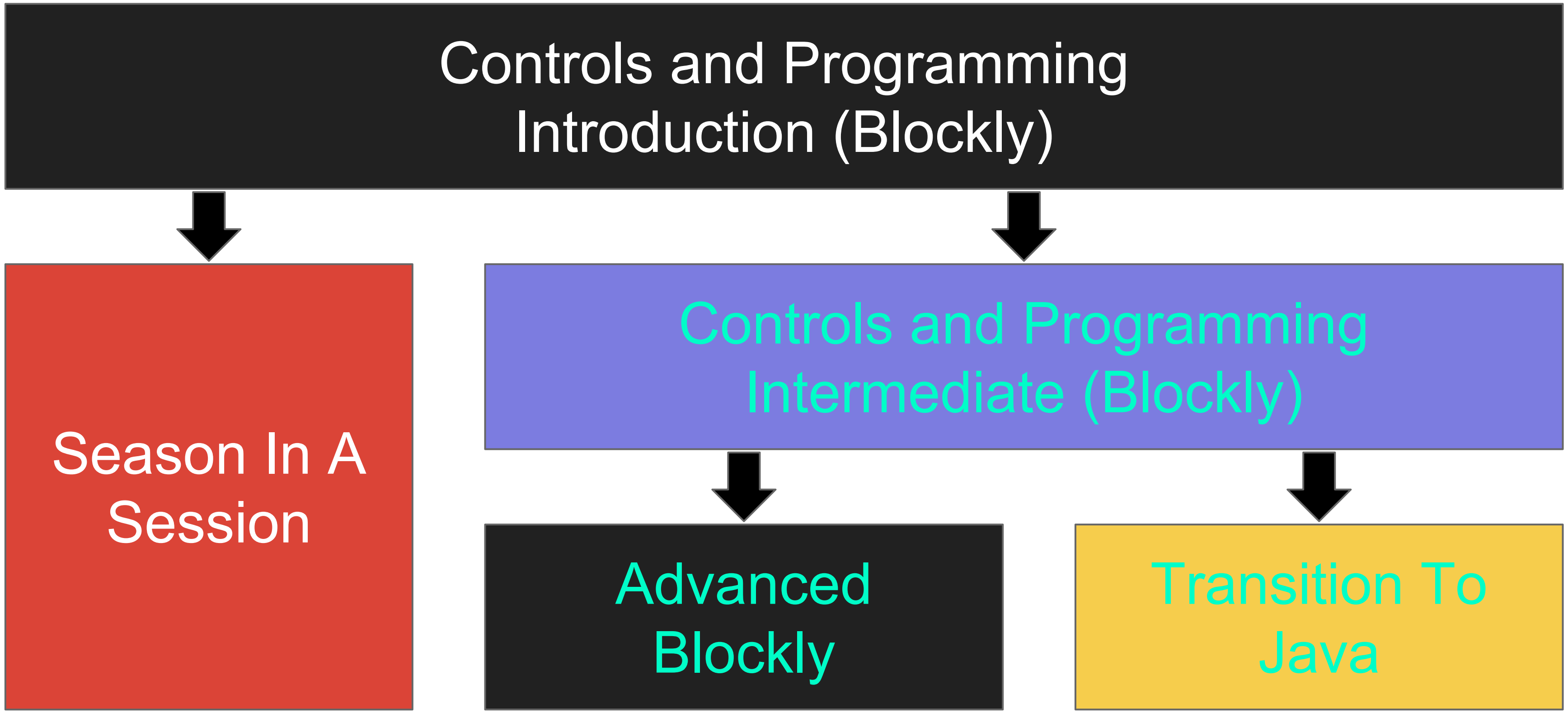
FTC Controls and Programming

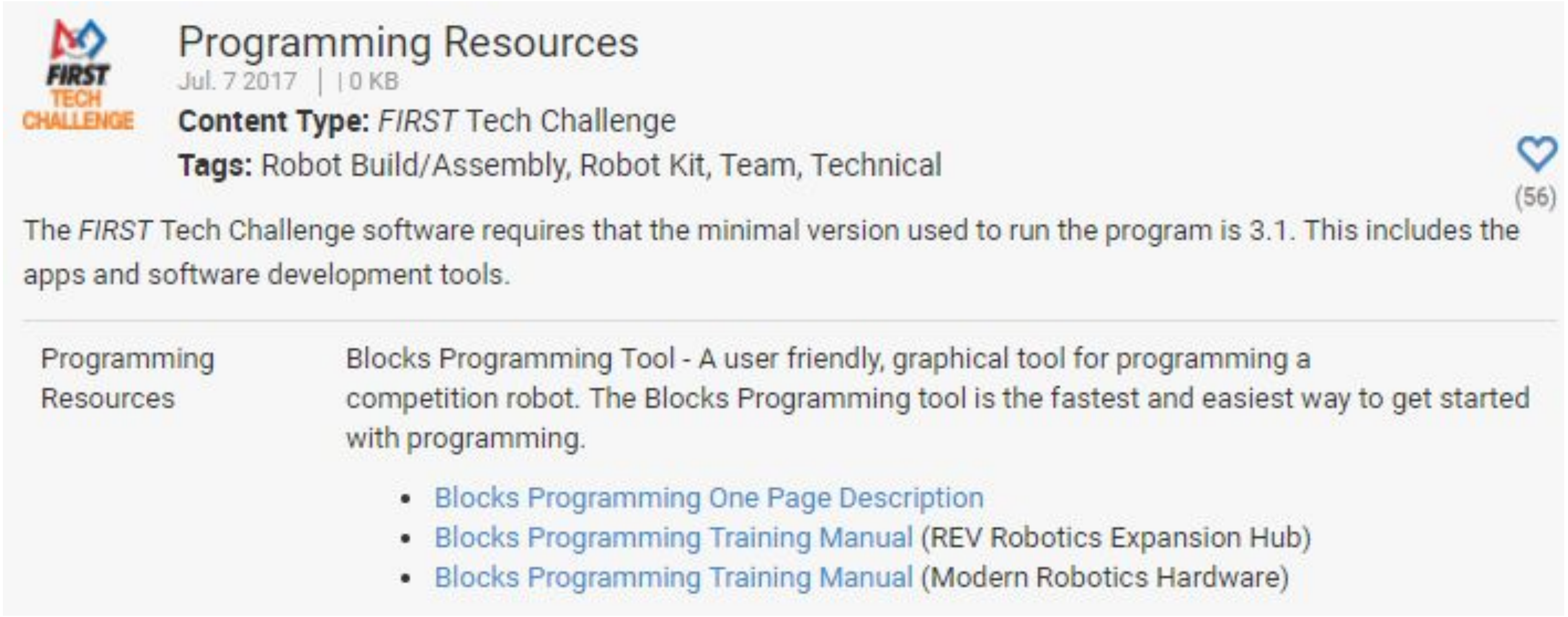
Jeff Ousley - Up Next!

Ken Pugsley and the Astromechs - Team 3409

Controls and Programming Sessions Today


Controls and Programming Introduction (Blockly)
Season In A Session
Controls and Programming Intermediate (Blockly)
Advanced Blockly
Transition To Java

https://www.firstinspires.org/resource-library/ftc/technology-information-and-resources



Programming Resources

Jul. 7 2017 | | 0 KB

Content Type: *FIRST* Tech Challenge

Tags: Robot Build/Assembly, Robot Kit, Team, Technical

(56)

The *FIRST* Tech Challenge software requires that the minimal version used to run the program is 3.1. This includes the apps and software development tools.

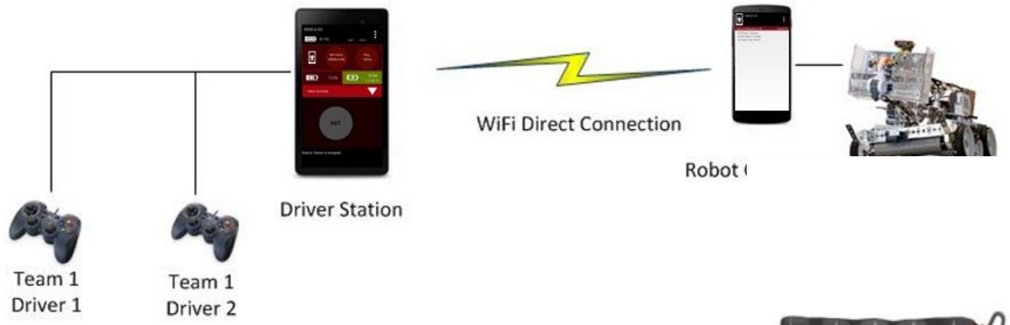
Programming Resources	Blocks Programming Tool - A user friendly, graphical tool for programming a competition robot. The Blocks Programming tool is the fastest and easiest way to get started with programming. • Blocks Programming One Page Description • Blocks Programming Training Manual (REV Robotics Expansion Hub) • Blocks Programming Training Manual (Modern Robotics Hardware)

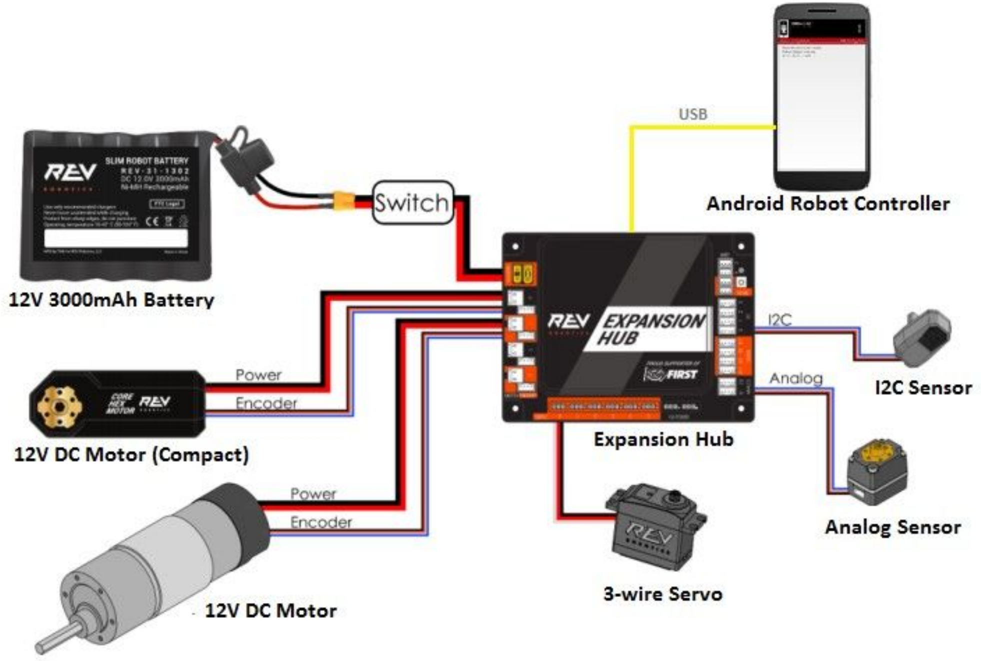
What We'll Cover Today

- Overall Theme - Hands On!
- Introduction
  - Overview and set-up electronics and phones
  - Basic Blockly teleop programming
- Intermediate
  - Autonomous and Vuforia
- Advanced (Blockly)
  - More Autonomous
  - Servos
- Advanced (Java)
  - Transition to Java
  - Android Studio vs OnBot

Introduction

Exercise 1 - Set Up Electronics and Phones

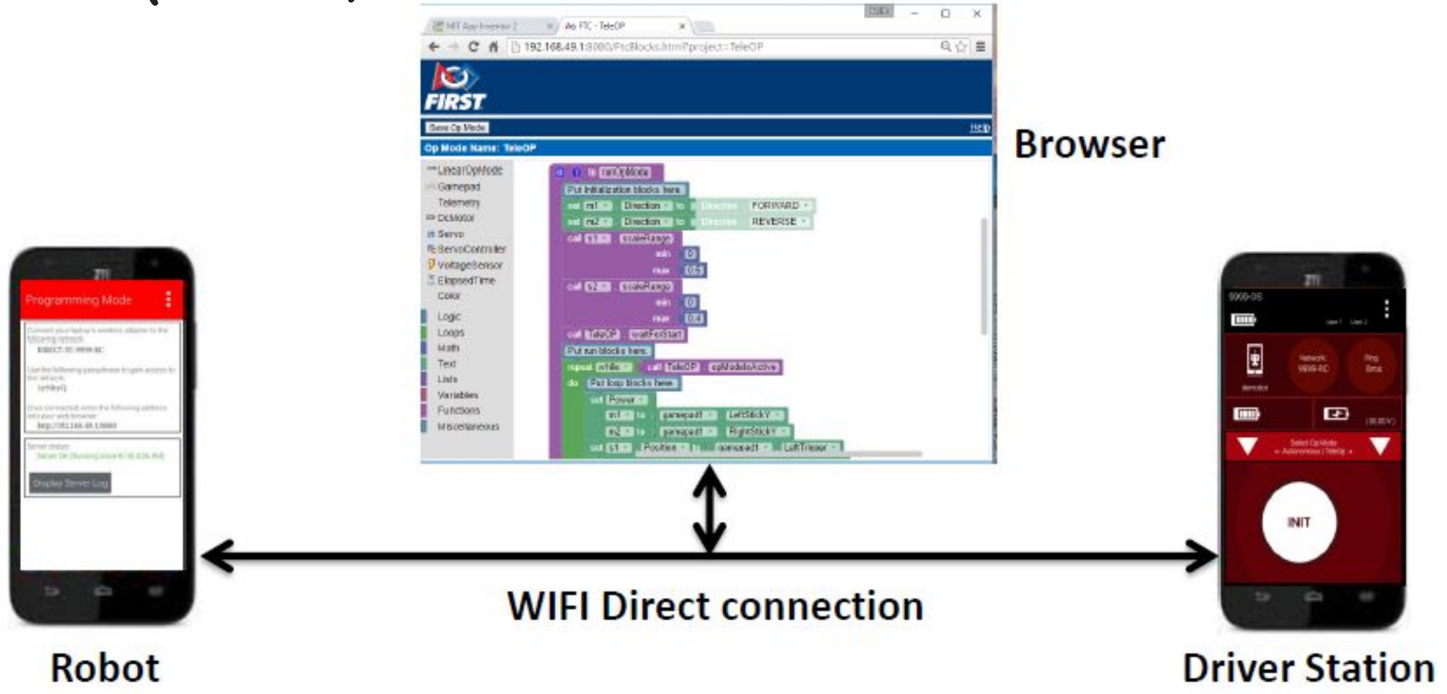




TO DO

1. Connect Electronics
2. Phone Software
3. Pair Phones
4. Configure Electronics in Software

Exercise 2 (intro) - Blocks Programming

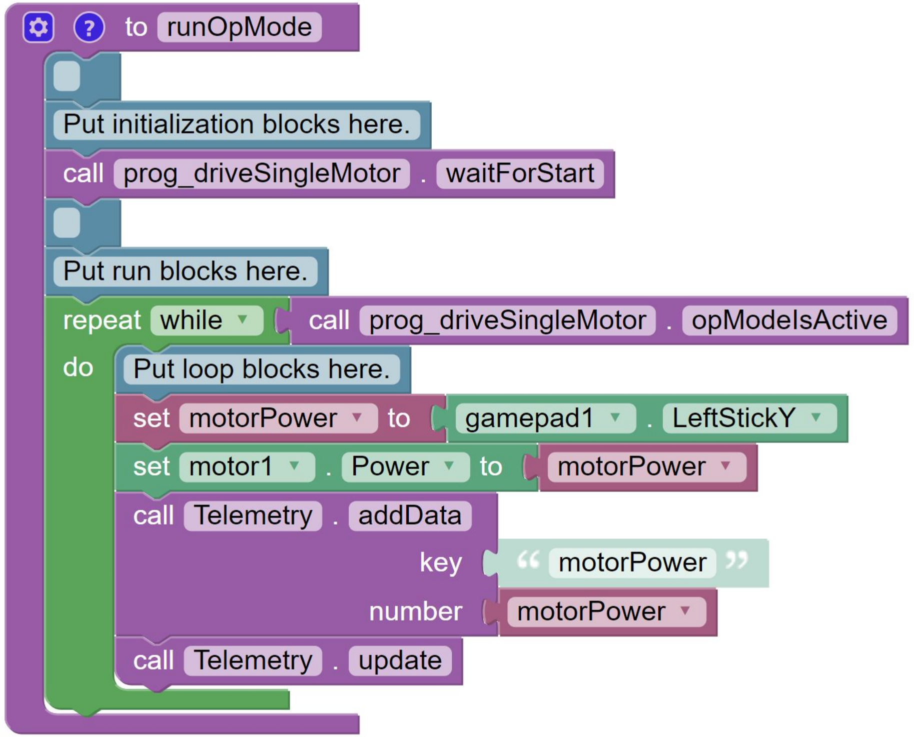


- Install FTC apps from Google Play Store or FIRST website.
- Put phones in programming mode (either side).
- Connect a computer over WIFI and program directly on the phone from Google Chrome.

Exercise 2 - Basic Teleop

- Set motor power from joystick values
- Output Telemetry of joystick values

Exercise 2 - Basic Teleop (example)



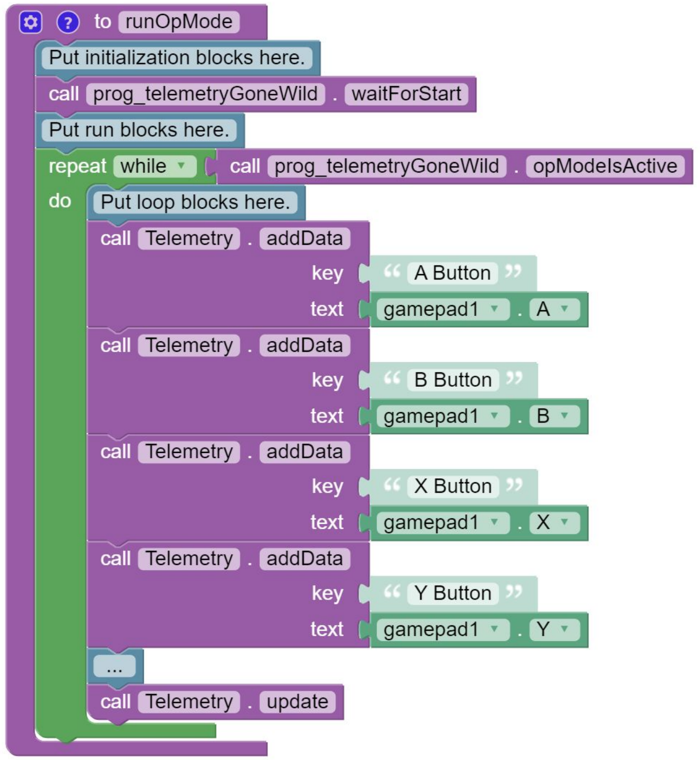
```
to runOpMode
    Put initialization blocks here.
    call prog_driveSingleMotor . waitForStart
    Put run blocks here.
    repeat while ▾ call prog_driveSingleMotor . opModeIsActive
    do  Put loop blocks here.
        set motorPower ▾ to gamepad1 ▾ . LeftStickY ▾
        set motor1 ▾ . Power ▾ to motorPower ▾
        call Telemetry . addData
                     key " motorPower "
                  number motorPower ▾
        call Telemetry . update
```

Exercise 2a - Telemetry Gone Wild!

Extra exercises for those who finish exercise 2 quickly

- Telemetry of all buttons/triggers from joystick
- Output elapsed time
- Output a counter that shows main op mode iterations

Exercise 2a - Telemetry Gone Wild! (example)



```
to runOpMode
    Put initialization blocks here.
    call prog_telemetryGoneWild . waitForStart
    Put run blocks here.
    repeat while call prog_telemetryGoneWild . opModeIsActive
    do  Put loop blocks here.
        call Telemetry . addData
                         key  " A Button "
                         text gamepad1 . A
        call Telemetry . addData
                         key  " B Button "
                         text gamepad1 . B
        call Telemetry . addData
                         key  " X Button "
                         text gamepad1 . X
        call Telemetry . addData
                         key  " Y Button "
                         text gamepad1 . Y
        ...
        call Telemetry . update
```

Intermediate Programming

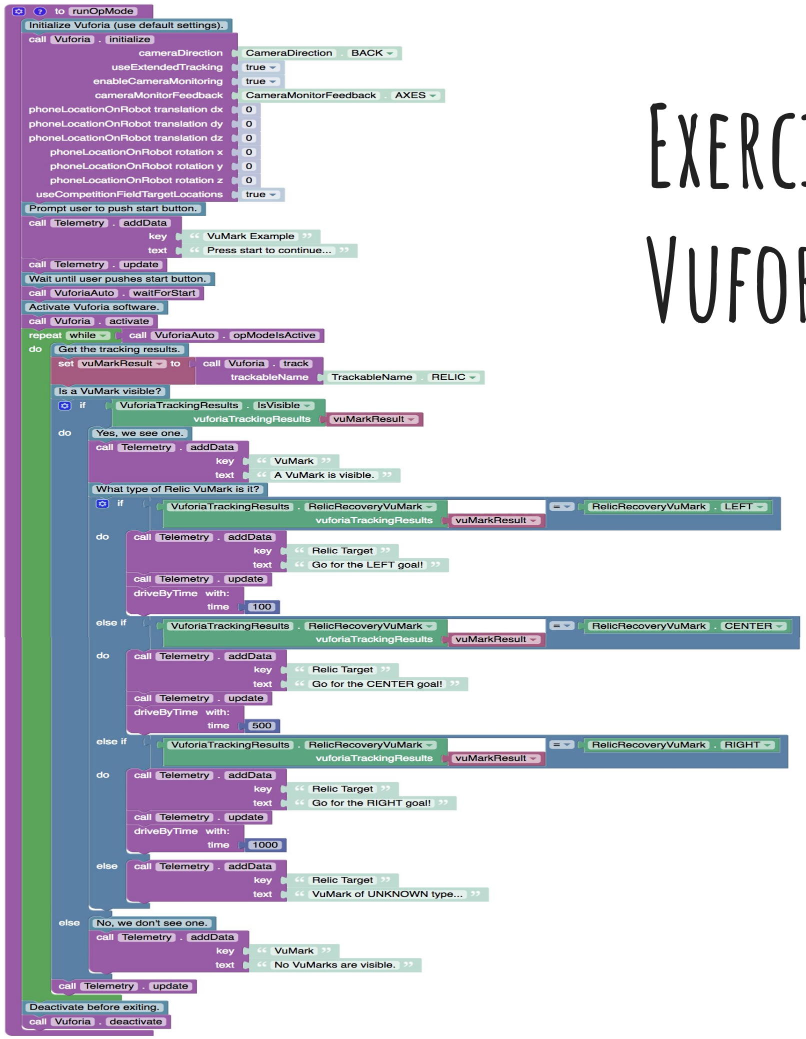
Exercise 3 - Get Vuforia Going

- Use the built-in example - ConceptVuMarkDetection
- Get Vuforia up and recognizing the VuMarks
- Understand what is going on
- Discussion of logistic issues (where the phone has to be placed)

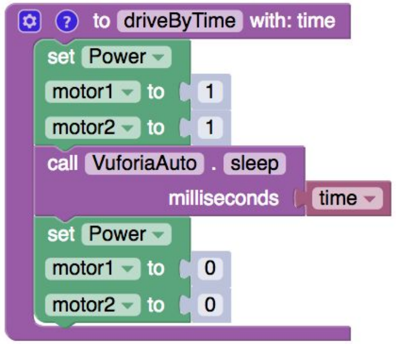
Exercise 4 - Simple Autonomous ft. Vuforia

- Start with the ConceptVuMarkDetection sample
- Drive for a specific time depending upon which image you see
- Use a function for driving a certain time

Exercise 4 - Simple Autonomous ft. Vuforia



```
to runOpMode
  Initialize Vuforia (use default settings).
  call Vuforia . initialize
                          cameraDirection  CameraDirection . BACK
                      useExtendedTracking  true
                   enableCameraMonitoring  true
                    cameraMonitorFeedback  CameraMonitorFeedback . AXES
  phoneLocationOnRobot translation dx  0
  phoneLocationOnRobot translation dy  0
  phoneLocationOnRobot translation dz  0
     phoneLocationOnRobot rotation x  0
     phoneLocationOnRobot rotation y  0
     phoneLocationOnRobot rotation z  0
   useCompetitionFieldTargetLocations  true
  Prompt user to push start button.
  call Telemetry . addData
                 key  " VuMark Example "
                text  " Press start to continue... "
  call Telemetry . update
  Wait until user pushes start button.
  call VuforiaAuto . waitForStart
  Activate Vuforia software.
  call Vuforia . activate
  repeat while  call VuforiaAuto . opModeIsActive
  do  Get the tracking results.
      set vuMarkResult to  call Vuforia . track
                                trackableName  TrackableName . RELIC
      Is a VuMark visible?
      if  VuforiaTrackingResults . IsVisible
                   vuforiaTrackingResults  vuMarkResult
      do  Yes, we see one.
          call Telemetry . addData
                          key  " VuMark "
                         text  " A VuMark is visible. "
          What type of Relic VuMark is it?
          if  VuforiaTrackingResults . RelicRecoveryVuMark              =  RelicRecoveryVuMark . LEFT
                             vuforiaTrackingResults  vuMarkResult
          do  call Telemetry . addData
                             key  " Relic Target "
                            text  " Go for the LEFT goal! "
              call Telemetry . update
              driveByTime  with:
                         time  100
          else if  VuforiaTrackingResults . RelicRecoveryVuMark         =  RelicRecoveryVuMark . CENTER
                             vuforiaTrackingResults  vuMarkResult
          do  call Telemetry . addData
                             key  " Relic Target "
                            text  " Go for the CENTER goal! "
              call Telemetry . update
              driveByTime  with:
                         time  500
          else if  VuforiaTrackingResults . RelicRecoveryVuMark         =  RelicRecoveryVuMark . RIGHT
                             vuforiaTrackingResults  vuMarkResult
          do  call Telemetry . addData
                             key  " Relic Target "
                            text  " Go for the RIGHT goal! "
              call Telemetry . update
              driveByTime  with:
                         time  1000
          else  call Telemetry . addData
                             key  " Relic Target "
                            text  " VuMark of UNKNOWN type... "
      else  No, we don't see one.
          call Telemetry . addData
                          key  " VuMark "
                         text  " No VuMarks are visible. "
      call Telemetry . update
  Deactivate before exiting.
  call Vuforia . deactivate
```



```
to driveByTime with: time
  set Power
    motor1  to  1
    motor2  to  1
  call VuforiaAuto . sleep
              milliseconds  time
  set Power
    motor1  to  0
    motor2  to  0
```

Advanced Programming (Blockly)

Exercise 5 - More Autonomous

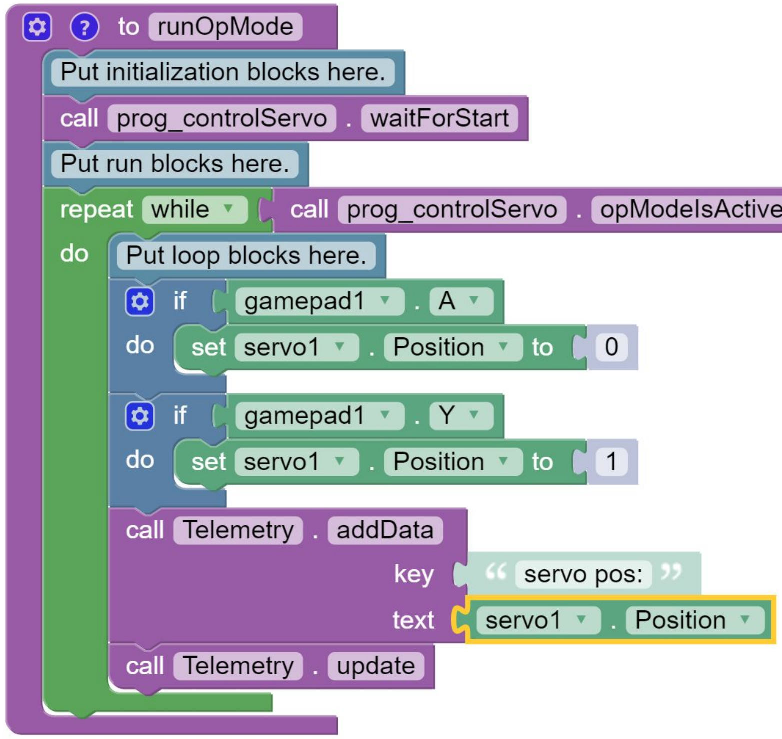
Take the code from exercise 4 and go further

- Drive a pattern based on the image you see
- Think about how to avoid duplicating code
- How precise can you get?

Exercise 6 - Control a Servo

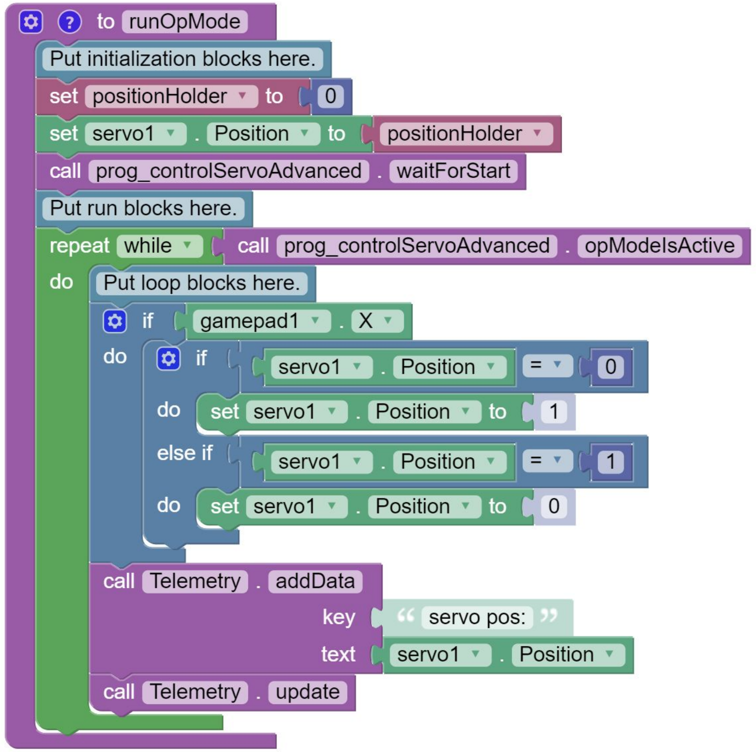
- Plug in a servo
- BASIC:
  - Control servo to two positions based on two different buttons
- ADVANCED:
  - Control servo to two positions using a single button toggle
  - What issues did you run into?
- EXTREME:
  - Control servo to two position using a latched single button

Exercise 3 - Control a Servo (BASIC example)



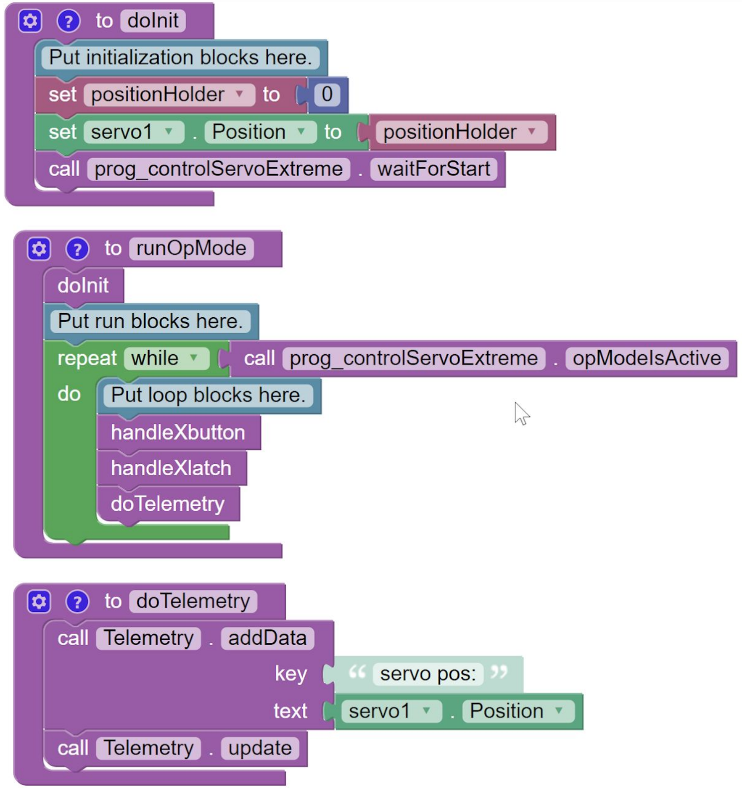
```
to runOpMode
  Put initialization blocks here.
  call prog_controlServo . waitForStart
  Put run blocks here.
  repeat while ▾  call prog_controlServo . opModeIsActive
  do
    Put loop blocks here.
    if gamepad1 ▾ . A ▾
    do  set servo1 ▾ . Position ▾ to 0
    if gamepad1 ▾ . Y ▾
    do  set servo1 ▾ . Position ▾ to 1
    call Telemetry . addData
                       key  " servo pos: "
                       text servo1 ▾ . Position ▾
    call Telemetry . update
```

Exercise 3 - Control a Servo (advanced)



```
to runOpMode
  Put initialization blocks here.
  set positionHolder to 0
  set servo1 . Position to positionHolder
  call prog_controlServoAdvanced . waitForStart
  Put run blocks here.
  repeat while call prog_controlServoAdvanced . opModeIsActive
  do  Put loop blocks here.
      if gamepad1 . X
      do  if servo1 . Position = 0
          do  set servo1 . Position to 1
          else if servo1 . Position = 1
          do  set servo1 . Position to 0
      call Telemetry . addData
                          key  " servo pos: "
                          text servo1 . Position
      call Telemetry . update
```

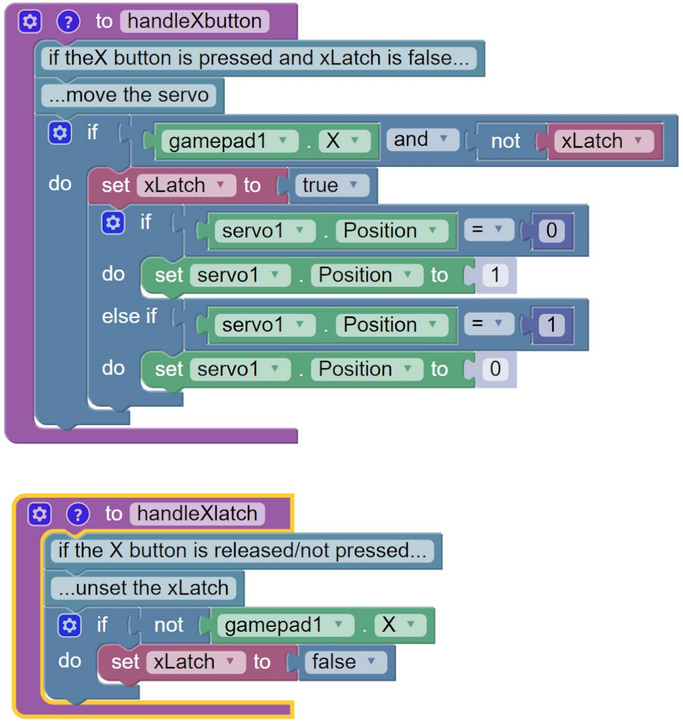
Exercise 3 - Control a Servo (extreme!)



```
to doInit
    Put initialization blocks here.
    set positionHolder to 0
    set servo1 . Position to positionHolder
    call prog_controlServoExtreme . waitForStart

to runOpMode
    doInit
    Put run blocks here.
    repeat while call prog_controlServoExtreme . opModeIsActive
    do  Put loop blocks here.
        handleXbutton
        handleXlatch
        doTelemetry

to doTelemetry
    call Telemetry . addData
                         key  " servo pos: "
                         text servo1 . Position
    call Telemetry . update
```



```
to handleXbutton
    if theX button is pressed and xLatch is false...
    ...move the servo
    if gamepad1 . X and not xLatch
    do  set xLatch to true
        if servo1 . Position = 0
        do  set servo1 . Position to 1
        else if servo1 . Position = 1
        do  set servo1 . Position to 0

to handleXlatch
    if the X button is released/not pressed...
    ...unset the xLatch
    if not gamepad1 . X
    do  set xLatch to false
```

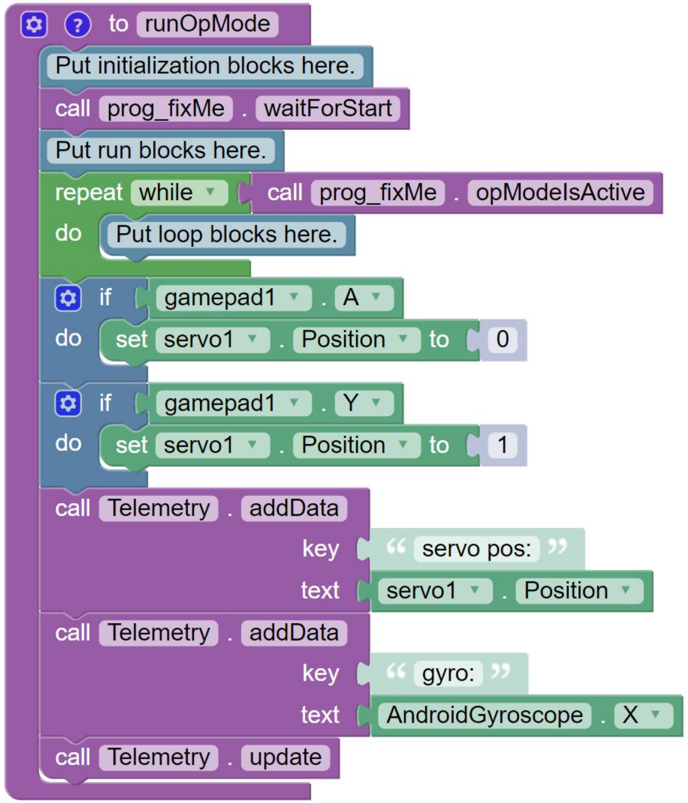
Other Stuff We Should Mention

- Troubleshooting
- Backup your code / versioning / multiple copies
- Config file naming / location / saving
- Consistency (naming variables / actuations / sensors)
- Labeling
- Comments

Troubleshooting!

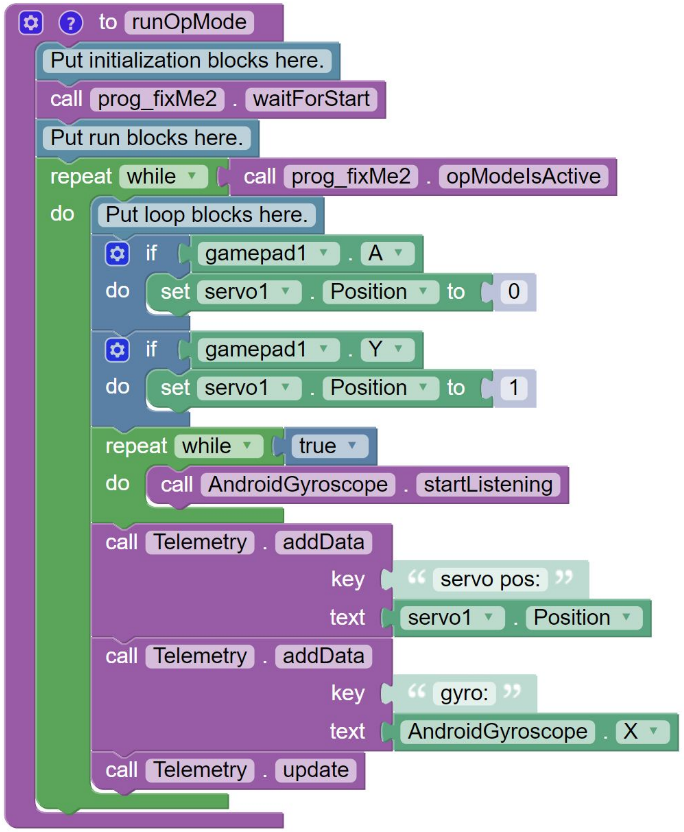
- Use Telemetry
- Watch for traps in loops
- Logs

Fix Me!



```
to runOpMode
    Put initialization blocks here.
    call prog_fixMe . waitForStart
    Put run blocks here.
    repeat while call prog_fixMe . opModeIsActive
    do  Put loop blocks here.
    if gamepad1 . A
    do  set servo1 . Position to 0
    if gamepad1 . Y
    do  set servo1 . Position to 1
    call Telemetry . addData
                        key  "servo pos:"
                        text servo1 . Position
    call Telemetry . addData
                        key  "gyro:"
                        text AndroidGyroscope . X
    call Telemetry . update
```

Fix Me 2!



```
to runOpMode
    Put initialization blocks here.
    call prog_fixMe2 . waitForStart
    Put run blocks here.
    repeat while ( call prog_fixMe2 . opModeIsActive )
    do  Put loop blocks here.
        if ( gamepad1 . A )
        do  set servo1 . Position to ( 0 )
        if ( gamepad1 . Y )
        do  set servo1 . Position to ( 1 )
        repeat while ( true )
        do  call AndroidGyroscope . startListening
        call Telemetry . addData
                         key  ( " servo pos: " )
                         text ( servo1 . Position )
        call Telemetry . addData
                         key  ( " gyro: " )
                         text ( AndroidGyroscope . X )
        call Telemetry . update
```

Saving Code

- backup your code!
- Backup Your Code!!
- BACKUP YOUR CODE!!!
    - Backup/download in blockly
    - Save the files off the phone
        - /root/sdcard/FIRST/blocks
        - Be sure to grab both the .js and .blk files per op mode
    - Save in multiple locations
- Use versioning during development
- Use Descriptive names for your op modes

Config File

- Be descriptive with config file names
- Always know which config file is which
- Backup your config file(s)
  - /root/sdcard/FIRST/<filename>.xml

Click to add Title ← utterly useless

- Be consistent and descriptive with names:
  - Variables
  - Functions
  - Objects
- Label stuff on your robot
  - Wires
  - Actuators
  - Sensors
- Lots of useful code comments
- Create documentation for anything you might forget or that might be useful

Acknowledgements

Some material borrowed from Purdue FIRST course

Some material from training @ St. Louis

Some material from *FIRST*

Some material from us!

Resources

Build Resources

https://www.firstinspires.org/node/5181

Programming Resources

https://www.firstinspires.org/node/5291

https://github.com/ftctechnh/ftc_app/wiki/Blocks-Tutorial