Android Studio-GitHub-Driver Station Guide

Into The Deep Season Repository: [s0hum/roboactive2425: For team 23405 Krypto Dragons (github.com)](https://github.com/s0hum/roboactive2425)

## Steps to connect Android Studio to Github

1. Back up any previous code by renaming C:\FTC to C:\FTC\_backup (this will back up any of your existing files to FTC\_backup so you can refer if you need to).
2. Go to **First Time Installers** section.

## First time installers

1. Open Android Studio
2. Click on **Get from VCS**
3. Provide the URL to the repository: [s0hum/roboactive2425: For team 23405 Krypto Dragons (github.com)](https://github.com/s0hum/roboactive2425)
4. Select the directory to install the code: **C:\FTC**
5. Use token for login: ghp\_dkQHr7OCNGD2a4hzN08XIsrNFCgRc028O4rR
6. This should clone the repository to your laptop.

## Committing changes to GIT

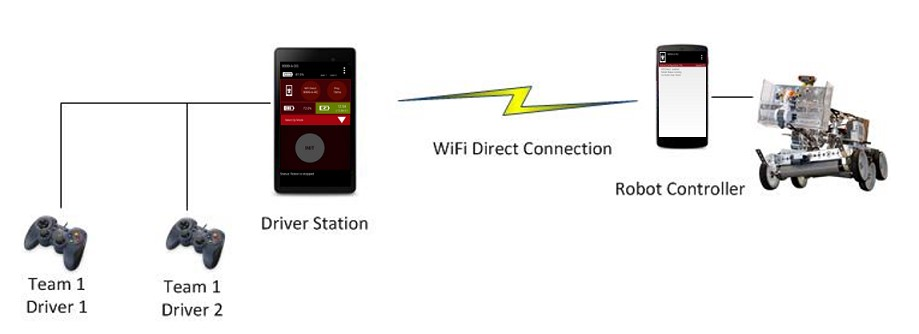
1. Go to Menu -> Git -> Commit.
   1. This will open a dialog. Make sure you are committing the correct changes. Add a comment about your change.
   2. You can keep committing code locally.
   3. Once you are ready, you can push all your changes to Github by going to Git -> Push.
   4. Verify the correct changes are being pushed.

## To get latest files from Git

1. Go to Git -> Pull.
2. This will pull the latest files to your repository.

## Getting Tensor Flow Android App running

1. Create a folder on your local computer.
2. Pull the code from Git.
3. Then import the code to Android Studio.
   1. Instructions for #2 and #3 are given here: <https://www.tensorflow.org/lite/android/quickstart#get_the_example_code>
4. Before you run the project, you’ll need to do the following.
5. You’ll need an Android Phone to use as a test device. First enable **Developer Options** on Android Phone using instructions below.
   1. [Enable Developer Options](https://ftc-docs.firstinspires.org/en/latest/programming_resources/tutorial_specific/android_studio/enabling_developer_options/Enabling-Developer-Options.html)
6. To run the project:
   1. From Android Studio, run the project by selecting Run > Run… and MainActivity.
   2. Select an attached Android device with a camera to test the app. Point to a cat picture and you’ll be able to recognize it.
7. **TBD (Next steps not figured out yet)** To use this for our model, you’ll go to the Android Studio -> assets folder.

Running Code using the Driver Hub

# Initial Set up

1. Install Android Studio
   1. <https://developer.android.com/studio>
   2. Create a folder named ‘ftc’ in C drive.
2. Create a github account with the same email that you have been invited to join team codebase.
3. Follow instructions in [this document](https://onedrive.live.com/edit.aspx?resid=72854863BA26F0AB!1942) to connect to Github and pull code to your laptop.
4. Open *MyFIRSTJavaOpMode*.*java .* This is a simple program which helps with initial hardware set up understanding.
5. Pair the Driver Station To Control Hub

Pairing Driver Station To Control Hub:

1. Go to Driver Hub; Settings; and Pair robot Controller(Make sure wifi is the same) If it does not work hold the power and restart the Driver Hub

2. Connect the game Controller to the Driver Station(Plug it in and do Start a/b)

3. Use the drop down to select the code and run.

Uploading Code to Control Hub

1. Connect laptop to Driver Station (Using the orange cable)

2. Make sure the robot is connected

3. Install the program (Click the green arrow to Run)

4. The control hub lights should flash blue even if it connected. This means it is uploading

5. After the upload is finished (Sound of reconnecting) run your program using the dropdown

Configuration Setup

To run a program using motors, servors or any other class variable, you need to set up configuration files on the hub.

1. Go to settings and configuration

2. To create a new config click new config or use a old one

3. Using the scan see what devices are you using

4. Find what ports you are using and name them as such with the names you used in your program.

5. Save the config and use it to run the program

6. More questions use this link: <https://github.com/ftctechnh/ftc_app/wiki/Configuring-Your-Hardware>

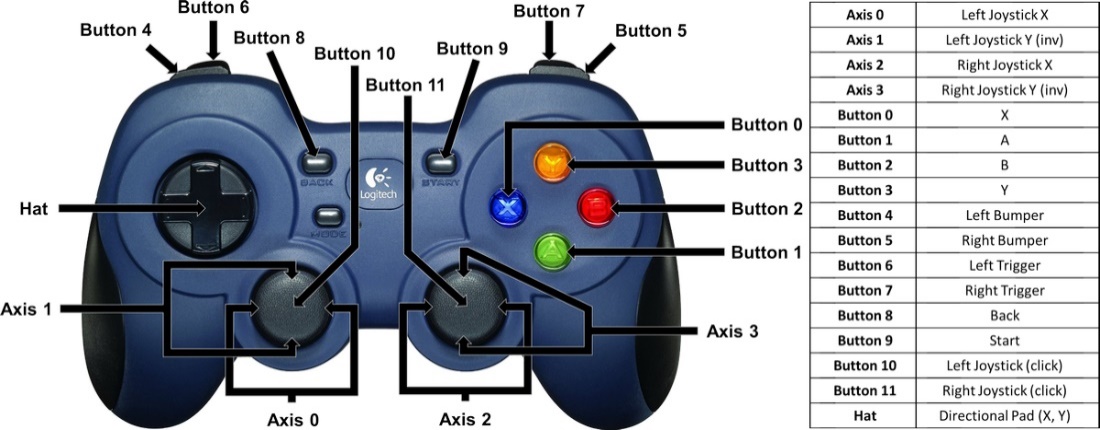
## Software checklist

* Attach the motors and servos to the correct ports
* Attach the battery to the control hub
* Turn on the driver hub
* If they are currently not paired, go to the settings in the to right corner and open pair with robot controller
* Select the WiFi that starts with FTC then some numbers
* If the WiFi is not there, double check everything and hold the power button on the control hub to restart
* Once you are paired go OnBot config and select the Program
* Now plug in the usb into the laptop and the usbc to the control hub
* Install the code by running green triangle
* Unplug the usbc cable and run the Program from Driver hub.
* Plug the game controllers and put them at the right player

## Concepts

1. FTC SDK – libraries to connect to Controller devices
2. Every object (device) is a Class. If you want a Servo, look for Servo class and import it.
   1. import com.qualcomm.robotcore.hardware.Servo;
3. Once you import the FTC librabry, you will define a variable representing that device. Example:
   1. private Servo intakeServo;
4. Now tell the program what your variable represents (connect variable to class type)
   1. intakeServo = hardwareMap.get(Servo.class, "intakeServo");
5. Now you are ready to use your variable to do operations
   1. intakeServo.setPosition(0.5);
6. You can find what operations you can use by looking at the Java class for that device (example: Servo.java) or autocomplete can help you too.

## Design

1. Gamepad layout and thinking of how-to program
   1. <https://gm0.org/en/latest/docs/software/tutorials/gamepad.html>
   2. The following is a picture of how you should think what each button will control (wrong gamepad)
      1. 
   3. <https://docs.revrobotics.com/ftc-kickoff-concepts/freight-frenzy-2021-2022/programming-teleoperated>

## Debug (what could go wrong?)

1. The driver hub and the control hub were not paired with each other.
   1. Configure via Settings on Driver hub. Driver hub needs to connect to Control Hub’s Wifi.
2. Servo could not be recognized
   1. We defined the configuration file on the driver hub to scan and recognize the servo. Check if the configuration port matches the configuration port.

## **Resources**

1. Java - <https://github.com/alan412/LearnJavaForFTC/blob/master/LearnJavaForFTC.pdf>
2. FTC Control System
   1. <https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/The-FTC-Control-System>
3. First Program (OpMode)
   1. <https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/Creating-and-Running-an-Op-Mode-(Android-Studio)>
4. Learn Java for FTC
   1. <https://github.com/alan412/LearnJavaForFTC/blob/master/LearnJavaForFTC.pdf>
5. Initial Hardware Set up
   1. <https://docs.revrobotics.com/duo-control/control-hub-gs/connect-to-the-control-hub-robot-control-console>
6. Running Initial Proram
   1. <https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/Creating-and-Running-an-Op-Mode-(Android-Studio>)
7. Running a motor
   1. <https://ftc-tricks.com/dc-motors/>

Team Emails:

* [ritvikkr@outlook.com](mailto:ritvikkr@outlook.com)
* [Raddythestar@outlook.com](mailto:Raddythestar@outlook.com)
* [vihaansarda@outlook.com,](mailto:vihaansarda@outlook.com)
* [Sohum.p.gupta@outlook.com](mailto:Sohum.p.gupta@outlook.com)

## Sample Program

package org.firstinspires.ftc.teamcode;  
  
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;  
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;  
import com.qualcomm.robotcore.hardware.DcMotor;  
import com.qualcomm.robotcore.hardware.DigitalChannel;  
import com.qualcomm.robotcore.hardware.DistanceSensor;  
import com.qualcomm.robotcore.hardware.Gyroscope;  
import com.qualcomm.robotcore.hardware.Servo;  
  
@TeleOp  
public class MyFIRSTJavaOpMode extends LinearOpMode {  
 private Gyroscope imu;  
 private DcMotor motorTest;  
 private DigitalChannel digitalTouch;  
 private DistanceSensor sensorColorRange;  
 private Servo servoTest;  
  
 @Override  
 public void runOpMode() {  
 //imu = hardwareMap.get(Gyroscope.class, "imu");  
 //motorTest = hardwareMap.get(DcMotor.class, "motorTest");  
 //digitalTouch = hardwareMap.get(DigitalChannel.class, "digitalTouch");  
 //sensorColorRange = hardwareMap.get(DistanceSensor.class, "sensorColorRange");  
 //servoTest = hardwareMap.get(Servo.class, "servoTest");  
  
 telemetry.addData("Status", "Initialized");  
 telemetry.update();  
 // Wait for the game to start (driver presses PLAY)  
 waitForStart();  
  
 // run until the end of the match (driver presses STOP)  
 while (opModeIsActive()) {  
 telemetry.addData("Status", "Running");  
 telemetry.update();  
 }  
 }  
}

Sample Program 2 – Program a Servo

package org.firstinspires.ftc.teamcode;  
  
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;  
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;  
import com.qualcomm.robotcore.hardware.Servo;  
//CRServo  
  
@TeleOp  
public class TeamServoExample extends LinearOpMode {  
 private Servo intakeServo;  
  
 @Override  
 public void runOpMode() {  
 intakeServo = hardwareMap.get(Servo.class, "intakeServo");  
 double tgtPower = 0;  
  
 telemetry.addData("Hello",", Team RoboActive");  
 telemetry.addData("Status", "Initialized");  
 telemetry.update();  
 // Wait for the game to start (driver presses PLAY)  
 waitForStart();  
  
 double position = 0.0;  
 // run until the end of the match (driver presses STOP)  
 while (opModeIsActive()) {  
 telemetry.addData("Status", "Running");  
 telemetry.addData("Status", "Game Started...");  
 telemetry.update();  
  
 tgtPower = -this.gamepad1.left\_stick\_y;  
 // motorTest.setPower(tgtPower);  
 // check to see if we need to move the servo.  
 if(gamepad1.y) {  
 // move to 0 degrees.  
 intakeServo.setPosition(0);  
 } else if (gamepad1.x || gamepad1.b) {  
 // move to 90 degrees.  
 intakeServo.setPosition(0.5);  
 } else if (this.gamepad1.left\_stick\_y == 1) {  
 // move to 180 degrees.  
 intakeServo.setPosition(1);  
 } else if (this.gamepad1.left\_stick\_y == -1) {  
 // move to 180 degrees.  
 intakeServo.setPosition(0.5);  
 } else if (gamepad1.a) {  
 // move to 180 degrees.  
 intakeServo.setPosition(1);  
 }  
 waitForStart();  
 telemetry.addData("Servo Position", intakeServo.getPosition());  
 telemetry.addData("Target Power", tgtPower);  
 telemetry.addData("Status", "Running");  
 telemetry.update();  
 }  
 }  
}