FTC Programming Workshop

OVERVIEW

BY ERROR 404: TEAM NAME NOT FOUND

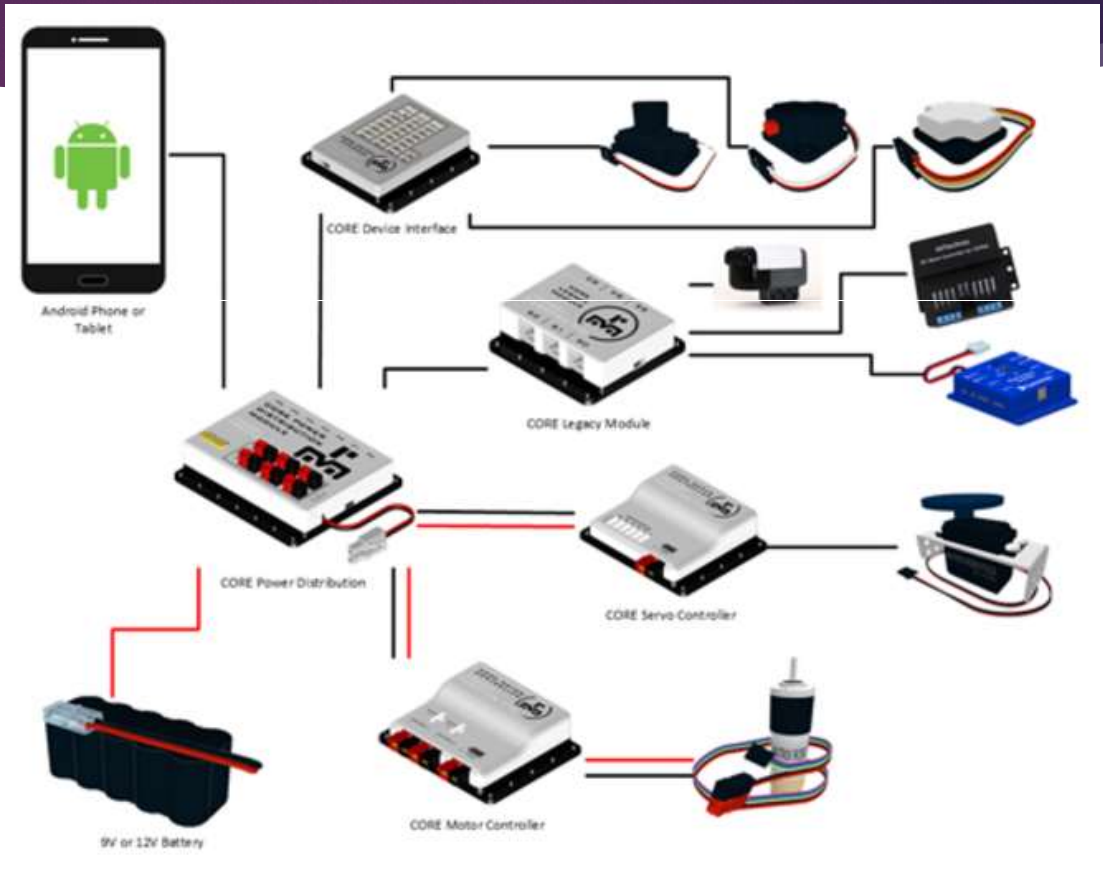
Definitions

- ***FTC "Next gen platform"***
  - - The next gen platform uses a tablet and a smartphone to control a robot. The tablet and smartphone are actually sophisticated, compact computers. These handheld computers have a central processor, volatile and non-volatile memory, input/output devices, and other features commonly found on modern computers.

- ***Android*** –
  - Android is the operating system that runs on these handheld devices. Similar to a laptop that has Microsoft Windows or MacOS as its operating system, a tablet or smartphone has its own operating system that manages the device's hardware and software components.
  - ZTE Speed Phones – FTC rules require 4.4+ (Kit-Kat) which comes already on the phone
  - Other phones: 6.0.1 (Marshmallow) or higher

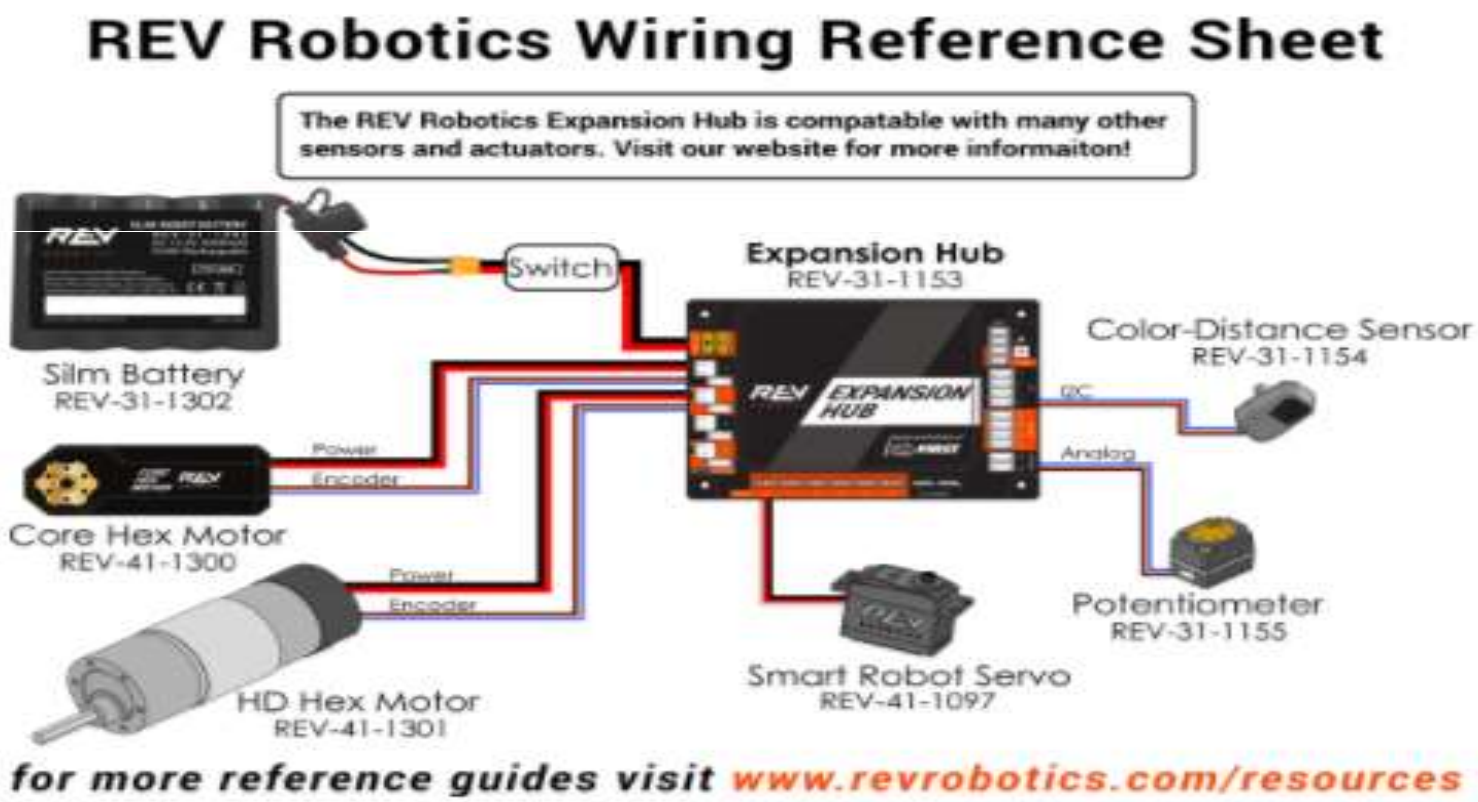
Two Choices on Robot Control System

Choice 1:
Control System from Modern Robotics



Two Choices on Robot Control System

Choice 2:
Control System from Rev Robotics



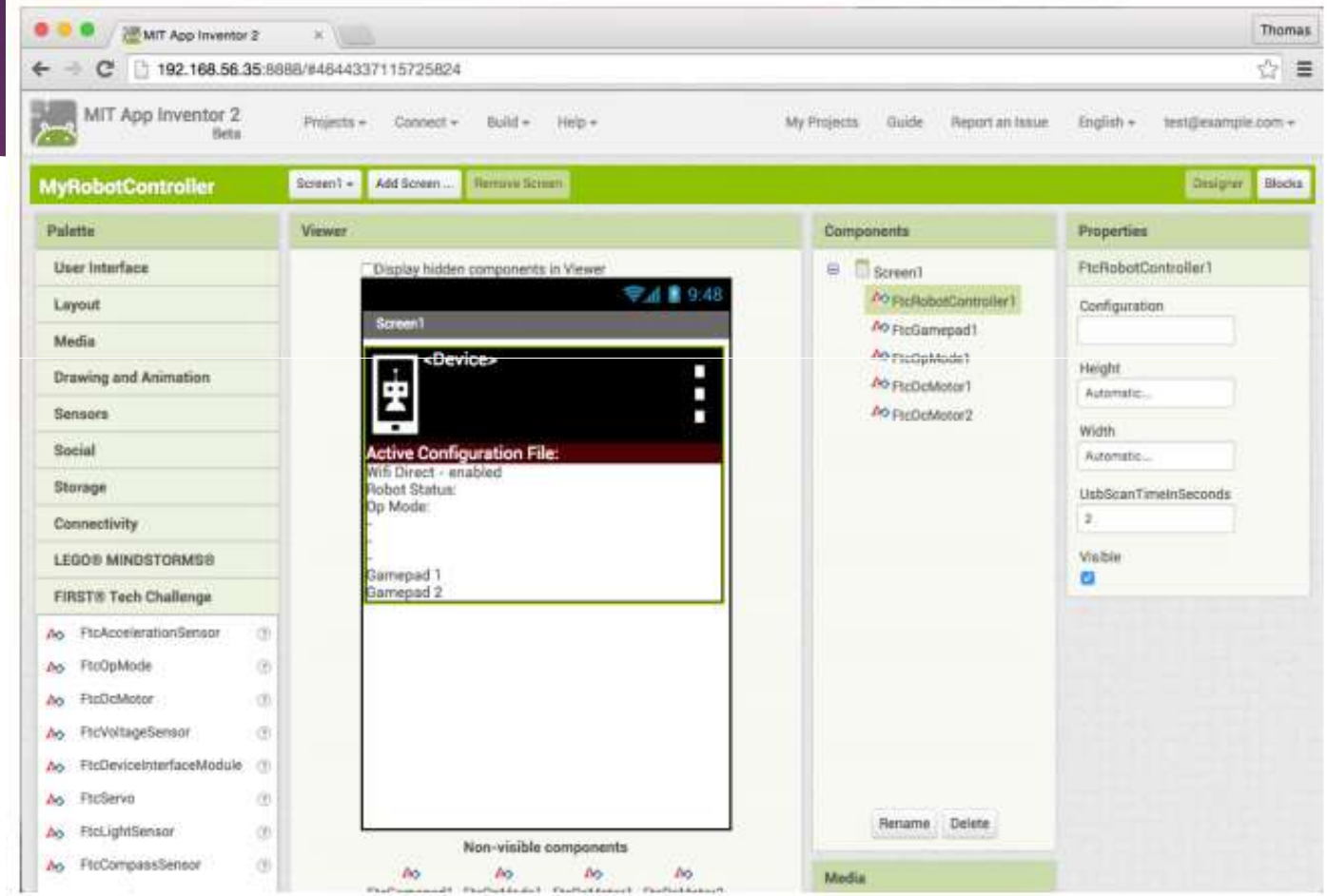
Programming Your Robot

- ***There are programming differences between the two control systems but mostly with sensors, not the basic drive functions***

- ***Four choices to program your robot***
  - Android Studio
  - MIT App Inventor
  - FTC Blocks Programming Development tool
  - Java Native Interface (JNI) and Android Native Development Kit (NDK)

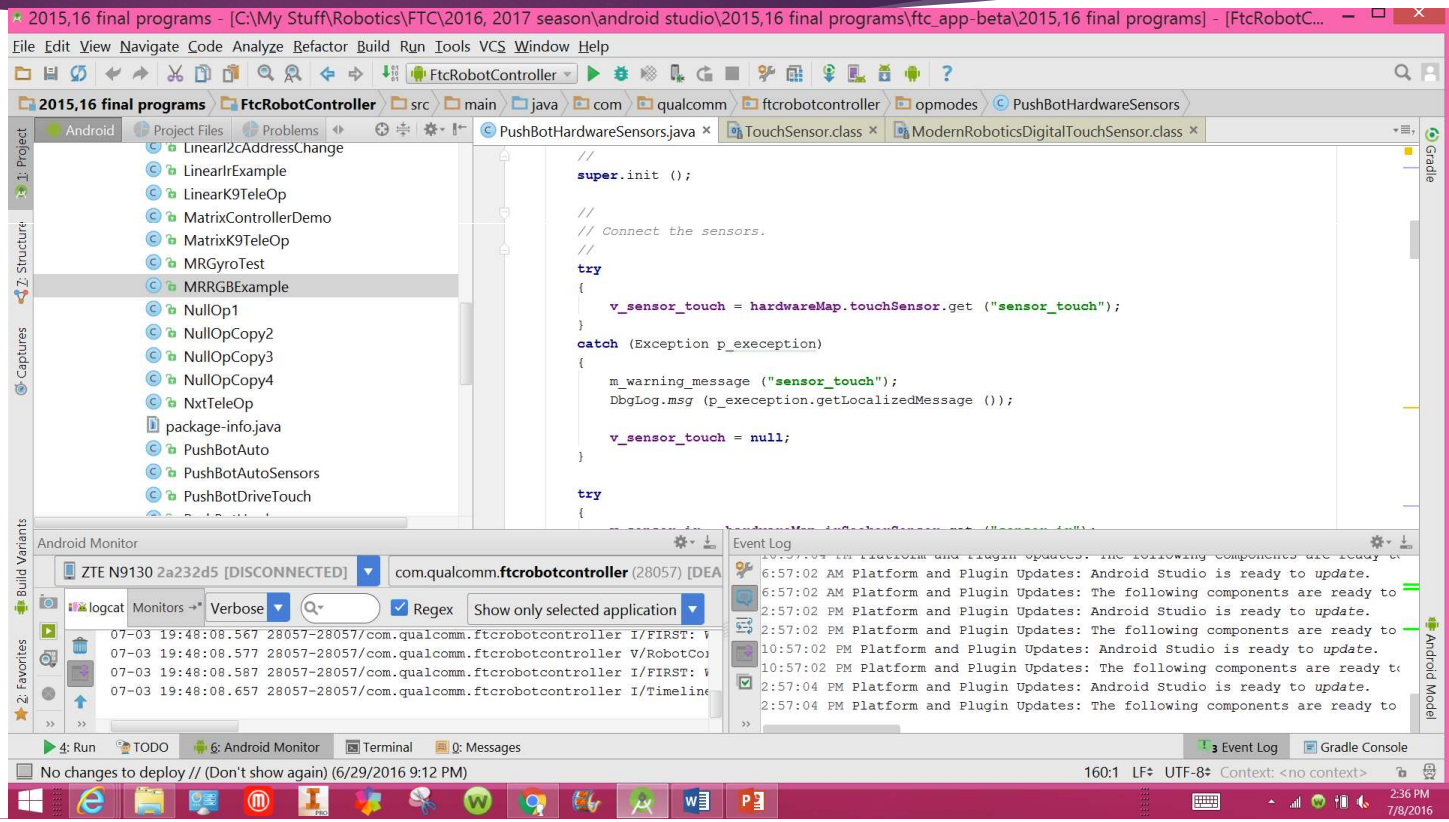
MIT App Inventor

- visual development environment
- Advantage: Java programming knowledge is not required.
- Disadvantage: More knowledgeable teams might find it too limiting.



Android Studio

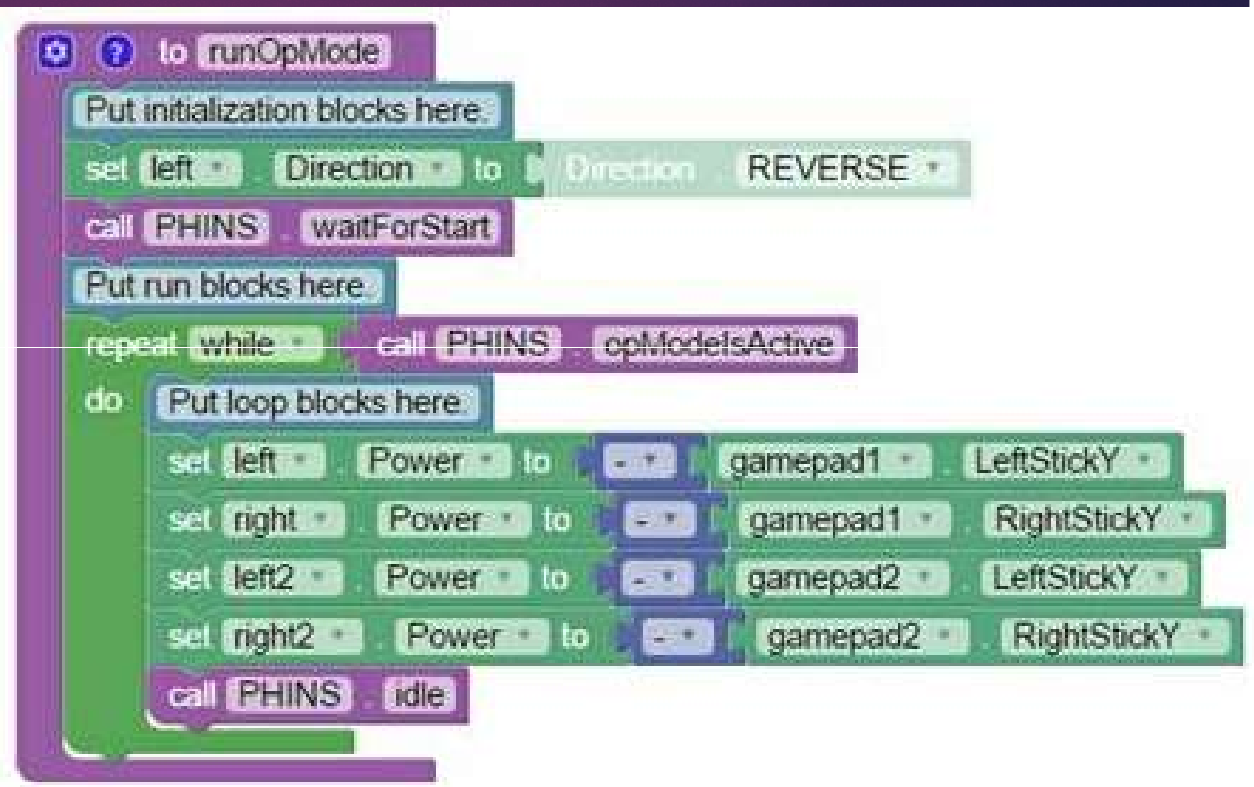
- "…the official Integrated Development Environment (IDE) for **Android** app development…. **Android Studio** is the fastest way to build high quality, performance apps for the Android platform, including phones and tablets …"
- Program directly in java language
- Advantage: more flexibility
- Disadvantage: Difficult to implement fully without familiarity with java



FTC Blocks

- Use Chromebook, laptop, or tablet to connect and download programs to robot controller phone.
- Similar to MIT App Inventor, but instead of downloading full app each time, just download individual OpModes

- Advantage: Easy to use
- Disadvantage: Appears you connect wirelessly, so it may not be allowed at competitions



4 motors example

Definitions

- ***SDK*** – Software Development Kit
  - - a set of tools that can be used to develop software applications. Include tools, libraries, documentation and sample code that would help a programmer to develop an application.
- ***JDK*** – Java Development Kit
  - "…forms an extended subset of a software development kit (SDK). It includes "tools for developing, debugging, and monitoring Java applications".
- ***API*** - **application programming interface** (**API**)
  - "…is a list of all classes that are part of the **Java** development kit (JDK). It includes all **Java** packages, classes, and interfaces, along with their methods, fields, and constructors. These prewritten classes provide a tremendous amount of functionality to a programmer."
- ***Gradle*** –
  - "…a tool used to help automate the build process for a complex piece of software. Android Studio uses Gradle to keep track of all the program module dependencies and to build and link these modules to create your app.

Definitions

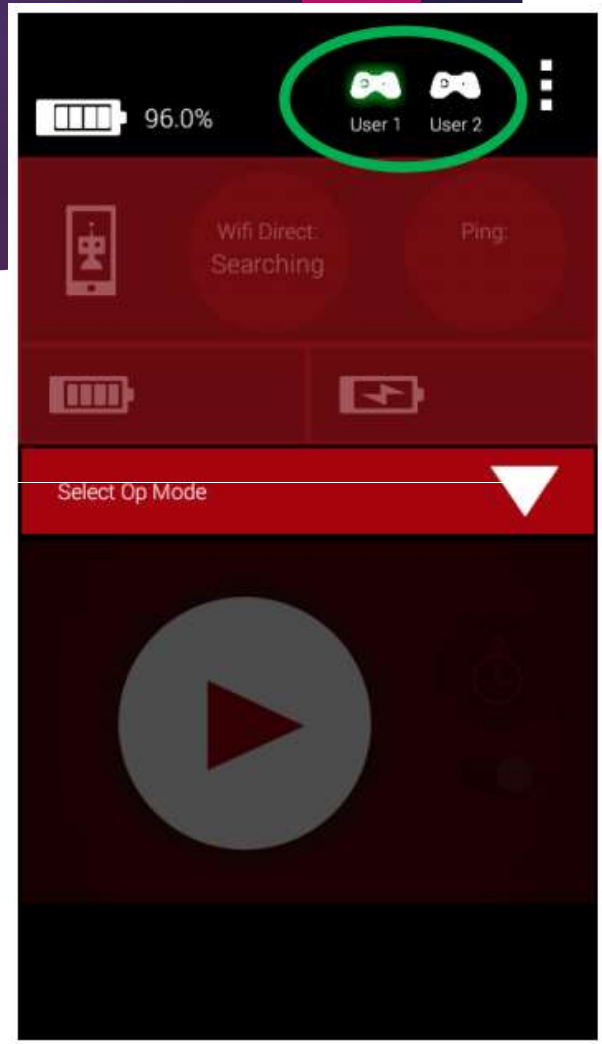
- ***Configuration file***
  - Provides a translation between meaningful names you give a device (ie: L_motor_rear, L_motor_front,…) to names that are meaningful to the robot (ie: serial #AH5600GBX port 1).
  - The device names you use in your program MUST match the names in your config file.
  - Note: It is helpful to use a sharpie to write the device serial number on each device (ie: motor controller, servo controller,…) because when you receive USB connection errors, it will be easier to track down the problem device.
- ***GitHub***
  - a cloud based system for collaboration and program sharing. This is where FIRST uploads the latest updates of the FTC SDK to make it available to all teams.

Java terminology

- ***methods*** –a collection of statements that are grouped together to perform an operation. When you call the System.out.println() **method**, for example, the system actually executes several statements in order to display a message on the console.
- ***classes*** – A **class** can contain fields and methods to describe the behavior of an object
  - Example: class = dc motor
  - Method = setPower, setMode, setdirection, getPower, getMode, getDirection
- ***package***
  - "…is a technique for organizing **Java** classes into namespaces …… providing modular programming. …..can be stored in compressed files called JAR files, allowing classes to be downloaded faster as groups rather than individually.
  - Examples: hardware, Modern Robotics, OpModes,…..

Definitions

- ***Op Modes***
  - "Preprogrammed robot behaviors (ie: autonomous or driver controlled) that you can launch from the driver station."
  - "…you will .. modify parts of the Robot Controller app to create your own op modes that will run on your robot. You will use a development kit (ie: Android Studio) to create, edit and register your own op modes for your Robot Controller app."
  - Two kinds of Op Modes
    - OpMode
    - Linear Op Mode



Within an OpMode – 4 defined methods you can edit to control your robot behavior

- ***init()***
  - When the driver pushes the "Init" button on the touch screen), the init() method for the selected op mode gets triggered. If you have any initialization tasks that you would like to do for your robot, you can place the initialize code into the init()method's body, but note that this method is only executed once, and the robot hardware is updated ***after*** this method exits.
- ***start()***
  - When the driver pushes the "Start" button on the touch screen, the start() method for the selected op mode gets triggered. If you have any initialization tasks that you would like for your robot to run right before the loop, you can override the start()method by adding public void start() { } to your opmode, and defining your own code inside the braces .
- ***loop()***
  - When a driver pushes the Start button on the driver station, the code that is written in the op mode's loop() method will be executed regularly (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 method is where you will put the bulk of your robot code for an op mode.
  - Don't use wait or sleep commands within a loop() as it could cause program failures. Use ifs not whiles.
- ***stop()***
  - When the robot controller receives a stop command from the driver station or when an emergency stop() condition occurs, the code in the stop() method gets executed. If you have any cleaning up to do after an op mode run has been completed, this is the place to put the cleanup code for your op mode. Just like the start()method, this method does nothing by default, but you can override it's behavior by adding public void stop() { } to your opmode.

Excerpts from an example OpMode

(not a complete program)

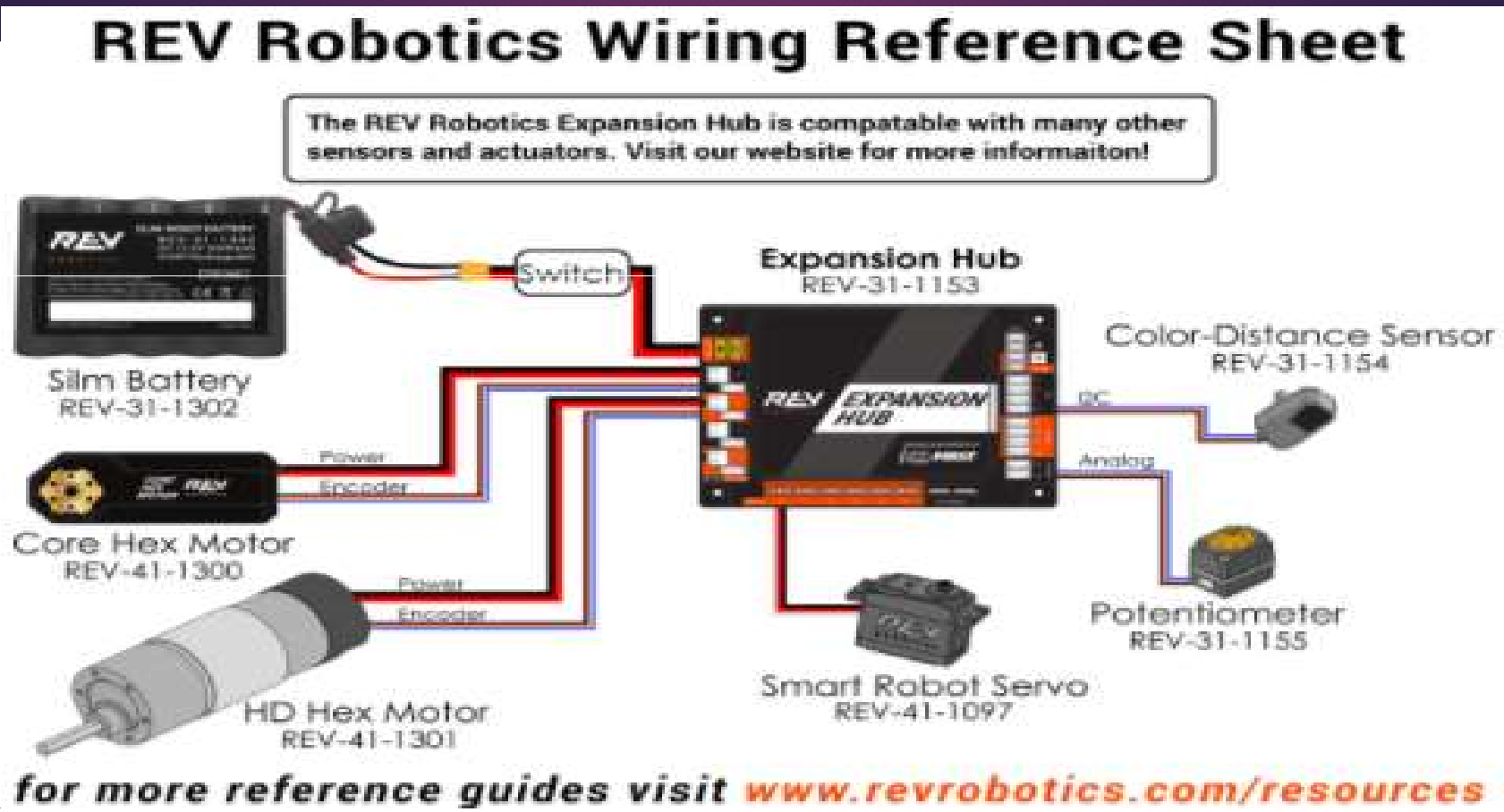
```
package com.qualcomm.ftcrobotcontroller.opmodes;
import …
@Autonomous(name="simpleExample", group="example")
public class SimpleExample extends OpMode {

(other stuff here…)
   @Override
   public void init() {

      motorLeft = hardwareMap.dcMotor.get("motor_1");
      motorLeft.setDirection(DcMotor.Direction.REVERSE);       }

   @Override
   public void loop() {
      motorLeft.setPower(.50);
      telemetry.addData("left tgt pwr",  "left  pwr: " + String.format("%.2f", left));  }

   @Override
   public void stop() {       }}
```

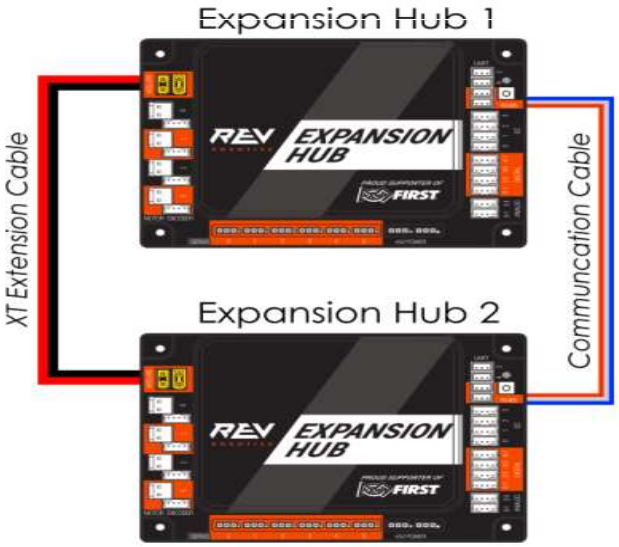
Excerpts from an example LinearOpMode – A customized OpMode

(not a complete program)

```
package com.qualcomm.ftcrobotcontroller.opmodes;
import …
(other stuff here…)
public class LinearSimpleExample extends LinearOpMode {

public void runOpMode() throws InterruptedException {
    motorLeft = hardwareMap.dcMotor.get("motor_1");
    motorLeft.setDirection(DcMotor.Direction.REVERSE);

    waitForStart();

    while (opModeIsActive()) {
      motorLeft.setPower(.50);
      telemetry.addData(" left motor", motorLeft.getPower());
      waitOneFullHardwareCycle();
    }
  }
}
```

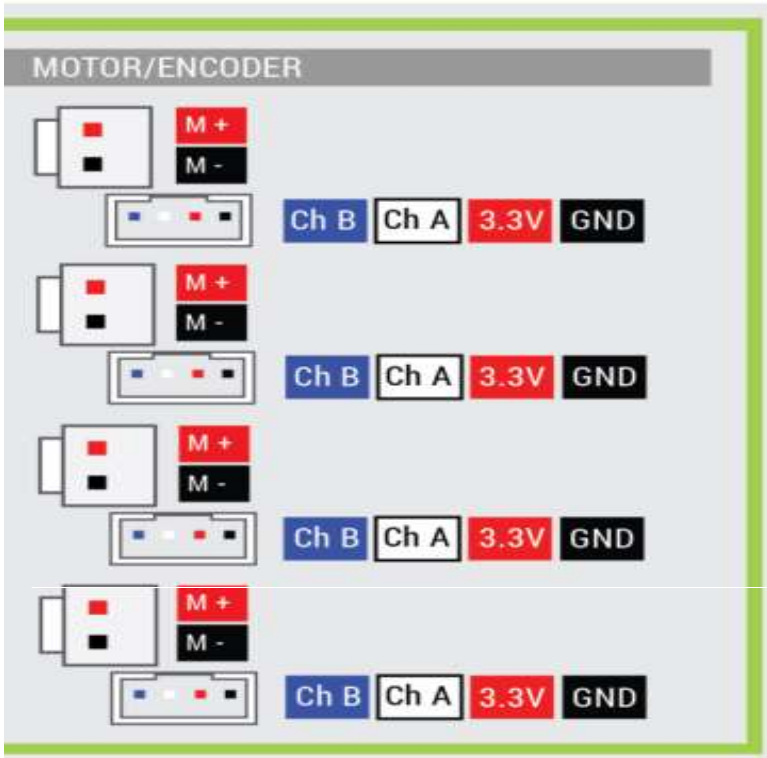
More Detail on Rev Robotics System

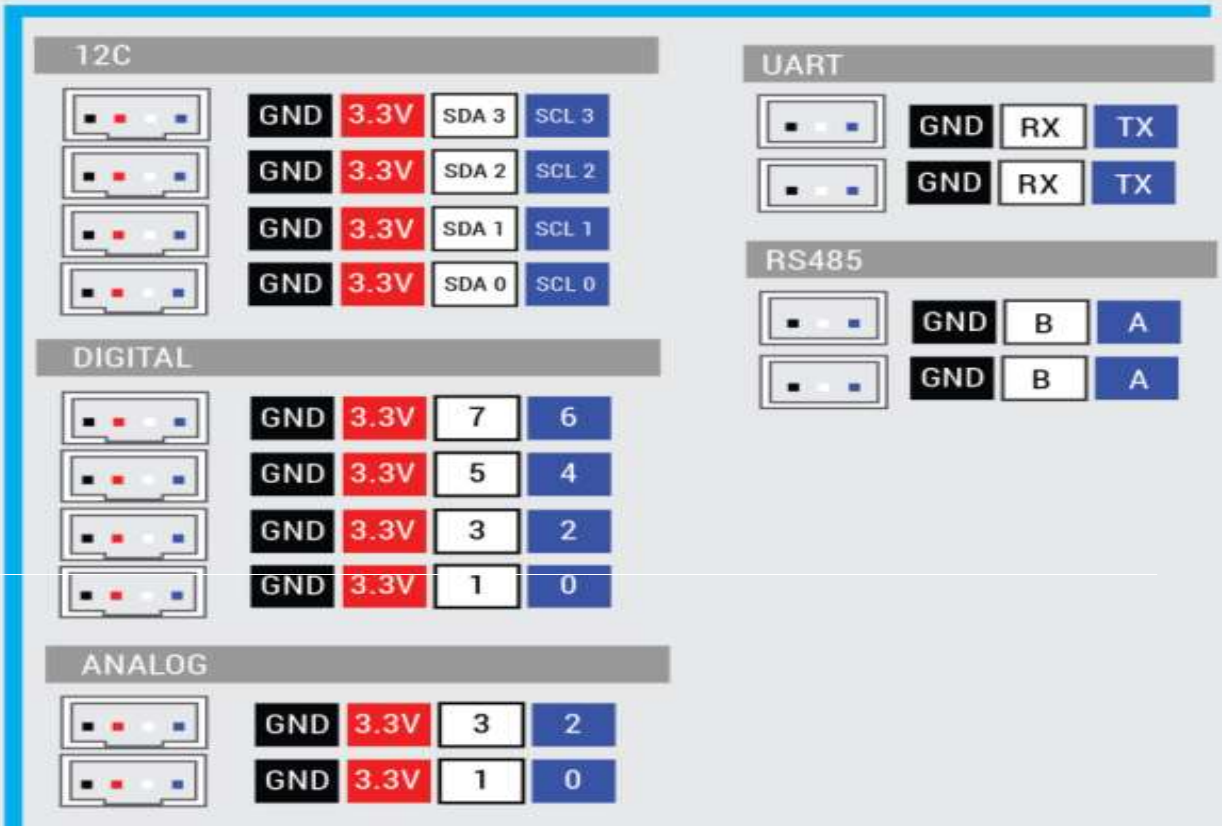




- Each hub replaces: CPDM, 2 Motor controllers, 1 servo controller, and 1 device interface module.
- FTC teams can have two hubs connected like this:

1.2 Port Pin Outs


MOTOR/ENCODER
M +
M -
Ch B
Ch A
3.3V
GND
M +
M -
Ch B
Ch A
3.3V
GND
M +
M -
Ch B
Ch A
3.3V
GND
M +
M -
Ch B
Ch A
3.3V
GND


12C
GND 3.3V SDA 3 SCL 3
GND 3.3V SDA 2 SCL 2
GND 3.3V SDA 1 SCL 1
GND 3.3V SDA 0 SCL 0
DIGITAL
GND 3.3V 7 6
GND 3.3V 5 4
GND 3.3V 3 2
GND 3.3V 1 0
ANALOG
GND 3.3V 3 2
GND 3.3V 1 0
UART
GND RX TX
GND RX TX
RS485
GND B A
GND B A

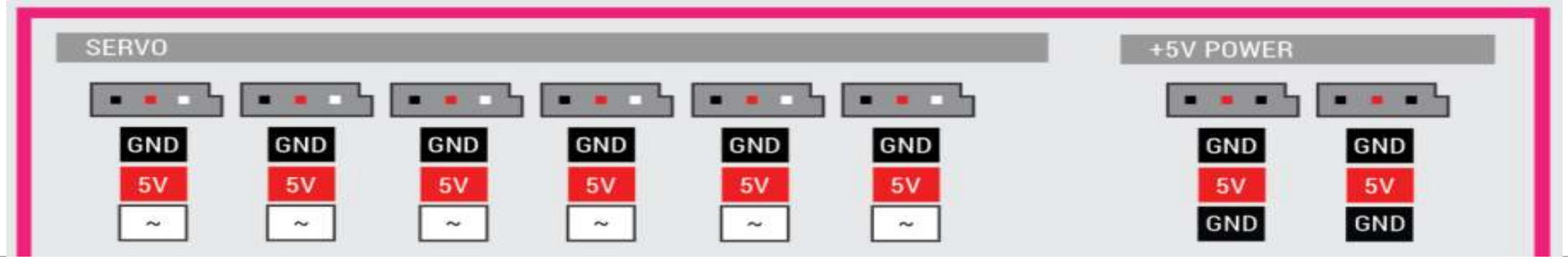

SERVO
GND 5V ~
GND 5V ~
GND 5V ~
GND 5V ~
GND 5V ~
GND 5V ~
+5V POWER
GND 5V GND
GND 5V GND

Figure 5: 3.3V to 5V Level Shifter Board

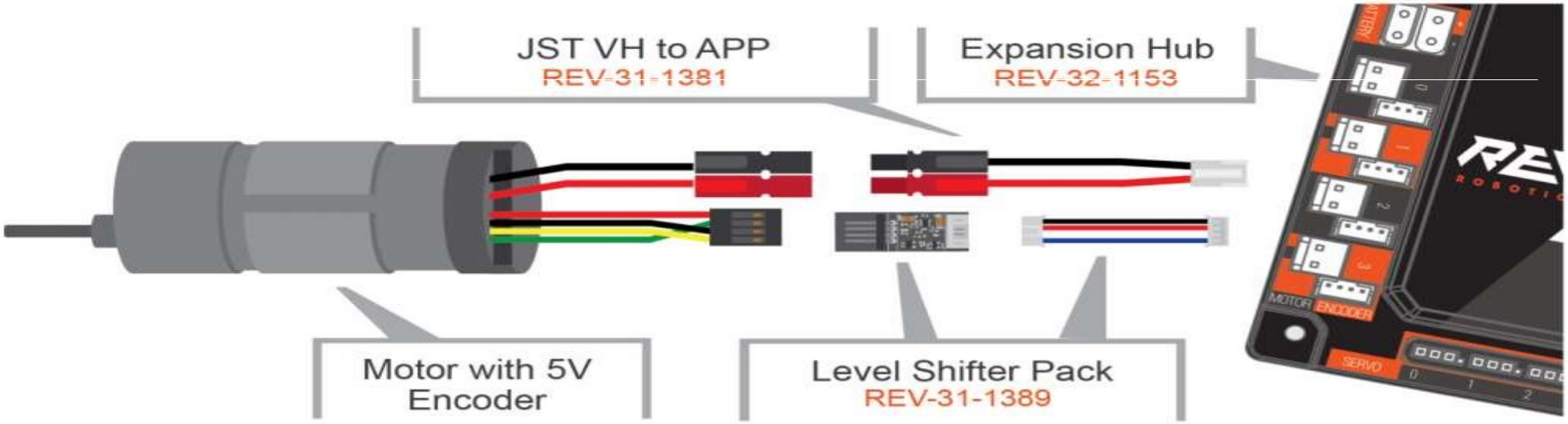
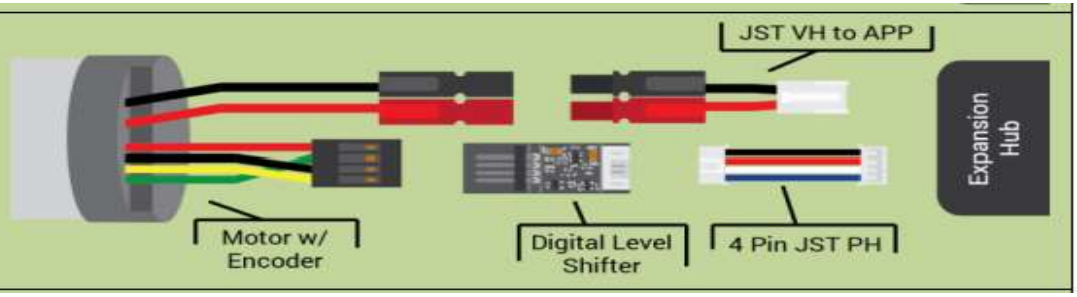


Figure 6: Connecting a Motor with 5V Encoder to an Expansion Hub

Figure 7: Connecting a Legacy 5V Sensor to an Expansion Hub

Sensor	Type	Compatible	Adapters Needed
Color Sensor 45-2018 Modern Robotics	I2C	Yes	
Optical Distance Sensor 45-2006 Modern Robotics	Analog	No	Not Officially Supported
Touch Sensor 45-2007 Modern Robotics	Analog	Yes	No Adapter Needed NOTE: Mechanical adapting IS required.
NeveRest Motor AM-3461, AM-3102, AM-2964a, AM-3103, AM-3104 AndyMark	Quad Encoder	Yes	

Logic level shifter
- FTC Store kit comes with 3
- $3/ea on Rev Robotics store

Splitter cable
- FTC Store kit does NOT give you any of these
- $3 for a 2 pack on Rev Robotics Store