



**EAGLE  
ROBOTICS**

# Introduction to FTC Blocks Programming

---

*2020 Georgia FIRST Tech Challenge Kickoff*

# Introduction

## **Who is this presentation for?**

- Anyone, new or returning to FTC who wants to get started programming on a team that uses Blocks
- New teams desiring to start programming a robot quickly

## **Who is this presentation NOT for?**

- Returning Blocks programmers or Java experts

## **What will be in this presentation?**

- Discussing the reasons for using Blocks
- Learning how to set up Blocks and program in it
- Working through creating basic example OpModes
- Tips and tricks from Eagle Robotics



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

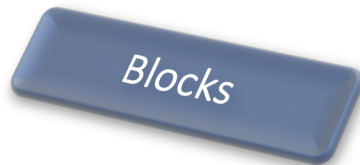
---

# About Blocks



# What is Blocks?

**There are three recommended programming interfaces for FTC:**



Learning curve increases

Setup time increases

Complexity increases

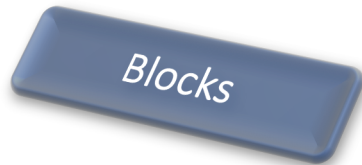
**Blocks is the easiest way to get started with programming in FTC!**



# What is Blocks?

A version of the Google Blockly language, packaged into the FTC Robot Controller software  
(originally added during the 2017-18 season)

## Why use Blocks?



Very minimal learning curve  
Minimal setup needed to program  
Only a web browser is needed

## Other considerations to be aware of:

- Teams must be connected directly to the robot controller to make code changes
- Harder to share code across multiple programs



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

---

# Options for Control Hardware

# Control Hardware: Robot Controllers

The Expansion Hub is only an interface for connecting to robot hardware – it does NOT run any code!

Robot code runs on an Android-based robot controller device, which can be connected to an Expansion Hub.



Continuing from previous seasons, a FIRST-allowed Android phone can be used as a controller. This connects to an Expansion Hub.

(Up to two Expansion Hubs can be used on a robot with an Android phone)



New this season is the REV Robotics Control Hub, which is an Expansion Hub WITH a built-in Android controller.

(Only one Expansion Hub can be used with the Control Hub, since the Control Hub already contains the ports of an Expansion Hub)



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

# Control Hardware: Robot Controllers

A summary of robot control options:



Android phone  
robot controller



Expansion Hub



Expansion Hub  
(optional)

OR



Control Hub



Expansion Hub  
(optional)

# Control Hardware: Robot Controllers



If using an extra Expansion Hub:



Control Hub

OR



First Expansion Hub



Via 3-pin JST PH



Into a UART port

Expansion Hub  
(optional)

OR



Control Hub

OR



Android phone  
robot controller



Via USB-A to Mini-USB



Expansion Hub  
(optional)

# Control Software

Two devices are core to the robot control/communication system:

## Robot controller



- Stores and processes team robot code
- Is a Control Hub or Android phone (connected to an Expansion Hub)
- Must receive start/stop instructions for the driver station to do anything

## Driver station

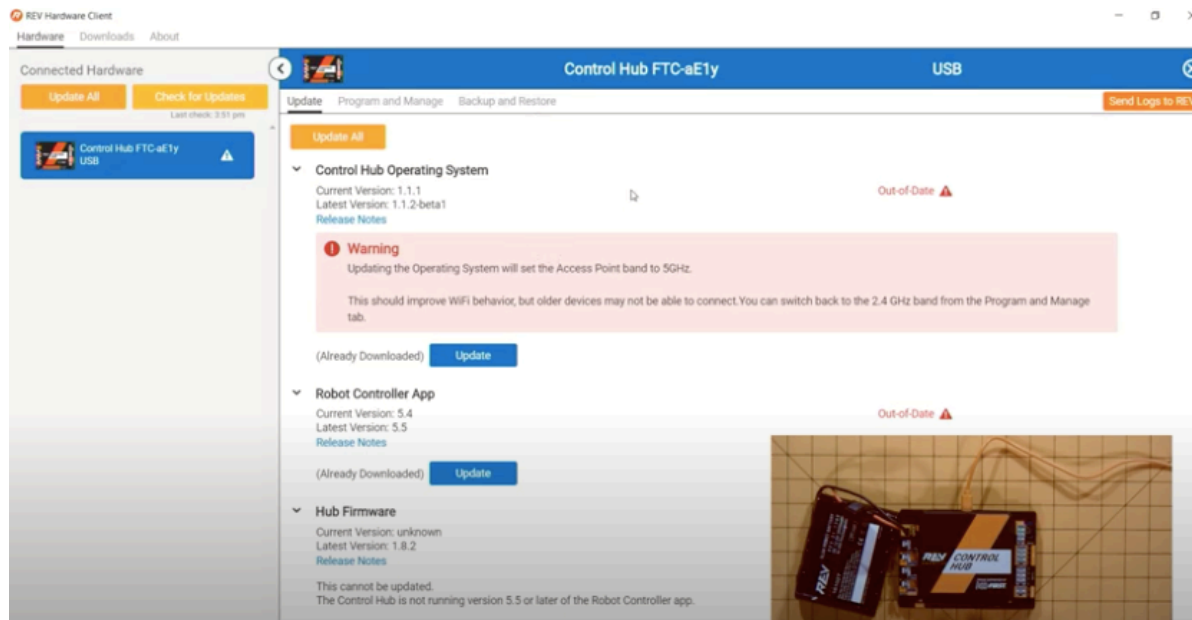


- Is an Android phone
- Connects to the robot controller and issues program start/stop instructions
- Up to two gamepads can be connected for driver-operated control



# Installing the Control Software

Download the REV Hardware Client from [revrobotics.com/software](http://revrobotics.com/software)





**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

---

# Setting up the Robot Controller



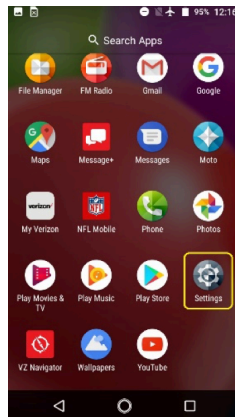
# Control Software

Setting up the robot controller:

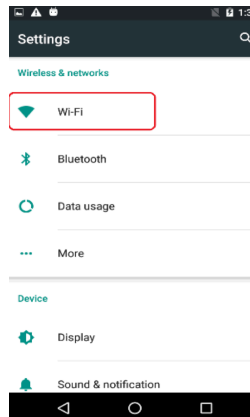


If using Android phone (version 7):  
(exact steps may differ based on phone model and OS version)

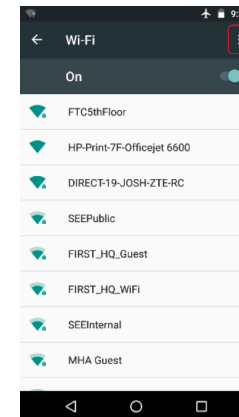
1. Open Settings by selecting it from the apps list.



2. Select Wi-Fi to load Wi-Fi options.



3. Tap the three vertical dots at the top right to show more options.



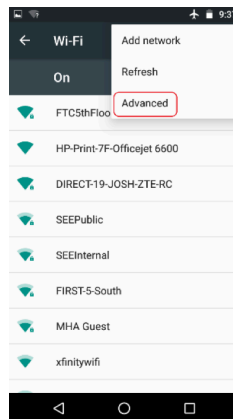
# Control Software

Setting up the robot controller:

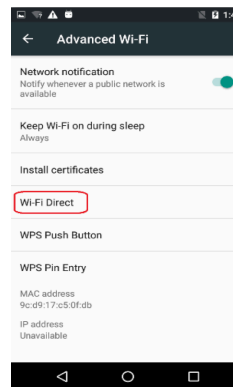


If using Android phone (version 7):  
(exact steps may differ based on phone model and OS version)

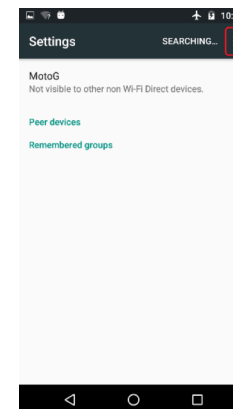
4. Select  
“Advanced”.



5. Tap “Wi-Fi  
Direct”.



6. Tap the  
three  
vertical dots  
to show  
additional  
options.



# Control Software

Setting up the robot controller:



If using Android phone (version 7):  
(exact steps may differ based on phone model and OS version)

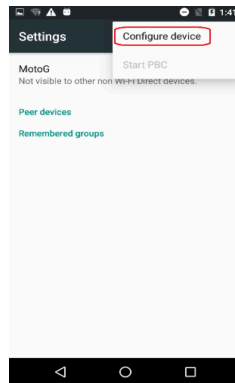
9999 – A – RC

Team  
number

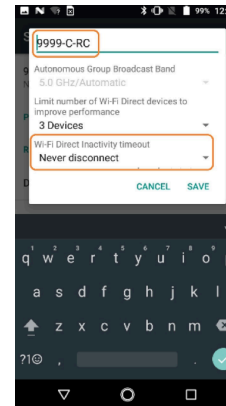
Group  
letter

Device  
type

7. Tap  
“Configure  
Device”.



8. Change the device  
name to match the  
FIRST-required  
naming convention,  
and optionally set  
the Wi-Fi Direct  
timeout to “Never  
disconnect”.



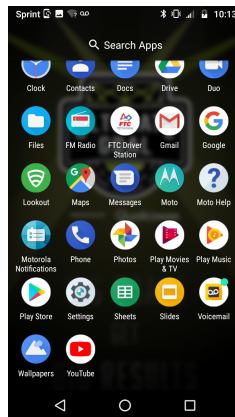
# Control Software

Setting up the robot controller:

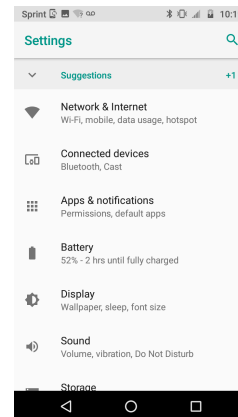


If using Android phone (version 8):  
(exact steps may differ based on phone model and OS version)

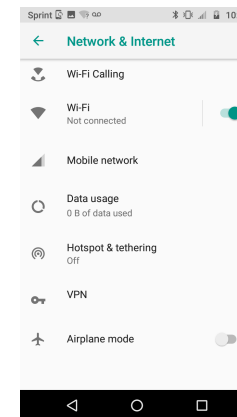
1. Open  
Settings by  
selecting it  
from the  
apps list.



2. Select  
“Network  
&  
Internet”.



3. Tap “Wi-Fi”.





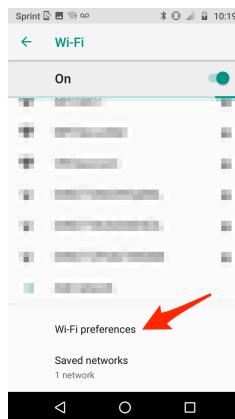
# Control Software

Setting up the robot controller:

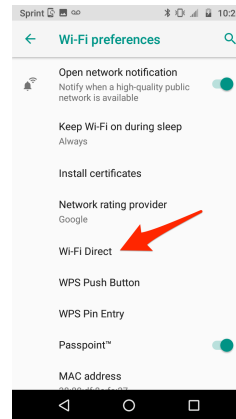


If using Android phone (version 8):  
(exact steps may differ based on phone model and OS version)

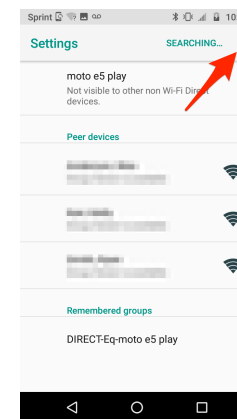
4. Select  
“Wi-Fi  
Preferences”



5. Under  
advanced  
options,  
select “Wi-  
Fi Direct”



6. Tap the  
three dots at  
the top right.



# Control Software

Setting up the robot controller:



If using Android phone (version 8):  
(exact steps may differ based on phone model and OS version)

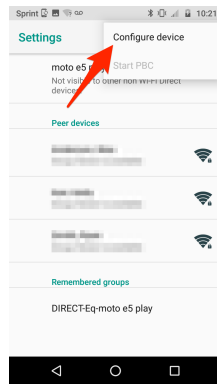
9999 – A – RC

Team  
number

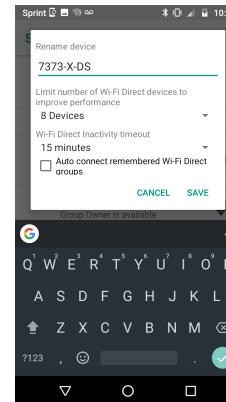
Group  
letter

Device  
type

7. Tap  
“Configure  
Device”.



8. Change the device  
name to match the  
FIRST-required  
naming convention,  
and optionally set  
the Wi-Fi Direct  
timeout to “Never  
disconnect”.



# Control Software

Setting up the robot controller:



If using Control Hub:

1. Connect your Control Hub to power and connect your computer to the Control Hub's Wi-Fi network.

(The default Control Hub network will have a network name starting with "FIRST-" or "FTC-", and a default password of "password". Make sure to change this!)

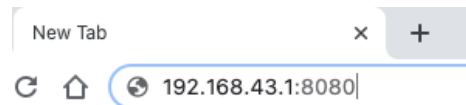
9999 – A – RC

Team  
number

Group  
letter

Device  
type

2. Open a web browser and navigate to 192.168.43.1:8080.



## Control Software

Setting up the robot controller:



If using Control Hub:

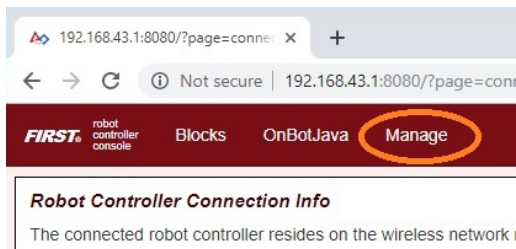
9999 – A – RC

Team  
number

Group  
letter

Device  
type

- Click “Manage” at the top of the page that loads.



- Change the Control Hub name to match the FIRST-required naming convention.  
(The group letter is only necessary if your team will use multiple Control Hubs at competition, paired with different driver stations.)







**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

---

# Setting up and Connecting the Driver Station



# Control Software

Setting up the driver station:



Android phone is the only option for driver station:  
(exact steps may differ based on phone model and OS version)

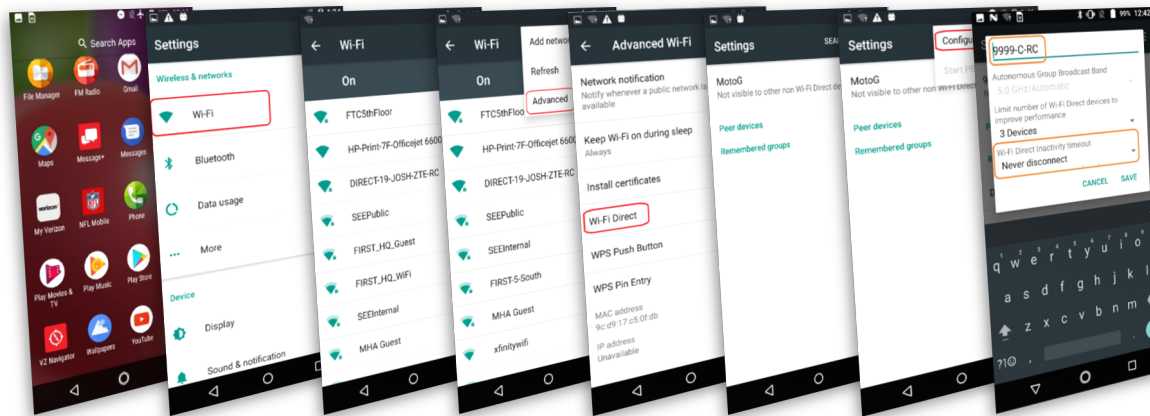
9999 – A – DS

Team  
number

Group  
letter

Device  
type

Repeat the same process as with the **Android robot controller phone** but append “DS” at the end of the phone name, rather than “RC”.



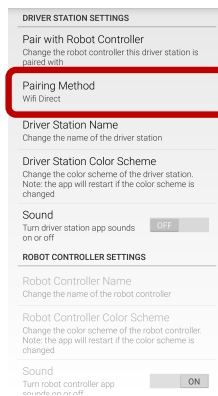


# Control Software

Connecting the driver station to the robot controller:

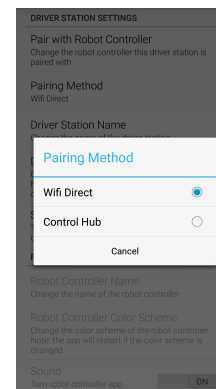
1. Open the FTC Driver Station app.
2. Tap the three dots at the top right corner of the screen.
3. Tap “Settings” from the pop-up menu.

4. Tap “Pairing Method” from the settings menu.



5. Select the method corresponding to the type of robot controller being used.

Robot Controller Type	Pairing Method
Android phone	Wi-Fi Direct
Control Hub	Control Hub



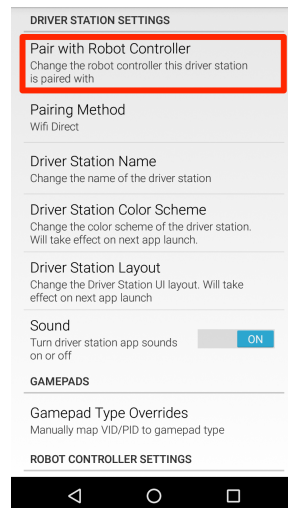
# Control Software

Connecting the driver station to the robot controller:

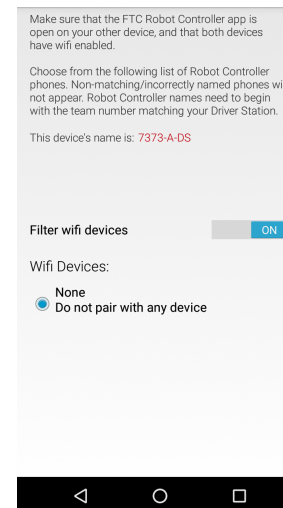


If using Android robot controller:  
(exact steps may differ based on phone model and OS version)

6. Tap “Pair with Robot Controller” from the DS app settings menu



7. Select the matching robot controller phone from the list.





**EAGLE  
ROBOTICS**

# Control Software

Connecting the driver station to the robot controller:



If using Control Hub:

6. Connect to the Control Hub's Wi-Fi network in your device's settings.  
That's it!



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

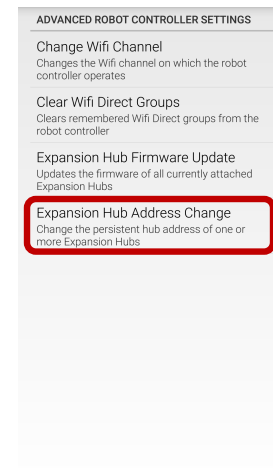
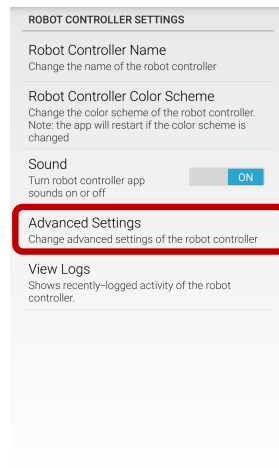
---

# Configuring Expansion Hub Addresses

# Control Software

Configuring Expansion Hub addresses:







1. Open the FTC Driver Station (or Robot Controller) app.
2. Tap the three dots at the top right corner of the screen.
3. Tap “Settings” from the pop-up menu.
4. Select “Advanced Settings under the “Robot Controller Settings” category.
5. Select Expansion Hub Address Change from the menu.



# Control Software

Do I have to go into the robot controller settings to see an Expansion Hub's address? **No!**

## Firmware Version 1.07.00 or Higher LED Codes

LED Status	LED Description	When	Hub Status
	Solid Blue	At Boot	Control Hub has power; Battery is >7V and is waiting to initialize communications.
	Solid Blue	Anytime	Hub is waiting for communication with the Driver Station Host. Control Hub has power; Battery is >7V.
 Address #	Solid Green with one or more blue blinks every ~5 Seconds	Anytime	Hub has power and active communication with the Android Platform. The number of blue blinks is the same as the Hub's address. The factory default address is 2 (  ).
	Blinking Blue	Anytime	Keep alive has timed out. Fault will clear when communication resumes.
	Blinking Orange	Anytime	Battery Voltage is lower than 7V. Either the 12V battery needs to be charged, or the Expansion Hub is running on USB power only. This fault will clear when battery voltage is raised above 7V. This will not be overwritten by the keep alive timeout pattern.



# Control Software

Setting up the robot hardware configuration:

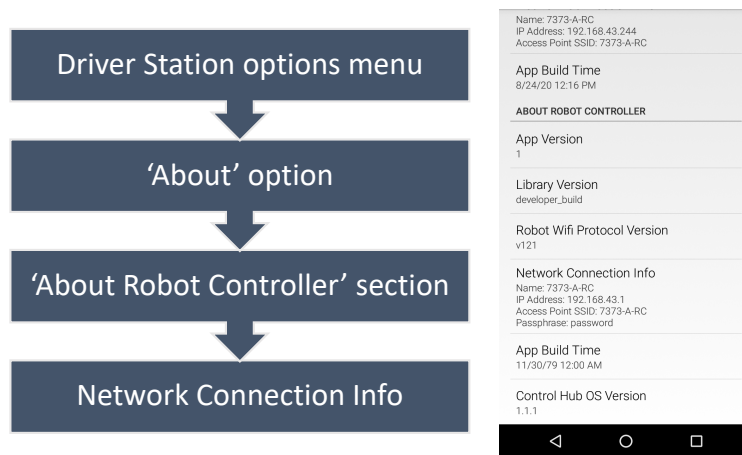
1. Open the FTC Driver Station app.
2. Tap the three dots at the top right corner of the screen.
3. Tap “Configure Robot” from the pop-up menu.
4. From there, you can enable or edit an existing configuration, or create a new configuration.



# Accessing Blocks

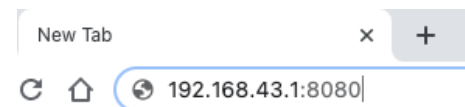
To access the Blocks interface from your computer, you will need to connect to the robot controller's Wi-Fi network.

1. Get the robot controller's network credentials.

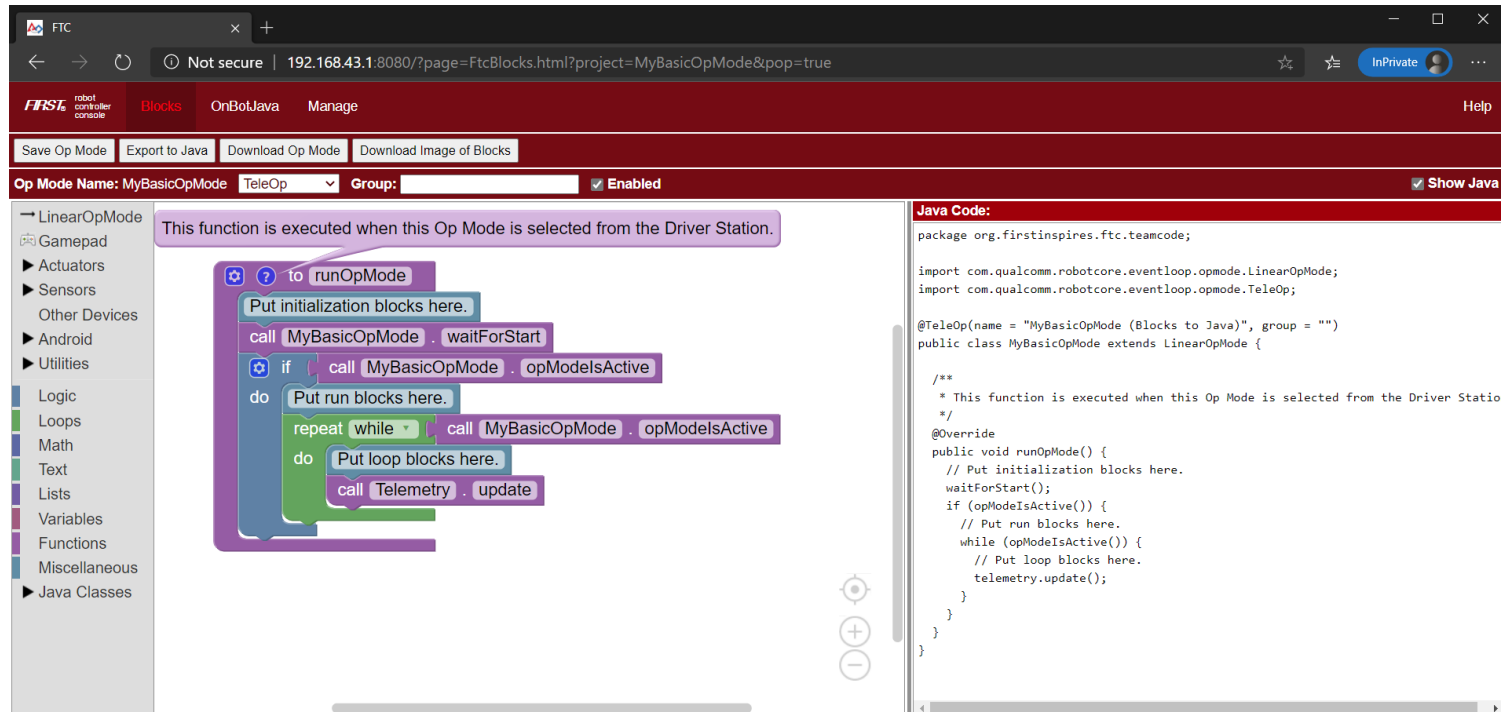


2. Connect to the robot controller's Wi-Fi network. If prompted for a security pin, select the option to enter a password instead.

3. Go in your web browser to the IP address shown in Part 1 but with :8080 appended to the end



# The Blocks Interface



The screenshot displays the FTC Blocks Programming Interface in a web browser. The interface is divided into several sections:

- Top Bar:** Includes the FTC logo, navigation tabs (Blocks, OnBotJava, Manage), and a Help button.
- Buttons:** Save Op Mode, Export to Java, Download Op Mode, and Download Image of Blocks.
- Op Mode Configuration:** Op Mode Name: MyBasicOpMode, Mode: TeleOp, Group: (empty), and a checkbox for Enabled.
- Left Panel:** A tree view showing categories like LinearOpMode, Gamepad, Actuators, Sensors, Other Devices, Android, Utilities, Logic, Loops, Math, Text, Lists, Variables, Functions, Miscellaneous, and Java Classes.
- Blocks Area:** Contains a block diagram for the `runOpMode` function. A callout box states: "This function is executed when this Op Mode is selected from the Driver Station." The blocks include:
  - `to runOpMode` (purple block)
  - `Put initialization blocks here.` (light blue block)
  - `call MyBasicOpMode . waitForStart` (purple block)
  - `if call MyBasicOpMode . opModelsActive` (purple block)
  - `do` loop (purple block) containing:
    - `Put run blocks here.` (light blue block)
    - `repeat while call MyBasicOpMode . opModelsActive` (green block)
    - `do` loop (green block) containing:
      - `Put loop blocks here.` (light blue block)
      - `call Telemetry . update` (purple block)
- Right Panel:** Shows the corresponding Java code for the blocks.

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;

@TeleOp(name = "MyBasicOpMode (Blocks to Java)", group = "")
public class MyBasicOpMode extends LinearOpMode {

    /**
     * This function is executed when this Op Mode is selected from the Driver Station
     */
    @Override
    public void runOpMode() {
        // Put initialization blocks here.
        waitForStart();
        if (opModeIsActive()) {
            // Put run blocks here.
            while (opModeIsActive()) {
                // Put loop blocks here.
                telemetry.update();
            }
        }
    }
}
```



**EAGLE  
ROBOTICS**

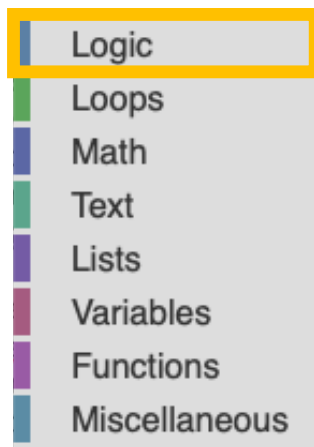
Introduction to FTC Blocks Programming

---

# The Blocks Interface



# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

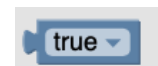
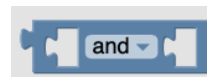
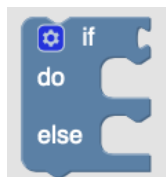
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

Storing values that can be accessed and/or updated later

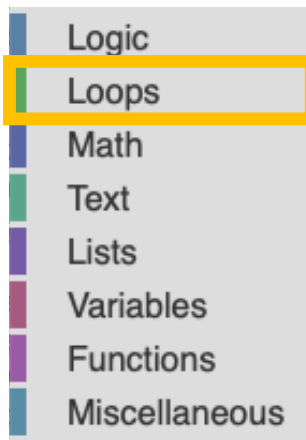
Creating sections of code that can be repeated

Comments, formatting, null, etc.





# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

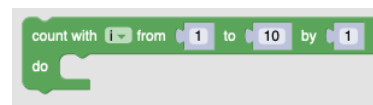
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

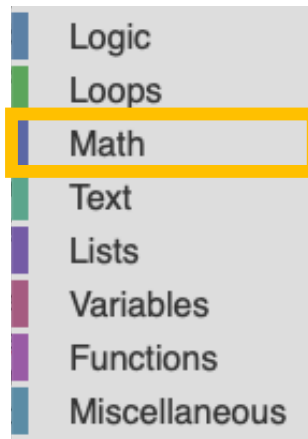
Storing values that can be accessed and/or updated later

Creating sections of code that can be repeated

Comments, formatting, null, etc.



# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

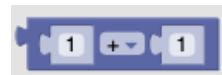
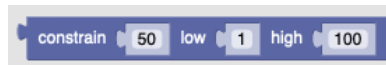
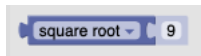
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

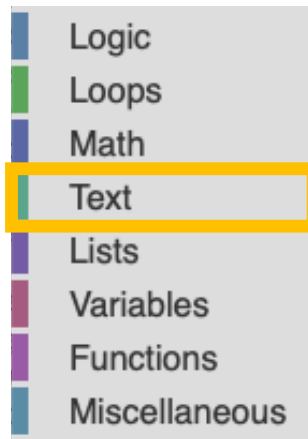
Storing values that can be accessed and/or updated later

Creating sections of code that can be repeated

Comments, formatting, null, etc.



# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

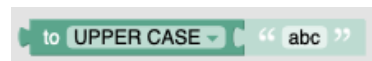
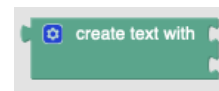
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

Storing values that can be accessed and/or updated later

Creating sections of code that can be repeated

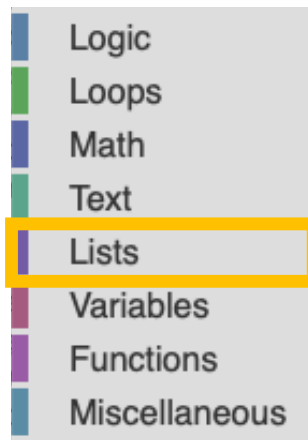
Comments, formatting, null, etc.







# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

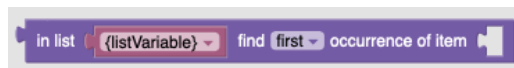
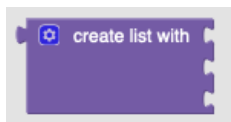
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

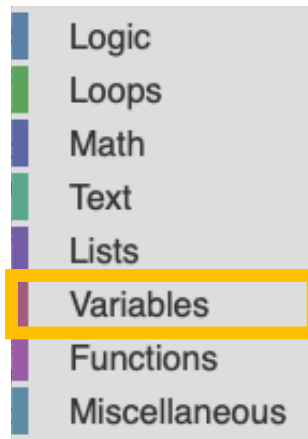
Storing values that can be accessed and/or updated later

Creating sections of code that can be repeated

Comments, formatting, null, etc.



# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

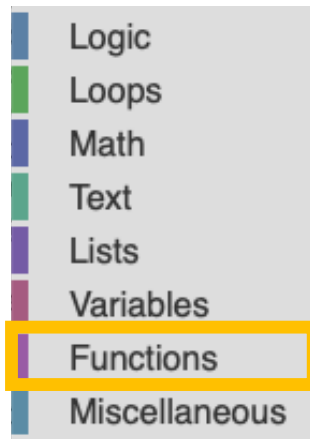
Storing values that can be accessed and/or updated later

Creating sections of code that can be repeated

Comments, formatting, null, etc.



# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

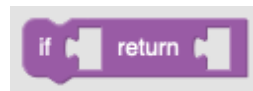
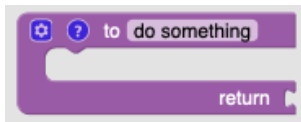
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

Storing values that can be accessed and/or updated later

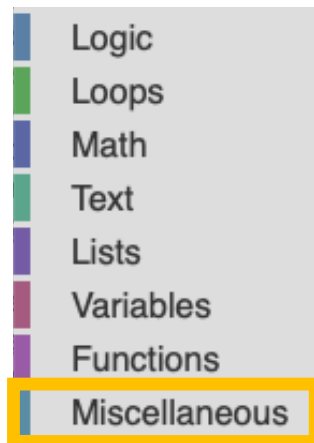
Creating sections of code that can be repeated

Comments, formatting, null, etc.





# The Blocks Interface



Logical processes and statements (i.e. true/false, if/else) – deciding which part of the program to run based on conditions

For repeating code, often iterating through a list or based on logical conditions

Various mathematical operations

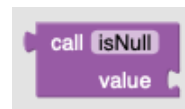
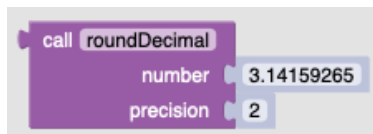
Functions related to text (creation, combination, analysis)

Functions related to lists (creation, updating list elements, retrieving values, etc.)

Storing values that can be accessed and/or updated later

Creating sections of code that can be repeated

Comments, formatting, null, etc.

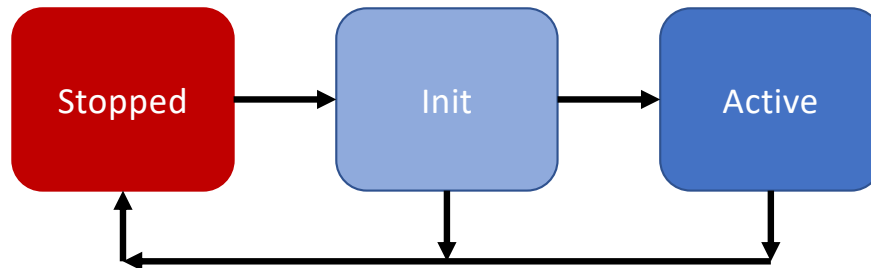


# The Blocks Interface (FTC-specific)

## → LinearOpMode

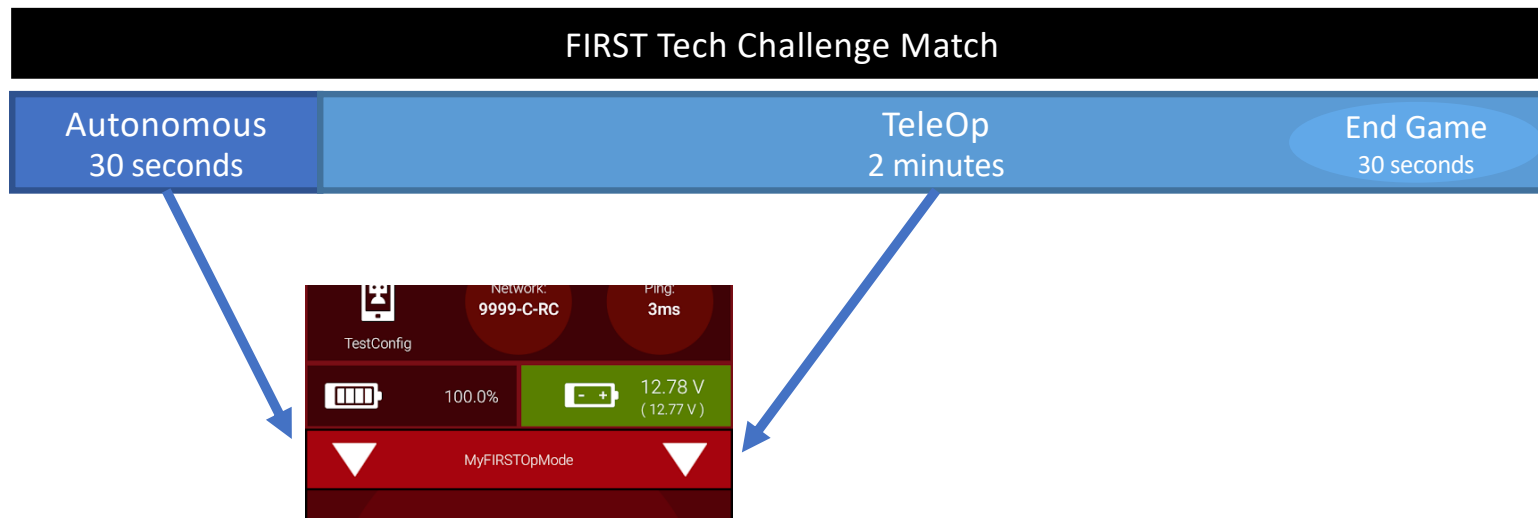
Holds functions and variables necessary for regulating program flow

Before looking at LinearOpMode capabilities, we need to understand the process of running a program.



# The Blocks Interface (FTC-specific)

Two types of OpModes:



*A section of the FTC Driver  
Station app*



# The Blocks Interface (FTC-specific)

## → LinearOpMode

Holds functions and variables necessary for regulating program flow

call LinearOpMode . waitForStart

Called during init to pause execution until OpMode is started

call LinearOpMode . sleep  
milliseconds 1000

Pauses execution for a specific amount of time

call LinearOpMode . opModeIsActive

Returns a Boolean (true/false) indicating whether the OpMode is still running

# The Blocks Interface (FTC-specific)

**Gamepad**

Contains variables for getting the state of gamepad controls

gamepad1 . LeftTrigger

gamepad1 . LeftBumper

gamepad1 . Y

gamepad1 . LeftStickX

gamepad1 . LeftStickY





# The Blocks Interface (FTC-specific)

▼ Actuators  
▼ DcMotor  
⇒ Dual

Contains functions for controlling robot motors  
(The Dual category allows for controlling two motors with one function call)

Example (setting motor power):

set right\_drive . Power to 1

Only motors registered in the active hardware configuration  
will show up in this drop-down!

Example (running to an encoder position – requires encoder to be connected):

set left\_arm . Mode to RunMode . RUN\_TO\_POSITION  
set left\_arm . TargetPosition to 800  
set left\_arm . Power to 0.5

# The Blocks Interface (FTC-specific)

▼ Actuators  
360 CRServo  
Servo

Contains functions for controlling robot servos

Both Servo and CRServo do the same thing underneath – they just provide different functions to control the servo. Whether a servo shows up under Servo or CRServo is dependent on how the servo was assigned in the hardware configuration.

set CRServo . Power to -1  
set CRServo . Power to 0  
set CRServo . Power to 1

Designed for servos  
that move continuously  
in a direction

=

set Servo . Position to 0  
set Servo . Position to 0.5  
set Servo . Position to 1

Designed for servos  
that rotate to a preset  
position



# The Blocks Interface (FTC-specific)

## ▼ Sensors

IMU-BNO055

IMU-BNO055.Parameters

DistanceSensor

REV Color/Range Sensor

VoltageSensor

## ▼ Other Devices

AnalogInput

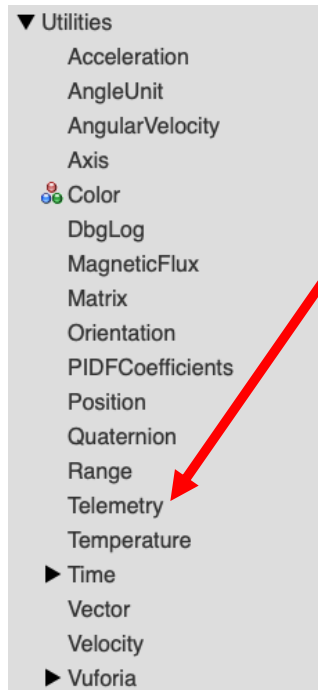
Contains functionality for reading data from different sensors on the robot

The categories to the left are a few of the available sensor options. Some sensors, like potentiometers, may appear in the “Other Devices” category.



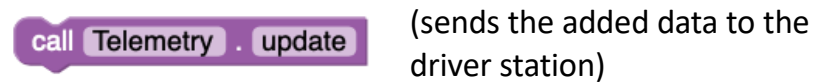
The IMU-BNO055 category gives capability for reading data from the REV Hub IMU (inertial measurement unit), giving teams features such as angle measurement, which can be useful when desiring to drive straight or turn.

# The Blocks Interface (FTC-specific)

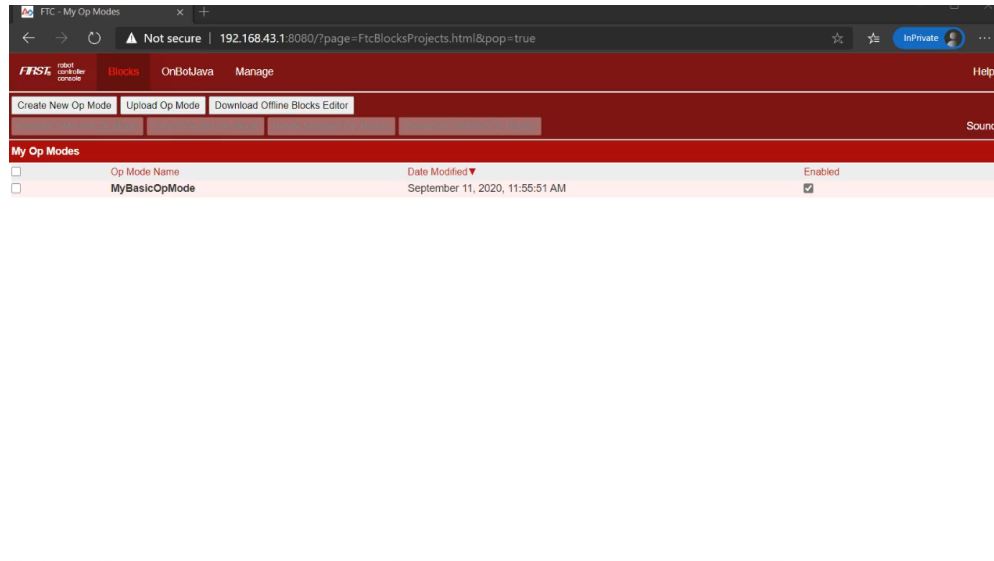


Telemetry is a vital part of creating and debugging FTC programs!

Only two blocks needed to send data to the driver station:

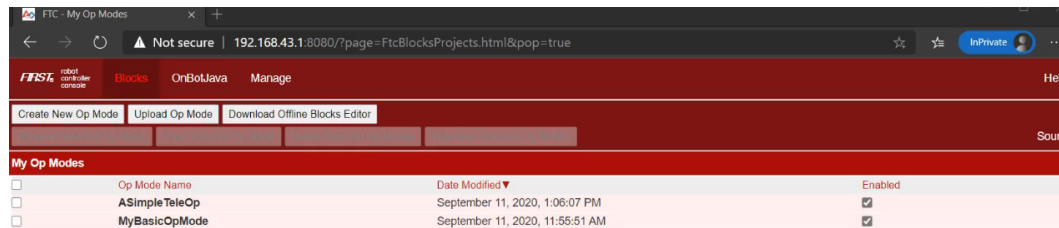


# Let's Try Some Examples!



1. Create a basic TeleOp program (driver-controlled) to drive two motors from a gamepad

# Let's Try Some Examples!



The screenshot shows a web browser window titled "FTC - My Op Modes". The address bar shows "192.168.43.1:8080/?page=FtcBlocksProjects.html&pop=true". The page has a red header with "FTC robot controller console" and tabs for "Blocks", "OnBotJava", and "Manage". Below the header are buttons for "Create New Op Mode", "Upload Op Mode", and "Download Offline Blocks Editor". A "Sounds" button is on the right. The main content area is titled "My Op Modes" and contains a table with the following data:

<input type="checkbox"/>	Op Mode Name	Date Modified▼	Enabled
<input type="checkbox"/>	ASimpleTeleOp	September 11, 2020, 1:06:07 PM	<input checked="" type="checkbox"/>
<input type="checkbox"/>	MyBasicOpMode	September 11, 2020, 11:55:51 AM	<input checked="" type="checkbox"/>

2. Create an autonomous program to move the robot forwards and backwards twice



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

---

# Extra Features to Know

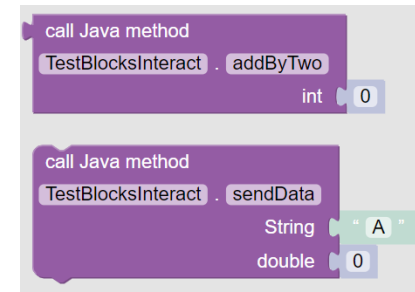
# Blocks and Java Together?

What if you have one team member that's familiar with Java, but other team members only know how to use blocks? Just use both! (New feature in FTC software v5.5)

```
import org.firstinspires.ftc.robotcore.external.ExportToBlocks;
import org.firstinspires.ftc.robotcore.external.Telemetry;

public class TestBlocksInteract {
    @ExportToBlocks
    public static int addByTwo(int number) {
        return number + 2;
    }

    @ExportToBlocks
    public static void sendData(String title, double value, Telemetry telemetry) {
        telemetry.addData( caption: "(BLOCKS) "+ title, value);
    }
}
```

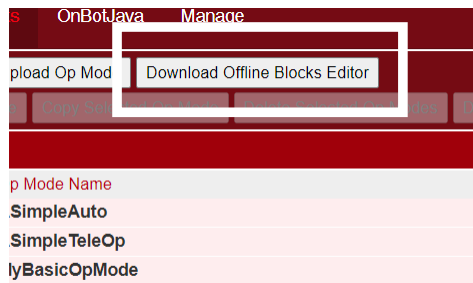


Public static methods in Java, annotated with `@ExportToBlocks`, are automatically added to Blocks.

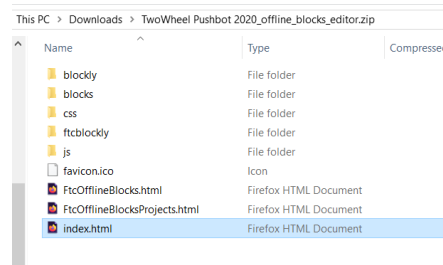


# Downloadable Blocks Editor

--



Download a Blocks editor unique to your current robot configuration.



Extract the zip file and open index.html



**Edit blocks!**  
Make sure to re-upload files to the actual robot controller when you're done.



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

---

# Conclusion

# Our Team's Recommendations

Our team's advice, based on past experience:

1. Have two programmers: one for autonomous, and one for TeleOp
2. Back up programs frequently, especially before competition!  
(Download the OpMode file, and save it to a shared team folder, with version numbers 1.0 for major changes, and 0.1 for minor changes)
3. Work on defining and using variables throughout your code to make the program easier for judges to understand
4. Comment EVERY section of code or function!



**EAGLE  
ROBOTICS**

Introduction to FTC Blocks Programming

---

## Wrap-Up

### Contact Information:



Team Email: [team7373robotics@gmail.com](mailto:team7373robotics@gmail.com)



Coach's Email: [bsmith@mtparanschool.com](mailto:bsmith@mtparanschool.com) (Brad Smith)



Team Website: [eaglerobotics.net](http://eaglerobotics.net)

### Blocks Resources:



[eaglerobotics.net/blocksprogramming](http://eaglerobotics.net/blocksprogramming)