2017 FTC Kick-Off

FTC programming Fundamentals

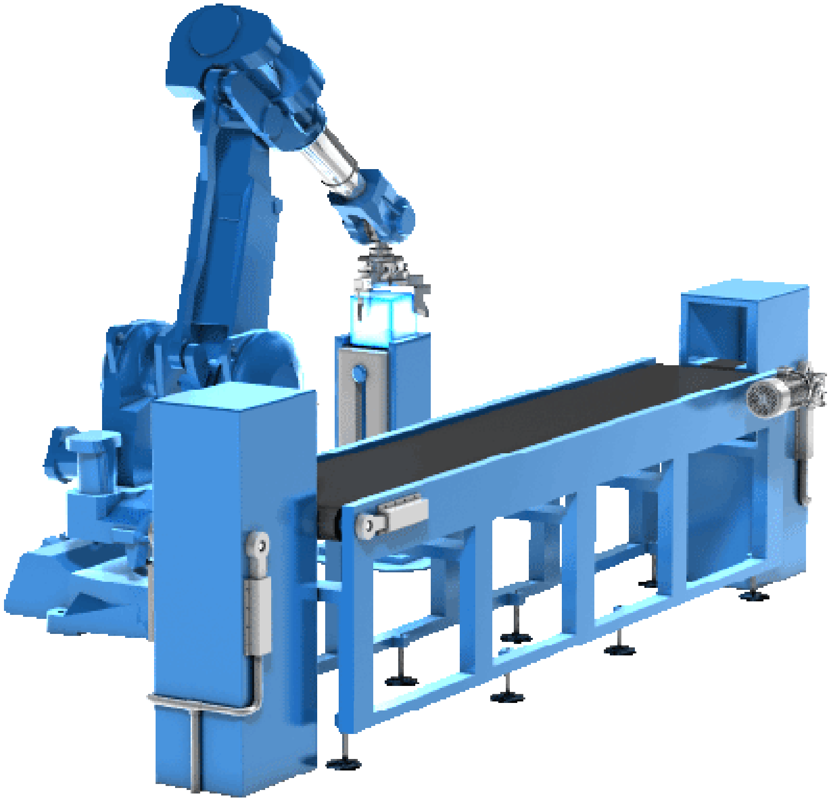
Frog Tech University, FRC Team 503
September 1, 2017

Course 104

Today's Goal…..

The goal of todays session is give you a very basic look at how to develop programming code for the robot

Programming Model
Compared to LEGO NXT

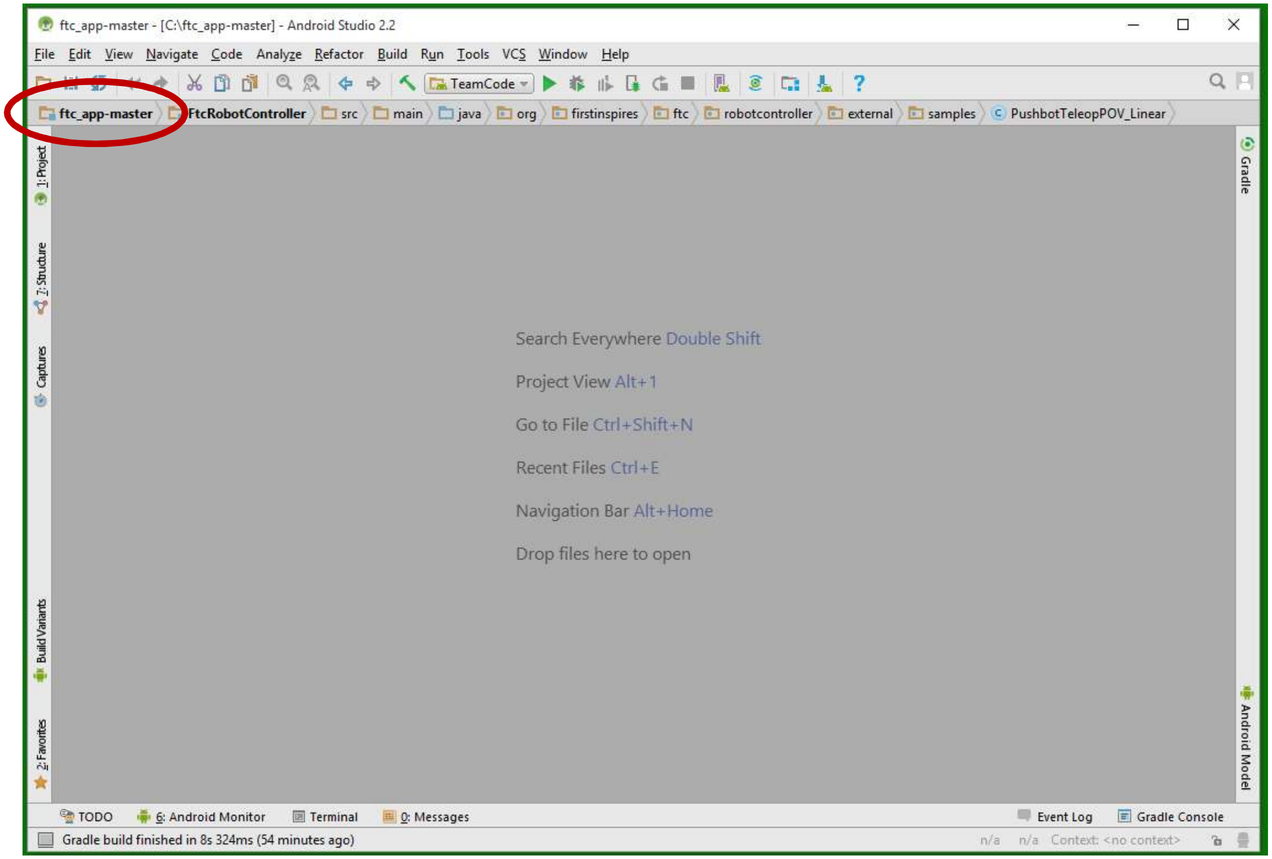


Guide

- The programming model for the new FTC control systems is different that the one used for the Lego NXT with tools like RobotC.
- Lego NXT used a linear programming model-which means things executed one after another in the code.
- The new FTC Java-based tools use an Event driven model.
- Sections of code fire off when a certain event occurs.
- For example the loop() method you are about to see fires off repeatedly until stopped.

How to Make changes

Select the ftc_app-master button



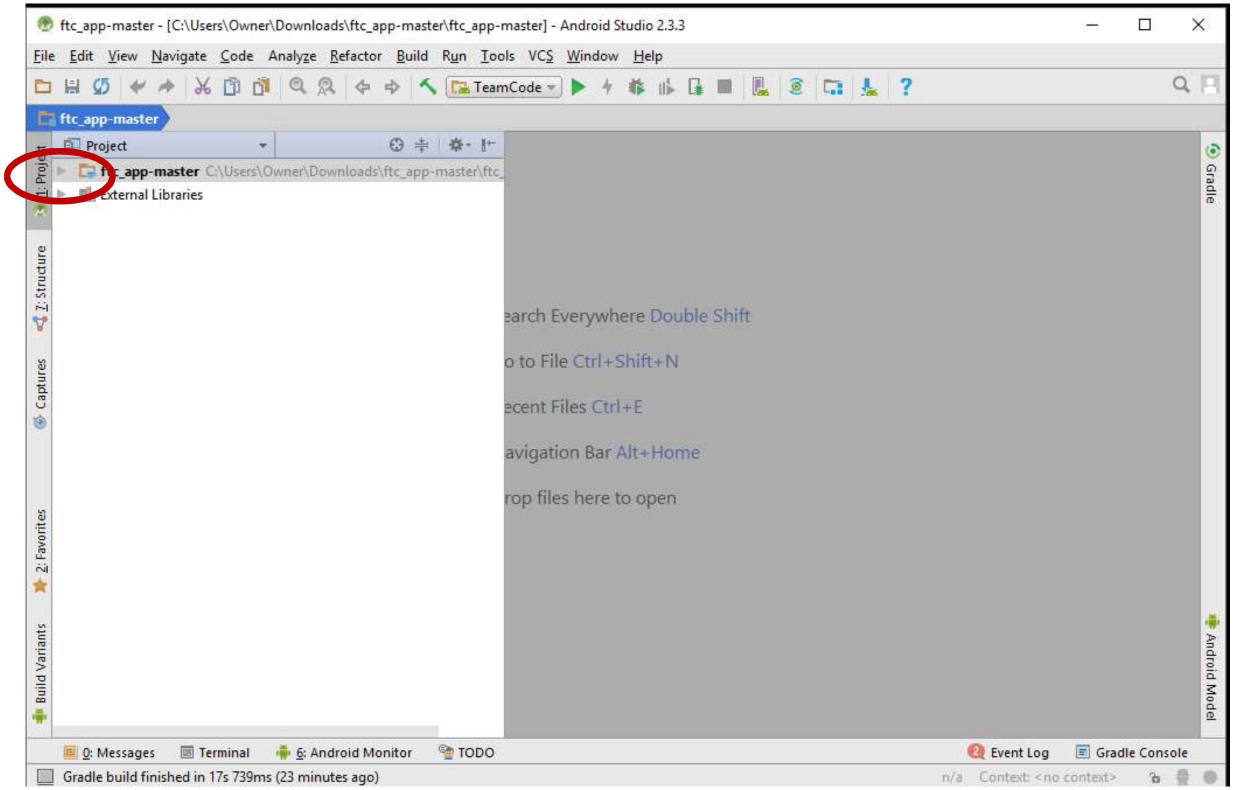
Gradle build finished in 8s 324ms (54 minutes ago)

Guide

- Open Android Studio
- The ftc_app-master application that you used to configure your machine will automatically open
- Click on the ‘ftc_app-master’ button on the menu bar (see red oval)

How to Make changes

Select the ftc_app-master button



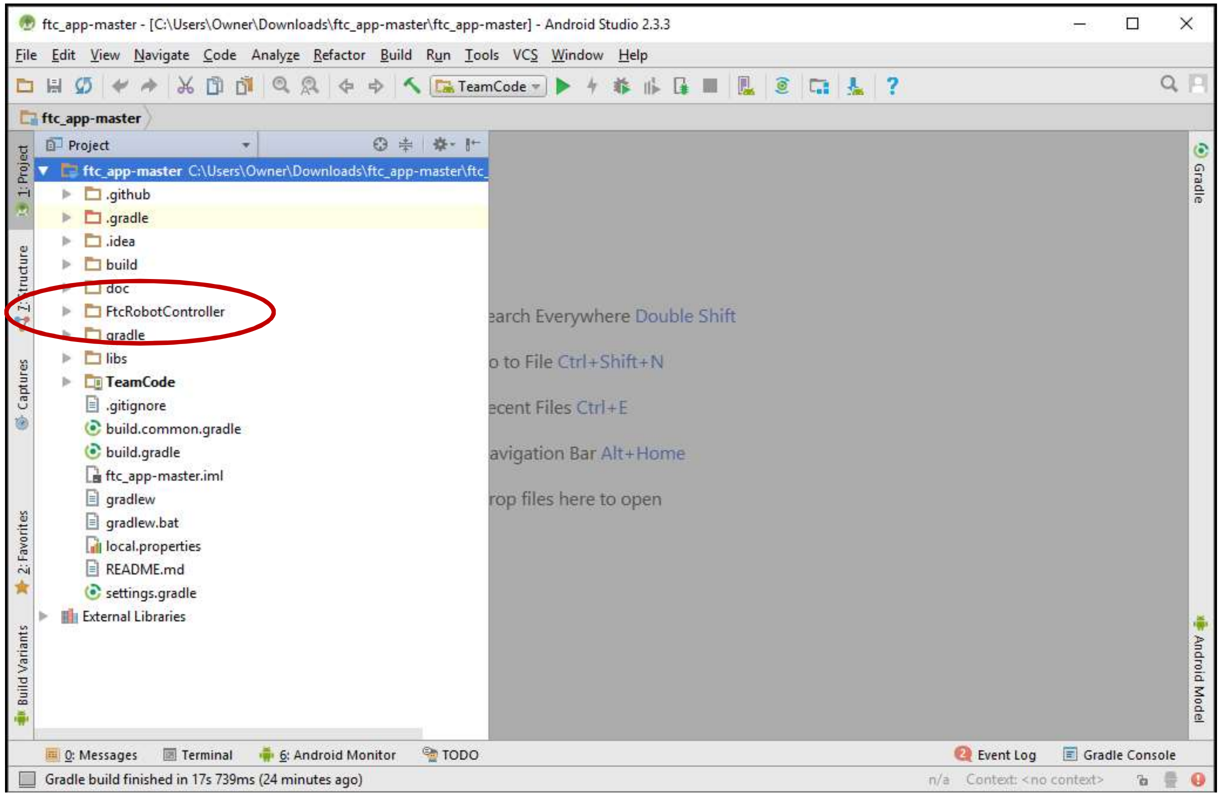
Gradle build finished in 17s 739ms (23 minutes ago)

Guide

- Click the ftc_app-master spin down button. (See red oval)

How to Make changes

Select the FtcRobotController Folder



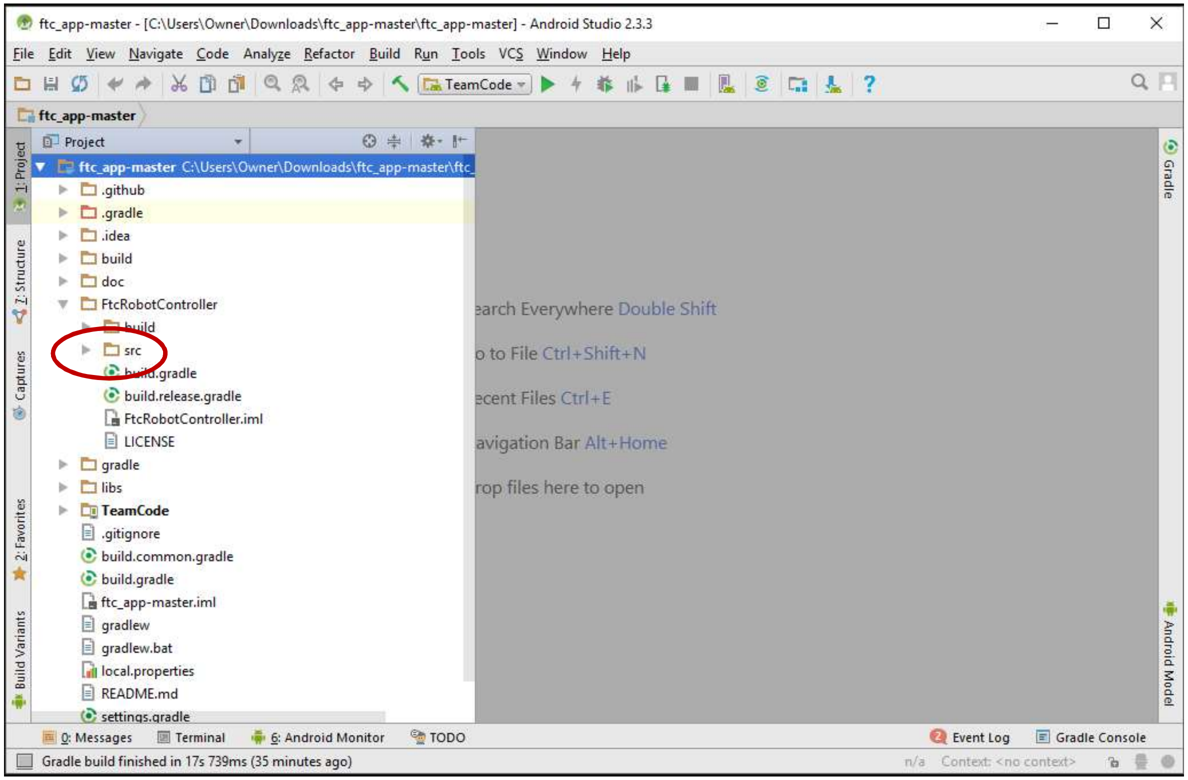
Gradle build finished in 17s 739ms (24 minutes ago)

Guide

- You will see the folders listed under the ftc_app-master folder. (The screen will look similar to the one on the left)
- There are two folders that you will generally work with:
  - FtcRobotController
    - This is where all the example code from FIRST is located
  - TeamCode
    - This is where you should put all of your programming changes
- Click the “FtcRobotController spin down button. (see red oval)

How to Make changes

Click the src button



ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - Android Studio 2.3.3

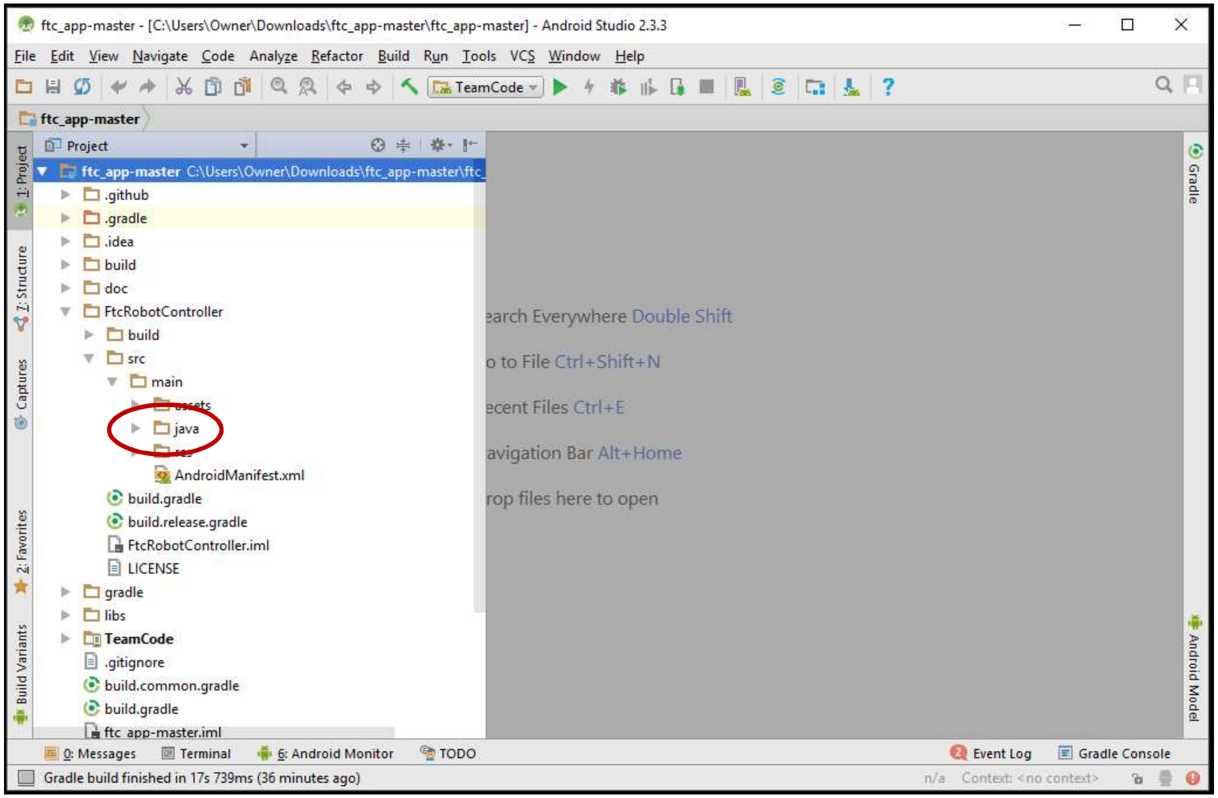
Gradle build finished in 17s 739ms (35 minutes ago)

Guide

- Click on the spin down arrow of the src folder. (see red oval)

How to Make changes

Click the java button

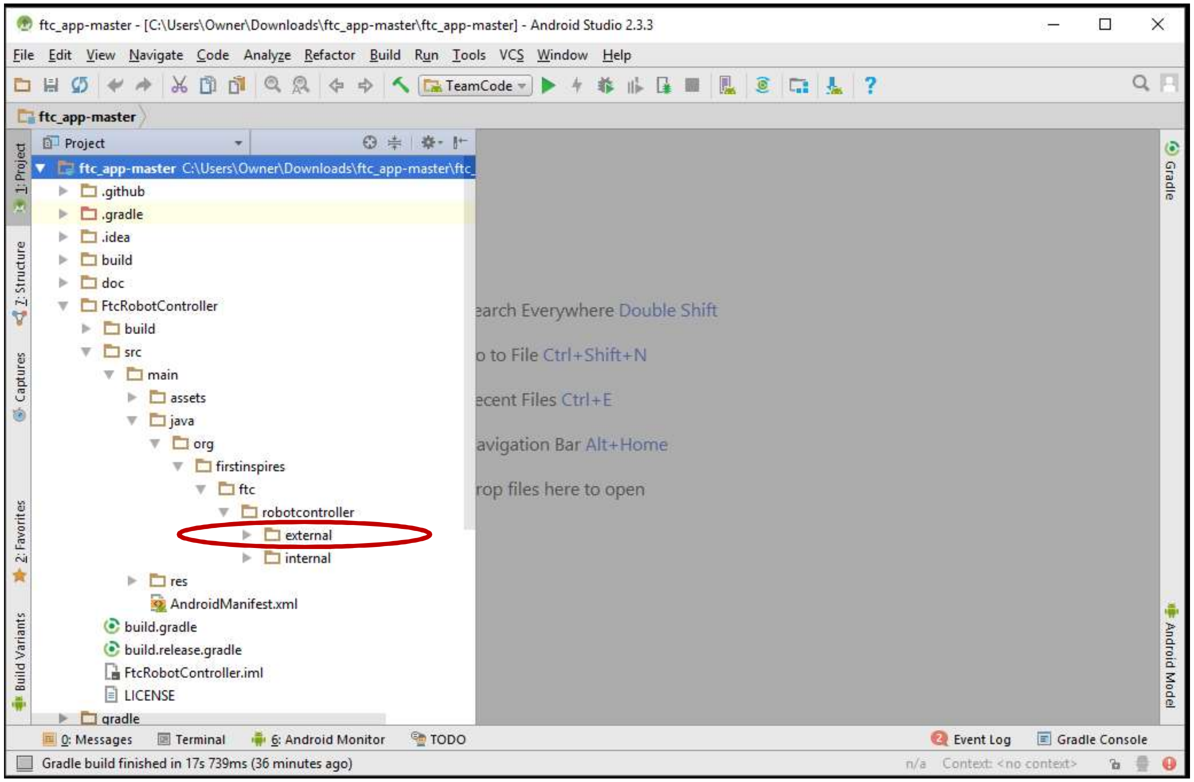


Gradle build finished in 17s 739ms (36 minutes ago)

Guide

- Click on the spin down arrow of the java folder. (see red oval)

How to Make changes
Click the external button

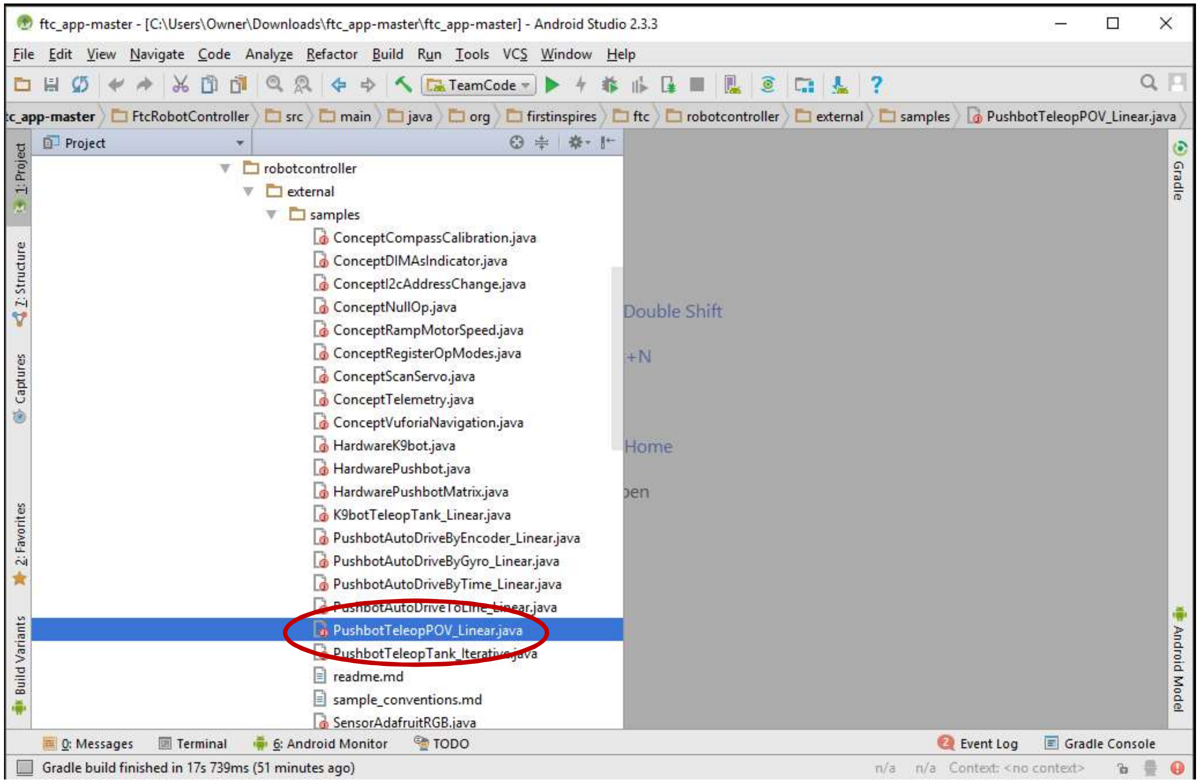


Gradle build finished in 17s 739ms (36 minutes ago)

Guide

- Click on the spin down arrow of the external folder. (see red oval)

How to Make changes
Find “PushbotTeleopPOV_Linear.java” file


ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - Android Studio 2.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
c_app-master 〉 FtcRobotController 〉 src 〉 main 〉 java 〉 org 〉 firstinspires 〉 ftc 〉 robotcontroller 〉 external 〉 samples 〉 PushbotTeleopPOV_Linear.java
Project
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
Double Shift
+N
Home
pen
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console
n/a n/a Context: <no context>

robotcontroller
- external
  - samples
    - ConceptCompassCalibration.java
    - ConceptDIMAsIndicator.java
    - ConceptI2cAddressChange.java
    - ConceptNullOp.java
    - ConceptRampMotorSpeed.java
    - ConceptRegisterOpModes.java
    - ConceptScanServo.java
    - ConceptTelemetry.java
    - ConceptVuforiaNavigation.java
    - HardwareK9bot.java
    - HardwarePushbot.java
    - HardwarePushbotMatrix.java
    - K9botTeleopTank_Linear.java
    - PushbotAutoDriveByEncoder_Linear.java
    - PushbotAutoDriveByGyro_Linear.java
    - PushbotAutoDriveByTime_Linear.java
    - PushbotAutoDriveToLine_Linear.java
    - PushbotTeleopPOV_Linear.java
    - PushbotTeleopTank_Iterative.java
    - readme.md
    - sample_conventions.md
    - SensorAdafruitRGB.java

Gradle build finished in 17s 739ms (51 minutes ago)

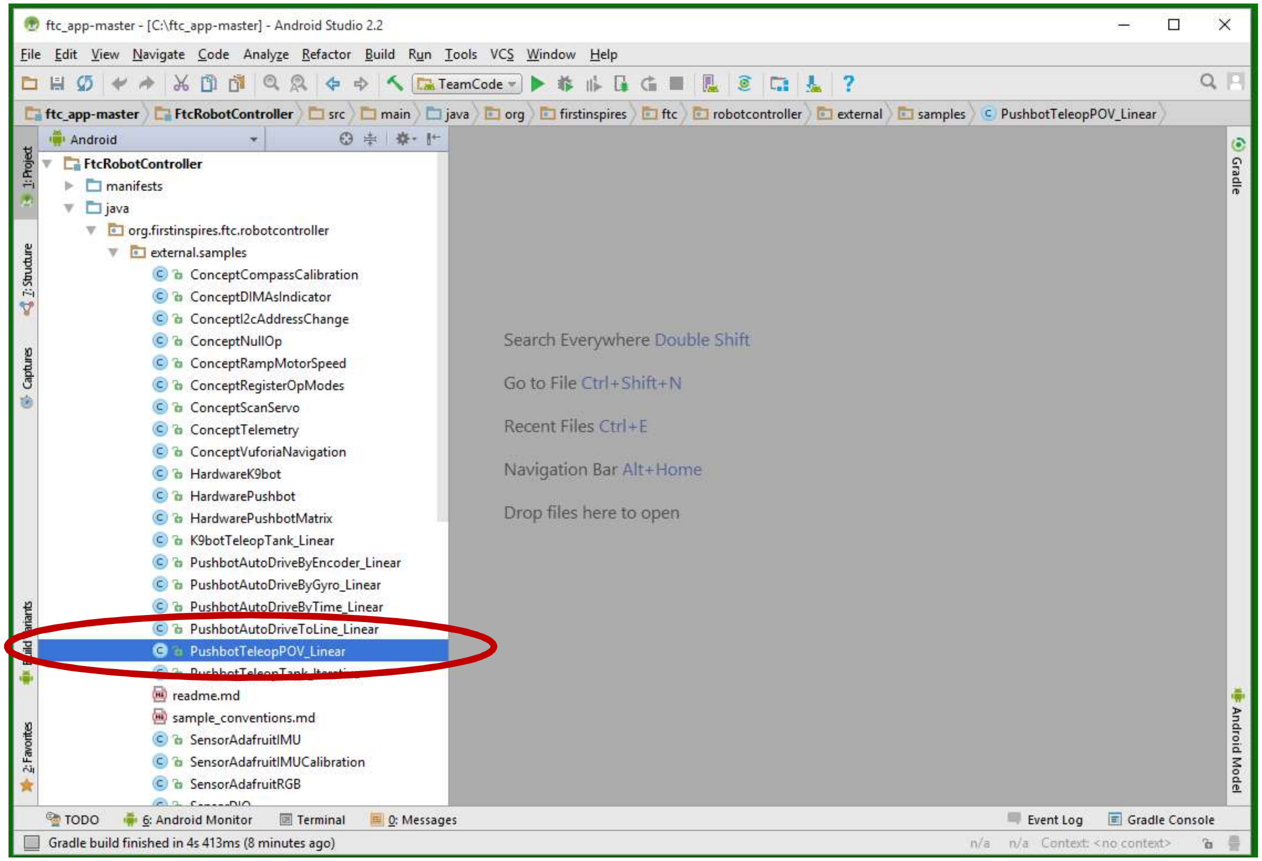
Guide

- This will open the samples folders similar to the screen on the left.
- Scroll down the page until you find “PushbotTeleopPOV_Linear.java”.
- Click on the “PushbotTeleopPOV_Linear.java” file to select it. (see red oval)

FROG FORCE 503 A LEAP AHEAD

How to Make changes

How to make a copy of a sample program



ftc_app-master - [C:\ftc_app-master] - Android Studio 2.2

- FtcRobotController
  - manifests
  - java
    - org.firstinspires.ftc.robotcontroller
      - external.samples
        - ConceptCompassCalibration
        - ConceptDIMAsIndicator
        - ConceptI2cAddressChange
        - ConceptNullOp
        - ConceptRampMotorSpeed
        - ConceptRegisterOpModes
        - ConceptScanServo
        - ConceptTelemetry
        - ConceptVuforiaNavigation
        - HardwareK9bot
        - HardwarePushbot
        - HardwarePushbotMatrix
        - K9botTeleopTank_Linear
        - PushbotAutoDriveByEncoder_Linear
        - PushbotAutoDriveByGyro_Linear
        - PushbotAutoDriveByTime_Linear
        - PushbotAutoDriveToLine_Linear
        - PushbotTeleopPOV_Linear
        - PushbotTeleopTank_Iterati
        - readme.md
        - sample_conventions.md
        - SensorAdafruitIMU
        - SensorAdafruitIMUCalibration
        - SensorAdafruitRGB
        - SensorDIO

Search Everywhere Double Shift

Go to File Ctrl+Shift+N

Recent Files Ctrl+E

Navigation Bar Alt+Home

Drop files here to open

Gradle build finished in 4s 413ms (8 minutes ago)

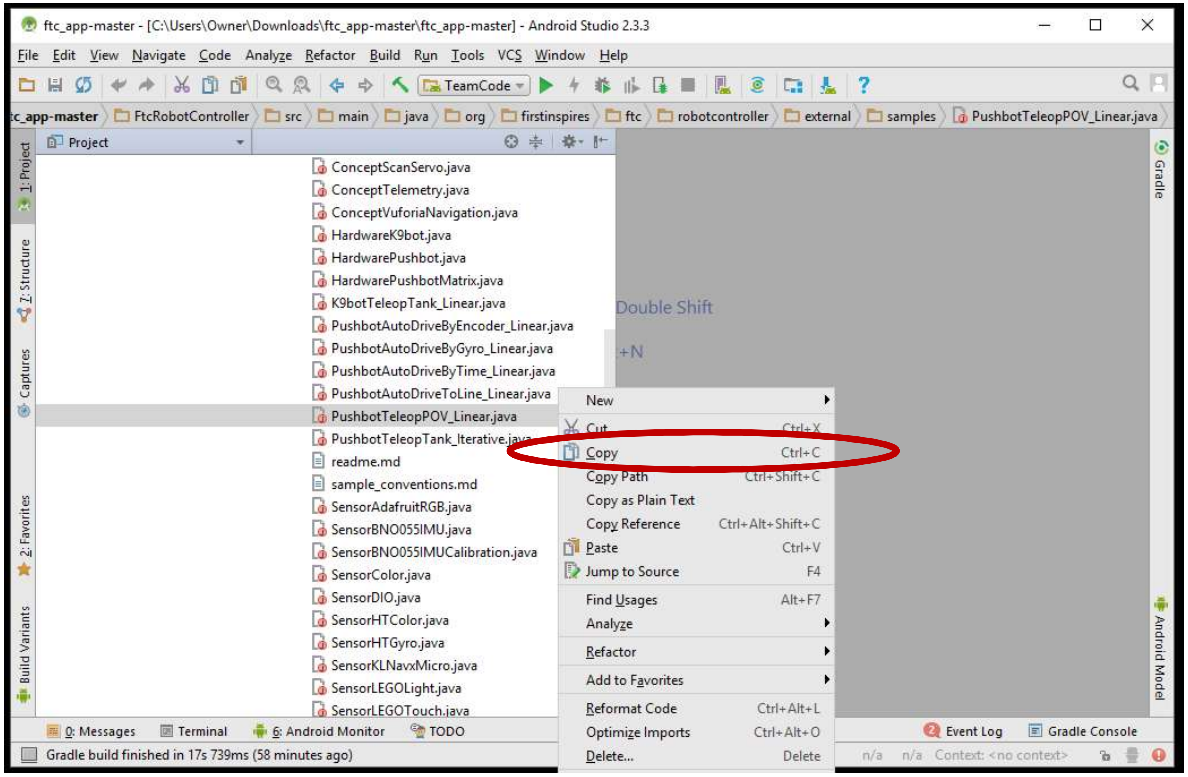
n/a n/a Context: <no context>

Guide – Follow these steps

- Select the sample program you want to copy
  - The example screen on the left shows the **PushbotTeleopPOV_Linear** sample program selected.
- On the mouse, click the right mouse button to bring up the sub menu

How to Make changes

How to make a copy of a sample program



ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - Android Studio 2.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
c_app-master FtcRobotController src main java org firstinspires ftc robotcontroller external samples PushbotTeleopPOV_Linear.java
Project
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
ConceptScanServo.java
ConceptTelemetry.java
ConceptVuforiaNavigation.java
HardwareK9bot.java
HardwarePushbot.java
HardwarePushbotMatrix.java
K9botTeleopTank_Linear.java
PushbotAutoDriveByEncoder_Linear.java
PushbotAutoDriveByGyro_Linear.java
PushbotAutoDriveByTime_Linear.java
PushbotAutoDriveToLine_Linear.java
PushbotTeleopPOV_Linear.java
PushbotTeleopTank_Iterative.java
readme.md
sample_conventions.md
SensorAdafruitRGB.java
SensorBNO055IMU.java
SensorBNO055IMUCalibration.java
SensorColor.java
SensorDIO.java
SensorHTColor.java
SensorHTGyro.java
SensorKLNavxMicro.java
SensorLEGOLight.java
SensorLEGOTouch.java
Double Shift
+N
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Jump to Source F4
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console
n/a n/a Context: <no context>

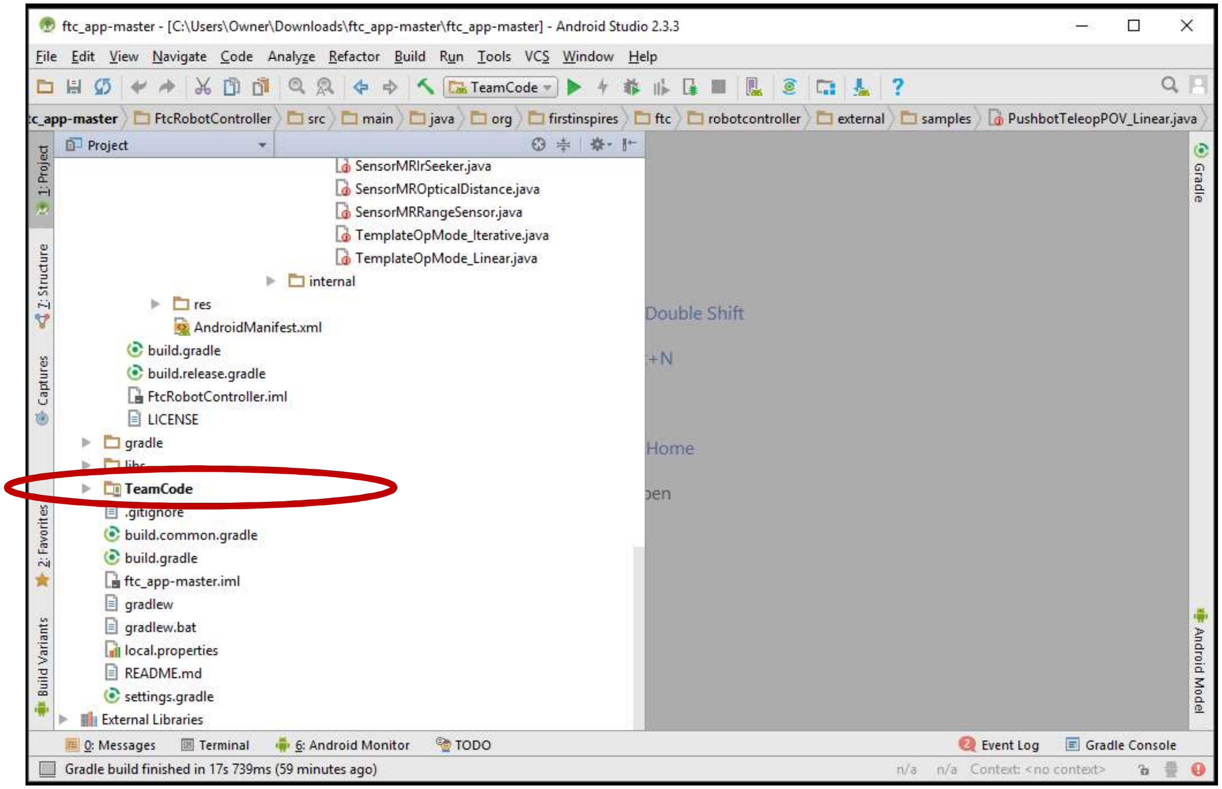
Gradle build finished in 17s 739ms (58 minutes ago)

Guide – Follow these steps

- Select Copy. (See red oval)

How to Make changes

How to make a copy of a sample program



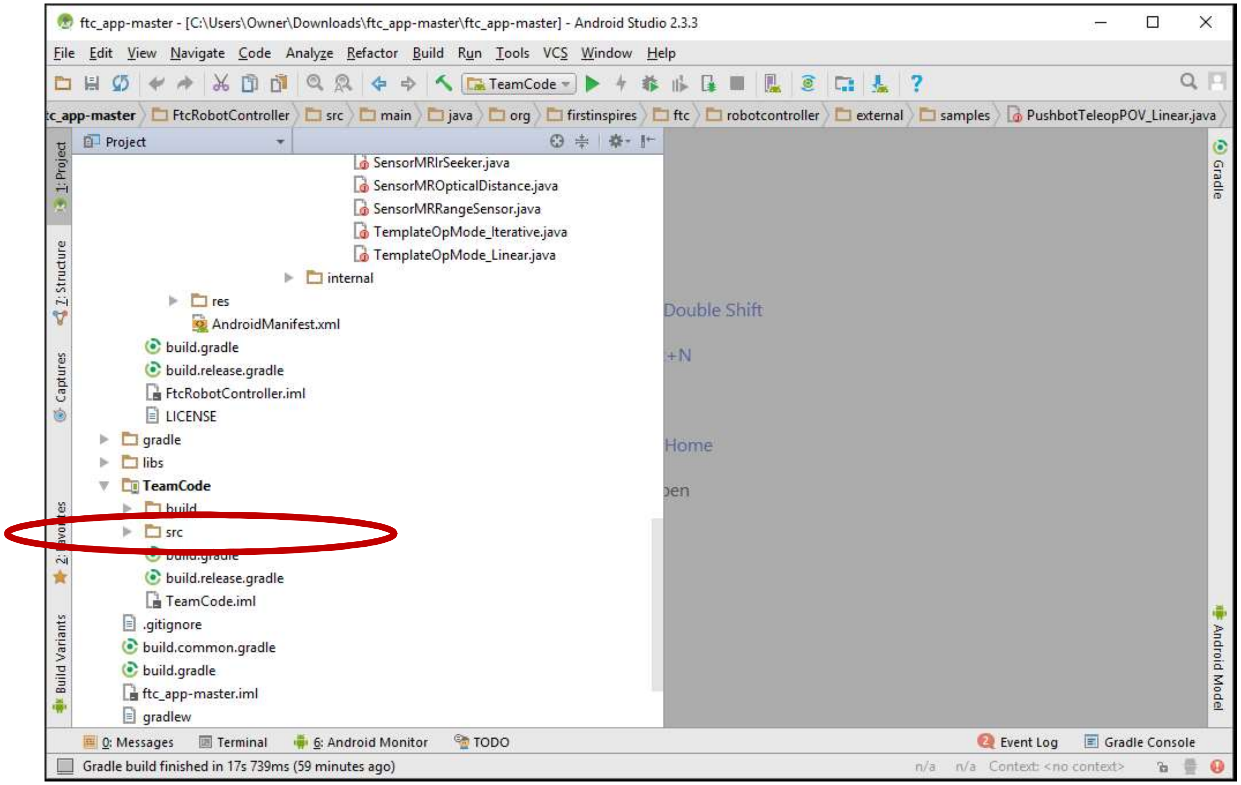
Gradle build finished in 17s 739ms (59 minutes ago)

Guide – Follow these steps

- Scroll down the Project navigation tree until you find the “TeamCode” folder.
- Click the “TeamCode” spin down button. (See red oval)

How to Make changes

How to make a copy of a sample program



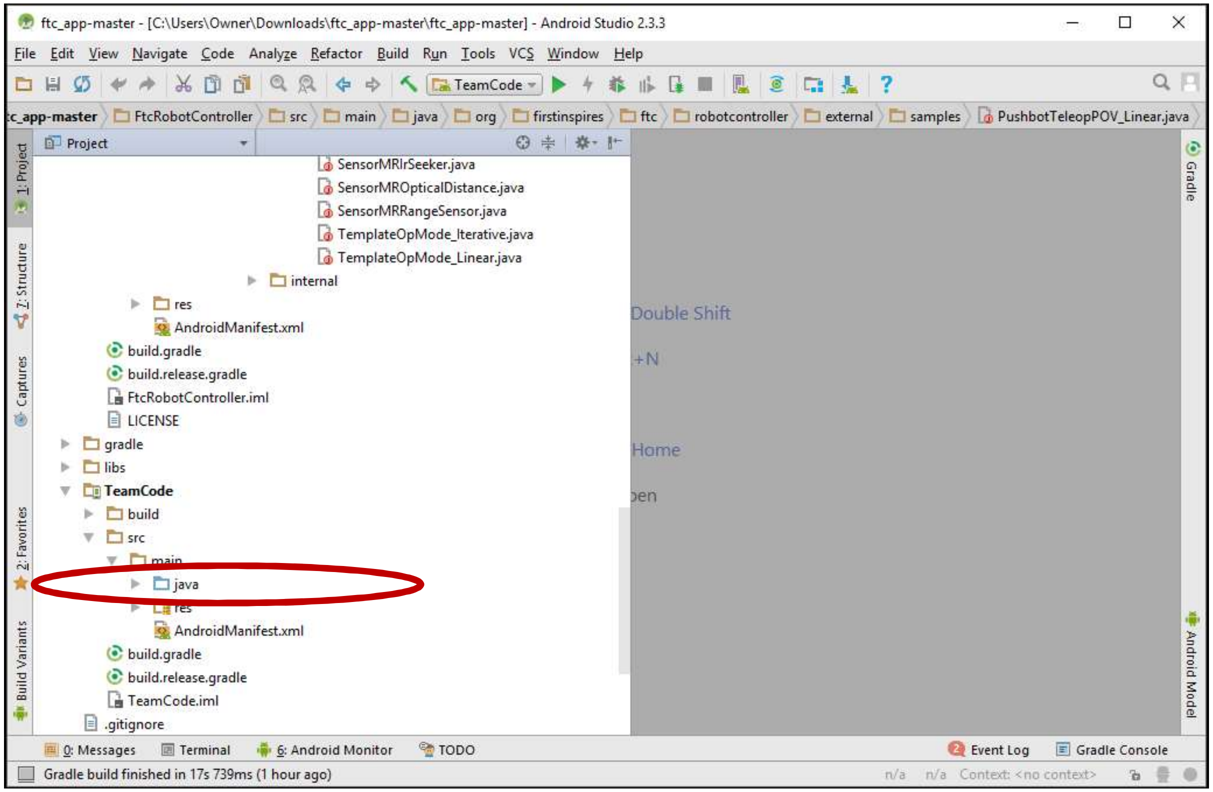
Gradle build finished in 17s 739ms (59 minutes ago)

Guide – Follow these steps

- Click the “src” spin down button. (See red oval)

How to Make changes

How to make a copy of a sample program



ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - Android Studio 2.3.3

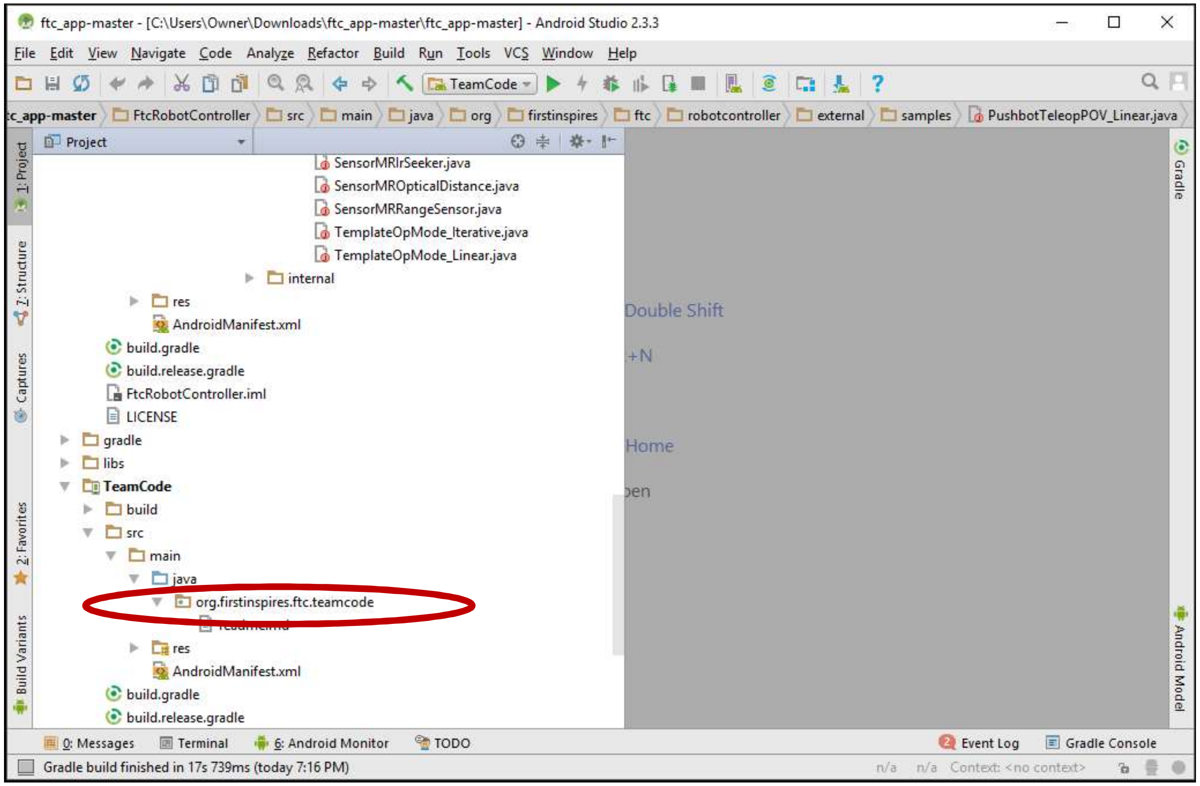
Gradle build finished in 17s 739ms (1 hour ago)

Guide – Follow these steps

- Click the “java” spin down button. (See red oval)

How to Make changes

How to make a copy of a sample program



ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - Android Studio 2.3.3

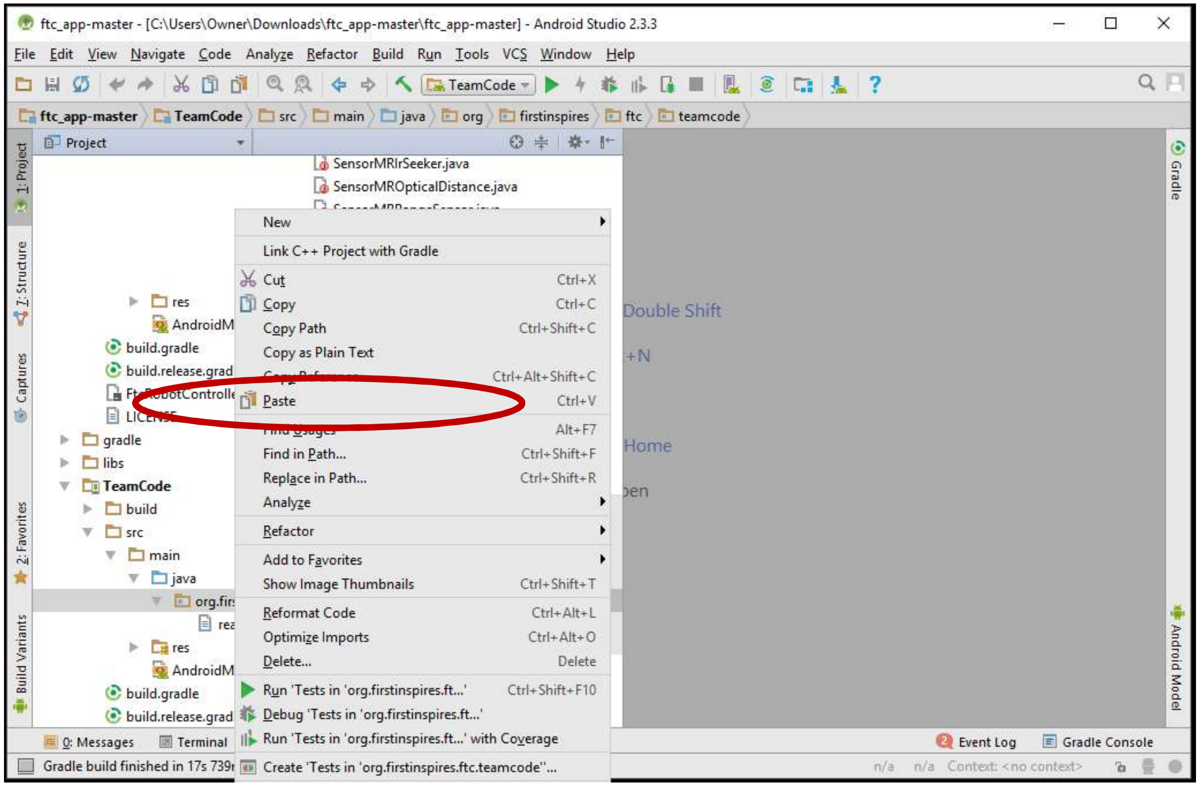
Gradle build finished in 17s 739ms (today 7:16 PM)

Guide – Follow these steps

- This will open the org.firstinspires.ftc.teamcode folder. (You should see a screen similar to the one on the left.
- Click the “org.firstinspires.ftc.teamcode” folder to select it.
- On the mouse, click the mouse right button to bring up the sub menu. (See red oval)

How to Make changes

How to make a copy of a sample program


ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - Android Studio 2.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-master 〉 TeamCode 〉 src 〉 main 〉 java 〉 org 〉 firstinspires 〉 ftc 〉 teamcode
Project
SensorMRIrSeeker.java
SensorMROpticalDistance.java
res
AndroidM
build.gradle
build.release.grad
FtcRobotControlle
LICENSE
gradle
libs
TeamCode
build
src
main
java
org.firs
rea
res
AndroidM
build.gradle
build.release.grad
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run 'Tests in 'org.firstinspires.ft...' Ctrl+Shift+F10
Debug 'Tests in 'org.firstinspires.ft...'
Run 'Tests in 'org.firstinspires.ft...' with Coverage
Create 'Tests in 'org.firstinspires.ftc.teamcode''...
Double Shift
+N
Home
pen
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
0: Messages
Terminal
Event Log
Gradle Console
n/a n/a Context: <no context>

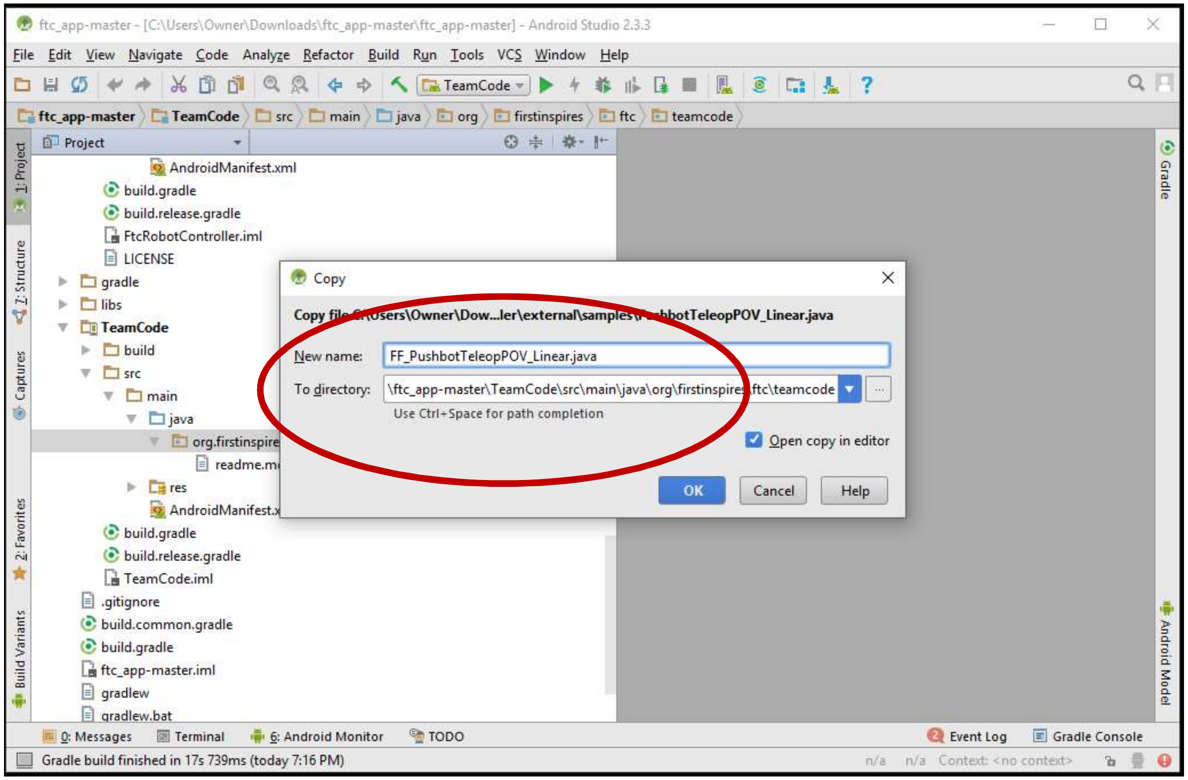
Gradle build finished in 17s 739r

Guide – Follow these steps

- This will open the context menu for the folder. You screen should look similar to the one on the left.
- Click 'Paste. (See red oval)

How to Make changes

How to make a copy of a sample program



Copy file C:\Users\Owner\Dow...ler\external\samples\PushbotTeleopPOV_Linear.java

New name: FF_PushbotTeleopPOV_Linear.java

To directory: \ftc_app-master\TeamCode\src\main\java\org\firstinspires\ftc\teamcode

Use Ctrl+Space for path completion

☑ Open copy in editor

Gradle build finished in 17s 739ms (today 7:16 PM)

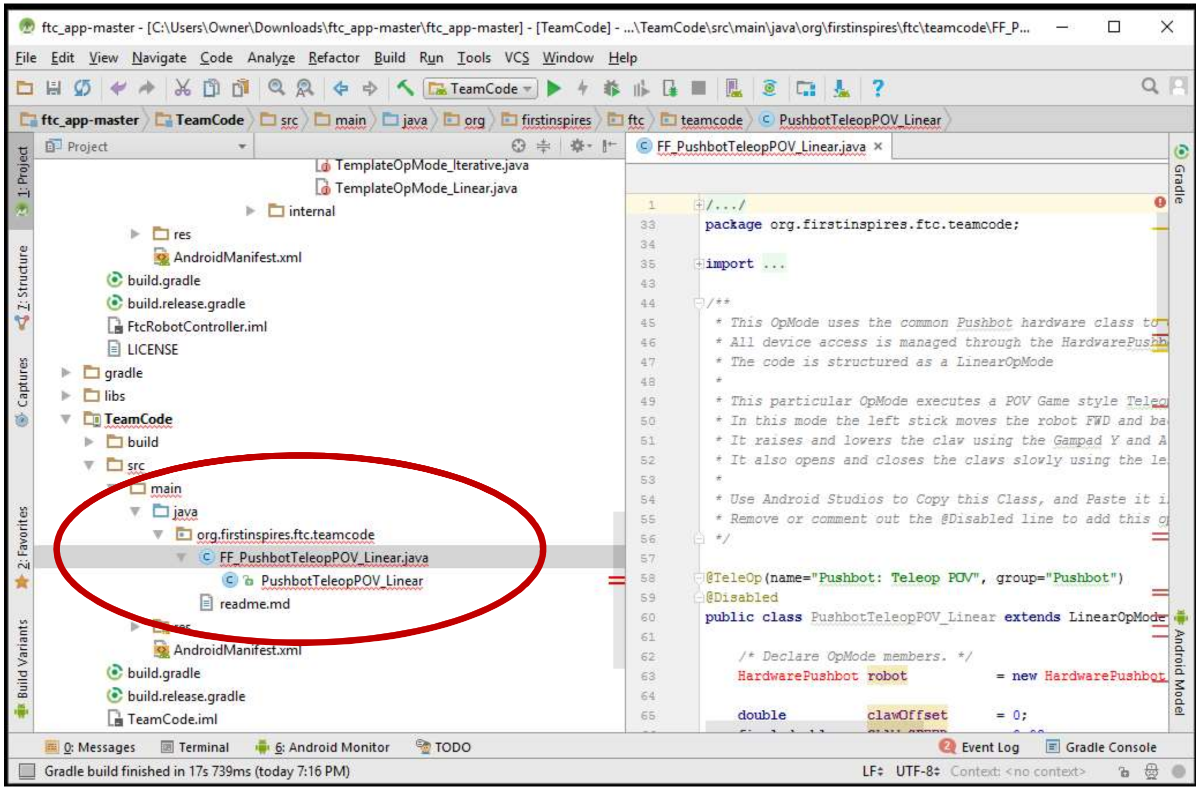
Guide – Follow these steps

- Enter the name you want for your copied sample program. In this example we entered **FF_PushbotTeleopPOV_Linear**
- Click “OK”

- ***Note: You do not need to rename the file when you copy it. But we are pretty sure that you will need to rename something as you develop your code, so we are going to walk you though renaming a file so that you know what to do when it comes up.***

How to Make changes

How to make a copy of a sample program


ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - [TeamCode] - ...\TeamCode\src\main\java\org\firstinspires\ftc\teamcode\FF_P...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-master TeamCode src main java org firstinspires ftc teamcode PushbotTeleopPOV_Linear
Project
1: Project
7: Structure
Captures
2: Favorites
Build Variants
TemplateOpMode_Iterative.java
TemplateOpMode_Linear.java
internal
res
AndroidManifest.xml
build.gradle
build.release.gradle
FtcRobotController.iml
LICENSE
gradle
libs
TeamCode
build
src
main
java
org.firstinspires.ftc.teamcode
FF_PushbotTeleopPOV_Linear.java
PushbotTeleopPOV_Linear
readme.md
res
AndroidManifest.xml
build.gradle
build.release.gradle
TeamCode.iml
FF_PushbotTeleopPOV_Linear.java
Gradle
Android Model
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console
LF⬍ UTF-8⬍ Context: <no context>

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

import ...

/**
 * This OpMode uses the common Pushbot hardware class to
 * All device access is managed through the HardwarePushb
 * The code is structured as a LinearOpMode
 *
 * This particular OpMode executes a POV Game style Teleo
 * In this mode the left stick moves the robot FWD and ba
 * It raises and lowers the claw using the Gampad Y and A
 * It also opens and closes the claws slowly using the le
 *
 * Use Android Studios to Copy this Class, and Paste it i
 * Remove or comment out the @Disabled line to add this o
 */

@TeleOp(name="Pushbot: Teleop POV", group="Pushbot")
@Disabled
public class PushbotTeleopPOV_Linear extends LinearOpMode

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot

    double          clawOffset      = 0;
```

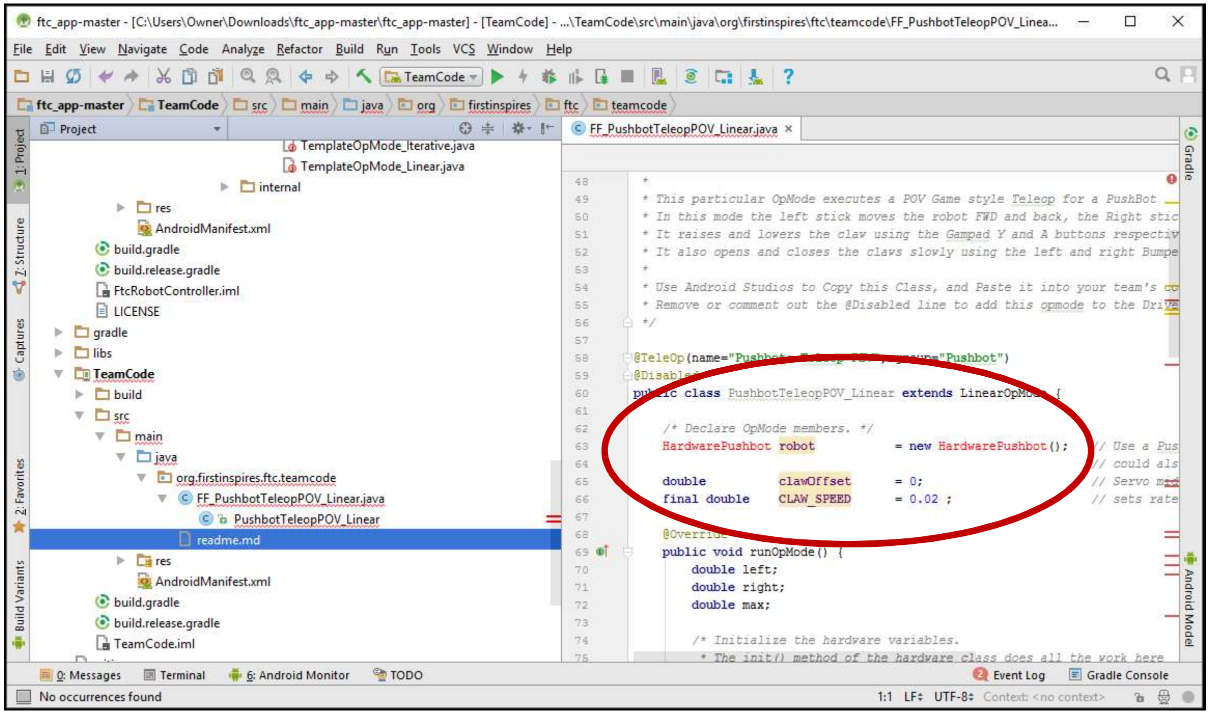
Gradle build finished in 17s 739ms (today 7:16 PM)

Guide – Follow these steps

- Once the paste completes you will see screen similar to the one on the left
- Note that there are squiggly red lines under the newly pasted FF_Pushbot_TeleopPOV_Linear" class.
- This is because we renamed the file and did not change the class name inside the class.
- We now need to change code inside the class to correct the error we just created.

How to Make changes

How to make a copy of a sample program


ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - [TeamCode] - ...\TeamCode\src\main\java\org\firstinspires\ftc\teamcode\FF_PushbotTeleopPOV_Linea...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-master > TeamCode > src > main > java > org > firstinspires > ftc > teamcode
Project
TemplateOpMode_Iterative.java
TemplateOpMode_Linear.java
internal
res
AndroidManifest.xml
build.gradle
build.release.gradle
FtcRobotController.iml
LICENSE
gradle
libs
TeamCode
build
src
main
java
org.firstinspires.ftc.teamcode
FF_PushbotTeleopPOV_Linear.java
PushbotTeleopPOV_Linear
readme.md
res
AndroidManifest.xml
build.gradle
build.release.gradle
TeamCode.iml
FF_PushbotTeleopPOV_Linear.java
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console
No occurrences found
1:1 LF÷ UTF-8÷ Context: <no context>

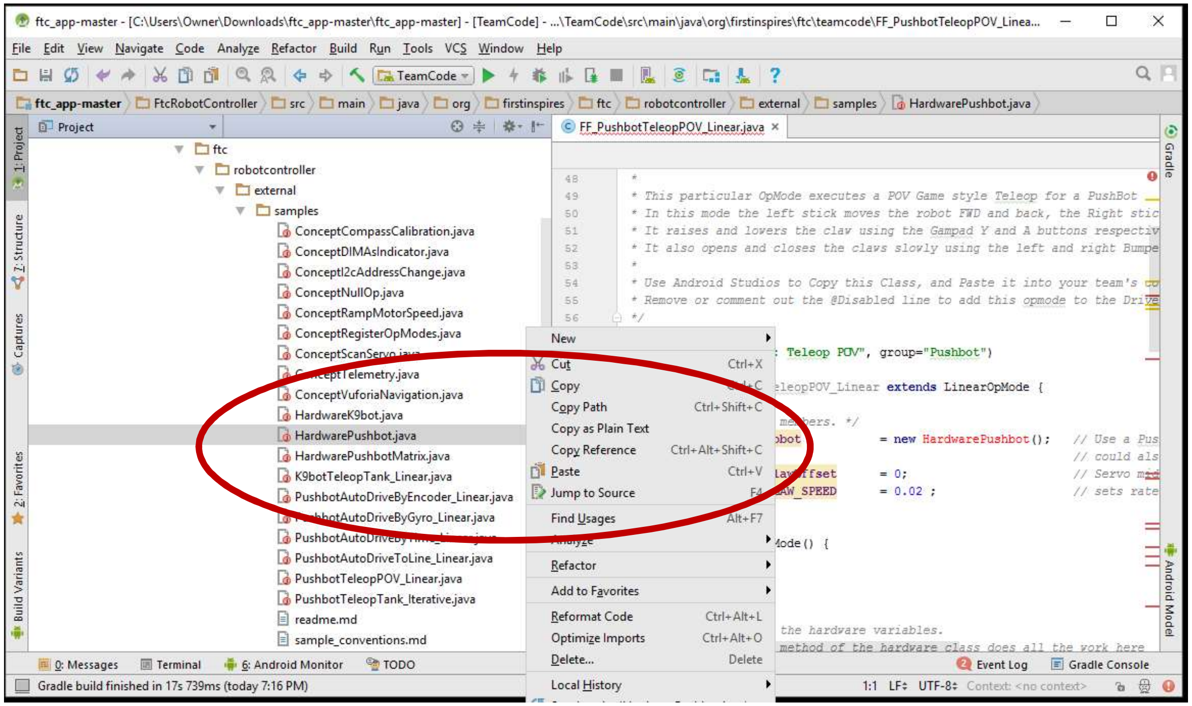
```
 *
 * This particular OpMode executes a POV Game style Teleop for a PushBot
 * In this mode the left stick moves the robot FWD and back, the Right stic
 * It raises and lowers the claw using the Gampad Y and A buttons respectiv
 * It also opens and closes the claws slowly using the left and right Bumpe
 *
 * Use Android Studios to Copy this Class, and Paste it into your team's co
 * Remove or comment out the @Disabled line to add this opmode to the Drive
 */

@TeleOp(name="Pushbot: Teleop POV", group="Pushbot")
@Disabled
public class PushbotTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot();   // Use a Pus
                                                               // could als
    double          clawOffset      = 0;                       // Servo mid
    final double    CLAW_SPEED      = 0.02 ;                   // sets rate

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

        /* Initialize the hardware variables.
         * The init() method of the hardware class does all the work here
```

Guide – Follow these steps

- Notice that the “HardwarePushBot robot = new HardwarePushBot(); line is red. (see red oval)
- This is because the PushBot_TeleopPOV_Linear class references another class called HardwarePushBot.
- This class (file) is missing from the org.firstinspires.ftc.teamcode folder.
- WE need to go back and copy that file from the external examples folder. We now need to change code inside the class to correct the error we just created.

How to Make changes

How to make a copy of a sample program


ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - [TeamCode] - ...\TeamCode\src\main\java\org\firstinspires\ftc\teamcode\FF_PushbotTeleopPOV_Linea...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-master FtcRobotController src main java org firstinspires ftc robotcontroller external samples HardwarePushbot.java
Project
FF_PushbotTeleopPOV_Linear.java
ftc
robotcontroller
external
samples
ConceptCompassCalibration.java
ConceptDIMAsIndicator.java
ConceptI2cAddressChange.java
ConceptNullOp.java
ConceptRampMotorSpeed.java
ConceptRegisterOpModes.java
ConceptScanServo.java
ConceptTelemetry.java
ConceptVuforiaNavigation.java
HardwareK9bot.java
HardwarePushbot.java
HardwarePushbotMatrix.java
K9botTeleopTank_Linear.java
PushbotAutoDriveByEncoder_Linear.java
PushbotAutoDriveByGyro_Linear.java
PushbotAutoDriveByTime_Linear.java
PushbotAutoDriveToLine_Linear.java
PushbotTeleopPOV_Linear.java
PushbotTeleopTank_Iterative.java
readme.md
sample_conventions.md
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Jump to Source F4
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Local History
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console
1:1 LF UTF-8 Context: <no context>

```
 *
 * This particular OpMode executes a POV Game style Teleop for a PushBot
 * In this mode the left stick moves the robot FWD and back, the Right stic
 * It raises and lowers the claw using the Gampad Y and A buttons respectiv
 * It also opens and closes the claws slowly using the left and right Bumpe
 *
 * Use Android Studios to Copy this Class, and Paste it into your team's co
 * Remove or comment out the @Disabled line to add this opmode to the Drive
 */

: Teleop POV", group="Pushbot")

eleopPOV_Linear extends LinearOpMode {

members. */
obot            = new HardwarePushbot();   // Use a Pus
                                            // could als
lawOffset       = 0;                        // Servo mid
LAW_SPEED       = 0.02 ;                    // sets rate

Mode() {

the hardware variables.
method of the hardware class does all the work here
```

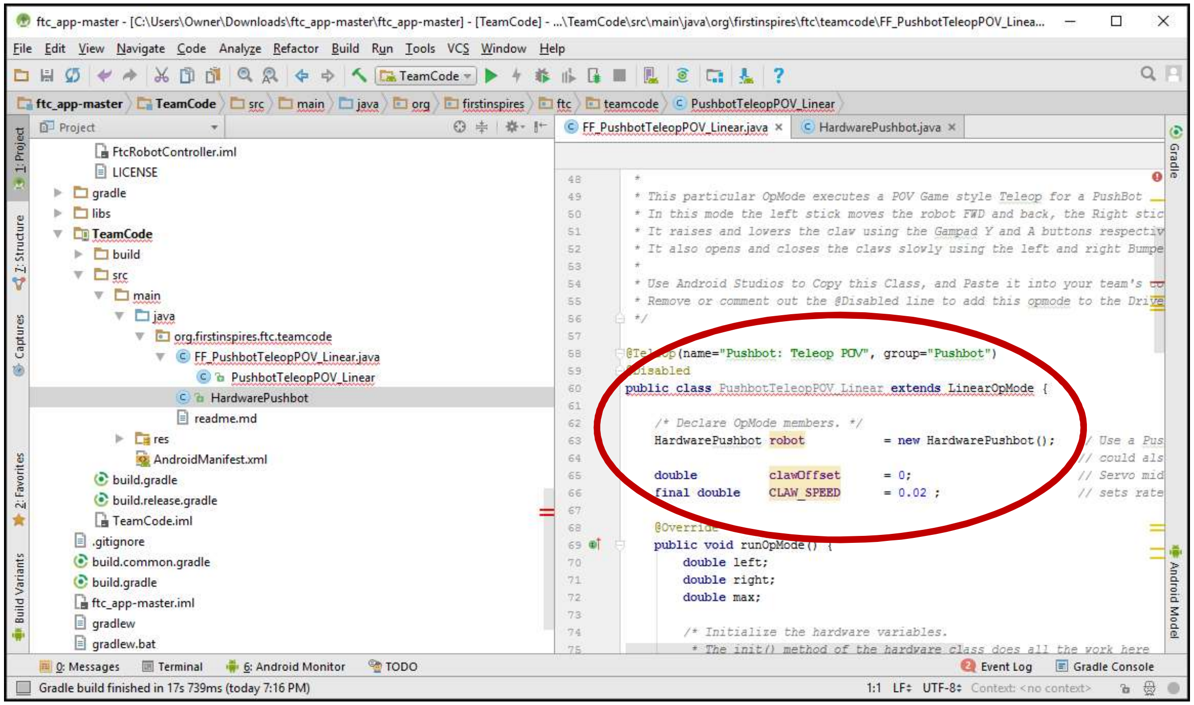
Gradle build finished in 17s 739ms (today 7:16 PM)

Guide – Follow these steps

- Go back to the external samples folder and find the file named “HardwarePushbot.java”.
- Follow the steps outlined previously to copy and paste it into the org.firstinspires.ftc.teamcode folder.
- I.e:
  - Select the HardwarePushbot file
  - Right mouse click to bring up sub menu
  - Select copy
  - Find the org.firstinspires.ftc.teamcode folder and select it
  - Right mouse click to bring up sub menu
  - Click Paste
  - When Past panel is display do not change the name and click the “OK” button.

How to Make changes

How to make a copy of a sample program



```
 *
 * This particular OpMode executes a POV Game style Teleop for a PushBot
 * In this mode the left stick moves the robot FWD and back, the Right stic
 * It raises and lowers the claw using the Gampad Y and A buttons respectiv
 * It also opens and closes the claws slowly using the left and right Bumpe
 *
 * Use Android Studios to Copy this Class, and Paste it into your team's co
 * Remove or comment out the @Disabled line to add this opmode to the Drive
 */

@Teleop(name="Pushbot: Teleop POV", group="Pushbot")
@Disabled
public class PushbotTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot();   // Use a Pus
                                                               // could als
    double          clawOffset      = 0;                       // Servo mid
    final double    CLAW_SPEED      = 0.02 ;                   // sets rate

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

        /* Initialize the hardware variables.
         * The init() method of the hardware class does all the work here
```

Gradle build finished in 17s 739ms (today 7:16 PM)

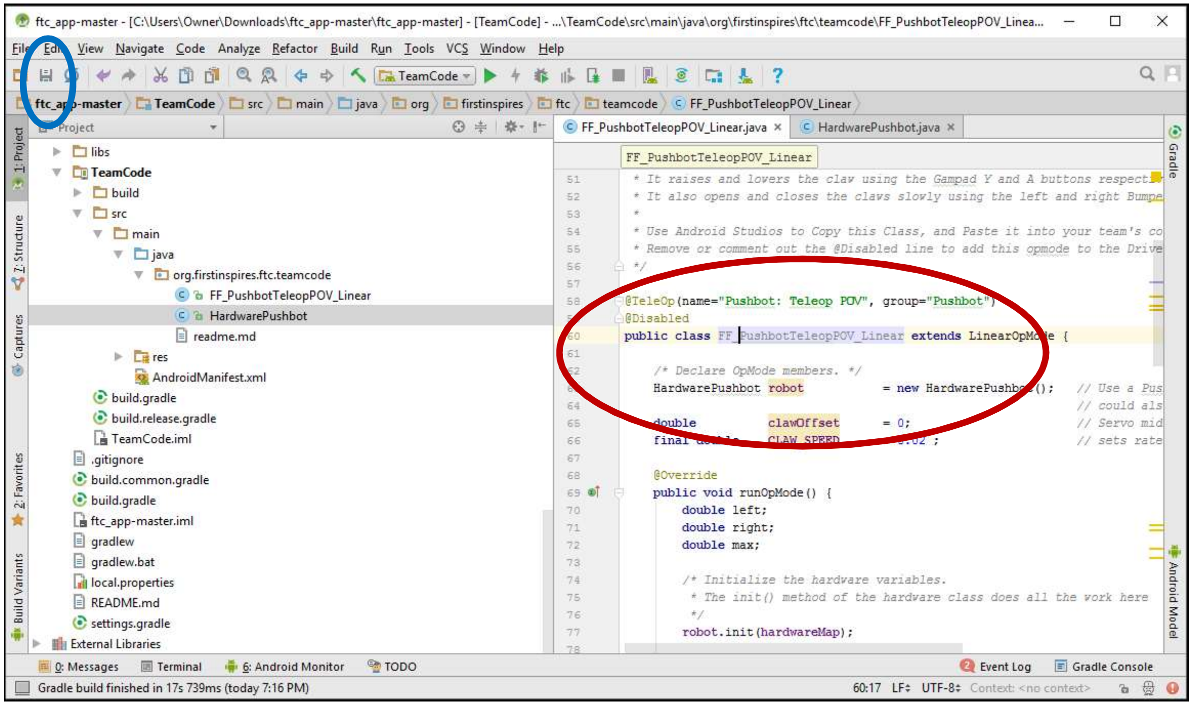
1:1 LF‡ UTF-8‡

Guide – Follow these steps

- You should now see a screen similar to the one on the left.
- Note that the HardwarePushBot file is now listed under the org.firstinspires_ftc_teamcode folder.
- Also notice that the HardwarePushBot robot line is no longer red. (The error has been fixed!)
- However, the public class PushbotTeleopPOV_Linera line is now red. (See red oval)
- This is because we named the file FF_PushbotTeleopPOV_Linear.
- Time to correct this error.

How to Make changes

How to make a copy of a sample program



ftc_app-master - [C:\Users\Owner\Downloads\ftc_app-master\ftc_app-master] - [TeamCode] - ...\TeamCode\src\main\java\org\firstinspires\ftc\teamcode\FF_PushbotTeleopPOV_Linea...

```java
 * It raises and lowers the claw using the Gampad Y and A buttons respecti
 * It also opens and closes the claws slowly using the left and right Bumpe
 *
 * Use Android Studios to Copy this Class, and Paste it into your team's co
 * Remove or comment out the @Disabled line to add this opmode to the Drive
 */

@TeleOp(name="Pushbot: Teleop POV", group="Pushbot")
@Disabled
public class FF_PushbotTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot();   // Use a Pus
                                                               // could als
    double          clawOffset      = 0;                       // Servo mid
    final double    CLAW_SPEED      = 0.02 ;                   // sets rate

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

        /* Initialize the hardware variables.
         * The init() method of the hardware class does all the work here
         */
        robot.init(hardwareMap);
```

Gradle build finished in 17s 739ms (today 7:16 PM)

60:17 LF⇕ UTF-8⇕ Context: <no context>

Guide – Follow these steps

- Simply change the “public class PushbotTeleopPOV_Linear” line to FF_PushbotTeleopPOV_Linear. (See red oval)
- Note that the squiggly red lines are now gone as all the errors have been corrected.
- Don’t forget to save your changes. Click the “Save” icon. (See Blue Oval)

Congratulations you have copied an OpMode and made changes for your team!

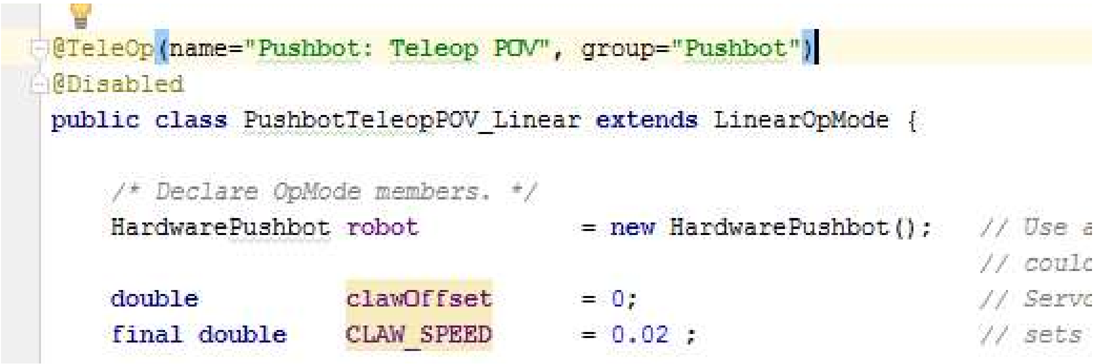
What is an Op Mode?
Definition

An Operational Mode (Op Mode) is a software module stored on the robot controller that you can execute from the driver station.

These op-mode software modules contain pre-programmed behaviors for your robot.

Registering an OpMode – Method 1

In 2016, FIRST has developed a new method to register OpModes – Compiler Directives



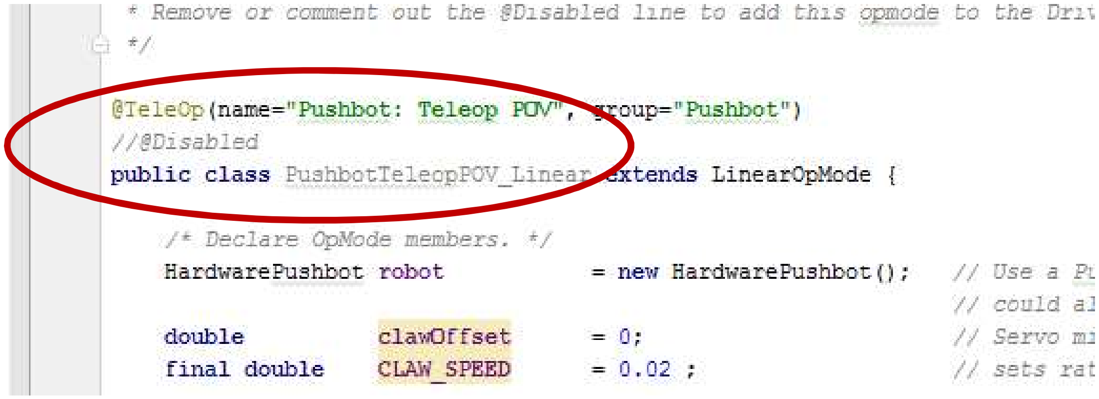
```
@TeleOp(name="Pushbot: Teleop POV", group="Pushbot")
@Disabled
public class PushbotTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot();   // Use a
                                                               // coulc
    double          clawOffset      = 0;                       // Servc
    final double    CLAW_SPEED      = 0.02 ;                   // sets
```

Guide

- In the external examples folder you can find a sample program called **PushbotTeleopPOV_Linear**
  - Open Android Studio and the FTC_App-master
  - Click the expand arrow for the FtcRobotController
  - Click the expand arrow on java
  - Click the expand arrow on org.firstinspires.ftc.robotcontroller
  - Click the expand arrow on external.samples
  - Select PushbotTeleopPOV_Linear
  - You will see the code snippet on the left
- The @ commands are instructions (directives) for the compiler
- **@Teleop registers an OpMode with the title “Pushbot: Teleop POV on the Driver station in a group called Pushbot**
  - Groups are a way to organize your OpModes on the driver station making them easier to find
- **By default this command is disabled (Note the @Disabled)**
  - This tells the compiler to ignore this command every time you compile the code

Registering an OpMode – Method 1

In 2016, FIRST has developed a new method to register OpModes – Class Annotations



```
 * Remove or comment out the @Disabled line to add this opmode to the Driv
 */

@TeleOp(name="Pushbot: Teleop POV", group="Pushbot")
//@Disabled
public class PushbotTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot();   // Use a Pu
                                                               // could al
    double          clawOffset      = 0;                       // Servo mi
    final double    CLAW_SPEED      = 0.02 ;                   // sets rat
```

Guide

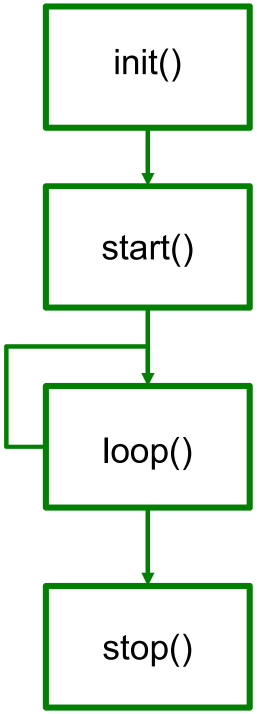
- You can comment out the disable command by inserting a // in front of the @Disabled
  - *This is the standard way in Java to insert a comment in your code – in effect you are commenting out the disabled class annotation*
- This will "enable" the command and register it on your Driverstation/Robot Controller
- Don't forget to save your changes!
- Now everytime you compile your program you will ensure that this OpMode is registeres so that you can execute it on the robot.
- This is the preferred method of registering Opmodes.
- There is another method for register OpModes. This involves changing the FtcOpModeRegister.java program in the internal samples folder. But this makes your code hard to maintain and is discourage by FIRST so we will skip those instructions-just know that there is a second method if you get stuck.

Anatomy of an OpMode
Overview

- FIRST has developed two methods for controlling opModes; **Iterative and Linear**
- **Iterative**
  - Is the same as last year
  - 4 methods are required for every program (Init, Start, Loop, Stop)
  - The Loop method is called repeatedly by the control program every 10-20 milliseconds for the entire Teleop Period
- **Linear**
  - Is new for the 2016-2017 season
  - You write your own loop within the program
  - You must call the WaitForStart() function before your main loop
  - You must add a robot.waitForTick(40) function to allow the robot control to run other functions

Anatomy of an OpMode – Iterative

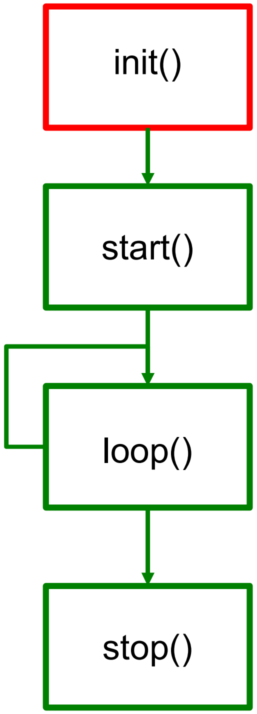
Overview



Guide

- Every opMode ***should/can*** contain 4 methods:
  - Init
  - Start
  - Loop
  - Stop
- Not all methods are required for every Op Mode

Anatomy of an OpMode – Iterative
init – Initialization

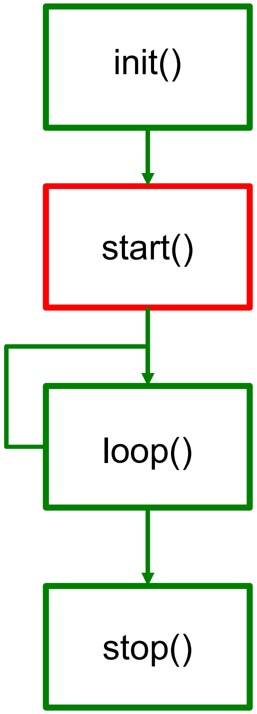


Guide

- This is for initialization tasks
- It is executed only once
- The robot is updated after the method exits-not as each line is the code is executing

Anatomy of an OpMode – Iterative

start – tasks before loop

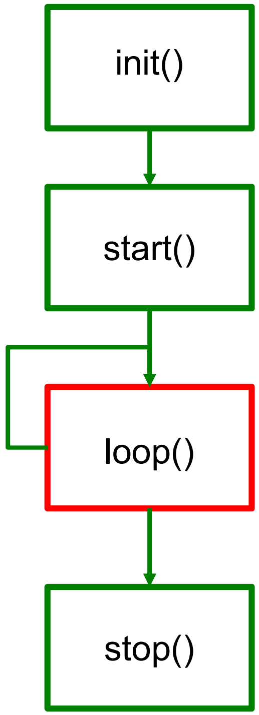


Guide

- This method is triggered when the driver pushes “Start” button on the touch screen
- It is executed only once
- If you have any initialization tasks that you want to execute right before the “loop” method you do so here by adding a public void start() {} to your opmode

Anatomy of an OpMode – Iterative

loop – stuff to do repeatedly

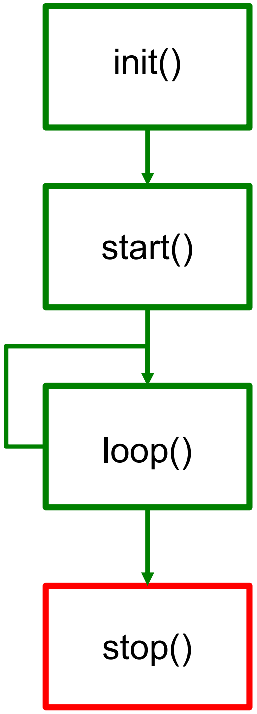


Guide

- When the driver pushed the “Start” button on the driver station, the code in the loop() method will execute repeatedly (approximately every 10-20 milliseconds)
- The robot controller app has a built in event loop that executes the contents of the loop() method repeatedly until a stop command is received from the driver station (or unless an emergency stop condition occurs)
- This is the method where you will put the bulk of your code

Anatomy of an OpMode – Iterative

stop – tasks to do when the loop is over

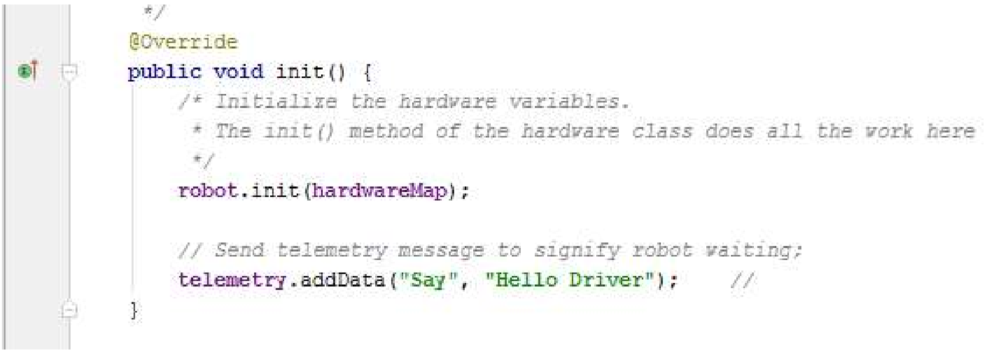


Guide

- When the robot controller receives a stop command from the driver station or when emergency stop() condition occurs, the code in the stop() method gets executed
- If you have an cleaning up to do after the opmode loop() method has been run, this is the place to put it
- Just like the start() method this method does nothing by default, you can override it's behavior by adding a public void stop() {} to your opmode
- Lets take a look at the sample code in Android Studio. We will use the K9TeleOp Op mode as it is a very straight forward for novice FTC programmers.

Anatomy of an OpMode – Iterative

init – PushbotTeleopTank_Iterative sample code



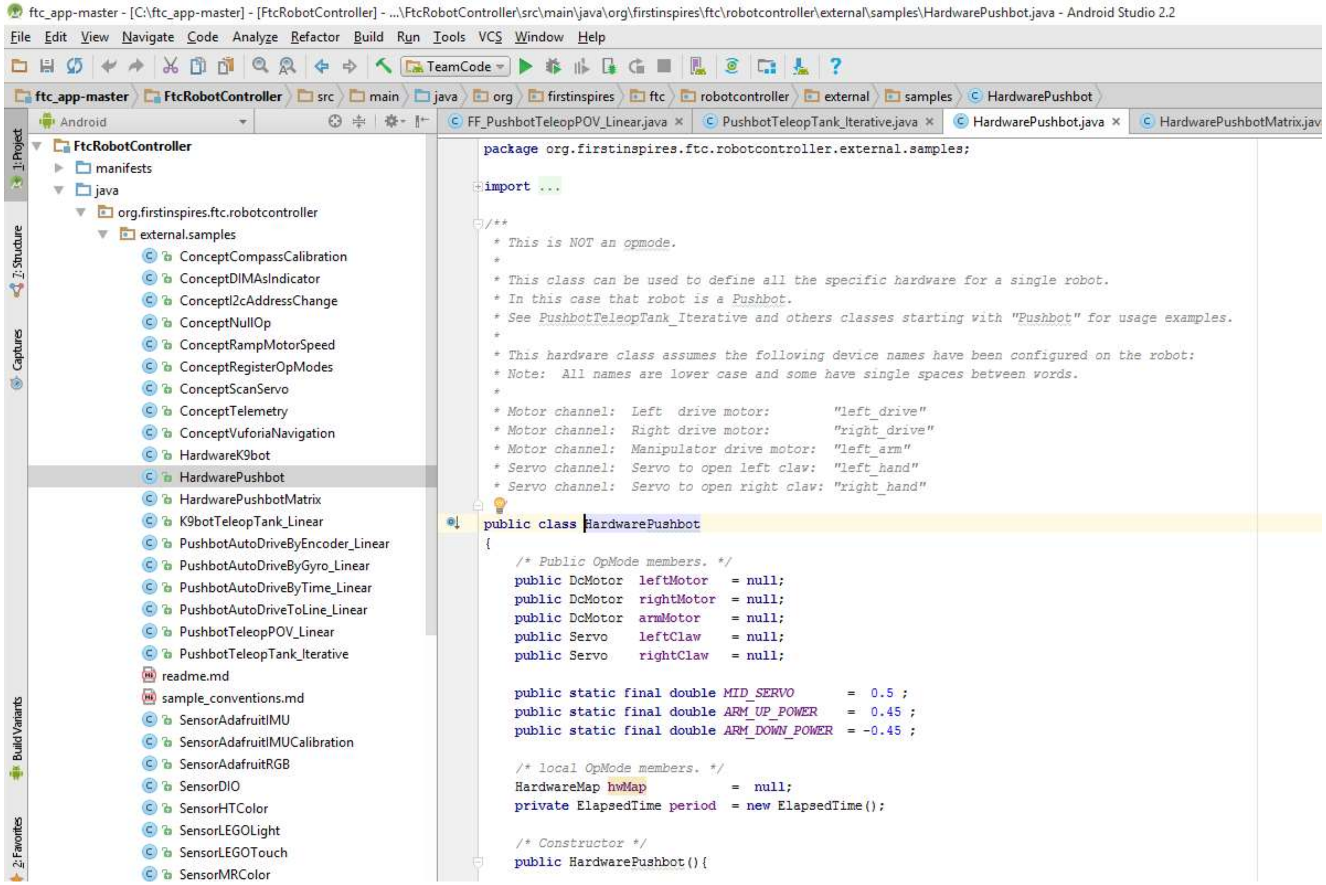
```
 */
@Override
public void init() {
    /* Initialize the hardware variables.
     * The init() method of the hardware class does all the work here
     */
    robot.init(hardwareMap);

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

Guide

- **The init method calls another program to get the hardware Map**
- **It also says Hello Driver to the Driver station when the OpMode initializes**

Anatomy of an OpMode – Iterative

init – PushbotTeleopTank_Iterative:HardwarePushbot sample code


ftc_app-master - [C:\ftc_app-master] - [FtcRobotController] - ...\FtcRobotController\src\main\java\org\firstinspires\ftc\robotcontroller\external\samples\HardwarePushbot.java - Android Studio 2.2
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-master FtcRobotController src main java org firstinspires ftc robotcontroller external samples HardwarePushbot
Android
FF_PushbotTeleopPOV_Linear.java
PushbotTeleopTank_Iterative.java
HardwarePushbot.java
HardwarePushbotMatrix.jav
1: Project
7: Structure
Captures
Build Variants
2: Favorites
FtcRobotController
manifests
java
org.firstinspires.ftc.robotcontroller
external.samples
ConceptCompassCalibration
ConceptDIMAsIndicator
ConceptI2cAddressChange
ConceptNullOp
ConceptRampMotorSpeed
ConceptRegisterOpModes
ConceptScanServo
ConceptTelemetry
ConceptVuforiaNavigation
HardwareK9bot
HardwarePushbot
HardwarePushbotMatrix
K9botTeleopTank_Linear
PushbotAutoDriveByEncoder_Linear
PushbotAutoDriveByGyro_Linear
PushbotAutoDriveByTime_Linear
PushbotAutoDriveToLine_Linear
PushbotTeleopPOV_Linear
PushbotTeleopTank_Iterative
readme.md
sample_conventions.md
SensorAdafruitIMU
SensorAdafruitIMUCalibration
SensorAdafruitRGB
SensorDIO
SensorHTColor
SensorLEGOLight
SensorLEGOTouch
SensorMRColor

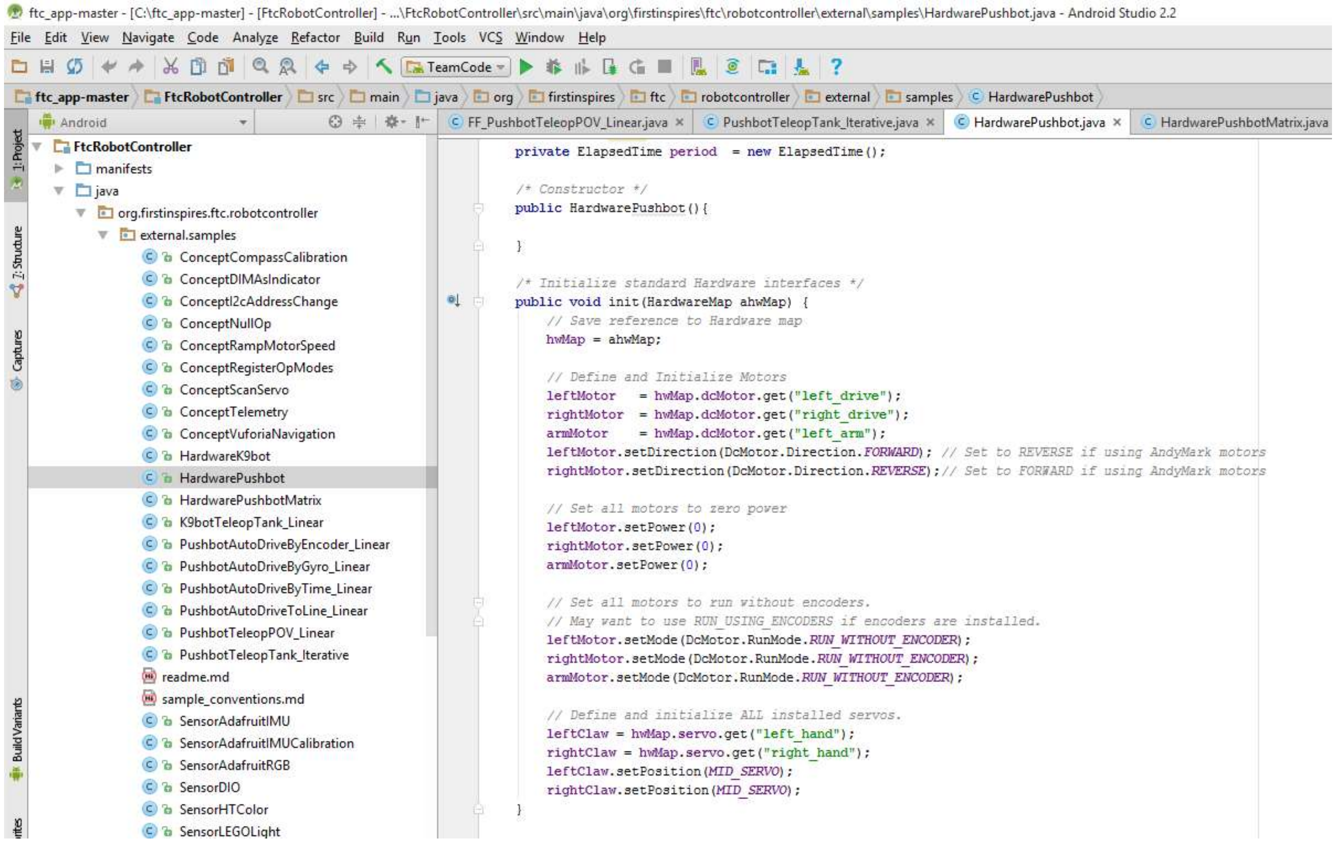
```
package org.firstinspires.ftc.robotcontroller.external.samples;

import ...

/**
 * This is NOT an opmode.
 *
 * This class can be used to define all the specific hardware for a single robot.
 * In this case that robot is a Pushbot.
 * See PushbotTeleopTank_Iterative and others classes starting with "Pushbot" for usage examples.
 *
 * This hardware class assumes the following device names have been configured on the robot:
 * Note:  All names are lower case and some have single spaces between words.
 *
 * Motor channel:  Left  drive motor:        "left_drive"
 * Motor channel:  Right drive motor:        "right_drive"
 * Motor channel:  Manipulator drive motor:  "left_arm"
 * Servo channel:  Servo to open left claw:  "left_hand"
 * Servo channel:  Servo to open right claw: "right_hand"

public class HardwarePushbot
{
    /* Public OpMode members. */
    public DcMotor  leftMotor   = null;
    public DcMotor  rightMotor  = null;
    public DcMotor  armMotor    = null;
    public Servo    leftClaw    = null;
    public Servo    rightClaw   = null;

    public static final double MID_SERVO       =  0.5 ;
    public static final double ARM_UP_POWER    =  0.45 ;
    public static final double ARM_DOWN_POWER  = -0.45 ;

    /* local OpMode members. */
    HardwareMap hwMap           =  null;
    private ElapsedTime period  = new ElapsedTime();

    /* Constructor */
    public HardwarePushbot(){
```

Guide

- **The Hardware Pushbot defines the hardware names and maps them to names used within the program**
- **These hardware names must be configured on the robot controller**
- **Note the leftMotor, rightMotor program names defined as DCMotors and set to null**
- **If you scroll down and look at the init method**

Anatomy of an OpMode – Iterative

init – PushbotTeleopTank_Iterative:HardwarePushbot sample code


ftc_app-master - [C:\ftc_app-master] - [FtcRobotController] - ...\FtcRobotController\src\main\java\org\firstinspires\ftc\robotcontroller\external\samples\HardwarePushbot.java - Android Studio 2.2
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TeamCode
ftc_app-master FtcRobotController src main java org firstinspires ftc robotcontroller external samples HardwarePushbot
Android
FF_PushbotTeleopPOV_Linear.java
PushbotTeleopTank_Iterative.java
HardwarePushbot.java
HardwarePushbotMatrix.java
1: Project
2: Structure
Captures
Build Variants
FtcRobotController
manifests
java
org.firstinspires.ftc.robotcontroller
external.samples
ConceptCompassCalibration
ConceptDIMAsIndicator
ConceptI2cAddressChange
ConceptNullOp
ConceptRampMotorSpeed
ConceptRegisterOpModes
ConceptScanServo
ConceptTelemetry
ConceptVuforiaNavigation
HardwareK9bot
HardwarePushbot
HardwarePushbotMatrix
K9botTeleopTank_Linear
PushbotAutoDriveByEncoder_Linear
PushbotAutoDriveByGyro_Linear
PushbotAutoDriveByTime_Linear
PushbotAutoDriveToLine_Linear
PushbotTeleopPOV_Linear
PushbotTeleopTank_Iterative
readme.md
sample_conventions.md
SensorAdafruitIMU
SensorAdafruitIMUCalibration
SensorAdafruitRGB
SensorDIO
SensorHTColor
SensorLEGOLight

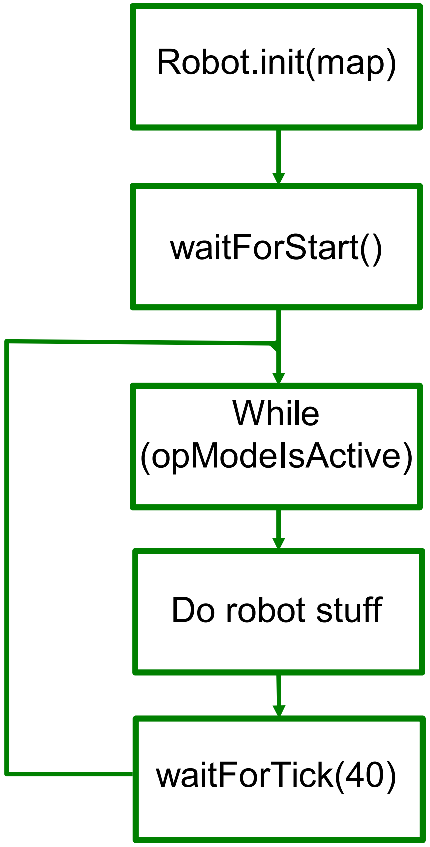
```java
    private ElapsedTime period  = new ElapsedTime();

    /* Constructor */
    public HardwarePushbot(){

    }

    /* Initialize standard Hardware interfaces */
    public void init(HardwareMap ahwMap) {
        // Save reference to Hardware map
        hwMap = ahwMap;

        // Define and Initialize Motors
        leftMotor   = hwMap.dcMotor.get("left_drive");
        rightMotor  = hwMap.dcMotor.get("right_drive");
        armMotor    = hwMap.dcMotor.get("left_arm");
        leftMotor.setDirection(DcMotor.Direction.FORWARD); // Set to REVERSE if using AndyMark motors
        rightMotor.setDirection(DcMotor.Direction.REVERSE);// Set to FORWARD if using AndyMark motors

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

        // Set all motors to run without encoders.
        // May want to use RUN_USING_ENCODERS if encoders are installed.
        leftMotor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        rightMotor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        armMotor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

        // Define and initialize ALL installed servos.
        leftClaw = hwMap.servo.get("left_hand");
        rightClaw = hwMap.servo.get("right_hand");
        leftClaw.setPosition(MID_SERVO);
        rightClaw.setPosition(MID_SERVO);
    }
```

Guide

- **Note the leftMotor is now mapped to a hardware name of "left_drive"**
- **And the rightMotor program name is mapped to the "right_drive" hardware name**
- **It is critical that these "left_drive" and "right_drive" match names in the configuration file on the Robot controller.**

FROG FORCE 503 A LEAP AHEAD

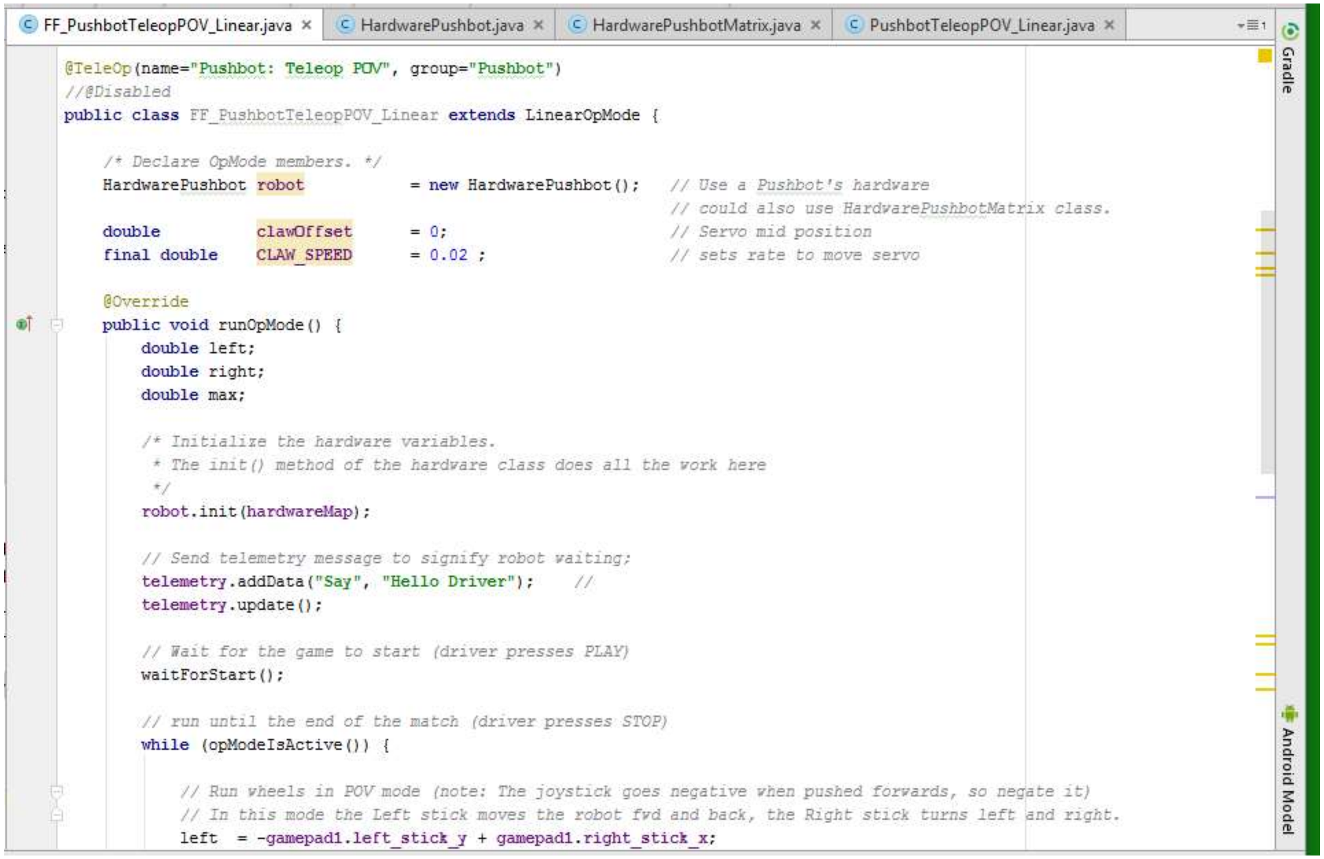
Anatomy of an OpMode – Linear
Overview



Guide

- Your program runs on its own, but it must call certain functions:
  - **Robot.Init** – gets the hardware map
  - **WaitForStart** – waits until the start button is pressed on the controller
  - **While(OpModeisActive)** – creates a loop that runs the commends that follow until the stop button is pressed
  - Do Robot stuff- this is where your code goes
  - **WaitForTick(40)** – interrupts your program for 40 milliseconds to allow the computer to run other processes

Anatomy of an OpMode – Linear
PushbotTeleopPOV_Linear sample code



FF_PushbotTeleopPOV_Linear.java
HardwarePushbot.java
HardwarePushbotMatrix.java
PushbotTeleopPOV_Linear.java
Gradle
Android Model

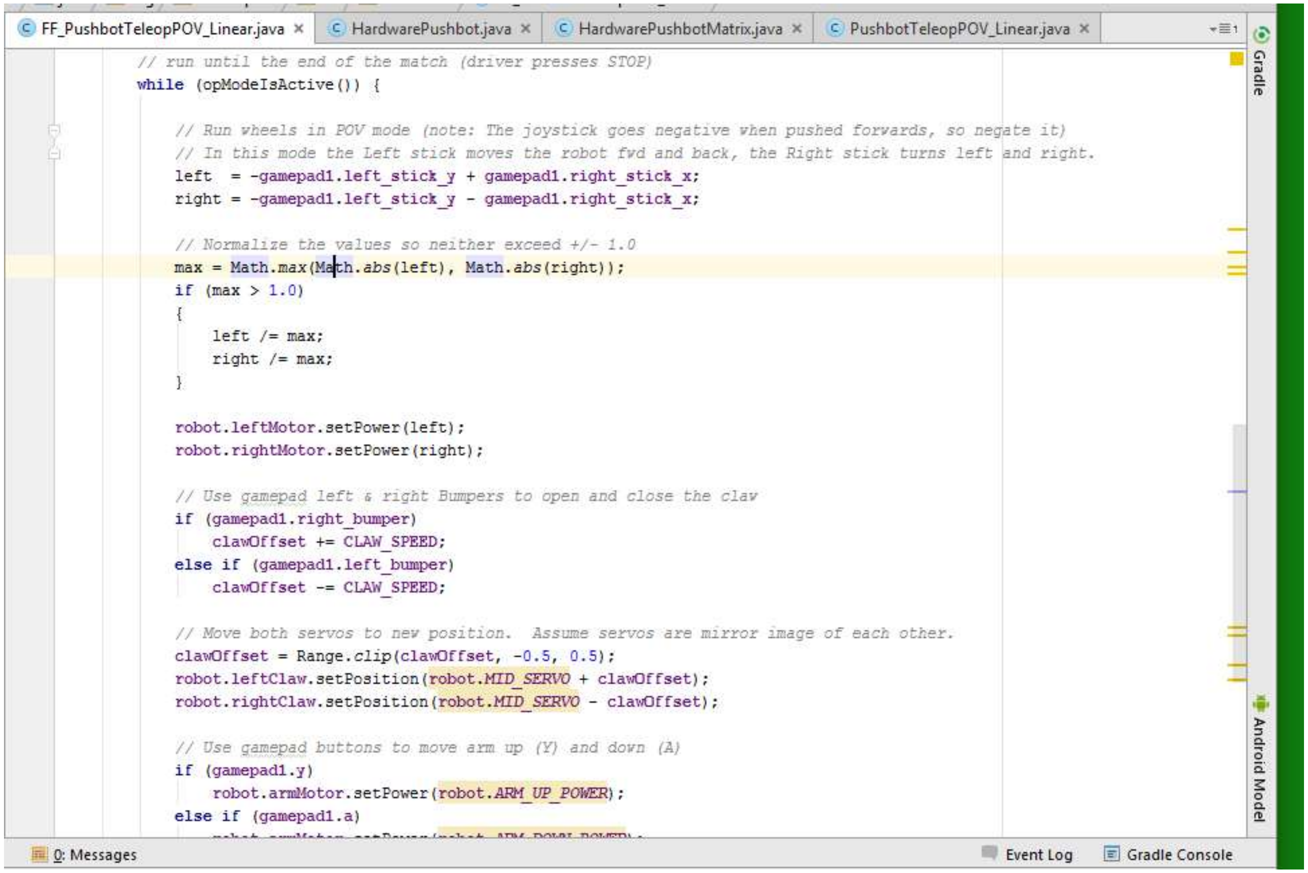
```java
@TeleOp(name="Pushbot: Teleop POV", group="Pushbot")
//@Disabled
public class FF_PushbotTeleopPOV_Linear extends LinearOpMode {

    /* Declare OpMode members. */
    HardwarePushbot robot           = new HardwarePushbot();   // Use a Pushbot's hardware
                                                               // could also use HardwarePushbotMatrix class.
    double          clawOffset      = 0;                       // Servo mid position
    final double    CLAW_SPEED      = 0.02 ;                   // sets rate to move servo

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

        /* Initialize the hardware variables.
         * The init() method of the hardware class does all the work here
         */
        robot.init(hardwareMap);

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

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

        // run until the end of the match (driver presses STOP)
        while (opModeIsActive()) {

            // Run wheels in POV mode (note: The joystick goes negative when pushed forwards, so negate it)
            // In this mode the Left stick moves the robot fwd and back, the Right stick turns left and right.
            left  = -gamepad1.left_stick_y + gamepad1.right_stick_x;
```

Guide

- Extends the Linear OpMode
- Defines robot as a new Hardware Pushbot Object
- Defines some global variables
- Defines a method called runOpMode
- Defines some local variables
- Call HardwarePushBot and runs the Init method to get the hardware map
- Says “Hello” to the driver
- Waits for the start button to be pressed

Anatomy of an OpMode – Linear

PushbotTeleopPOV_Linear sample code


FF_PushbotTeleopPOV_Linear.java
HardwarePushbot.java
HardwarePushbotMatrix.java
PushbotTeleopPOV_Linear.java
Gradle
Android Model
0: Messages
Event Log
Gradle Console

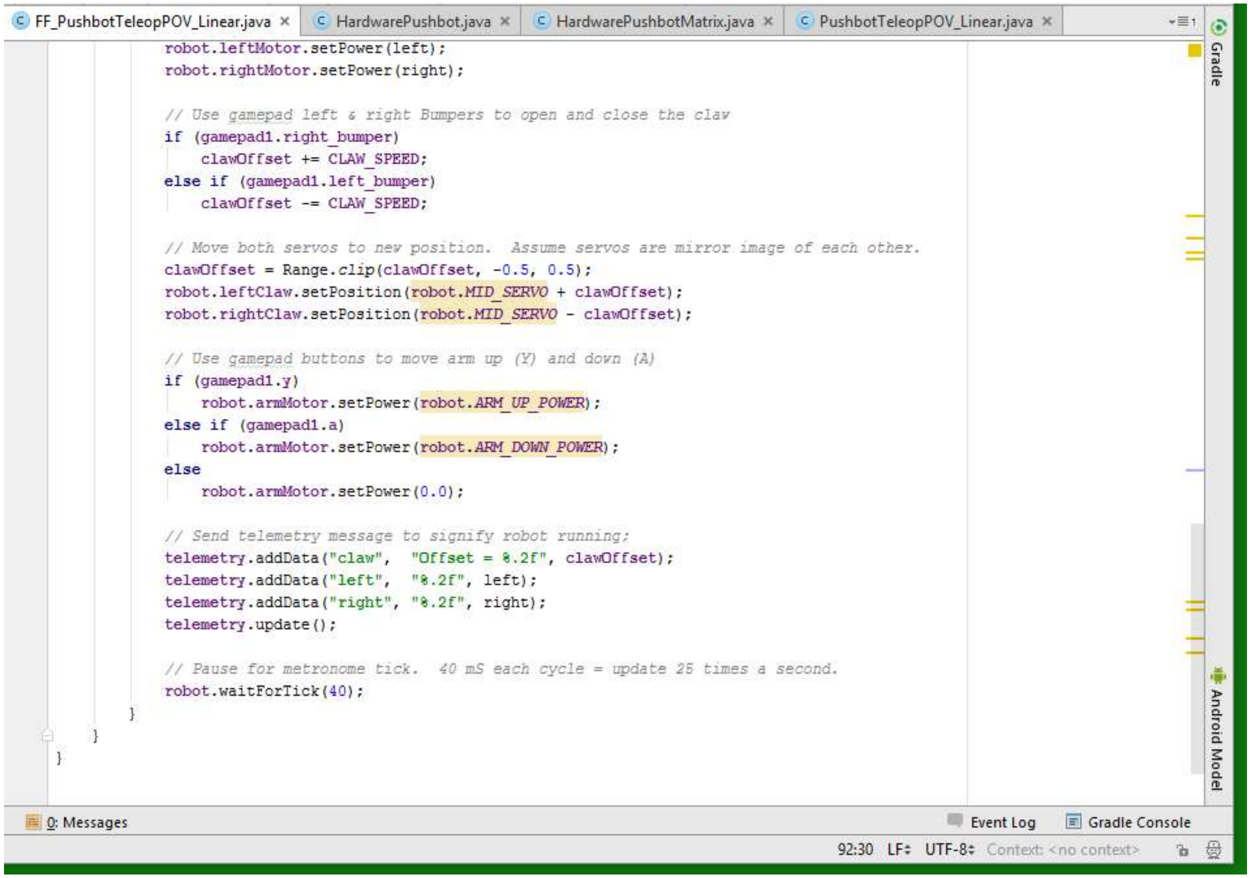
```java
        // run until the end of the match (driver presses STOP)
        while (opModeIsActive()) {

            // Run wheels in POV mode (note: The joystick goes negative when pushed forwards, so negate it)
            // In this mode the Left stick moves the robot fwd and back, the Right stick turns left and right.
            left  = -gamepad1.left_stick_y + gamepad1.right_stick_x;
            right = -gamepad1.left_stick_y - gamepad1.right_stick_x;

            // Normalize the values so neither exceed +/- 1.0
            max = Math.max(Math.abs(left), Math.abs(right));
            if (max > 1.0)
            {
                left /= max;
                right /= max;
            }

            robot.leftMotor.setPower(left);
            robot.rightMotor.setPower(right);

            // Use gamepad left & right Bumpers to open and close the claw
            if (gamepad1.right_bumper)
                clawOffset += CLAW_SPEED;
            else if (gamepad1.left_bumper)
                clawOffset -= CLAW_SPEED;

            // Move both servos to new position.  Assume servos are mirror image of each other.
            clawOffset = Range.clip(clawOffset, -0.5, 0.5);
            robot.leftClaw.setPosition(robot.MID_SERVO + clawOffset);
            robot.rightClaw.setPosition(robot.MID_SERVO - clawOffset);

            // Use gamepad buttons to move arm up (Y) and down (A)
            if (gamepad1.y)
                robot.armMotor.setPower(robot.ARM_UP_POWER);
            else if (gamepad1.a)
                robot.armMotor.setPower(robot.ARM_DOWN_POWER);
```

Guide

- Creates a loop that continues to loop until OpModeIsActive is False
  - ***It is set to false when the Stop button is pressed on the controller***
- Get the joystick settings
- Makes sure the values do not exceed 1.0
- Sets the motor power
- Gets the claw settings from the controller
- Sets the motor speed of the claw to match

Anatomy of an OpMode – Linear

PushbotTeleopPOV_Linear sample code



```java
            robot.leftMotor.setPower(left);
            robot.rightMotor.setPower(right);

            // Use gamepad left & right Bumpers to open and close the claw
            if (gamepad1.right_bumper)
                clawOffset += CLAW_SPEED;
            else if (gamepad1.left_bumper)
                clawOffset -= CLAW_SPEED;

            // Move both servos to new position.  Assume servos are mirror image of each other.
            clawOffset = Range.clip(clawOffset, -0.5, 0.5);
            robot.leftClaw.setPosition(robot.MID_SERVO + clawOffset);
            robot.rightClaw.setPosition(robot.MID_SERVO - clawOffset);

            // Use gamepad buttons to move arm up (Y) and down (A)
            if (gamepad1.y)
                robot.armMotor.setPower(robot.ARM_UP_POWER);
            else if (gamepad1.a)
                robot.armMotor.setPower(robot.ARM_DOWN_POWER);
            else
                robot.armMotor.setPower(0.0);

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

            // Pause for metronome tick.  40 mS each cycle = update 25 times a second.
            robot.waitForTick(40);
        }
    }
}
```

Guide

- Cchek the controller for the arm power
- Set the arm power
- Send telemetry data (display on the driver station) claw and motor values
- Give up control of the processor for 40 milliseconds
- Do it all over again

Congratulations you have completed the basic programming course!!!!!!

Questions?

FROG FORCE 503 A LEAP AHEAD

What we are doing today will transform tomorrow's culture.