Congratulations! Welcome to FTC software. There’s a lot of smiles and a lot of tears ;w;

**Table of Contents:**

**General Process / First Time**

**1.1 First time setting up phones**

**1.2 First time setting up Android Studio**

**1.3 First time setting up Github**

**1.4 First time setting up an OpMode**

**1.5 First time setting up encoders**

**Common phone specific issues**

**2.1 Phone errors**

**2.2 Connection issues**

**Rev Hubs**

**3.1 Set up your wires**

**3.2 Wire problems**

**3.3 What do the colors on the Expansion Hub mean?**

**3.4 When the Expansion Hubs can’t see each other** (how sad)

**3.5 When Expansion Hubs can’t see the phones**

**3.6 When the phone can’t see the Expansion Hub (aka the Procedure)**

**Battery Troubles**

**4.1 Expansion Hub not seeing the battery**

**Android Studio**

**5.1 General Advice**

**5.2 Your computer cannot see the phone**

**5.3 Issues downloading code to phone**

**Github**

**6.1 Github Desktop not connecting to Android Studio**

**6.2 Merging issues**

**State Machine**

**7.1 Cannot telemetry()**

**7.2 Next state null not working**

**7.3 When you run, the state machine is trying to do a lot of things at once**

**7.4 Things are not working quite right**

**Encoders**

**PID**

**9.1 Robot keeps veering back and forth**

**Sensors**

**10.1 If your sensor can’t be seen by the Expansion Hub**

**10.2 If your sensor doesn’t work the way you expected**

**10.3 MR Gyro Sensor**

**10.4 Rev IMU Sensor**

**10.5 MR Range Sensor**

**10.6 Rev Color Distance Sensor**

**10.7 Touch Sensor**

**General Process/First Time**

**1.1 First time setting up the phones:**

* In case you didn’t know, the phones we use for FTC are of two types that communicate with each other: **Robot Controller (RC)** or **Driver Station (DS)**. The Robot Controller is plugged into the robot and runs the code that you write, while Driver Station is the phone from which you choose programs for Robot Controller to run. :)
* Turn on your phones if they aren’t open already - and go to your phone’s settings
  + No matter what model you use, you should run into an option near the bottom to look and edit your phone’s general information. After you have done this, change your phone’s names to “**[team#]-RC”** and **“[team#]-DS”** for the robot controller and driver station respectively. (ex: 7890-RC and 7890-DS)
    - If you have an android phone, sometimes the name will automatically show up with “[phone]” before the rest of the name. We can’t really control this, so don’t worry too much. Judges will understand, though it can make things a bit inconvenient.
    - To name your phones, go into the DS app and then click “Settings” on the upper right hand corner. Once in Settings, scroll down and find “Driver Station Name.” Click it and change the name of your DS. To change the name of your RC, just scroll further down in Settings and then click “Robot Controller Name.” You can also change the name of the RC by going into Settings in the Robot Controller App and then clicking “Robot Controller Name.”
* Go to the Google Play Store and download the FTC Robot Controller app for RC and FTC Driver station for the DS - wait a minute or so. You must hum a catchy tune as you wait or else it won’t work
* After you have downloaded the apps, it’s time to deal with something a bit more frustrating - connecting the phones!
  + First, go into the internet settings of each of your phones and go to their advanced wifi settings, and then scroll down on the following menu and select ‘Wifi Direct’
    - On your RC phone, you should see the DS phone in your bluetooth connection options and vice versa. Select DS from your RC phone and you should get a prompt to connect from your DS phone (accompanied by a loud noise). Now it’s time to go into the specific robo apps
  + Open both DS and RC
  + On your DS app, click on the stacked three buttons to access settings, and then select ‘Pair With Robot Controller’
  + On the next screen, set the ‘Filter Wifi Devices’ option to ‘OFF’ - you should see the name of your RC in the list.
    - *You will also see the names of other teams’ phones - please do not touch those even if your phone’s name is not present*
  + Select your RC and go back to the default screen of the app. You should get a prompt to connect your phones - select connect, and you’ll be good! The name of your RC phone should show up at the top of your DS screen in white.
    - *If the prompt does not show up, close the apps and open DS before RC. DS should provide a prompt for you to connect to RC, and then you’ll receive a prompt on RC to connect to DS.*

**1.2 First time setting up Android Studio**

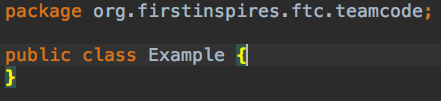
* Go to this website to download Android Studio, the IDE that we use: <https://developer.android.com/studio>
* Now, you must download the FTC SDK. <https://github.com/ftctechnh/ftc_app>
  + On the right hand side, there is a green button that reads “Clone or Download.” Then, download the ZIP file and then unzip it. Once you have done that, open it in Android Studio.
* To make a new class, look at the left side of Android Studio and open “TeamCode > src > main > java > org > firstinspires > ftc > teamcode.” And then right click on the folder teamcode. On the top of the options menu, hover over “New” and then click “Java Class.”

**1.3 First time setting up Github**

* On established teams:
* If you don’t have one yet, please create a GitHub account (make sure your password is safe!): <https://github.com/join?source=header-repo>
* After doing so, let another member of your team know so you can be added to your Team’s group
* If you’re on a new team
  + Oh hey, your team color is pink, right? Nice.
  + Anyway, join github as you have already done - and then click the plus sign next to your profile picture and select ‘new organization’
    - Enter the organization name - it should include your team name and number.
    - Add an email - if you have a software head, you may want to use theirs
    - DO NOT PAY. Get the free version, please!
  + Alright, now you can add team members and then select your organization info
* Creating a Repository
  + Go to the previous plus sign next to your profile picture, and then type in a name which has your team# and the game’s name (you can do whatever else you’d like with this, but at least have these for organization’s sake)
  + After this, you select ‘upload files’ and select the encapsulating folder from android studio
* Setting up Git:
  + There are two ways that you can commit your code to the Git repo. Copying and pasting code directly into the repo files is NOT one of them.
  + Please use either GitHub Desktop (downloadable from the GitHub Website) or use Git directly in your Terminal / Powershell

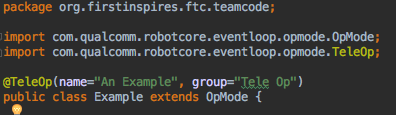
**1.4 First time setting up an OpMode**

* When you first create a new Java class, it will look something like this:

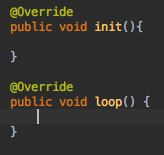


However, in FTC our classes look different

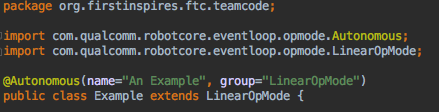
* If you’re making a TeleOp, then the beginning of your class should look something like this:



* For a TeleOp code, you need to have these two structures:

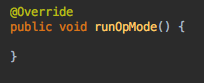


* If you’re making an autonomous code, then the beginning of your class should look something like this (a lot of this is done automatically):



Important things to note in this image are:

* The name you want to appear on the phone goes inside the quotes “An Example”
* The actual name of the file goes after public class
* The type of code this is goes after extends… in this case it is LinearOpMode
  + For Autonomous, you need this structure:



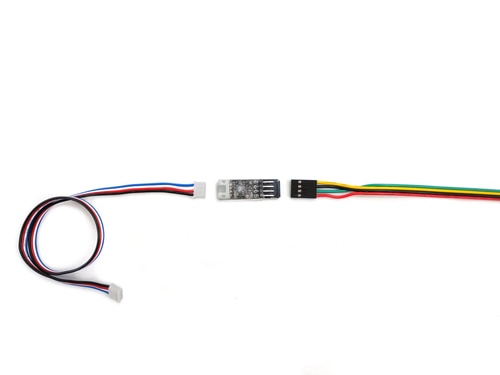
* It is important to note that to program for FTC, you need to import a lot of libraries. For example, to work with a DcMotor you need to import DcMotor.

**1.5 First time setting up encoders**

* Electrical:
  + Locate the ports on the Expansion Hubs and Motors where you can insert the encoder. They should be right next to where the motor wire is.
  + Then, find an encoder and plug it in. An encoder looks like this if you’re using an MR motor:



You’ll need to plug this into your MR motor, but afterwards you’ll need to convert it to a wire which can be plugged into the rev:



If you’re using REV motors (the black ones with a hexagonal hole), you can just use the blue wire alone

* Once you’ve plugged in the encoders to your motors, plug them into the ports labelled “ENCODER” next to the motor wire that they are connected to.



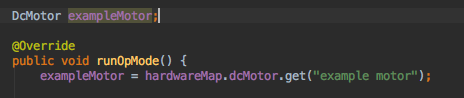
* Make sure that your encoders are plugged in the right orientation when you convert the MR ones to the REV ones - if you don’t, you may end up confusing a perfectly good wire for a broken one
  + A generally foolproof way of orienting the wires is to find the black wire, and making sure it goes to the ‘GND’ section of the converter (for ground)



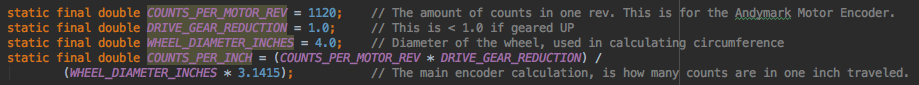
* Setting up encoders in your code:
  + Firstly, in order to use encoders, you need objects to use those encoders on. So, you need to have this import statement, importing DcMotors.



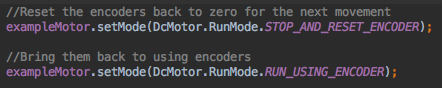
* You will also need to declare your motors and hardware map them. The hardware map is name you will enter as the motor on your phones.



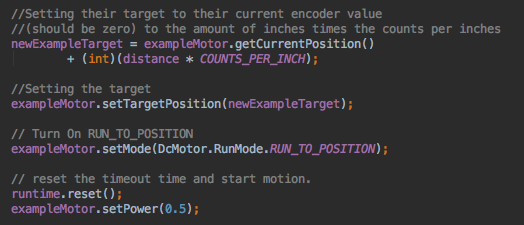
* Now, to set up your encoders you will need to enter this calibration code:



* Then, in the structure runOpMode(), you need this to finish setting up your encoders:



* To actually have your encoders work, first make a variable for how far you’d like to go (a good name is distance) whose data type is a double. Then, in your code you will want this to set up the logic every time you use encoders:



* Tip: Since you need to use this when your use encoders, putting it in a method will greatly increase efficiency.
  + If you would like your robot to drive straight for a specific distance, use the distance variable (which returns how far the robot has gone) and stop the robot when it reaches your goal distance.

**Common phone specific issues**

*Most of these errors show up when you select the ‘init’ or ‘play’ - a loud sound also plays, so no worries if you get a bit startled*

**2.1 Phone errors:**

* *There are many errors that may be displayed on the phone before or after running the code. Sometimes these errors will prevent you from running code and others require you to restart the robot, losing precious time*
* “Warning: REV Robotics USB Expansion Hub Portal [...] detached”
  + This means that your phone is not connected to the Expansion Hubs or that it cannot see it.
  + Make sure to check your wire connection and ports (unplug and replug) and restart the app and robot.
* “Warning: problem with ‘...’”
  + This means that something is not connected to the Expansion Hub. It could be a sensor, servo, dcmotor, etc. It will display the name of the object.
  + Make sure to check the wire connection and the ports. If it’s plugged in, push it in more. If that does not work, unplug the wire and then plug it back in.
  + Then, make sure to check your config, it needs to be exactly the same name as you hardware mapped in your code (it is case sensitive). Make sure there is no space after the name in your config.
  + If you run your code and disregard this message, there might be an emergency stop (look below)
* “Robot Status: EMERGENCY STOP

Error: User code threw an uncaught exception: IllegalArgumentException - Unable to find a hardware device with the name ‘...’”

* This means that you’re running your code and it cannot find one of the objects you hardware mapped.
* Check all of your wire connections. If they’re plugged in, unplug it and plug it back in.
* Also, check your config and make sure it is the same name as the one in your code. It is case sensitive and also make sure that there is no space after the name in your config.
* “Robot Status: running

Error: OpMode ‘...’ stuck in stop(). Restarting robot controller app.”

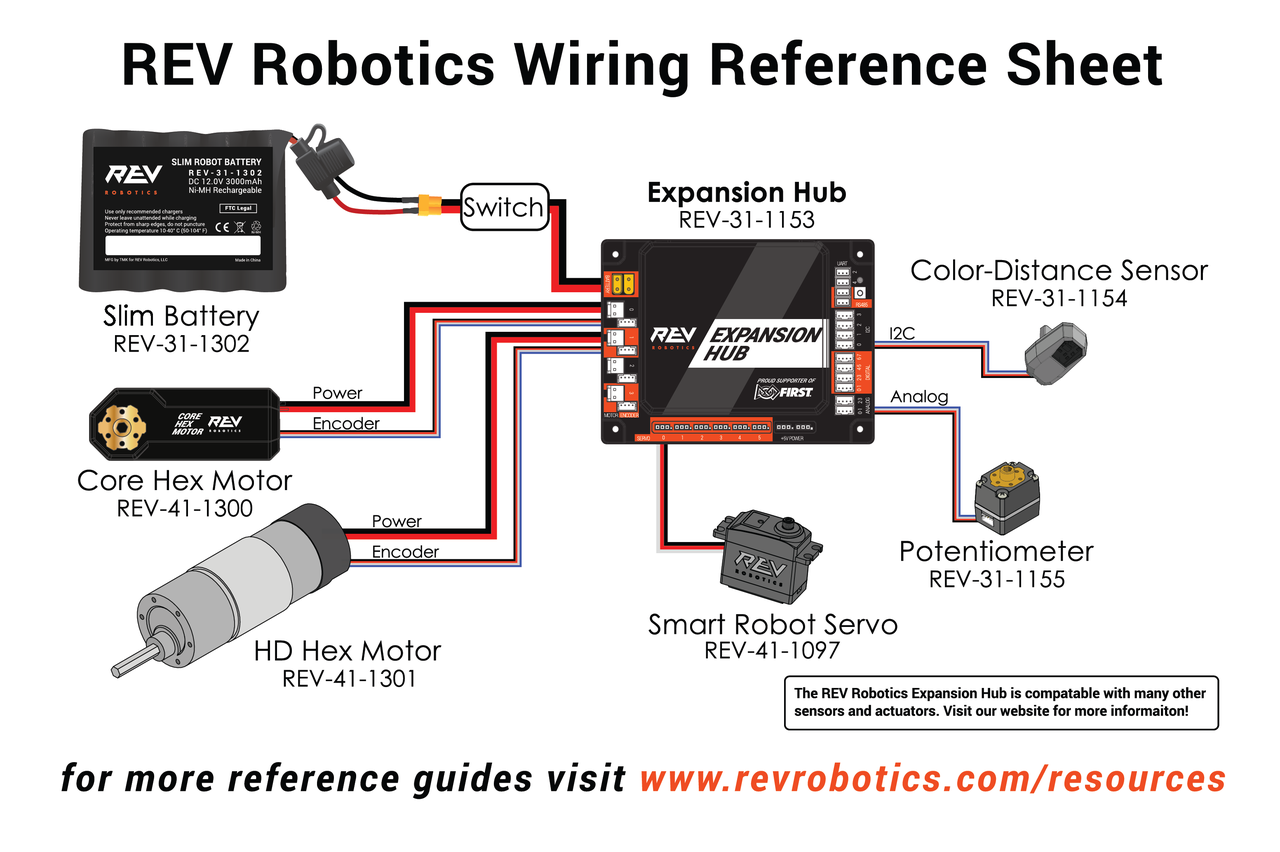
* This means that you were in a loop when you stopped the code. As a result, you got “stuck in stop()”
* Restart the robot and then go through your code. Check where you have loops and make sure that their conditions can be satisfied. Oftentimes adding a condition that checks if the OpMode is active ( opModeIsActive() ) will help.

**2.2 Connection Issues**

* *Sometimes your phones will not be able to connect. Do not fear!*
* First, restart both of the apps. Then, open the DS first and then after waiting for it to start, open the RC. Press connect if prompted. If that does not work, restart the phones and then try to connect them again, making sure to first open DS and then to open RC.
* If this does not work, go to Settings on your DS. It is in the top right corner. In Driver Station Settings, make sure that the pairing method is wifi direct and then click “Pair with Robot Controller.” On the bottom half, you’ll find a list of Wifi Devices. If you find the RC that you wish to connect to on the list, click it and then connect. If it does not show up, turn off filtering wifi devices and then scroll through and find your RC.
* If none of these methods work, try going into Settings in the DS and go into “Advanced Settings,” then click “Change Wifi Channel.”

**Rev Hubs**

**3.1 Set up your wires:**



**3.2 Wire problems**

* Important wire spaces
  + While almost all of the spaces in the REV are open, there are some that you must leave open in configuration to make sure that certain internal functions of the REV work correctly
    - If you **plan to use the IMU**, leave 12C bus port 0 open - it will already have a hardware map nap of ‘imu.’ (Before doing anything with the electronics it might be a good idea to check what the IMU does if you are not sure whether or not you are going to use it)
    - When you’re **using two REVs**, make sure the area labeled ‘RS485’ in the UART port is where you plug in the connection wire. Anywhere else and your phone will not be able to see the REV it is not directly plugged into
      * If adjusting the wire to the right place doesn’t work, try using another expansion hub just in case
* Sometimes, the only wires available will be a bit too wobbly to leave unattended while the robot is running on its own. In this case, you may need to do some tricks with tape
* What tape to use
  + **Electrical tape** is most certainly the best option in most cases in terms of stickiness and safety. It’s a bit stretchy, so you’ll want to work with another person. Have them hold the wire into the hole while you wrap at least two pieces of tape around the connection space. Try to have it surrounded at all times
* Tape techniques
  + When wrapping, make sure to tightly wrap the offending wire and port together. If your tape is only on the wire, that is not helpful (unless you have an exposed wire). The important thing is to make sure your tape doesn’t stick together and to wrap it securely.
* If your wire is broken, make sure to label it “BROKEN” with masking tape so nobody else has to find out the hard way.

**3.3 What do the colors on the Expansion Hubs mean?**

* Green:
  + This means that both the battery and phone is connected to your Expansion Hub. Everything is good!
  + Also, if it occasionally blinks blue, don’t panic, everything is still good.
* Blue:
  + This means that the battery is connected to the Expansion Hub. The phone is not connected. If your phone should be connected, refer to “**3.5 When Expansion Hubs can’t see the phone.**”
* Orange:
  + This means the phone is connected to the Expansion Hub but the battery is not. If your battery should be connected, refer to “**4.1 Expansion Hub not seeing battery”**

**3.4 When Expansion Hubs can’t see each other**

* If you’re using two expansion hubs, sometimes they will not be able to see each other or the phone will not be able to see both of them
  + First, make sure the area labeled ‘RS485’ in the UART port is where you plugged in the connection wire. Anywhere else and your phone will not be able to see the REV it is not directly plugged into
    - If adjusting the wire to the right place doesn’t work, try unplugging it and then plugging it back in. If this does not work, replace the wire.
      * If your wire is broken, make sure to label it “BROKEN” with masking tape so nobody else has to find out the hard way.
  + Then, try plugging in your phone into the other Expansion Hub. If that does not work, then plug your battery into the other Expansion Hub.
  + If none of this works, try going into the DS’ Settings in the top right hand corner. Scroll down to “Advanced Settings” and then click “Expansion Hub Address Change.” Try changing the addresses of both of the Expansion Hubs. Sometimes this works. We don’t know why, but it does.
* If nothing works, try replacing one of your expansion hubs with a new one. If this does not immediately work, check your wire and try switching around the phone and battery (as further explained above) and try changing the addresses of the Expansion Hubs. It is important to keep testing out different parts of the system in order to try and isolate the problem. By doing the above methods you can narrow down where exactly something has gone wrong and potentially identify a broken piece of equipment.

**3.5 When Expansion Hubs can’t see the phone**

* You know that the Expansion Hubs cannot see the phone because the LED will be blue (if the battery is connected) or there will be no light (if no battery is connected)
* Check to see if the wire is fully connected.
  + Push in the wire and if that does not work, unplug it and then plug it back in. Then, change out the wire. After that, plug it into the other Expansion Hub.
  + If your wire is broken, make sure to label it “BROKEN” (along with the current date) with masking tape so nobody else has to find out the hard way.
* Next, check the port on your phone
  + Test to make sure your phone can be seen by your computer in Android Studio. If it can’t, then your port may be broken. But make sure you’re testing with a trusted USB-phone wire, those are much easier (and much cheaper) to replace than phones
* Then, borrow another RC and see if the Expansion Hubs can see that one (lights are green). If it cannot, then your Expansion Hubs have a problem. If it can, then your phone may have a problem.
  + Remember, before declaring that your Expansion Hubs or phones are broken, double check the wire and make sure they are properly plugged in.

**3.6 When the phone can’t see the Expansion Hub (aka ‘The Procedure’)**

1. Sometimes, your phone won’t see the Expansion Hub despite the fact that you’ve tried replacing the wire and the hub. This is most likely due to the fact that your hub’s firmware has not been updated. In this case, you can force the phone to recognize the hub with some simple steps:
   1. Disconnect the battery from the hub
   2. Unplug the phone from the hub
   3. Restart the phone
   4. Give the phone a few moments to turn on/de-lag, and then get off of the lock screen
   5. Plug the phone back into the hub - it may take a minute, but *do not open the app*. The whole point of this procedure is to have the phone recognize the hub and open it on its own
   6. If the app still does not open after a minute or so, plug the battery into the REV. At this point the app should be open and the hub should be seen
   7. If the app *still* does not open, then you are able to open the app - there’s a chance the procedure still worked.
      1. If the procedure *does not* work, then check your phone wire and hub again.

**Battery Troubles**

**4.1 Expansion Hub Not Seeing The Battery**

* Switch
  + Hey, is your switch clicked to the ‘O’ instead of the ‘I’? Select the ‘I’ to actually turn the robot on...
* Connection in Wire
  + Your wire might not be plugged in properly so make sure to unplug it and then plug it back in! Also, it might be broken so swap it out. After that, check the port, it might be broken.
  + If your wire is broken, make sure to label it “BROKEN” (and the current date) with masking tape so nobody else has to find out the hard way.
* Many of your **testing issues** can be explained with a **dead battery**
  + As you test, the voltage listed on your DS app should read between 11-14. On the DS this will show up as a vibrant green. When it falls below this, performance can be inconsistent, especially if you’ve hard-coded your autonomous (try your best not to hard code your autonomous)
    - There’s a simple solution for this, just disconnect your battery from your robot, and replace it with one that’s been charging - in an optimal situation, the light on the charger will be green indicating that you are taking a battery which is full. When you plug your lower battery in, make sure the light on the charger is red or else the charger may not work anymore.
* Fuses
  + If your robot is not being recognized by the REV, double take a look at the trapezoid-looking covering which is on the battery wire. Unsheath it. You should see a fuse which looks something like this:
    - If you don’t see a fuse:
      * The solution is simple - get a fuse and push it into the battery. It will take a bit more force than you may expect, so don’t be surprised if you have to use a support to get it in fully. If you can get the sheath fully on as it was when there was no fuse, you’re in the clear.
    - If you see a fuse with a broken ‘u’:
      * This means your current fuse is broken, too much electricity went through and it can’t regulate traffic anymore. To fix this, grab some needle-nose pliers and pull - since it’s broken feel free to tilt it horizontally to loosen it up. Then just implement a new fuse as described above
    - If you see a fuse which isn’t yellow:
      * This fuse shouldn’t work properly with the battery
    - If you see a fuse which is fully intact:
      * Press down on the fuse until it adjusts fully into the hole. If it’s already in there, then your problem is somewhere else.
* Charging
  + It is important to note that the chargers will indicate a red light for a battery that is charging but is not quite full, and a green light for a fully charged battery. However, it will also show green if the battery is missing a fuse among other issues. Double check.

**Android Studio**

*First thing - don’t update anything*

**5.1 General advice**

* In general, if you have any bugs in the IDE, first try restarting the Android Studio app.
  + For example, if your Android Studio says there is an error in the class but doesn’t actually highlight the error, quit out of Android studio and then reopen the app.

**5.2 Your computer cannot see the phone**

* On your phone’s settings, make sure that “USB debugging” is enabled
* When you first connect your phone and computer, there will be a pop up that asks if the computer is trusted. Make sure to click “ok.” If you do not, the computer and phone cannot be connected.
  + Also, make sure to connect your phone and computer using a USB-phone wire which is designed for data transfer and not just charging
* First, try checking the connection, make sure the wire is plugged in properly. Then, unplug it and then plug it back in.
* If that does not work, try:
  + Restarting the app
  + Restarting the phones
* Then, try to plug the USB into a different port on the computer. If that does not work, try swapping out the wire and if this works, please label the wire as broken.
* If it does not, then borrow a phone from another team. If this works, the port on your phone may be broken and you may need a different phone.
* After that, try using a different computer to download the code.
  + *If you utilize this solution, do not rely on it. It is important to not rely on just one computer for code transfer, so try to fix the offending computer ASAP*

**5.3 Issues downloading code to Phone**

* If your issue is that the computer cannot see the phone, then refer to “ **Your computer cannot see the phone” (5.2)**
* First check that the wire is properly inserted into your devices and that “USB debugging” is enabled on your phone:
* Solution 1:
  + Swap out the wire, if that works, label the wire as broken
  + If the issue persists, proceed to *Solution 2*
* Solution 2:
  + Restart the phone
  + If the issue persists, proceed to *Solution 3*
* Solution 3:
  + Swap out the phone
  + If the problem persists, check that it is not the issue mentioned below the issue title, if not, try each solution again. If the issue persists thereafter, check your computer.
* If this does not work, strap yourselves in for the most F U N experience of your life.
  + Between each of these steps, make sure to check to see if the code downloads. Do this by changing the name of the file (the one in quotes which shows up on your phone) every time.
  + 
  + First, take out all of the files except for your TeleOp file and check that that downloads.
  + Then, add in one autonomous file at a time, each time making sure that they can download.
  + Next, go into the code you were working on.
    - First, add in your constructor and imports. Check to make sure it downloads.
    - Then, start declaring your variables. Make sure it downloads.
    - After that, add in your methods in small groups (for example, add in a few methods at a time). Check to see if it downloads every time.
    - After this, it will work :)))))
    - For an extra bit of fun, put your files into a difference checker and find that there is no difference between the files you have and what’s in those files.

**Github**

**6.1 Github Desktop not connecting to Android Studio**

- First check your computer’s connection, then:

* Solutions:
  + Restart Github Desktop and/or your computer
  + Make sure that your repository is at the right filepath in both GitHub Desktop and Android Studio
  + Reset your repository by either cloning it again through the interface or by downloading it, setting up a filepath, and then adding that to GitHub Desktop
  + Redownload GitHub Desktop AND reset the repo (above step)
  + If nothing works then you might just need to learn Git commands and try to use that. Do NOT copy and paste code into the repo!

**6.2 Merging Issues**

* Fork path.
* Get previous version of afflicted code

**State Machine**

* For a little bit about State Machines, check folder in All FTC Teams → Software Library called *State Machine Setup*
  + Read the README
  + Glance through the example codes - they’re very helpful for understanding what to do for state machines

**7.1 Cannot telemetry**

* Your states for the state machine are not OpModes, so you cannot telemetry any variables in the state
  + But luckily the state machine is an OpMode! So you can use telemetry there
  + If you want a telemetry within the states you will have to use a getter method
    - Since telemetry only works inside of your state machine, not the individual states, you must use a getter method to obtain certain variables from the state which is running so it’s present in the machine.

**7.2 Next state null not working**

* Sometimes State Machine doesn’t like it when it leads into a null state
  + To fix: make your own state that just stops the robot by setting all of its motor power to zero

**7.3 When you run, the state machine is trying to do a lot of things at once**

* Make sure you distinguish between *start method* (initializes everything) and *update method* (what it does until it calls next state)

**7.4 Things are still not working quite right**

* Keep track of parameter vs. field
  + State machines are very much based on what you give them, and it’s very to have accidentally submitted variables in the wrong order into the statemachine. Android Studio should show you what the name of each desired variable is when you are inputting them

**Encoders**

* Encoders are useful on your robot because they allow you to input a distance for your robot to move. Without encoders, you depend on timing the motors to go a distance which means it is dependant on the material you work on and more importantly, the battery voltage.
* **Encoder Errors**
  + If you’re testing code which relies on encoders, you may notice that one of your motors is lagging behind or speeding ahead of the others
    - Before tightening in the wheel and taking up table space, it may be best to double check if the encoder associated with said motor is loose or not. If it isn’t, replace it and test again. If your encoder isn’t connected properly the motor’s data cannot be tracked which would lead to drag
    - Is every motor being initialized with .RUN\_USING\_ENCODERS and .RUN\_TO\_POSITION? The example code only utilizes one motor, so make sure you didn’t forget any in setup
  + If the numbers you put in for distance feel arbitrary compared to how much the robot is actually moving, double check that the specifications (ie the symbolic constant DRIVE\_MOTOR\_DIAMETER and COUNTS\_PER\_INCH) are true to life. If your robot’s movements are still not consistent at this point, then you should see if your battery is working or if there is something off with your wires

**PID**

* PID or PID controller is a useful tool because it calculates error of where your robot is vs. where your robot should be
  + For example, say you want your robot to move in a straight line → PID will help you get back on that course - in case you veer off for any reason
  + For a brief summary of what PID is: <https://en.wikipedia.org/wiki/PID_controller>

**9.1 Robot keeps veering back and forth**

* You made margin of error too small
  + so every little thing makes the robot correct itself
* You made correction too large
  + So every time it encounters error - it corrects itself too much
* Sometimes it’s a little bit of both → plug in numbers and test it out

**Sensors**

**10.1 If your sensor can’t be seen by the expansion hub**

* First, check the wire connection. Make sure it’s properly plugged in and unplug it and then plug it back in.
* Next, if you are using a Modern Robotics sensor with a Rev Expansion Hub, you will need some special parts:



* After that, check your configuration. Make sure that the sensor is connected to the right port as stated in your config. Also, make sure the names in your config are the same ones you hardware mapped to. Check to make sure there is no space after the names in the config as that will mess it up.
* If this still does not work, plug your sensor into a different port, just make sure to also change the config.

**10.2 If your sensor doesn’t work the way you expected**

* Perhaps you expected your robot to stop five inches from a wall. Maybe you wanted it to follow a colored line. Maybe you wanted it to turn 45º. If your robot isn’t doing what it is supposed to, first, check your code and make sure your logic works.
  + After that, check your wire connections. Make sure your sensor is properly plugged in and your DS isn’t returning any errors.
  + Once you have done that, you will want to make a test code (no tele-op or autonomous movement) to see if the sensor returns the values or acts the way you thought it would. There’s a chance there’s an existing one in the external files of the FTC repository on github.
    - First, create and set up a java class. Make sure you import everything you need and make sure you make it an OpMode. If you do not know how to do this, see **“1.4 First time setting up an OpMode.”**
    - After that, telemetry all of the values you want to use. For example, if you’re testing a distance sensor, you want to telemetry out the distance values and measure how accurate they are with a ruler.
      * Make sure that the sensor works the way you expect, and if it doesn’t, change your code!

**10.3 MR Gyro Sensor**

* The MR Gyro is a sensor that senses the rate of rotation in the x, y, and z axes and returns a heading value based off of changes in the z axis. In short, it measures how much you have turned.
* Here is a very useful guide to using the gyro sensor: <https://modernroboticsinc.com/Content/Images/uploaded/Sensors/Modern_Robotics_Gryo_Sensor-Steering_Tutorial.pdf>
* Make sure that you are giving the sensor enough calibration time. This means the robot should not be moving while the gyro is attempting to calibrate or your angles will be wrong.
* For turning, you often have to use loops.
  + If you get stuck in stop (refer to “**2.1 Phone errors”**), then add an extra condition so that it only runs while (opModeIsActive() == true).

**10.4 Rev IMU Sensor**

* The Rev IMU is also similar to the MR Gyro in that it also measures the angle of your turn. It is built into your expansion hub. All you need to do is to leave the I2C 0 port open and name the IMU there. Don’t put anything else in the 12C0 port or use it for anything other than the IMU

**10.5 MR Range Sensor**

* The MR Range is a distance sensor that uses both ultrasonic and light to figure out distances more accurately. Ultrasonic is better for longer distances and light is more accurate up close so the range sensor combines the best of both worlds.
* Something to keep in mind is that you have to constantly update the distance. If you’re using a loop, make sure that the distance is updated every time. You wouldn’t want to measure the distance once and keep comparing to that value. (This is a mistake that has been made before.)
* The FTC SDK also has a great sample code that teaches you how to use the range sensor.

**10.6 Rev Color Distance Sensor**

* If you are trying to use this sensor for distance → don’t
  + It’s a very good Color Sensor, but not a very good distance sensor
  + Stick with MR Range for distance
  + If you do try to use it for distance, it won’t give very accurate values

**10.7 Touch Sensor**

* There’s a button on the touch sensor and touch sensor knows when button pressed: it’s pretty simple
* Using the touch sensor:
  + You can use a Boolean (touch sensor is true or touch sensor is false) → because the sensor is either activated or it’s not
* Some possible issues:
  + Make sure the touch sensor is *actually* being pressed when it touches something. You might need to have it pressed harder.
  + You might have to move the sensor around to find the perfect position

Thanks for reading the software guide. Remember that if none of the solutions work sometimes the best software guides are the people around you!