

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)

```
graph TD; A[Controls and Programming Introduction (Blockly)] --> B[Season In A Session]; A --> C[Controls and Programming Intermediate (Blockly)]; C --> D[Advanced Blockly]; C --> E[Transition To Java];
```

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 | 10 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

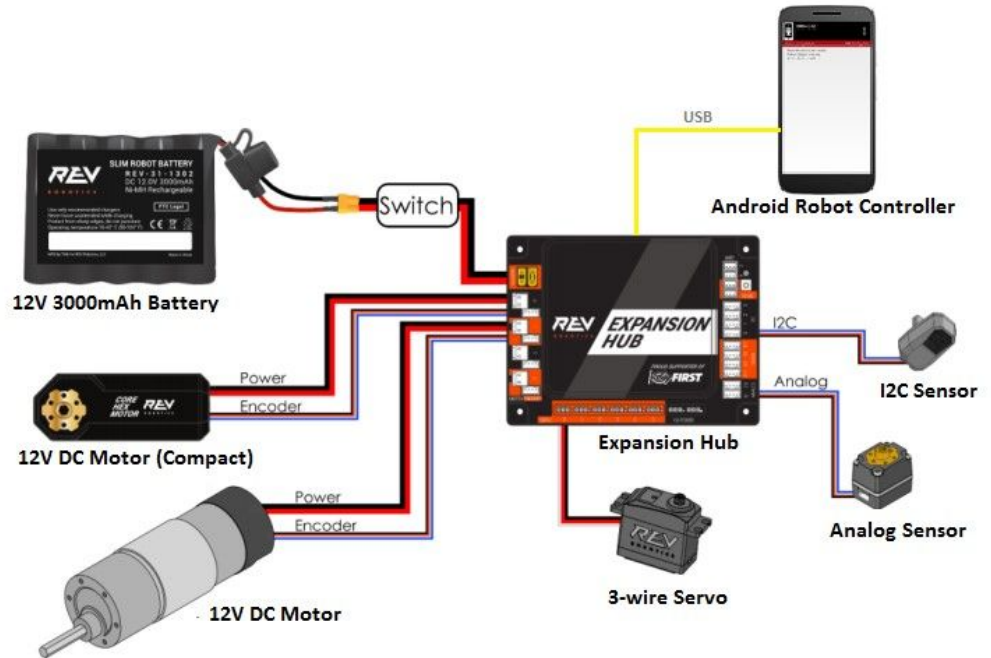
INTRODUCTION

EXERCISE 1 - SET UP ELECTRONICS AND PHONES

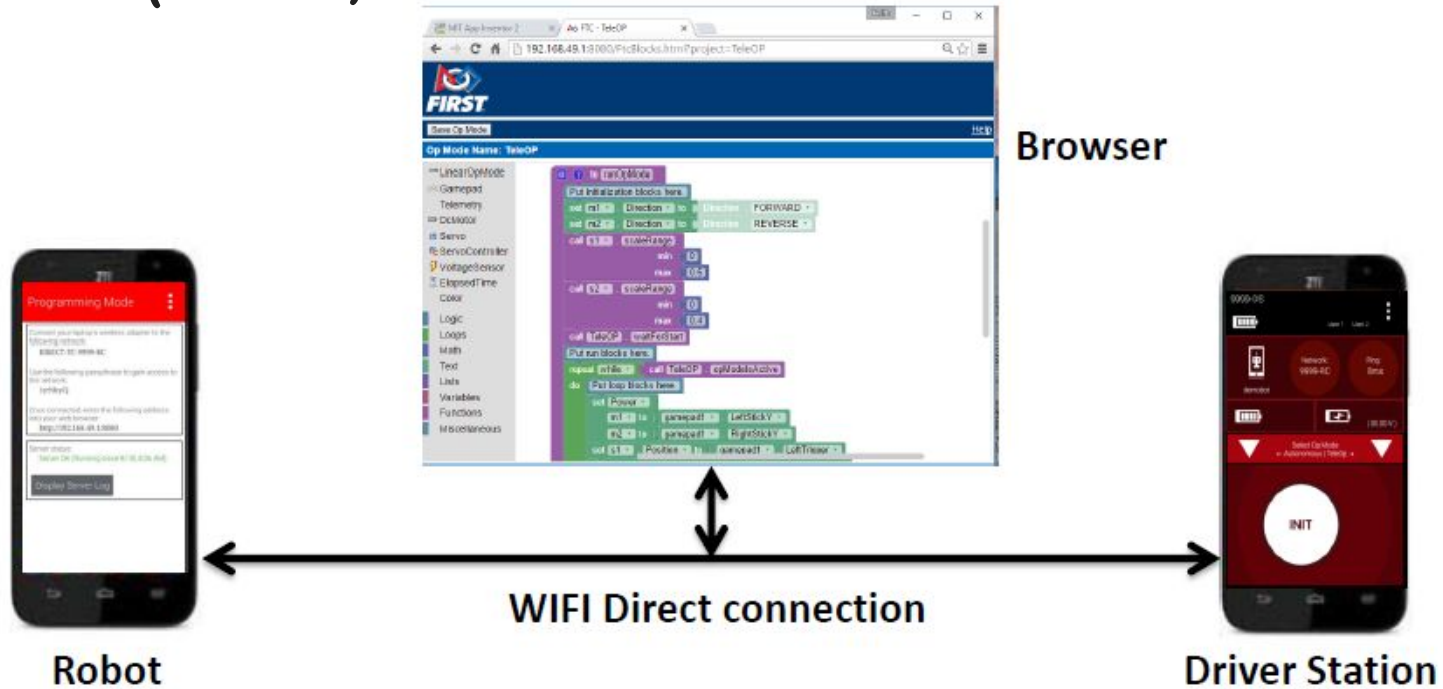


TO DO

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 . opModelsActive
  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
```

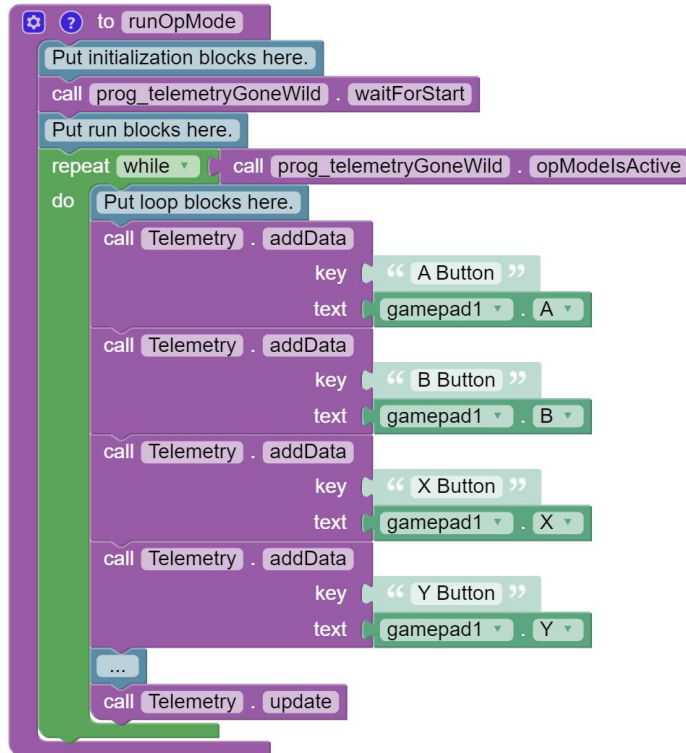
The image shows a Scratch-style code editor with a purple background. The code is written in a block-based language. It starts with a 'to runOpMode' block. Below it are two blue blocks: 'Put initialization blocks here.' and 'call prog_driveSingleMotor . waitForStart'. Another blue block 'Put run blocks here.' is followed by a green 'repeat while' block. The 'repeat while' block has a condition 'call prog_driveSingleMotor . opModelsActive'. Inside the 'do' loop, there are four blocks: 'Put loop blocks here.', 'set motorPower to gamepad1 . LeftStickY', 'set motor1 . Power to motorPower', and 'call Telemetry . addData'. The 'addData' block has two inputs: 'key' with the value 'motorPower' and 'number' with the value 'motorPower'. Finally, there is a 'call Telemetry . update' block.

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 . opModelsActive
  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
```

The image shows a Scratch code block for a telemetry program. The code is written in a purple Scratch script block and is organized into several sections:

- Initialization:** A comment block "Put initialization blocks here." followed by a "call" block for "prog_telemetryGoneWild" with the "waitForStart" method.
- Run Section:** A comment block "Put run blocks here." followed by a "repeat while" loop. The loop condition is "call prog_telemetryGoneWild . opModelsActive".
- Loop Body:** A "do" block containing several "call Telemetry . addData" blocks. Each block has two arguments: "key" and "text". The keys are "A Button", "B Button", "X Button", and "Y Button". The text values are "gamepad1 . A", "gamepad1 . B", "gamepad1 . X", and "gamepad1 . Y".
- Update:** A "call Telemetry . update" block at the end of the loop.

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 true
    call VuforiaAuto . opModelsActive
  do
    Get the tracking results.
    set vuMarkResult to call Vuforia . track
      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
        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
        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
        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 . opModelsActive
  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
```

The image shows a Scratch script for controlling a servo motor. The script is contained within a 'to runOpMode' block. It starts with a comment 'Put initialization blocks here.', followed by a 'call prog_controlServo . waitForStart' block. Another comment 'Put run blocks here.' is followed by a 'repeat while call prog_controlServo . opModelsActive' block. Inside the repeat loop, there is a 'do' block containing several sub-blocks: a comment 'Put loop blocks here.', an 'if gamepad1 . A' block with a 'do' block 'set servo1 . Position to 0', an 'if gamepad1 . Y' block with a 'do' block 'set servo1 . Position to 1', a 'call Telemetry . addData' block with 'key' set to ' servo pos: ' and 'text' set to 'servo1 . Position', and finally a 'call Telemetry . update' block. The 'text' field in the 'addData' block is highlighted with a yellow border.

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 . opModelsActive
  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
```

The image shows a Scratch script for controlling a servo motor. The script is organized into initialization and run phases. In the initialization phase, a variable 'positionHolder' is set to 0, and the servo's position is set to this value. A 'waitForStart' block is used to ensure the program waits for the user to press a button before starting. The run phase is enclosed in a 'repeat while' loop that continues as long as 'opModelsActive' is true. Inside this loop, an 'if' block checks for the 'X' button on 'gamepad1'. If pressed, another 'if' block checks the current servo position. If it is 0, it is set to 1; if it is 1, it is set to 0, effectively toggling the servo's position. Finally, the current position is added to the Telemetry system with the key 'servo pos:' and the value 'servo1 . Position', and the Telemetry system is updated.

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 . opModelsActive
  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

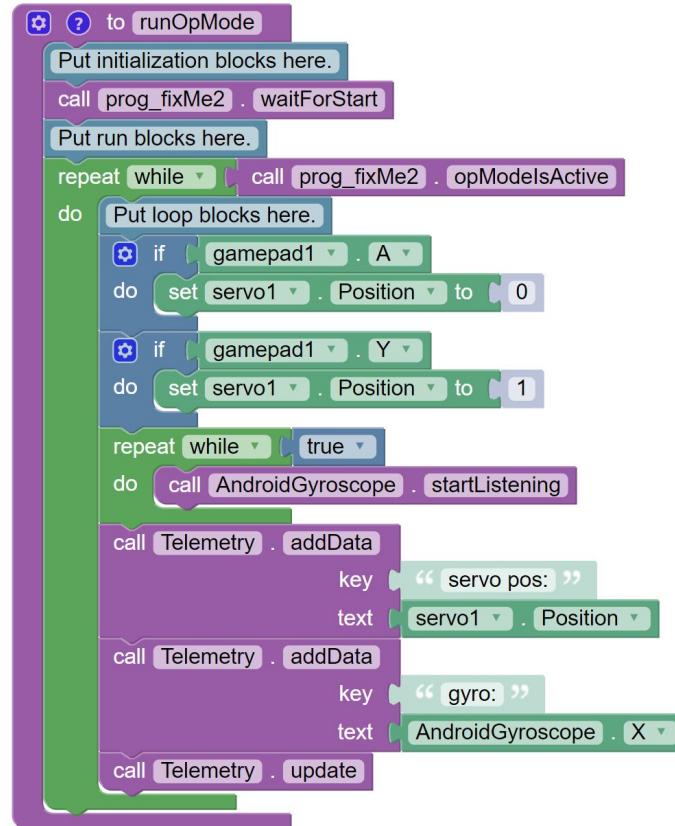
TROUBLESHOOTING!

- 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 . opModelsActive
  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 . opModelsActive
  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
```

The image shows a Scratch code block for a function named 'runOpMode'. The code is written in a block-based language and is enclosed in a purple border. It starts with a comment 'Put initialization blocks here.' followed by a call to 'prog_fixMe2 . waitForStart'. Another comment 'Put run blocks here.' is followed by a 'repeat while' loop. The loop condition is 'call prog_fixMe2 . opModelsActive'. Inside the loop, there is a 'do' block containing several actions: an 'if' block for 'gamepad1 . A' that sets 'servo1 . Position' to 0, another 'if' block for 'gamepad1 . Y' that sets 'servo1 . Position' to 1, a 'repeat while' loop with condition 'true' containing a call to 'AndroidGyroscope . startListening', two 'call Telemetry . addData' blocks (one for 'servo pos: ' and 'servo1 . Position', and another for 'gyro: ' and 'AndroidGyroscope . X'), and finally a 'call Telemetry . update' block.

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