. Gradle is similar tools like *make* or *Apache Ant*. Android Studio uses Gradle to keep track of all of the program module dependencies, and to build and link these modules to create your app.

You can click on the **Gradle Console** tab in the lower right hand corner of the screen. This will open the Gradle console window and you should see messages from the gradle process appear. If your build is successful, you should see the words “BUILD SUCCESSFUL” appear in the gradle window.

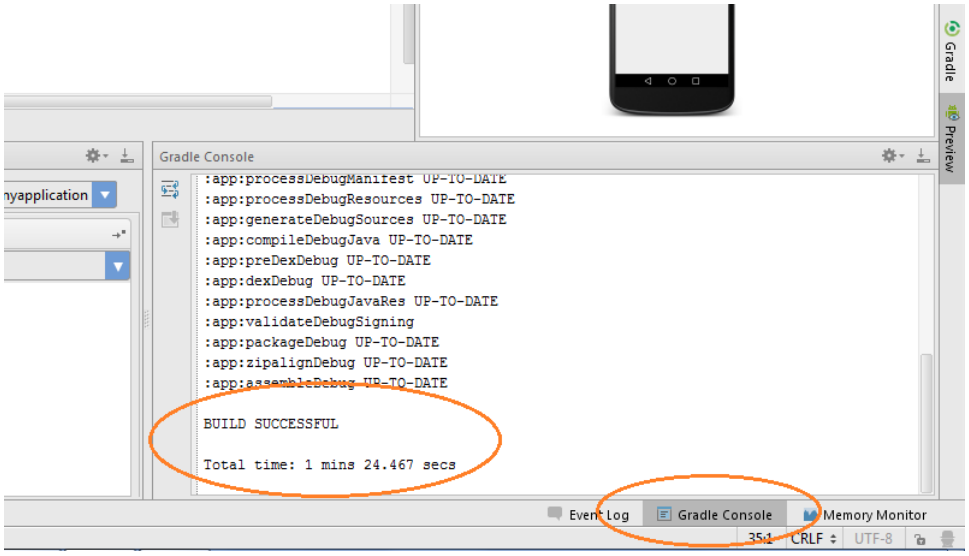

Figure 78 - When build is complete, the words "BUILD SUCCESSFUL" should appear in the Gradle console.

Once your build is complete, Android Studio should ask you to select a target device for your app. Make sure your phone is selected and hit **OK** to install the app onto the phone. Note that in the screen grab below, the phone that is listed is an LGE Nexus 5.

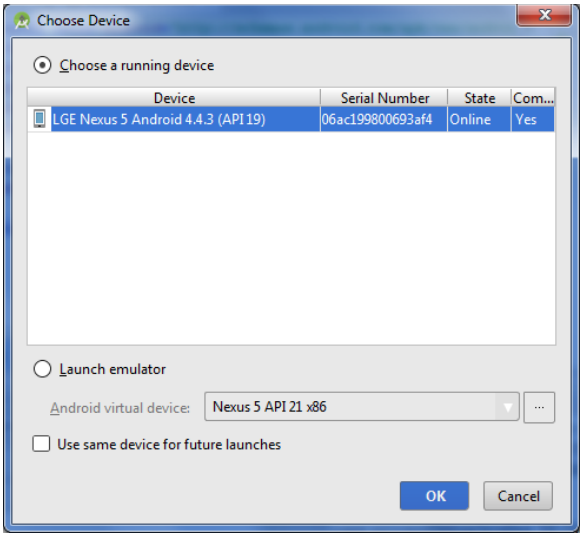

Figure 79 - Android Studio should prompt you for the target device.

Once the app has been installed it should auto-launch and you should see your activity with the words "Hello world!" on the top left hand corner of the screen. Congratulations! You successfully built your first app!

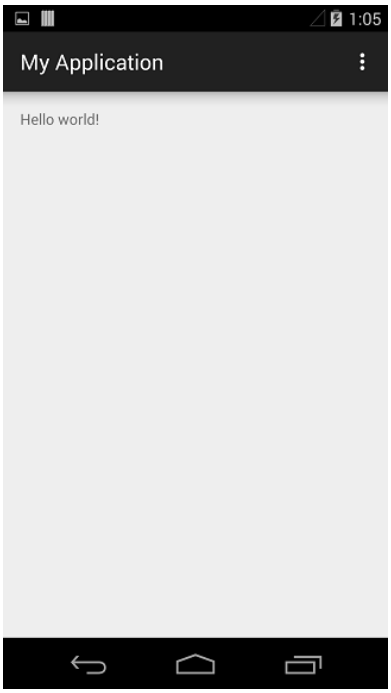

Figure 80 - Congratulations! You've just built your first app!

8 Using Your Robot

Now that you have successfully built your first Android app, let's focus on the driver station and robot controller.

8.1 FTC Driver Station

The driver station is the Android device that sits near the team drivers and is connected to one or two USB gamepads. Teams use the touch screen and the gamepads to operate their robot remotely. The driver station Android device runs a special app called the *FTC Driver Station*.

8.1.1 Installing the FTC Driver Station App

The FTC Driver Station app is located on Google Play. To install the app, you first need to connect the ZTE phone to an available wireless network that has access to the Internet. Launch the **Settings** activity on your phone, and select the **Wi-Fi** item to display the Wi-Fi activity.

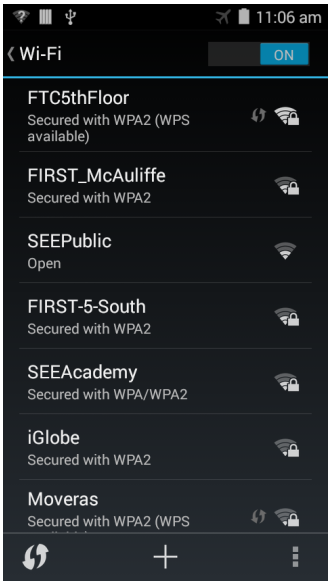

Figure 81 - Select your wireless network from the Wi-Fi activity.

Select your desired wireless network from the Wi-Fi activity and provide the password information required to access the wireless network.

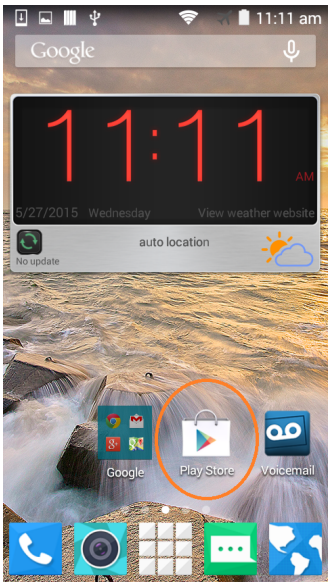

Figure 82 - Launch the Google Play Store app.

Once you have connected successfully to your wireless network, launch the Google **Play Store** app from your phone. The Play Store app might prompt you to either login to an existing Google account or create a new one. Follow the onscreen instructions to either create a new (free) account or login to your existing account.

Once you have successfully logged in to Google Play, click on the search icon (a little magnifying glass) and search for the phrase “FTC Driver Station”

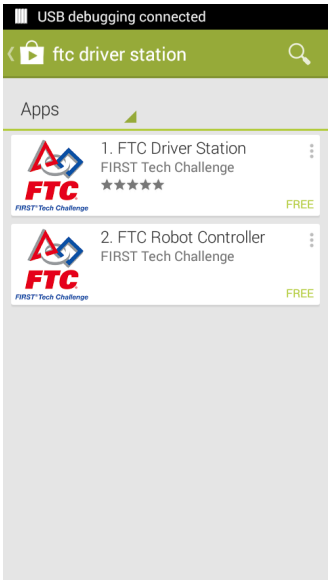

Figure 83 - Search for the phrase "FTC Driver Station"

Once you have found the FTC Driver Station app, click on it and follow the onscreen instructions to install. Note that the application might prompt you to enter a credit card number or some other

method of payment. The app is free and no payment method is required. You should be able to hit the "Skip" button to skip the process of providing a method of payment.

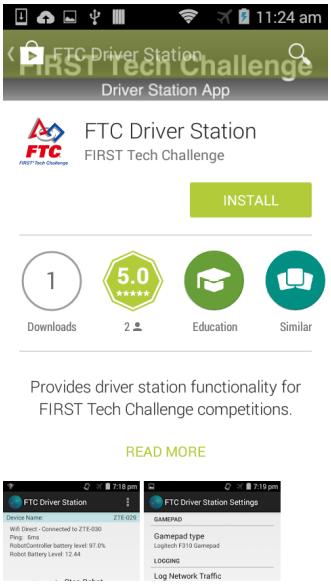

Figure 84 - Follow the on-screen instructions to install the app.

If you were able to install the app successfully, then there should be an FTC Driver station icon on your Android device.

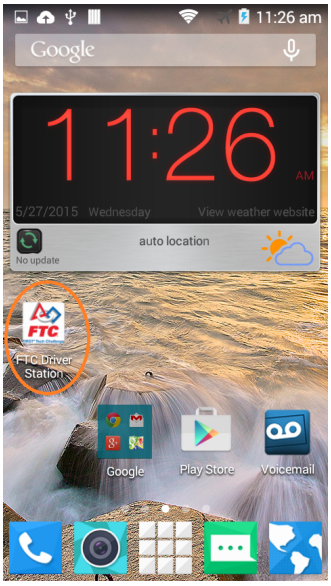

Figure 85 - After installing the app, you should see an icon on your Android's screen.

IMPORTANT NOTE: After you have successfully installed the app, go to your Wi-Fi settings menu and "forget" the wireless network that you used to connect to Google Play. In general, you do not want to be connected to any wireless networks with the exception of your FTC Robot Controller device.

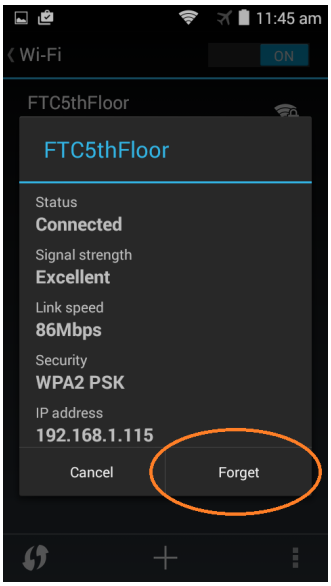

Figure 86 - Don't forget to forget the wireless network after the installation is complete!

IMPORTANT NOTE: You do not want to install the FTC Driver Station on the same Android device as the FTC Robot Controller app. These two apps should be installed on separate devices. One of the apps (the Robot Controller) will configure the device to operate as a WiFi Direct Group Owner. It is important that the two apps are installed and run on separate devices.

8.1.2 Launching the FTC Driver Station

Figure 87 depicts an Android tablet with a single Logitech F310 gamepad controller connected. This Android tablet has the FTC Driver Station app running. This app is used to communicate with and control the robot.

Figure 87 - Driver Station with a single gamepad connected.

To launch the FTC Driver Station app, simply touch its icon on the screen (see Figure 85).

The FTC Driver Station app will launch into its main screen (see Figure 88). Towards the top of the screen, you can see some information about the status of the wireless connection. Right now in Figure 88 the driver station is not connected to a robot. If the driver station were connected to a robot, you would see ping statistics (i.e., it would show the time it takes for a message to get sent to the robot and get acknowledged by the robot) as well as other status and wireless connection info.

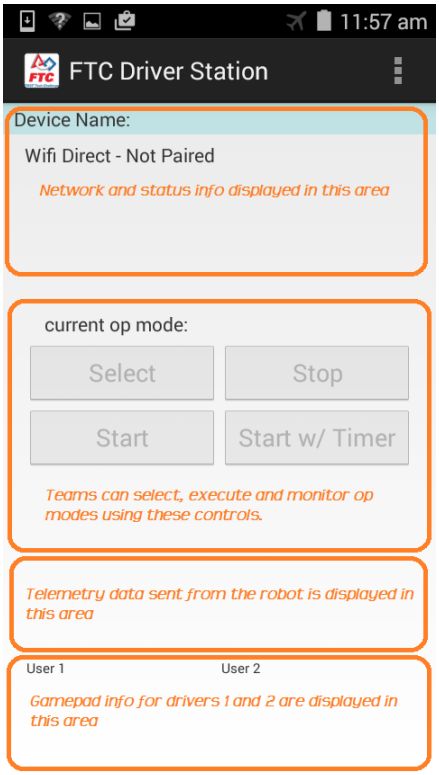

Figure 88 - Main screen for FTC Driver Station.

Towards the middle of the screen, there are buttons that you can use to select the *operational mode* or *op mode* of the robot. Op modes are pre-programmed robot behaviors (autonomous and driver controlled) that you can launch from the driver station. In Figure 88 the buttons are greyed out because the driver station has not yet connected to a robot.

Below the Op Mode controls is an area on the screen where telemetry data is displayed. The new platform has the ability to send information from the robot to the driver station. This information (such as servo position, motor power, sensor data, etc.) can be displayed on the driver station.

Towards the bottom of the screen, there is an area that is used to display information about the gamepads that are connected to the Android device. The left side of the area is used to display information about User 1 (i.e., driver #1). The right side of the area is used to display information about User 2 (i.e., driver #2).

8.1.3 Driver Station Menu

Towards the upper right corner of the main screen, there is a set of three vertical dots. Touch these dots to display a pop-up menu of FTC Driver Station selections.

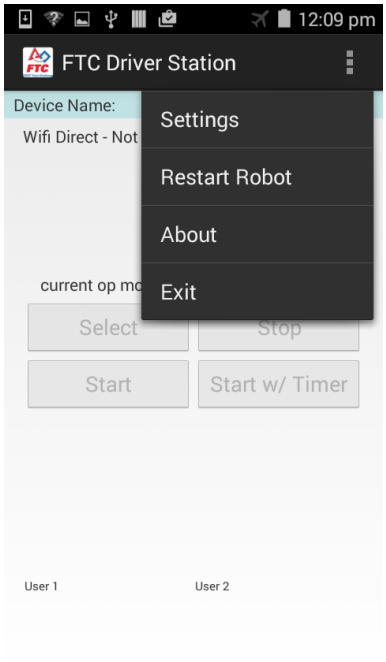

Figure 89 - Click on the dots in the upper right hand corner to display menu options.

8.1.3.1 Settings Screen

If you select the **Settings** the FTC Driver Station Settings screen will appear. This is the screen that you can use to modify the settings for your device.

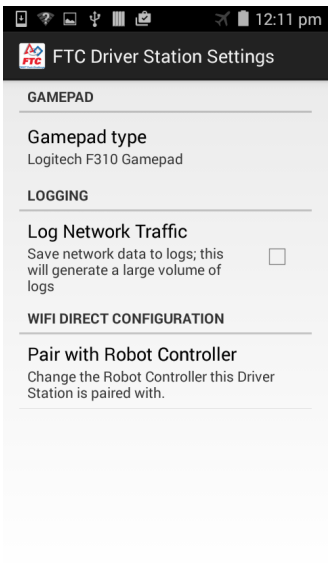

Figure 90 - The Settings screen.

Through the FTC Driver Station Settings menu, you can select which type of gamepad is connected to the device. Note that we recommend the use of the Logitech F310 gamepad controller with the device set in Xbox controller emulation mode (i.e., with the switch on the bottom set to "X").

Before you use your gamepad, make sure that the Logitech F310 is in Xbox emulation mode. This means that the little switch on the bottom of the controller should be set to the "X" position.

Figure 91 - Make sure the switch is in the "X" position.

The Settings menu also lets you select the option to "Log Network Traffic". This option can be useful if you are debugging your wireless connection (it logs information about packets sent to and from the robot controller), but this option can use up all of the storage on your Android device very quickly. We do not recommend that you use this option and you should never have this feature enabled during a match.

The Settings menu also has an option to pair the driver station to the robot controller. You need to connect or pair your driver station to a specific robot controller. We will review this pairing process in greater detail later on in this document.

8.1.3.2 Restart Robot

The FTC Driver Station menu has an option called **Restart Robot.** This option will do a remote soft restart of the robot controller. Sometime the robot controller might enter a stop mode. In the event that this happens, the driver can issue a remote restart command to the robot controller with this option. The Driver Station needs to be connected to a Robot Controller in order to restart the robot.

8.1.3.3 About Menu

Selecting the **About** menu option will bring up a screen which displays info about the app version and the build version of your FTC Driver Station app.

8.1.4 Using the Gamepads

As it was mentioned in a previous section, if you are using the Logitech F310 gamepad with your driver station, then make sure the switch on the bottom of the pad is set to the "X" position. The driver station app is designed to work with up to two gamepads.

When you connect your gamepads to the Android device (through a USB hub), you need to tell the FTC Driver Station app which gamepad will be used to represent driver #1 (User 1) and which gamepad will be used to represent driver #2 (User 2).

You can select which driver a gamepad will represent by pushing the **Start** key on the gamepad while simultaneously pressing the **A** button if you want to be driver #1, or the **B button** if you want to be driver #2.

Figure 92 - Press Start + A to be driver 1 or Start + B to be driver 2.

When you first connect your gamepads to the Android device, you must push a button combination to designate which driver your gamepad will represent.

Once you have designate which driver your gamepad will represent the gamepad status information will appear below the appropriate area near the bottom of the screen.

For example, supposed the gamepad that represents driver #1 is active, then the lower left hand corner of the screen will be highlighted green and gamepad status information (such as the x and y positions of the left and right joysticks, plus the button presses and the status of the trigger) will be displayed.

If the gamepad that represents driver #2 is active, then the lower right hand corner of the screen will be highlighted blue. The gamepad status information for that driver will be displayed on the lower right hand area of the screen.

Remember, before you can use a gamepad, you need to tell the Driver Station which driver that gamepad represents.

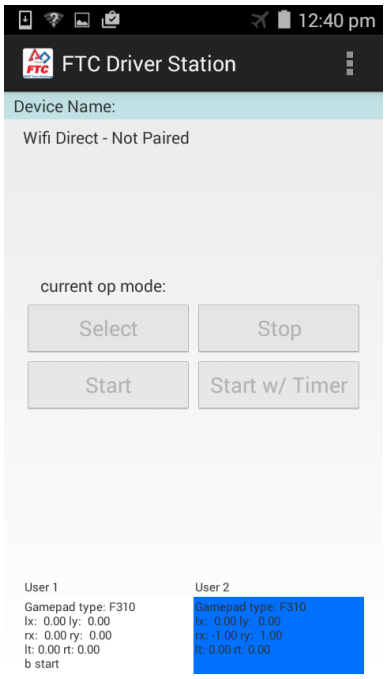

Figure 93 - Driver #1 gamepad info is on left, #2 is on right.

Note that for the joysticks on the gamepads, the values range from -1 to +1. For the x axis, -1 represents the far left range of motion and +1 represents the far right range. For the y axis -1 represents the topmost position and +1 represents the bottommost position.

8.1.5 Op Mode Controls

The center of the FTC Driver Station main screen contains controls that can be used to select an operation mode or op mode of a robot. Op modes are preprogrammed robot behaviors that you can select and execute from the driver station. The **Select** button will display a list of available op modes that can run on your robot. The **Start** button is used to start the selected op mode. The **Stop** button stops the current op mode. The **Start w/ Timer** button will start the op mode and let it run for a predetermined amount of time.

The buttons will not be enabled until the Driver Station is connected wirelessly to a robot controller. We will revisit the Op Mode controls in greater detail at a later section of this manual.

8.2 Robot Controller

The Robot Controller is the Android device that is mounted on the robot frame. It acts as the "brains" of the robot and runs a special app known as the FTC Robot Controller app. In this training manual you will be instructed on how to modify parts of the Robot Controller app to create your own op modes that will run on your robot. You will use a development kit to create, edit and register your own op modes for your Robot Controller app. However, you can also download a copy of the robot controller that includes some sample op modes pre-installed.

8.2.1 Installing the Robot Controller App

The FTC Robot Controller app is also available for download from the Google Play store. The instructions are almost identical to the instructions provided in section 8.1.1 of this document (see page 53). Connect to a wireless network with your phone and use Google Play to search for "FTC Robot Controller". Install the app with Google Play and don't forget to forget the wireless network that you had connected to in order to access the Internet.

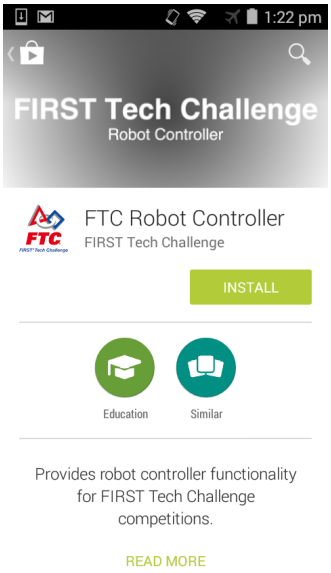

Figure 94 - The FTC Robot Controller is available on Google Play.

IMPORTANT NOTE: After you have successfully installed the app, go to your Wi-Fi settings menu and "forget" the wireless network that you used to connect to Google Play. In general, you do not want to be connected to any wireless networks with the exception of your FTC Driver Station device (or if you are doing wireless ADB, which is an advanced topic not covered in this document).

IMPORTANT NOTE: You do not want to install the FTC Driver Station on the same Android device as the FTC Robot Controller app. These two apps should be installed on separate devices. One of the apps (the Robot Controller) will configure the device to operate as a WiFi Direct Group Owner. It is important that the two apps are installed and run on separate devices.

8.2.2 Launching the Robot Controller App

After you install the app, there should be an FTC Robot Controller icon on your Android devices main screen. Touch the icon to launch the Robot Controller.

Figure 95 - After you install the app, an icon should be visible on your Android device.

The FTC Robot Controller app should also auto launch whenever it is plugged into the Power Module and connected to the USB electronic modules on the robot. When you first connect the Robot Controller Android device to the Power Module the Android operating system might prompt you on whether or not it's OK to launch the FTC Robot Controller app whenever this USB device is connected. You should check the "Use by default for this USB device" option and hit "OK". Note that the Android operating system might prompt you multiple times (once per connected USB module). You should check the "Use by default…" option and hit OK each time you are prompted. After you have configured the FTC Robot Controller to be the default app for the connected USB modules, this app should auto-launch whenever it detects any of the previously detected USB modules.

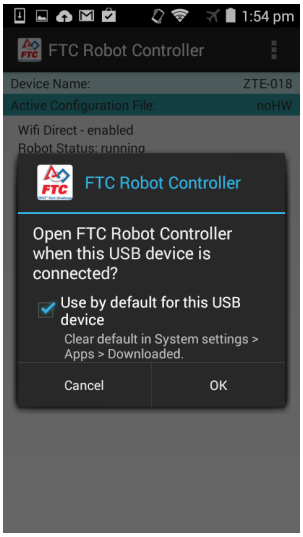

Figure 96 - Check the "Use by default..." Option and hit "OK"

8.2.3 Creating a Configuration File for the Robot Controller

When you first launch the Robot Controller app, it might complain to you because it doesn't have an active configuration file that it can use. The app needs to know what hardware devices are connected to the Robot Controller.

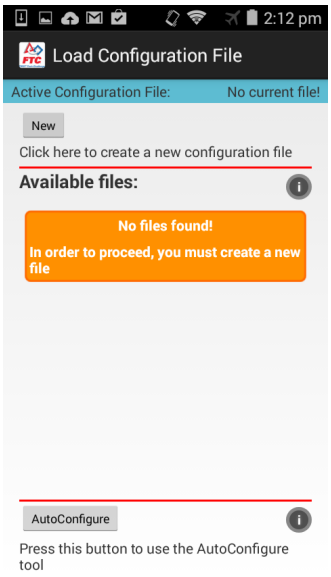

Figure 97 – The Robot Controller app might complain if it doesn't detect a configuration file.

In order to create a configuration file, if you are in the Active Configuration file screen of Figure 97 simply press the "New" button to create a new hardware configuration file for your robot.

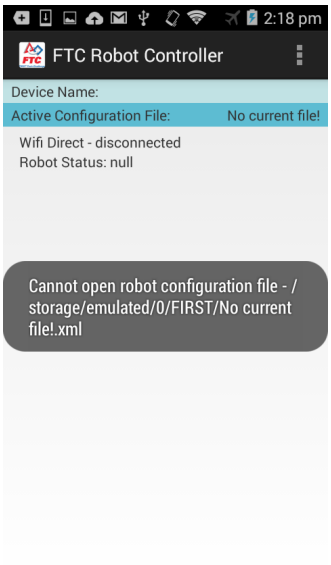

Figure 98 - Main screen of Robot Controller app (no hardware configuration file specified yet).

If you are in the main FTC Robot Controller screen (see Figure 98) you can create a new hardware configuration by first launching the Settings menu. Touch the three dots in the upper right hand corner of the main screen to bring up the **Settings** menu.

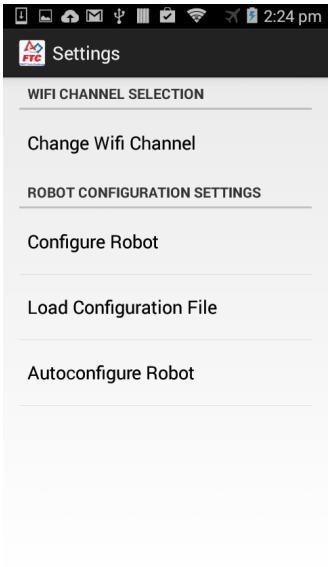

Figure 99 - Select "Configure Robot" from the settings menu.

Select the “Configure Robot” option from the settings menu to create a new hardware profile. This should launch the Configure Robot activity. We will use this activity to scan for attached hardware devices. Make sure your robot is powered on (the switch on the Power Module should be the main on/off switch for the robot) before you connect your Android device to the Power Module. Go to the Configure Robot screen and press the “Scan” button to scan for attached hardware devices.

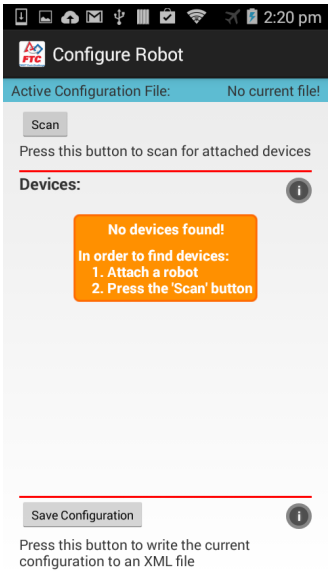

Figure 100 – The Configure Robot activity is used to scan for and configure attached devices.

After you have pushed the "Scan" button the app should detect any hardware modules connected to the device through the Power Module.

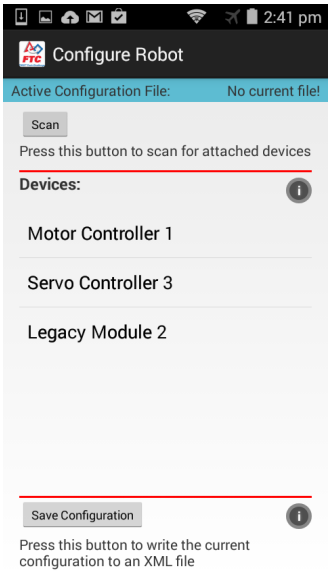

Figure 101 - Scan should auto detect the attached modules.

In Figure 101 the app detected three USB modules, a motor controller, a servo controller, and a legacy module. The app has the ability to auto detect the USB hardware modules, but you still need to configure each module and tell the robot controller what devices are connected to each module.

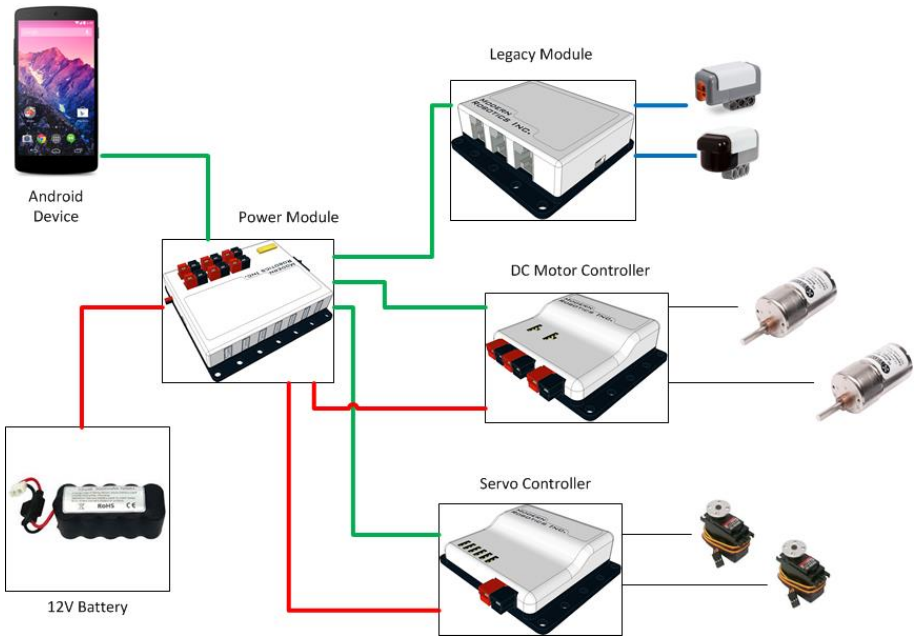

Figure 102 - This diagram shows how the devices are connected on the typical K9 bot.

Let's take a look at Motor Controller 1. For this training manual, we assume that Motor Controller 1 has two DC motors connected to ports 1 and 2 on the motor controller. If we touch Motor Controller 1 on the screen, a new screen will appear which will allow us to specify which motor ports are active and what names we will use to refer to each port.

In Figure 103 (see below) the motor controller has two motor channels active (1 and 2). The names of each channel are "motor_1" and "motor_2" respectively. Note that the names you assign for devices (including motors, servos, sensors, and even the hardware module itself) are important because later on when you want to write a custom "op mode" to control your robot, you will need to refer to these names within your program in order to access these devices on your robot.

Once you have configured the robot, click on the Done button to exit the Edit Motor Controller screen.

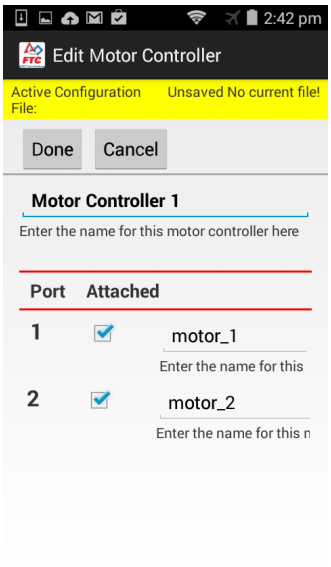

Figure 103 - Configuring the Motor Controller.

After you have configured the motor controller, you'll need to do the same for the servo controller and the legacy module. For this training manual the servo controller has two active channels (1 and 6) and the names of the servos are "servo_1" and "servo_6"respectively (see Figure 104).

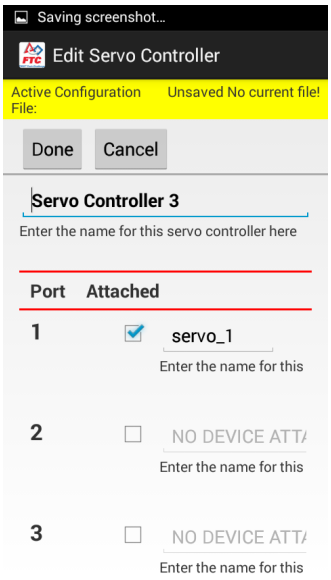

Figure 104 - Edit Servo Controller activity.

For this manual, the Legacy Module has an NXT-compatible Hitechnic IR Seeker V2 called "ir_seeker" connected to port #2 of the Legacy Module and an NXT reflected light sensor called "light_sensor" connected to port #3 of the Legacy Module (see Figure 105).

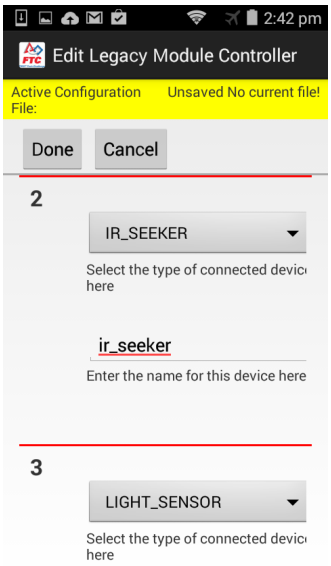

Figure 105 - Edit Legacy Module Controller activity.

Important note! *The names of the devices that you assign in the edit screens are very important. The Robot Controller app will use these names as references to the corresponding devices. When you create your own Op Mode to run on your robot, your Java code will have to refer to these device names in order to be able to find and control these devices with your Java code. In our example, if you want to be able to control the first motor configured in Figure 103 then your Java code will need to refer to it by its name ("motor_1").*

After you have provided the details for the motor controller, servo controller and legacy module, you are ready to save the hardware configuration to the Android device. To save your configuration, press the "Save Configuration" button on the Configure Robot screen and type in the name of your configuration file and hit the OK button.

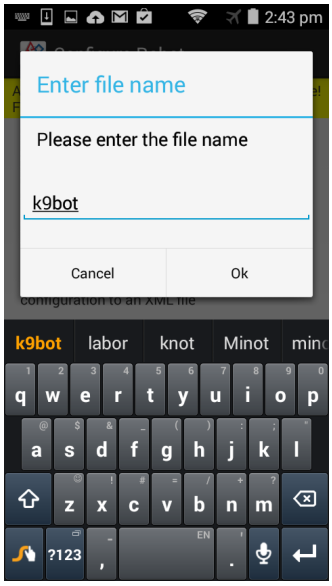

Figure 106 - Press the Save Configuration button and then specify the file name.

After you have saved your configuration file the Configure Robot screen should display the name of the current active configuration file in the upper right hand corner of the screen.

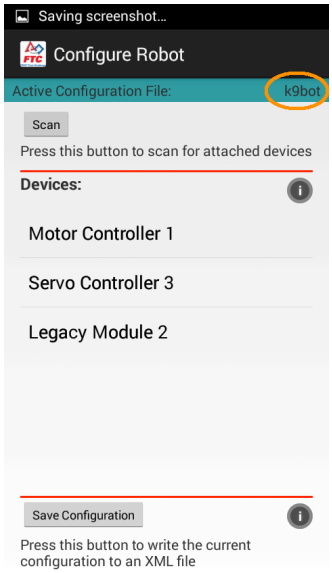

Figure 107 - The name of the active configuration file is displayed in the upper right hand corner.

Press the back arrow button on your Android device to return to the main FTC Robot Controller screen. The name of the active configuration file should now be visible in the upper right hand corner of the main screen.

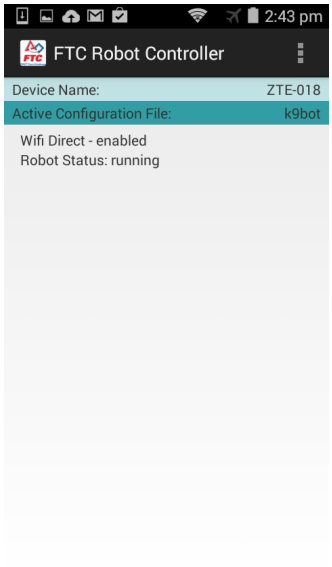

Figure 108 - The active configuration file is displayed in the upper right hand corner of the main screen.

8.2.3.1 Module Serial Numbers

Before we move on, it is important to note that each configuration file for a Robot Controller is matched to the specific hardware on the machine. Each USB enabled device on the robot has its own unique serial number. This unique serial number is stored in the configuration file. If you replace one of the USB-enabled devices (such as a Legacy Module) with another equivalent device, you will have to edit the configuration file for your robot since the serial number of the new device will not match the serial number of the original device.

Similarly, if you were to remove your Android device from one robot and attach it to a new robot, you would have to create a new configuration file since the serial numbers of the devices on the new robot would not match the serial numbers that were stored in the original configuration file.

8.2.4 Pairing the Driver Station to the Robot Controller

If your robot controller is properly configured to match the installed hardware on your robot, then you are almost ready to begin controlling your robot. Before you can control your robot from your driver station, you must first "pair" the driver station to the robot controller.

The new control system uses a technology known as WiFi Direct to establish a unique and persistent connection between the driver station and the robot controller. For this system, the robot controller acts as the WiFi Direct *group owner*. Once a driver station has been paired to a robot controller, then the driver station will always look for that robot, until it is paired with a different robot controller.

In order to pair your two devices, you need to make sure that both Android devices are powered on and the FTC Robot Controller app and the FTC Driver Station app are running on their respective machines. Your Driver Station should display its WiFi direct status. In the screen shots below the driver station and the robot controller are not yet paired together.

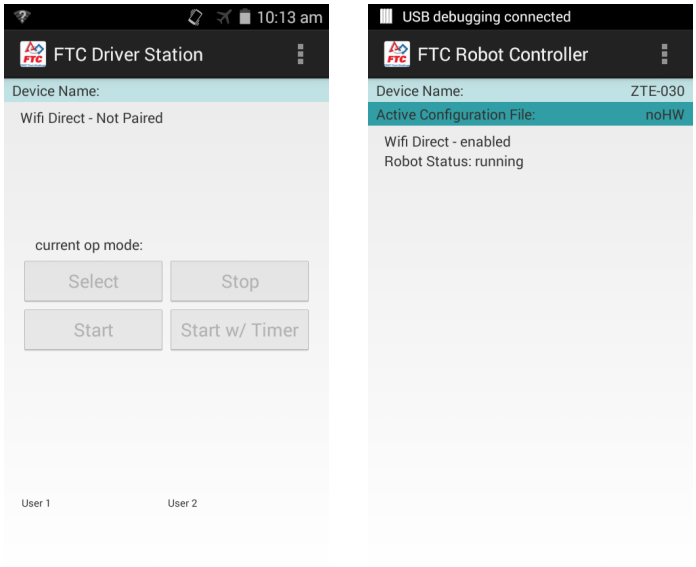

Figure 109 - The driver station and robot controller have not yet been paired together.

The pairing process is initiated through the driver station app. To start the pairing process touch the three vertical dots on the upper right hand corner of the FTC Driver Station app and then select "Settings" from the pop up menu.

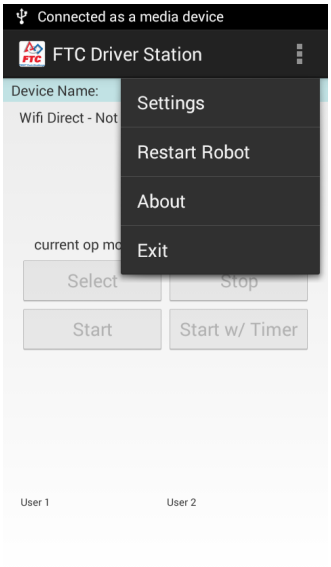

Figure 110 - Touch the three dots and select "Settings" from the pop-up menu.

Once you have launched the FTC Driver Station Settings menu, press the “Pair with Robot Controller” option to start the pairing process.

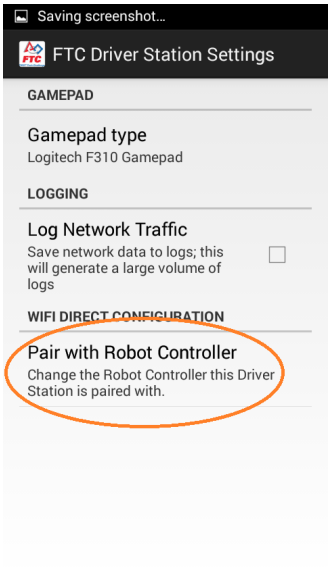

Figure 111 - Touch the "Pair with Robot Controller" option to start the pairing process.

From the driver station’s wifi direct pairing activity, look for your robot controller in the list of available devices. In the example below, the robot controller is named “ZTE-030”. Select the device and then use the Android back arrow to make a connection request from your driver station to the robot controller.

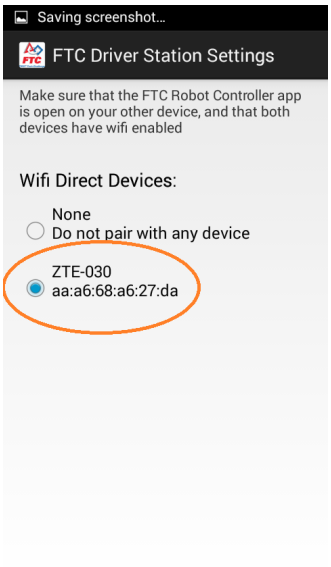

Figure 112 - Look for your robot controller in the list of available devices.

If this is the first time you are pairing to the robot controller, the robot controller device might display a prompt asking you if it is OK to allow an Android device to establish a WiFi Direct connection. In the example below, the driver station ("ZTE-029") has requested a connection with the robot controller ("ZTE-030"). Click on "Accept" to accept the connection request.

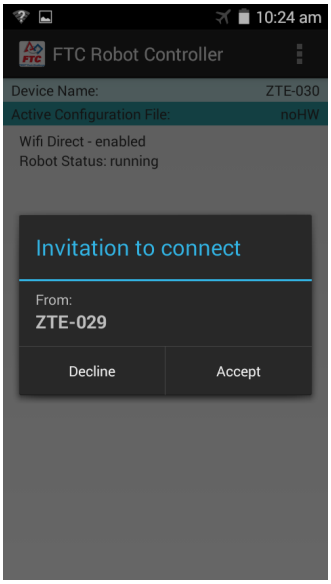

Figure 113 - Click Accept to accept the WiFi Direct connection request.

Once you have successfully paired your device to the robot controller, you should see a WiFi Direct message on the driver station main screen indicating that the driver station is now connected to a robot controller.

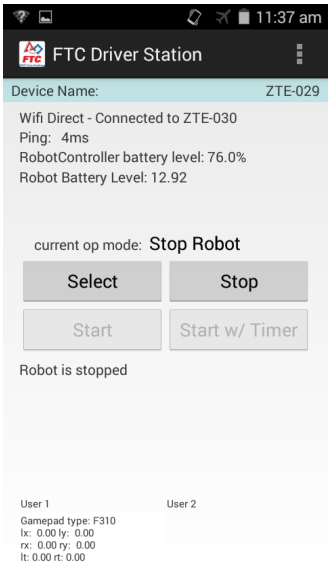

Figure 114 – This driver station is now connected to a robot controller named "ZTE-030".

Once your driver station is paired with its robot controller you are ready to begin executing op modes!

8.3 Selecting and Running Op Modes

Operational Modes or *Op Modes* are program modules that you can execute to have your robot perform specific behaviors. The downloaded Android apps that you installed should include some pre-installed op modes that you can select and run. If your robot has been successfully paired to the driver station, and if the robot is running (i.e., Legacy Module is powered on and connected), then the Op Mode buttons on the FTC Driver Station app should now be enabled.

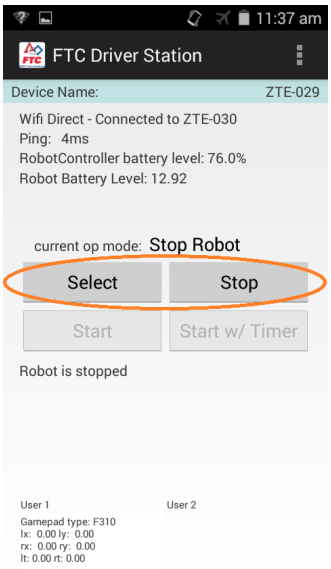

Figure 115 - Op Mode Select and Stop buttons are enabled. Robot is in Stop mode.

Note that if the Driver Station app is connected successfully to the Robot Controller app, then some ping statistics should be visible at the top left hand corner of the Driver Station screen. In the example of Figure 115 the ping statistics reads "4 ms". This means it took 4 milliseconds for the driver station to send a packet to the robot and for the robot to acknowledge the packet and send its acknowledgement back to the driver station.

The ping round trip time is a rough measure of network quality. As the round trip time increases, the network performance usually degrades. In our testing, an ideal round trip time is anything on the order of 10 milliseconds or lower. Sometimes you might see the ping time period increase if there is a surge in network traffic or interference. If the period starts to increase to the order of or greater than 100 milliseconds, then there might be excessive network traffic or noise on the wireless channel that your system is using.

Touch the **Select** button to display a list of the available Op Modes for this robot. You can select an Op Mode from the list and the FTC Driver Station should indicate that the Op Mode has been queued for execution. For our demonstration, there should be an op mode called *K9TeleOp* in the list. Select the mode from the list.

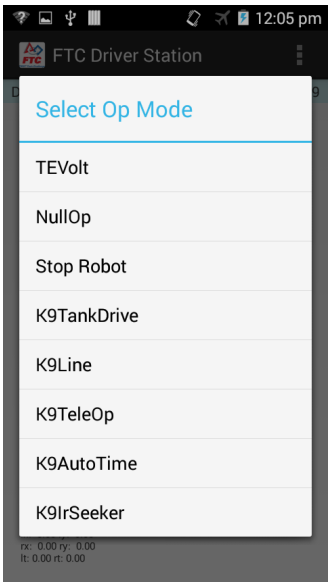

Figure 116 - Select an Op Mode to run from the list.

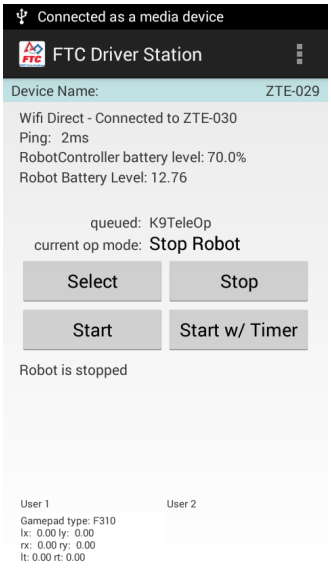

Figure 117 - Once an Op Mode has been selected, it will appear as "queued" and the Start buttons will be enabled.

Once you have selected an op mode from the list, the FTC Driver Station screen should change. The name of the selected op mode should appear next to the word “queued:” towards the top of the screen. Also, the **Start** and **Start w/ Timer** buttons should be enabled.

If you press the **Start** button, then the queued op mode will run indefinitely until the program terminates or until the **Stop** button is pressed. If you press the **Start w/ Timer** button, then the queued op mode will run for a preset amount of time (currently 30 seconds).

In our example, if you have selected the K9TeleOp program, press the **Start** button to run a simple driver-controlled program.

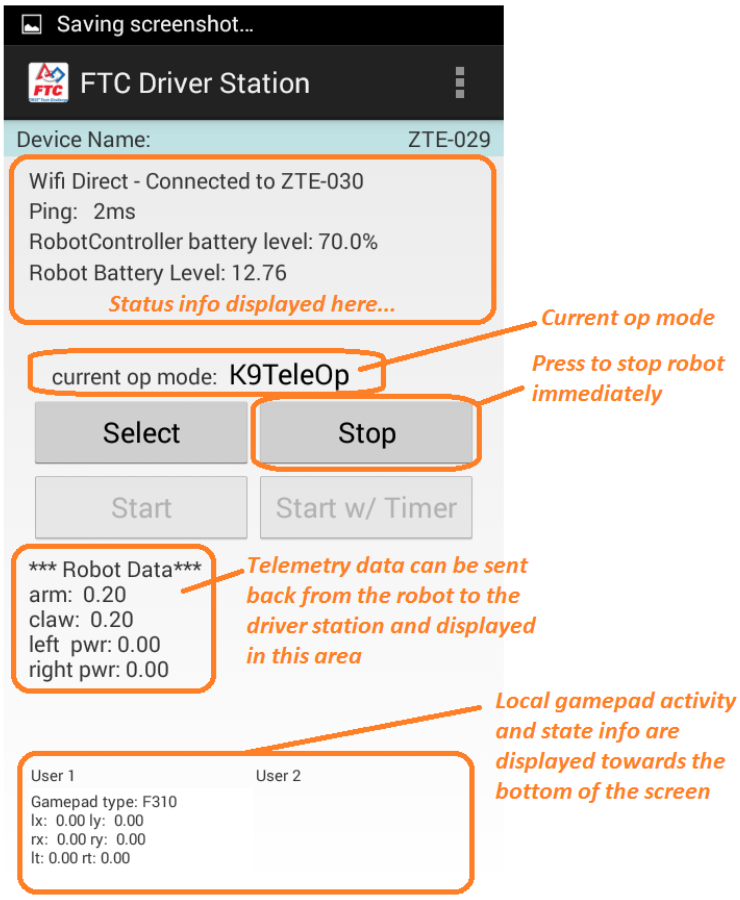

Figure 118 - The screen displays useful info during an op mode run.

The driver station screen displays useful information during an op mode run. The WiFi connection status info (ping time) is continuously updated at the top of the screen. The name of the current running op mode is displayed below the ping information.

The **Select** and **Stop** mode buttons are still enabled. You can hit the **Stop** button to force the robot to stop immediately.

Underneath the Op Mode controls, there is an area reserved for telemetry data from the robot. The software development kit that has been developed by QualComm has a feature that lets you send data (any kind of data, battery voltage levels, sensor data, motor speed and position, time, etc.) back from

the robot to the driver station. This information is displayed in the area below the Op Mode controls on the Driver Station app.

This telemetry feature is useful for getting remote information from the robot. We will learn how to send telemetry data to the driver station when we write our own op mode program modules.

At the bottom of the screen gamepad activity and state information are displayed. Remember, you have to hit the "Start" button and the "A" key on the gamepad to assume the role of User 1 or the "Start" button and the "B" key to assume the role of User 2.

If you were able to run *K9TeleOp* properly, drive the robot around for a bit and play with the controls. K9TeleOp only uses one gamepad (User 1). The left joystick on the gamepad controls the drive train. The yellow "Y" button raises the manipulator arm. The green "A" button lowers the arm. The blue "X" button closes the gripper. The red "B" button opens the gripper.

Once you have had a chance to drive the robot around and test the controls, hit the "Stop" button to stop the op mode run.

Running a timed op mode is very similar to running a regular op mode. However, instead of pushing the **Start** button, you simply press the **Start w/ Timer** button. The op mode will begin running and a countdown timer will appear next to the word "Stop" on the stop button. When the timer gets to zero, the driver station sends a stop command to the robot controller.

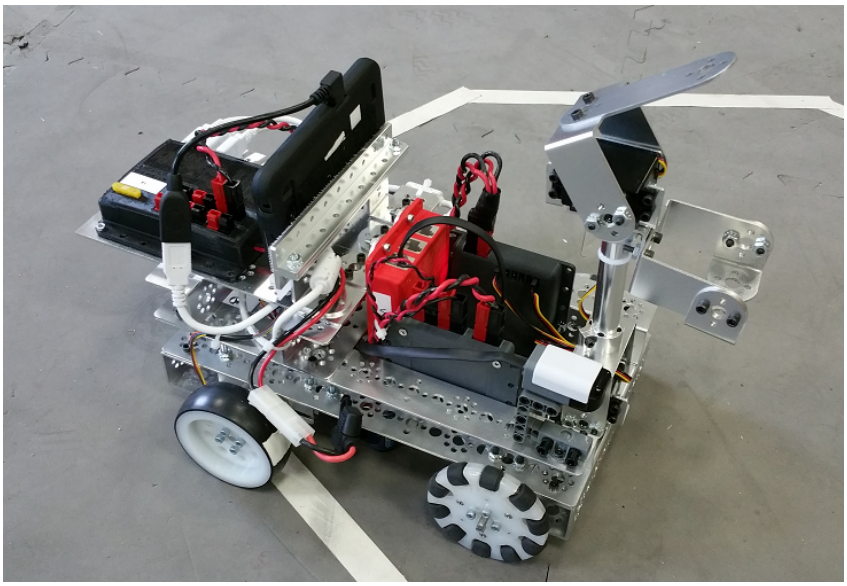

Figure 119 - Run the K9TeleOp op mode to drive your test bot.

9 The QualComm FTC Robot SDK

Now that you've seen how to use the Robot Controller app, let's learn about how the app is made. The Robot Controller app was built using a *software development kit* or *SDK* that was created by QualComm specifically for the *FIRST* Tech Challenge competition. The QualComm SDK includes the libraries that are needed to communicate with the robot hardware (such as the Modern Robotics Legacy Module, or the DC Motor and Servo controllers). The SDK also includes the source code and resource files that define the look and behavior of the Robot Controller app.

In order to write op modes that will define custom behavior for your robot, you will need to install a copy of the QualComm SDK onto your computer and learn how to modify the op mode program modules.

9.1 Accessing the SDK on GitHub

The FTC is available for download on GitHub. The FTC SDK repository is currently located at the following URL,

https://github.com/ftctechnh/ftc_app

If you have a GitHub account you and you know how to use git (a version control tool) you can clone the repository to your local computer as you would do for any other GitHub repository.

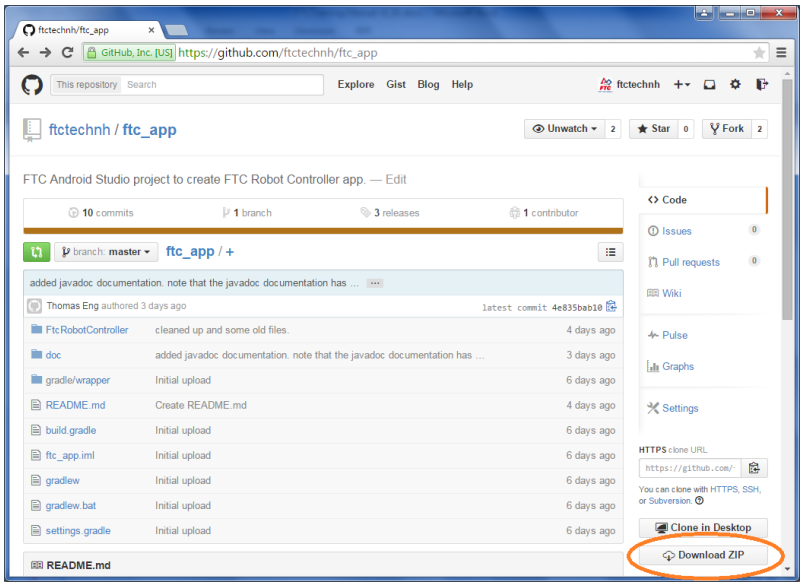

Figure 120 - SDK is available on GitHub. Press "Download ZIP" to download the project folder as a .ZIP file.

If you are not familiar with how to use git, you can also download the project folder as a compressed file (.zip). To download the project folder as a compressed archive, look towards the lower right hand side of the repository's webpage for a button labeled "Download ZIP". This will start the download process. Once the file has been downloaded, locate the downloaded file and move it to its desired folder on your computer.

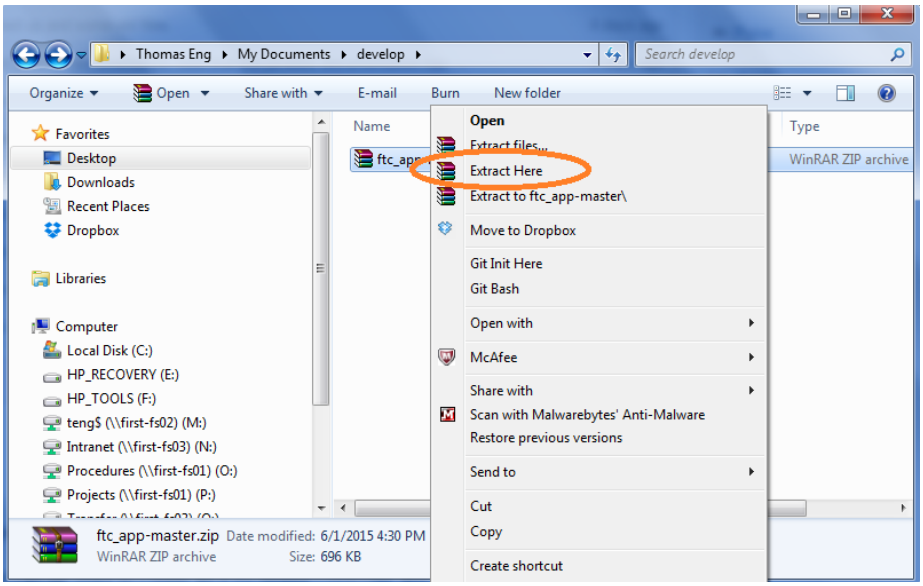

Figure 121 - Follow your computer's instructions for extracting the contents of the .ZIP file.

Extract the contents of the .ZIP file so that you have the uncompressed project folder.

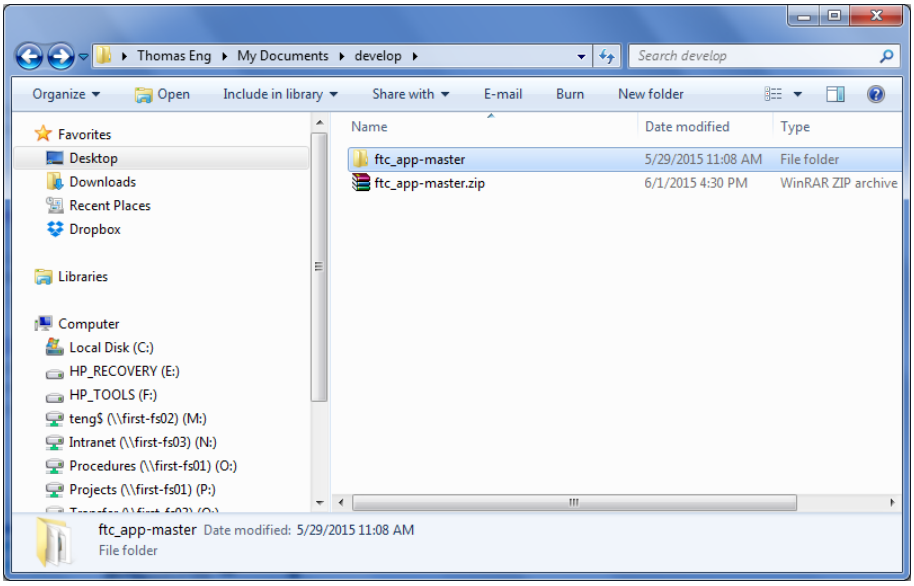

Figure 122 – The .ZIP contains a project folder ("ftc_app-master") that can be imported into Android Studio.

Once you have extracted the contents of the .ZIP file, you can import the project folder into Android Studio. If your Android Studio environment is currently opened to another project, select **File->Close Project** to close the current project.

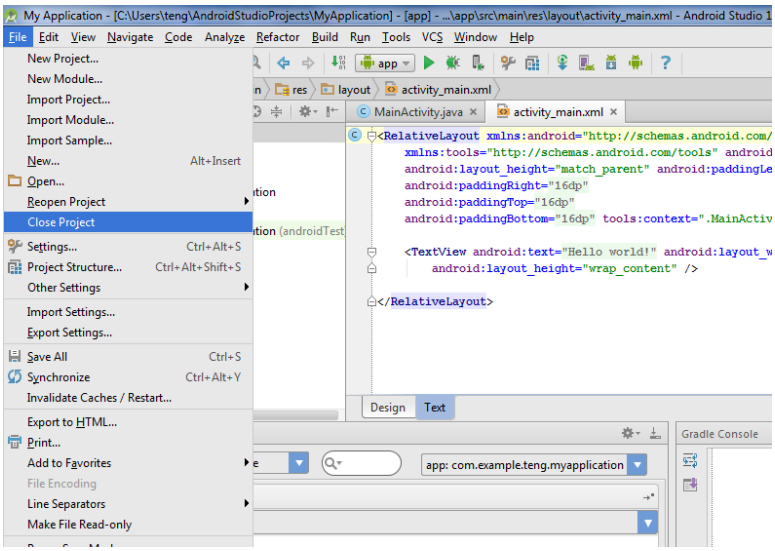

Figure 123 - Select File->Close Project to close the current project.

After you have closed the current project, you should see the Android Studio Welcome screen. Select the option **Import project (Eclipse ADT, Gradle, etc.)** from the Quick Start menu on the Welcome screen. Important: make sure you select the "Import project (Eclipse ADT, Gradle, etc.)" option (and not the "Open an Existing Android Studio project" option).

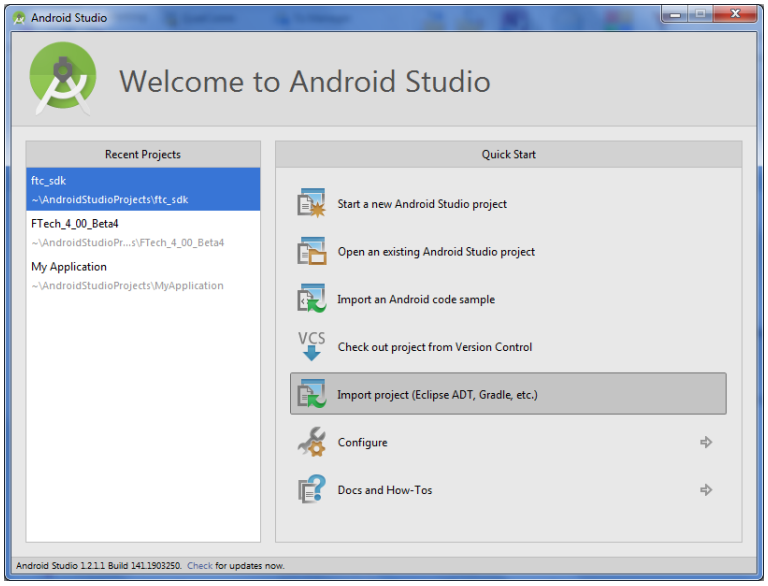

Figure 124 - Select Import Non-Android Studio project from the Quick Start menu.

It might take Android Studio a while for it to generate a directory structure that you can browse to look for your project file. Browse the directory to look for the QualComm SDK and select it then click on the **OK** button to import the project into Android Studio.

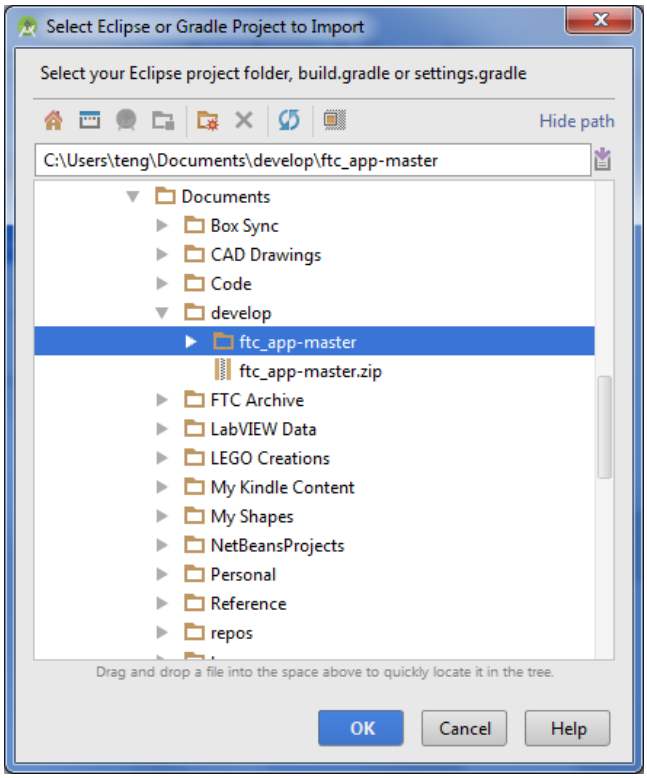

Figure 125 - Find the QualComm SDK project folder and select it. Hit OK to begin importing the project.

Android Studio will display a progress bar and begin to import the QualComm SDK project folder.

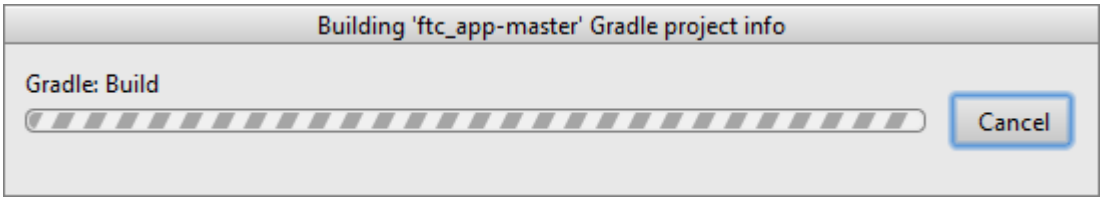

Figure 126 - Android Studio will begin to import the project.

Once you Android Studio has completed the import process your Android Studio window might look like the following screenshot (see Figure 127). Click on the “Project” tab on the left side of the Android Studio window to see the project folder view.

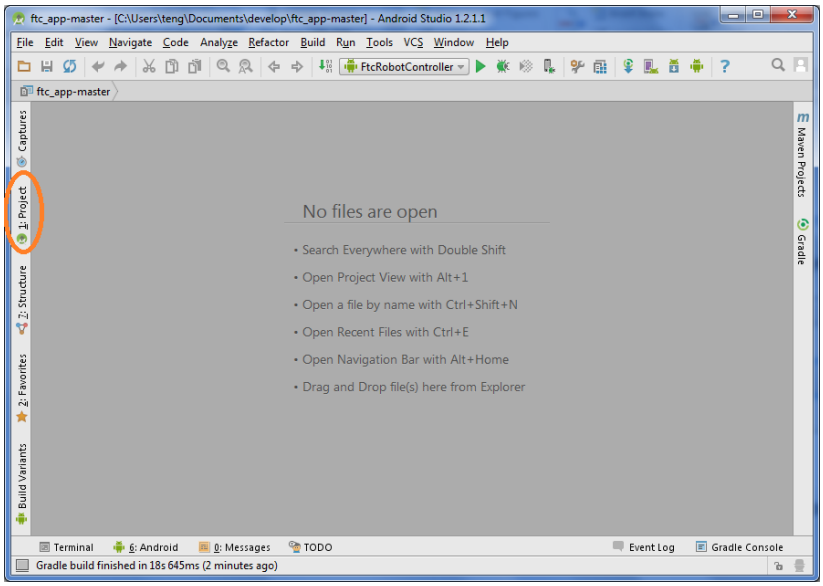

Figure 127 - Click on the "Project" tab to view the project folder in Android Studio.

In the left hand pane of the Android Studio window, you should see the FtcRobotController Project folder. You can click on the folder to expand it and see its subfolders and the project structure.

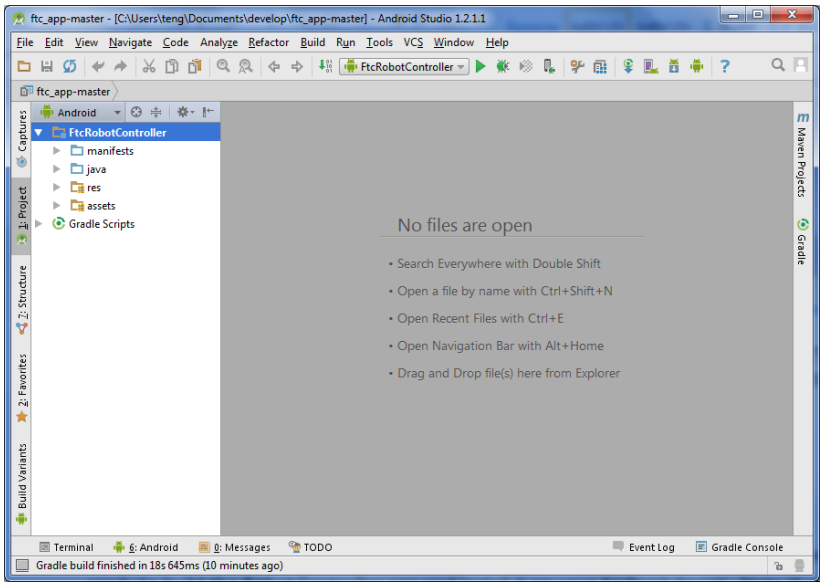

Figure 128 - Expand the FtcRobotController to view the project structure.

9.2 Building the Robot Controller App

If you were able to import the project successfully into your Android Studio environment, then you are ready to build the Robot Controller app and install it on your Android device. Note that you only need to modify the Robot Controller app to program custom behaviors for your robot. The Driver Station app typically does not change and teams will download it from Google Play onto their driver station.

Before you build the Robot Controller app, make sure your ZTE Speed phone is in USB Debugging mode. Connect the ZTE Speed to the computer with a USB cable. If the ZTE Speed prompts you to allow USB Debugging hit **OK** to allow it.

After you have selected the FtcRobotController configuration, select **Run->Run 'FtcRobotController'** from the menu at the top of the screen. This will initiate the build process.

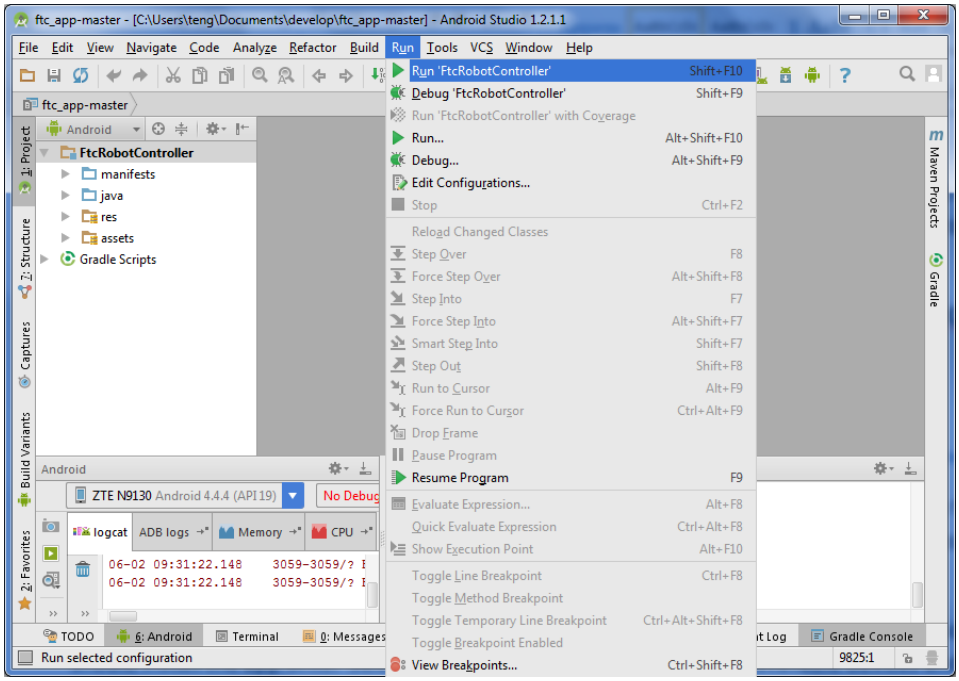

Figure 129 - Select Run->Run 'FtcRobotController' to compile the FtcRobotController folder.

While Android Sturdio is building the app for you, you can click on the **Gradle Console** tab towards the lower right hand side of the screen to see the status messages from the gradle software as it builds and links the modules needed for the app.

When Android Studio successfully completes the build, it should prompt you to select which device it should install the app onto. Select the ZTE Speed phone from the list of available devices and hit **OK**.

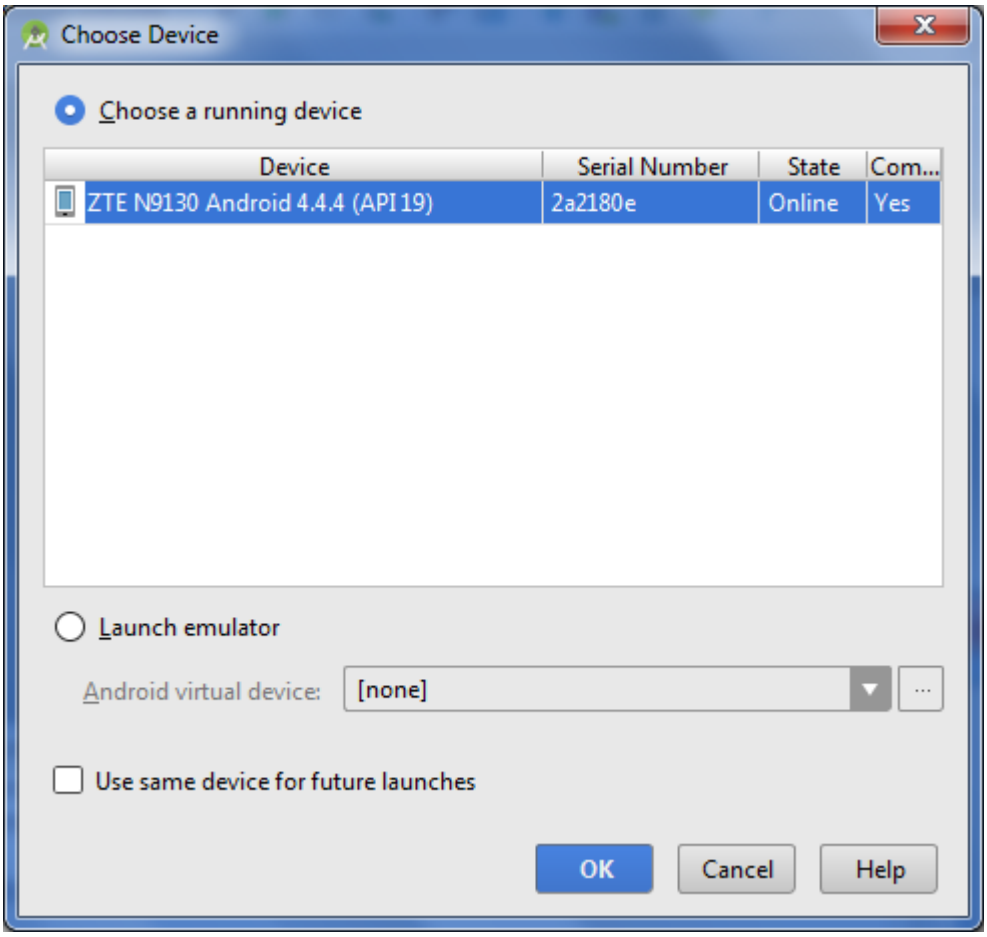

Figure 130 - When Android Studio is done building the app it will ask you to select a target device for the app.

After you hit the **OK** button Android Studio will attempt to install the FtcRobotController app onto the ZTE Speed phone. When this happens, you might get a warning message from Android Studio, indicating that it detected that the device already has an application with the same package, but different signature. If you encounter this message, simply press **OK** to allow Android Studio to uninstall the old app from the phone and to replace it with your newly compiled version of the app.

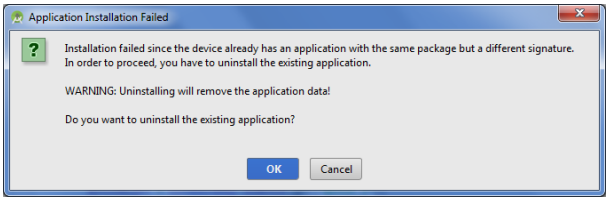

Figure 131 - You might get a warning indicating from Android Studio that the target device already has a copy of the application installed, but with a different signature. Hit OK to uninstall the app and replace it with your copy of the app.

Once Android Studio has successfully installed the app onto your phone, the Gradle Console should display the words "BUILD SUCCESSFUL" indicating that the app was built and installed onto the target device. Congratulations, you have just built your first FTC-related app!

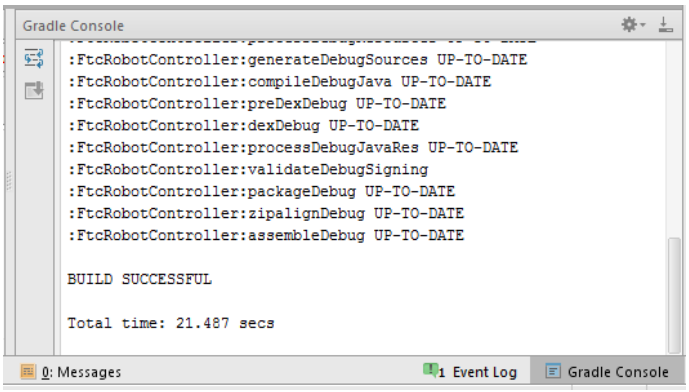

Figure 132 - Once the build is complete and the app installed, the Gradle Console will indicate success.

Once you have built the FTC Robot Controller app and installed it on your phone, you can configure it and operate it like you did with the pre-installed copy of the app.

9.3 Building Your Own Op Mode

Now that you have learned how to compile the robot controller app, let's take a look at how to make our own op mode. As we mentioned earlier in this document, an op mode is a software module stored on the robot controller that you can execute from the driver station. These op mode software modules contain pre-programmed behaviors for your robot.

Op modes are useful because you can create a variety of them that will allow your robot to behave in different ways, depending on the competition requirements. For example, you might have one op mode that you use for the autonomous portion of a match and another for the driver-controlled portion. You might also have multiple op modes, for different tasks during the autonomous period.

The new robot control platform lets you create multiple op modes that can be selected and executed remotely from the driver station. Let's take a look at how the op modes are organized in the FtcRobotController program folder in your QualComm SDK Android Studio project.

9.3.1 FtcRobotController *opmodes* Folder

The source code (i.e., the Java instructions that are used to create the program module) for the op modes are stored in a subfolder of the FtcRobotController program folder. If you use the project browser on the left hand side of your Android Studio development environment, you can browse the FtcRobotController folder and navigate to the following subdirectory,

FtcRobotController->java->com.qualcomm.ftcrobotcontroller->opmodes

This folder contains all of the files for the robot controller app's op modes.

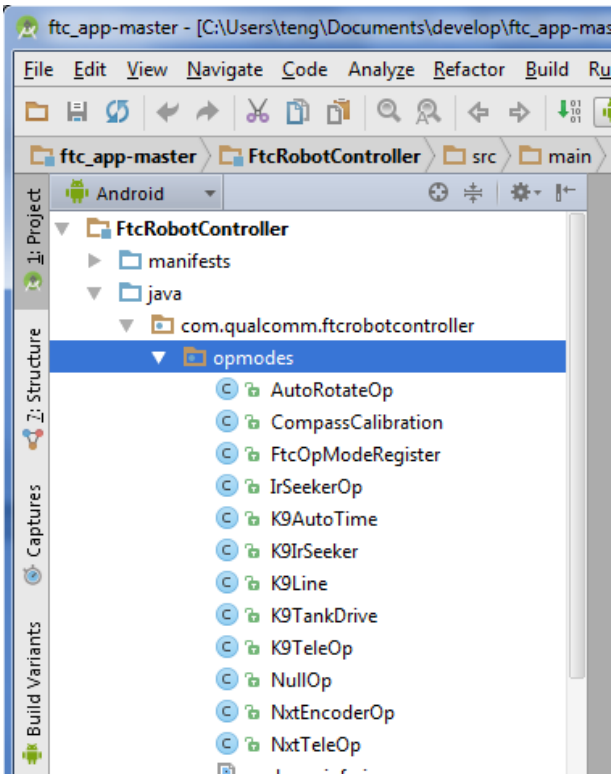

Figure 133 - The FtcRobotController->java->com.qualcomm.ftcrobotcontroller->opmodes folder contains the op mode files.

9.3.2 FtcOpModeRegister.java File

Inside the *opmodes* folder, there is a special file called "FtcOpModeRegister.java". Note that when you browse the *opmodes* folder in Android Studio, you will see the word "FtcOpModeRegister" and not "FtcOpModeRegister.java". The ".java" suffix is not displayed in the Project Browser.

Double click on the FtcOpModeRegister file in the project browser window. The contents of the file should appear in the main preview/text view of the screen.

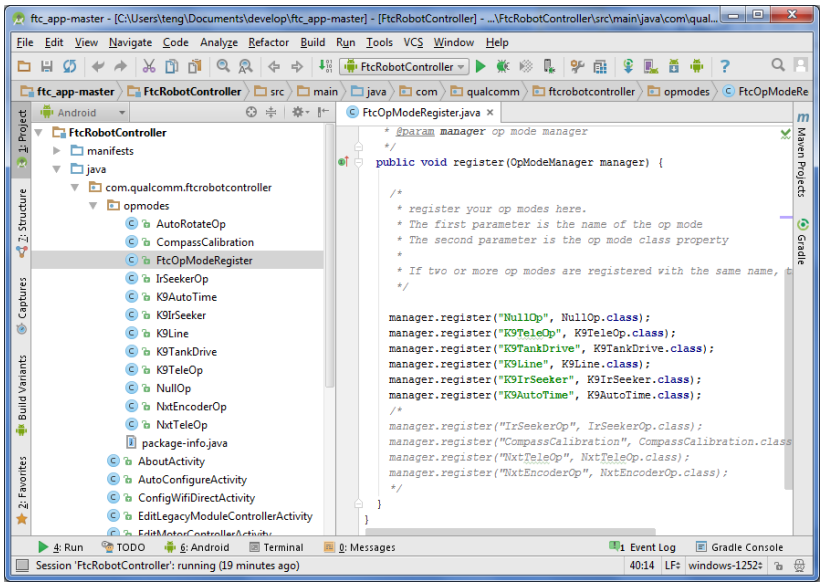

Figure 134 - FtcOpModeRegister contains a list of op modes that the robot controller will share with the driver station.

Let's take a close look at the FtcOpModeRegister file. If you look at the FtcOpModRegister Java code you will see the following lines of code,

```
manager.register("NullOp", NullOp.class);
manager.register("K9TeleOp", K9TeleOp.class);
manager.register("K9TankDrive", K9TankDrive.class);
manager.register("K9Line", K9Line.class);
manager.register("K9IrSeeker", K9IrSeeker.class);
manager.register("K9AutoTime", K9AutoTime.class);
```

Figure 135 - This code adds NullOp, K9TeleOp, and other op modes to the register.

What this Java code is doing is creating a register (i.e., a list) of op modes that we would like to make available to the driver station for selection and execution. In the example code of Figure 135 there are six op modes that are being registered. These op modes ("NullOp", "K9TeleOp", "K9TankDrive", "K9Line", "K9IrSeeker", and "K9AutoTime") will all be available to select from the list of op modes on the FTC Driver Station.

If you would like to make an op mode available to the driver station, you must first register it here. The following statement registers an op mode K9TeleOp and gives it a name of "K9TeleOp" for the registry.

Manager.register("K9TeleOp", K9TeleOp.class);

9.3.3 Op Mode Life Cycle

Let's take a look at what the source code file for a typical op mode looks like.

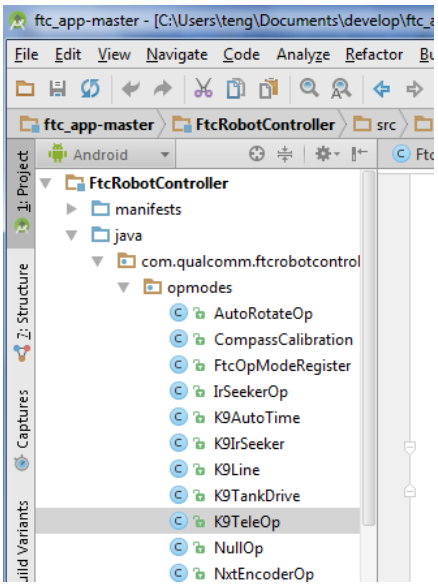

Figure 136 - Find the K9TeleOp op mode in the project browser.

In the left hand side of your Android Studio environment, use the project browser to find the file "K9TeleOp" and double click on the file to open it up:

K9TeleOp.java

```
/.../

package com.qualcomm.ftcrobotcontroller.opmodes;

import ...

/**
 * TeleOp Mode
 * <p>
 * Enables control of the robot via the gamepad
 */
public class K9TeleOp extends OpMode {

    /*
     * Note: the configuration of the servos is such that
     * as the arm servo approaches 0, the arm position moves up (away from the floor).
     * Also, as the claw servo approaches 0, the claw opens up (drops the game element).
     */
    // TETRIX VALUES.
    final static double ARM_MIN_RANGE  = 0.20;
    final static double ARM_MAX_RANGE  = 0.90;
    final static double CLAW_MIN_RANGE  = 0.20;
    final static double CLAW_MAX_RANGE  = 0.7;

    // position of the arm servo.
    double armPosition;

    // amount to change the arm servo position.
    double armDelta = 0.1;

    // position of the claw servo
    double clawPosition;

    // amount to change the claw servo position by
    double clawDelta = 0.1;
```

Figure 137 - The K9TeleOp source code.

If you browse the source code, you will discover that the K9TeleOp is a Java class that reads gamepad input that was sent to the robot controller from the driver station. The op mode uses this gamepad input to control the DC and servo motors on the robot.

If you take a look at the structure of the K9TeleOp code, you will see that there are three functions or *methods* that are defined for the class. These methods are named ***start***, ***loop*** and ***stop***.

Every op mode that you will create, should have its own *start*, *loop* and *stop* methods in its class definition. These three methods correspond to the three distinct stages of an op mode's life cycle:

1. `start()` – When an op mode is executed by a driver (i.e., when the driver pushes the "Start" button on the touch screen), the `start()` method for the selected op mode gets triggered. If you have any initialization tasks that you would like to do for your robot, you can place the initialize code into the `start()`method's body.
2. `loop()` – When a driver pushes the **Start** button on the driver station, the code that is written in the op mode's `loop()` method will be executed regularly (approximately every 25 milliseconds). The robot controller app has a built in *event loop* that executes the contents of the `loop()` method repeatedly, until a stop command is received from the driver station (or unless an emergency stop condition occurs). This method is where you will put the bulk of your robot code for an op mode.
3. `stop()` – When the robot controller receives a stop command from the driver station or when an emergency stop() condition occurs, the code in the `stop()` method gets executed. If you have any cleaning up to do after an op mode run has been completed, this is the place to put the cleanup code for your op mode.

Note that the programming model that is described in this section is different from the linear programming model that is used to program a LEGO NXT with a tool like RobotC. The op modes that are described in this section utilize an *event driven* programming model rather than a *linear* programming model. With a linear programming model, commands are executed sequentially one after the other. The programming model used by the op modes is a little different. With the event driven programming model, the loop() method will get executed repeatedly, until the op mode run is terminated. This programming model is similar to how a device like an Arduino is programmed.

9.3.3.1 The start() Method

Let's take a look at the `start()` method of the K9TeleOp op mode (see Figure 138).

```
86      @Override
87      public void start() {
88
89
90          /*
91           * Use the hardwareMap to get the dc motors and servos by name. Note
92           * that the names of the devices must match the names used when you
93           * configured your robot and created the configuration file.
94           */
95
96          /*
97           * For the demo Tetrix K9 bot we assume the following,
98           *   There are two motors "motor_1" and "motor_2"
99           *   "motor_1" is on the right side of the bot.
100          *   "motor_2" is on the left side of the bot and reversed.
101          *
102          * We also assume that there are two servos "servo_1" and "servo_6"
103          *    "servo_1" controls the arm joint of the manipulator.
104          *    "servo_6" controls the claw joint of the manipulator.
105          */
106         motorRight = hardwareMap.dcMotor.get("motor_2");
107         motorLeft = hardwareMap.dcMotor.get("motor_1");
108         motorLeft.setDirection(DcMotor.Direction.REVERSE);
109
110         arm = hardwareMap.servo.get("servo_1");
111         claw = hardwareMap.servo.get("servo_6");
112
113         // assign the starting position of the wrist and claw
114         armPosition = 0.2;
115         clawPosition = 0.2;
116     }
```

Figure 138 - Any code placed in the start() method gets executed when an op mode is first enabled.

We can see that when the `start()` method is executed, the robot controller will look for a *dcMotor* called "motor_2" and assign it to a variable called *motorRight*. We also see that the robot controller will look for another *dcMotor* called "motor_1" and assign it to a variable called *motorLeft*.

Do the names "motor_1" and "motor_2" look familiar to you? They should since these are the names that were used when we created a configuration file for our robot (see section 8.2.3 of this manual).

Here's an important concept to remember, when you configure hardware for your robot using the FTC Robot Controller app, the names you assign to the hardware devices will be used to reference these devices in your op mode code. For example, if you configured your robot and named a servo motor "servo_gripper" than in your op mode program, when you want to find and modify this servo, you will have to use the name "servo_gripper" as a reference to the device. If you accidentally told the robot controller to look for a servo named "servo_grabber" then the controller would give you an error, because its memory map will only have a reference to a servo named "servo_gripper".

In `start()` the method of the K9TeleOp op mode (see Figure 138) there is also some code to get a reference to the two servos on the K9 bot.

9.3.3.2 The loop() Method

Let's take a brief look at the `loop()` method (see Appendix A). The `loop()` method contains the source code that gets executed by the robot controller repeatedly, until a stop command is received. At

the beginning of the `loop()` method contains code to read gamepad #1. The code uses the left joystick on gamepad #1 to control the left and right motors to make the robot move.

The `loop()` method also contains code to check if certain buttons ("A", "Y", "X" and "B") are pressed. If the "A" button is pressed, the robot controller will increment the position of the arm's servo (which will cause the arm to move downwards – for the K9 bot, increasing values of the arm servo position correspond to a lowering of the arm). If the "Y" button is pressed, the controller will decrease the arm servo position (which effectively raises the arm).

Similarly if controller detects that the "X" button is being pressed, the controller will close the claw by increase the claw's servo position (for the claw servo, high values of the servo position correspond to the claw being closed). If the controller detects that the "B" button is being pressed, it will open the claw slightly by decrease the claw's servo position.

IMPORTANT NOTE: for the event driven programming model it is important that you do not *block* within the loop() method. This means that in the loop() method you do not want your code to dwell/wait within the method. Your code should do what it needs to do, then exit the loop() method as soon as possible. For example, you should avoid sleeping or waiting within the loop() event. There are certain tasks that need to be executed for the FTC Robot Controller app. These tasks occur in between the loop() events. If your program waits or blocks within the loop() event it could cause negative effects for your robot controller.

9.3.3.3 Telemetry Statements

In the `loop()` method for the K9TeleOp program module you can see that there are some telemetry statements in the code.

```
telemetry.addData("Text", "*** Robot Data***");
telemetry.addData("arm", "arm:   " + Double.toString(armPosition));
telemetry.addData("claw", "claw: " + Double.toString(clawPosition));
telemetry.addData("left tgt pwr",  "left  pwr: " + Float.toString(left));
telemetry.addData("right tgt pwr", "right pwr: " + Float.toString(right));
```

Figure 139 - There are several telemetry statements in the example run() method.

The new robot control platform allows you to send telemetry data from the robot back to the driver station. In Figure 139 you can see that a method called addData is called five times in a row. This addData method, which is a member of an object called *telemetry* lets the user send text data back to the robot controller. The first argument of the addData method is a string that acts as a key or index for the piece data. The second argument is the actual data or message.

These telemetry messages will be underneath the op mode controls of the FTC Driver Station app. The telemetry messages are refreshed regularly as the op mode continues to run.

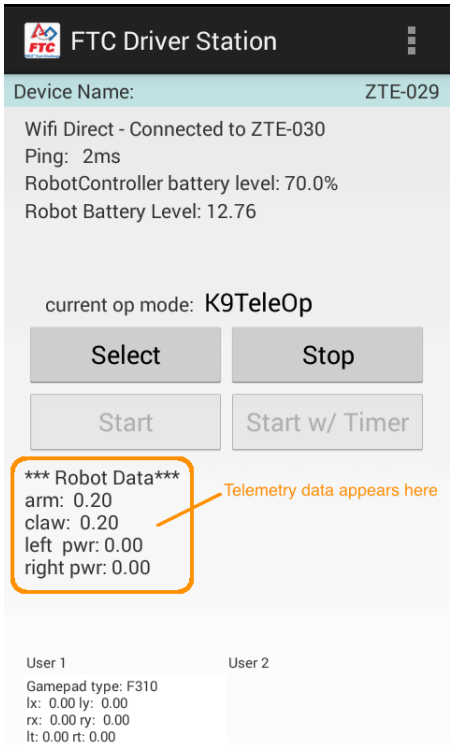

Figure 140 - Telemetry data gets refreshed regularly while the op mode is running.

9.3.3.4 The stop() Method

When the robot controller receives a stop command (or if it has to stop due to an emergency stop condition) the `stop()` method of the op mode will be run. For the K9TeleOp example, the `stop()` method is empty. You do not need to write code to turn off the motors or servos when a stop command is received. The Robot Controller framework will do this automatically for you (for safety reasons).

```
/*
 * Code to run when the op mode is first disabled goes here
 *
 * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#stop()
 */
@Override
public void stop() {

}
```

Figure 141 - The stop() method for K9TeleOp is empty.

However, if you do have some cleanup items that you would like to do after your op mode run is complete, you can place these tasks in the `stop()` method of the op mode.

9.3.4 Making Your Own Op Mode

Let's make our own op mode. To make it easy, instead of making our own op mode from scratch, we will simply copy an existing op mode, rename it, register it, then build the app and run the new op mode. We will use the K9TeleOp program as the basis of our "new" op mode.

In Android Studio, right mouse click on the K9TeleOp program file and select **Copy** from the pop-up menu.

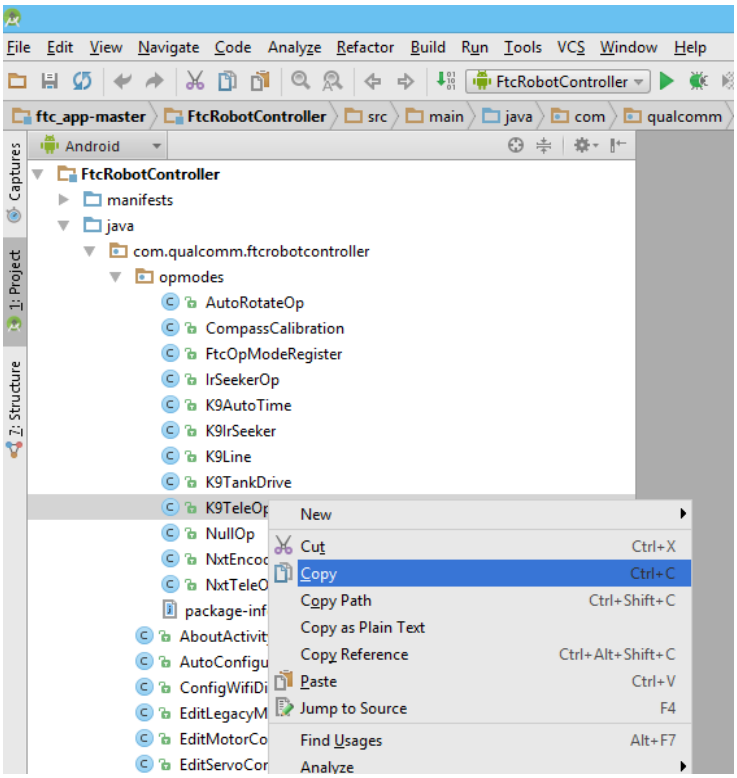

Figure 142 - Right mouse click on K9TeleOp and select Copy from the pop-up menu

After you have copied the file, right mouse click on the *opmodes* folder and select **Paste** from the pop-up menu.

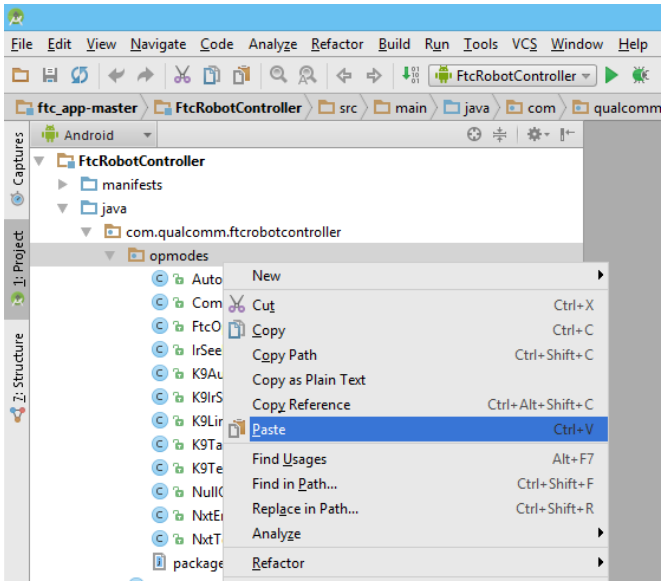

Figure 143 - Right mouse click the opmodes folder and select Paste from the pop-up menu.

Android Studio should prompt you for a new name for the new Java class that you are creating by pasting the file copy to the folder. Change the New Name text box to "MyTeleOp" and hit **OK** to create a new class.

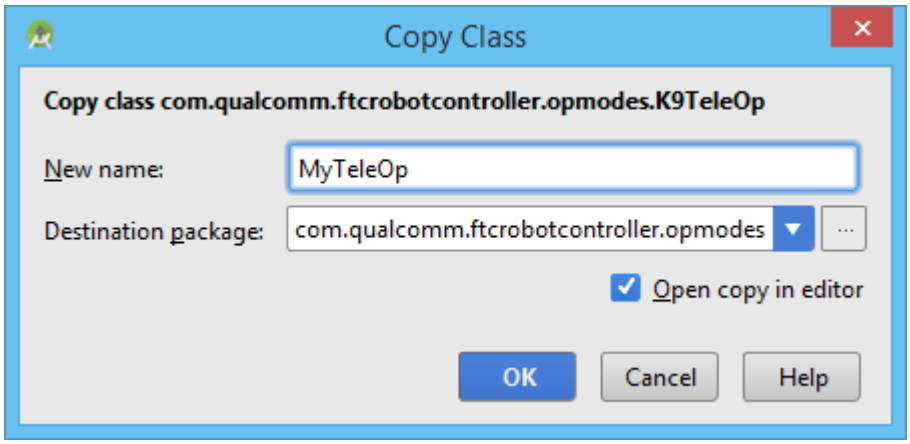

Figure 144 - Change the name to "MyTeleOp" and hit the OK button.

Android Studio will create a new class called "MyTeleOp" that will appear in the *opmodes* folder in the project browser. The contents of the newly created class should be the same as the contents of "K9TeleOp", except the name of the class should be changed to "MyTeleOp".

If you were to try and build and run your Robot Controller app, the newly created op mode would not appear in the selection list on the driver station. The newly created op mode must be registered in the FtcOpModeRegister file before it will appear in the driver station's selection list.

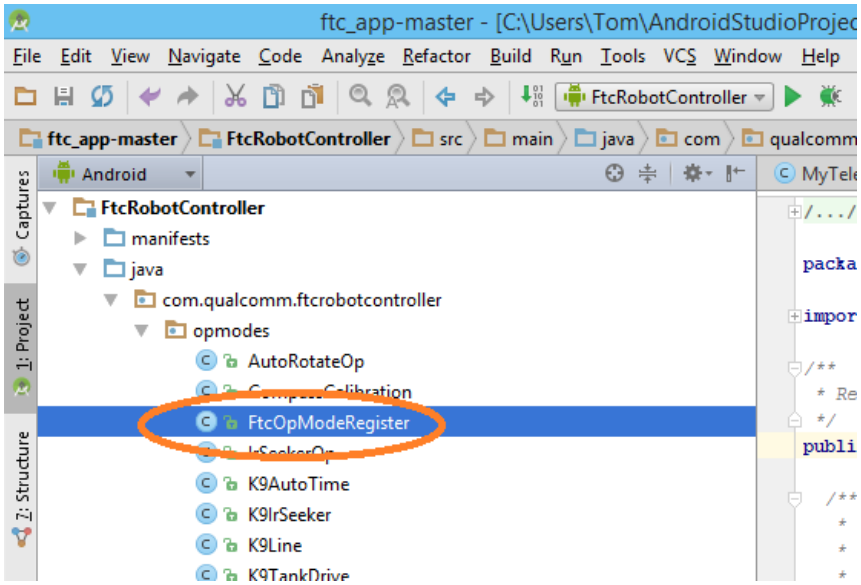

Figure 145 - Double click on FtcOpModeRegister to open the file for editing.

In Android Studio, double click on the FtcOpModeRegister class in the *opmodes* folder to open the file in the editor. In the editor window, look for the following line,

```
 *
 * If two or more op modes are registered with the same name, the
 */

manager.register("NullOp", NullOp.class);
manager.register("K9TeleOp", K9TeleOp.class);
manager.register("K9TankDrive", K9TankDrive.class);
manager.register("K9Line", K9Line.class);
manager.register("K9IrSeeker", K9IrSeeker.class);
manager.register("K9AutoTime", K9AutoTime.class);
/*
manager.register("IrSeekerOp", IrSeekerOp.class);
manager.register("CompassCalibration", CompassCalibration.class);
```

Figure 146 - Look for the line that reads "manager.register("K9AutoTime", K9AutoTime.class);".

Use your mouse and keyboard to insert the following text after the line shown in Figure 146.

```
 * If two or more op modes are registered with the same
 */

manager.register("NullOp", NullOp.class);
manager.register("K9TeleOp", K9TeleOp.class);
manager.register("K9TankDrive", K9TankDrive.class);
manager.register("K9Line", K9Line.class);
manager.register("K9IrSeeker", K9IrSeeker.class);
manager.register("K9AutoTime", K9AutoTime.class);
manager.register("MyTeleOp", MyTeleOp.class);
/*
manager.register("IrSeekerOp", IrSeekerOp.class);
```

Figure 147 – Insert "manager.register("MyTeleOp", MyTeleOp.class);" into the program module.

Android Studio should auto-save the file. Make sure your Android device is connected and that USB Debugging is enabled. Select **Run->Run FtcRobotController** from the **Run** menu and install the app onto your target device.

Once the app has been built, you should be able to load the existing configuration file. When you install the app it does not overwrite the file.

If you were able to successfully build and install the revised app, the next time you press the **Select** button on the Driver Station app, the "MyTeleOp" op mode should appear.

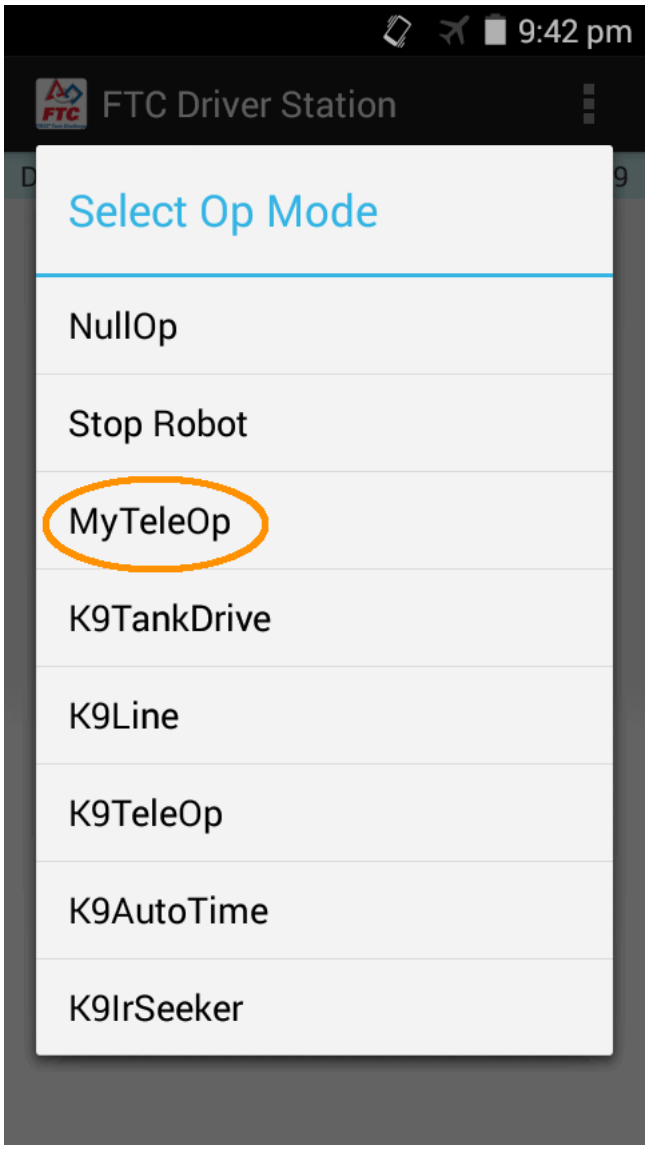

Figure 148 - MyTeleOp should now be available as an op mode.

9.4 QualComm SDK Documentation

QualComm has generated documentation for their FTC software development kit. The documentation is available as a compressed archive file. When you uncompress the file, you find a folder that contains a series of HTML documents that describe in detail the classes and methods that are available with the QualComm SDK.

You can use this documentation to create your own custom robot behaviors.

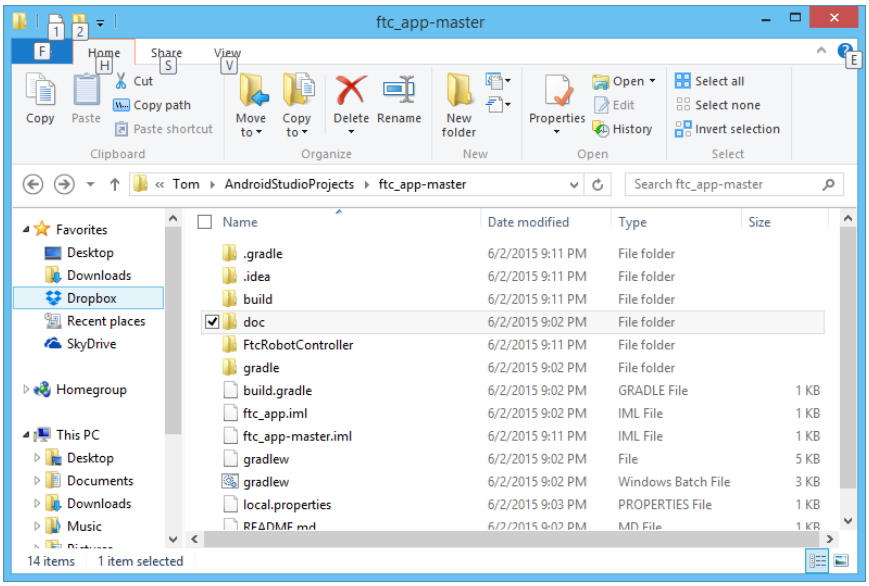

Figure 149 – You must uncompress the archive file to access the documentation.

To access the documentation, open the *doc* folder, and then open the *javadoc* subfolder. Double click on the file *index.html* to launch your web browser and to view the documentation.

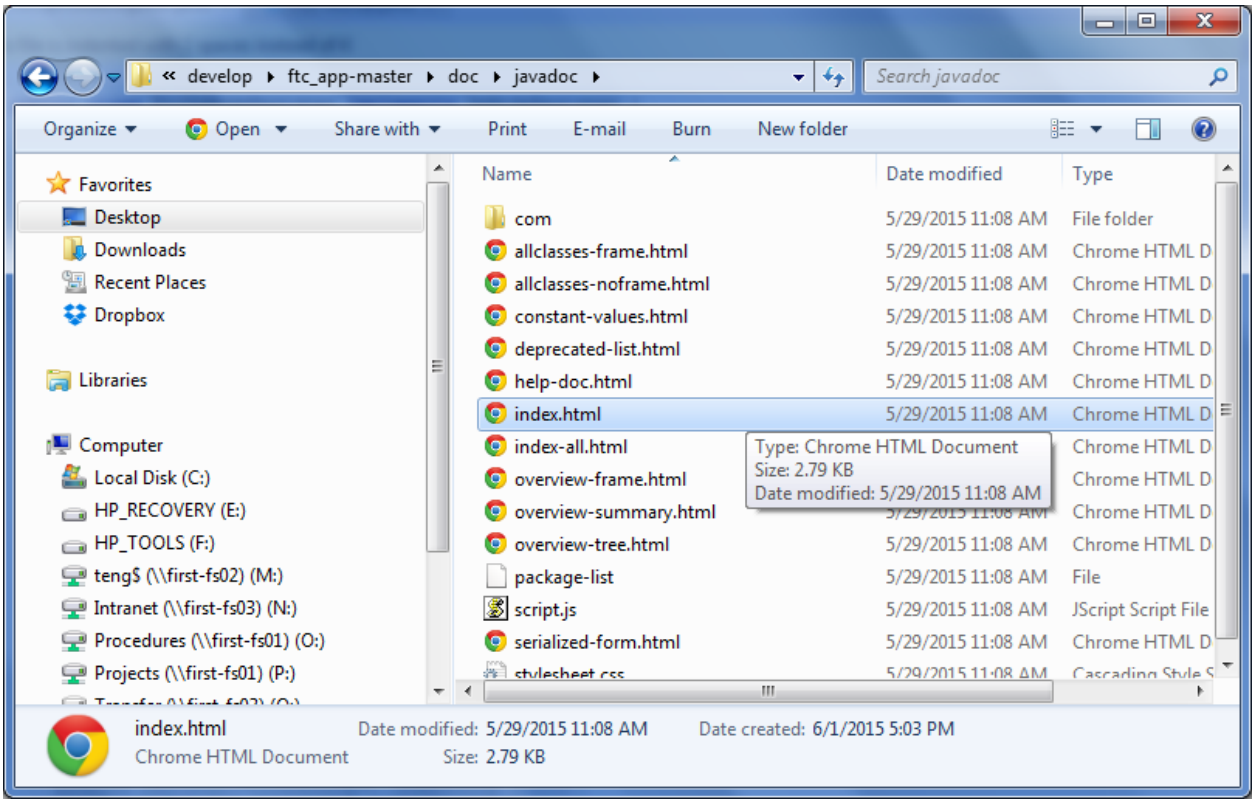

Figure 150 - Open the JavaDoc folder and double click on index.html to view the documentation.

You can browse the various topics in the user documentation through a standard web browser.

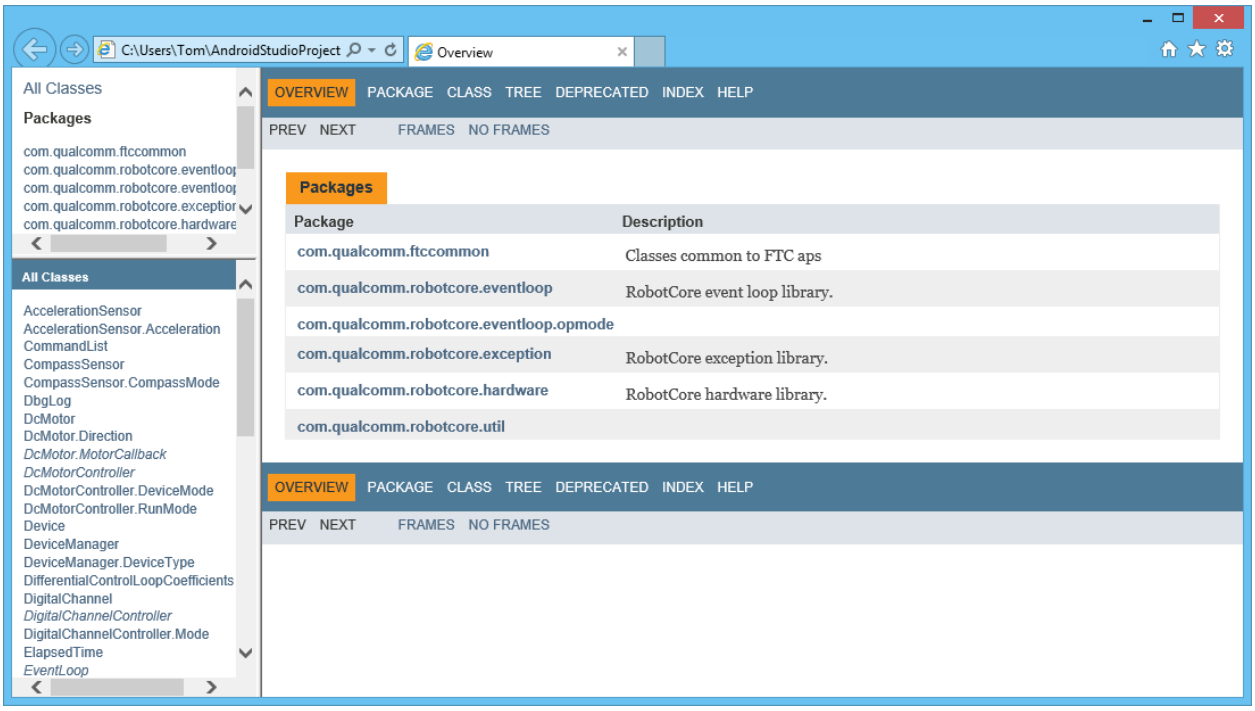

Figure 151 - You can use a standard web browser to view the documentation.

10 Conclusion

Due to time constraints this user manual only covers the very basics of operating and programming a next-gen, Android-based FTC robot. If you were able to follow and complete all of the tasks that are outlined in this manual, then you are ready to begin writing custom op modes for your robot.

You can take any of the example op modes (particularly the ones that begin with the phrase “K9”) and use them as the basis for your own custom op mode. Also, you should consult the QualComm SDK documentation to learn what classes are available and to learn the syntax of the methods available through the SDK.

Thank you for participating in today’s training event!!!

Appendix A K9TeleOp Example Code

```
package com.qualcomm.ftcrobotcontroller.opmodes;

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.util.Range;

/**
 * TeleOp Mode
 * <p>
 * Enables control of the robot via the gamepad
 */
public class K9TeleOp extends OpMode {

   /*
    * Note: the configuration of the servos is such that
    * as the arm servo approaches 0, the arm position moves up (away from the floor).
    * Also, as the claw servo approaches 0, the claw opens up (drops the game element).
    */
   // TETRIX VALUES.
   final static double ARM_MIN_RANGE  = 0.20;
   final static double ARM_MAX_RANGE  = 0.90;
   final static double CLAW_MIN_RANGE  = 0.20;
   final static double CLAW_MAX_RANGE  = 0.7;

   // position of the arm servo.
   double armPosition;

   // amount to change the arm servo position.
   double armDelta = 0.1;

   // position of the claw servo
   double clawPosition;

   // amount to change the claw servo position by
   double clawDelta = 0.1;

   DcMotor motorRight;
   DcMotor motorLeft;
   Servo claw;
   Servo arm;

   /**
    * Constructor
    */
   public K9TeleOp() {

   }

   /*
    * Code to run when the op mode is first enabled goes here
    *
    * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#start()
    */
   @Override
   public void start() {


      /*
       * Use the hardwareMap to get the dc motors and servos by name. Note
       * that the names of the devices must match the names used when you
       * configured your robot and created the configuration file.
       */

      /*
       * For the demo Tetrix K9 bot we assume the following,
       *   There are two motors "motor_1" and "motor_2"
       *   "motor_1" is on the right side of the bot.
```

```
     *   "motor_2" is on the left side of the bot and reversed.
     *
     * We also assume that there are two servos "servo_1" and "servo_6"
     *    "servo_1" controls the arm joint of the manipulator.
     *    "servo_6" controls the claw joint of the manipulator.
     */
    motorRight = hardwareMap.dcMotor.get("motor_2");
    motorLeft = hardwareMap.dcMotor.get("motor_1");
    motorLeft.setDirection(DcMotor.Direction.REVERSE);

    arm = hardwareMap.servo.get("servo_1");
    claw = hardwareMap.servo.get("servo_6");

    // assign the starting position of the wrist and claw
    armPosition = 0.2;
    clawPosition = 0.2;
}

/*
 * This method will be called repeatedly in a loop
 *
 * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#run()
 */
@Override
public void loop() {

    /*
     * Gamepad 1
     *
     * Gamepad 1 controls the motors via the left stick, and it controls the
     * wrist/claw via the a,b, x, y buttons
     */

    // throttle: left_stick_y ranges from -1 to 1, where -1 is full up, and
    // 1 is full down
    // direction: left_stick_x ranges from -1 to 1, where -1 is full left
    // and 1 is full right
    float throttle = -gamepad1.left_stick_y;
    float direction = gamepad1.left_stick_x;
    float right = throttle - direction;
    float left = throttle + direction;

    // clip the right/left values so that the values never exceed +/- 1
    right = Range.clip(right, -1, 1);
    left = Range.clip(left, -1, 1);

    // scale the joystick value to make it easier to control
    // the robot more precisely at slower speeds.
    right = (float)scaleInput(right);
    left =  (float)scaleInput(left);

    // write the values to the motors
    motorRight.setPower(right);
    motorLeft.setPower(left);

    // update the position of the arm.
    if (gamepad1.a) {
        // if the A button is pushed on gamepad1, increment the position of
        // the arm servo.
        armPosition += armDelta;
    }

    if (gamepad1.y) {
        // if the Y button is pushed on gamepad1, decrease the position of
        // the arm servo.
        armPosition -= armDelta;
    }

    // update the position of the claw
    if (gamepad1.x) {
        clawPosition += clawDelta;
```

```
        }

        if (gamepad1.b) {
            clawPosition -= clawDelta;
        }

          // clip the position values so that they never exceed their allowed range.
          armPosition = Range.clip(armPosition, ARM_MIN_RANGE, ARM_MAX_RANGE);
          clawPosition = Range.clip(clawPosition, CLAW_MIN_RANGE, CLAW_MAX_RANGE);

        // write position values to the wrist and claw servo
        arm.setPosition(armPosition);
        claw.setPosition(clawPosition);



        /*
         * Send telemetry data back to driver station. Note that if we are using
         * a legacy NXT-compatible motor controller, then the getPower() method
         * will return a null value. The legacy NXT-compatible motor controllers
         * are currently write only.
         */
          telemetry.addData("Text", "*** Robot Data***");
          telemetry.addData("arm", "arm:  " + String.format("%.2f", armPosition));
          telemetry.addData("claw", "claw:  " + String.format("%.2f", clawPosition));
          telemetry.addData("left tgt pwr",  "left  pwr: " + String.format("%.2f", left));
          telemetry.addData("right tgt pwr", "right pwr: " + String.format("%.2f", right));

    }

    /*
     * Code to run when the op mode is first disabled goes here
     *
     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#stop()
     */
    @Override
    public void stop() {

    }

    /*
     * This method scales the joystick input so for low joystick values, the
     * scaled value is less than linear.  This is to make it easier to drive
     * the robot more precisely at slower speeds.
     */
    double scaleInput(double dVal)  {
        double[] scaleArray = { 0.0, 0.05, 0.09, 0.10, 0.12, 0.15, 0.18, 0.24,
                0.30, 0.36, 0.43, 0.50, 0.60, 0.72, 0.85, 1.00, 1.00 };

        // get the corresponding index for the scaleInput array.
        int index = (int) (dVal * 16.0);
        if (index < 0) {
            index = -index;
        } else if (index > 16) {
            index = 16;
        }

        double dScale = 0.0;
        if (dVal < 0) {
            dScale = -scaleArray[index];
        } else {
            dScale = scaleArray[index];
        }

        return dScale;
    }

}
```