Intro to Android Studio

FTC 7129

Presenters

Knut Peterson, Project Manager, Robo Raiders FTC 7129

Jamari Morrison, Programming Lead, Robo Raiders FTC 7129

Learning Goals

- **How component mapping works**
- **Combining flow charting with programming**
- **Working with phone configuration files**

Schedule

FTC 7129

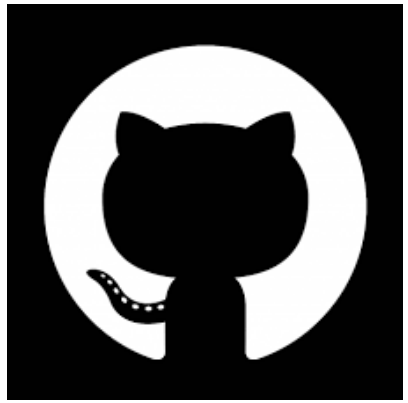
- **Programming Components – What will we be coding against?**
- **Flow Charting – Programming with a plan**
- **File Registry and Configuration Files – Telling the phone what's there**
- **Lessons Learned – Avoiding mistakes to save your time and sanity**
- **Questions**

Programming Components

FTC 7129

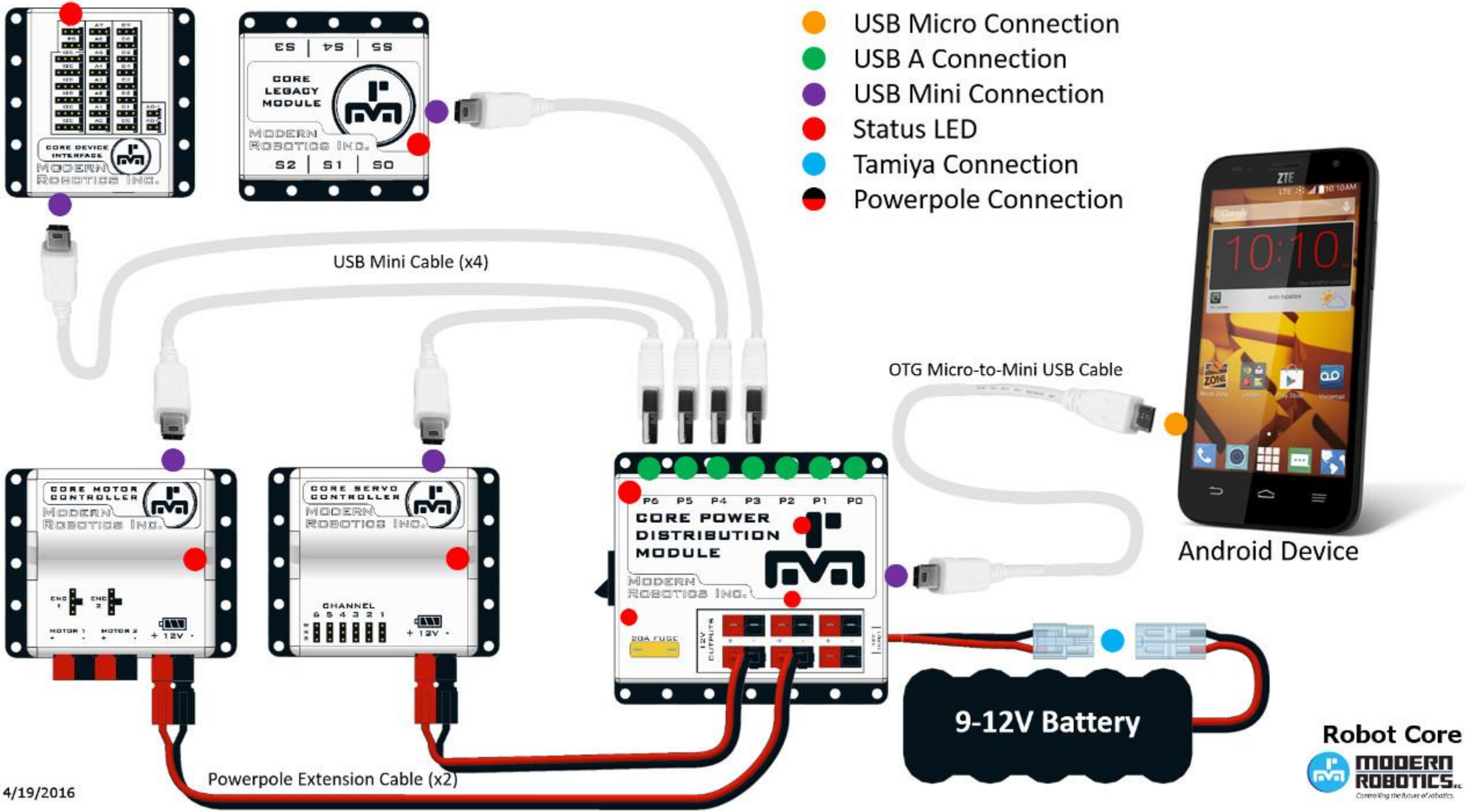
Software

- **Coding – Android Studio**
- **File Sharing and Syncing - GitHub**
- **SDK – ftc_app**



Hardware

- **Motor, Servo, and Sensor Controllers**
- **Core power distribution module**
- **Phone**
- **Controllers**


USB Micro Connection
USB A Connection
USB Mini Connection
Status LED
Tamiya Connection
Powerpole Connection
CORE DEVICE INTERFACE
MODERN ROBOTICS INC.
CORE LEGACY MODULE
MODERN ROBOTICS INC.
S5 S4 S3
S2 S1 S0
USB Mini Cable (x4)
OTG Micro-to-Mini USB Cable
ZTE
10:10
Android Device
CORE MOTOR CONTROLLER
MODERN ROBOTICS INC.
ENC 1
ENC 2
MOTOR 1
MOTOR 2
+ 12V -
CORE SERVO CONTROLLER
MODERN ROBOTICS INC.
CHANNEL
6 5 4 3 2 1
+ 12V -
P6 P5 P4 P3 P2 P1 P0
CORE POWER DISTRIBUTION MODULE
MODERN ROBOTICS INC.
20A FUSE
12V OUTPUTS
9-12V Battery
Powerpole Extension Cable (x2)
4/19/2016
Robot Core
MODERN ROBOTICS
Controlling the future of robotics

Hardware Components

FTC 7129

- **Hand held controllers –** Give input to driver station phone.

- **Driver Station Phone –** Takes input from hand held controllers and sends that information to the robot controller phone.

- **Robot Controller Phone –** Provides code for the robot to run.

Hardware Components

FTC 7129

- **Battery** – Gives power to robot
- **Core Power Distribution Module**
  - Splits information taken from phone and give to respective modules
  - Distributes power from battery to the modules

Hardware Components

FTC 7129

- **Motor –** Used to move robot components with unlimited rotation

- **Motor Controller –** Transfers power and code from core PDM to motors

Hardware Components

- **Servo Controller –** Transfers power and code from core PDM to servos.

- **Servo –** Less powerful motor
  - Different varieties and ranges of motion
  - Only able to be moved to a set position and cannot have their power altered

Hardware Components

- **Device Interface Module –** Gives power to and takes information from sensors.
- **Gyro Sensor –** Reports sensors' x, y and z coordinates.
- **Color Sensor –** Reports what is being reflected to the sensor.
- **Touch Sensor –** Reports if a sensor button is or is not pressed.

Android Studio File Registry

FTC 7129

Registering Android Programs

- FtcOpModeRegister tells the phone what files to download. Files must be registered in this file for them to be downloaded to your phone.

- Register OpModes with this line of code:

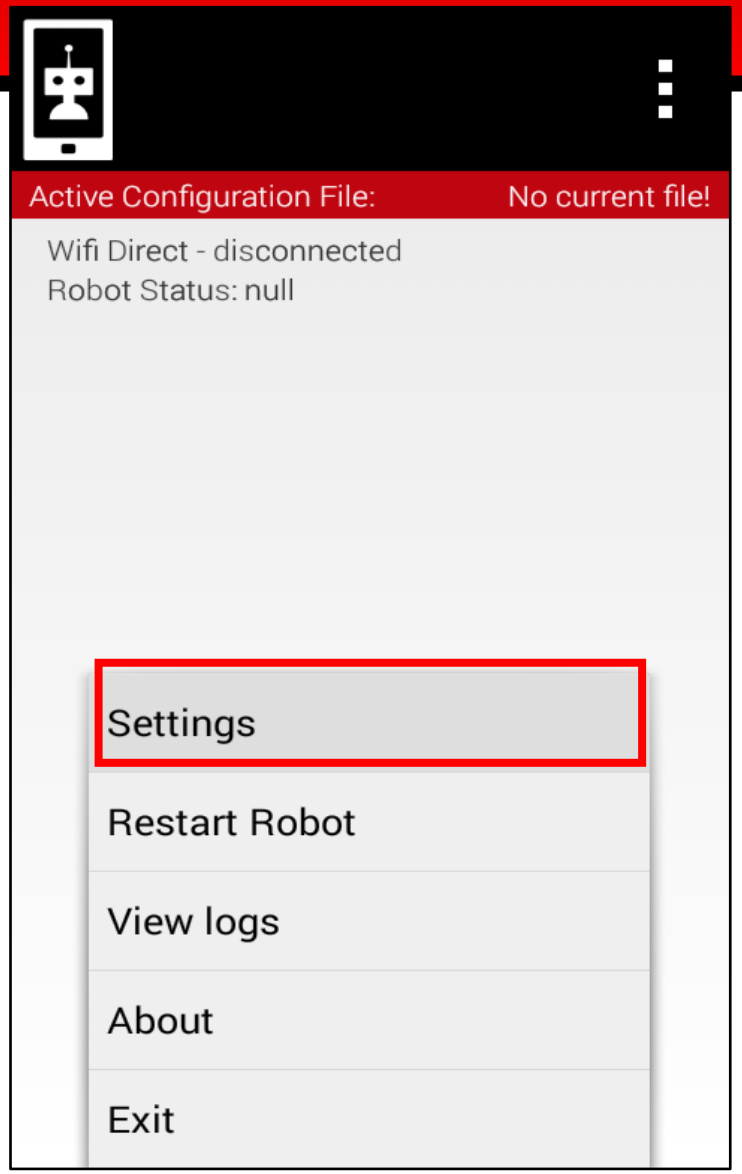
```
manager.register( “prettyGoodProgram“ , prettyGoodProgram.class);
```

Phone Configuration Files

FTC 7129

Creating a configuration file

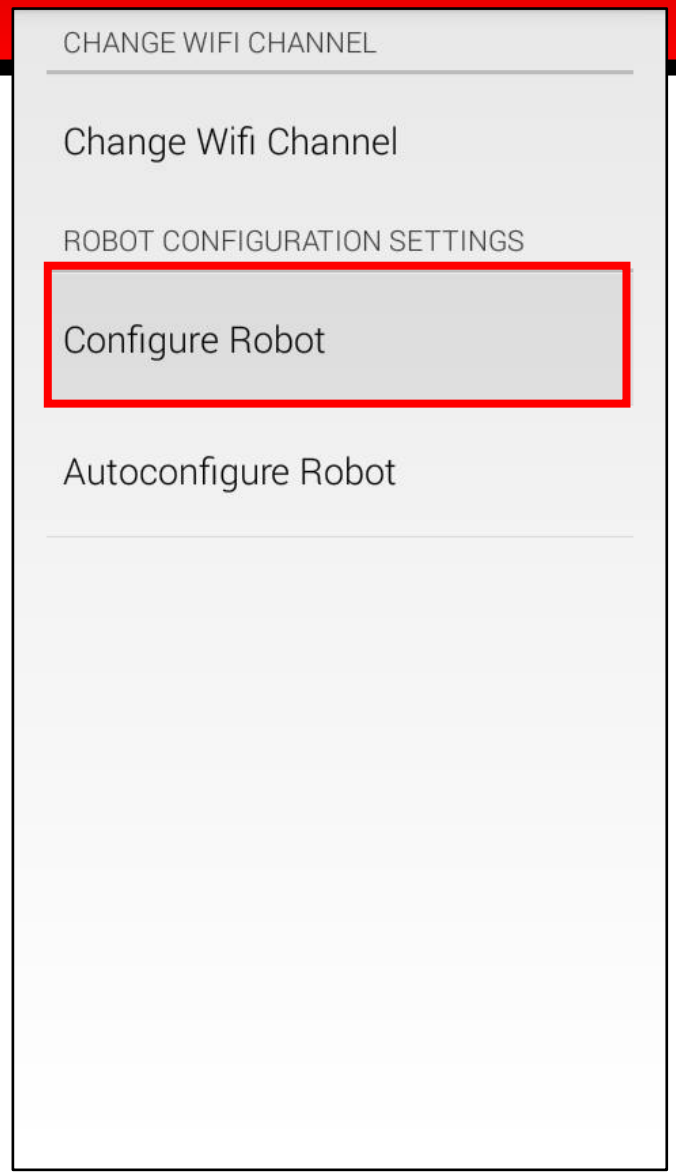
- Open the robot controller app, go into settings.

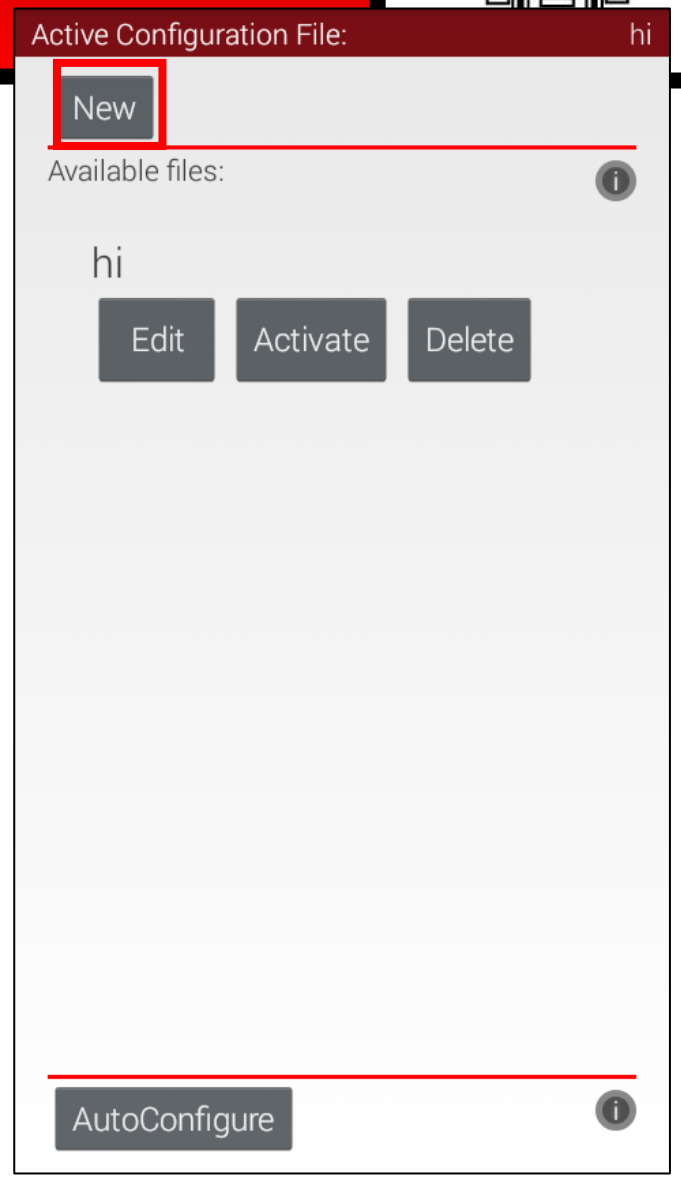


Phone Configuration Files

Creating a configuration file

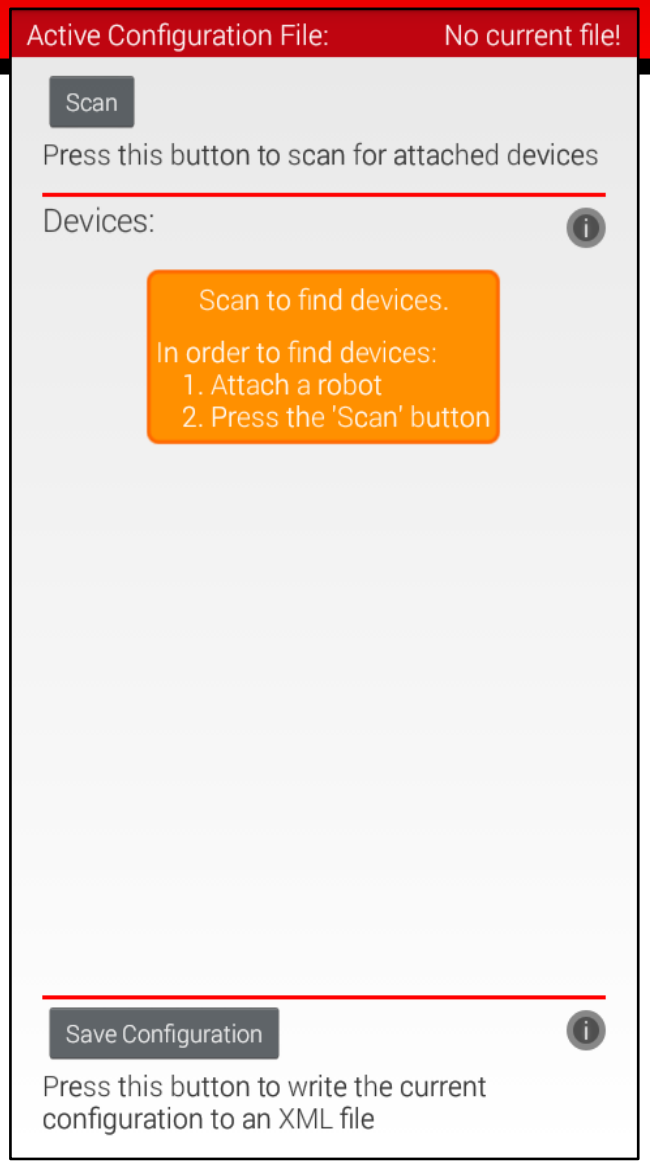
- Select “Configure Robot” inside of the settings menu.
- Press the “New” button inside of the configuration file menu.

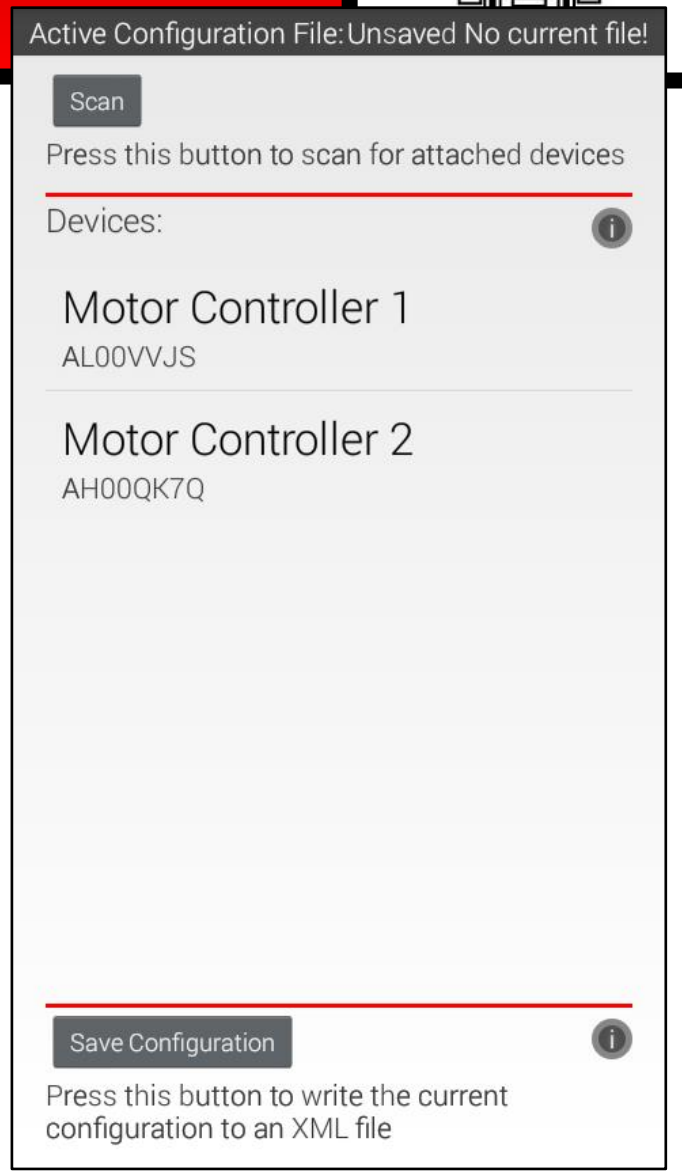




Creating a Configuration File

- Inside of this new file, hook the phone up to your robot, turn the core PDM's power on, and press scan.
- After scanning, select a controller which has a motor sensor or servo which is being used in your code.

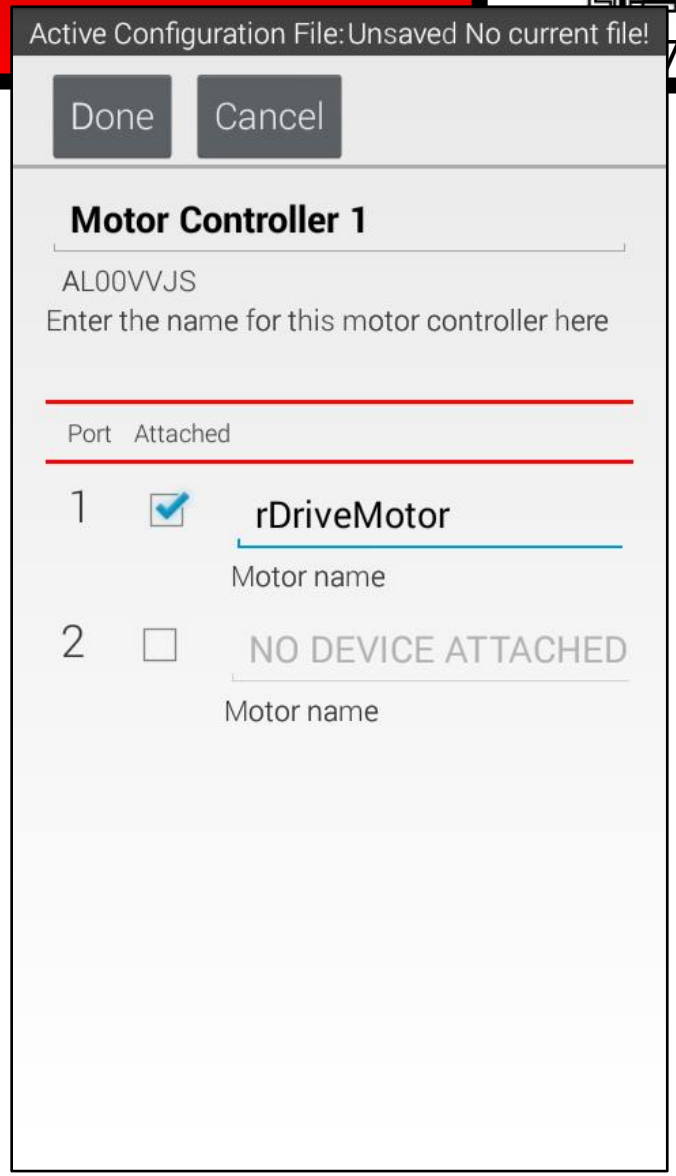




Creating a Configuration File

- Once the controller is selected check the “Attached” box and name the motor as it is called in your code.

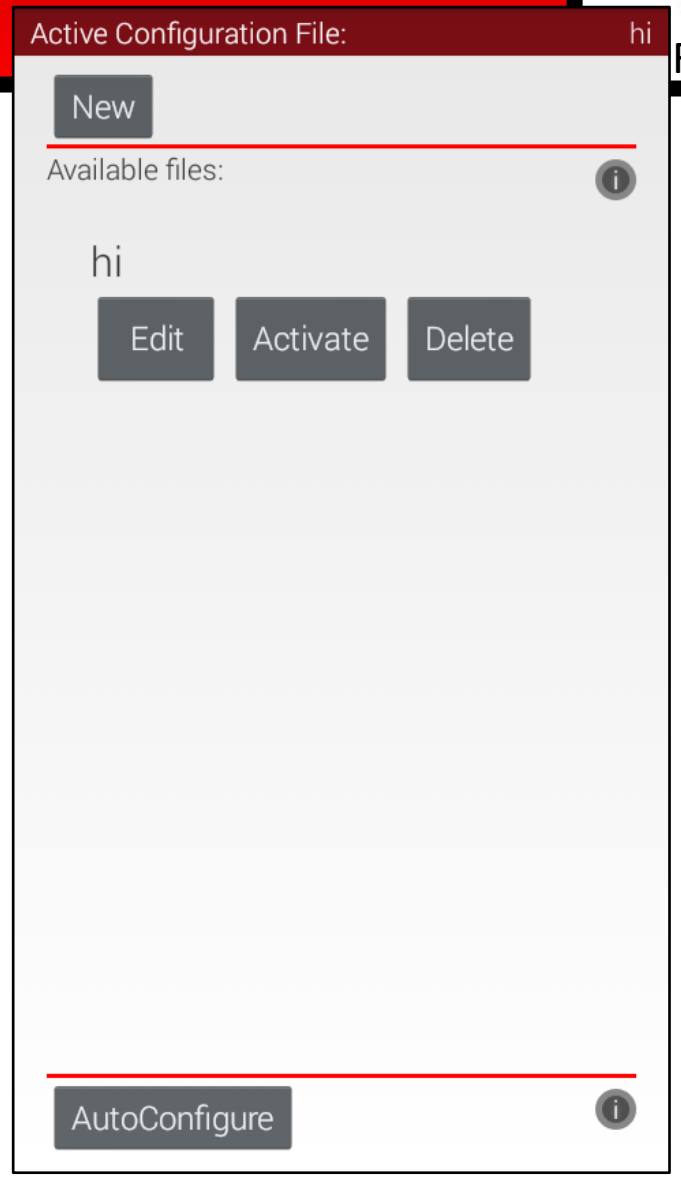
```
rDriveMotor = hardwareMap.dcMotor.get( “rDriveMotor“);
```



Modifying Configuration Files

- Configuration files will be changed often; whenever motors, servos, and sensors are renamed, added or removed from the program.

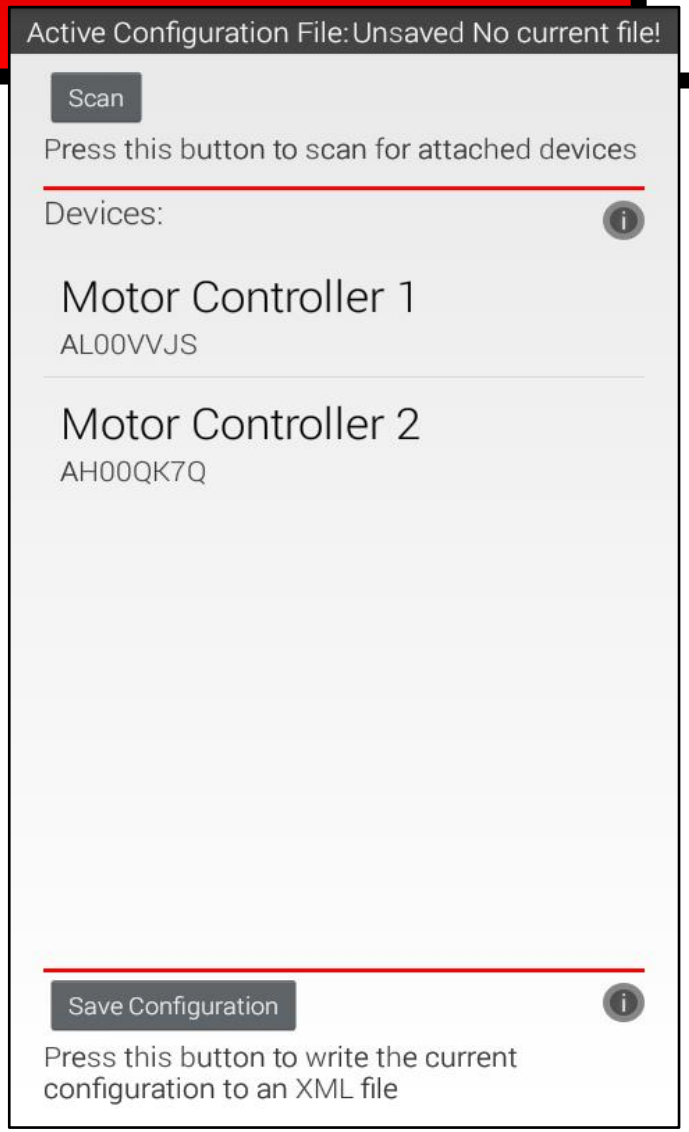
- Open a previously made configuration file by selecting Edit under the file name.

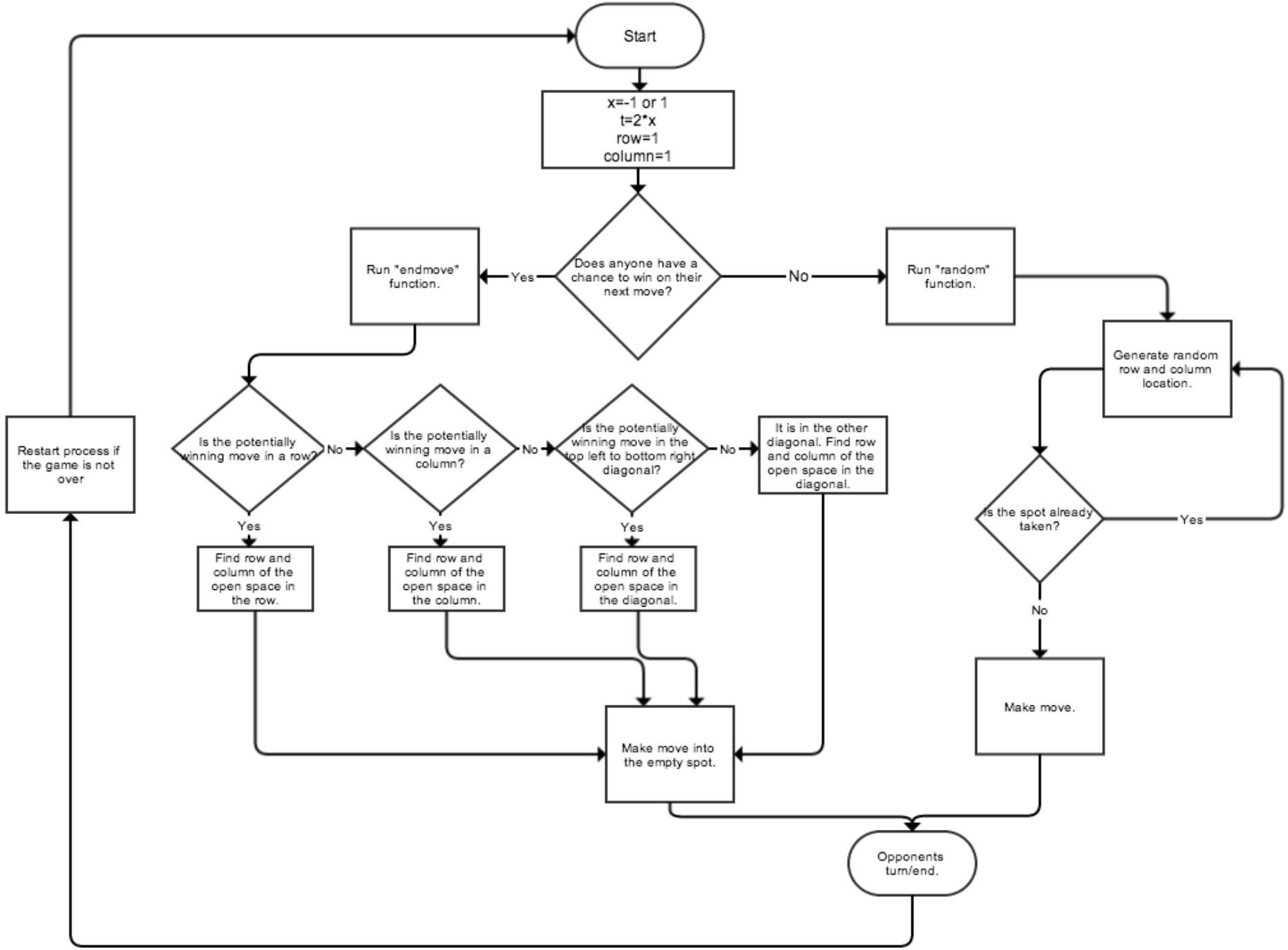


Configuration File Troubleshooting

FTC 7129

- **Scanning will delete the contents of the file (Servo, motor, and sensor names) and replace them with blank modules.**
- Select the controller which needs to be edited.




Start
x=-1 or 1
t=2*x
row=1
column=1
Does anyone have a chance to win on their next move?
Yes
Run "endmove" function.
No
Run "random" function.
Generate random row and column location.
Is the spot already taken?
Yes
No
Make move.
Is the potentially winning move in a row?
No
Yes
Find row and column of the open space in the row.
Is the potentially winning move in a column?
No
Yes
Find row and column of the open space in the column.
Is the potentially winning move in the top left to bottom right diagonal?
No
Yes
Find row and column of the open space in the diagonal.
It is in the other diagonal. Find row and column of the open space in the diagonal.
Make move into the empty spot.
Opponents turn/end.
Restart process if the game is not over

Flowcharting

FTC 7129

- What is a flowchart?
  - Diagram that maps out a process
- Why Use flowcharts?
  - Work through a design before building
  - Model programming logic
  - Analyze processes
  - Communicates high level information abstracted from designs/programs

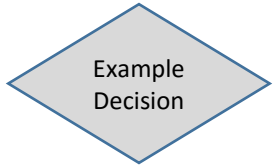
Flowcharting Symbols (Basic)

FTC FIRST® Tech Challenge

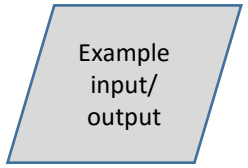
ROBORAIDERS FTC 7129

- Ovals signify the start or end or a process

- Rectangles show instructions or actions



- Diamonds highlight where you must make a decision



- Parallelograms show input and output. For us that often means the use of sensors

- Arrows show relationships between the other shapes

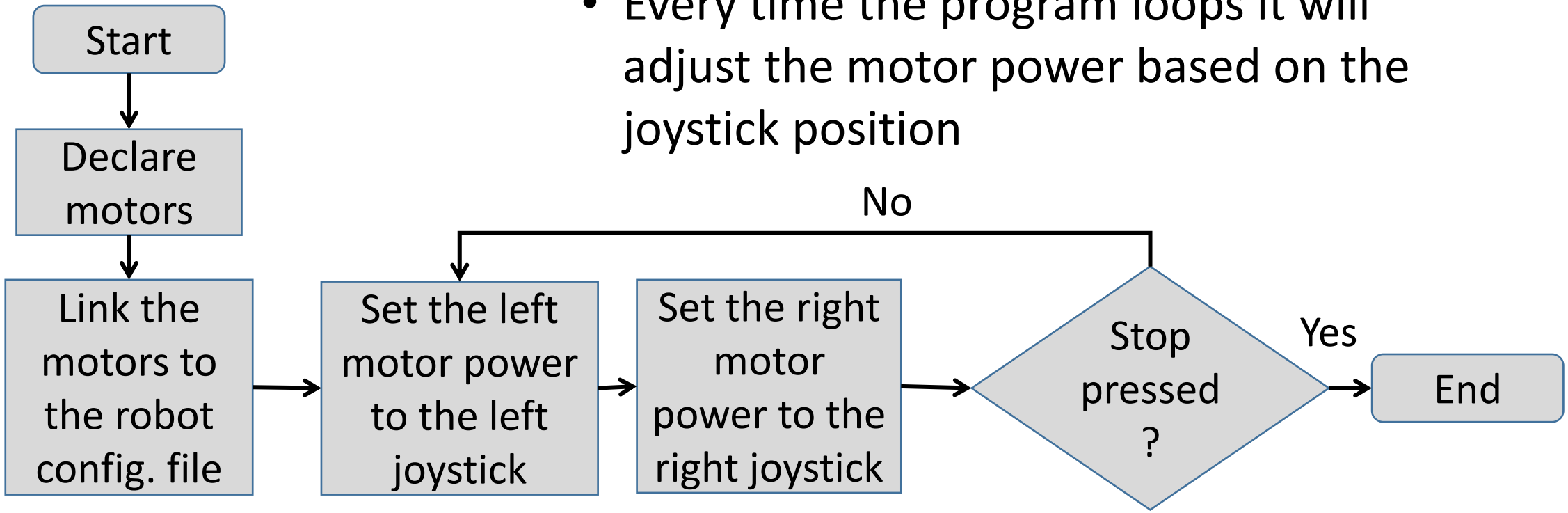
Flowcharting Best Practices

FTC 7129

- Rough sketch on paper – much faster
- In software tools or drawings – use standard symbols
- Name decision blocks, processes, and arrows
- Layout consistency is important
  - Direction: Left -> Right; Top -> Bottom
  - Symbol Sizing
  - Spacing

Flowchart Example

- Basic Tele-op (remote control) program
  - Every time the program loops it will adjust the motor power based on the joystick position



```
package com.qualcomm.ftcrobotcontroller.opmodes;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.DcMotor;

public class Teleop extends OpMode{

   DcMotor LeftDrive;
   DcMotor RightDrive;

   public void init(){
      LeftDrive = hardwareMap.dcMotor.get("LeftDrive");
      RightDrive = hardwareMap.dcMotor.get("RightDrive");
   }
   public void loop(){
      LeftDrive.setPower(gamepad1.left_stick_y);
      RightDrive.setPower(gamepad1.right_stick_y);
   }
}
```

- All Opmodes need to have both the “init” and “loop” sections.
- The “loop” section is especially handy for tele-op programs since we don’t have to create a loop – one is already there.
- For Autonomous, use a LinearOpMode – It doesn’t need a “loop” section

Lessons Learned

- Beware of outdated help files on the FTC Forums
- Make mentorship connections with other teams and professionals
- Look at example code to make your own
- Go through the example programs
- Flowcharts are your friends
- There are resources out there now – on our website, YouTube tutorials, etc.
- Take advantage of free Java tools to hone your skills
- Jump in! – I knew absolutely nothing a year ago

Resources

- **Check out our Website! http://roboraiders.net**
- **Android Studio and FTC SDK download/setup instructions**
  - Phone Update Management
  - Past Power-Points
  - Lots of other resources – more being added
  - These slides will be posted there

- **YouTube is now teeming with FTC programming tutorials**
- **Free Java Tutorials at Codecademy.com**