FTC New Platform Workshop

presented

By

FLY SET

FIRST® in Texas

TECHNICBOTS

TEXAS WORKFORCE COMMISSION

FTC TEAM #8565

FLY SET

New Platform Software

Samuel Liu / Brandon Wang

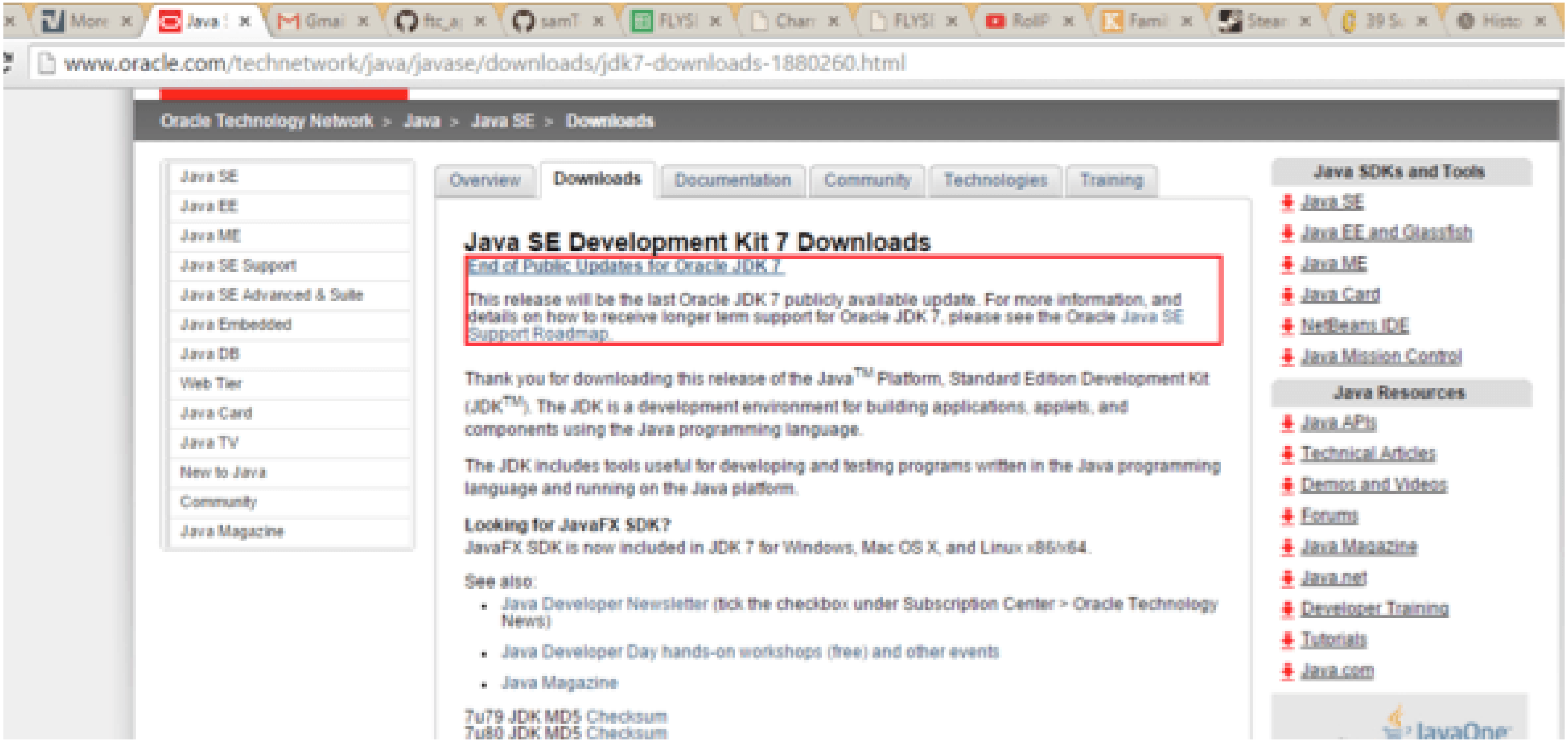
New Platform Software Part I

Samuel Liu

Overview

- Installation
- Android Studio
- Event Driven Programming Model
- Run_To_Position Demonstration
- Tank Drive vs. Steering Drive

Install Java 7 SDK



Java SE Development Kit 7 Downloads

End of Public Updates for Oracle JDK 7

This release will be the last Oracle JDK 7 publicly available update. For more information, and details on how to receive longer term support for Oracle JDK 7, please see the Oracle Java SE Support Roadmap.

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Looking for JavaFX SDK?
JavaFX SDK is now included in JDK 7 for Windows, Mac OS X, and Linux x86/x64.

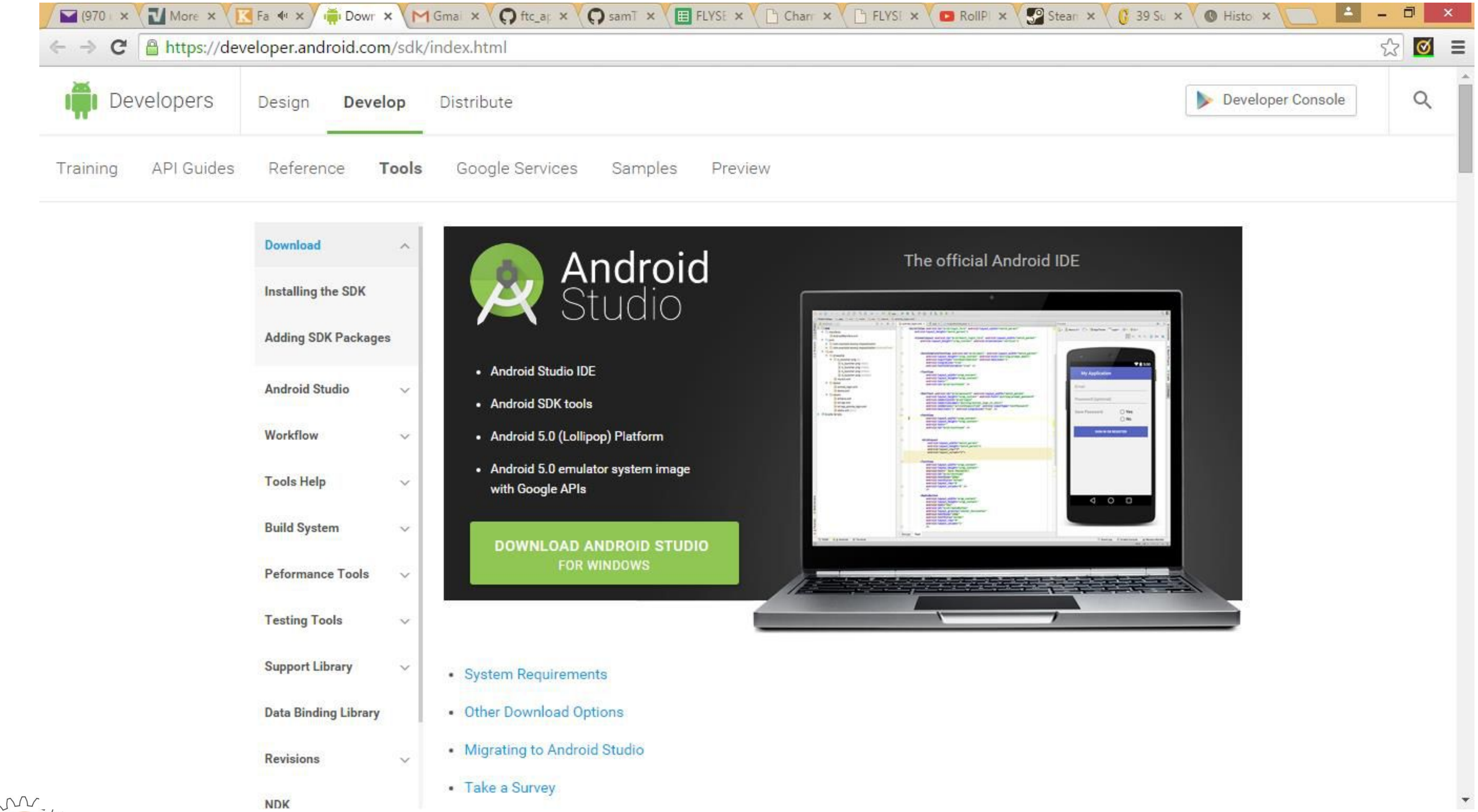
See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

7u79 JDK MD5 Checksum
7u80 JDK MD5 Checksum

Install Android Studio

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

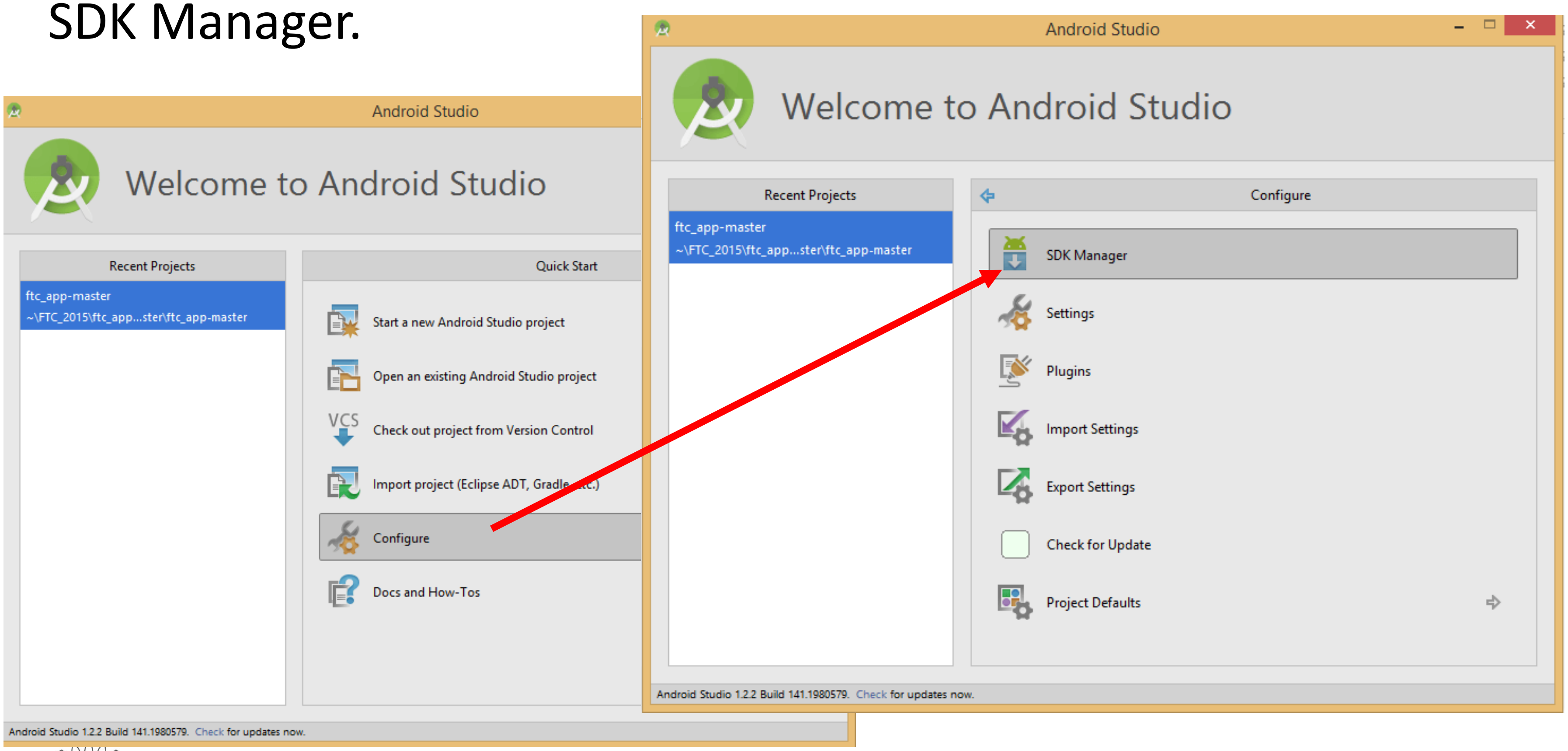


- Android Studio IDE
- Android SDK tools
- Android 5.0 (Lollipop) Platform
- Android 5.0 emulator system image with Google APIs

- System Requirements
- Other Download Options
- Migrating to Android Studio
- Take a Survey

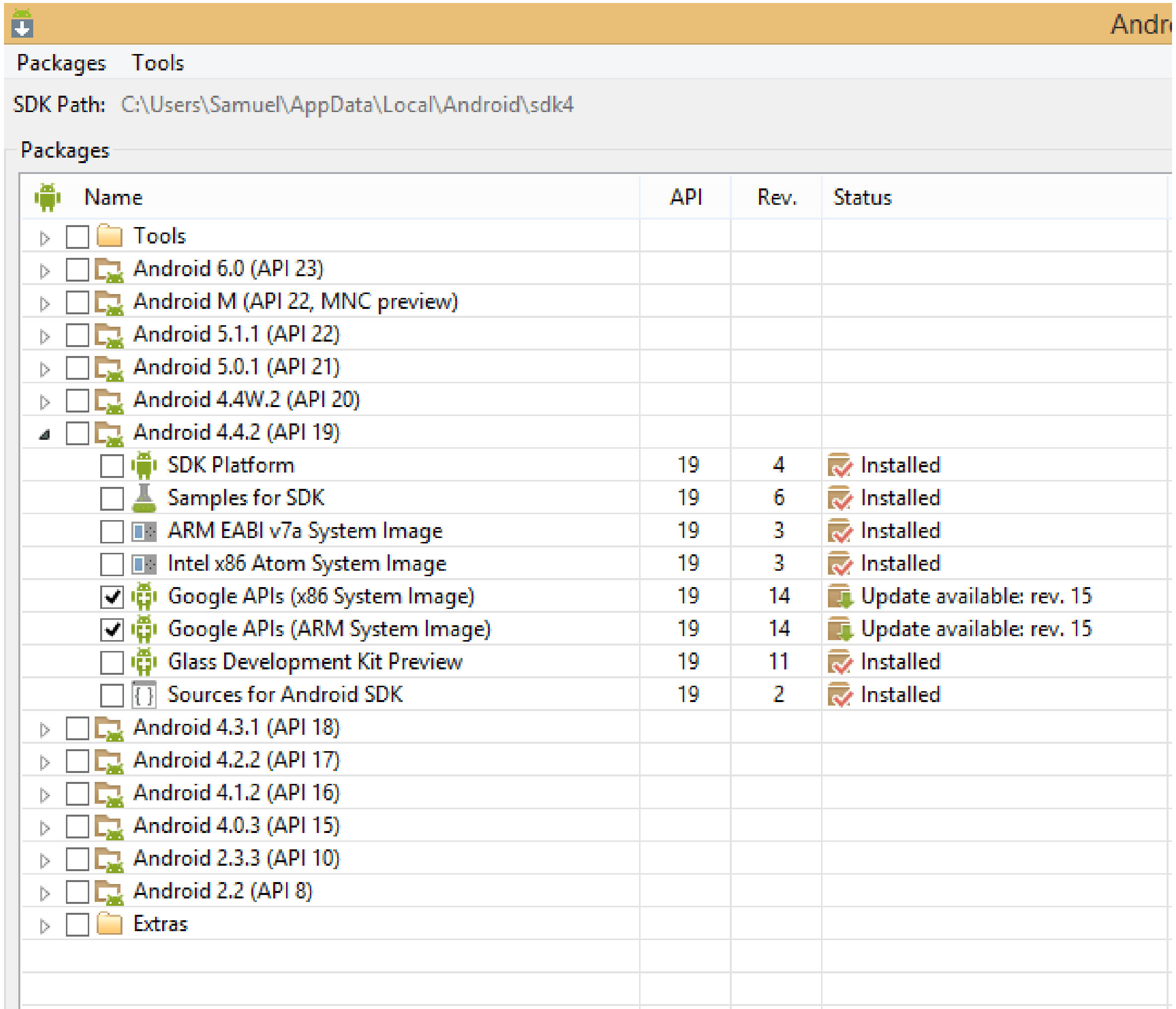
Install Android SDK (API 19)

Run Android Studio(it may have started), choose Configure -> SDK Manager.


Android Studio
Welcome to Android Studio
Recent Projects
ftc_app-master
~\FTC_2015\ftc_app...ster\ftc_app-master
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.2.2 Build 141.1980579. Check for updates now.
Android Studio
Welcome to Android Studio
Recent Projects
ftc_app-master
~\FTC_2015\ftc_app...ster\ftc_app-master
Configure
SDK Manager
Settings
Plugins
Import Settings
Export Settings
Check for Update
Project Defaults
Android Studio 1.2.2 Build 141.1980579. Check for updates now.

Android SDK Manager

ZTE phone runs Android version 4.4.4 which has API 19.

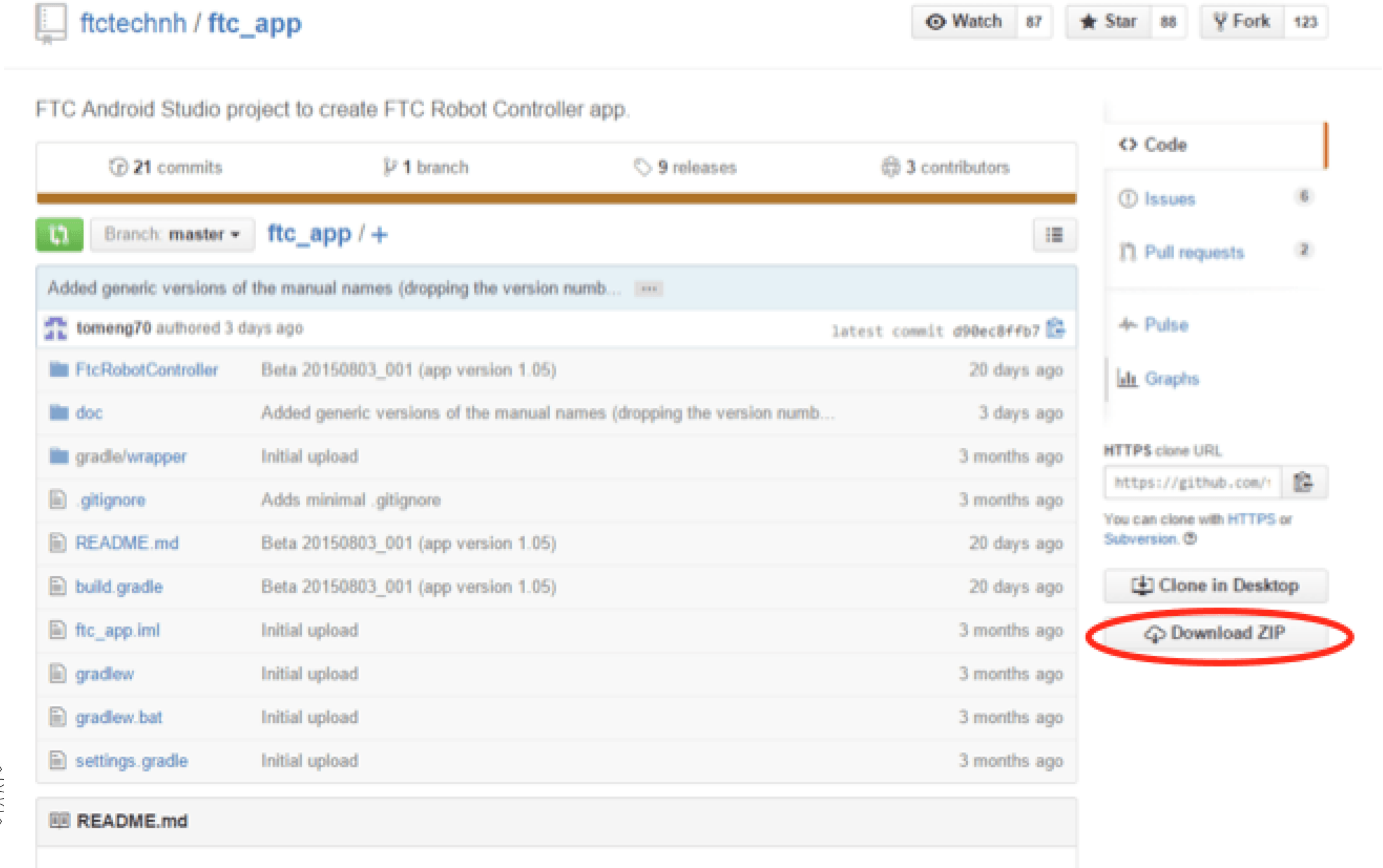


SDK Path: C:\Users\Samuel\AppData\Local\Android\sdk4

Name	API	Rev.	Status
☐ Tools			
☐ Android 6.0 (API 23)			
☐ Android M (API 22, MNC preview)			
☐ Android 5.1.1 (API 22)			
☐ Android 5.0.1 (API 21)			
☐ Android 4.4W.2 (API 20)			
☐ Android 4.4.2 (API 19)			
☐ SDK Platform	19	4	Installed
☐ Samples for SDK	19	6	Installed
☐ ARM EABI v7a System Image	19	3	Installed
☐ Intel x86 Atom System Image	19	3	Installed
☑ Google APIs (x86 System Image)	19	14	Update available: rev. 15
☑ Google APIs (ARM System Image)	19	14	Update available: rev. 15
☐ Glass Development Kit Preview	19	11	Installed
☐ Sources for Android SDK	19	2	Installed
☐ Android 4.3.1 (API 18)			
☐ Android 4.2.2 (API 17)			
☐ Android 4.1.2 (API 16)			
☐ Android 4.0.3 (API 15)			
☐ Android 2.3.3 (API 10)			
☐ Android 2.2 (API 8)			
☐ Extras			

Install FTC SDK

https://github.com/ftctechnh/ftc_app



FTC Android Studio project to create FTC Robot Controller app.

Added generic versions of the manual names (dropping the version numb...

tomeng70 authored 3 days ago — latest commit d90ec8ffb7

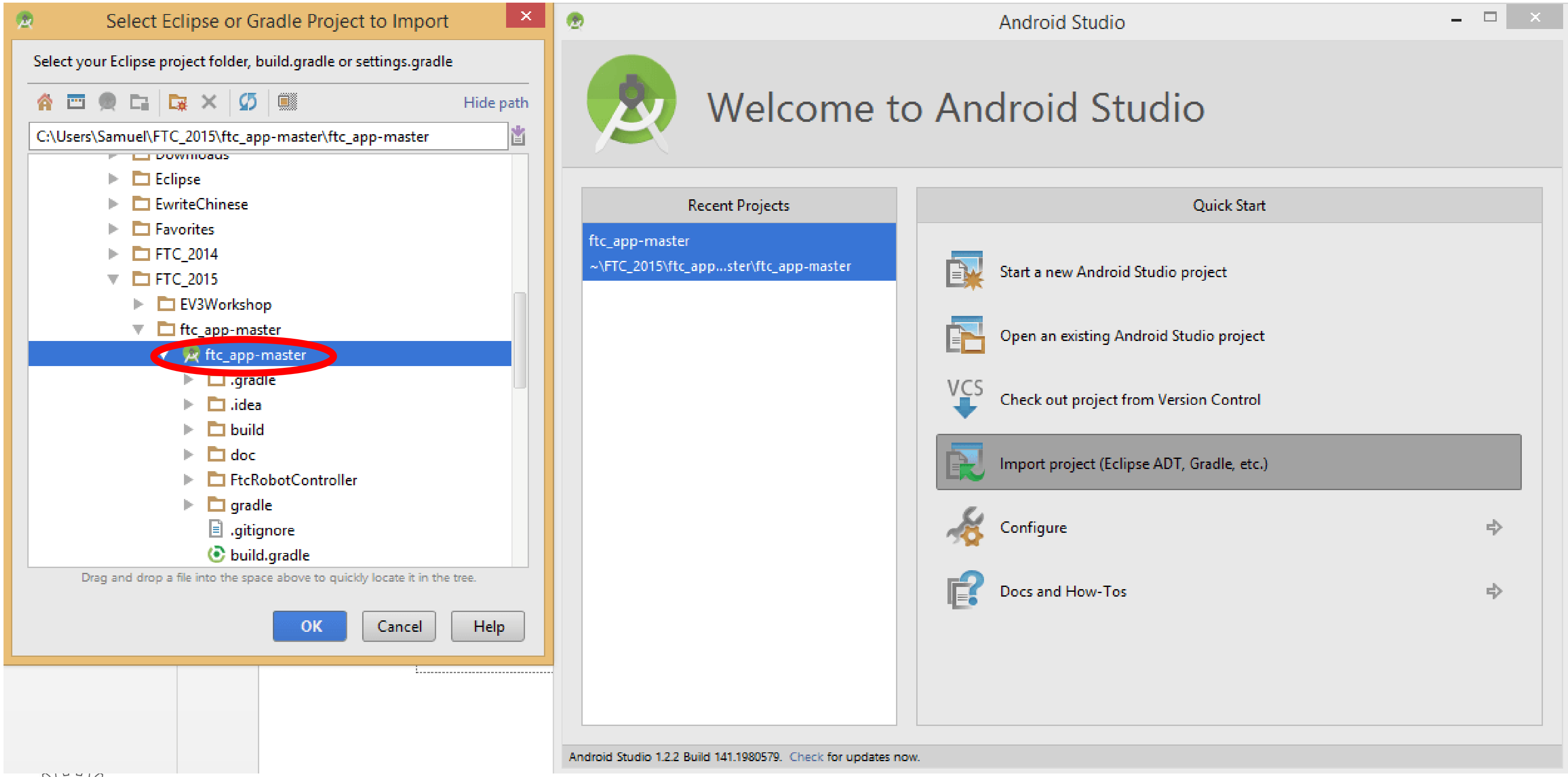
FtcRobotController	Beta 20150803_001 (app version 1.05)	20 days ago
doc	Added generic versions of the manual names (dropping the version numb...	3 days ago
gradle/wrapper	Initial upload	3 months ago
.gitignore	Adds minimal .gitignore	3 months ago
README.md	Beta 20150803_001 (app version 1.05)	20 days ago
build.gradle	Beta 20150803_001 (app version 1.05)	20 days ago
ftc_app.iml	Initial upload	3 months ago
gradlew	Initial upload	3 months ago
gradlew.bat	Initial upload	3 months ago
settings.gradle	Initial upload	3 months ago

You can clone with HTTPS or Subversion.

README.md

Import Project

Restart Android Studio and choose “Import Project”.



Select Eclipse or Gradle Project to Import

Select your Eclipse project folder, build.gradle or settings.gradle

C:\Users\Samuel\FTC_2015\ftc_app-master\ftc_app-master

Drag and drop a file into the space above to quickly locate it in the tree.

OK
Cancel
Help
Android Studio
Welcome to Android Studio
Recent Projects
ftc_app-master
~\FTC_2015\ftc_app...ster\ftc_app-master
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos

Android Studio 1.2.2 Build 141.1980579. Check for updates now.

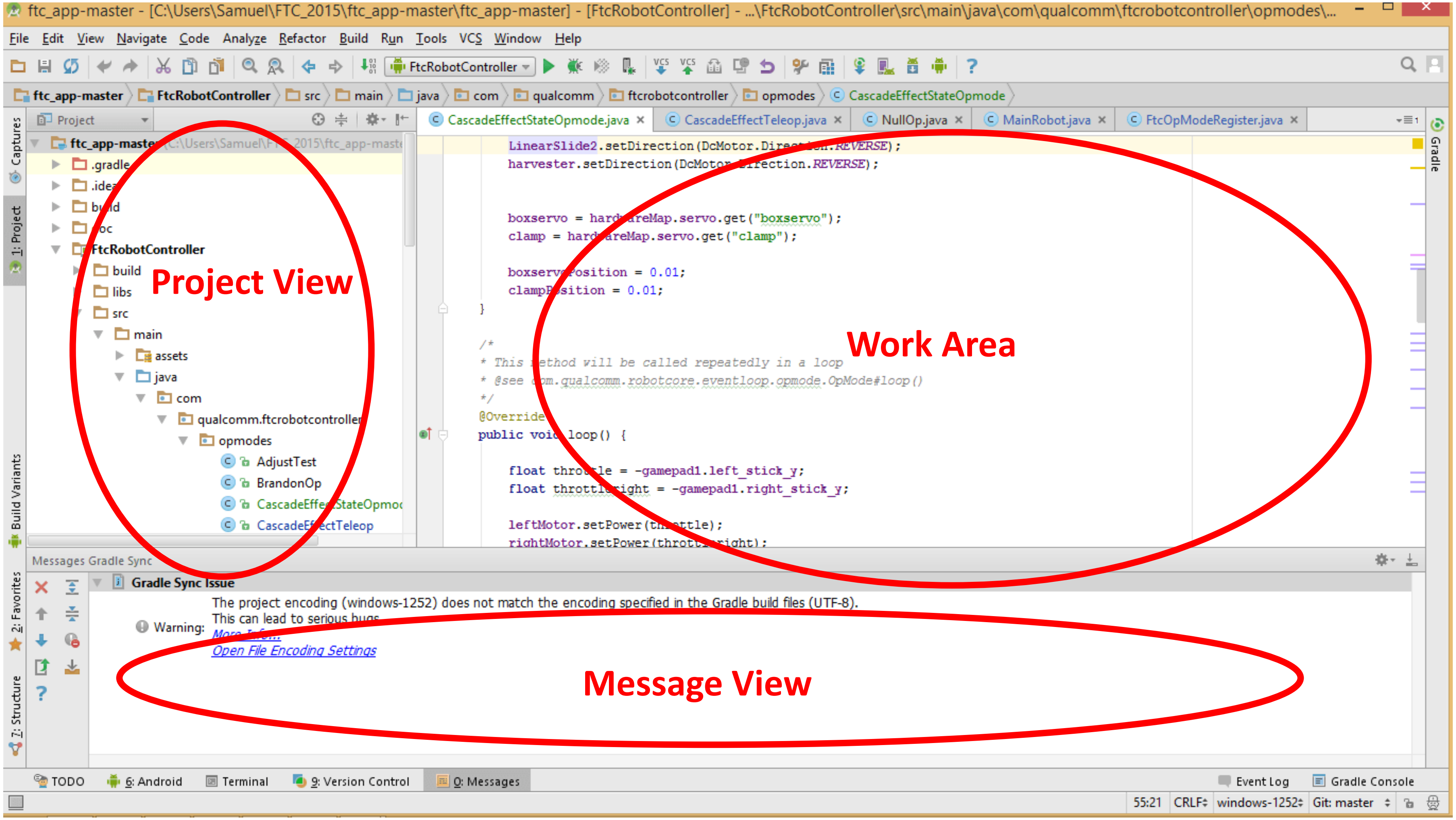
Project Build

Android Studio will automatically start to compile and build.

It takes a few minutes; Android Studio indexes the FTC SDK and won’t allow any more builds until done.

Now you are in Android Studio and ready to create your own robot program!

Android Studio Layout


ftc_app-master - [C:\Users\Samuel\FTC_2015\ftc_app-master\ftc_app-master] - [FtcRobotController] - ...\FtcRobotController\src\main\java\com\qualcomm\ftcrobotcontroller\opmodes\...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
FtcRobotController
ftc_app-master > FtcRobotController > src > main > java > com > qualcomm > ftcrobotcontroller > opmodes > CascadeEffectStateOpmode
Project
ftc_app-master (C:\Users\Samuel\FTC_2015\ftc_app-maste
.gradle
.idea
build
doc
FtcRobotController
build
libs
src
main
assets
java
com
qualcomm.ftcrobotcontroller
opmodes
AdjustTest
BrandonOp
CascadeEffectStateOpmod
CascadeEffectTeleop
CascadeEffectStateOpmode.java
CascadeEffectTeleop.java
NullOp.java
MainRobot.java
FtcOpModeRegister.java
Captures
1: Project
Build Variants
2: Favorites
7: Structure
Gradle
Project View
Work Area
Message View
Messages Gradle Sync
TODO
6: Android
Terminal
9: Version Control
0: Messages
Event Log
Gradle Console
55:21
CRLF
windows-1252
Git: master

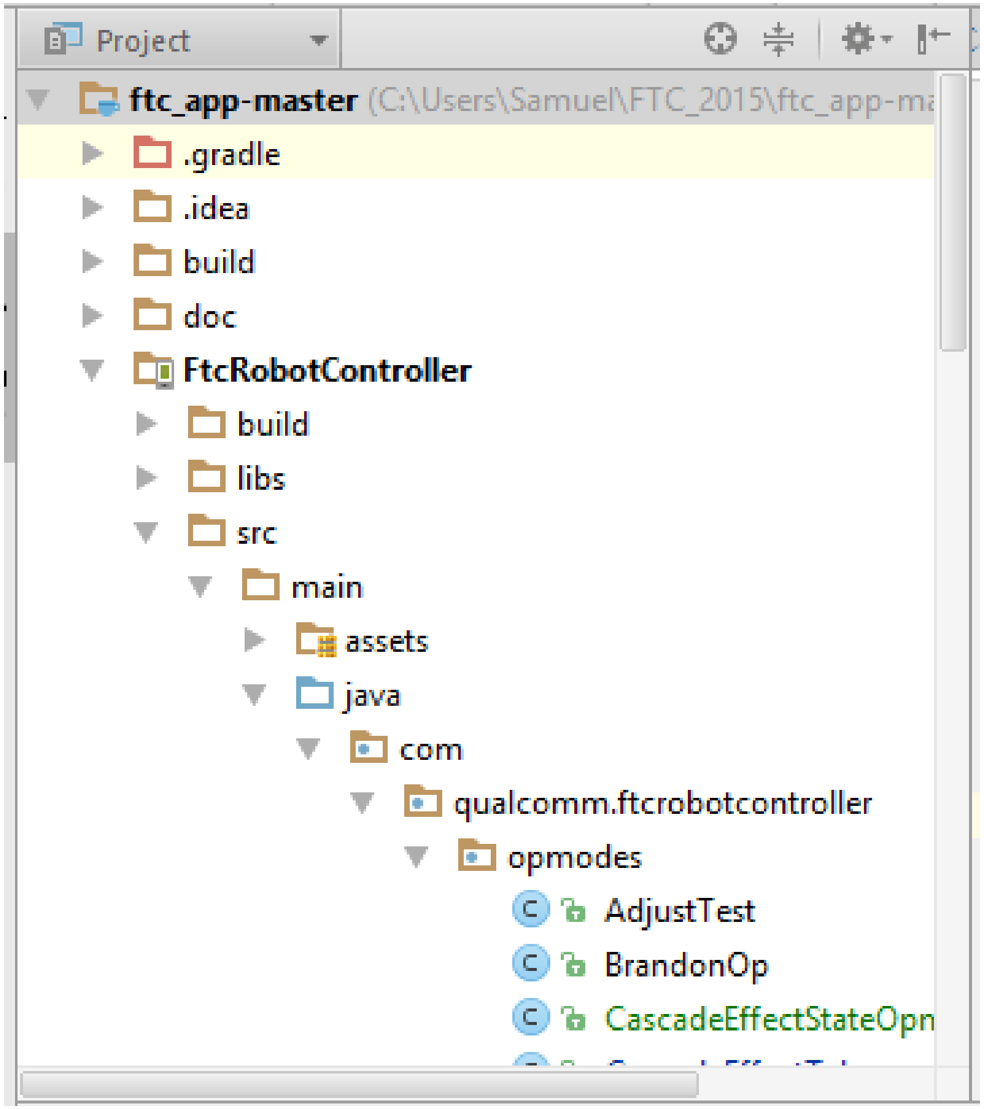
```
        LinearSlide2.setDirection(DcMotor.Direction.REVERSE);
        harvester.setDirection(DcMotor.Direction.REVERSE);


        boxservo = hardwareMap.servo.get("boxservo");
        clamp = hardwareMap.servo.get("clamp");

        boxservoPosition = 0.01;
        clampPosition = 0.01;
    }

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

        float throttle = -gamepad1.left_stick_y;
        float throttleright = -gamepad1.right_stick_y;

        leftMotor.setPower(throttle);
        rightMotor.setPower(throttleright);
```

Gradle Sync Issue

Warning: The project encoding (windows-1252) does not match the encoding specified in the Gradle build files (UTF-8).
This can lead to serious bugs.
More Info...
Open File Encoding Settings

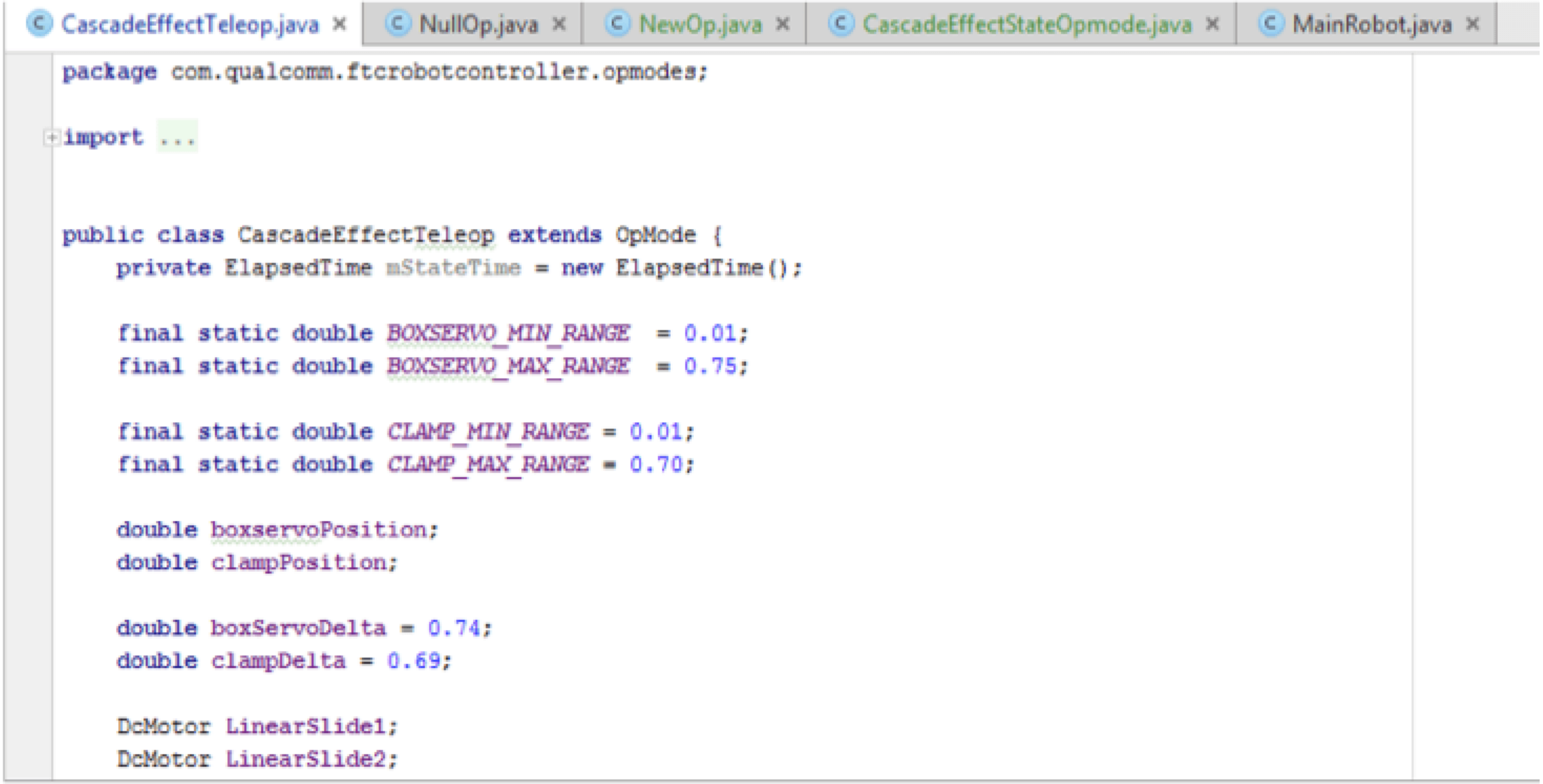
Project View

The project view is where you can navigate through your Android projects and classes.



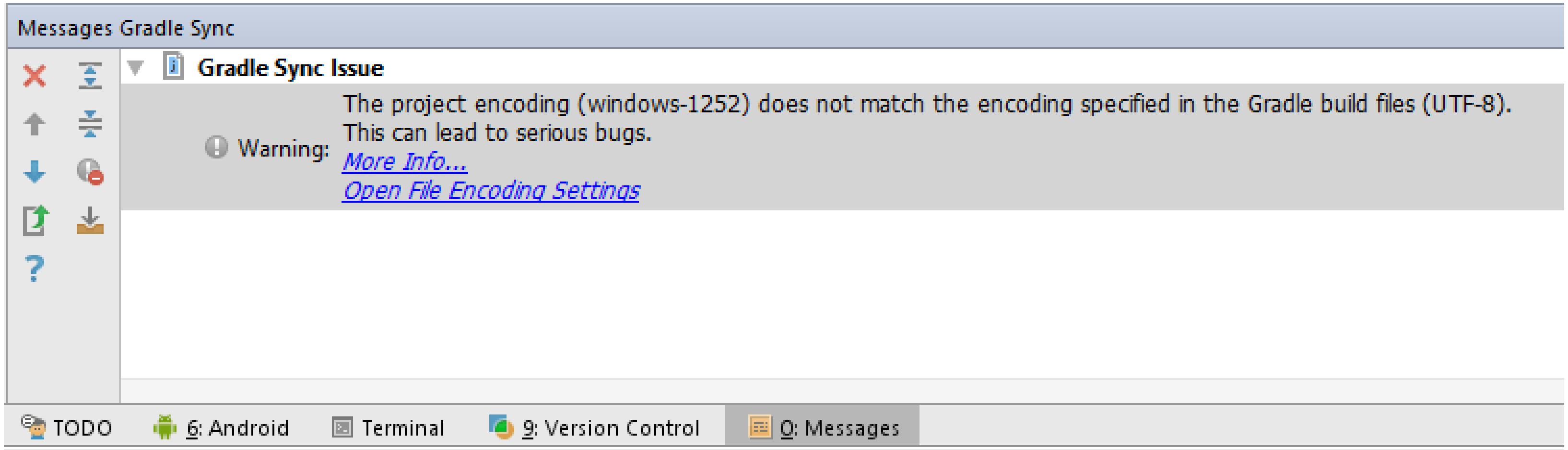
```
ftc_app-master (C:\Users\Samuel\FTC_2015\ftc_app-ma
    .gradle
    .idea
    build
    doc
    FtcRobotController
        build
        libs
        src
            main
                assets
                java
                    com
                        qualcomm.ftcrobotcontroller
                            opmodes
                                AdjustTest
                                BrandonOp
                                CascadeEffectStateOpn
```

Work Area



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

import ...


public class CascadeEffectTeleop extends OpMode {
    private ElapsedTime mStateTime = new ElapsedTime();

    final static double BOXSERVO_MIN_RANGE  = 0.01;
    final static double BOXSERVO_MAX_RANGE  = 0.75;

    final static double CLAMP_MIN_RANGE = 0.01;
    final static double CLAMP_MAX_RANGE = 0.70;

    double boxservoPosition;
    double clampPosition;

    double boxServoDelta = 0.74;
    double clampDelta = 0.69;

    DcMotor LinearSlide1;
    DcMotor LinearSlide2;
```

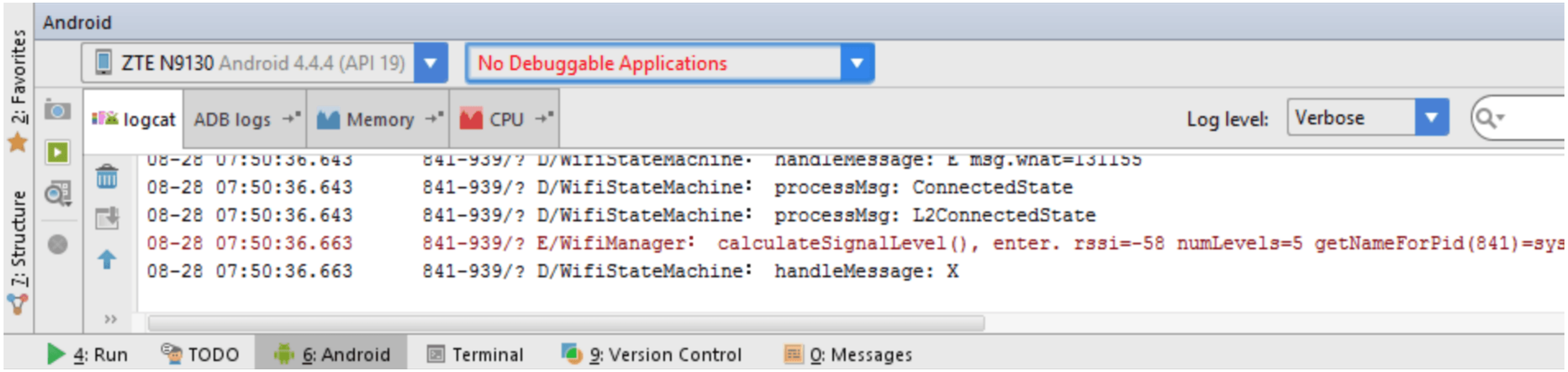
Message View



Messages Gradle Sync

Gradle Sync Issue

Warning: The project encoding (windows-1252) does not match the encoding specified in the Gradle build files (UTF-8).
This can lead to serious bugs.
More Info...
Open File Encoding Settings



Android

ZTE N9130 Android 4.4.4 (API 19)

No Debuggable Applications

Log level: Verbose

```
08-28 07:50:36.643      841-939/? D/WifiStateMachine:  handleMessage: E msg.what=131155
08-28 07:50:36.643      841-939/? D/WifiStateMachine:  processMsg: ConnectedState
08-28 07:50:36.643      841-939/? D/WifiStateMachine:  processMsg: L2ConnectedState
08-28 07:50:36.663      841-939/? E/WifiManager:  calculateSignalLevel(), enter. rssi=-58 numLevels=5 getNameForPid(841)=sys
08-28 07:50:36.663      841-939/? D/WifiStateMachine:  handleMessage: X
```

Event Driven Programming

- A programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads

- In the FTC SDK context, the event is the looping event generated from framework

- Different from RobotC’s linear programming model where code is executed sequentially from beginning to end

OpMode

OpMode: Different modes a user can run the robot in (e.g., Autonomous, Teleop) and written in one Java Class

Your OpMode Java Class must extend the superclass in FTC SDK:

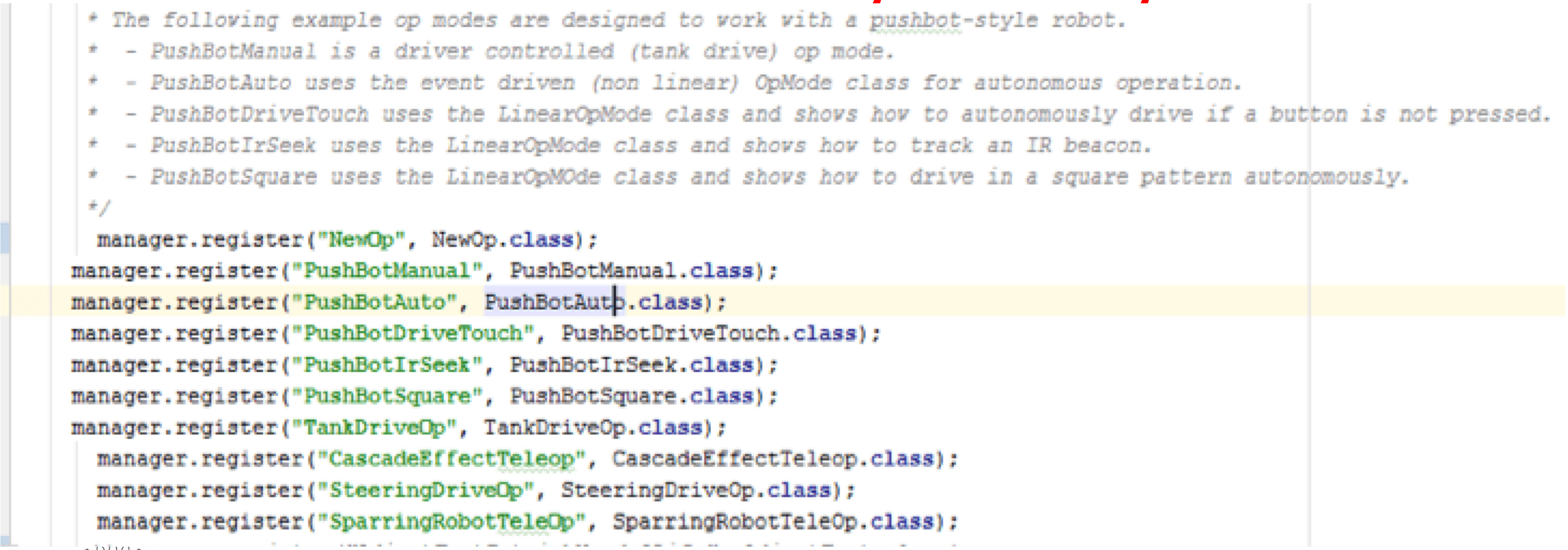
com.qualcomm.robotcore.eventloop.opmode.OpMode

```
public class TankDriveOp extends OpMode {
```

Registering an Op Mode

For the Driver Station App to recognize your op mode, you need to register it in FtcOpModeRegister Java Class.

BE CAREFUL – The opmode list message between driver station and robot controller can only hold 256 bytes.



```
 * The following example op modes are designed to work with a pushbot-style robot.
 *  - PushBotManual is a driver controlled (tank drive) op mode.
 *  - PushBotAuto uses the event driven (non linear) OpMode class for autonomous operation.
 *  - PushBotDriveTouch uses the LinearOpMode class and shows how to autonomously drive if a button is not pressed.
 *  - PushBotIrSeek uses the LinearOpMode class and shows how to track an IR beacon.
 *  - PushBotSquare uses the LinearOpMOde class and shows how to drive in a square pattern autonomously.
 */
  manager.register("NewOp", NewOp.class);
manager.register("PushBotManual", PushBotManual.class);
manager.register("PushBotAuto", PushBotAuto.class);
manager.register("PushBotDriveTouch", PushBotDriveTouch.class);
manager.register("PushBotIrSeek", PushBotIrSeek.class);
manager.register("PushBotSquare", PushBotSquare.class);
manager.register("TankDriveOp", TankDriveOp.class);
  manager.register("CascadeEffectTeleop", CascadeEffectTeleop.class);
  manager.register("SteeringDriveOp", SteeringDriveOp.class);
  manager.register("SparringRobotTeleOp", SparringRobotTeleOp.class);
```

The Life Cycle of an Op Mode

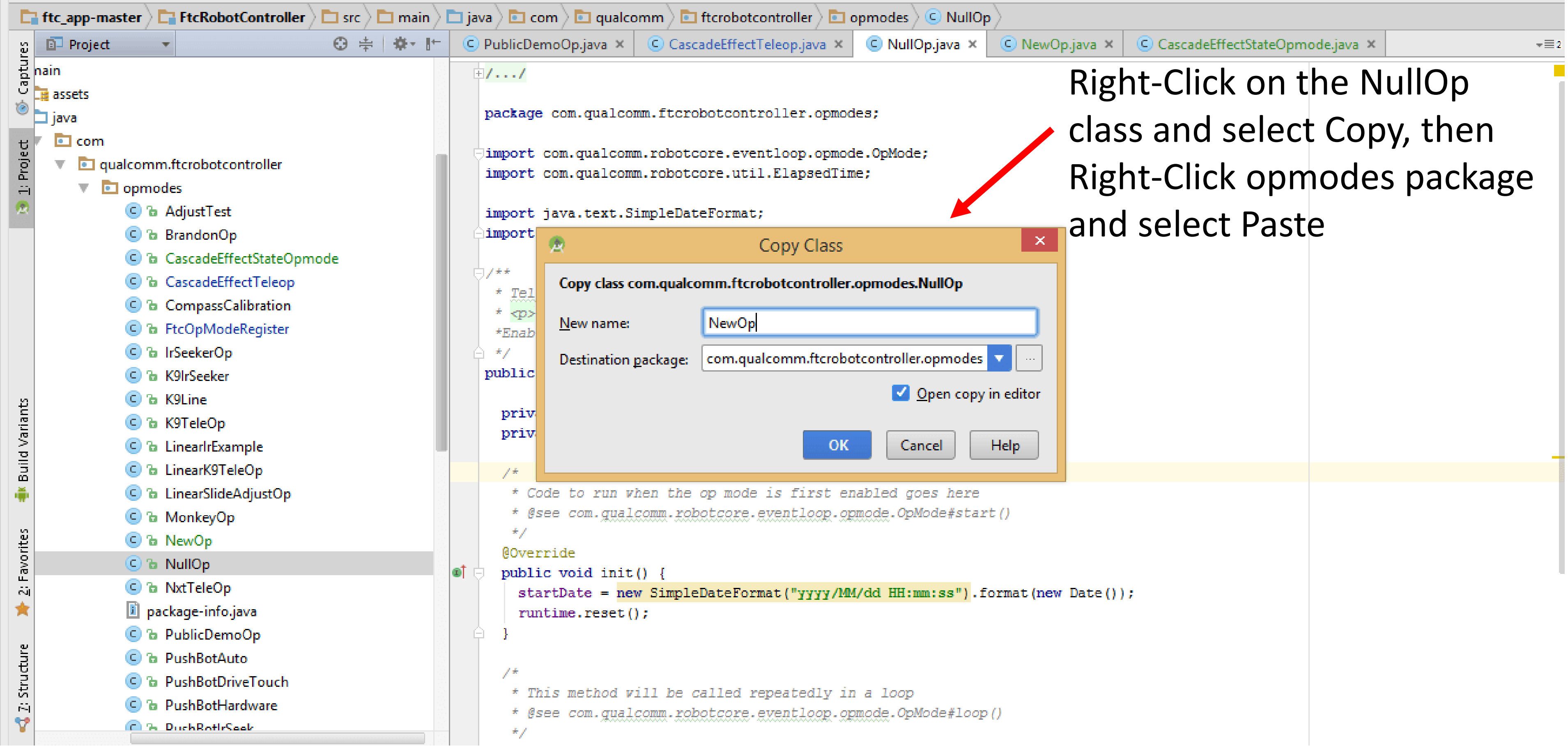
init() – Used to perform initialization tasks, can only be performed once. Triggered when “arm” button pressed on the driver station.

start() – The difference between this and init() is that this mode is triggered when the op mode starts. You can also run initialization tasks, they are executed right before the loop.

loop() – This part of the code is regularly executed, every 10 to 20 milliseconds, this makes up the main body of the op mode.

stop() – When the program is stopped, the code in this block is executed. This is used for cleanup after running through an op mode.

Creating Your Own OpMode



```
/.../

package com.qualcomm.ftcrobotcontroller.opmodes;

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.util.ElapsedTime;

import java.text.SimpleDateFormat;
import

/**
 * Tel
 * <p>
 *Enab
 */
public

  priv
  priv

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

Copy Class

Copy class com.qualcomm.ftcrobotcontroller.opmodes.NullOp

New name: NewOp

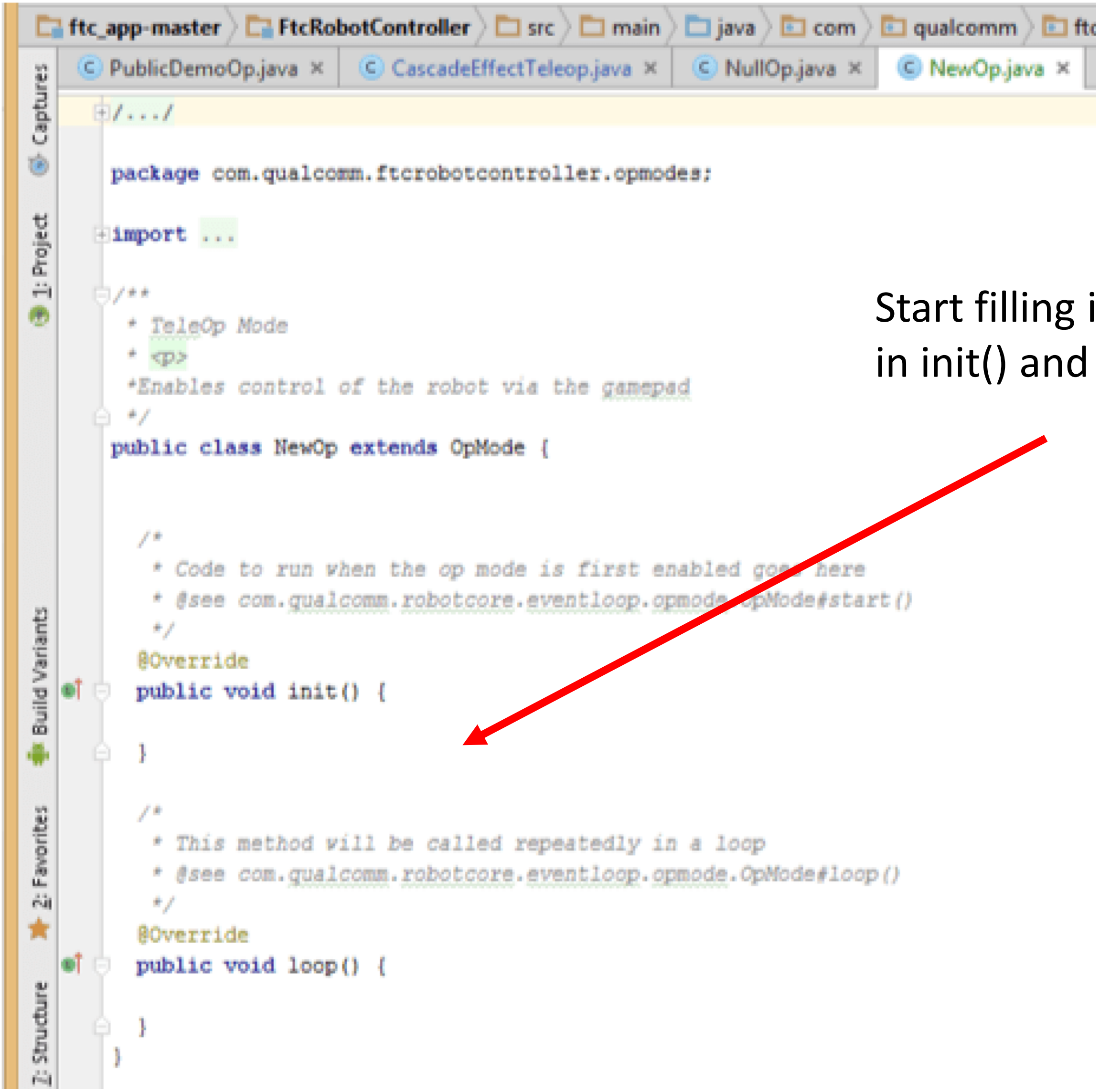
Destination package: com.qualcomm.ftcrobotcontroller.opmodes

☑ Open copy in editor

OK Cancel Help

Right-Click on the NullOp class and select Copy, then Right-Click opmodes package and select Paste

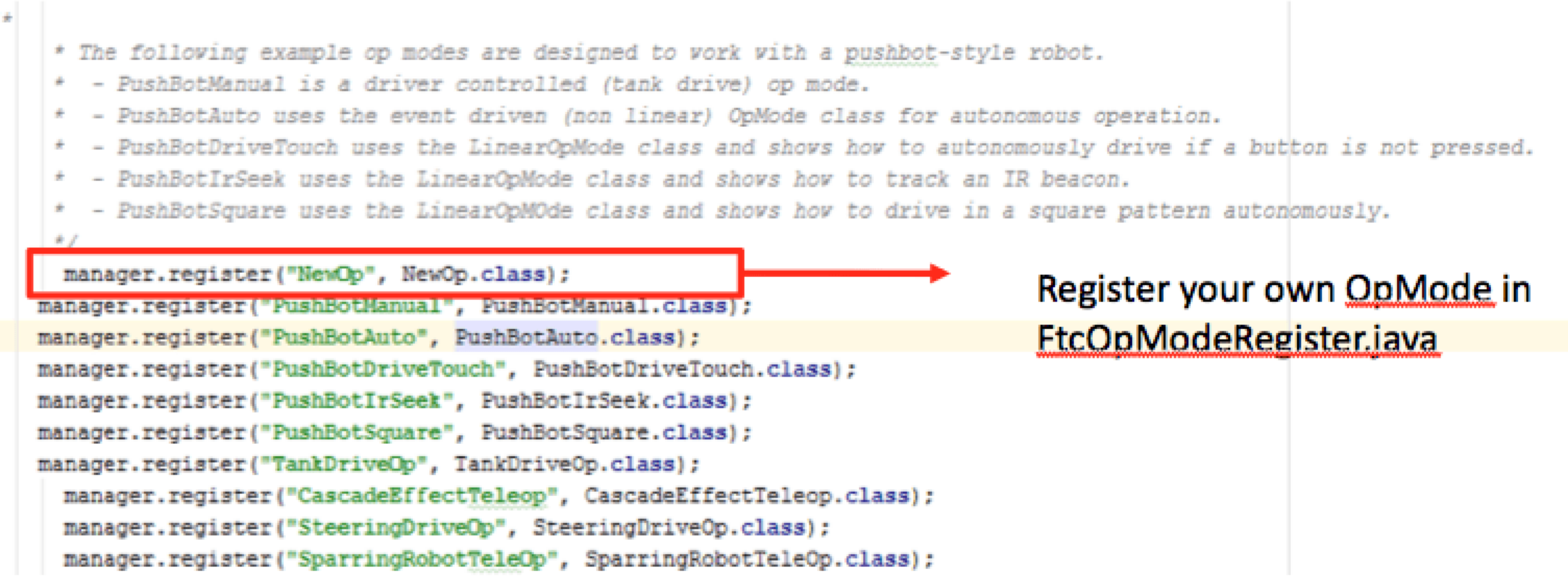
Writing logic for Your Own Op Mode



```
/.../

package com.qualcomm.ftcrobotcontroller.opmodes;

import ...

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


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

  }

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

  }
}
```

Start filling in your own code in init() and loop() methods

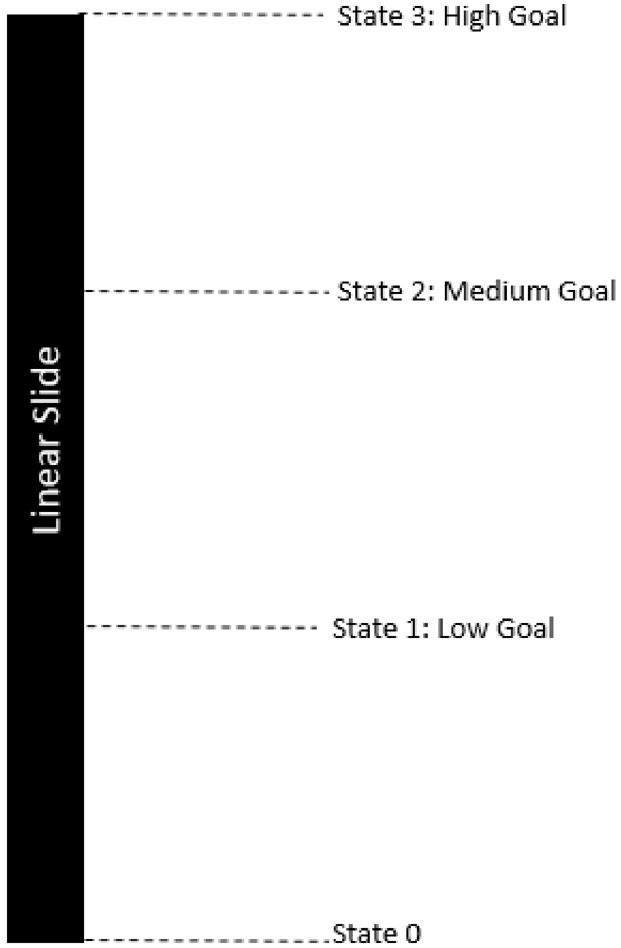
Register Your Own Op Mode



```
*
  * The following example op modes are designed to work with a pushbot-style robot.
  *   - PushBotManual is a driver controlled (tank drive) op mode.
  *   - PushBotAuto uses the event driven (non linear) OpMode class for autonomous operation.
  *   - PushBotDriveTouch uses the LinearOpMode class and shows how to autonomously drive if a button is not pressed.
  *   - PushBotIrSeek uses the LinearOpMode class and shows how to track an IR beacon.
  *   - PushBotSquare uses the LinearOpMOde class and shows how to drive in a square pattern autonomously.
  */
   manager.register("NewOp", NewOp.class);
 manager.register("PushBotManual", PushBotManual.class);
 manager.register("PushBotAuto", PushBotAuto.class);
 manager.register("PushBotDriveTouch", PushBotDriveTouch.class);
 manager.register("PushBotIrSeek", PushBotIrSeek.class);
 manager.register("PushBotSquare", PushBotSquare.class);
 manager.register("TankDriveOp", TankDriveOp.class);
   manager.register("CascadeEffectTeleop", CascadeEffectTeleop.class);
   manager.register("SteeringDriveOp", SteeringDriveOp.class);
   manager.register("SparringRobotTeleOp", SparringRobotTeleOp.class);
```

Register your own OpMode in FtcOpModeRegister.java

State Machine Programming



Linear Slide

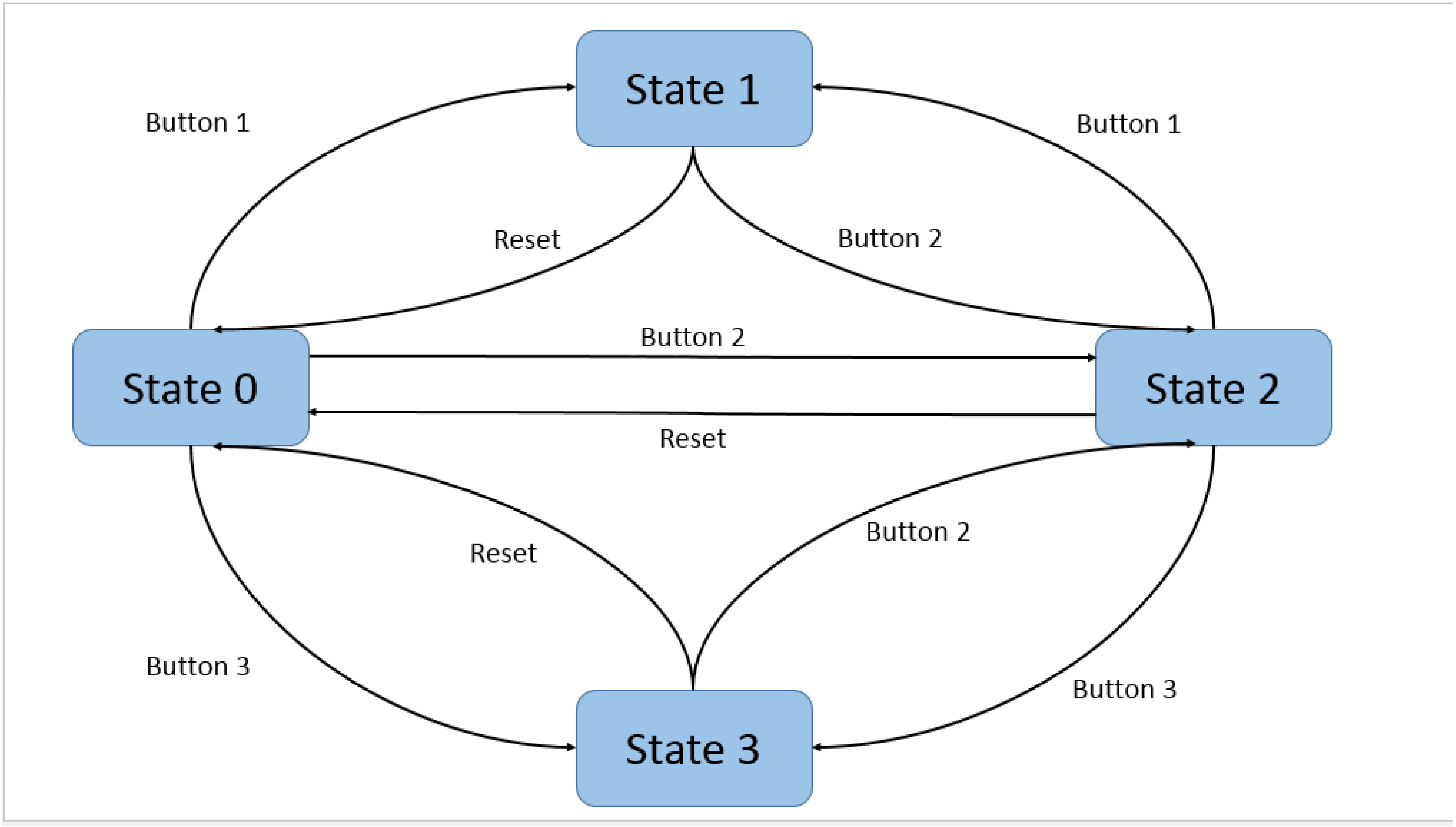
State 3: High Goal

State 2: Medium Goal

State 1: Low Goal

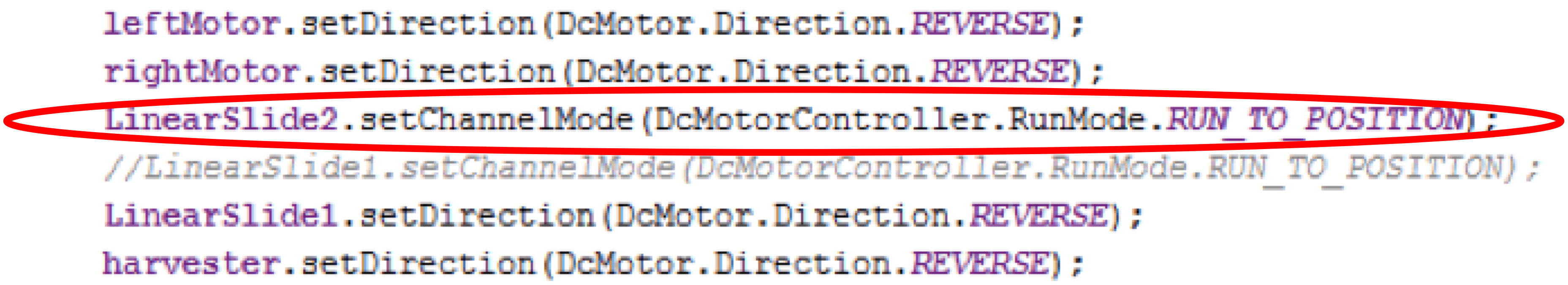
State 0

State Machine Programming



Run_To_Position

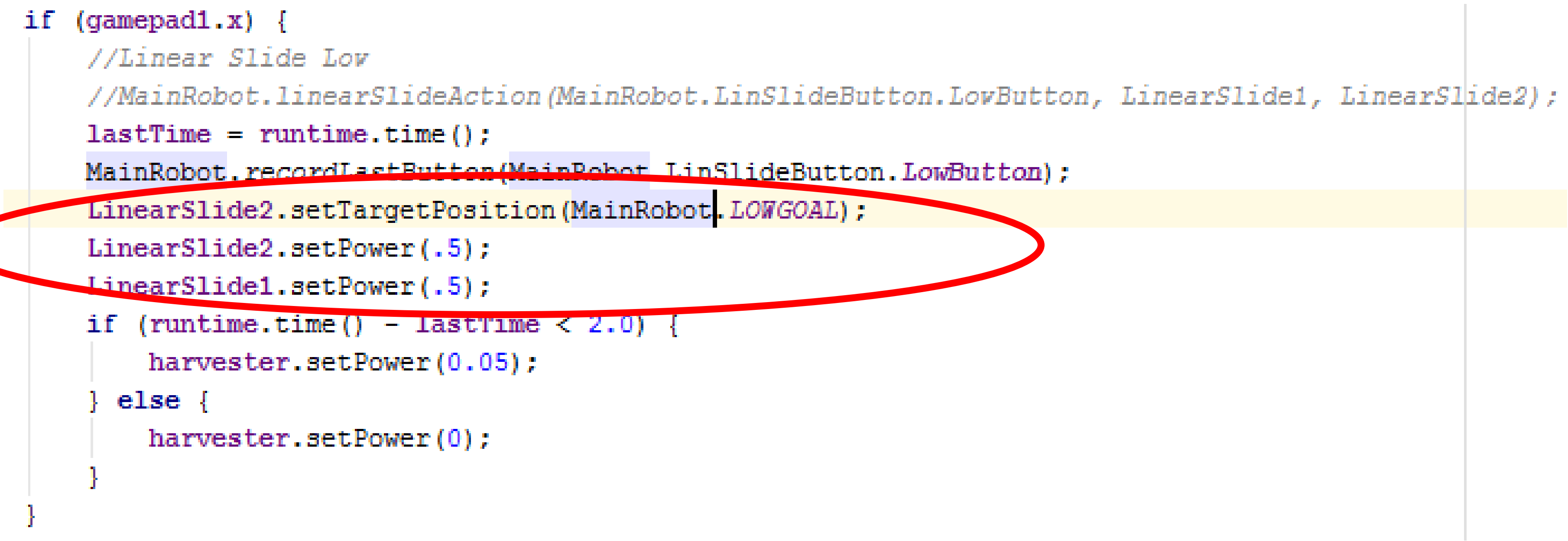
- Set the motor with Run_To_Position mode in init() method



```
leftMotor.setDirection(DcMotor.Direction.REVERSE);
rightMotor.setDirection(DcMotor.Direction.REVERSE);
LinearSlide2.setChannelMode(DcMotorController.RunMode.RUN_TO_POSITION);
//LinearSlide1.setChannelMode(DcMotorController.RunMode.RUN_TO_POSITION);
LinearSlide1.setDirection(DcMotor.Direction.REVERSE);
harvester.setDirection(DcMotor.Direction.REVERSE);
```

Run_To_Position

- Set the target position for the motor in loop() method



```java
if (gamepad1.x) {
    //Linear Slide Low
    //MainRobot.linearSlideAction(MainRobot.LinSlideButton.LowButton, LinearSlide1, LinearSlide2);
    lastTime = runtime.time();
    MainRobot.recordLastButton(MainRobot.LinSlideButton.LowButton);
    LinearSlide2.setTargetPosition(MainRobot.LOWGOAL);
    LinearSlide2.setPower(.5);
    LinearSlide1.setPower(.5);
    if (runtime.time() - lastTime < 2.0) {
        harvester.setPower(0.05);
    } else {
        harvester.setPower(0);
    }
}
```

Run_To_Position Demo

Tank Drive vs. Steering Drive

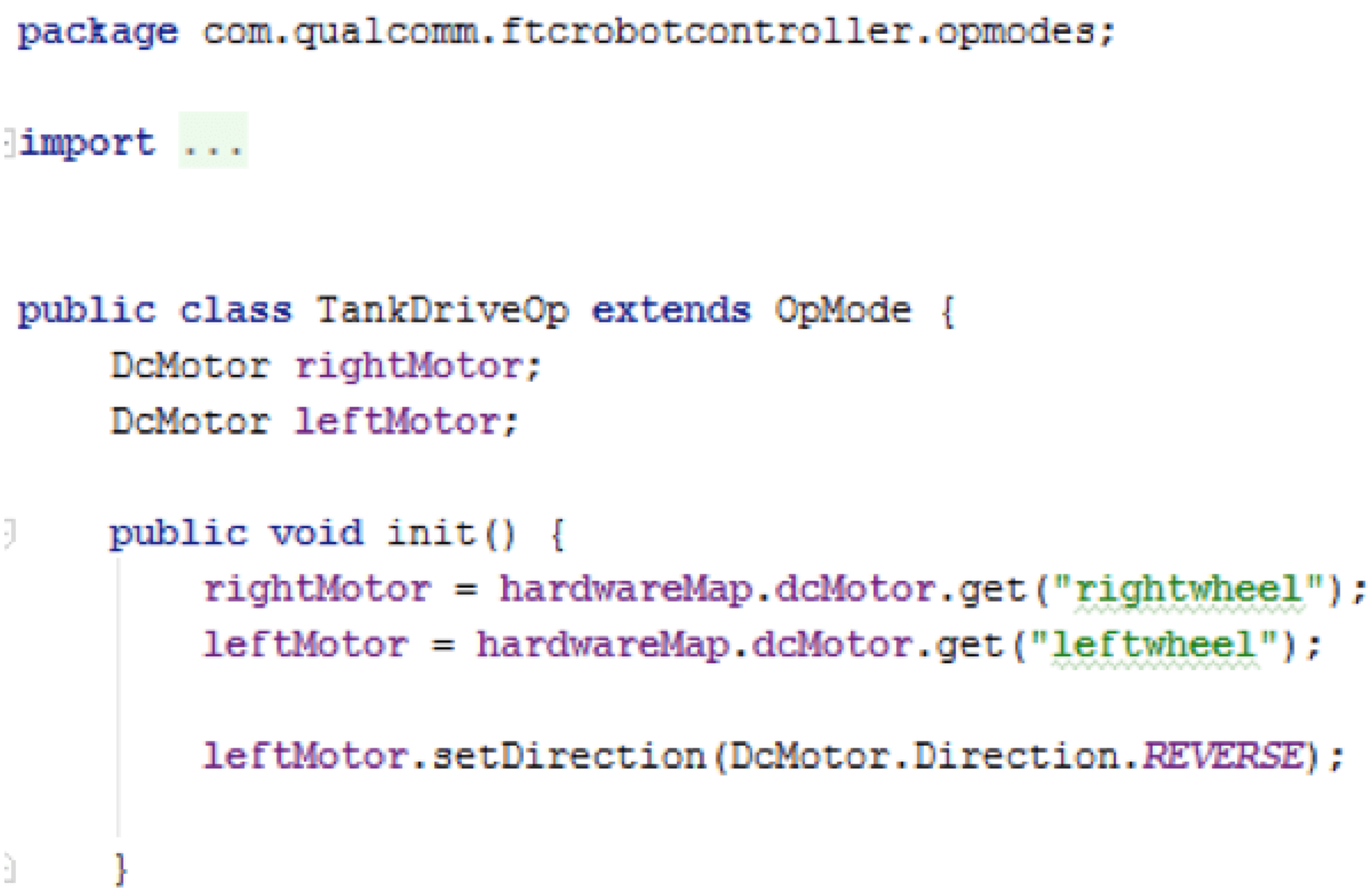
Tank Drive:

- Allows for manual control over both wheels of the robot
- Uses both joysticks

Steering Drive:

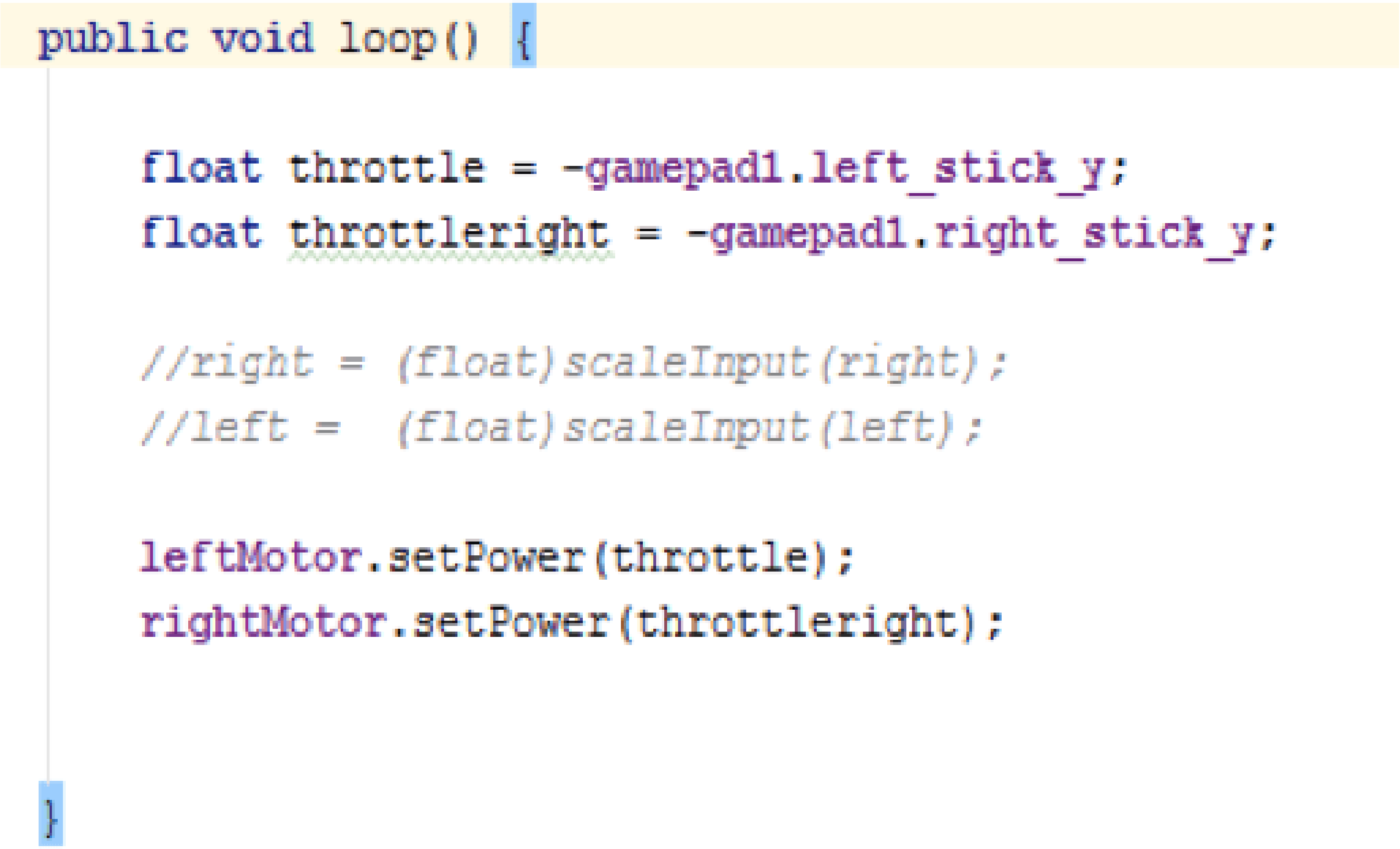
- Manual control over the whole robot, not individual wheel
- Uses only 1 joystick

Common Initialization



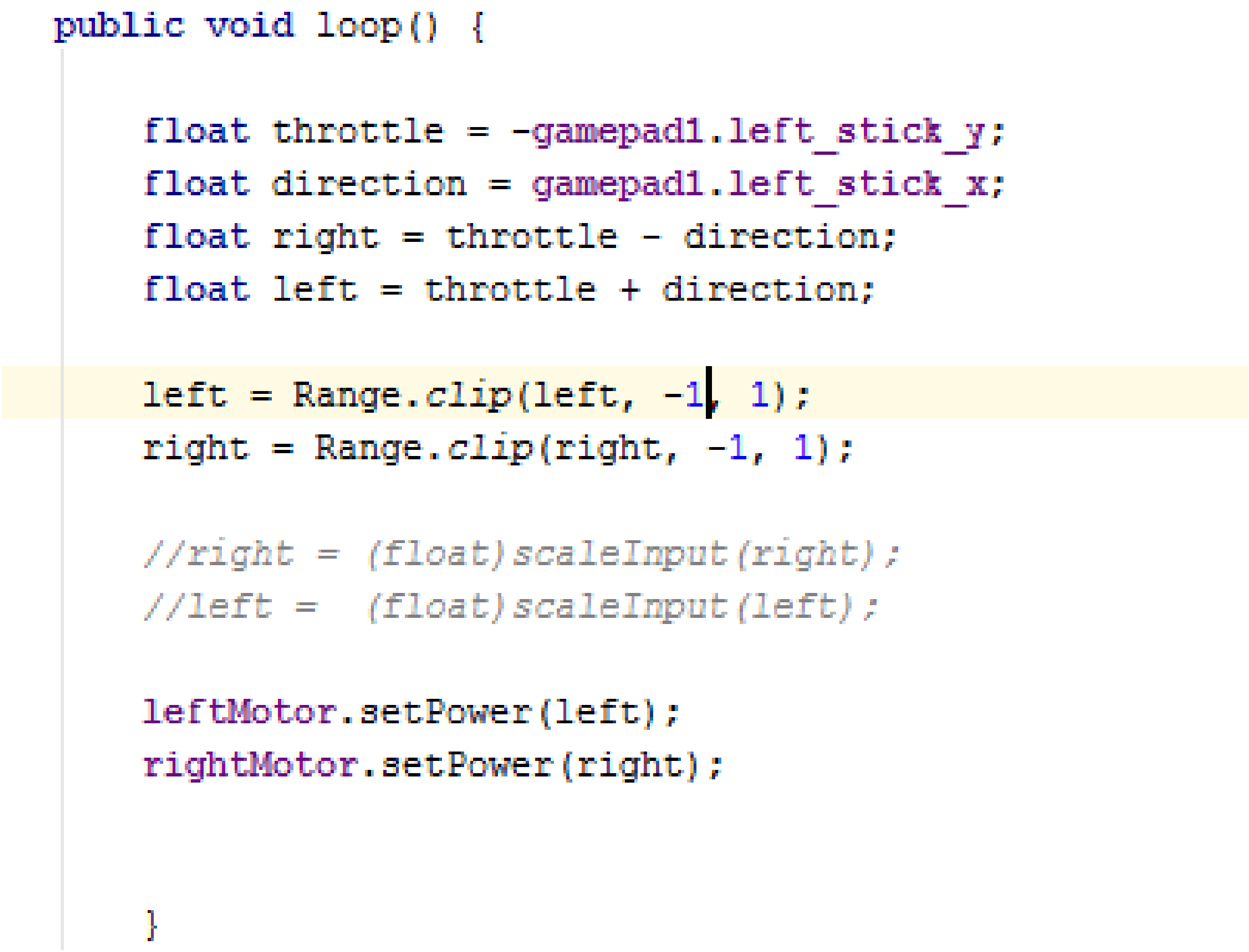
```
package com.qualcomm.ftcrobotcontroller.opmodes;

import ...


public class TankDriveOp extends OpMode {
    DcMotor rightMotor;
    DcMotor leftMotor;

    public void init() {
        rightMotor = hardwareMap.dcMotor.get("rightwheel");
        leftMotor = hardwareMap.dcMotor.get("leftwheel");

        leftMotor.setDirection(DcMotor.Direction.REVERSE);

    }
```

Tank Drive Loop Method



```
public void loop() {

    float throttle = -gamepad1.left_stick_y;
    float throttleright = -gamepad1.right_stick_y;

    //right = (float)scaleInput(right);
    //left =  (float)scaleInput(left);

    leftMotor.setPower(throttle);
    rightMotor.setPower(throttleright);



}
```

Steering Drive Loop Method



```
public void loop() {

    float throttle = -gamepad1.left_stick_y;
    float direction = gamepad1.left_stick_x;
    float right = throttle - direction;
    float left = throttle + direction;

    left = Range.clip(left, -1, 1);
    right = Range.clip(right, -1, 1);

    //right = (float)scaleInput(right);
    //left =  (float)scaleInput(left);

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


    }
```

New Platform Software
Part II

Brandon Wang

Overview

- Linear OpMode
- Sensor API
- Code Structure
- GitHub Basics

Linear OpMode

- Introduced in the August 3rd Beta release.
- An alternative to the event-driven style.
- Closer to the old RobotC programming style.
- Runs commands sequentially.

The Details

- Must extend `com.qualcomm.robotcore.eventloop.opmode.LinearOpMode` class.
- Does not use `public void init() or public void loop().`
- Use `public void runOpMode()`
- Uses methods such as `sleep()` and `waitOneHardwareCycle()` to wait before moving on to the next command.

LinearIR Example

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

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

public class LinearIrExample extends LinearOpMode {

  @Override
  public void runOpMode() throws InterruptedException {

     // setup hardware devices

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

     // wait for the IR seeker to detect a signal

     // wait for the robot to center on the beacon

     // now approach the beacon

     // wait until we are close enough

     // stop the motors

    }
}
```

Current Linear OpMode Bug

“If you use the LinearOpMode class and you create a loop that does not have any interruptible statements within the loop, then when you try to stop the op mode while it is in your loop (by pushing the Stop button the driver station) the op mode will continue to run and the motors and servos can continue to operate!

This is potentially dangerous and could also damage your robot. If you use a LinearOpMode class and use a loop inside, make sure you have an interruptible statement within your loop.

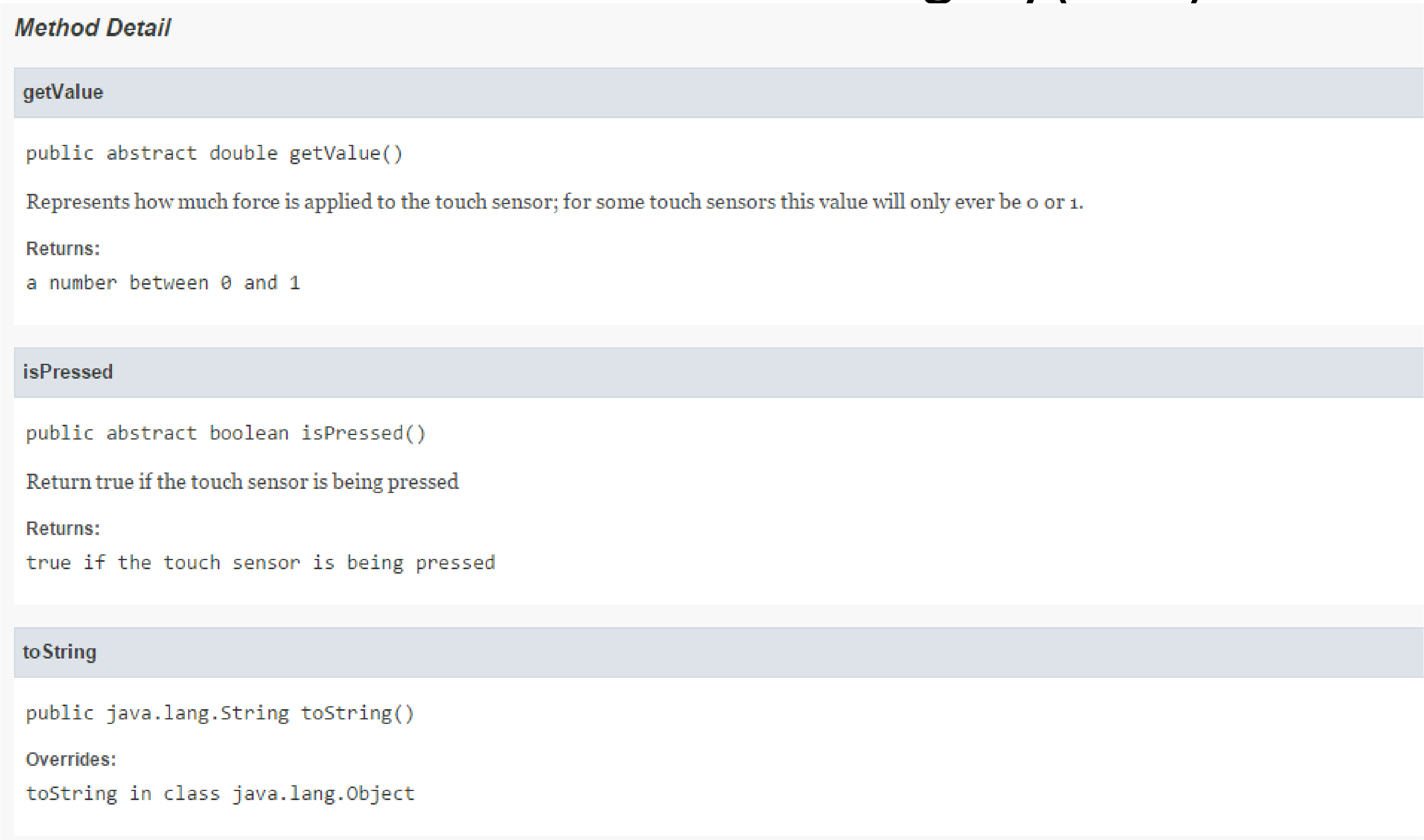
The Fix

In all loops, include an Interruptible statement including

- LinearOpMode.OpModeIsActive()
- LinearOpMode.waitForStart()
- LinearOpMode.waitOneHardwareCycle()
- LinearOpMode.sleep()
- Thread.sleep()

Touch Sensor API

- Works with both new and Legacy(NXT)



Method Detail

getValue

```
public abstract double getValue()
```

Represents how much force is applied to the touch sensor; for some touch sensors this value will only ever be 0 or 1.

Returns:

`a number between 0 and 1`

isPressed

```
public abstract boolean isPressed()
```

Return true if the touch sensor is being pressed

Returns:

`true if the touch sensor is being pressed`

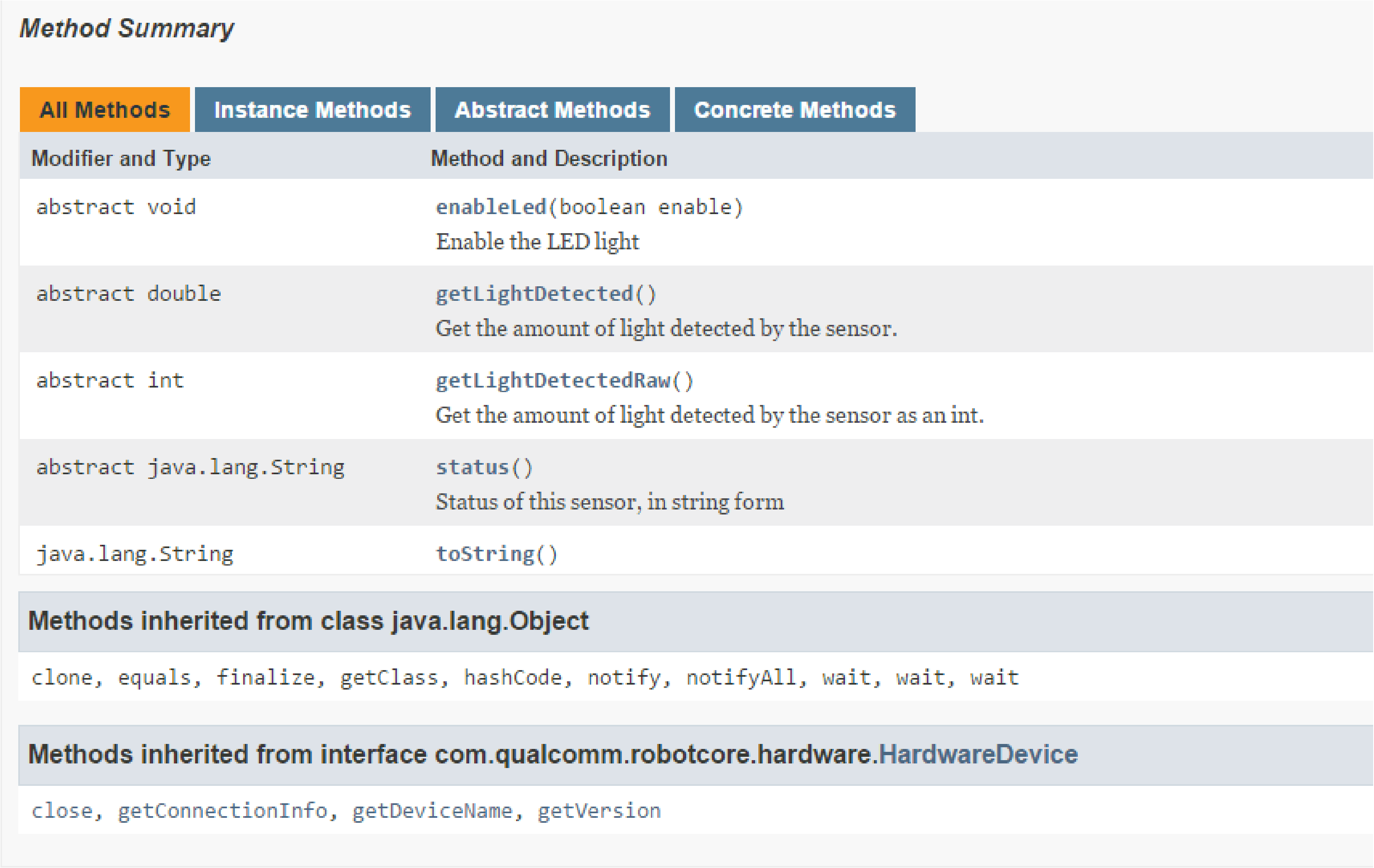
toString

```
public java.lang.String toString()
```

Overrides:

`toString in class java.lang.Object`

Optical Distance Sensor API



Method Summary

Modifier and Type	Method and Description
`abstract void`	`enableLed(boolean enable)` Enable the LED light
`abstract double`	`getLightDetected()` Get the amount of light detected by the sensor.
`abstract int`	`getLightDetectedRaw()` Get the amount of light detected by the sensor as an int.
`abstract java.lang.String`	`status()` Status of this sensor, in string form
`java.lang.String`	`toString()`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Methods inherited from interface com.qualcomm.robotcore.hardware.HardwareDevice

`close, getConnectionInfo, getDeviceName, getVersion`

IR Seeker V3

Two different OpModes provided by FTC as Example:

1. IrSeekerOp – Basic event-driven op mode.

2. LinearIrExample - Linear Op version

Mounting

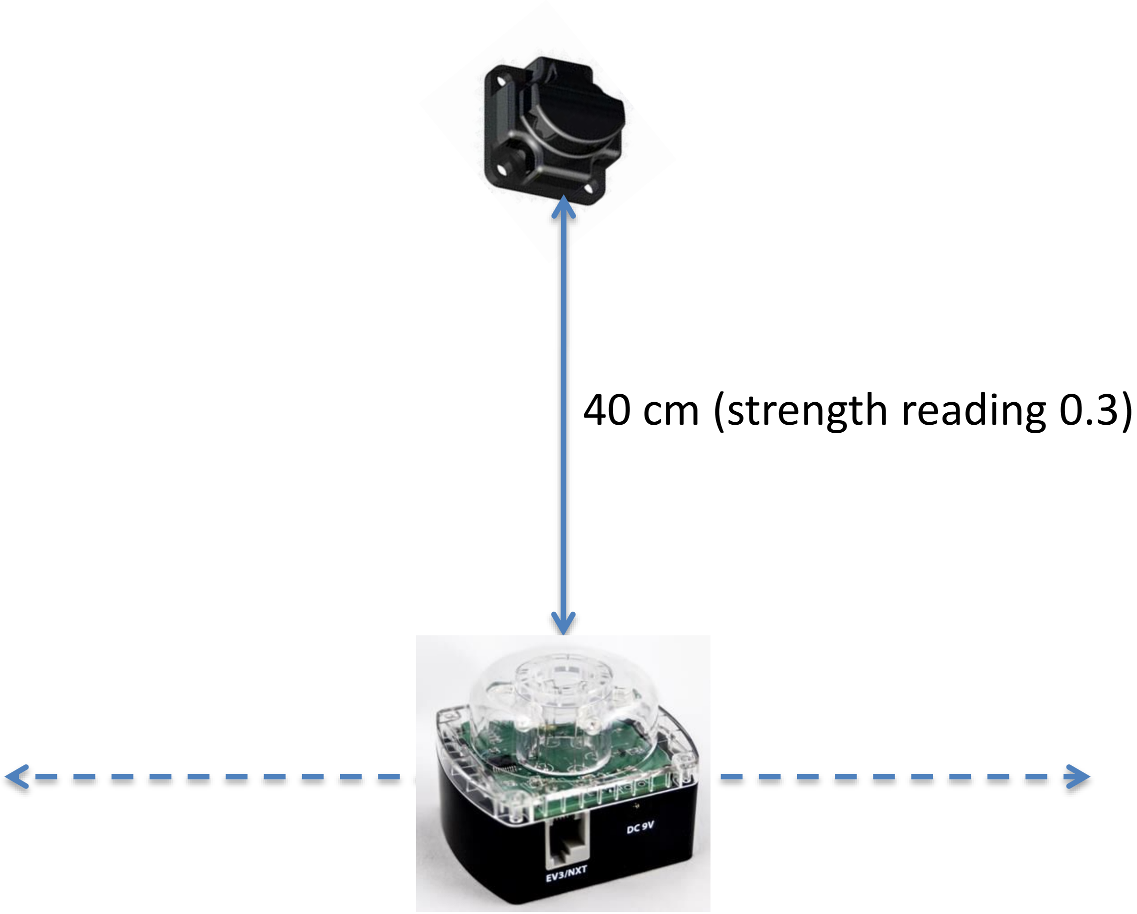
IR Sensor to CDIM

- Black wire = ground
- Plug into side with black strip of the Core Device Interface Module (CDIM)

On the Robot

- Sensor flat
- Curved section in middle facing directly forward

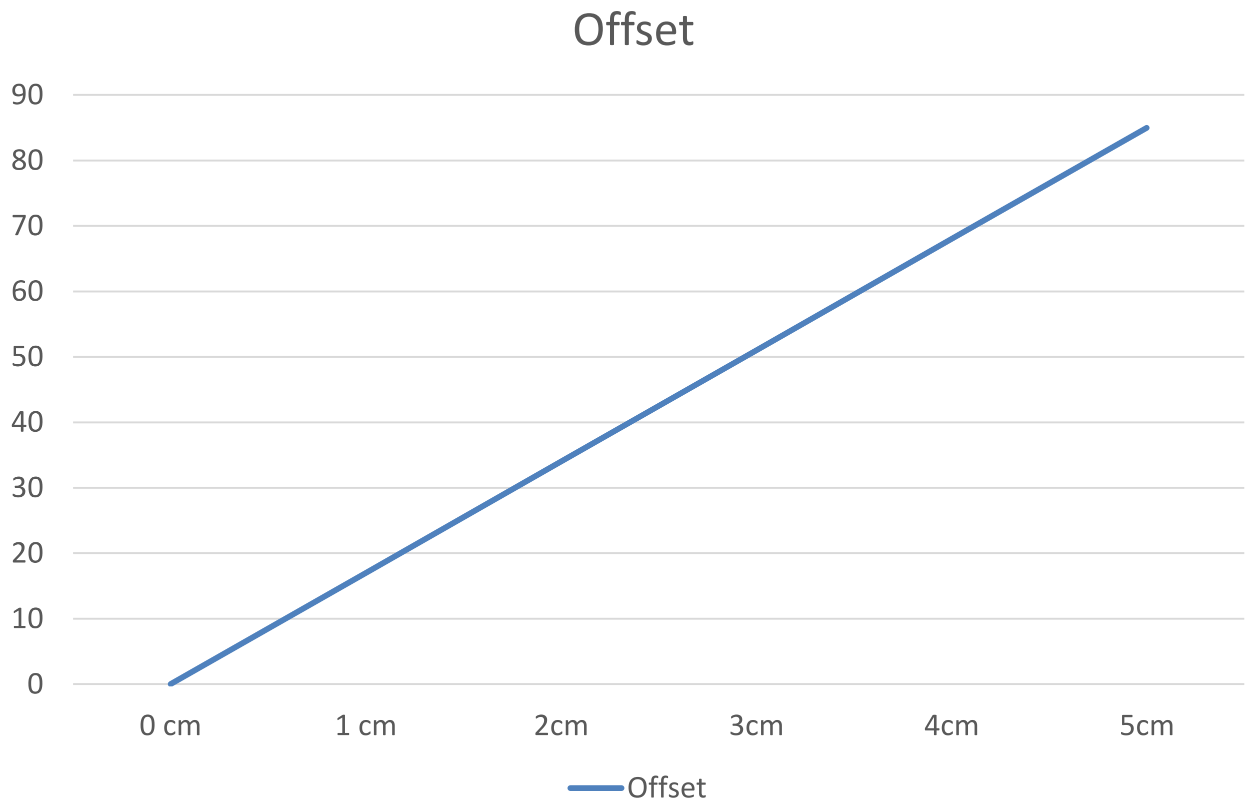
IR Seeker Test Setup



40 cm (strength reading 0.3)

- Credit to FTC Forum user 2009FTC3491

Angle Reading vs. Beacon Offset

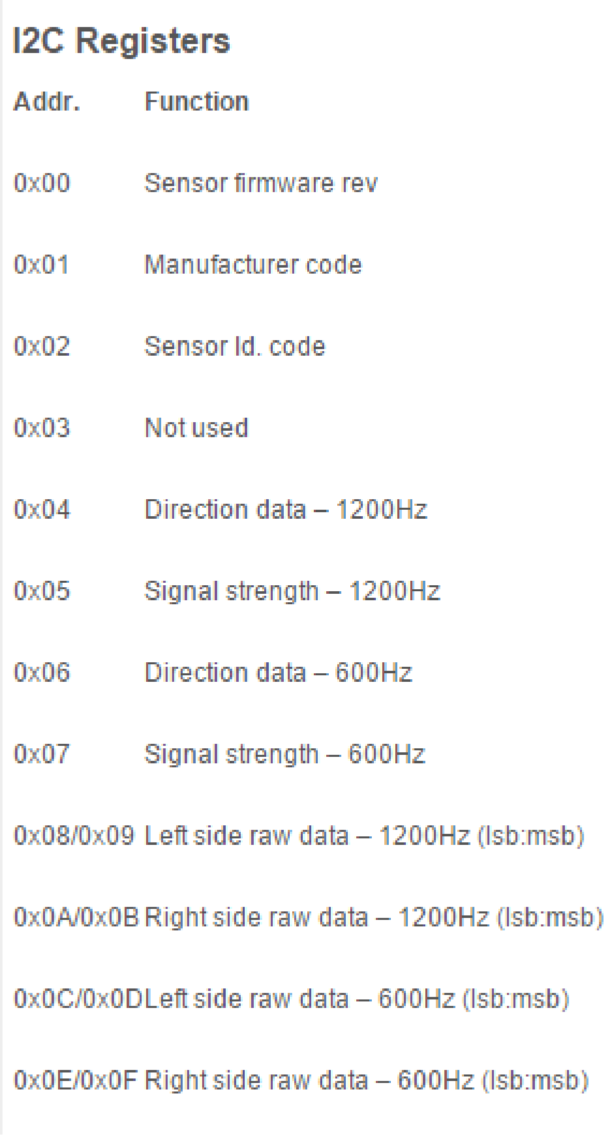


- Credit to FTC Forum user 2009FTC3491

Class IrSeekerSensor Methods

abstract double	getAngle() Estimated angle in which the signal is coming from
abstract IrSeekerSensor.IrSeekerIndividualSensor[]	getIndividualSensors() Get a list of all IR sensors attached to this seeker.
abstract IrSeekerSensor.Mode	getMode() Get the device mode
abstract double	getStrength() IR Signal strength
abstract void	setMode(IrSeekerSensor.Mode mode) Set the device mode
abstract boolean	signalDetected() Returns true if an IR signal is detected

I2C Register



I2C Registers

Addr.	Function
0x00	Sensor firmware rev
0x01	Manufacturer code
0x02	Sensor Id. code
0x03	Not used
0x04	Direction data – 1200Hz
0x05	Signal strength – 1200Hz
0x06	Direction data – 600Hz
0x07	Signal strength – 600Hz
0x08/0x09	Left side raw data – 1200Hz (lsb:msb)
0x0A/0x0B	Right side raw data – 1200Hz (lsb:msb)
0x0C/0x0D	Left side raw data – 600Hz (lsb:msb)
0x0E/0x0F	Right side raw data – 600Hz (lsb:msb)

Bug

Jonathan Berling, Qualcomm:

The signalDetected() method is not working as expected with the IrSeekerV3. It should be looking at signal strength and not the angle.

- The signalDetected() method is looking at registers 4 and 6 (angle) instead of 5 and 7 (signal strength).
- Qualcomm has admitted this is a bug, and will hopefully get fixed in the next release.

IrSeekerOp – Part 1

```
• public class IrSeekerOp extends OpMode {

    final static double MOTOR_POWER = 0.25; // Higher values will
  cause the robot to move faster

    final static double HOLD_IR_SIGNAL_STRENGTH = 0.20; // Higher
  values will cause the robot to follow closer

    IrSeekerSensor irSeeker;

    @Override
    public void init() {
      irSeeker = hardwareMap.irSeekerSensor.get("ir_seeker");

  }
    @Override
    public void loop() {
      double angle = 0;
      double strength = 0;
```

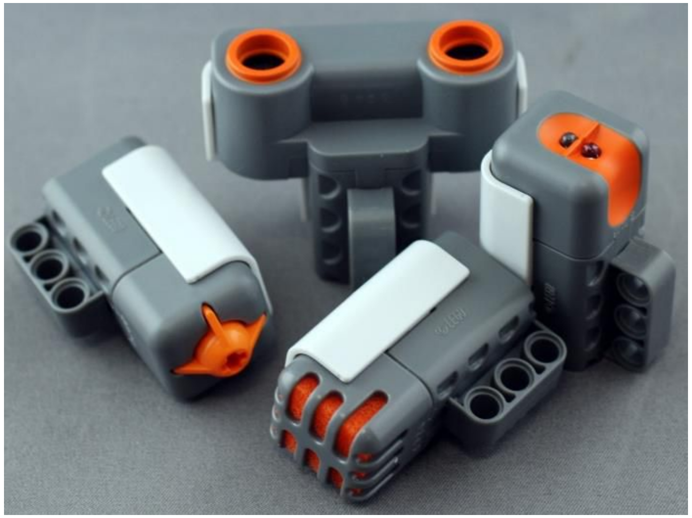
IRSeekerOp - Part 2

```
 // Is an IR signal detected?
    if (irSeeker.signalDetected()) {
      // an IR signal is detected

      // Get the angle and strength of the signal
      angle = irSeeker.getAngle();
      strength = irSeeker.getStrength();

      /*
           Moves according to the direction and strength.
      */
} else {
      // no IR signal is detected
      motorRight.setPower(0.0);
      motorLeft.setPower(0.0);
    }
    telemetry.addData("angle", angle);
    telemetry.addData("strength", strength);

   DbgLog.msg(irSeeker.toString());
```

Gyro

- 4? Options
- Hitechnic Gyro sensor
  - Suspected that the Android platform not fast enough to handle
- Motorola Motor G (Kit Kat)
  - recommended device for international teams in next gen guide which has a built in gyro sensor
- Bosch IMU as a gyro substitute
  - Being tested by teams and results will be published to the FTC forum
- Possible new gyro sensor from Modern Robotics?

NXT Sensors

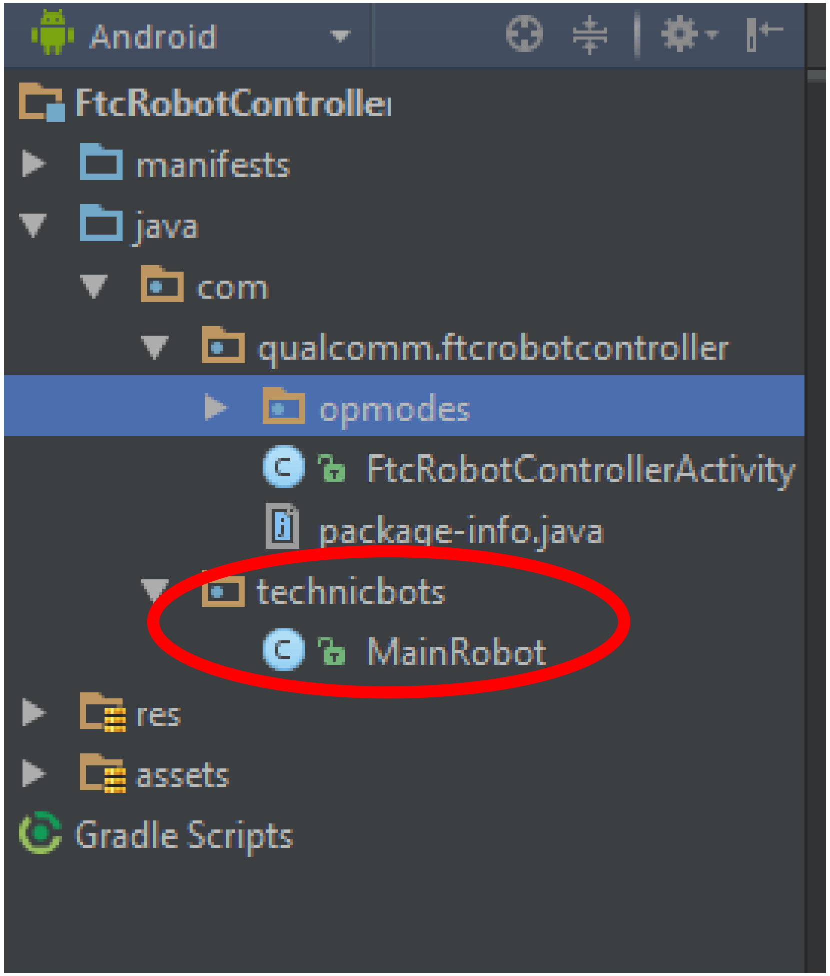


The old NXT Sensors
(Through the Core Legacy Module)

Future ModernRobotics Sensors (From the website)

Sensors in the JavaDoc: Acceleration, Compass, Gyro, Optical Distance Sensor, Touch, IR, Ultrasonic

Code Structure



```
Android
FtcRobotControlle
  manifests
  java
    com
      qualcomm.ftcrobotcontroller
        opmodes
        FtcRobotControllerActivity
        package-info.java
      technicbots
        MainRobot
  res
  assets
Gradle Scripts
```

- Note the “technicbots” package, and the MainRobot class inside it.
- There will be a separate class for each robot.
- The MainRobot class contains methods specific to that robot that are used in multiple opmodes.
- Eg. A state machine for controlling the Linear Slide on a particular robot.

Usage

Usage is as simple as importing

```
import com.technicbots.MainRobot;
```

Then, you can use the MainRobot class in your opmodes.

```
MainRobot.moveLinearSlide(LinearSlide1);
```

GitHub

- A Web-based Git repository hosting service.
- Offers distributed revision control and source code management (SCM) functionality among the team.
- Integrated into Android Studio.

Why Use GitHub? (A hypothetical example)

Say you and a team member are both updating pages on the same website. You make your changes, save them, and upload them back to the website. So far, so good.

The problem comes when your team member is working on the same page as you at the same time. One of you is about to have your work overwritten and erased.

The Solution (Version Control)

But because GitHub keeps a “snapshot” of every change ever made, you and your coworker can each upload your revisions to the same page, and GitHub will save two copies. Later, you can merge your changes together without losing any work along the way. You can even revert to an earlier version at any time.

Github Explanation

- Two level setup
  - – Consists of your workspace/local repository, and the remote server.
- Workflow:

Step 1: Commit to local repository (Copy on computer)

Step 2: Push to remote server.

Recap

- Linear OpMode
- Sensor APIs
- Code Structure
- GitHub Basics

The afternoon session will cover how to setup and use GitHub for your team development.