

# FTC New Platform Workshop

presented

By



**FTC TEAM #8565**

# Setup GitHub for Team Development

Brandon Wang



# Step 1: Making and Sharing



# Creating GitHub accounts

All of the programmers will need one!



## Why you'll love GitHub.

Powerful features to make software development more collaborative.



Great collaboration starts with communication.



Friction-less development across teams.

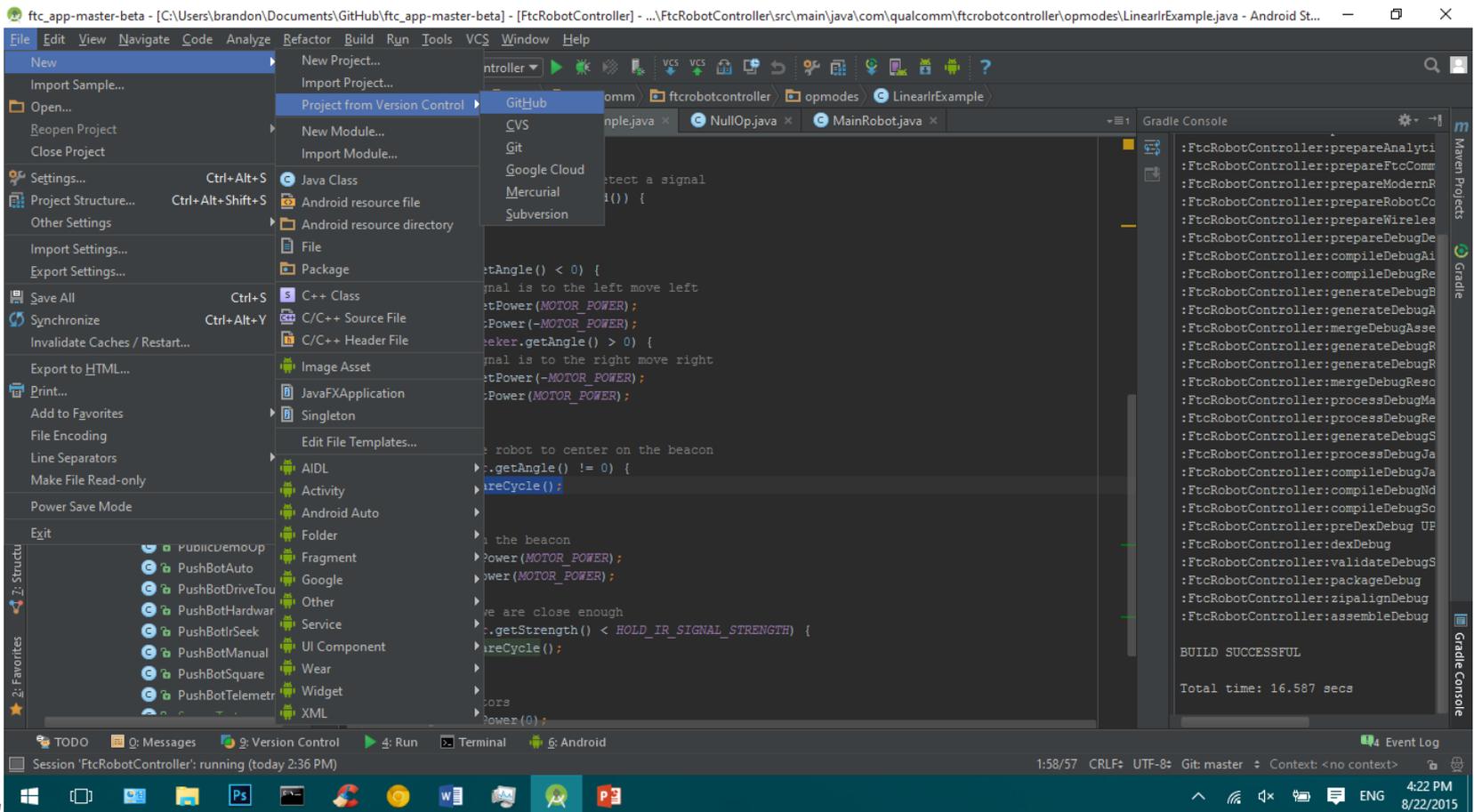


World's largest open source community.



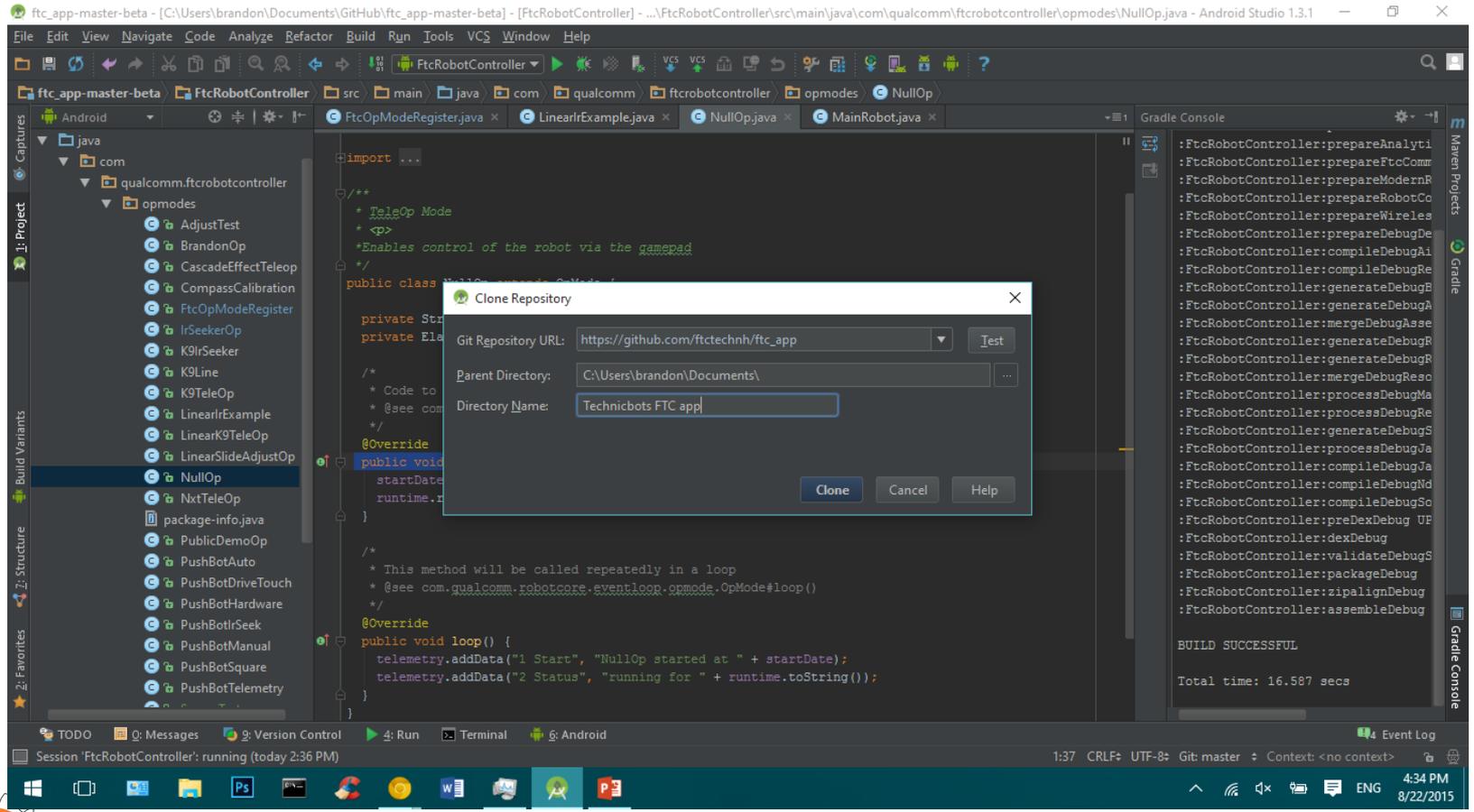
# Importing project from remote Github repository

Go to File > New > Project from Version Control > Github.



# FTC SDK Repository

The FTC repository is located at [https://github.com/ftctechnh/ftc\\_app](https://github.com/ftctechnh/ftc_app)

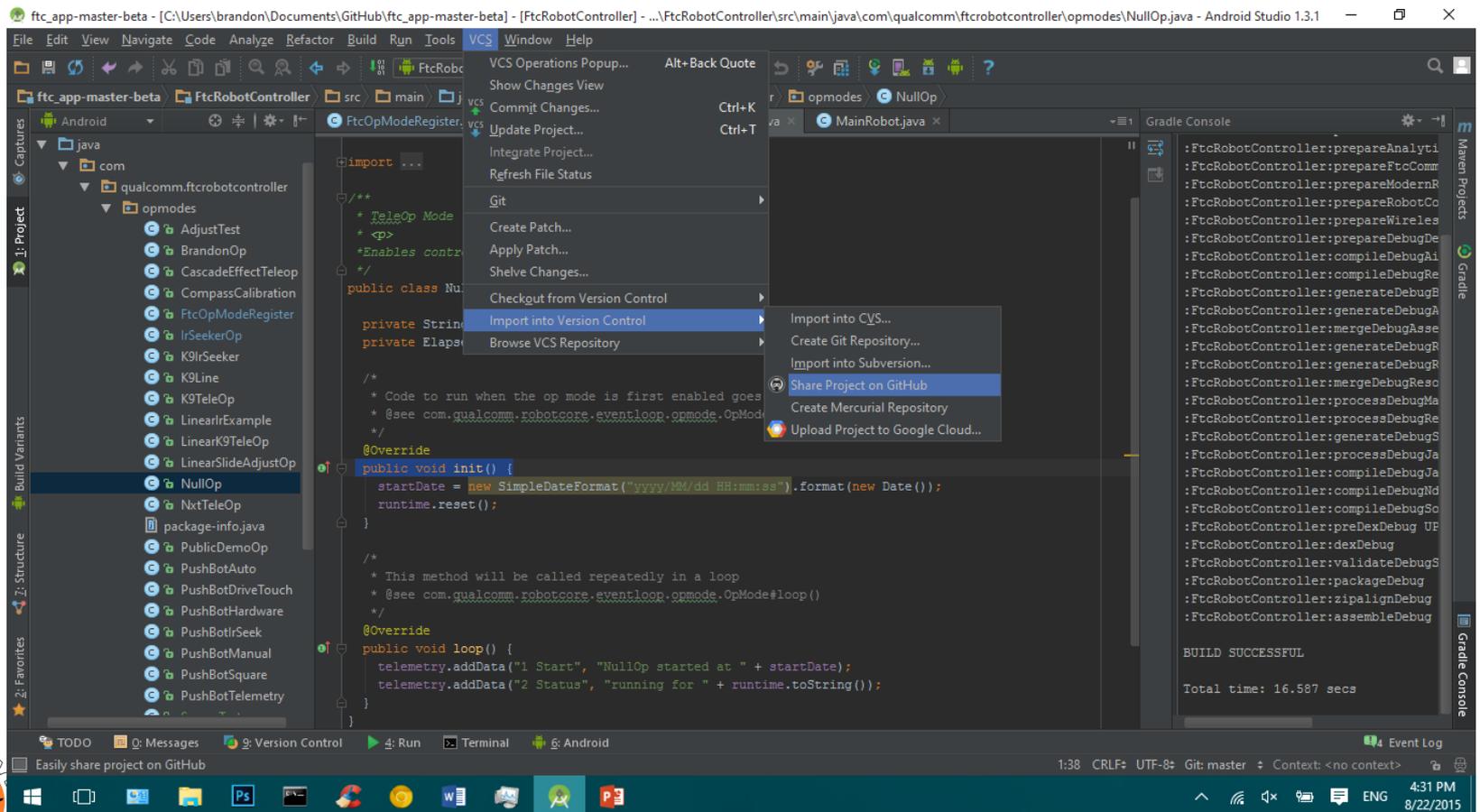


# Uploading your copy of the FTC project to GitHub



# Uploading your copy of the FTC project to GitHub

VCS > Import into Version Control > Share Project on Github.



The screenshot shows the Android Studio interface with the VCS menu open. The path is: VCS > Import into Version Control > Share Project on Github. The main editor shows the code for the NullOp class, and the Gradle Console at the bottom right displays the output of a successful build.

```
ftc_app-master-beta - [C:\Users\brandon\Documents\GitHub\ftc_app-master-beta] - [FtcRobotController] - ...\FtcRobotController\src\main\java\com\qualcomm\ftcrobotcontroller\opmodes\NullOp.java - Android Studio 1.3.1
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

VCS Operations Popup... Alt+Back Quote

- Show Changes View
- Commit Changes... Ctrl+K
- Update Project... Ctrl+T
- Integrate Project...
- Refresh File Status
- Git
  - Create Patch...
  - Apply Patch...
  - Shelve Changes...
  - Checkout from Version Control
  - Import into Version Control
    - Import into CVS...
    - Create Git Repository...
    - Import into Subversion...
    - Share Project on Github
    - Create Mercurial Repository
    - Upload Project to Google Cloud...
  - Browse VCS Repository

```
import ...  
/**  
 * TeleOp Mode  
 * <p>  
 * Enables control of the robot.  
 */  
public class NullOp {  
  
    private String startdate;  
    private ElapsedTime runtime = new ElapsedTime();  
  
    /**  
     * Code to run when the op mode is first enabled goes  
     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode  
     */  
    @Override  
    public void init() {  
        startDate = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(new Date());  
        runtime.reset();  
    }  
  
    /**  
     * This method will be called repeatedly in a loop  
     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#loop()  
     */  
    @Override  
    public void loop() {  
        telemetry.addData("1 Start", "NullOp started at " + startDate);  
        telemetry.addData("2 Status", "running for " + runtime.toString());  
    }  
}
```

Gradle Console

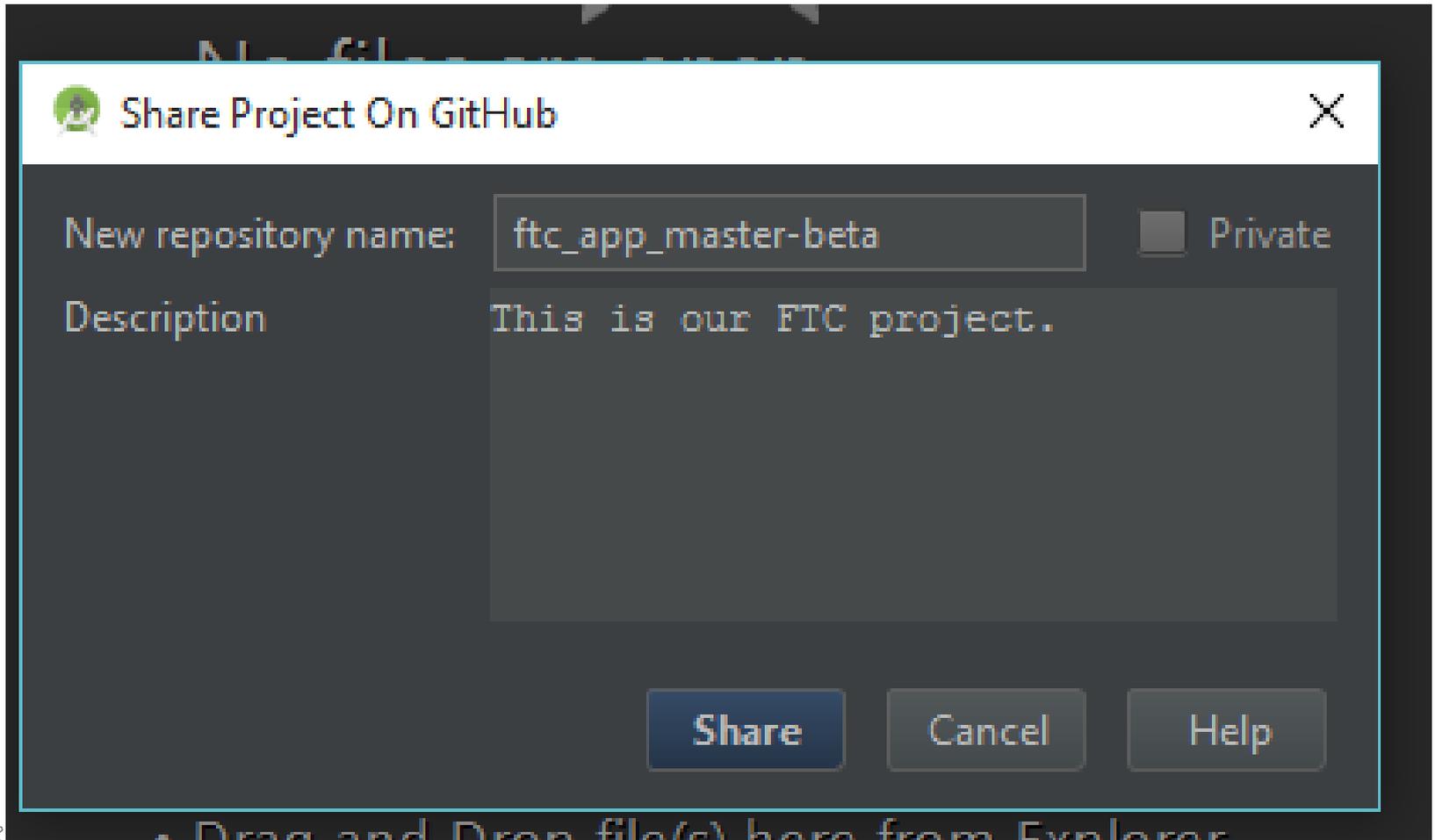
```
:FtcRobotController:prepareAnalyti  
:FtcRobotController:prepareFtcComm  
:FtcRobotController:prepareModernR  
:FtcRobotController:prepareWireles  
:FtcRobotController:prepareDebugDe  
:FtcRobotController:compileDebugAi  
:FtcRobotController:compileDebugRe  
:FtcRobotController:generateDebugB  
:FtcRobotController:generateDebugA  
:FtcRobotController:mergeDebugAsse  
:FtcRobotController:generateDebugR  
:FtcRobotController:generateDebugR  
:FtcRobotController:mergeDebugReso  
:FtcRobotController:processDebugMa  
:FtcRobotController:processDebugRe  
:FtcRobotController:generateDebugS  
:FtcRobotController:processDebugJa  
:FtcRobotController:compileDebugJa  
:FtcRobotController:compileDebugId  
:FtcRobotController:compileDebugSo  
:FtcRobotController:preDexDebug UF  
:FtcRobotController:dexDebug  
:FtcRobotController:validateDebugS  
:FtcRobotController:packageDebug  
:FtcRobotController:zipalignDebug  
:FtcRobotController:assembleDebug  
  
BUILD SUCCESSFUL  
  
Total time: 16.587 secs
```

1:38 CRLF UTF-8 Git: master Context: <no context>

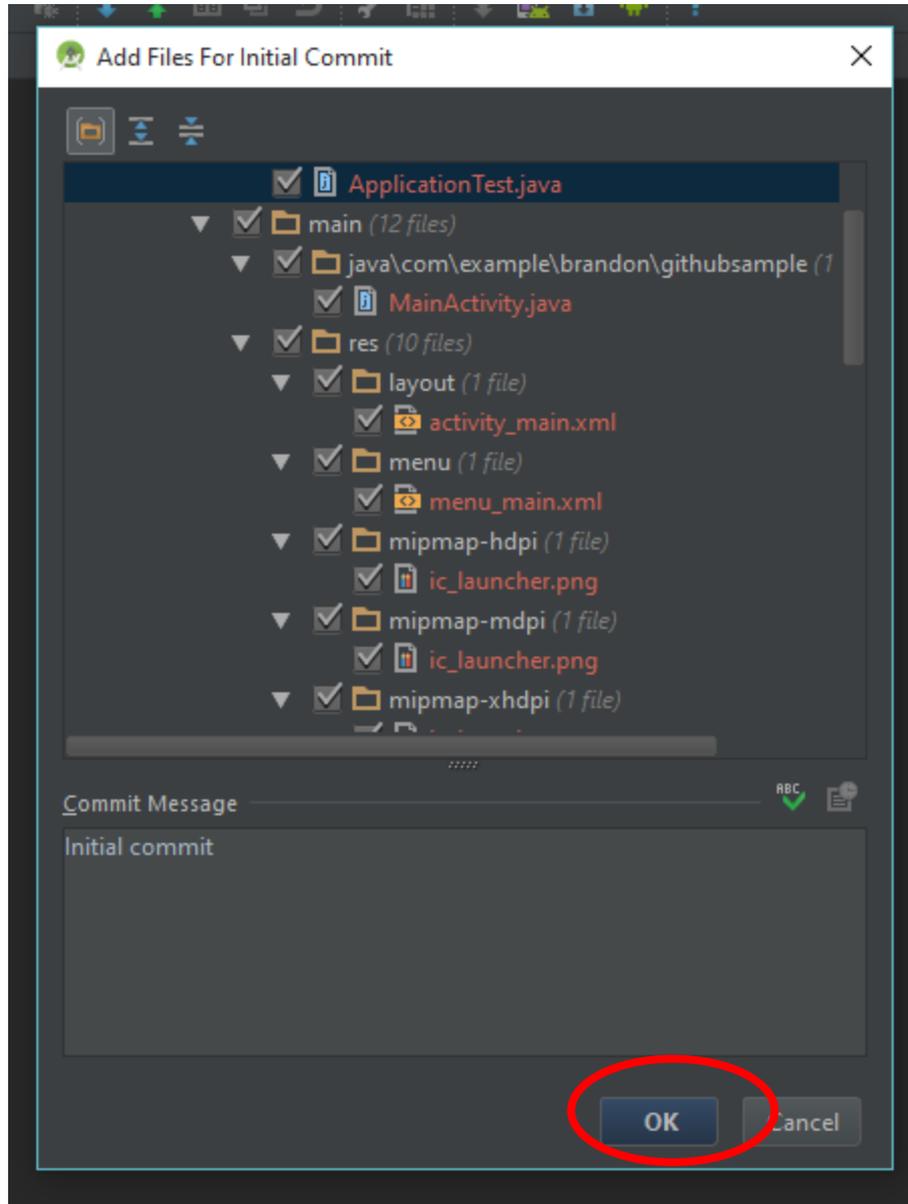
4:31 PM 8/22/2015



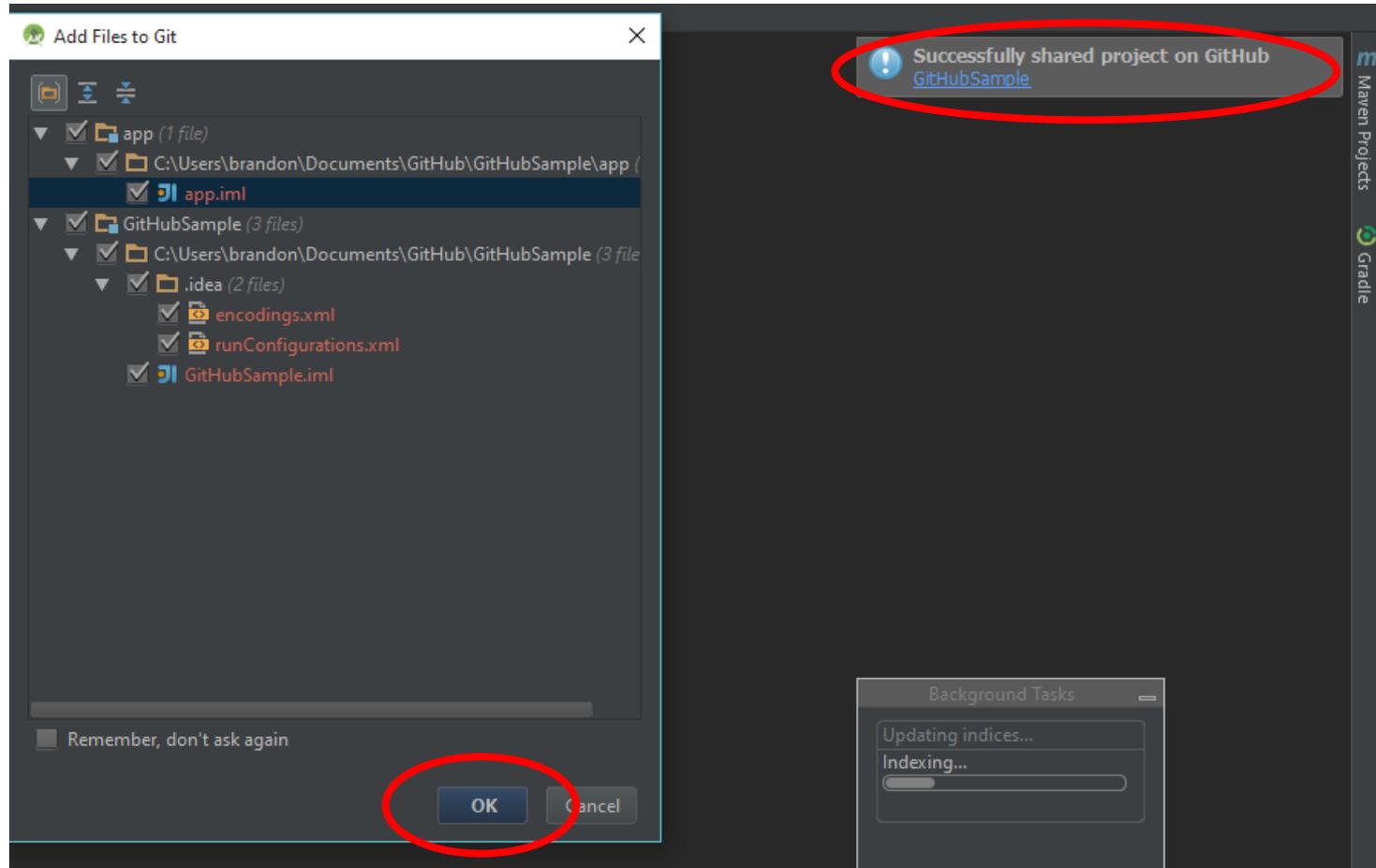
# Uploading your copy of the FTC project to GitHub



# Uploading your copy of the FTC project to GitHub



# Uploading your copy of the FTC project to GitHub



# Your copy of the FTC project on GitHub

This screenshot shows the GitHub interface for the repository 'samTechnicbots / ftc\_app-master-beta'. At the top, there is a search bar and navigation links for 'Pull requests', 'Issues', and 'Gist'. The repository name is displayed, along with statistics: 24 commits, 1 branch, 0 releases, and 2 contributors. A navigation bar shows the current branch as 'master' and a link to 'ftc\_app-master-beta / +'. Below this, a commit message is visible: 'Trying to get IR Seeker and other sensors to work.' by 'brandontechnicbots' 6 days ago. A file tree lists various files and folders, including 'FtcRobotController', 'doc', 'gradle/wrapper', '.gitignore', 'README.md', 'build.gradle', 'gradlew', and 'gradlew.bat'. On the right side, there are links for 'Code', 'Issues', 'Pull requests', 'Wiki', 'Pulse', and 'Graphs'. At the bottom right, there are options to clone the repository via HTTPS, SSH, or Subversion, and a 'Clone in Desktop' button.

This repository Search

Pull requests Issues Gist

🔔 + 🏠

📄 samTechnicbots / ftc\_app-master-beta

👁 Unwatch 2 ⭐ Star 0 🍴 Fork 0

📄 24 commits 🌿 1 branch 📦 0 releases 👤 2 contributors

🔗 Branch: master ftc\_app-master-beta / +

Trying to get IR Seeker and other sensors to work.

🏠 brandontechnicbots authored 6 days ago latest commit 699aa923d7

📁 FtcRobotController	Trying to get IR Seeker and other sensors to work.	6 days ago
📁 doc	Initial commit	13 days ago
📁 gradle/wrapper	Initial commit	13 days ago
📄 .gitignore	Initial commit	13 days ago
📄 README.md	Initial commit	13 days ago
📄 build.gradle	Initial commit	13 days ago
📄 gradlew	Initial commit	13 days ago
📄 gradlew.bat	Initial commit	13 days ago

🔗 Code

🔔 Issues 0

🔗 Pull requests 0

📖 Wiki

🔔 Pulse

📊 Graphs

HTTPS clone URL

https://github.com/ 📄

You can clone with HTTPS, SSH, or Subversion. 📄

📄 Clone in Desktop



# Adding Team Members (On the GitHub site)

brandontechnicbots / Temp

Unwatch 1 Star 0

Options

Collaborators

Webhooks & services

Deploy keys

Collaborators

Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

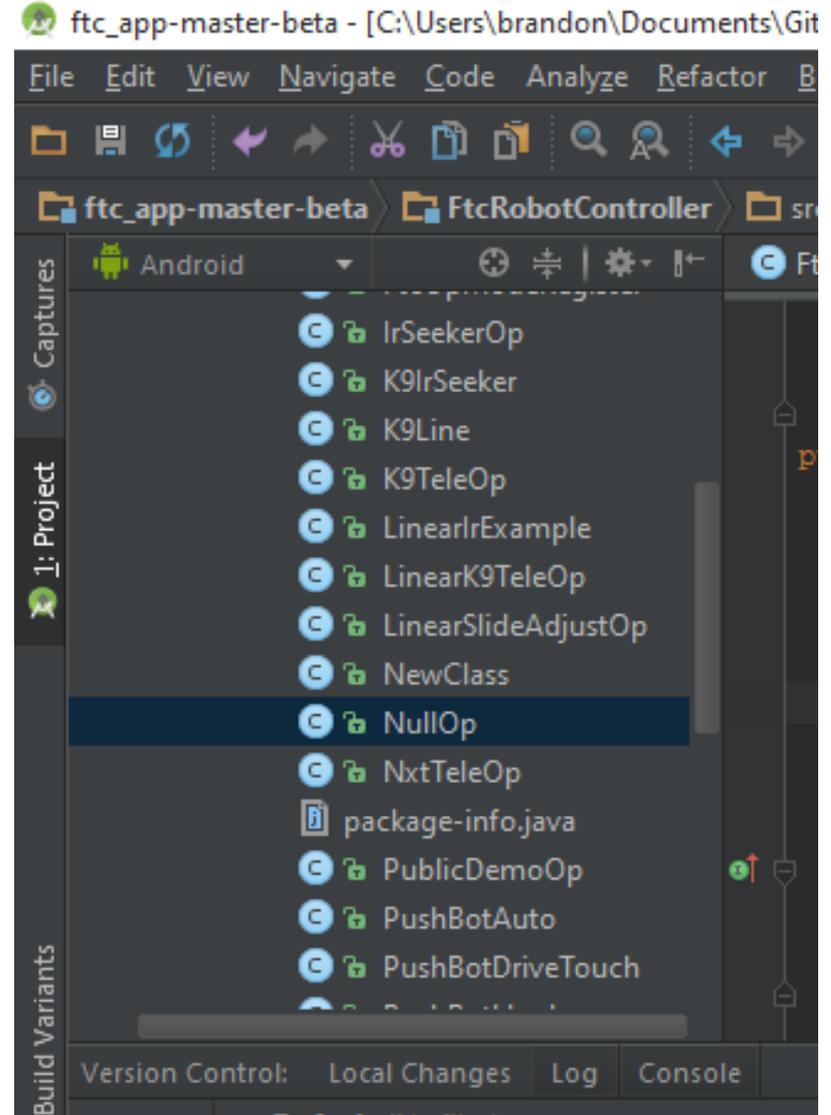
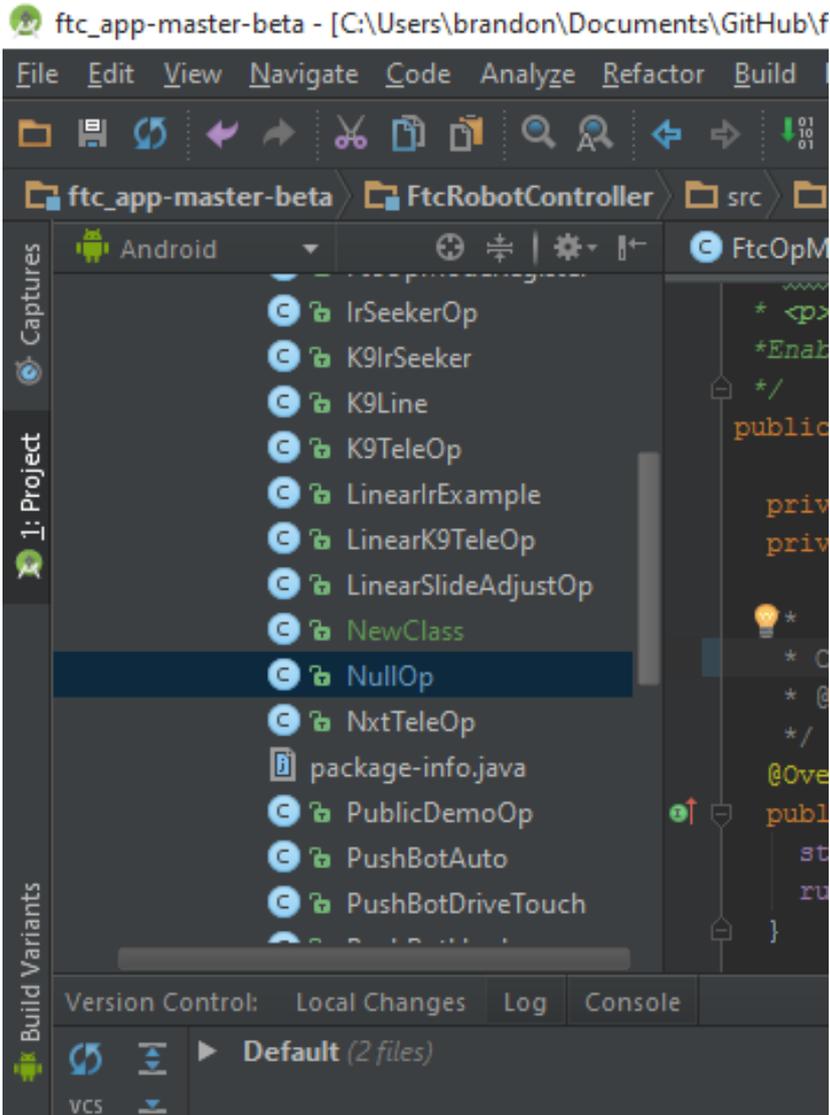
Add collaborator



# Step 2: The Workflow



# Color Coding



# Making Commits

- VCS > Commit Changes (Or Ctrl+K).

The screenshot shows an IDE window for a project named 'ftc\_app-master-beta'. The 'VCS' menu is open, and 'Commit Changes...' is selected, with the keyboard shortcut 'Ctrl+K' displayed. The 'Gradle Console' on the right shows a successful build with the message 'BUILD SUCCESSFUL' and a total time of 17.336 secs. The 'Event Log' at the bottom shows the build process starting at 7:42:44 PM and finishing at 7:44:55 PM. The IDE interface includes a project tree on the left, a code editor in the center, and a toolbar at the top.

```
ftc_app-master-beta - [C:\Users\brandon\Documents\GitHub\ftc_app-master-beta] - [FtcRobotController] - ...\FtcRobotController\src\main\java\com\qualcomm\ftcrobotcontroller\opmodes\LinearIrExample.java - Android St...
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

VCS Operations Popup... Alt+Back Quote  
Show Changes View  
Commit Changes... Ctrl+K  
Update Project... Ctrl+T  
Integrate Project...  
Refresh File Status  
Git  
Create Patch...  
Apply Patch...  
Shelve Changes...  
Checkout from Version Control  
Import into Version Control  
Browse VCS Repository

ftc\_app-master-beta FtcRobotController src main opmodes LinearIrExample  
CompassCalibration  
FtcOpModeRegister  
IrSeekerOp  
K9IrSeeker  
K9Line  
K9TeleOp  
LinearIrExample  
LinearK9TeleOp  
LinearSlideAdjustOp  
NullOp  
NxtTeleOp  
package-info.java  
PublicDemoOp  
PushBotAuto  
PushRntDriveTouch

```
// motorRight  
// wait for  
waitForStart  
  
// wait for  
while (!irS  
sleep(100  
}  
  
if (irSeeker  
// if the  
motorRight.setPower(MOTOR_POWER);  
motorLeft.setPower(-MOTOR_POWER);  
} else if (irSeeker.getAngle() > 0) {  
// if the signal is to the right move right  
motorRight.setPower(-MOTOR_POWER);  
motorLeft.setPower(MOTOR_POWER);
```

Gradle Console

```
:FtcRobotController:prepareDebugAndroidTestDependencies UP-TO-DATE  
:FtcRobotController:compileDebugAndroidTestAidl UP-TO-DATE  
:FtcRobotController:processDebugAndroidTestManifest UP-TO-DATE  
:FtcRobotController:compileDebugAndroidTestRenderscript UP-TO-DATE  
:FtcRobotController:generateDebugAndroidTestBuildConfig UP-TO-DATE  
:FtcRobotController:generateDebugAndroidTestAssets UP-TO-DATE  
:FtcRobotController:mergeDebugAndroidTestAssets UP-TO-DATE  
:FtcRobotController:generateDebugAndroidTestResValues UP-TO-DATE  
:FtcRobotController:generateDebugAndroidTestResources UP-TO-DATE  
:FtcRobotController:mergeDebugAndroidTestResources UP-TO-DATE  
:FtcRobotController:processDebugAndroidTestResources UP-TO-DATE  
:FtcRobotController:generateDebugAndroidTestSources UP-TO-DATE
```

BUILD SUCCESSFUL  
Total time: 17.336 secs

Event Log

```
7:42:44 PM Gradle sync started  
7:44:37 PM Gradle sync completed  
7:44:45 PM Executing tasks: [:FtcRobotController:generateDebugSources, :FtcRobotController:  
7:44:55 PM Gradle build finished in 17s 888ms
```

Version Control: Local Changes Log  
Default (No files)

7:10:10 PM Event Log

70:16 CRLF+ UTF-8+ Git: SensorWork Context: <no context>

9:20 PM  
8/23/2015

# Making Commits

- Remember to write a detailed description to help identify changes later.

The screenshot displays the 'Commit Changes' dialog in an IDE. The dialog is titled 'Commit Changes' and shows a list of files to be committed. The files listed are:

- FtcRobotController (4 files)
- C:\Users\brandon\Documents\GitHub\ftc\_app-master-beta
- FtcOpModeRegister.java
- IrSeekerOp.java
- MonkeyOp.java
- SensorTest.java

The dialog also shows the commit message: 'Trying to get IR Seeker and other sensors to work.' Below the message, there are options for 'Before Commit' actions:

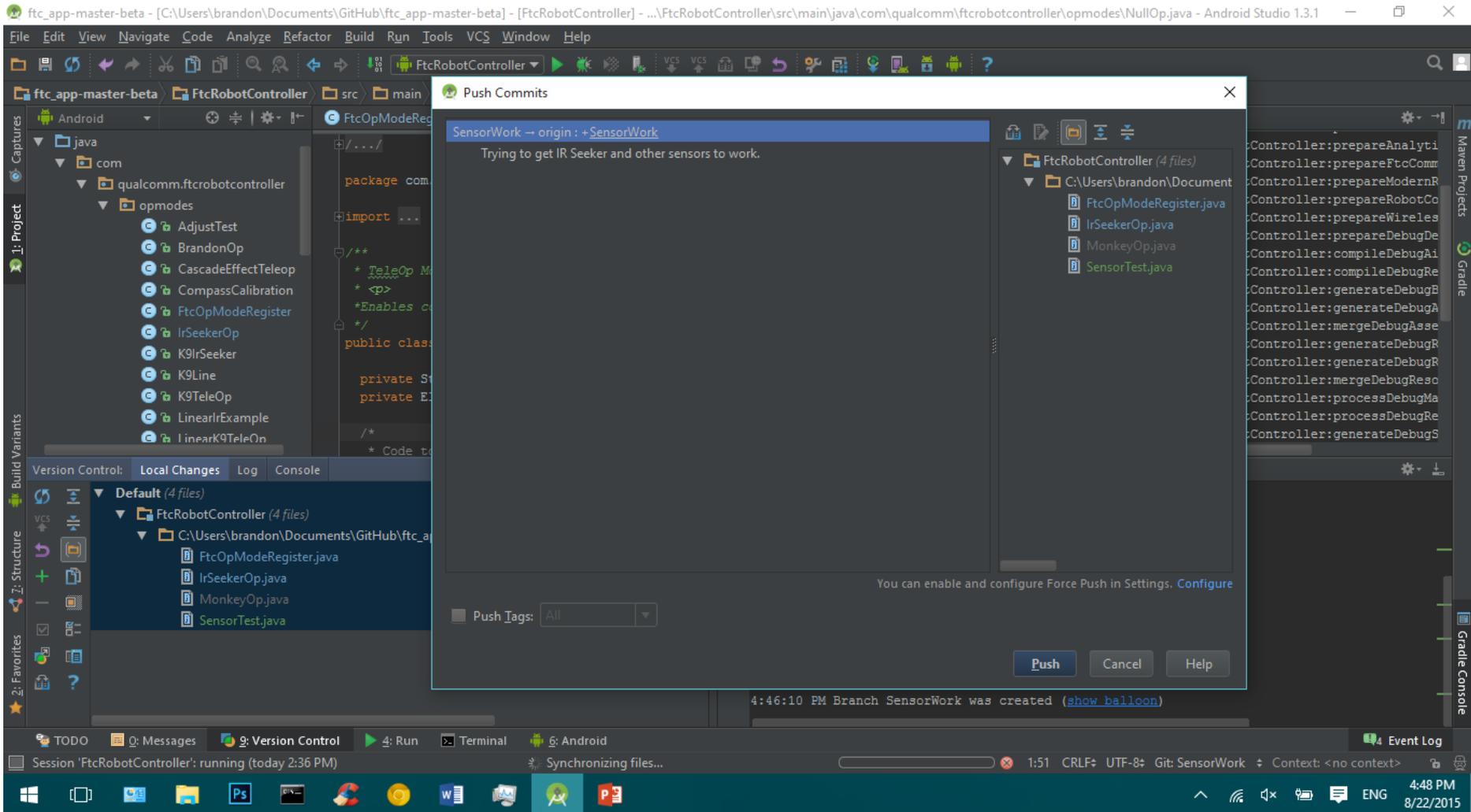
- Amend commit
- Reformat code
- Rearrange code
- Optimize imports
- Perform code analysis
- Check TODO (Show All) [Configure](#)
- Cleanup
- Update copyright

The dialog has buttons for 'Commit', 'Cancel', and 'Help'. The background shows the IDE interface with a project tree on the left and a console window on the right. The console window shows the output of the build process, including the command 'gradle assembleDebug' and the output 'BUILD SUCCESSFUL'.

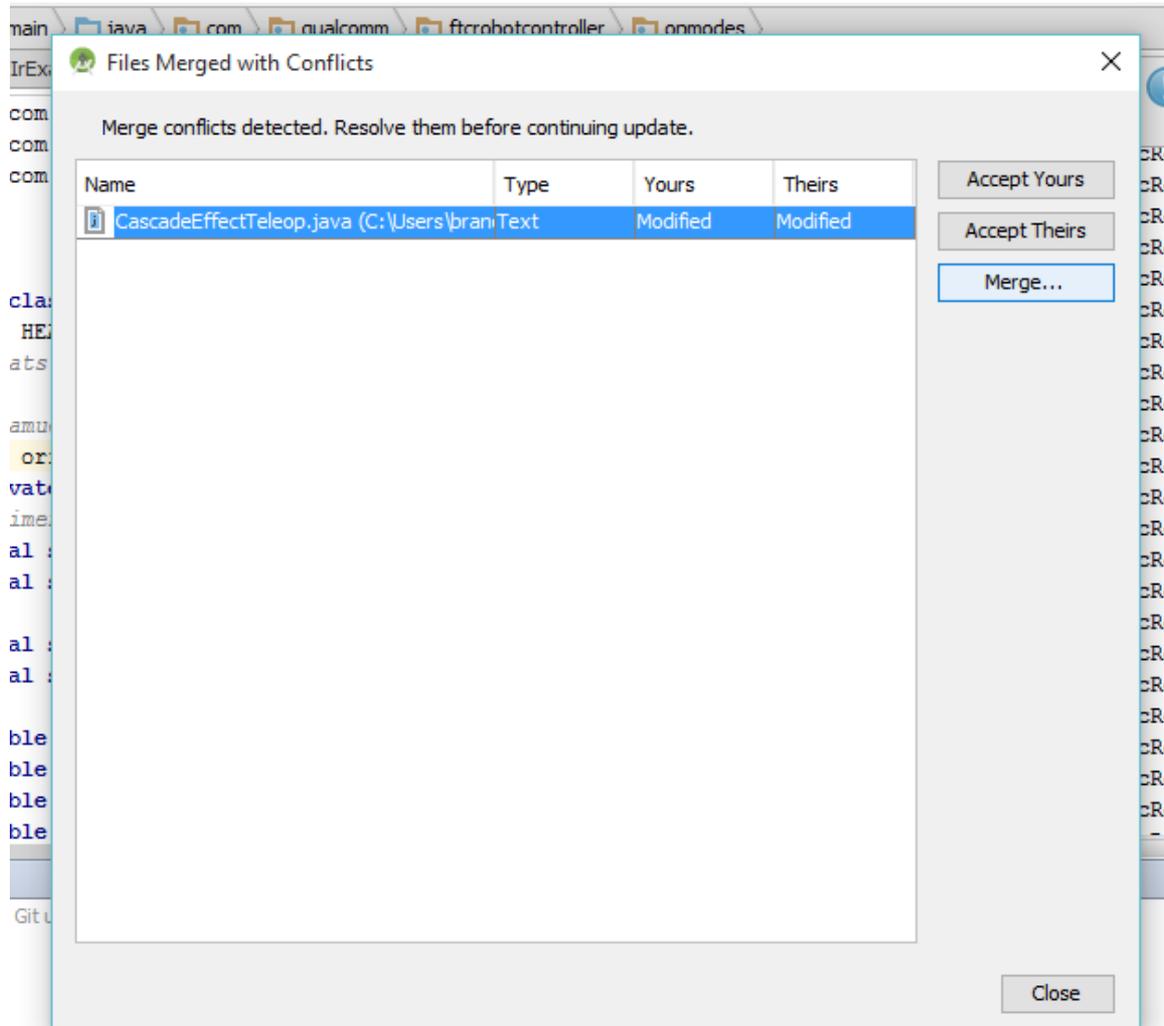


# Pushing

- Here you can choose what commits you want to push.



# Pushing (Resolving Conflicts)



# Pushing (Resolving Conflicts)

The screenshot shows a 'Merge Revisions' dialog box in an IDE. The dialog has three main columns: 'Local Changes', 'Merge Result', and 'Changes from Server (revision b92...)'. The code is displayed with line numbers 1 through 19. Various lines are highlighted with colors: grey for deleted, blue for changed, green for inserted, and red for conflict. Conflict markers like 'X' and '«' are present on several lines. At the bottom, there is a summary '23 changes, 2 conflicts' and a legend for Deleted, Changed, Inserted, and Conflict. 'Apply' and 'Revert' buttons are visible at the bottom right.

Local Changes	Merge Result	Changes from Server (revision b92...)
1 package com.qu	1 package com.qualc	1 package com.qu
2	2	2
3 import com.qua	3 import com.qualco	3 import com.qua
4 import com.qua	4 import com.qualco	4 import com.qua
5 import com.qua	5 import com.qualco	5 import com.qua
6 import com.qua	6 import com.qualco	6 import com.qua
7 import com.qua	7 import com.qualco	7 « X import com.qua
8 import com.qua	8 import com.qualco	8 « X import com.qua
9 import com.tec	9 import com.techn	9 import com.tec
10	10	10
11	11	11
12 public class C	12 public class Cas	12 « X
13 //cats are X »	13 private Elaps	13 public class C
14 private El	14 //experimental	14 « X //Samuel's
15 //experimental X »	15 final static	15 private El
16 final stat	16 final static	16 //experimental
17 final stat	17	17 final stat
18	18 final static	18 « X final stat
19 final stat	19	19

23 changes, 2 conflicts

Deleted Changed Inserted Conflict

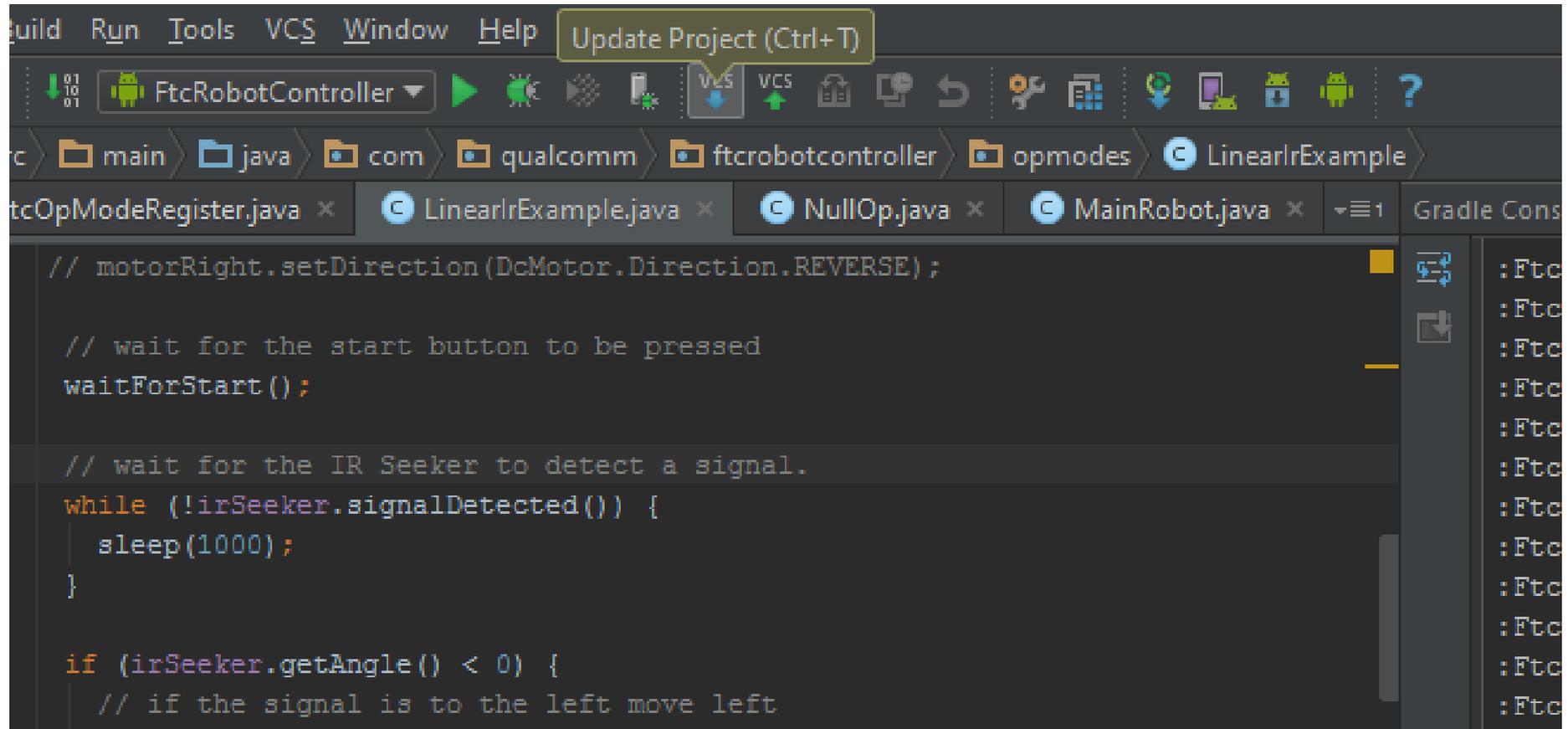
Apply Revert

Close



# Pulling

[GitHub\ftc\_app-master-beta] - [FtcRobotController] - ...\FtcRobotController\src\main\java\com\qualcomm\ftcrobotcontrolle



```
Build Run Tools VCS Window Help Update Project (Ctrl+T)
01 01 FtcRobotController
VCS VCS
c main java com qualcomm ftcrobotcontroller opmodes LinearlExample
FtcOpModeRegister.java LinearlExample.java NullOp.java MainRobot.java Gradle Cons
// motorRight.setDirection(DcMotor.Direction.REVERSE);

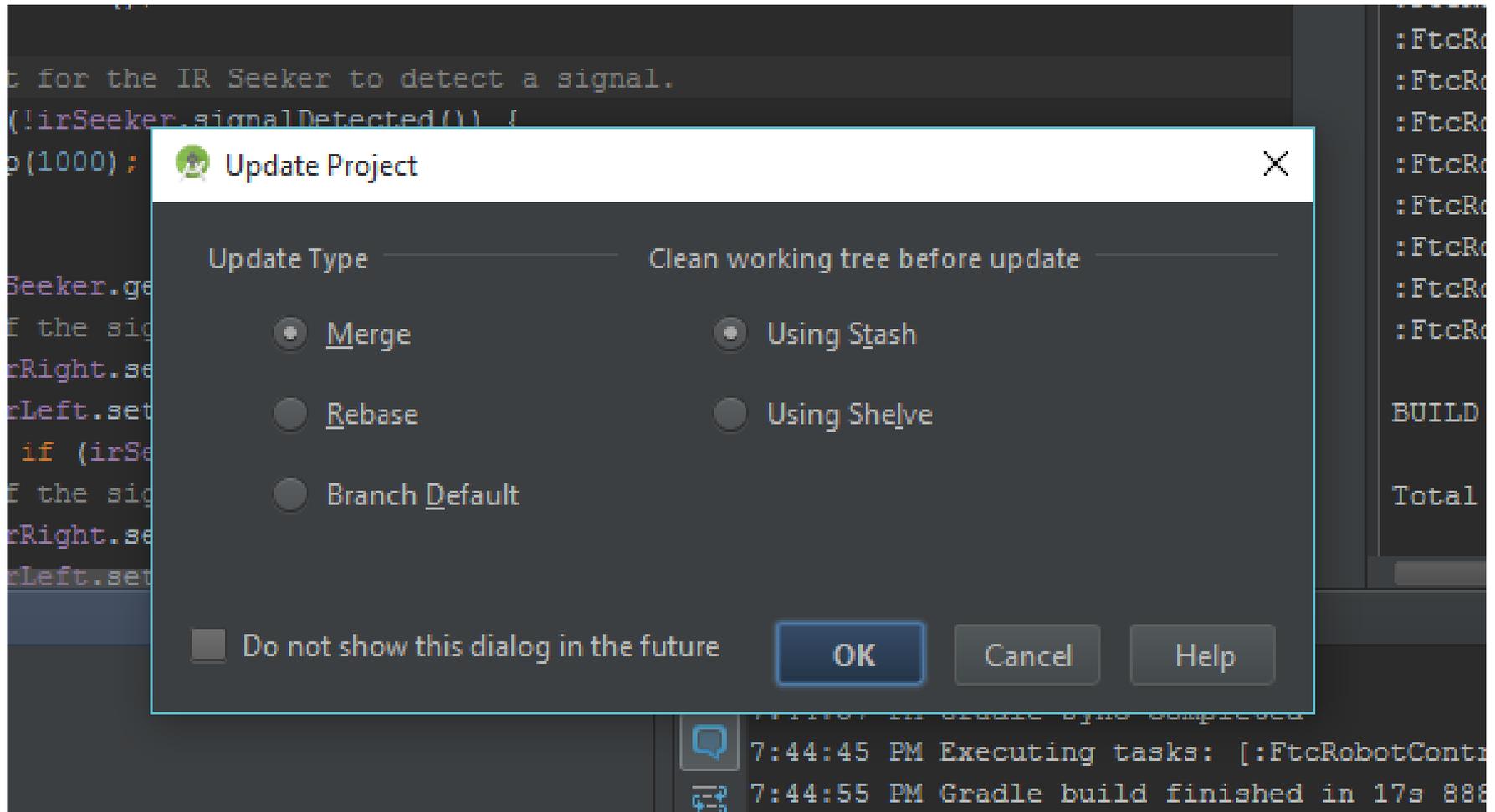
// wait for the start button to be pressed
waitForStart();

// wait for the IR Seeker to detect a signal.
while (!irSeeker.signalDetected()) {
    sleep(1000);
}

if (irSeeker.getAngle() < 0) {
    // if the signal is to the left move left
```



# Pulling



# What Not To Do

Inbox - brandonbigbroth x Recent - Google Drive x Science Fair Project - Wun x Editing ftc\_app-master-be x Brandon

GitHub, Inc. [US] https://github.com/samTechnibots/ftc\_app-master-beta/edit/master/FtcRobotController/src/main/java/com/qualcomm/ftcr

Programming 3D Printing Wunderlist Windchill Clever Math 25 Sentences School CYLC TSHIRT FTC

ftc\_app-master-beta / FtcRobotController / src / main / java / com / qualcomm / ftcrobotcontroller / opmodes /

FtcOpModeRegister.java or cancel

<> Edit file Preview changes Spaces 4 No wrap

```
1 /* Copyright (c) 2014, 2015 Qualcomm Technologies Inc
2
3 All rights reserved.
4
5 Redistribution and use in source and binary forms, with or without modification,
6 are permitted (subject to the limitations in the disclaimer below) provided that
7 the following conditions are met:
8
9 Redistributions of source code must retain the above copyright notice, this list
10 of conditions and the following disclaimer.
11
12 Redistributions in binary form must reproduce the above copyright notice, this
13 list of conditions and the following disclaimer in the documentation and/or
14 other materials provided with the distribution.
15
16 Neither the name of Qualcomm Technologies Inc nor the names of its contributors
17 may be used to endorse or promote products derived from this software without
18 specific prior written permission.
19
20 NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE GRANTED BY THIS
21 LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
23 THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
24 ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
25 FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
26 DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
27 SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
```

Windows taskbar: 9:35 PM 8/23/2015



# Wireless ADB Setup and Demo

Justin Jiang



# Overview

- What is ADB?
- Setup Wireless ADB between Android Studio and Robot Controller Phone
- Workflow with wireless ADB
- Live Demo



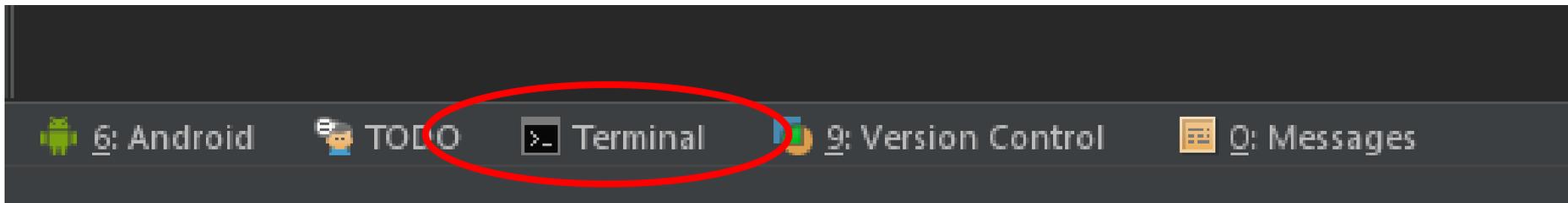
# Android Debug Bridge (ADB)

- The bridge between the Android Studio and Android Devices for debugging purpose
- Allow issuing commands within Android Studio to the connected Android Devices and collecting information from Android devices to Android Studio
- USB ADB
- Wireless ADB
  - Using home wireless router ←
  - Using laptop as hotspot



# Setup Wireless ADB

1. Go to the 'Terminal' tab at the bottom of Android Studio



# Setup Wireless ADB in Android Studio

2. Issue command: “adb kill-server”
3. Issue command: “adb usb” (Should see message ‘restarting in USB mode’ and phone appearing in the list)
4. Issue command: “adb tcpip 5555”
5. Go to your phone to find out your phone’s ip address on the home network, Settings > Wifi > “Connected Wifi” > IP Address
6. Issue command: “adb connect <IP Address>”
7. Unplug your usb cable
8. Issue command: “adb devices” to confirm your Android device via wireless connection is listed
9. Your device should also been seen in ‘Android’ tab at the bottom of Android Studio



# Workflow with Wireless ADB

1. At beginning of each debug session, set up the wireless ADB between Android Studio and the Robot Controller Phone based on the instructions in the previous slide
2. Pair Robot Controller Phone with Driver Station phone in the normal procedure
3. Write/Update your program and run the app from Android Studio, select the Robot Controller Phone on the wireless ADB created earlier
4. The pairing will be interrupted, the FtcRobotControllerApp will be updated and restarted, upon restarting, the pairing with Driver station phone is automatically resumed
5. Test your program with your Driver Station phone (repeat Steps 3-5 as many times as you need in the debugging session)



# Live Demo

