

| Code Tutorial

| The purpose of this tutorial is to demonstrate the setting up of Android Studio, the copying of the FTC Github, and the programming of a simple motor and servo.

| Prerequisites:

1. KNOW HOW TO LOOK SOMETHING UP AND FOLLOW INSTRUCTIONS
 - If something appears normal, it is, any exceptions will be specifically mentioned.
2. Git - Install it.

| A. Download Android Studio

| Just go to the [link](#) and follow instructions there.

| B. Clone the [ftcrobotcontroller Repository](#)

| 1. Open the terminal.

| 2. Using commands, create and navigate to a specific folder.

Use

```
> mkdir <foldername>
```

to make a folder,

```
> cd <foldername>
```

to move to a specific directory,

and use

```
> ls
```

to list all the directories.

| 3. Clone the repo into the folder.

When at the desired folder, use

```
> git clone https://github.com/FIRST-Tech-Challenge/FtcRobotController.git
```

to clone the repository.

| 4. Open Android Studio.

Create an EMPTY new project. Find your folder (remember the location). Then, open the little android icon. You have opened your project.

| C. Code

Create the java class in `TeamCode/java`

You are highly encouraged to not copy paste the code, merely analyze it.

This is to create a car driven by the work of a DC Motor, and steered by the servo.

This will be used in an "RC Car" that will be driven at the club fair. Notice how the implementation uses dead-zones (only making the triggers/sticks activate outside a certain range) to make the controlling more video-game like. You will need to solve certain problems that emerge, such as the one solved by `interval()`.

Important: The documentation for the FTC library is [here](#)

```
package org.firstinspires.ftc.teamcode;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Servo;
```

```

import org.firstinspires.ftc.robotcore.external.Telemetry;

@TeleOp(name = "Drive Mode")
public class tutorial extends OpMode {
    //Define Variables
    float tr_deadzone_lo = 0.1; //The lowest possible value that a trigger
needs to activate.
    float tr_deadzone_hi = 0.9; //The highest possible value that a trigger
can have before it outputs 1
    float deadzone_brake = 0.5; //The value at which the brake will activate
    float st_deadzone = 0.3; // The value that the steer will start
activation with (this is an absolute value, so it must be x in [-1,1]
without being in [-0.3,0.3]

    //Define parts used
    DcMotor drive;
    Servo steer;

    //This changes a value of an interval [lo_end, hi_end] to the
corresponding value in the interval [0,1].
    public float interval(float input, float lo_end, float hi_end){
        float v = (input - lo_end) / (hi_end - lo_end);
        return v;
    }

    //Run once at initialization
    @Override
    public void init(){
        Telemetry.addData("Code", "Loaded");
        Telemetry.update();
    }

    //Looped infinitely until program is stopped
    @Override
    public void loop(){
        //Set drive power based on trigger input
        if(gamepad1.left_trigger >= tr_deadzone_lo){
            Telemetry.addData("LT", toString(gamepad1.left_trigger));
            drive.setPower(double(((gamepad1.left_trigger - 0.1) / 0.8)));
        } else if (gamepad1.left_trigger >= tr_deadzone_hi) {
            Telemetry.addData("LT", toString(1.0));
            drive.setPower(1.0);
        }

        //Set brakes to Right trigger
        if(gamepad1.right_trigger >= deadzone_brake){
            Telemetry.addData("Brake", "Activated");

```

```
        drive.setPower(0);
    }
    //Set servo turn
    if(gamepad1.left_stick_x>=st_deadzone || gamepad1.left_stick_x <= -
st_deadzone){
        Telemetry.addData("Steer:",
toString(interval(gamepad1.left_stick_x, -1,1)));
        steer.setPosition(interval(gamepad1.left_stick_x, -1, 1));
    }
    Telemetry.update();
}
}
```