



Training and Support



2017-2018 *FIRST*[®] Tech Challenge Android Studio Programming Training Manual



Revision History		
Revision	Date	Description
1.0	09/27/2017	Initial Release

Table of Contents

1	Introduction	4
1.1	What is FIRST® Tech Challenge?	4
1.2	FIRST Tech Challenge Core Values	4
2	Gracious Professionalism®	4
2.1	Gracious Professionalism for Volunteers.....	5
3	Youth Protection Program	5
3.1	Youth Protection Expectations and Guidelines	5
3.2	NOTICE OF NON-DISCRIMINATION	5
4	Training Manual Introduction	6
4.1	The Java Programming Language.....	6
4.2	Point-to-Point Control System	6
4.3	REV Robotics Expansion Hub	6
5	Required Materials.....	7
6	Setting Up Your Smartphones.....	10
6.1	Renaming Your Smartphones	10
6.2	Installing the FIRST Tech Challenge Apps.....	14
6.3	Placing Phones into Airplane Mode with Wi-Fi On.....	18
6.4	Pairing the Driver Station to the Robot Controller	19
7	Connecting Devices to an Expansion Hub.....	23
7.1	Connecting 12V Power to the Expansion Hub	23
7.2	Connecting a Motor to the Expansion Hub.....	26
7.3	Connecting a Servo to the Expansion Hub.....	27
7.4	Connecting a Color-Distance Sensor to the Expansion Hub	28
7.5	Connecting a Touch Sensor to the Expansion Hub	29
8	Configuring Your Hardware on the Robot Controller	30
9	Installing Android Studio.....	43
9.1	Downloading & Installing Android Studio	43
9.1.1	Android Developer Website.....	43
9.1.2	System Requirements	43
9.1.3	Java Development Kit.....	44
9.1.4	Downloading and Installing Android Studio	44

9.2	Disabling Android Studio Instant Run	45
9.2.1	What is Instant Run?	45
9.2.2	Locating Instant Run Settings.....	45
9.2.3	Additional Information.....	46
9.3	Downloading the Android Studio Project Folder	46
9.3.1	Downloading the Android Studio Project Folder	46
9.3.2	Extracting the Contents of the Archived Project File	48
9.3.3	Importing the FTC Project into Android Studio.....	49
10	Writing an Op Mode with Android Studio	51
10.1	Enabling Developer Options	51
10.2	Creating & Running an Op Mode	52
10.2.1	What’s an Op Mode?	52
10.2.2	Android Studio	53
10.2.3	TeamCode Module.....	53
10.2.4	Javadoc Reference Information	54
10.2.5	Enabling Auto Import.....	54
10.2.6	Sample Op Modes	55
10.2.7	Creating Your FIRST Op Mode.....	55
10.2.8	Examining the Structure of Your Op Mode.....	57
10.2.9	Building and Installing Your Op Mode	59
10.2.10	Running Your Op Mode.....	61
10.2.11	Modifying Your Op Mode to Control a Motor	63
10.2.12	Running Your Op Mode with a Gamepad Connected.....	64
11	Controlling a Servo Motor with an Op Mode	66
11.1	What is a Servo Motor?	66
11.2	Modifying Your Op Mode to Control a Servo	67
12	Using Sensors	68
12.1	Color-Distance Sensor	68
12.2	Touch Sensor.....	69
13	FIRST Tech Challenge Reference Information.....	70
13.1	Javadoc Reference Pages	70
13.2	Control System Wiki.....	70
13.3	Technology Forum	70
14	Troubleshooting.....	71

14.1 Driver Station Appears Unresponsive 71

14.2 Warning: problem communicating... 72

1 Introduction

1.1 What is FIRST® Tech Challenge?

FIRST Tech Challenge is a student-centered program that focuses on giving students a unique and stimulating experience. Each year, teams engage in a new Game where they design, build, test, and program autonomous and driver operated robots that must perform a series of tasks.

They also cultivate life skills such as:

- Planning, brainstorming, and creative problem-solving.
- Research and technical skills.
- Collaboration and teamwork.
- Appreciating differences and respecting the ideas and contributions of others.

To learn more about FIRST Tech Challenge and other FIRST® Programs, visit www.firstinspires.org.

FIRST Tech Challenge is MORE THAN ROBOTSSM! While competing, students develop personal and professional skills they will be able to rely on throughout their life.

1.2 FIRST Tech Challenge Core Values

FIRST asks everyone who takes part in FIRST Tech Challenge to uphold the following values:

- We display *Gracious Professionalism*[®] with everyone we engage with and in everything we do.
- We act with integrity.
- We have fun.
- We are a welcoming community of students, mentors, and volunteers.
- What we learn is more important than what we win.
- We respect each other and celebrate our diversity.
- Students and adults work together to find solutions to challenges.
- We honor the spirit of friendly competition.
- We behave with courtesy and compassion for others always.
- We act as ambassadors for FIRST and FIRST Tech Challenge.
- We inspire others to adopt these values.

2 Gracious Professionalism[®]

FIRST uses this term to describe our programs' intent and is shared with all young people engaging in FIRST programs. At FIRST, team members help other team members, but they also help other teams.

Gracious Professionalism[®] is not clearly defined for a reason. It has different meanings to everyone.

Some possible meanings of *Gracious Professionalism* include:

- Gracious attitudes and behaviors are win-win.
- Gracious folks respect others and let that respect show in their actions.
- Gracious Professionals make valued contributions in a way that is pleasing to others and to themselves.

In FIRST, *Gracious Professionalism* teaches teams and student participants:

An example of *Gracious Professionalism* is patiently listening to a team's question and providing support despite having several pressing things to do on the day of the event.

- Learn to be strong competitors, but also treat one another with respect and kindness in the process.
- Avoid leaving anyone feeling as if they are excluded or unappreciated.
- Knowledge, pride and empathy should be comfortably and genuinely blended.

In the end, *Gracious Professionalism*® is part of everyday life. When professionals use their knowledge in a graciously and individuals act with integrity and sensitivity, everyone wins, and society benefits.

Watch Dr. Woodie Flowers explain *Gracious Professionalism* in this [short video](#).

2.1 Gracious Professionalism for Volunteers

It is a good idea to spend time going over this concept with volunteers. Provide volunteers with real-life examples of *Gracious Professionalism* in practice before, during, and after the event and recognize great *Gracious Professionalism* when you see it in action!

3 Youth Protection Program

The *FIRST* YPP sets minimum standards recommended for all *FIRST* activities. Adults working in *FIRST* programs must be knowledgeable of the standards set by the *FIRST* YPP, as well as those set by the school or organization hosting their team.

3.1 Youth Protection Expectations and Guidelines

Coaches and mentors should read and follow the [FIRST Youth Protection Program guide](#). Anything labeled as required is mandatory in the United States and Canada, and cannot be waived without approval from the *FIRST* Youth Protection Department. *FIRST* recommends that the standards set forth in the *FIRST* Youth Protection Program guide be applied outside of the United States and Canada to the extent possible. At a minimum, local regulations regarding youth protection must be complied with.

Most up to date forms are available here: <http://firstinspires.org/resource-library/youth-protection-policy>

The US Screening process, the Canadian Screen process, Frequently Asked Questions (FAQ), and additional information are on the *FIRST* Youth Protection Program Website: <http://firstinspires.org/resource-library/youth-protection-policy>

3.2 NOTICE OF NON-DISCRIMINATION

For Inspiration and Recognition of Science and Technology (*FIRST*®) does not discriminate based on race, color, national origin, sex, disability, age, status as a veteran who served in the military, religion, gender, gender identity, or gender expression in its programs and activities.

Keep updated at: <http://www.firstinspires.org/about/legal-notices>

4 Training Manual Introduction

This document shows how to set up, configure and program the control system used for the *FIRST* Tech Challenge competition using Google’s Android Studio integrated development environment (IDE). The basic examples in this document use the REV Robotics Expansion Hub as the primary input/output module. For detailed information about the REV Robotics Expansion Hub, refer to the *REV Robotics Expansion Hub Guide* which is available from the REV Robotics website (<http://www.revrobotics.com/>).

4.1 The Java Programming Language

This tutorial assumes that you have a sound understanding of the Java programming language. If you do not know Java, then you should consider using the FTC Blocks Programming Tool, which is a visual development tool. Information about the FTC Blocks Programming Tool can be found at the following address:

https://github.com/ftctechnh/ftc_app/wiki/Writing-an-Op-Mode-with-FTC-Blocks

Or, you can learn the Java programming language by completing the Oracle Java Tutorial, which is available at the following address:

<https://docs.oracle.com/javase/tutorial/>

4.2 Point-to-Point Control System

Teams competing in the *FIRST* Tech Challenge use a *point-to-point* system to control their robots.



Figure 1 – The control system has a Driver Station connected wirelessly to a Robot Controller.

The *Robot Controller* acts as the “brains” of the robot and is mounted on the frame of the robot. The *Driver Station* lets humans interact with the Robot Controller through a wireless connection.

4.3 REV Robotics Expansion Hub

The REV Robotics Expansion Hub is the electronic *input/output* (or “I/O”) module that lets the Robot Controller talk to the robot’s motors, servos, and sensors. The Robot Controller communicates with the Expansion Hub through a USB connection.

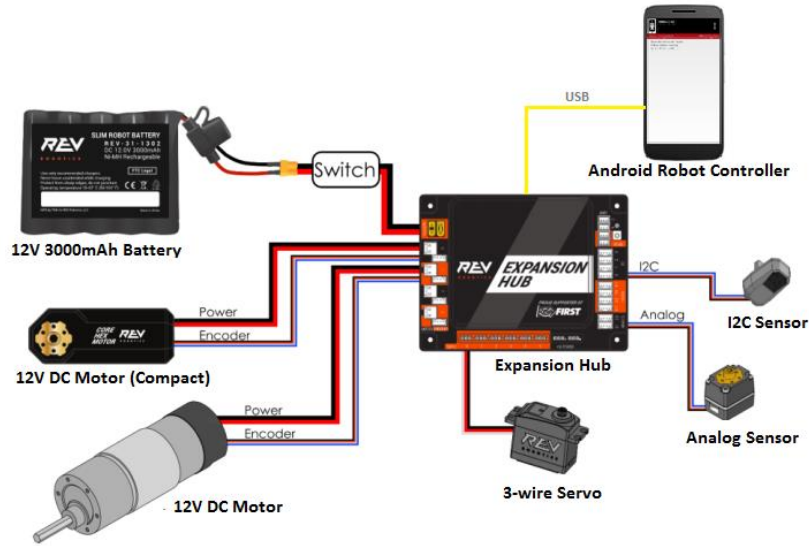

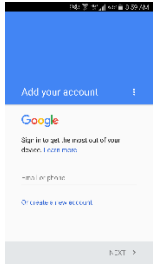



Figure 2 - The Expansion Hub lets the Robot Controller talk to the sensors, motors and servos.

5 Required Materials

To follow along with the examples in this document, you will need the following items:

Two (2) <i>FIRST</i> -approved Android smartphones ¹	
A Google account (available for free from Google) to access the Google Play store	
Wireless Internet access	

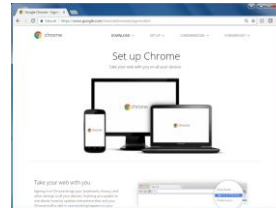
¹ Consult the official *FIRST* Tech Challenge Game Manual, Part 1 for a list of approved devices.

Laptop with Microsoft Windows 7, 8 or 10 and Wi-Fi capability.

Note that your laptop should also have the most current service packs and system updates from Microsoft.



Javascript-enabled web browser (Google Chrome is the recommended browser).



REV Robotics Expansion Hub (REV-31-1153)





REV Robotics Switch, Cable, and Bracket (REV-31-1387)



REV Robotics Tamiya to XT30 Adapter (REV-31-1382)



<p>FIRST-approved 12V Battery (such as Tetrax W39057)</p>	
<p>FIRST-approved 12V DC Motor (such as Tetrax W39530, with power cable W41352)²</p>	
<p>REV Robotics Anderson to JST VH Cable (REV-31-1381)</p>	
<p>180-Degree Standard Scale Servo (such as Hitec HS-485HB)</p>	
<p>REV Robotics Color Sensor with 4-Pin JST PH Cable (REV-31-1154)</p>	
<p>REV Robotics Touch sensor with 4-Pin JST PH Cable (REV-31-1425)</p>	

² Note that for the examples listed in this document, it is recommended that the user builds a simple structure using a compatible build kit (such as TETRAX Max) to properly secure the DC motor and prevent it from rolling about uncontrollably while running the sample op modes.

USB Type A male to type mini-B male cable	
Two (2) micro USB OTG adapters	
Logitech F310 USB Gamepad	

6 Setting Up Your Smartphones

6.1 Renaming Your Smartphones

You will need to have two *FIRST*-approved Android smartphones for the control system. One phone will be used as the Robot Controller, the other will be used as the Driver Station.³

By establishing unique names for your phones, you can ensure that your phones will be communicating with each other rather than other teams' phones at meets and tournaments. The official rules of the *FIRST* Tech Challenge require that you change the Wi-Fi name of your smartphones to include your team number and “-RC” if the phone is a Robot Controller or “-DS” if it is a Driver Station. A team can insert an additional dash and a letter (“A”, “B”, “C”, etc.) if the team has more than one set of Android phones.

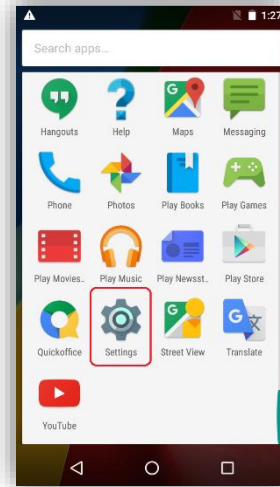
If, for example, a team has a team number of 9999 and the team has multiple sets of phones, the team might decide to name one phone “9999-C-RC” for the Robot Controller and the other phone “9999-C-DS” for the Driver Station. The “-C” indicates that these devices belong to the third set of phones for this team.

³ Note that the screen images for this document were made using a pair of Motorola Android smartphones. The actual screen images on your smartphones might differ slightly from the images depicted in this document.

Renaming Your Smartphones (Time Needed to Complete Task: 5 Minutes per Phone)

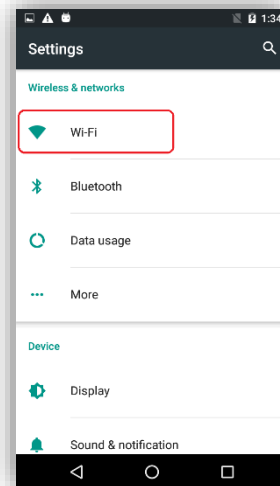
Step 1:

Click on **Settings** icon to display the Android Settings screen.



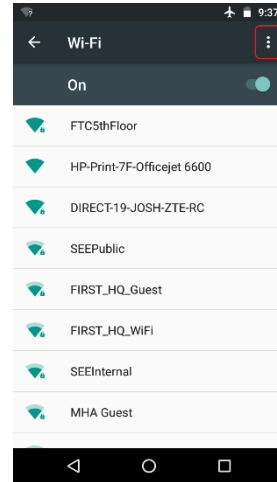
Step 2:

Click on **Wi-Fi** to launch the Wi-Fi screen.



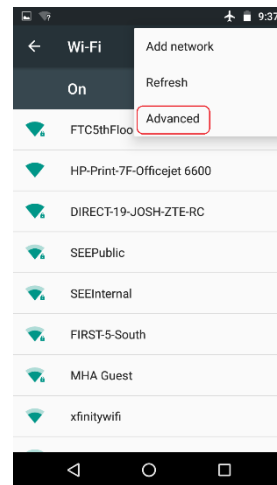
Step 3:

Touch the three vertical dots in upper right-hand corner to display a pop-up menu.



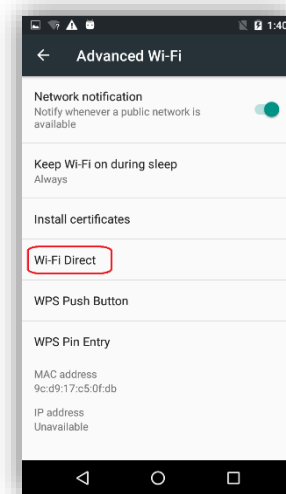
Step 4:

Select **Advanced** from the pop-up menu to display the Advanced Wi-Fi screen.



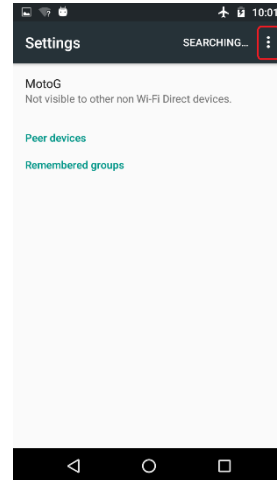
Step 5:

Select **Wi-Fi Direct** to display the Wi-Fi Direct menu.

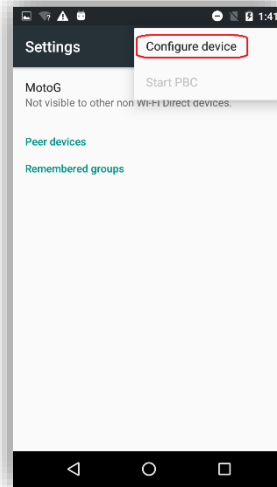


Step 6:

Touch the three vertical dots to display the pop-up menu.

**Step 7:**

Select **Configure Device** from the pop-up menu.

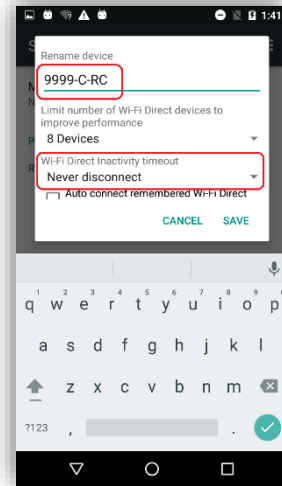


Step 8:

In the **Configure Device** screen, use the touch keypad to type in the new name of the device. If it is going to be a Robot Controller device, specify your team number and “-RC”. If it is going to be a Driver Station, specify your team number and “-DS”.

You can also set the Wi-Fi Direct Inactivity timeout to “Never disconnect” and then hit the **SAVE** button to save your changes.

Note that in the screenshot shown to the right, the team number is “9999”. The “-C” indicates that this is from the third pair of smartphones for this team. The “-RC” indicates that this phone will be a Robot Controller.

**Step 9:**

After you have renamed your phone, power cycle (i.e., restart) your smartphone.

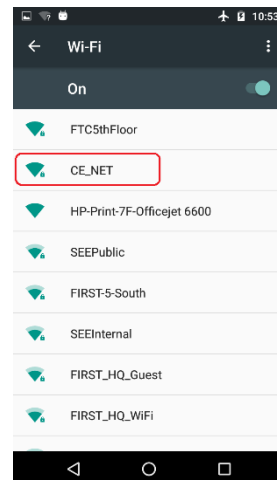
6.2 Installing the FIRST Tech Challenge Apps

The FTC apps are available to download for free from the Google Play store. You will need to have your Android phones connected to a Wi-Fi network that has Internet access before you can access the Google Play store. You will also need a Google account to be able to download the apps from the Google Play store.

Installing the FIRST Tech Challenge Apps (Time Needed to Complete Task: 7.5 Minutes per Phone)

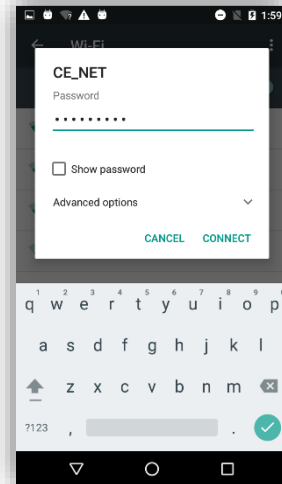
Step 1:

From the Android Wi-Fi screen look for the name of your wireless network (“CE_NET” in this example) and touch the wireless network name to login to the network.

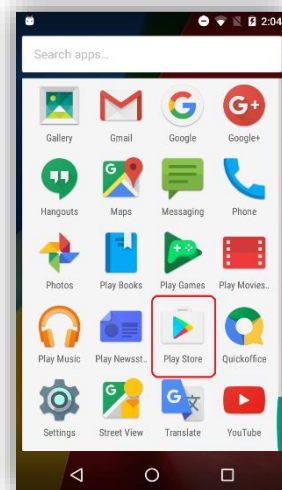


Step 2:

Specify the password using the touch keypad and hit **CONNECT** to connect to this wireless network.

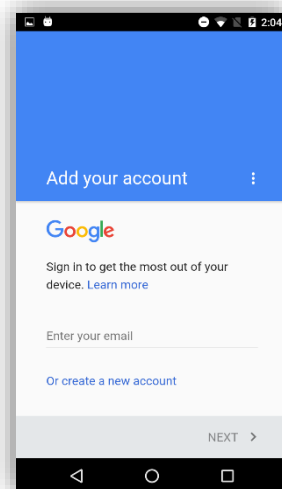
**Step 3:**

Find the Google Play Store icon on your phone and click it to launch the Google Play Store app.

**Step 4:**

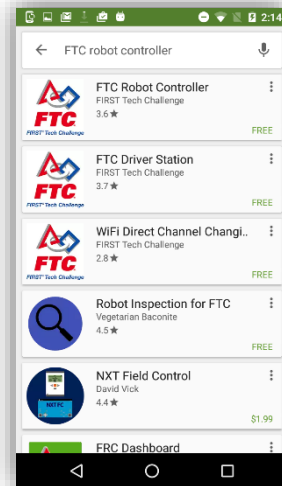
If you haven't signed into your Google account yet, follow the onscreen instructions to log into your Google account.

If you don't have a Google account, follow the onscreen instructions to create a new account.



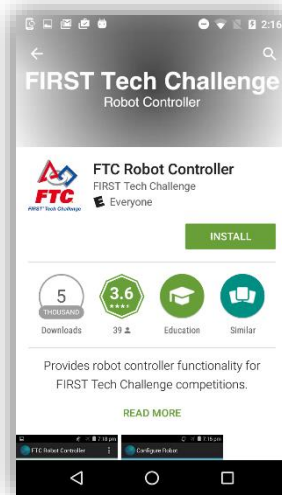
Step 5:

In the search window of the Google Play app, type in the words “FTC Robot Controller” to find the Robot Controller or “FTC Driver Station” to find the appropriate FTC app for your phone.

**Step 6:**

Tap on the app in the Google Play listing to bring up the installation screen. Follow the onscreen instructions to install the appropriate app for your phone.

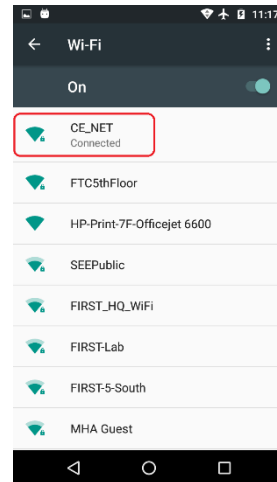
Important note: When you install the FTC apps, **only install one FTC app** (FTC Robot Controller or FTC Driver Station) **per phone**. You should avoid installing both apps onto the same phone. Doing so can cause Wi-Fi connection problems. You should only install the FTC Robot Controller app onto the phone that will be the Robot Controller and the FTC Driver Station app onto the phone that will be the Driver Station.



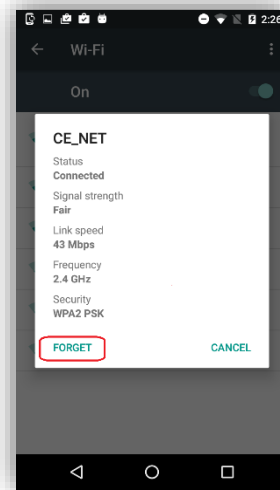
Step 7:

After you have successfully installed the app, you should forget the external wireless network on your phone.

Go to the Android Wi-Fi screen, find the name of the currently connected network, and tap on the network name to bring up a pop-up box with info about the network.

**Step 8:**

Click on **FORGET** button to forget the wireless network.



6.3 Placing Phones into Airplane Mode with Wi-Fi On

For the FIRST Tech Challenge competitions, it is important that you place your Robot Controller and Driver Station phones into Airplane mode but keep their Wi-Fi radios turned on. This is important because you do not want any of the cellular telephone functions to be enabled during a match. The cellular telephone functions could disrupt the function of the robot during a match.

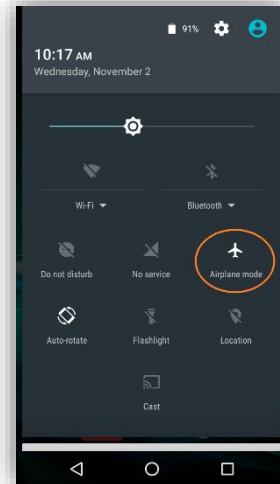
Placing Phones into Airplane Mode with Wi-Fi On (Time Needed to Complete Task: 2.5 Minutes per Phone)

Step 1:

On the main Android screen of each smartphone, use your finger to slide from the top of the screen down towards the bottom of the screen to display the quick configuration screen.

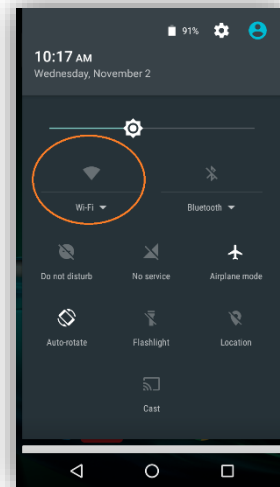
Note that for some smartphones you might have to swipe down more than once to display the quick configuration screen, particularly if there are messages or notifications displayed at the top of your screen.

Look for the Airplane mode icon (which is shaped like an airplane) and if the icon is not activated, touch the icon to put the phone into airplane mode.



Step 2:

Placing the phone into airplane mode will turn off the Wi-Fi radio. If the Wi-Fi icon has a diagonal line through it (see Step 1 above), then the Wi-Fi radio is disabled. You will need to touch the “Wi-Fi” icon on the quick configuration screen to turn the Wi-Fi radio back on.



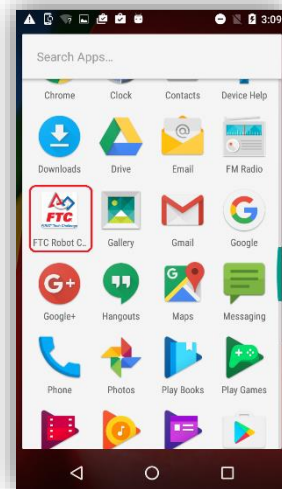
6.4 Pairing the Driver Station to the Robot Controller

Once you have successfully installed the FTC apps onto your Android phones, you will want to establish a secure wireless connection between the two devices. This connection will allow your Driver Station phone to select op modes on your Robot Controller phone and send gamepad input to these programs. Likewise, it will allow your op modes running on your Robot Controller phone to send telemetry data to your Driver Station phone where it can be displayed for your drivers. The process to connect the two phones is known as “pairing.”

Pairing the Driver Station to the Robot Controller (Time Needed to Complete Task: 10 minutes)

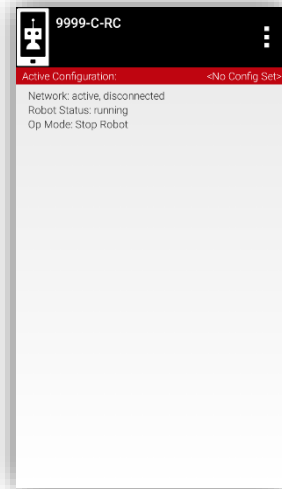
Step 1:

On the main Robot Controller smartphone, look for the FTC Robot Controller icon. Tap on the icon to launch the Robot Controller app.



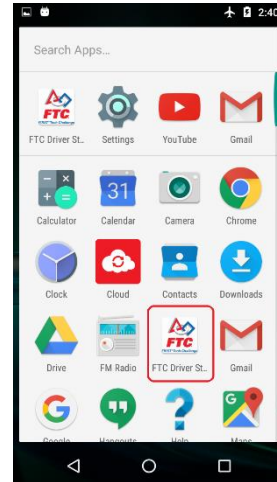
Step 2:

Verify that the Robot Controller app is running. The **Robot Status** field should read “running” if it is working properly.



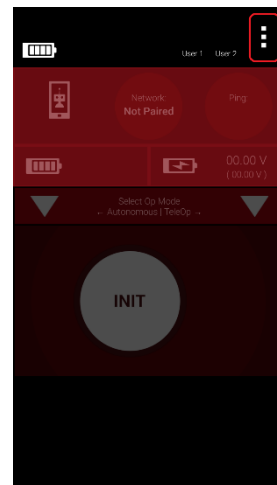
Step 3:

On your Driver Station phone, find the FTC Driver Station app icon. Tap on the icon to launch the Driver Station app.



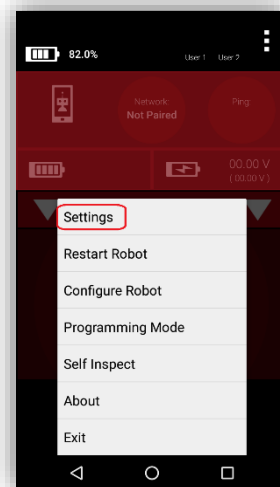
Step 4:

Touch the three vertical dots on the upper right hand corner of the main screen of the FTC Driver Station app. This will launch a pop-up menu.



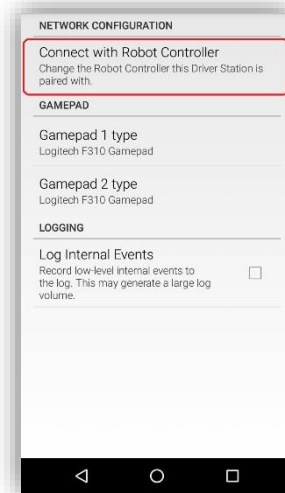
Step 5:

Select **Settings** from the pop-up menu.



Step 6:

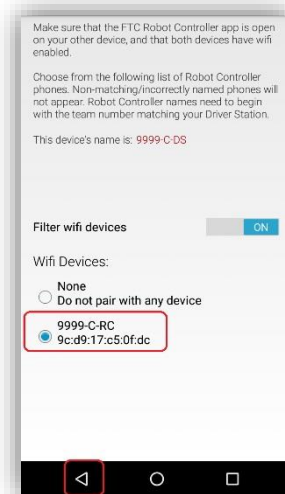
From the Settings screen, look for and select “Connect with Robot Controller” to launch the Connect with Robot Controller screen.

**Step 7:**

Find the name of your Robot Controller from the list and select it.

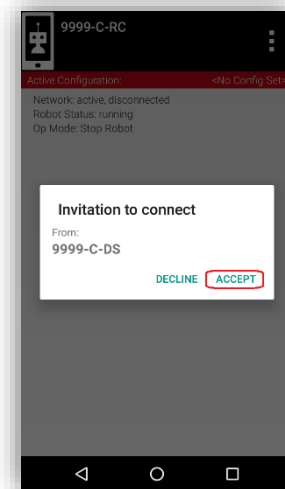
After you have made your selection, use the back-arrow key to return to the Settings screen.

Then press the back-arrow key one more time to return to the main Driver Station screen.

**Step 8:**

When the Driver Station returns to its main screen, the first time you attempt to connect to the Robot Controller a prompt should appear on the Robot Controller screen.

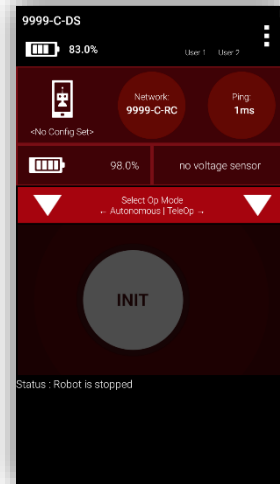
Click on the “ACCEPT” button to accept the connection request from the Driver Station.



Step 9:

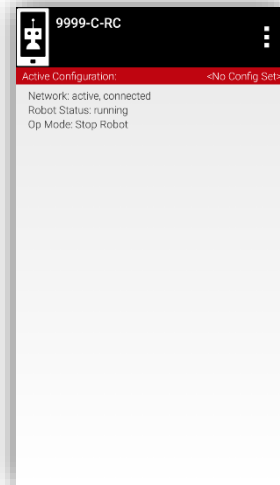
Verify that the Driver Station screen has changed and that it now indicates that it is connected to the Robot Controller.

The name of the Robot Controller’s remote network (“9999-C-RC” in this example) should be displayed in the **Network** field on the Driver Station.

**Step 10:**

Verify that the Robot Controller screen has changed and that it now indicates that it is connected to the Driver Station.

The **Network** status should read “active, connected” on the Robot Controller’s main screen.



7 Connecting Devices to an Expansion Hub

This section demonstrates how to connect and configure devices to your REV Robotics Expansion Hub. For the examples in this document, we will connect the Expansion Hub through a switch to a 12V battery. We will then connect a DC motor, a servo and some sensors to the Expansion Hub.

7.1 Connecting 12V Power to the Expansion Hub

The Expansion Hub draws power from a 12V rechargeable battery. For safety reasons, the battery has a 20A fuse built in. A mechanical switch is used to turn on/turn off the power.

Connecting 12V Power to the Expansion Hub (Time Needed to Complete Task: 5 minutes)

Step 1:

If your 12V battery has a Tamiya style connector, connect the Tamiya to XT30 adapter cable to the matching end of the switch cable.



Do not plug connect the 12V battery to the Tamiya adapter yet. We will connect the battery during a later step.

Step 2:

Connect the other end of the switch cable to a matching XT30 port on the Expansion Hub.



Step 3:

Verify that the switch is in the OFF position.



Step 4:

Connect the 12V battery to the Tamiya to XT30 cable.



Step 5:

Turn on the switch and verify that the Expansion Hub is drawing power from the battery. Note that the Expansion Hub's LED should be illuminated (notice blue LED in image below).

**Step 6:**

Turn off the switch and verify that the Expansion Hub is off. Note that the Expansion Hub's LED should not be illuminated.



7.2 Connecting a Motor to the Expansion Hub

The Expansion Hub can drive up to four (4) 12V DC motors per Expansion Hub. The Expansion Hub uses a type of electrical connector known as a 2-pin JST VH connector. At the time that this document was written, the FIRST-approved 12V DC motors were equipped with Anderson Powerpole connectors. An adapter cable is used to connect the Anderson Powerpole connectors to the Expansion Hub motor port.

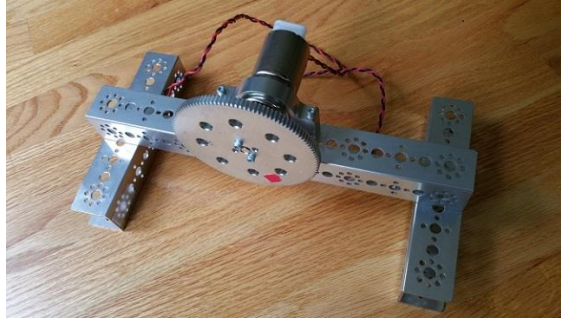


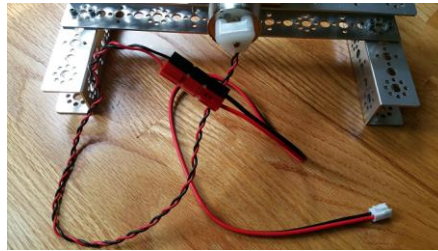
Figure 3 - Motor test rig made from Tetrix build components.

For the examples in this document, FIRST recommends that the user build a simple rig to secure the motor in place and prevent it from moving about during the test runs. The image above shows a Tetrix motor installed in a rig built with a Tetrix motor mount and some Tetrix C-channels. A gear was mounted on the motor shaft to make it easier for the user to see the rotation of the shaft.

Connecting a 12V DC Motor to the Expansion Hub (Time Needed to Complete Task: 2.5 minutes)

Step 1:

Connect the Anderson Powerpole end of the motor's power cable to the Powerpole end of the Anderson to JST VH adapter cable.



Step 2:

Connect the other end of the Anderson to JST VH adapter cable into the motor port labeled "0" on the Expansion Hub.



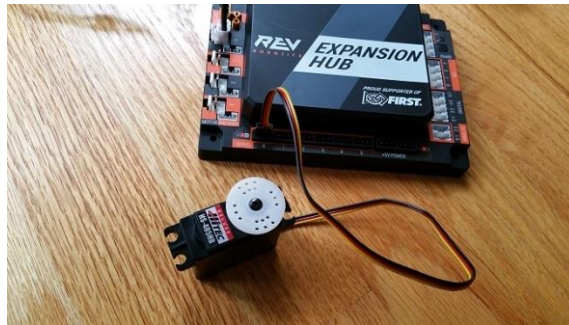
7.3 Connecting a Servo to the Expansion Hub

The REV Robotics Expansion Hub has 6 built-in servo ports. The servo ports accept the standard 3-wire header style connectors commonly found on servos. Note that ground pin is on the left side of the servo port.

Connecting a Servo to the Expansion Hub (Time Needed to Complete Task: 2.5 minutes)

Step 1:

Connect the servo cable to the servo port labeled “0” on the Expansion Hub. Note that the ground pin is on the left side of the servo port.



Step 2:

Verify that the black ground wire of the servo cable matches the ground pin of the servo port (which is aligned on the left side of the port).



7.4 Connecting a Color-Distance Sensor to the Expansion Hub

The Expansion Hub has 4 independent I2C buses. Each bus has its own port on the Hub. We will connect a REV Robotics Color-Distance sensor to the I2C bus #0 on the Expansion Hub.

Connecting a Color-Distance Sensor to the Expansion Hub (Time Needed to Complete Task: 2.5 minutes)

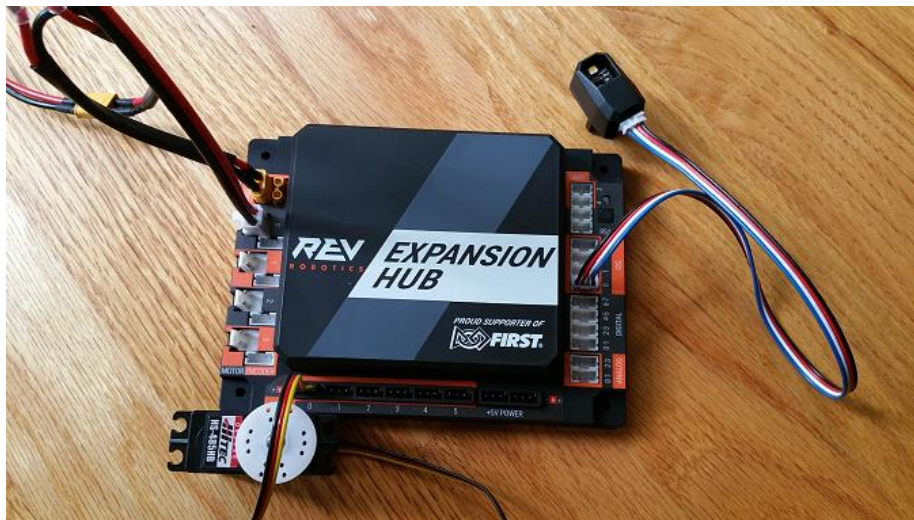
Step 1:

Connect the one end of the 4-pin JST PH cable to the REV Robotics Color-Distance sensor.



Step 2:

Plug the other end of the 4-pin JST PH cable to the I2C port labeled "0" on the Expansion Hub.



7.5 Connecting a Touch Sensor to the Expansion Hub

The Expansion Hub has 4 independent digital input/output (I/O) ports. Each port has two digital I/O pins for a total of 8 digital I/O pins on an Expansion Hub. You will connect a REV Robotics Touch sensor to one of the digital I/O ports.

Note that in the case of the REV Robotics Touch Sensor, the device has a connector port for a 4-pin sensor cable. However, the device only needs to connect to one of the two available digital I/O pins. For the REV Robotics Touch Sensor, the second digital I/O in the port is the one that gets connected when a standard REV Robotics 4-pin JST PH cable is used. For the “0-1” port, it is the pin labeled “1” that gets connected through the 4-pin cable. Similarly, for the “2-3” port, it is the pin labeled “3” that gets connected through the 4-pin cable.

Connecting a Touch Sensor to the Expansion Hub (Time Needed to Complete Task: 2.5 minutes)

Step 1:

Connect the one end of the 4-pin JST PH cable to the REV Robotics Touch sensor.



Step 2:

Plug the other end of the 4-pin JST PH cable to digital I/O port labeled “0-1” on the Expansion Hub.



8 Configuring Your Hardware on the Robot Controller

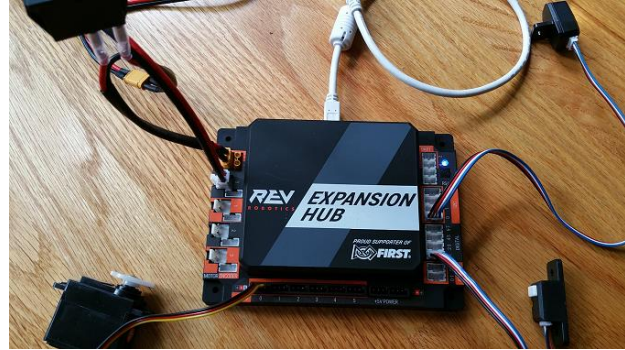
Before you can communicate with the motor, servo and sensors that are connected to the Expansion Hub, you first must create a configuration file on your Robot Controller smartphone, so that the Robot Controller will know what hardware is available on the Expansion Hub.

Creating a Configuration File on the Robot Controller (Time Needed to Complete Task: 20 minutes)

Connecting the Phone to the Expansion Hub

Step 1:

Power on the Expansion Hub by turning on the power switch.

**Step 2:**

Plug the Type B Mini end of the USB cable into the USB mini port on the Expansion Hub.

**Step 3:**

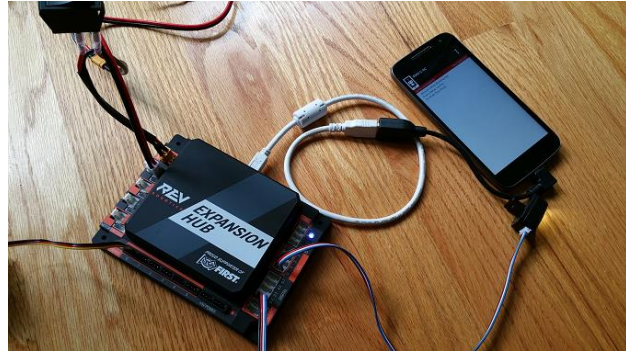
Plug the Type A end of the USB cable into the OTG adapter.



Step 4:

Verify that your Robot Controller smartphone is powered on and unlocked. Plug in the USB Micro OTG adapter into the OTG port of the Robot Controller phone.

Note that when the OTG adapter is plugged into the smartphone, the phone will detect the presence of the Expansion Hub and launch the Robot Controller app.

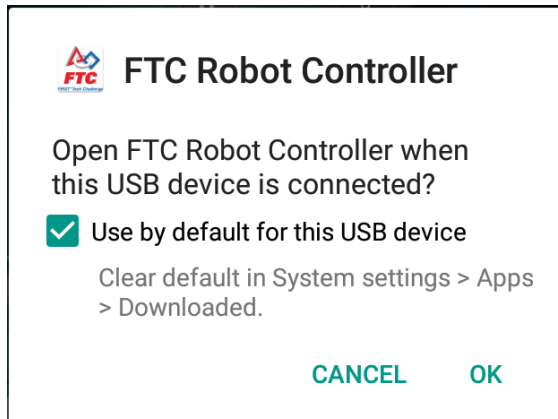
**Step 5:**

The first time you connect the Robot Controller smartphone to the Expansion Hub, the Android operating system should prompt you to ask if it is OK to associate the newly detected USB device (which is the Expansion Hub) with the FTC Robot Controller app.

Important Information!

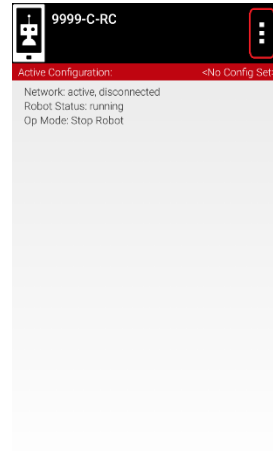
You might be prompted multiple times to associate the USB hardware with the FTC Robot Controller. Whenever you are prompted by your phone with this message, you should **always** select the “Use by default for this USB device” option and hit the “OK” button to associate the USB device with the FTC Robot Controller app.

If you fail to make this association, then the Robot Controller app might not reliably connect to this Expansion Hub the next time you turn your system on.

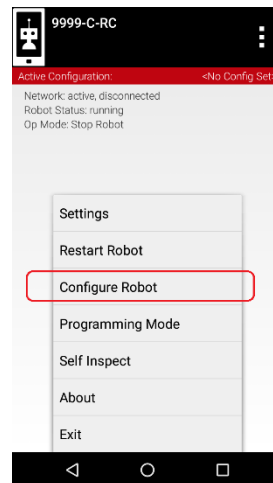


Step 6:

Touch the three vertical dots in the upper right hand corner of the Robot Controller. This will launch a pop-up menu.

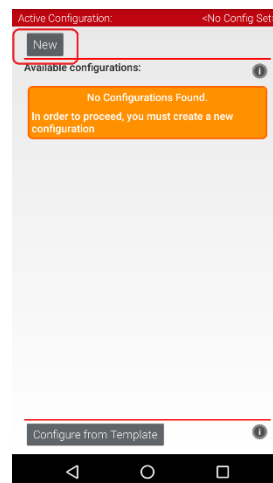
**Step 7:**

Select **Configure Robot** from the pop up menu to display the Configuration screen.

**Step 8:**

If your Robot Controller does not have any existing configuration files, the screen will display a message indicating that you need to create a file before proceeding.

Hit the “New” button to create a new configuration file for your Robot Controller



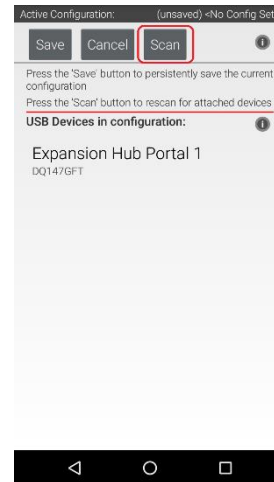
Step 9:

When the new configuration screen appears, the Robot Controller app will do a scan of the USB bus to see what devices are connected to the phone.

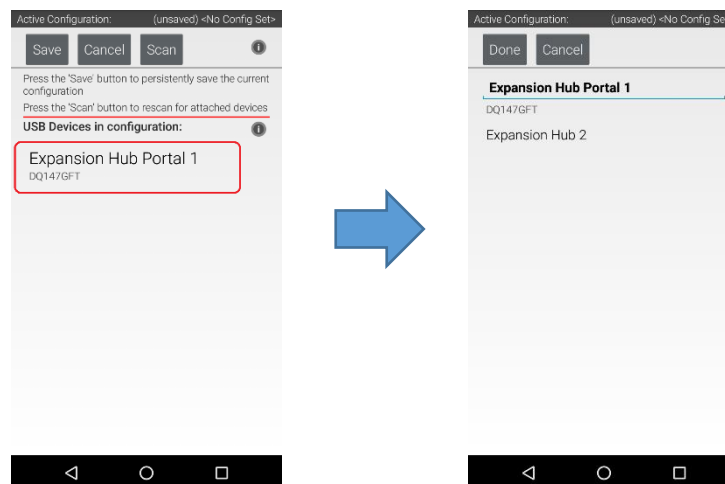
It will display the devices that it found in a list underneath the words “USB Devices in configuration.” You should see an entry that says something like “Expansion Hub Portal 1” in the list.

Your Expansion Hub is listed as a **Portal** because it is directly connected to the Robot Controller phone through the USB cable.

If you do not see your Expansion Hub Portal listed, check the wired connections and then press the **Scan** button one or two times to see if the phone detects the device on a re-scan of the USB bus.

**Step 12:**

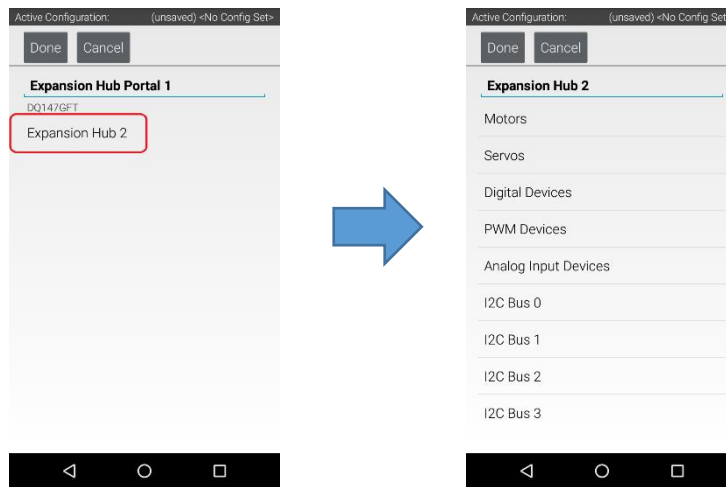
Touch the Portal listing (“Expansion Hub Portal 1” in this example) to display what Expansion Hubs are connected through this Portal.



Since we only have a single Expansion Hub connected, we should only see a single Expansion Hub configured (“Expansion Hub 2” in this example).

Step 13:

Touch the Expansion Hub listing (“Expansion Hub 2” in this example) to display the Input/Output ports for that device.

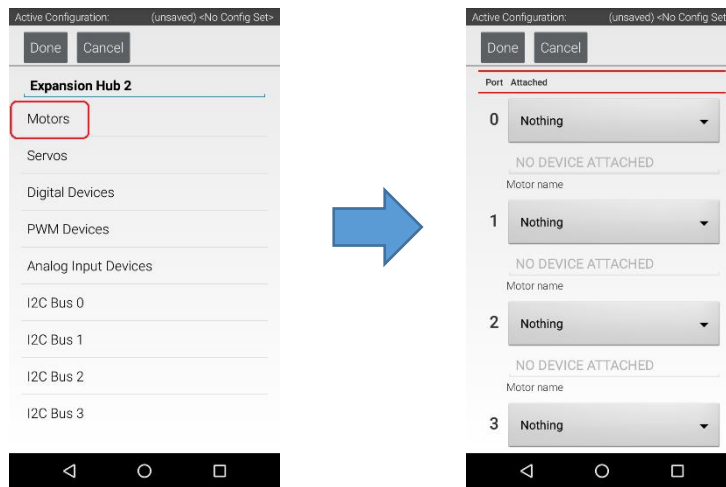


The screen should switch and list all the motor, servo and sensor ports that are available on the selected Expansion Hub.

Configuring the DC Motor

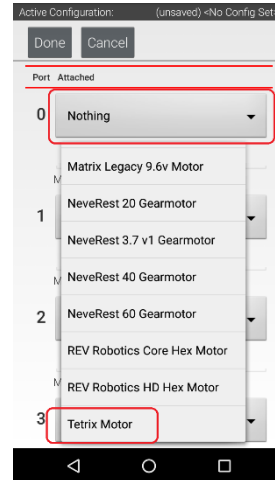
Step 14:

Touch the word **Motors** on the screen to display the Motor Configuration screen.

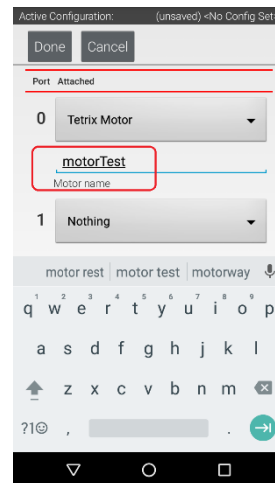


Step 15:

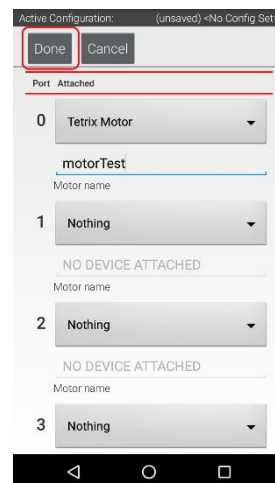
Since we installed our motor onto port #0 of the Expansion Hub, use the dropdown control for port 0 to select the motor type (Tetrix Motor for this example).

**Step 16:**

Use the touch screen keypad to specify a name for your motor ("motorTest" in this example).

**Step 17:**

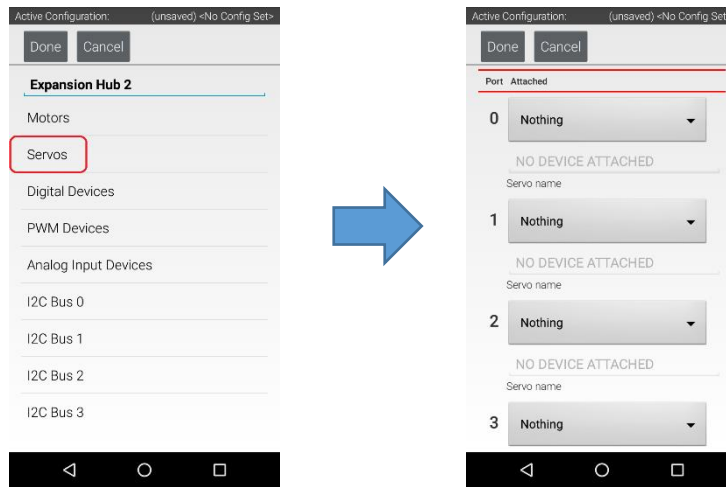
Press the **Done** button to complete the motor configuration. The app should return to the previous screen.



Configuring the Servo

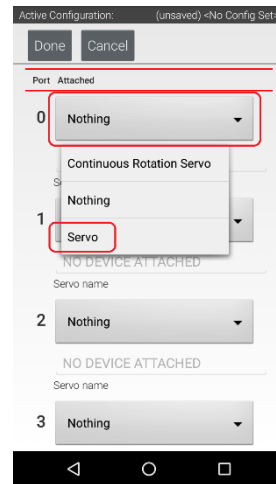
Step 18:

Touch on the word **Servos** on the screen to display the Servo Configuration screen.



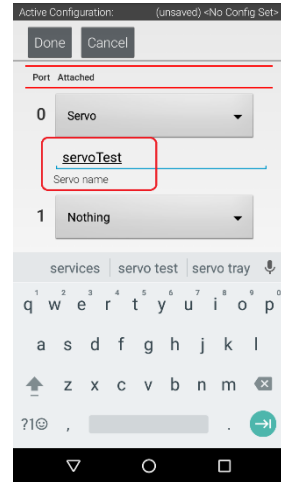
Step 19:

Use the dropdown control to select “Servo” as the servo type for port #0.

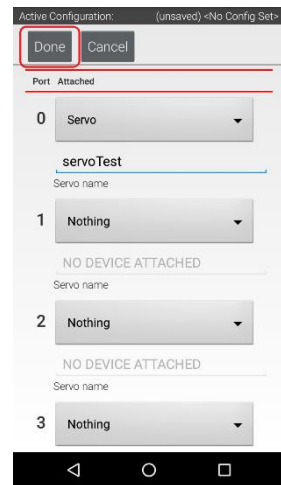


Step 20:

Use the touch pad to specify the name of the servo (“servoTest” for this example) for port #0.

**Step 21:**

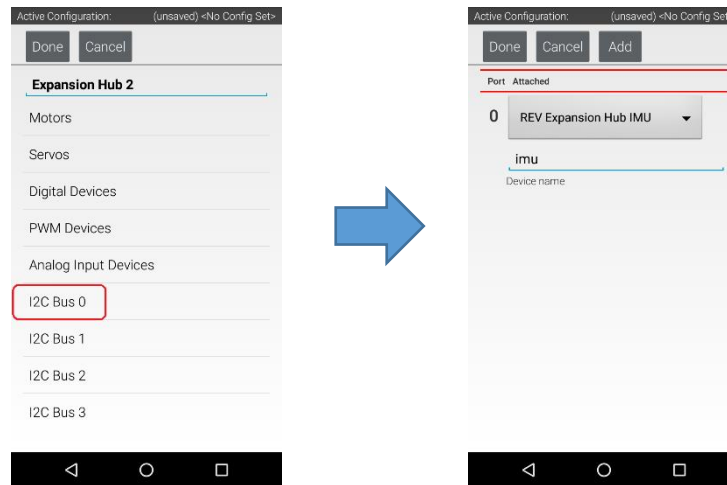
Press the **Done** button to complete the servo configuration. The app should return to the previous screen.



Configuring the Color Sensor

Step 22:

Touch the words **I2C Bus 0** on the screen to launch the I2C configuration screen for this I2C bus.



The Expansion Hub has four independent I2C buses, labeled “0” through “3”. In this example, since you connected the Color Sensor to the port labeled “0”, it resides on I2C Bus 0.

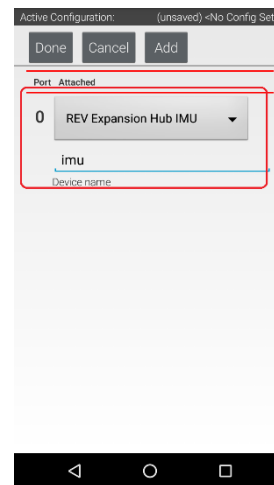
Step 23:

Look at the I2C Bus 0 screen. There should already be a sensor configured for this bus.

The Expansion Hub has its own built-in inertial measurement unit (IMU) sensor. This sensor can be used to determine the orientation of a robot, as well as measure the accelerations on a robot.

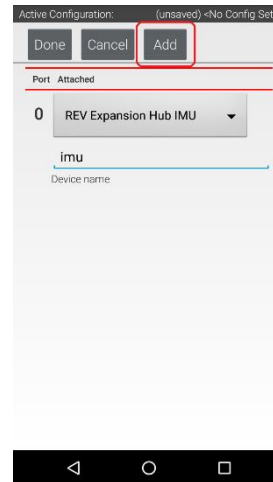
The built-in IMU is internally connected to I2C Bus 0 on each Expansion Hub. Whenever you configure an Expansion Hub using the Robot Controller, the app automatically configures the IMU for I2C Bus 0.

You will need to add another I2C device for this bus to be able to configure the color sensor.

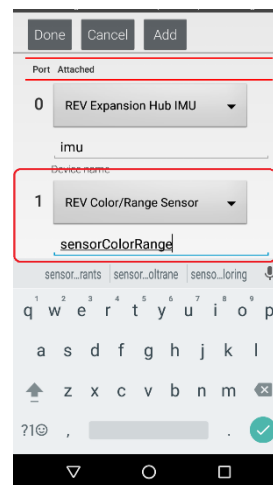


Step 24:

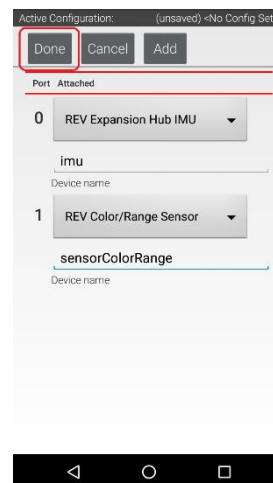
Press the **Add** button to add another I2C device to this bus.

**Step 25:**

Select “REV Color/Range Sensor” from the dropdown selector for this new device. Use the touchscreen keyboard to name this device “sensorColorRange”.

**Step 26:**

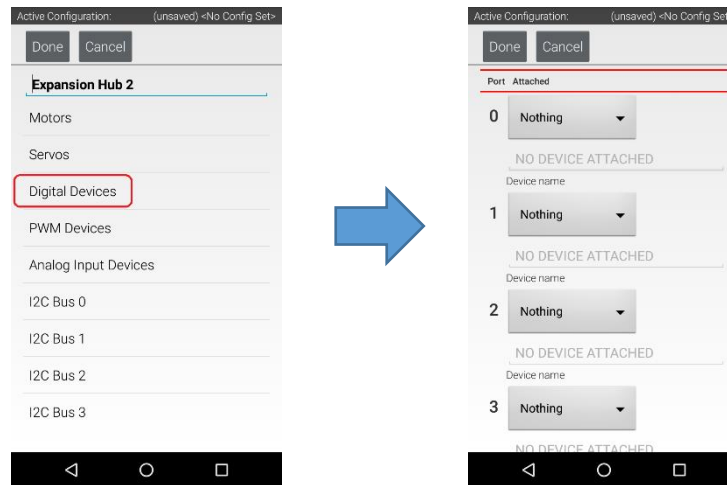
Press the **Done** button to complete the I2C sensor configuration. The app should return to the previous screen.



Configuring the Touch Sensor

Step 27:

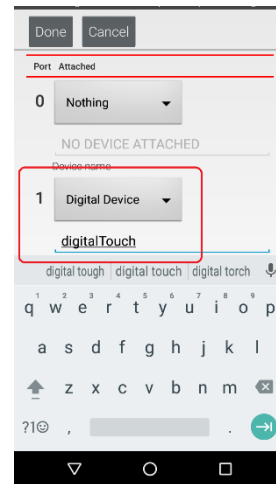
Touch the words **Digital Devices** on the screen to launch the Digital I/O configuration screen.



Step 28:

Use the touch screen to add a “Digital Device” for port #1 and name the device “digitalTouch”.

Notice that we are configuring the Touch Sensor on port #1 instead of port #0. This is because when the REV Robotics Touch Sensor is connected to a digital port using a standard 4-wire JST sensor cable, it is the second digital pin that is connected. The first pin remains disconnected.



Step 29:

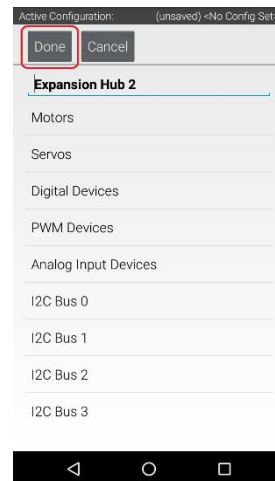
Press the **Done** button to return to the previous screen.



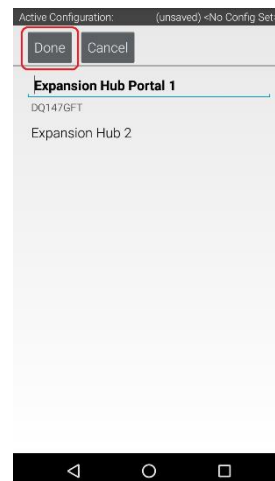
Saving the Configuration Information

Step 30:

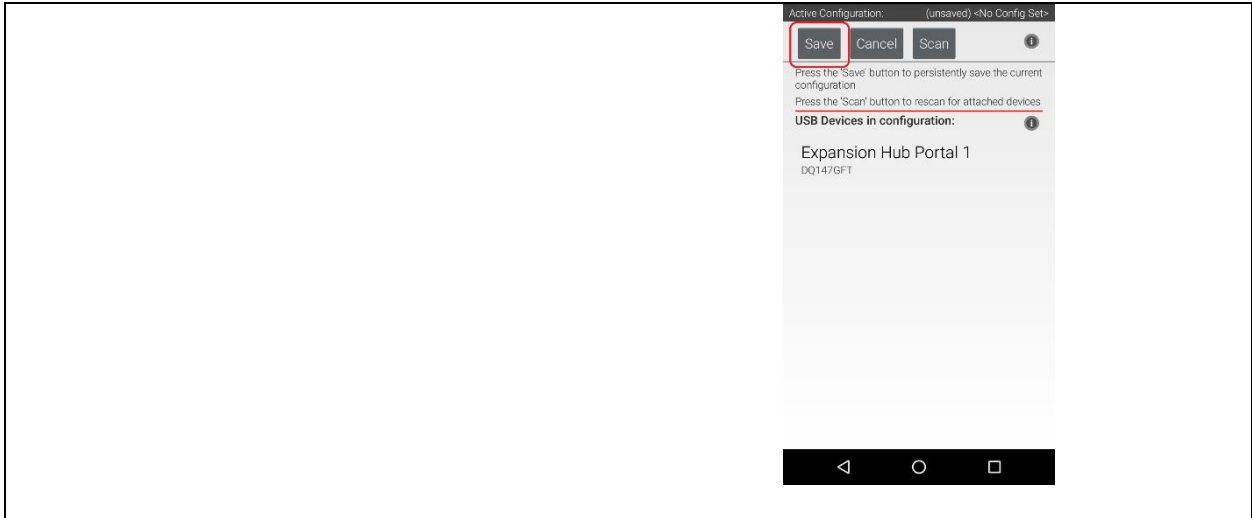
Press the **Done** button to go up one level in the configuration screens.

**Step 31:**

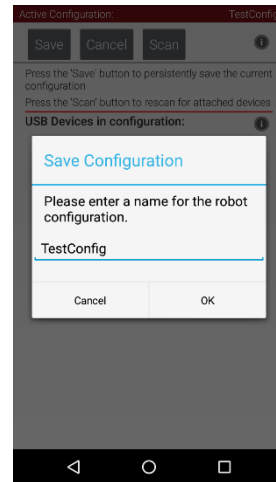
Press the **Done** button again to return to the highest level in the configuration screens.

**Step 32:**

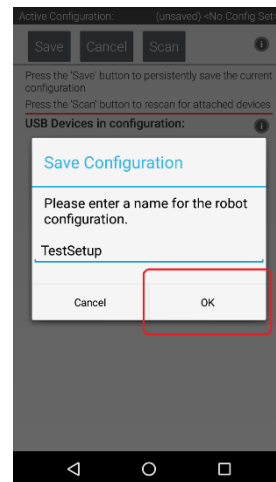
Press the **Save** button.



Step 33:
When prompted, specify a configuration file name using the touchscreen's keypad (use "TestConfig" for this example).

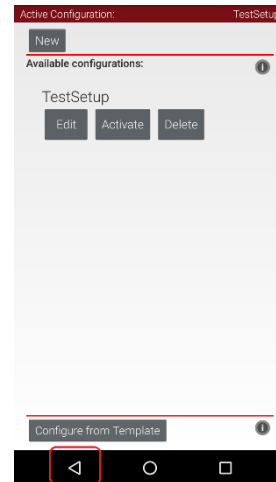


Step 34:
Press the **OK** button to save your configuration information using that file name.

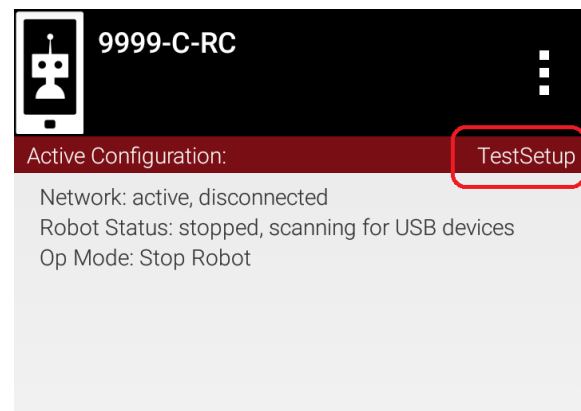


Step 35:

After the configuration file has been saved, touch the Android back-arrow button to return to the main screen of the app.

**Step 36:**

Verify that the configuration file is the active configuration file on the main Robot Controller screen.



9 Installing Android Studio

9.1 Downloading & Installing Android Studio

9.1.1 Android Developer Website

Android Studio is distributed freely by Google, and 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>

Android Studio is available on the Windows, MacOS, and Linux operating systems.

9.1.2 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>

9.1.3 Java Development Kit

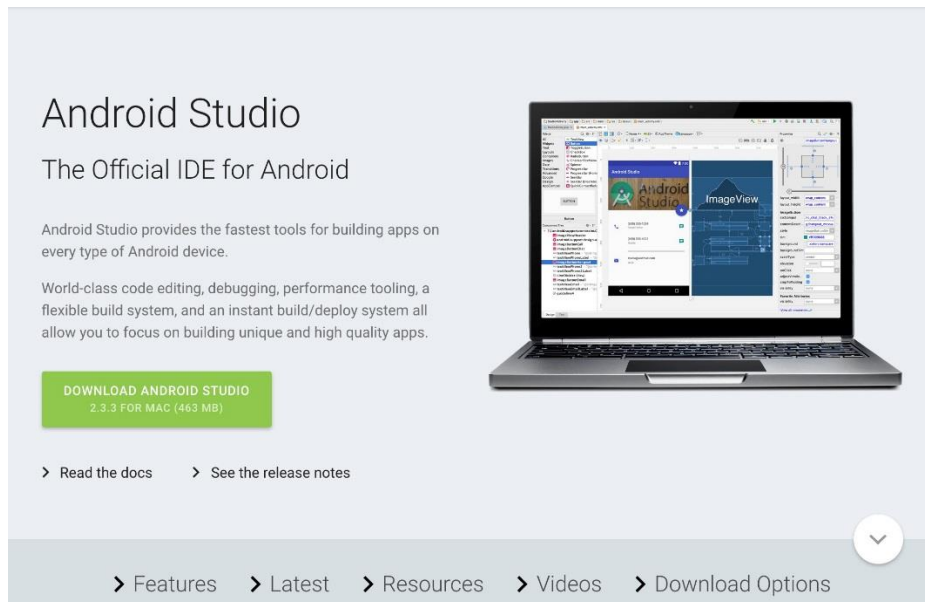
Earlier versions of Android Studio used to require that the user install the Java Development Kit software separately. Current versions of Android Studio incorporate the Java development software as part of the entire install package. It is no longer necessary (or recommended) to install the Java Development Kit separately.

9.1.4 Downloading and Installing Android Studio

Once you have verified that your laptop satisfies the minimum system requirements, you can go to the Android developer's website to download and install Android Studio:

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

Click on the green "DOWNLOAD ANDROID STUDIO" button to start the download process.



Accept the license terms and then push the blue "DOWNLOAD ANDROID STUDIO" button to download the software.



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

9.2 Disabling Android Studio Instant Run

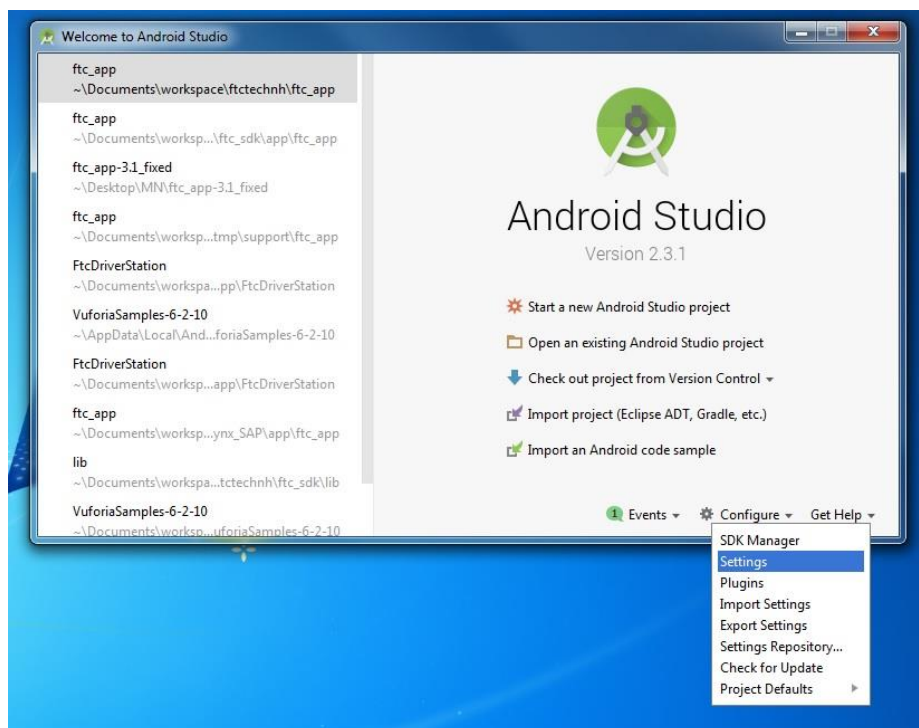
9.2.1 What is Instant Run?

If you are an Android Studio user, one of the most important steps to take is to disable Android Studio Instant Run. Instant Run is a feature that is designed to streamline the development process by reducing the time to apply code changes to your app. Unfortunately, Instant Run is limited in function and when used with the FIRST Tech Challenge Android Studio project folder, can cause severe and difficult-to-troubleshoot problems.

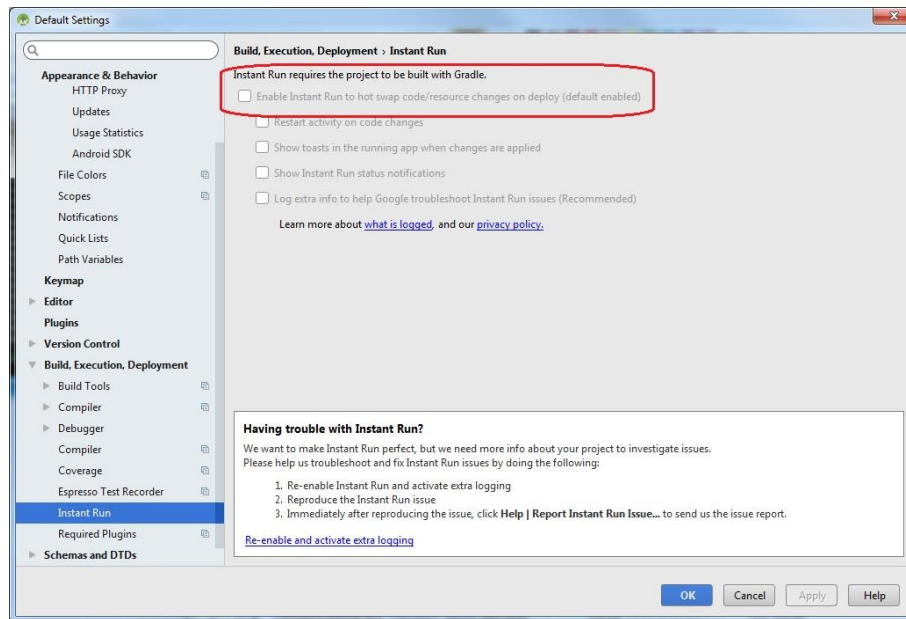
Teams who use Android Studio **must disable Instant Run**.

9.2.2 Locating Instant Run Settings

When you first launch Android Studio a Welcome screen should appear. You can navigate to the Instant Run Settings from this Welcome screen by selecting the "Configure->Settings" item from the "Configure" dropdown list in the lower right hand corner of the screen.



On the left hand side of the Settings window, there should be a category called "Build, Execution, Deployment". Within this category, click on the "Instant Run" subcategory to display the Instant Run settings for your Android Studio installation. By default, Instant Run is enabled when you first install Android Studio. Uncheck the "Enable Instant Run to hot swap code/resource changes on deploy (default enabled)" option and then click on the "OK" button to disable Instant Run.



9.2.3 Additional Information

The Google Android Developer website has additional information about Instant Run. It also has instructions on how to disable this feature:

<https://developer.android.com/studio/run/index.html#disable-ir>

9.3 *Downloading the Android Studio Project Folder*

9.3.1 Downloading the Android Studio Project Folder

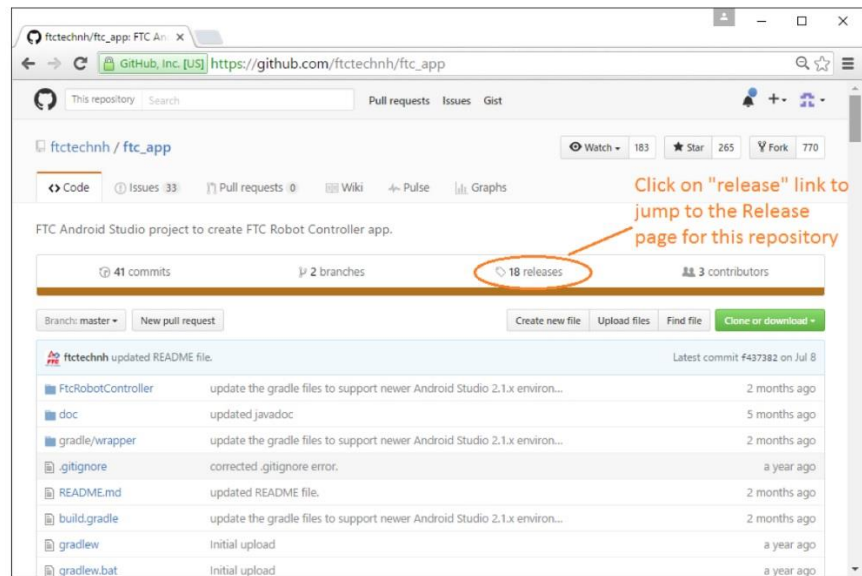
The FTC SDK can be downloaded from a GitHub repository. GitHub is a web-based version control company that lets individuals and organizations host content online. In order to access the FTC software, you will need to have a GitHub account. You can create one for free by visiting the GitHub website:

<https://github.com/>

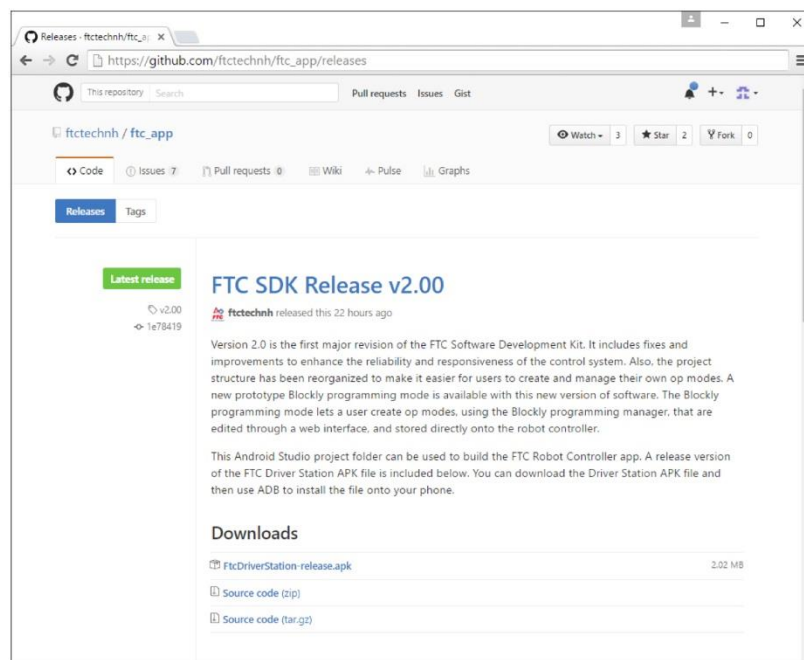
Once you have created a GitHub account, you can visit the FTC public repository for the FTC software:

https://github.com/ftctechnh/ftc_app

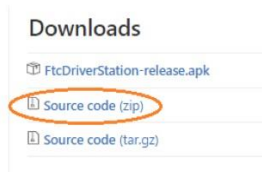
Important note for advanced GitHub users: this tutorial assumes that the user is a novice with respect to using GitHub and the git version control software. If you are a GitHub power user, you can use git to create a local copy of the ftc_app repository. This document, however, does not explain how to use git to access the repository. It provides instructions on downloading the repository as a .ZIP file instead.



From the main repository web page, click on the “releases” link to jump to the Releases page for the repository. The Releases page should list the available software releases for the repository. The latest release should be displayed near the top of the page.



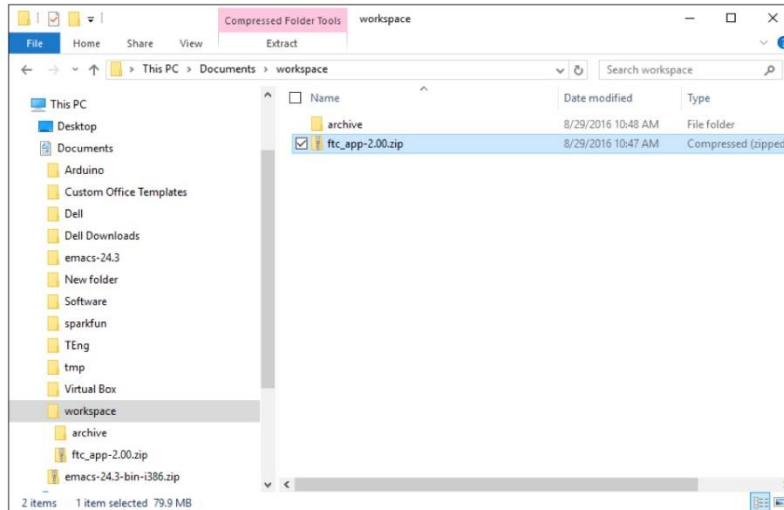
Each software release should include a **Downloads** section that you can use to download the software that you will need to program your robot.



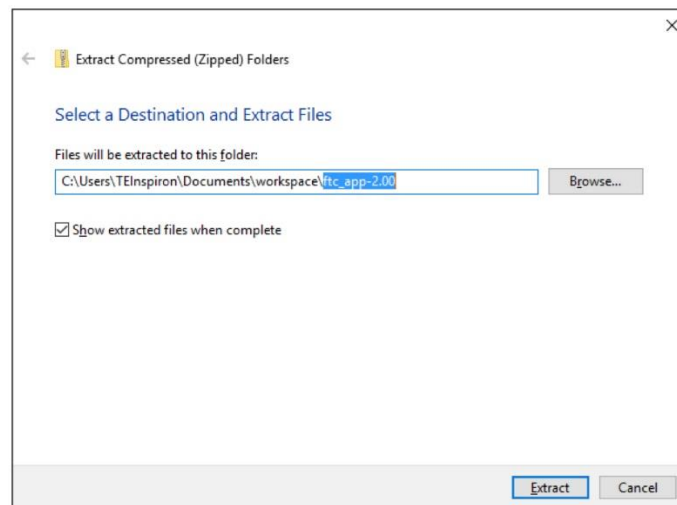
Click on the Source code (zip) link to download the compressed Android Studio project folder.

9.3.2 Extracting the Contents of the Archived Project File

Once you have downloaded the archived (.ZIP) project file you can move this file to the location of your choice.



Before you can import the FTC project into Android Studio, you must first extract the contents of the archived project file. For Windows users, right mouse click on the file and select “Extract All” from the pop up menu. Windows should prompt you to select a destination for the extracted project folder. The dialog that appears should look similar to the one show in the figure below.



Highlight the suggested name for the destination folder (in the figure above, the suggested name is “ftc_app-2.00”) and change the destination folder name into something more user friendly. In this example, we will change the name of the destination folder to “mycopy”.

Files will be extracted to this folder:

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



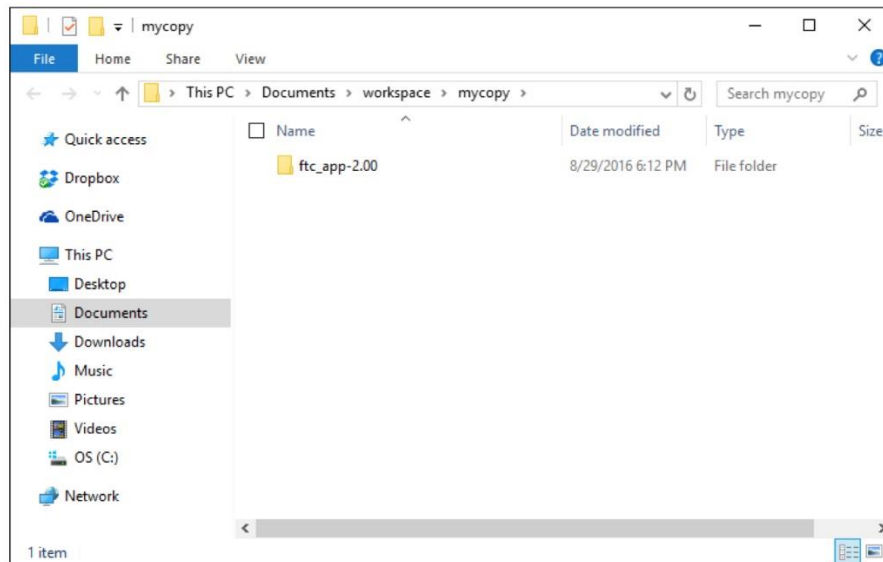
Rename destination folder...



Files will be extracted to this folder:

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

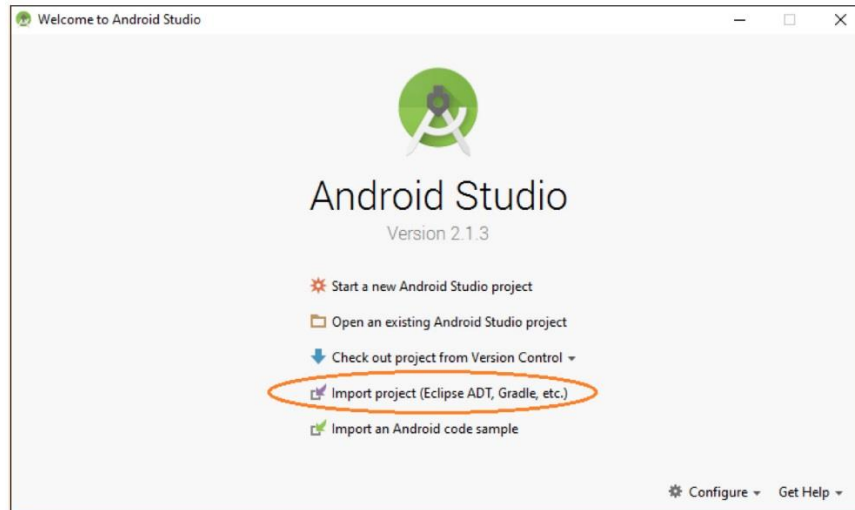
After you have renamed the destination folder, extract the contents of the archive to the folder. After the extraction process is complete, verify that the project folder was successfully extracted to its target destination.



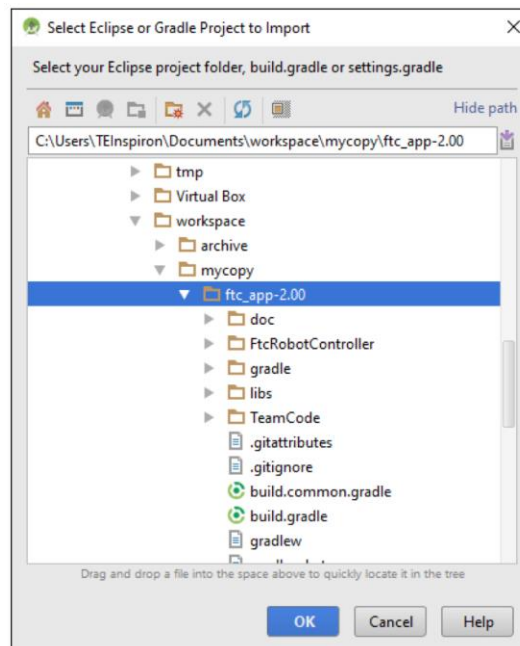
Once you have successfully extracted the contents of the archived file, you are ready to import the FTC project into Android Studio.

9.3.3 Importing the FTC Project into Android Studio

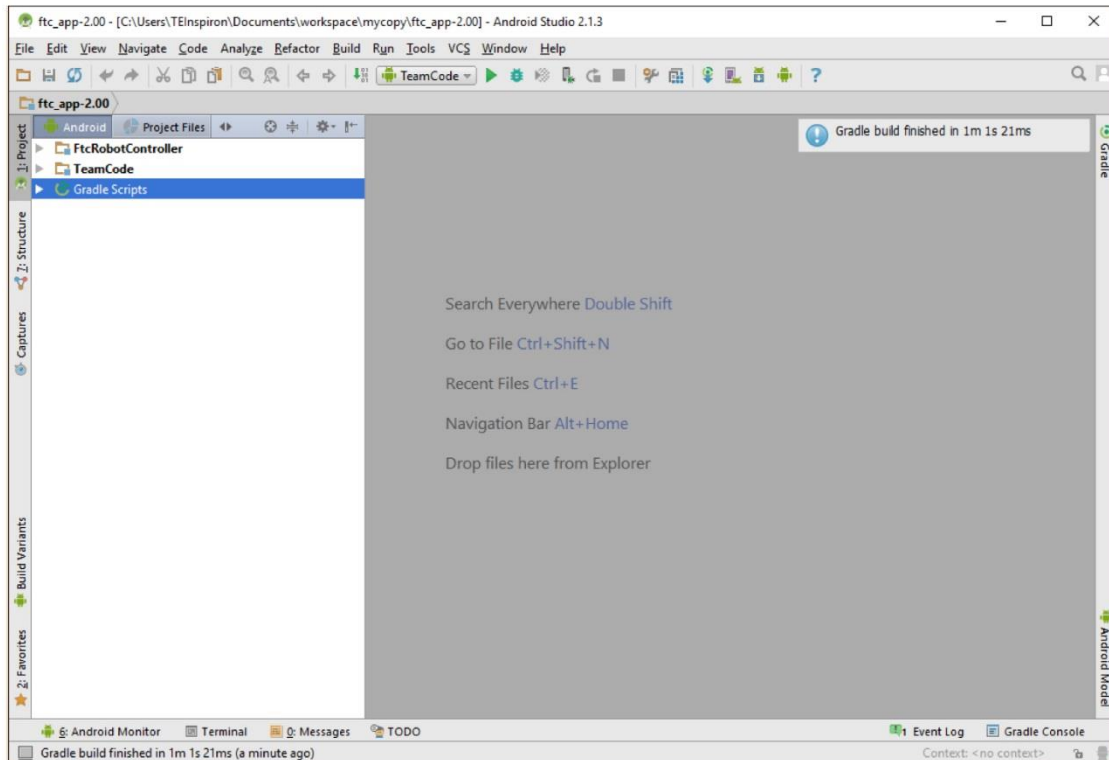
In order to import the FTC Project, you will need to launch the Android Studio software on your computer. On the main Android Studio Welcome screen, select the option to “Import project (Eclipse, ADT, Gradle, etc.)” to begin the import process.



Android Studio should prompt you to select the project folder that you would like to import. Use the file browser in the pop up dialog box to locate and then select the folder that you extracted in a section of this document. Make sure you select the extracted project folder (and not the .ZIP file which might have a similar name to the extracted folder). Hit the “OK” button to import the selected project into Android Studio.



In the figure above the project folder called “ftc_app-2.00” is selected to be imported into Android Studio. It might take Android Studio several minutes to import the project. Once the project has been successfully imported, the screen should look similar to the one depicted in the image below.



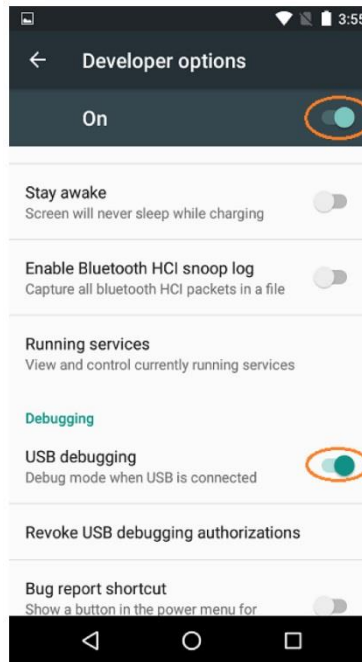
10 Writing an Op Mode with Android Studio

10.1 Enabling Developer Options

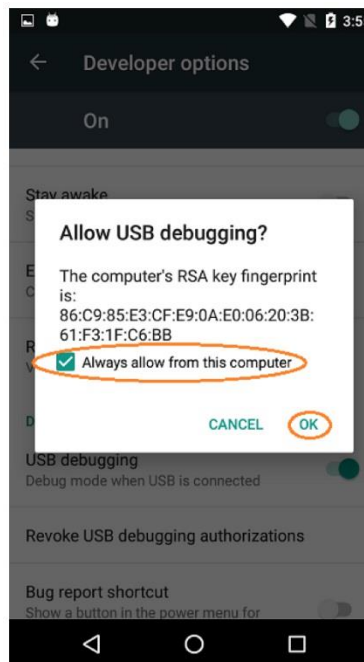
After you have configured your Android phone, you will also have to make sure that your phone is in developer mode before you will be able to install apps onto the phone using the tools that are included with Android Studio. The Android Developer website contains information on how to enable Developer Options onto your phone. If you visit the following link and read the section entitled “Enabling On-device Developer Options” you will see that you can enable Developer Options on your Android phone by going to Settings>About phone on the phone, and then tapping the Build number seven times.

<https://developer.android.com/studio/run/device.html#device-developer-options>

In order to be able to use the Android Studio tools to install apps onto your phone, you will need to make sure that the Developer Options and USB debugging are enabled for both of your phones.



When you first connect a phone to your computer with Android Studio running, the phone might prompt you if it is OK to allow the computer to have USB debugging access to the phone. If this happens, make sure that you check the “Always allow from this computer” option and hit the OK button to allow USB debugging.



10.2 Creating & Running an Op Mode

10.2.1 What's an Op Mode?

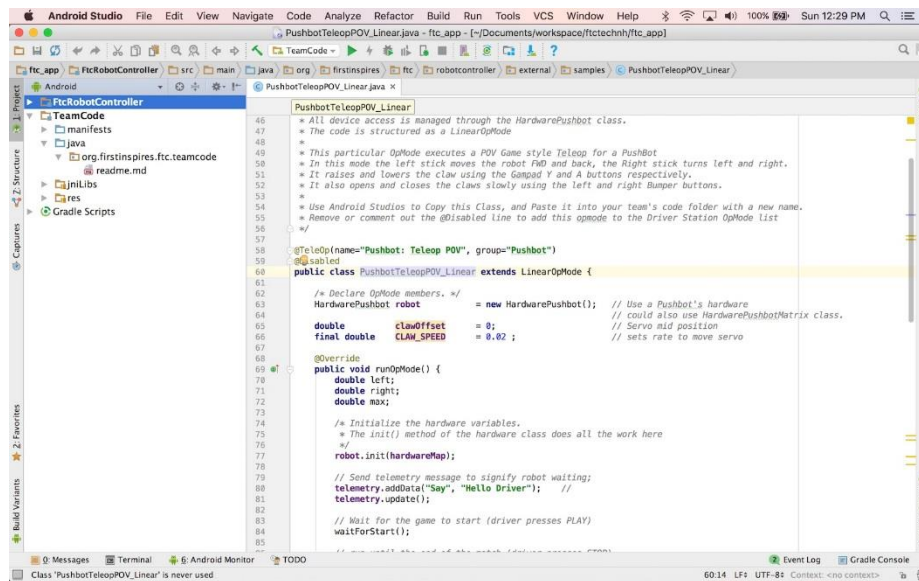
During a typical FIRST Tech Challenge match, a team's robot must perform a variety of tasks to score points. For example, a team might want their robot to follow a white line on the competition floor and then score a game element into a goal autonomously during a match. Teams write programs called “op modes” (which stands for “operational

modes”) to specify the behavior for their robot. These op modes run on the Robot Controller phone after being selected on the Driver Station phone.

Teams who are participating in the *FIRST* Tech Challenge have a variety of programming tools that they can use to create their own op modes. This document explains how to use Google’s Android Studio integrated development environment (IDE) to write an op mode for an FTC robot.

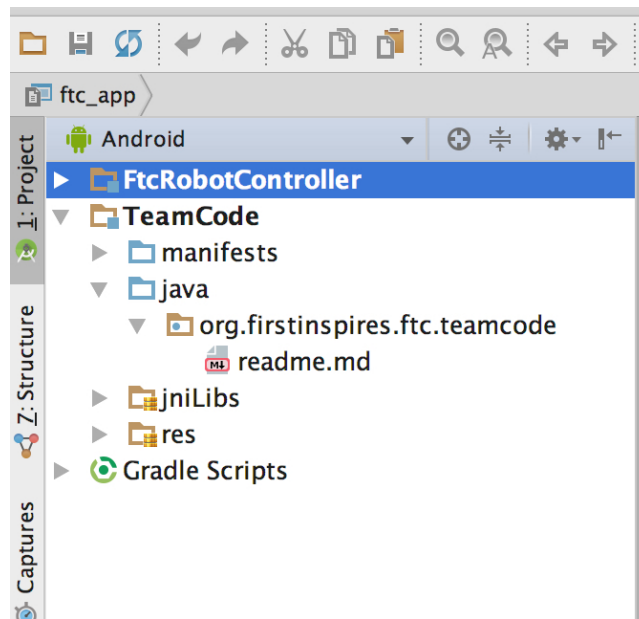
10.2.2 Android Studio

Android Studio is an advanced integrated development environment for creating Android apps. This tool is the same tool that professional Android app developers use. Android Studio is only recommended for advanced users who have extensive Java programming experience.



10.2.3 TeamCode Module

If you successfully imported the FTC Android Studio project folder, you will see on the project browser an Android module named "TeamCode". The FTC Android Studio project folder will be used to build a version of the FTC Robot Controller app that includes the custom op modes that you will write to control your competition robot.



When you create your classes and op modes, you will create them in the `org.firstinspires.ftc.teamcode` package that resides in the TeamCode module. This package is reserved for your use within the FTC Android Studio project folder.

10.2.4 Javadoc Reference Information

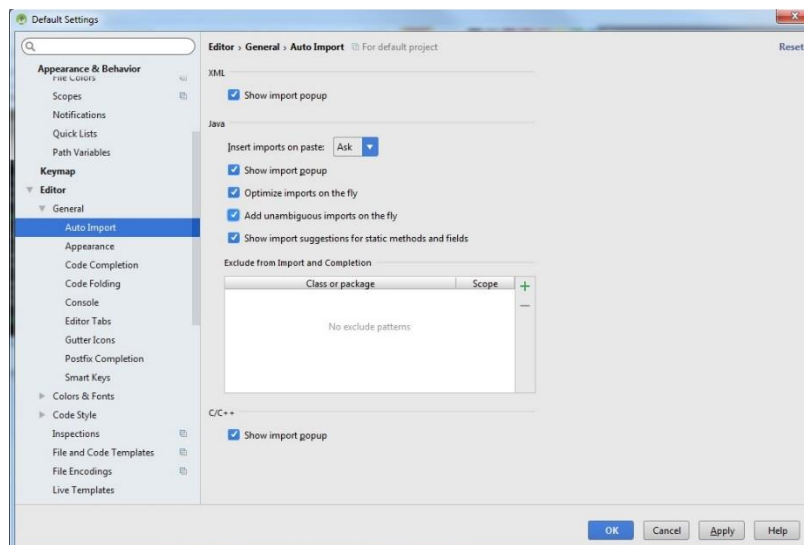
The Javadoc reference documentation for the FTC SDK is available online. Visit the following URL to view the FTC SDK documentation:

http://ftctechnh.github.io/ftc_app/doc/javadoc/index.html

10.2.5 Enabling Auto Import

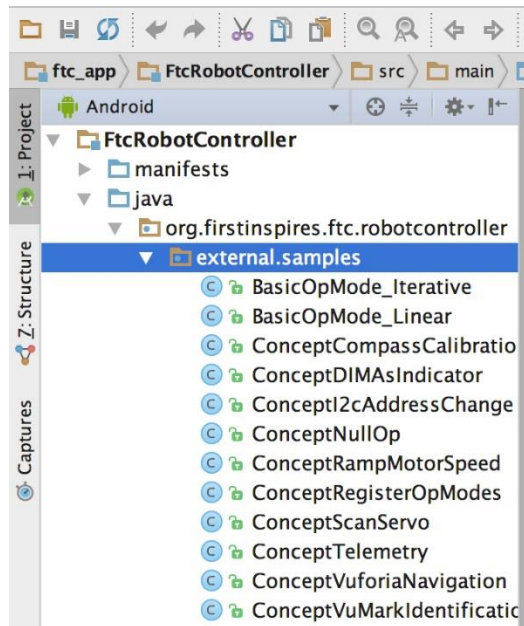
The auto import feature of Android Studio is a convenient function that helps save time as you write your op mode. If you would like to enable this feature, select the Editor->General->Auto Import item from the Android Studio Settings screen. This will display the editor's auto import settings.

Check the "Add unambiguous imports on the fly" so that Android Studio will automatically add the required import statements for classes that you would like to use in your op mode.



10.2.6 Sample Op Modes

A great way to learn how to program a robot is to examine the sample op modes that are included with the FTC Android Studio project folder. You can locate these files in the FtcRobotController module in the package "org.firstinspires.ftc.robotcontroller.external.samples".



If you would like to use a sample op mode, copy it from the "org.firstinspires.ftc.robotcontroller.external.samples" package and move it to the "org.firstinspires.ftc.teamcode".

In your newly copied op mode, look for the following annotation,

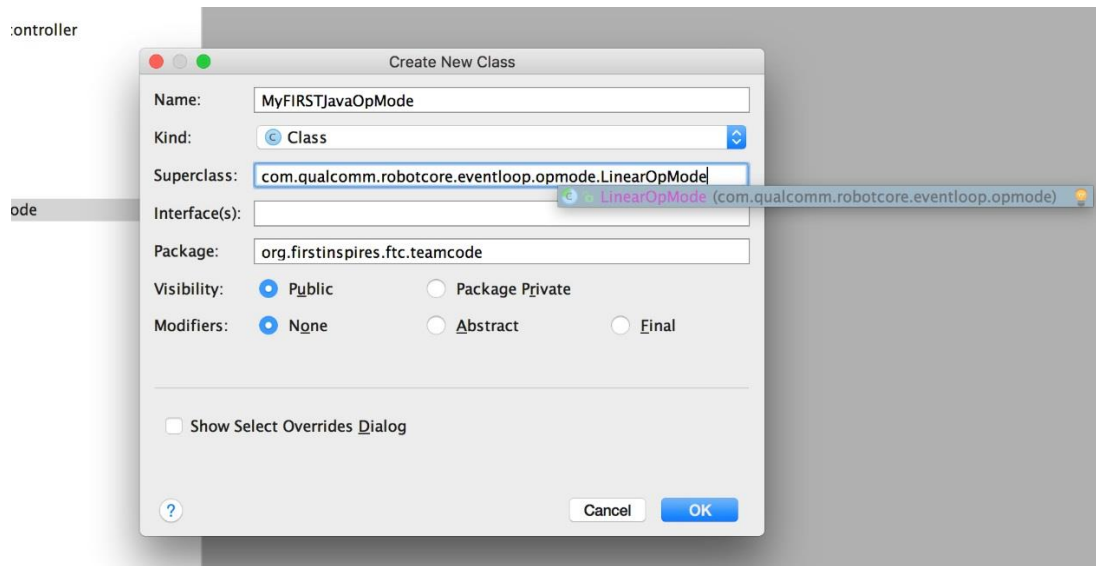
```
@Disabled
```

and comment out this line to enable the op mode and allow it to be run on the Robot Controller:

```
//@Disabled
```

10.2.7 Creating Your FIRST Op Mode

Right mouse click on the "org.firstinspires.ftc.teamcode" package and select New->Java Class from the pop-up menu. The Create New Class dialog box appear. Specify the name of the new class as "MyFIRSTJavaOpMode" and specify as its superclass the class LinearOpMode which is in the package "com.qualcomm.robotcore.eventloop.opmode".



Press the OK button to create the new class. The source code for the new class should appear in the editing pane of the Android Studio user interface.

```
MyFIRSTJavaOpMode.java x
MyFIRSTJavaOpMode
1 package org.firstinspires.ftc.teamcode;
2
3 import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
4
5 /**
6  * Created by tom on 9/19/17.
7  */
8
9 public class MyFIRSTJavaOpMode extends LinearOpMode {
10 }
11
```

Modify the main portion of your op mode so that it looks like the following code (note that the package definition and some import statements have been omitted in the following source code):

```

@TeleOp
public class MyFIRSTJavaOpMode extends LinearOpMode {
    private Gyroscope imu;
    private DcMotor motorTest;
    private DigitalChannel digitalTouch;
    private DistanceSensor sensorColorRange;
    private Servo servoTest;

    @Override
    public void runOpMode() {
        imu = hardwareMap.get(Gyroscope.class, "imu");
        motorTest = hardwareMap.get(DcMotor.class, "motorTest");
        digitalTouch = hardwareMap.get(DigitalChannel.class, "digitalTouch");
        sensorColorRange = hardwareMap.get(DistanceSensor.class, "sensorColorRange");
        servoTest = hardwareMap.get(Servo.class, "servoTest");

        telemetry.addData("Status", "Initialized");
        telemetry.update();
        // Wait for the game to start (driver presses PLAY)
        waitForStart();

        // run until the end of the match (driver presses STOP)
        while (opModeIsActive()) {
            telemetry.addData("Status", "Running");
            telemetry.update();
        }
    }
}

```

We will use this source code as the framework for your first op mode. Note that Android Studio automatically saves your source code as you are editing it.

Congratulations! You've written an op mode. It does not do much, but we will modify it to make it more useful.

10.2.8 Examining the Structure of Your Op Mode

It can be helpful to think of an op mode as a list of tasks for the Robot Controller to perform. For a linear op mode, the Robot Controller will process this list of tasks sequentially. Users can also use control loops (such as a while loop) to have the Robot Controller repeat (or iterate) certain tasks within a linear op mode.



If you think about an op mode as a list of instructions for the robot, this set of instructions that you created will be executed by the robot whenever a team member selects the op mode called “MyFIRSTJavaOpMode” from the list of available op modes for this Robot Controller.

Let’s look at the structure of your newly created op mode

At the start of the op mode there is an annotation that occurs before the class definition. This annotation states that this is a tele-operated (i.e., driver controlled) op mode:

```
@TeleOp
```

If you wanted to change this op mode to an autonomous op mode, you would replace the “@TeleOp” with an “@Autonomous” annotation instead.

You can see from the sample code that an op mode is defined as a Java class. In this example, the op mode name is called “MyFIRSTJavaOpMode” and it inherits characteristics from the LinearOpMode class.

```
public class MyFIRSTJavaOpMode extends LinearOpMode {
```

You can also see that the sample op mode includes five private member variables for this op mode. These variables will hold references to the five devices that you configured earlier for the configuration file of your Robot Controller.

```
private Gyroscope imu;
private DcMotor motorTest;
private DigitalChannel digitalTouch;
private DistanceSensor sensorColorRange;
private Servo servoTest;
```

Next, there is an overridden method called runOpMode. Every op mode of type LinearOpMode must implement this method. This method gets called when a user selects and runs the op mode.

```
@Override
public void runOpMode() {
```

At the start of the runOpMode method, the op mode uses an object named hardwareMap to get references to the hardware devices that are listed in the Robot Controller’s configuration file:

```
imu = hardwareMap.get(Gyroscope.class, "imu");
motorTest = hardwareMap.get(DcMotor.class, "motorTest");
digitalTouch = hardwareMap.get(DigitalChannel.class, "digitalTouch");
sensorColorRange = hardwareMap.get(DistanceSensor.class, "sensorColorRange");
servoTest = hardwareMap.get(Servo.class, "servoTest");
```

The hardwareMap object is available to use in the runOpMode method. It is an object of type HardwareMap class.

Note that when you attempt to retrieve a reference to a specific device in your op mode, the name that you specify as the second argument of the HardwareMap.get method must match the name used to define the device in your configuration file. For example, if you created a configuration file that had a DC motor named “motorTest”, then you must use this same name (it is case sensitive) to retrieve this motor from the hardwareMap object. If the names do not match, the op mode will throw an exception indicating that it cannot find the device.

In the next few statements of the example, the op mode prompts the user to push the start button to continue. It uses another object that is available in the runOpMode method. This object is called telemetry and the op mode uses the addData method to add a message to be sent to the Driver Station. The op mode then calls the update method to send the message to the Driver Station. Then it calls the waitForStart method, to wait until the user pushes the start button on the driver station to begin the op mode run.

```
telemetry.addData("Status", "Initialized");
telemetry.update();
// Wait for the game to start (driver presses PLAY)
waitForStart();
```

Note that all linear op modes should have a waitForStart statement to ensure that the robot will not begin executing the op mode until the driver pushes the start button.

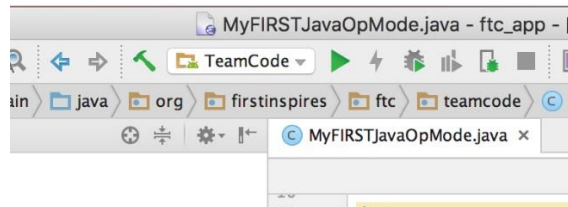
After a start command has been received, the op mode enters a while loop and keeps iterating in this loop until the op mode is no longer active (i.e., until the user pushes the stop button on the Driver Station):

```
// run until the end of the match (driver presses STOP)
while (opModeIsActive()) {
    telemetry.addData("Status", "Running");
    telemetry.update();
}
```

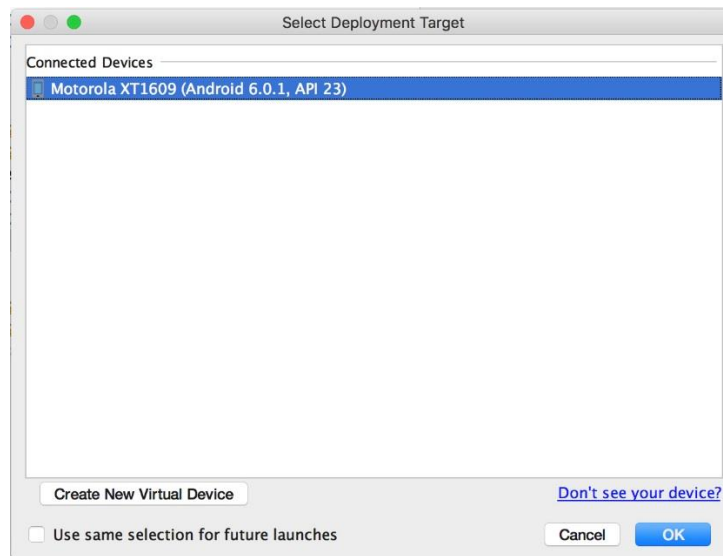
As the op mode iterates in the while loop, it will continue to send telemetry messages with the index of “Status” and the message of “Running” to be displayed on the Driver Station.

10.2.9 Building and Installing Your Op Mode

Verify that the Robot Controller phone is connected to your laptop and that the laptop has USB debugging permission for the phone. Look towards the top of the Android Studio user interface and find the little green Play or Run button (which is represented by a green triangle) next to the words “Team Code”. Press this green button to build the Robot Controller app and to install it onto your phone.

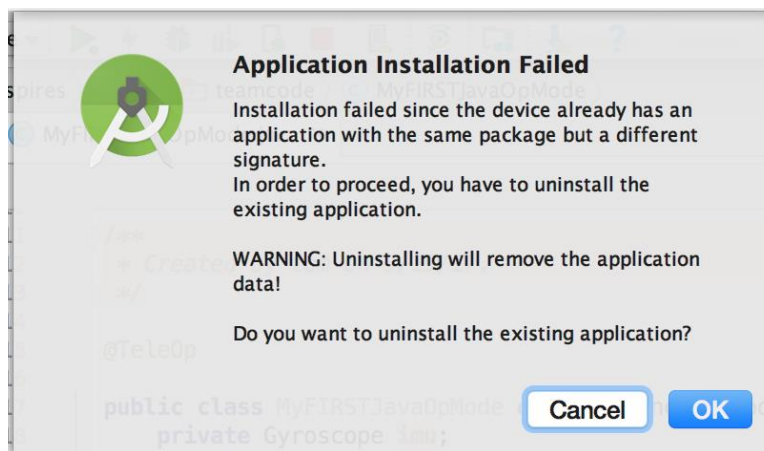


Android Studio should prompt you to select a target device to install the Robot Controller app. Your screen might look something like the image shown below.

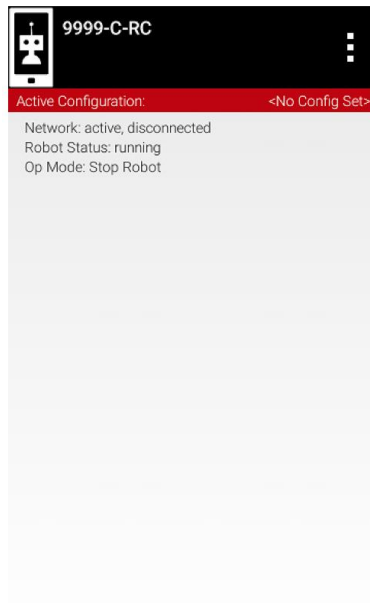


Make sure that you select the correct target device. In the figure above the Motorola phone is selected as the target device. Hit OK to build the APK file and install it on the target device.

Note that if you previously installed a copy of the FTC Robot Controller app from the Google Play store, the installation of your newly built app will fail the first time you attempt to install it. This is because Android Studio detects that the app that you just build has a different digital signature than the official version of the FTC Robot Controller app that was installed from Google Play.



If this happens, Android Studio will prompt you if it's OK to uninstall the previous (official) version of the app from your device and replace it with the updated version of the app. Select "OK" to uninstall the previous version and to replace it with your newly created Robot Controller App (see image above).

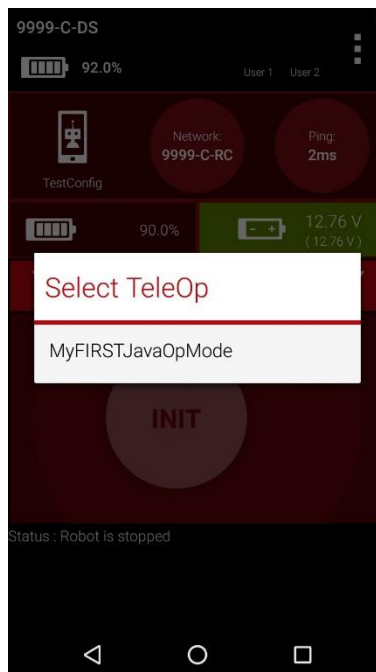


If the installation was successful, the Robot Controller app should be launched on the target Android device.

10.2.10 Running Your Op Mode

If you successfully built and installed your updated Android app with your new op mode, then you are ready to run the op mode. Verify that the Driver Station is still connected to the Robot Controller. Since you designated that your example op mode is a tele-operated op mode, it will be listed as a “TeleOp” op mode.

On the Driver Station, use the “TeleOp” dropdown list control to display the list of available op modes. Select your op mode (“MyFIRSTJavaOpMode”) from the list.



Press the INIT button to initialize the op mode.



The op mode will execute the statements in the runOpMode method up to the waitForStart statement. It will then wait until you press the start button (which is represented by the triangular shaped symbol) to continue.



Once you press the start button, the op mode will continue to iterate and send the "Status: Running" message to the Driver Station. To stop the op mode, press the square-shaped stop button.



Congratulations! You ran your first java op mode!

10.2.11 Modifying Your Op Mode to Control a Motor

Let's modify your op mode to control the DC motor that you connected and configured for your REV Expansion Hub. Modify the code for the program loop so that it looks like the following:

```
// run until the end of the match (driver presses STOP)
double tgtPower = 0;
while (opModeIsActive()) {
    tgtPower = -this.gamepad1.left_stick_y;
    motorTest.setPower(tgtPower);
    telemetry.addData("Target Power", tgtPower);
    telemetry.addData("Motor Power", motorTest.getPower());
    telemetry.addData("Status", "Running");
    telemetry.update();
}
```

If you look at the code that was added, you will see that we defined a new variable called target power before we enter the while loop.

```
double tgtPower = 0;
```

At the start of the while loop we set the variable `tgtPower` equal to the negative value of the gamepad1's left joystick:

```
tgtPower = -this.gamepad1.left_stick_y;
```

The object `gamepad1` is available for you to access in the `runOpMode` method. It represents the state of gamepad #1 on your Driver Station. Note that for the F310 gamepads that are used during the competition, the Y value of a joystick ranges from -1, when a joystick is in its topmost position, to +1, when a joystick is in its bottommost position. In the example code above, you negate the `left_stick_y` value so that pushing the left joystick forward will result in a positive

power being applied to the motor. Note that in this example, the notion of forwards and backwards for the motor is arbitrary. However, the concept of negating the joystick y value can be very useful in practice.



The next set of statements sets the power of motorTest to the value represented by the variable tgtPower. The values for target power and actual motor power are then added to the set of data that will be sent via the telemetry mechanism to the Driver Station.

```
motorTest.setPower(tgtPower);
telemetry.addData("Target Power", tgtPower);
telemetry.addData("Motor Power", motorTest.getPower());
```

After you have modified your op mode to include these new statements, build and install the updated app onto your Robot Controller Android device.

10.2.12 Running Your Op Mode with a Gamepad Connected

Your op mode takes input from a gamepad and uses this input to control a DC motor. To run your op mode, you will need to connect a Logitech F310 gamepad to the Driver Station.

Before you connect your gamepad to the phone, verify that the switch on the bottom of the gamepad is set to the “X” position.



Connect the gamepad to the Driver Station using the Micro USB OTG adapter cable.



Your example op mode is looking for input from the gamepad designated as the user or driver #1. Press the Start button and the A button simultaneously on the Logitech F310 controller to designate your gamepad as user #1. Note that pushing the Start button and the B button simultaneously would designate the gamepad as user #2.



If you successfully designated the gamepad to be user #1, you should see a little gamepad icon above the text “User 1” in the upper right hand corner of the Driver Station Screen. Whenever there is activity on gamepad #1, the little icon should be highlighted in green. If the icon is missing or if it does not highlight in green when you use your gamepad, then there is a problem with the connection to the gamepad.

Select, initialize and run your “MyFIRSTJavaOpMode” op mode. It is important to note that whenever you rebuild an op mode, you must stop the current op mode run and then restart it before the changes that you just built take effect.

If you configured your gamepad properly, then the left joystick should control the motion of the motor. As you run your op mode, be careful and make sure you do not get anything caught in the turning motor. Note that the User #1 gamepad icon should highlight green each time you move the joystick. Also note that the target power and actual motor power values should be displayed in the telemetry area on the Driver Station.



11 Controlling a Servo Motor with an Op Mode

In this section, you will modify your op mode to control a servo motor with the buttons of the gamepad.

11.1 What is a Servo Motor?

A servo motor is a special type of motor. A servo motor is designed for precise motion. A typical servo motor has a limited range of motion.

In the figure below, “standard scale” 180-degree servo is shown. This type of servo is popular with hobbyists and with FIRST Tech Challenge teams. This servo motor can rotate its shaft through a range of 180 degrees. Using an electronic module known as a *servo controller* you can write an op mode that will move a servo motor to a specific position. Once the motor reaches this target position, it will hold the position, even if external forces are applied to the shaft of the servo.



Figure 4 – An example of a servo motor that is commonly used on FIRST Tech Challenge robots.⁴

⁴ Image taken from the Pitsco website on 10/17/2016 (see https://c10645061.ssl.cf2.rackcdn.com/product/icongo/icg_39197_180deg servo.jpg).

Servo motors are useful when you want to do precise movements (for example, sweep an area with a sensor to look for a target or move the control surfaces on a remotely controlled airplane).

11.2 Modifying Your Op Mode to Control a Servo

Let's modify your op mode to add the logic required to control a servo motor. For this example, you will use the buttons on the Logitech F310 gamepad to control the position of the servo motor.

With a typical servo, you can specify a target position for the servo. The servo will turn its motor shaft to move to the target position, and then maintain that position, even if moderate forces are applied to try and disturb its position.

For the FIRST Tech Challenge control system, you can specify a target position that ranges from 0 to 1 for a servo. A target position of 0 corresponds to zero degrees of rotation and a target position of 1 corresponds to 180 degrees of rotation for a typical servo motor.



Figure 5 - A typical servo motor can rotate to and hold a position from 0 to 180 degrees.

In this example, you will use the colored buttons on the right side of the F310 controller to control the position of the servo. Initially, the op mode will move the servo to the midway position (90 degrees of its 180-degree range). Pushing the yellow "Y" button will move the servo to the zero-degree position. Pushing the blue "X" button or the red "B" button will move the servo to the 90-degree position. Pushing the green "A" button will move the servo to the 180-degree position.



Figure 6 – The colored buttons on the right side of the gamepad will be used to control servo position.

Modify your op mode to add the following code:

```
// run until the end of the match (driver presses STOP)
double tgtPower = 0;
while (opModeIsActive()) {
    tgtPower = -this.gamepad1.left_stick_y;
    motorTest.setPower(tgtPower);
    // check to see if we need to move the servo.
    if(gamepad1.y) {
        // move to 0 degrees.
        servoTest.setPosition(0);
    } else if (gamepad1.x || gamepad1.b) {
        // move to 90 degrees.
        servoTest.setPosition(0.5);
    } else if (gamepad1.a) {
        // move to 180 degrees.
        servoTest.setPosition(1);
    }
    telemetry.addData("Servo Position", servoTest.getPosition());
    telemetry.addData("Target Power", tgtPower);
    telemetry.addData("Motor Power", motorTest.getPower());
    telemetry.addData("Status", "Running");
    telemetry.update();
}
```

This added code will check to see if any of the colored buttons on the F310 gamepad are pressed. If the Y button is pressed, it will move the servo to the 0-degree position. If either the X button or B button is pressed, it will move the servo to the 90-degree position. If the A button is pressed, it will move the servo to the 180-degree position. The op mode will also send telemetry data on the servo position to the Driver Station.

After you have modified your op mode, you can build it and then run it. Verify that gamepad #1 is still configured and then use the colored buttons to move the position of the servo.

12 Using Sensors

12.1 Color-Distance Sensor

A *sensor* is a device that lets the Robot Controller get information about its environment. In this example, you will use a REV Robotics Color-Distance sensor to display range (distance from an object) info to the driver station.

The Color-Range sensor uses reflected light to determine the distance from the sensor to the target object. It can be used to measure close distances (up 5" or more) with reasonable accuracy. Note that at the time this document was most recently edited, the REV Color-Range sensor saturates around 2" (5cm). This means that for distances less than or equal to 2", the sensor returns a measured distance equal to 2" or so.

Modify your op mode to add a telemetry statement that will send the distance information (in centimeters) to the Driver Station.

```
telemetry.addData("Servo Position", servoTest.getPosition());
telemetry.addData("Target Power", tgtPower);
telemetry.addData("Motor Power", motorTest.getPower());
telemetry.addData("Distance (cm)", sensorColorRange.getDistance(DistanceUnit.CM));
telemetry.addData("Status", "Running");
telemetry.update();
```

After you have modified your op mode, push the build button, then run the op mode to verify that it now displays distance on your Driver Station. Note that if the distance reads "NaN" (short for "Not a Number") it probably means that your sensor is too far from the target (zero reflection). Also note that the sensor saturates at around 5 cm.

12.2 Touch Sensor

The REV Robotics Touch Sensor can be connected to a digital port on the Expansion Hub. The Touch Sensor is HIGH (returns TRUE) when it is not pressed. It is *pulled* LOW (returns FALSE) when it is pressed.



Figure 7 - REV Robotics Touch Sensor.

The Expansion Hub digital ports contain two digital pins per port. When you use a 4-wire JST cable to connect a REV Robotics Touch sensor to an Expansion Hub digital port, the Touch Sensor is wired to the second of the two digital pins within the port. The first digital pin of the 4-wire cable remains disconnected.

For example, if you connect a Touch Sensor to the “0,1” digital port of the Expansion Hub, the Touch Sensor will be connected to the second pin (labeled “1”) of the port. The first pin (labeled “0”) will stay disconnected.

Modify the code in your op mode that occurs *before* the `waitForStart` command to set the digital channel for input mode.

```
// set digital channel to input mode.
digitalTouch.setMode(DigitalChannel.Mode.INPUT);

telemetry.addData("Status", "Initialized");
telemetry.update();
// Wait for the game to start (driver presses PLAY)
waitForStart();
```

Also, modify the code in your while loop to add an if-else statement that checks the state of the digital input channel. If the channel is LOW (false), the touch sensor button is pressed and being pulled LOW to ground. Otherwise, the touch sensor button is not pressed.

```
// is button pressed?
if (digitalTouch.getState() == false) {
    // button is pressed.
    telemetry.addData("Button", "PRESSED");
} else {
    // button is not pressed.
    telemetry.addData("Button", "NOT PRESSED");
}

telemetry.addData("Status", "Running");
telemetry.update();
```

Rebuild your op mode, then reinitialize and restart your op mode. The op mode should now display the state of the button (“PRESSED” or “NOT PRESSED”).

13 FIRST Tech Challenge Reference Information

13.1 Javadoc Reference Pages

As you start to write more complicated op modes, you will need to use more features of the *FIRST* Tech Challenge software development kit (SDK). You can reference online Javadoc material that provide descriptions of the available *FIRST* Tech Challenge-related classes and methods, at the following web address:

http://ftctechnh.github.io/ftc_app/doc/javadoc/index.html

13.2 Control System Wiki

There is an up-to-date online wiki for the FTC Android control system. This document serves as a convenient online reference for all topics related to the control system:

https://github.com/ftctechnh/ftc_app/wiki

13.3 Technology Forum

Registered teams can create user accounts on the *FIRST* Tech Challenge forum. Teams can use the forum to ask questions and receive support from the *FIRST* Tech Challenge community.

The technology forum can be found at the following address:

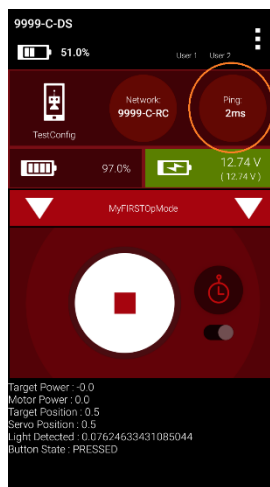
<https://ftcforum.usfirst.org/forum/ftc-technology?156-FTC-Technology>

14 Troubleshooting

14.1 Driver Station Appears Unresponsive

If you are ready to run an op mode, but the Driver Station is unresponsive and you cannot initialize or start your selected op mode, check the following items:

1. Verify that the Driver Station is properly paired to the Robot Controller.
2. Make sure that the Robot Controller is not in Programming Mode.
3. Check the *ping* times on the Driver Station main screen. The ping time is the average time it takes for the Driver Station to send a message to the Robot Controller and for the Robot Controller to acknowledge that it received the message. If the ping time is low (< 20 msec) the wireless connection between the Driver Station and Robot Controller is good. If the ping time is consistently high (> 50 msec) there could be some wireless interference in your venue that is causing the problems between the Driver Station and the Robot Controller.



14.2 Warning: problem communicating...

If you are trying to run an op mode and you notice error messages like the ones displayed below, it could be that your wired connection between the phone and the electronic modules is bad.

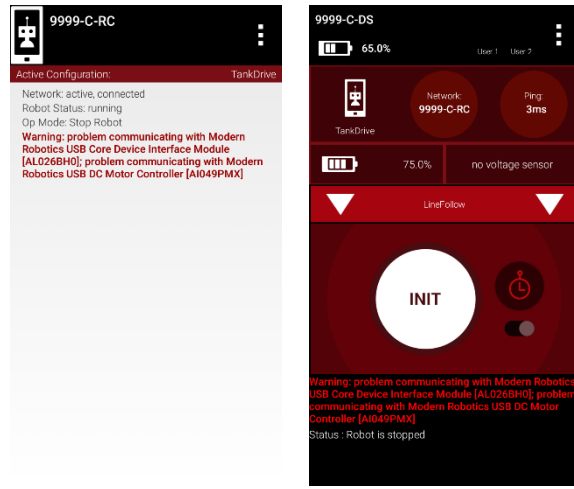


Figure 8 - A "problem communicating with..." message often indicates a bad connection between the phone and modules.

If you notice this error message, here are some things you can try:

1. Verify that the USB cable connecting the phone to the Expansion Hub is secure and well connected.
2. Verify that the 12V power cables connecting the battery to the switch and the Expansion Hub are properly secured and connected. Also, verify that the power switch is in the on position.
3. Try to do a "Restart Robot" from the pop up menu (touch the three vertical dots in the upper right hand screen of the Robot Controller or Driver Station apps).
4. If that does not work, disconnect the USB cable from the phone, then shut down the main power switch on the Expansion Hub. Wait for 5 seconds, then power the device back on and reconnect the USB cable to the phone.