



FIRST[®]
GLOBAL

Block Programming Guide

REV
R O B O T I C S

TABLE OF CONTENTS

1	Getting Started.....	1
1.1	Prerequisites	1
2	Introduction.....	2
2.1	What is an OpMode?	2
2.2	Blocks Programming Interface.....	2
3	Block Programming Concepts.....	3
3.1	Set-up	3
	Default Op Mode Description	4
3.2	Op Mode to Control a DC Motor	5
4	Control a Servo Motor	12
4.1	Set-up	12
4.2	What is a Servo Motor?	12
4.3	Joystick Mapping Strategy.....	12
4.4	Op Mode to Control a Servo Motor.....	14
5	Sensors – Color Sensor Example	19
5.1	Set-up	19
5.2	What is a Sensor?	19
5.3	Op Mode to Read I2C Color Sensor	19
6	Sample Op Modes	20
6.1	Basic Tank Drive	20
7	Troubleshooting.....	21
7.1	Troubleshooting Tips	21

Revision	Release Date	Document Changes
Rev 0	3/31/2017	Initial Release
Rev 1	5/7/2018	Rev 1 Release
Rev 2	7/22/2018	Rev 2 Release

1 Getting Started

1.1 Prerequisites

In order to complete the exercises that are contained in this document, it is assumed you have done the following:

1. Read and complete all steps in the “Control System Startup Guide” from the [FIRST Global Website](#).
2. Be familiar with setting up a configuration via the “Configure Robot” menu selection on the Driver Station.

2 Introduction

The *FIRST* Global Challenge uses an Android-based control system for its competition robots. Students who participate in the *FIRST* Global Challenge can use the *blocks programming* interface to customize the behavior of their competition robots. This document provides an overview of the blocks programming interface. You will learn how to create, edit and deploy op modes using this development tool.

2.1 What is an OpMode?

It can be helpful to think of an op mode as a list of tasks for the Robot Controller to perform. The Robot Controller will process this list of tasks sequentially. During a *FIRST* Global Challenge match, a team's robot has to perform a variety of tasks in an effort to score points. For example, a team might want their robot to autonomously follow a line on the competition floor during a match to help them score points. Teams write "op modes" (which stand for "operational modes") to specify the behavior of their robot.

2.2 Blocks Programming Interface

The "*blocks programming interface*" is powered by Google's *Blockly* software. It is a user friendly, drag and drop, programming tool. Jigsaw-shaped programming blocks are organized on a design "canvas". Op modes are formed by arranging these blocks to create program logic. The resulting op modes are saved onto the Robot Controller.

In Figure 1 the main body of the op mode is defined by the outer purple bracket that has the words "to runOpMode". As the help tip indicates, this block is run when this op mode, "MyFIRSTOpMode" in this example, is run from the Driver Station.

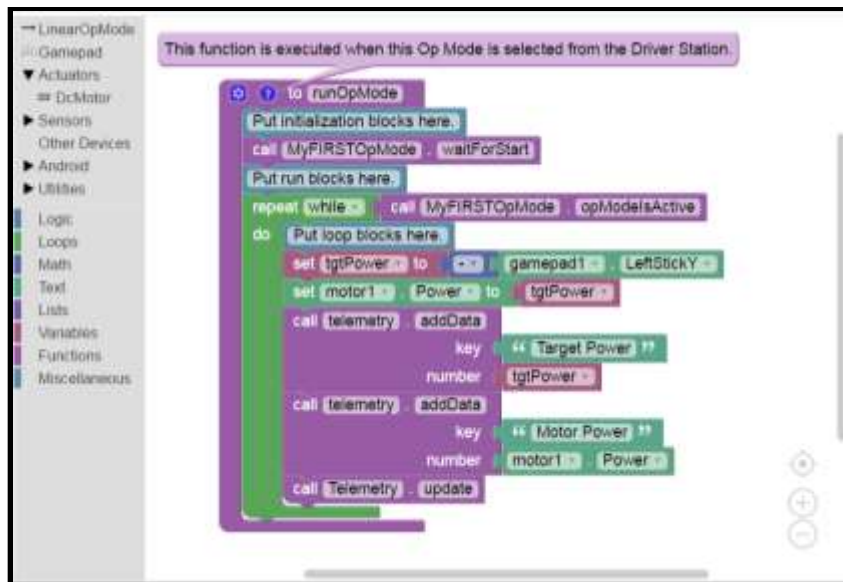


Figure 1 - Arrange jigsaw-shaped programming blocks to create the logic op modes

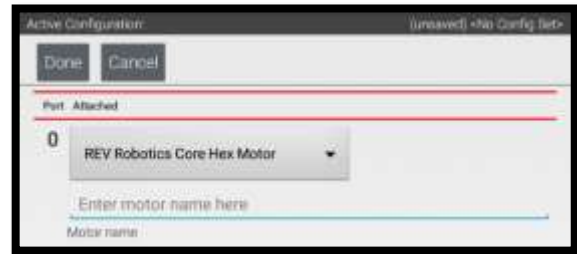
3 Block Programming Concepts

3.1 Set-up

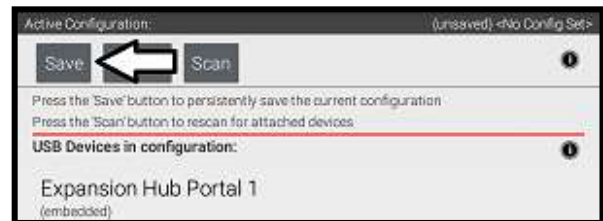
Create a new Robot Configuration with one motor:

Assign a "REV Robotics Core Hex Motor" to "port 0"

Type "motor1" where it says 'enter motor name here' to name the motor



Save the configuration, name it "firstConfig"

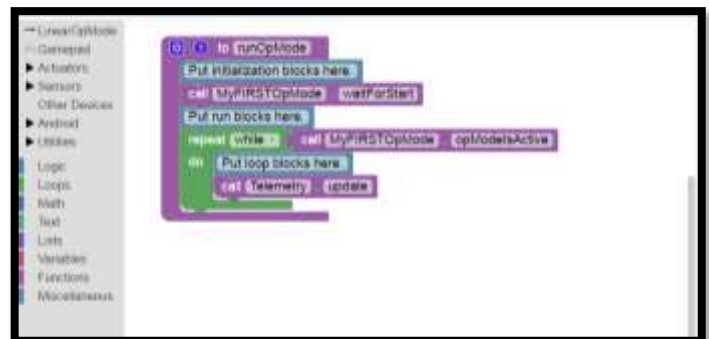


Enter the programming interface either on the driver station tablet or on a computer.

Select the "Create New Op Mode" button at the top of the projects page and name the Op Mode MyFIRSTOpMode



A default block is created.



Default Op Mode Description

The default Op Mode is shown in Figure 2. The default op mode creates a basic program structure which can be updated to create a custom robot task.

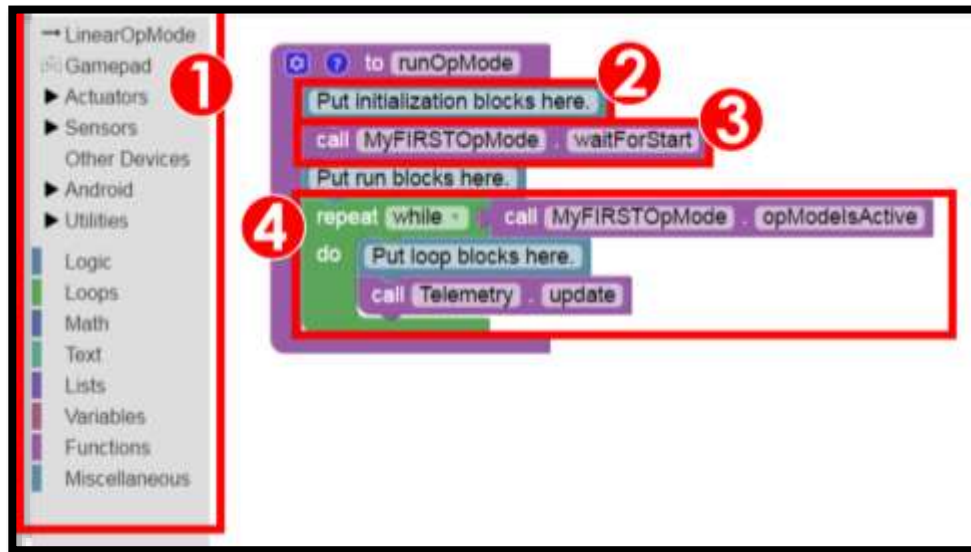


Figure 2: Default Op Mode

1. The gray area to the left is called the ToolBox. The ToolBox contains libraries. Each library contains a list of associated blocks
2. Blue blocks are always comments. Comments are placed in an op mode to make it easier for humans to read. Add comments to describe what the following code does

Comments are not read by the robot.

Initialization programming blocks are placed after the “Put initialization blocks here” comment, and before the “call MyFIRSTOpMode.waitForStart” block. These blocks run after the INIT button on the Driver Station is pressed
3. When the Robot Controller reaches the block labeled “call MyFIRSTOpMode.waitForStart”, it will wait for the Start button on the Driver Station to be pressed. Any subsequent code will be run after the Start button has been pressed
4. The green block labeled “repeat while call MyFirstOpMode.opModeIsActive” is an iterative, looping control structure. This green control block will run the blocks in the “do” portion of the block as long as the condition “call MyFIRSTOpMode.opModeIsActive” is true. Blocks included in the “do” portion of the “repeat while” block will be run repeatedly from top to bottom. After the Stop button on the Driver Station is pressed, “call MyFIRSTOpMode.opModeIsActive” is false, ending the repetition

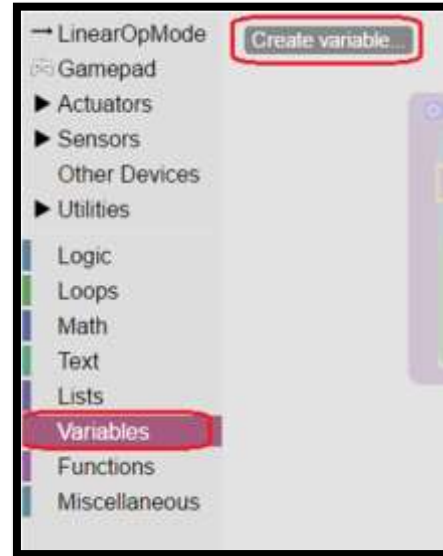
3.2 Op Mode to Control a DC Motor

Now that we have a default Op Mode, it needs to be updated to make the robot do something. As an example, we are going to modify the default Op Mode to control a DC motor using a gamepad.

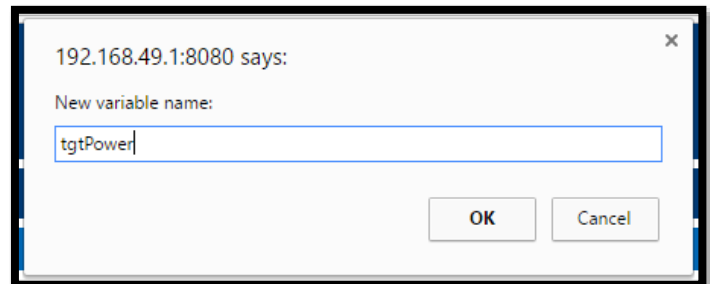
Edit Default Op Mode to Control a DC Motor

1. In the ToolBox select the "Variables" library

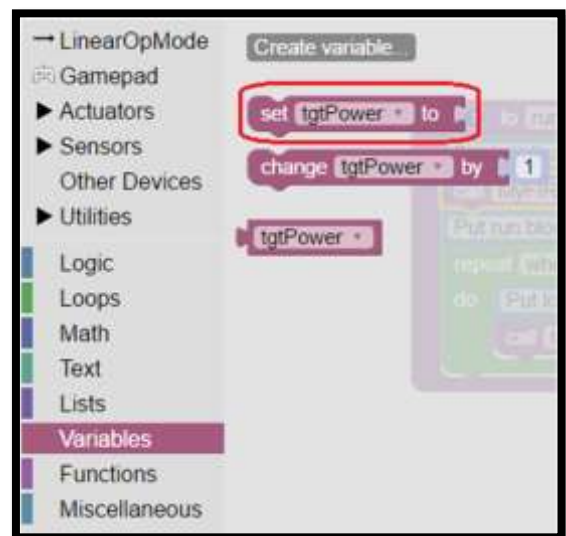
Select the initial option, "Create variable" to create a new variable



2. When prompted, name the variable "tgtPower"

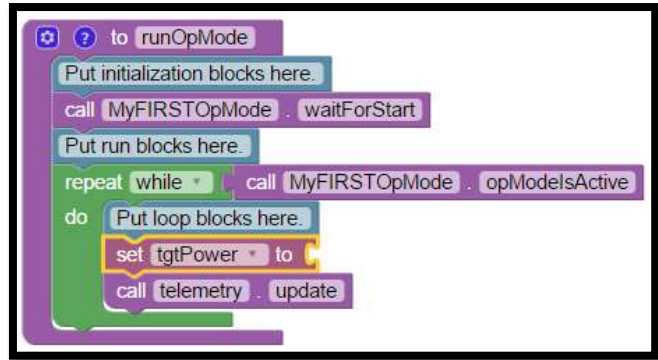


3. After creating a new variable, additional blocks appear in the "Variables" library list



4. Click on “set tgtPower to” and drag the block after the “Put loop blocks here” comment

The “set tgtPower to” block will snap into position

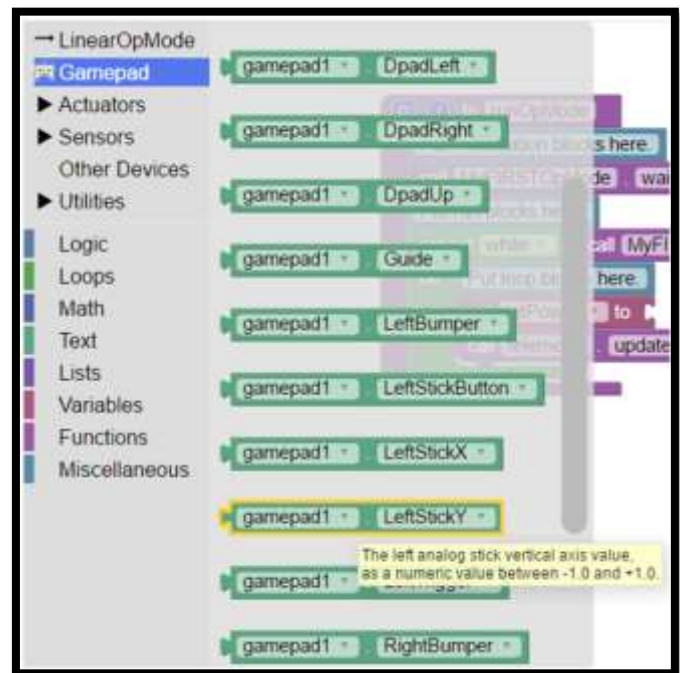


Now that we have declared (created) a variable, we need to assign it a value. In this example we want the variable “tgtPower” to hold the value of the Y joystick value. This will allow the human to move a joystick and a motor on the robot will move in response.

5. In this example, we will assign the gamepad left joystick Y value to control the speed of the motor

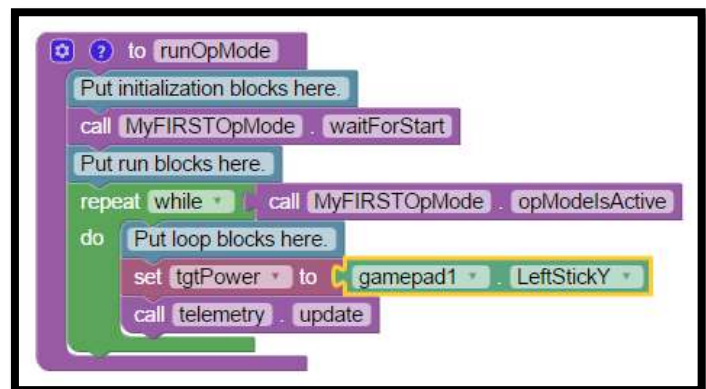
Select the “Gamepad” library

The select the “gamepad1.LeftStickY” block



6. Drag the “gamepad1.LeftStickY” block onto the right side of the “set tgtPower to” block

Now “tgtPower” is repeatedly set to the Y value of the left joystick.



- On the gamepad the Y value of joystick ranges from -1 to +1, top to bottom respectively. With the left joystick in the top position, the variable "tgtPower" is set to -1.

Positive valued motor power results in forward motion. Negative valued motor power results in reverse motion.

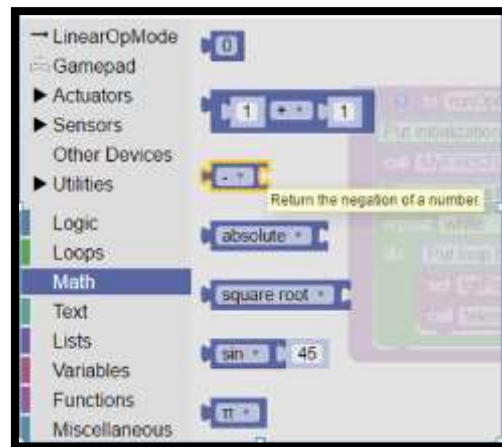
With this code, full forward on the joystick would make the motor go full speed in reverse.



- Since it would make more sense for the motor to drive forward at full power with the joystick in the top position, and full power in reverse when the joystick is in the bottom position, we need to invert the joystick Y value.

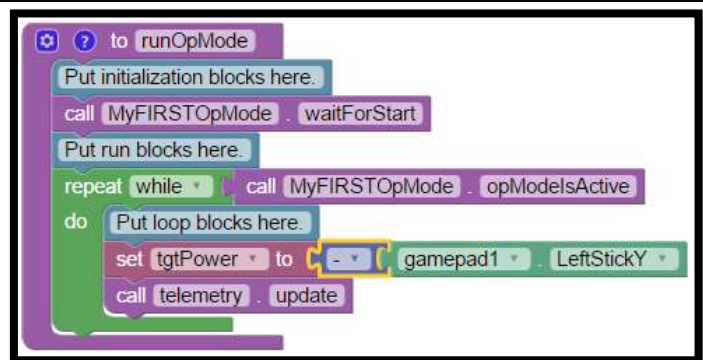
Click on the "Math" library

Select the negative (-) symbol



- Drag the negative symbol into place between the set "tgtPower" variable and the "gamepad1.LeftStickY" block

Now the variable "tgtPower" is set to +1 with the joystick in the topmost position, and -1 with the joystick in the bottommost position



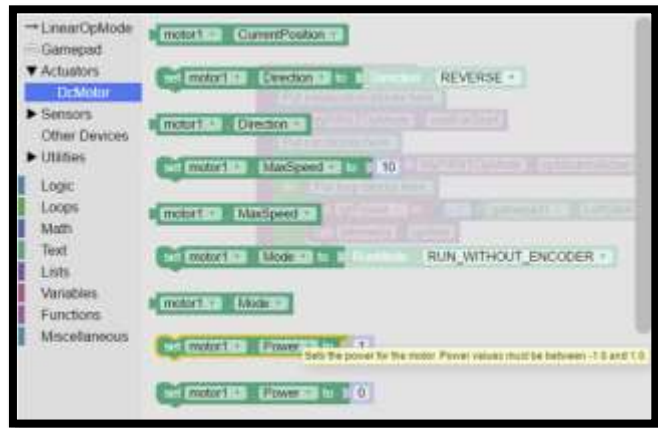
Above we have created code that captures the value of the Y joystick, this is the user input, inverts the value and assigns it to a variable "tgtPower". Now we need to add code that sets the motor output to be assign to "tgtPower"

- Click on the "Actuators" library

Select the "DcMotor" library

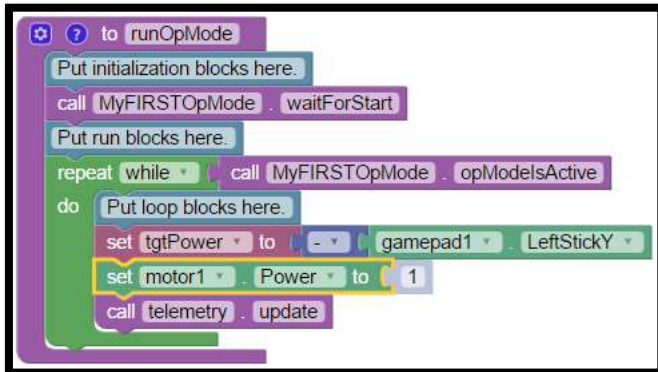


11. Select the “set motor1.Power to 1” programming block



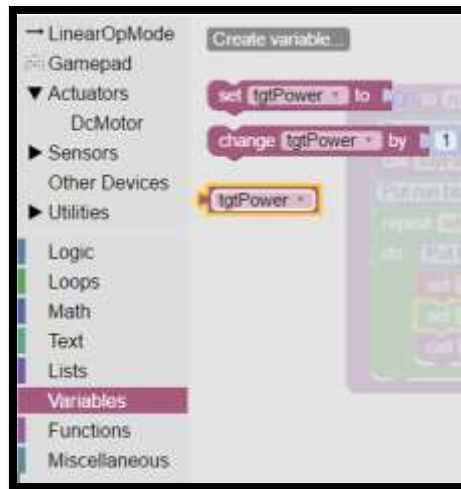
12. Drag the “set motor1.Power to 1” block below the “set tgtPower to” block

The “set motor1.Power to” has a default value of 1. The motor power will be set to 1. Valid motor power values range from -1 to +1.

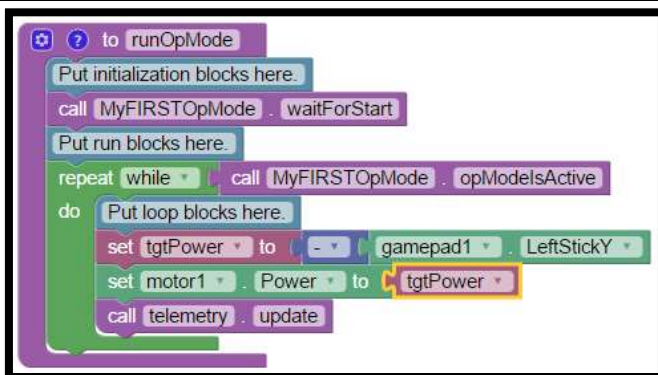


13. Select the “Variables” library

Select the “tgtPower” block



14. Replace the default value with the “tgtPower” block



This op mode now maps the position of the left joystick Y axis to a speed and direction of motor1.

3.2.1 Insert Telemetry Statements

It is a good practice to display information about the status of the robot on the driver station tablet using Telemetry statements. This telemetry code can be used to display things like sensor data, motor status, gamepad state, etc.

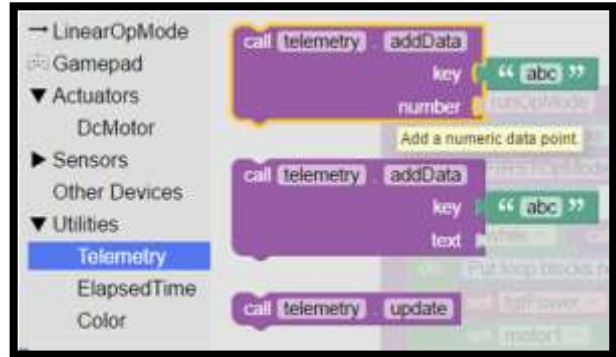
Building on the motor control example in the previous section, we can add telemetry statements to display the value of the variable "tgtPower" on the driver station screen.

Use Telemetry Statements to Display Data on the Driver Station

1. Select the "Utilities" library from on the ToolBox

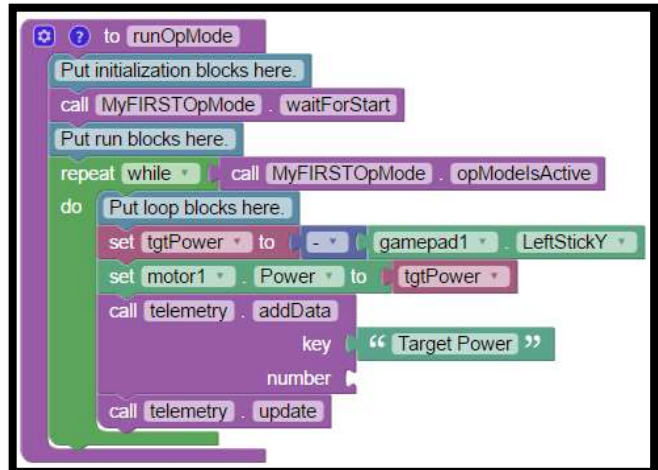
Select the "Telemetry" library

Select the "call telemetry.addData(key, number)" block



2. Drag the "call telemetry.addData(key, number)" block below the "set motor1.Power to" block

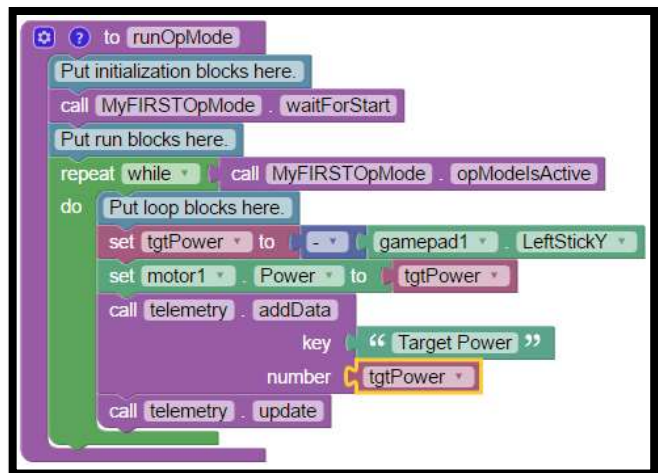
Select the green text block "abc" and edit the text to read "Target Power"



3. Select the "Variables" library

Select the "tgtPower" block

Drag the block to the "number" parameter on the telemetry programming block



The Robot Controller will send the value of "tgtPower" to the Driver Station with a key (label), "Target Power"

Now we will add another value to display, actual motor power.

- Repeat this process and name the new key "Motor Power"

```

to runOpMode
  Put initialization blocks here.
  call MyFIRSTOpMode . waitForStart
  Put run blocks here.
  repeat while call MyFIRSTOpMode . opModelsActive
  do Put loop blocks here.
    set tgtPower to gamepad1 . LeftStickY
    set motor1 . Power to tgtPower
    call telemetry . addData
      key "Target Power"
      number tgtPower
    call telemetry . addData
      key "Motor Power"
      number motor1 . Power
    call telemetry . update
  
```

- Use the numerical value of motor power for the telemetry number parameter

Select "Actuators"

Select "DcMotor"

Select "motor1.Power"

- Drag the "motor1.Power" block to the "number" parameter the telemetry block

```

to runOpMode
  Put initialization blocks here.
  call MyFIRSTOpMode . waitForStart
  Put run blocks here.
  repeat while call MyFIRSTOpMode . opModelsActive
  do Put loop blocks here.
    set tgtPower to gamepad1 . LeftStickY
    set motor1 . Power to tgtPower
    call telemetry . addData
      key "Target Power"
      number tgtPower
    call telemetry . addData
      key "Motor Power"
      number motor1 . Power
    call Telemetry . update
  
```

The driver station will now display the target power (from the left Y joystick) and the actual motor1 power.

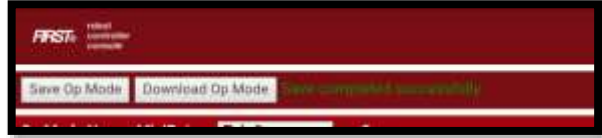
3.2.2 Save and Run Your Op Mode

Save and Run your Op Mode

1. Select "Save Op Mode"

Op mode is saved to the Control Hub

"Save completed successfully" appears in green text



If you are having a problem saving your op mode, refer to the [troubleshooting section](#) of this guide.

2. Run your op mode, this is covered in the "Control System Startup Guide".

4 Control a Servo Motor

4.1 Set-up

Edit your “`firstConfig`” Robot Configuration to add a Servo.

1. Assign a “Servo” to “port 0”
2. Name the servo “`servo1`”
3. Save the configuration, keeping the name “`firstConfig`”
4. Return to edit “`MyFIRSTOpMode`” by selecting the op mode from the “Projects” page. If you are using a laptop, refresh your browser to update the configuration changes.

4.2 What is a Servo Motor?

A servo motor is a special type of motor. A servo motor is designed for precise motion. A typical servo motor has a limited range of motion. Figure 3 is a “standard scale” 270-degree servo. The servo in your FIRST Global kit is a Smart Robot Servo which has some additional features, but by default the servo motor can rotate its shaft through a range of 270 degrees. Using an electronic module known as a *servo controller* you can write an op mode that will move the servo motor to a specific position (angle). Once the motor reaches this target position, it will hold the position, even if external forces are applied to the shaft of the servo.



Figure 3: Servo Motor

4.3 Joystick Mapping Strategy

With a typical servo, you can specify a target position for the servo and the servo will turn moving the output shaft to the target position (angle). It will maintain that position, even if moderate forces are applied trying to move the servo off target. Target positions range from 0 to 1. A target position of 0 corresponds to zero degrees of rotation, and a target position of 1 corresponds to 270 degrees of rotation (Figure 4).

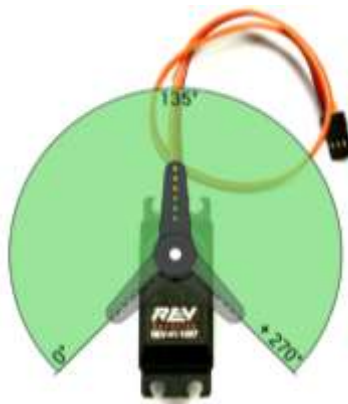


Figure 4: The Smart Robot Servo Motor can rotate to and hold a position from 0 to 270 degrees

For this example, we will use the right joystick Y-axis to control the motion of the servo. Recall joystick values range from -1 to +1, so the first thing we need to do is rescale the joystick range (-1 to +1) to match the servo range (0 to +1).

Joystick Range Scaling Example

Rescale the joystick range from $+1 \rightarrow -1$ to $0 \rightarrow +1$

1. Standard Joystick Range

Input

$+1 \rightarrow -1$



2. Invert the Range

Input $\times (-1)$

$-1 \rightarrow +1$



3. Translate the Range

Input $\times (-1) + 1$

$0 \rightarrow +2$



4. Scale the Range

$\frac{\text{Input} \times (-1) + 1}{2}$

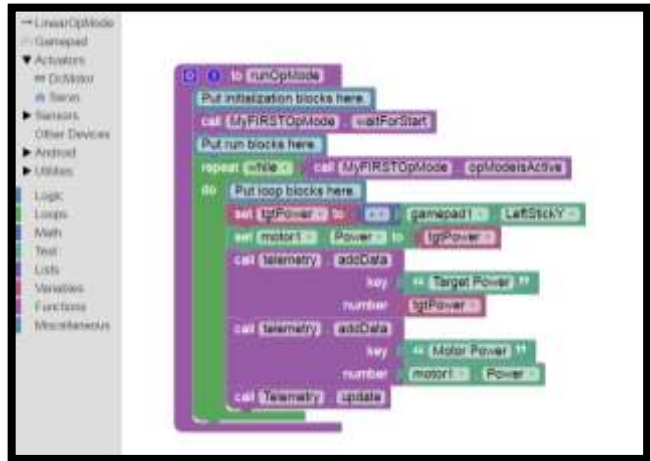
$0 \rightarrow +1$



4.4 Op Mode to Control a Servo Motor

Adding Servo Control using a Joystick

1. Using the "MyFIRSTOpMode" op mode from the previous section, find the "Servo" library listed under "Actuators"



2. Create a new variable to keep track of the target servo position

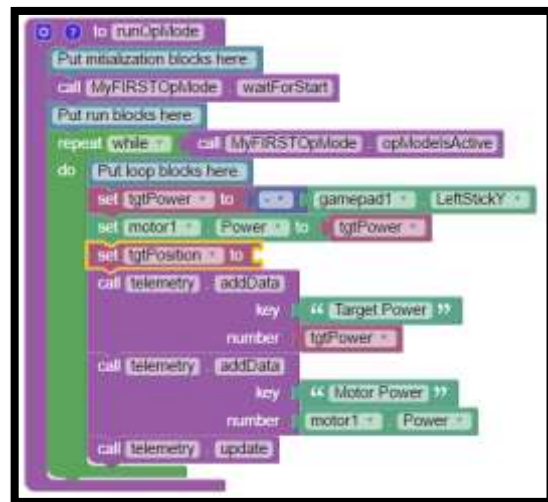
From the blocks Tool Box select the "Variables" library

Select "Create variable" and create a new variable called "tgtPosition"



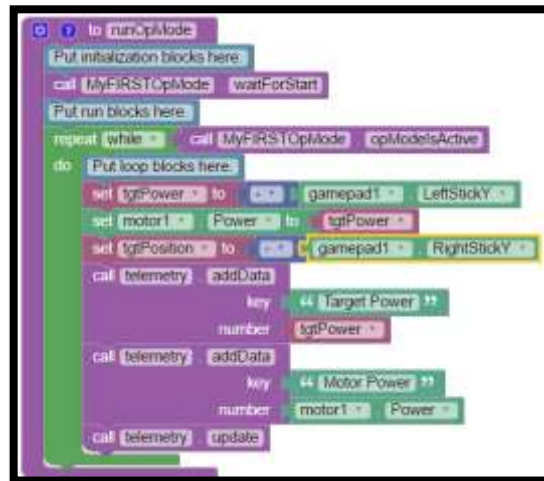
3. Select the "set tgtPosition to" block

Drag it below the "set motor1.Power to" block in the while loop



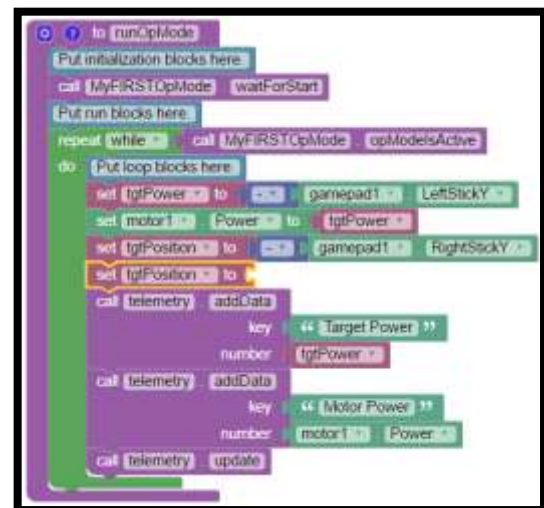
- Use the design blocks that are contained in the "Math" and "Gamepad" libraries

Set the "tgtPosition" variable to the negative value of the right joystick

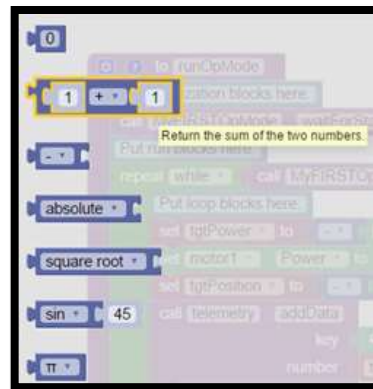


- Select another "set tgtPosition to" block

Drag it to the position below the first "set tgtPosition to" block



- Select the "Math" library
Select the "addition" block



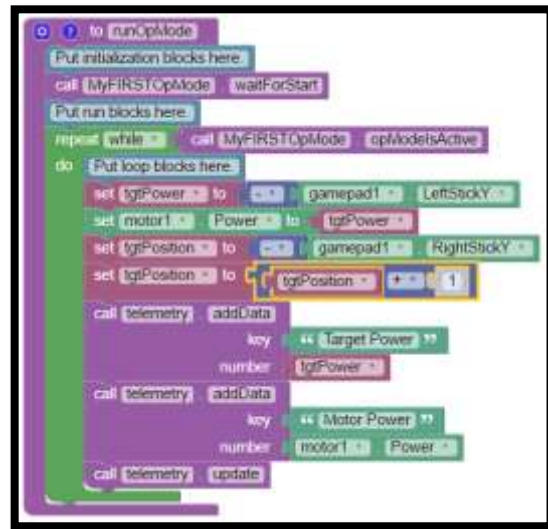
- Drag the addition block to the right-hand side of the second "set tgtPosition to" block



8. Select the "Variables" library

Select "tgtPosition"

Drag the "tgtPosition" block to the left argument of the binary expression block

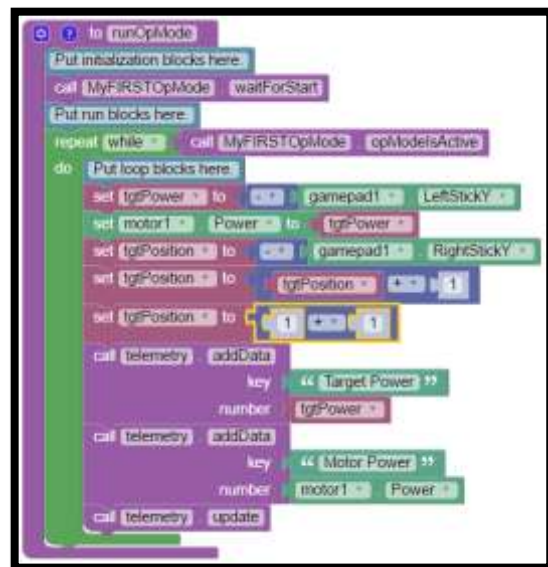


9. Select the "Variables" library

Drag a third "set tgtPosition to" block into the op mode

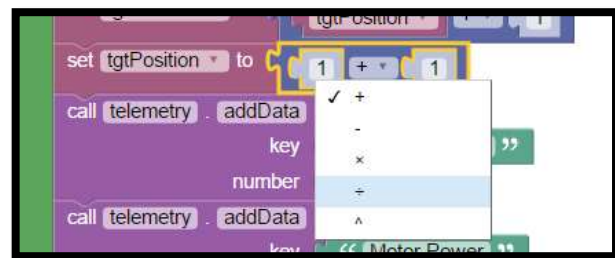
Select the "Math" library

Drag a binary expression block onto the "set tgtPosition to" block

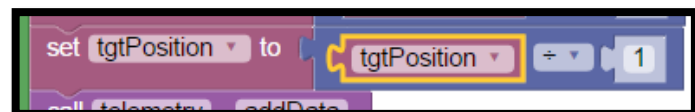


10. Select the binary operator menu of the "addition" binary expression block.

Select the division symbol "÷".



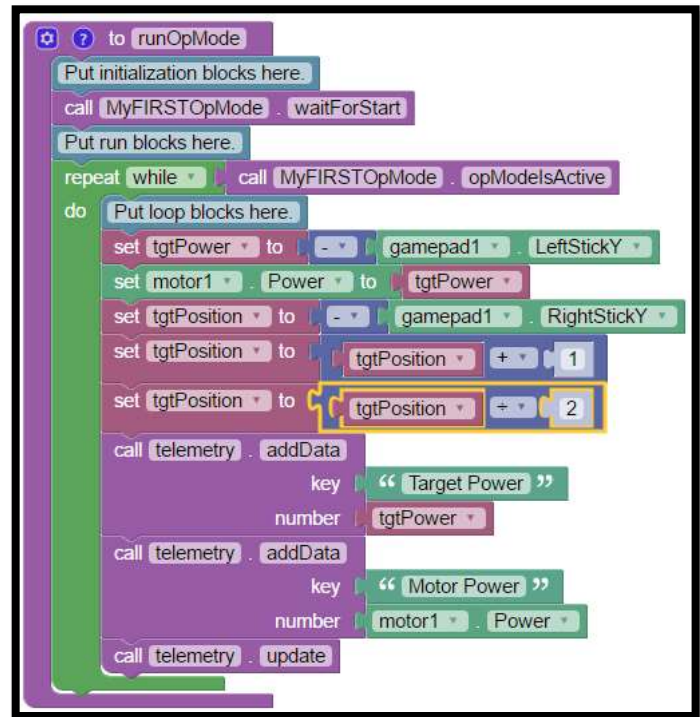
11. Drag a "tgtPosition" block to the left argument of the division block



12. Change the right operand to 2



13. Your op mode should look like this

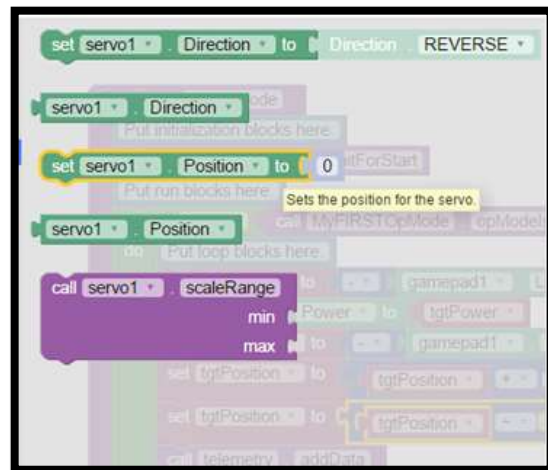


14. Set the servo position to the value of the variable "tgtPosition".


Select the "Actuators" library

Select the "Servo" library

Select the "set servo1.Position to" block



15. Drag the "set servo1.Position to" block below the third "set tgtPosition to" block



16. Select the "Variables" library

Drag a "tgtPosition" block to the right side of the "set servo1.Position to" block

```
to runOpMode
  Put initialization blocks here
  call MyFIRSTOpMode.waitForStart
  Put run blocks here
  repeat while call MyFIRSTOpMode.opModelsActive
  do Put loop blocks here
    set tgtPower to gamepad1.LeftStickY
    set motor1.Power to tgtPower
    set tgtPosition to gamepad1.RightStickY
    set tgtPosition to tgtPosition + 1
    set tgtPosition to tgtPosition + 2
    set servo1.Position to tgtPosition
    call telemetry.addData
      key "Target Power"
      number tgtPower
    call telemetry.addData
      key "Motor Power"
      number motor1.Power
    call telemetry.update
```

17. Modify your op mode to include two additional telemetry blocks

```
to runOpMode
  Put initialization blocks here
  call MyFIRSTOpMode.waitForStart
  Put run blocks here
  repeat while call MyFIRSTOpMode.opModelsActive
  do Put loop blocks here
    set tgtPower to gamepad1.LeftStickY
    set motor1.Power to tgtPower
    set tgtPosition to gamepad1.RightStickY
    set tgtPosition to tgtPosition + 1
    set tgtPosition to tgtPosition + 2
    set servo1.Position to tgtPosition
    call telemetry.addData
      key "Target Power"
      number tgtPower
    call telemetry.addData
      key "Motor Power"
      number motor1.Power
    call telemetry.addData
      key "Target Position"
      number tgtPosition
    call telemetry.addData
      key "Servo Position"
      number servo1.Position
    call telemetry.update
```

Save your op mode and run it

5 Sensors – Color Sensor Example

5.1 Set-up

Edit your “firstConfig” Robot Configuration to add a Color Sensor.

1. Select I2C Bus 1
2. Add a “REV Color Sensor V2”
3. Name the sensor “color1”
4. Save the configuration, keeping the name “firstConfig”
5. Return to edit “MyFIRSTOpMode” by selecting the op mode from the “Projects” page. If you are using a laptop, refresh your browser to update the configuration changes.

5.2 What is a Sensor?

A sensor is a device that measures a physical property about its environment and responds in some fashion.

5.3 Op Mode to Read I2C Color Sensor

Reading from an I2C Color/Range Sensor

1. Add a telemetry block with a number argument
2. Add “Distance” as the key to the telemetry block
3. Select the “Sensors” library
4. Select the LynxI2cColorRangeSensor
5. Select the “call color1.getDistance” block
6. Drag the “cal color1.getDistance” block to the number argument of the telemetry block.

```
to runOpMode
  Put initialization blocks here.
  call MyFIRSTOpMode . waitForStart
  Put run blocks here.
  repeat while call MyFIRSTOpMode . opModelsActive
  do
    Put loop blocks here.
    set tgtPower to -- gamepad1 . LeftStickY --
    set motor1 . Power to tgtPower
    set tgtPosition to -- gamepad1 . RightStickY --
    set tgtPosition to tgtPosition + 1
    set tgtPosition to tgtPosition + 2
    set servo . Position to tgtPosition
    call telemetry . addData
      key "Target Power"
      number tgtPower
    call telemetry . addData
      key "Motor Power"
      number motor1 . Power
    call telemetry . addData
      key "Target Position"
      number tgtPosition
    call telemetry . addData
      key "Servo Position"
      number servo . Position
    call telemetry . addData
      key "Distance"
      number call color1 . getDistance
      unit DistanceUnit . CM
    call Telemetry . update
```

6 Sample Op Modes

NOTE: These examples do not build on the previous example code.

6.1 Basic Tank Drive

The following op mode assumes that you have two DC motors called “left_drive” and “right_drive” and that the “right_drive” motor should be reserved so that a positive power setting for that motor will move the robot forward.

The op mode shown in Figure 5 uses a tank drive scheme. The left joystick (y-axis direction) controls the left motor and the right joystick (y-axis direction) controls the right motor. To move the robot forward, the user pushes both joysticks forward. To turn to the left, the user pushes the right joystick forward and pulls the left stick backwards.

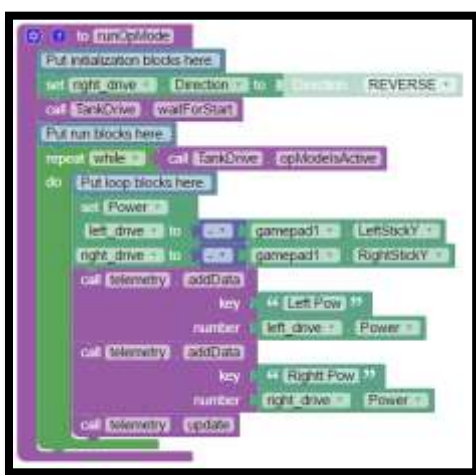


Figure 5: TankDrive Sample Op Mode

This op mode and a few other samples are found preconfigured on the Control Hub. When creating a new Op Mode pressing the drop down menu label Sample: will set the sample Op Mode in Blocks for editing.



Figure 6: Selecting Sample Op Modes

7 Troubleshooting

7.1 Troubleshooting Tips

In this section we provide some basic tips on troubleshooting problems that you might encounter when using the blocks programming mode server to write op modes for your Robot Controller.

7.1.1 I do not see my Wi-Fi network name on my laptop.

For some Windows devices, the laptop might not display your blocks programming mode Wi-Fi network in its list of available networks. We believe that this problem occurs with some Windows 10 machines (and also possibly with some Windows 8 machines).

If you are having problems seeing your Hosted Programming Wi-Fi network in your list of available networks, make sure that your Driver Station is paired and connected to your Robot Controller (see “Control System Startup Guide” from the [FIRST Global Website](#)). Also, make sure that your Windows device has its most current updates installed from Microsoft.

If this does not resolve the issue, then you might have to manually connect your Windows computer to the blocks programming mode Wi-Fi network.

You can manually connect to this network as if the network were a *hidden* network (i.e., a network that does not broadcast its presence to other Wi-Fi devices). In the lower right hand corner of the Windows 10 desktop, click on the network icon in the system tray (see Figure 7) to display a list of available Wi-Fi networks. If you still do not see your blocks programming mode network listed, then scroll to the bottom of the list and look for the item “Hidden Network”.

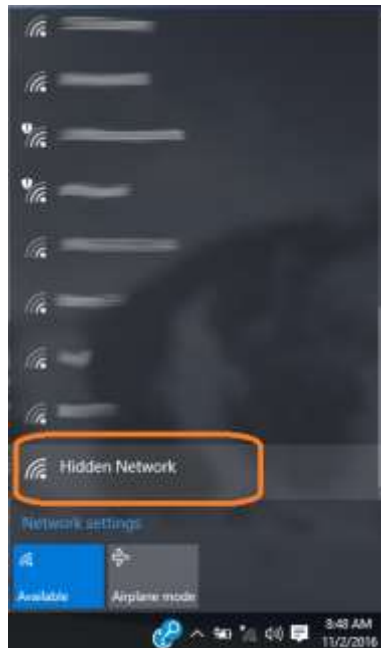


Figure 7 – “Hidden Network” Option is Listed at the bottom of the Window’s Wi-Fi List

Click on the “Hidden Network” listing to start the connection process. The listing should display a “Connect” button. Make sure the option “Connect automatically” is checked and then click on the “Connect” button to continue with the process (Figure 8).

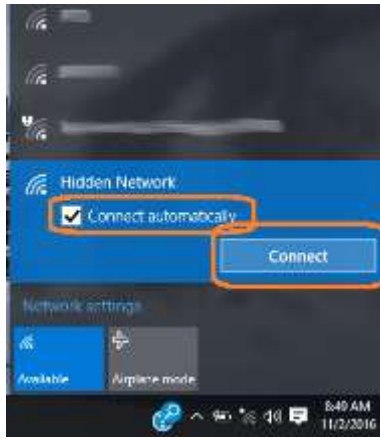


Figure 8 – Check “Connect automatically” and click “Connect”

The computer should prompt you for the name or *SSID* of your blocks programming mode Wi-Fi network (Figure 9). You should type in the network name that is displayed in the Programming Mode window of the Android device (see “Control System Startup Guide” from the [FIRST Global Website](#)). Note, the SSID or network name is *case sensitive*. Make sure the capitalization of the name that you enter matches the capitalization of the name displayed in the Programming Mode Window.



Figure 9 - Enter in the name of the blocks programming mode Wi-Fi network

The computer should then prompt you for the passphrase to access this Wi-Fi network (Figure 10). You should type in the network passphrase that is displayed in the Programming Mode window of the Android device (see “Control System Startup Guide” from the [FIRST Global Website](#)). Note that the passphrase is *case sensitive*. Make sure that your spelling and capitalization matches the original spelling and capitalization shown on the Programming Mode screen.

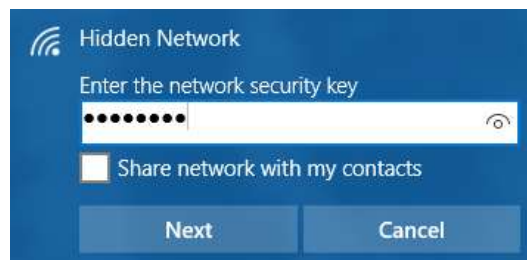


Figure 10 – Enter the passphrase for the blocks programming mode network

Your computer will prompt you on whether or not you want to make your PC discoverable by other devices on this network. Click “Yes” to continue (Figure 11).

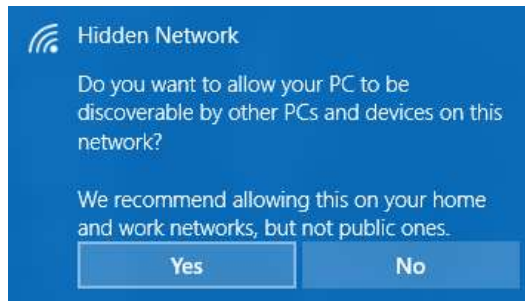


Figure 11 - Click on "Yes" to allow other devices to be able to discover your computer on this network

The computer will attempt to connect to your network. Note that it could take several minutes before it connects (Figure 12).

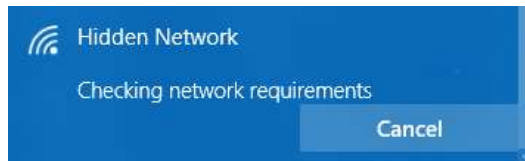


Figure 12 – The computer will check the network requirements and attempt to connect to the network

If you were able to successfully connect to the network, it will appear in the list of networks on your computer (Figure 13). This connection could take up to 60 seconds to establish. Note that when your computer is connected to the blocks programming mode Wi-Fi network, *it will not have access to the Internet.*

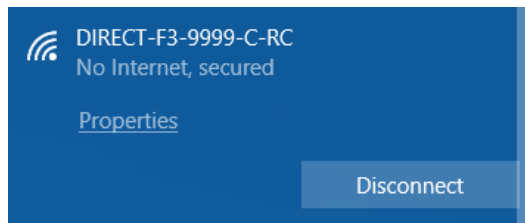


Figure 13 –Windows notification of successful connection to the blocks programming mode

7.1.2 “Save Project Failed, Error code 0.”

If you attempt to save the op mode that you are currently editing, but you receive an error message indicating that the “Save project failed. Error code 0.” you might have not be connected to the blocks programming mode sever (Figure 14).

To correct this issue, you will need to reconnect to the blocks programming server on the Control Hub.

1. Make sure that your laptop is connected to the blocks programming mode Wi-Fi network
2. Press the “Save Op Mode” button again to re-attempt the save operation.

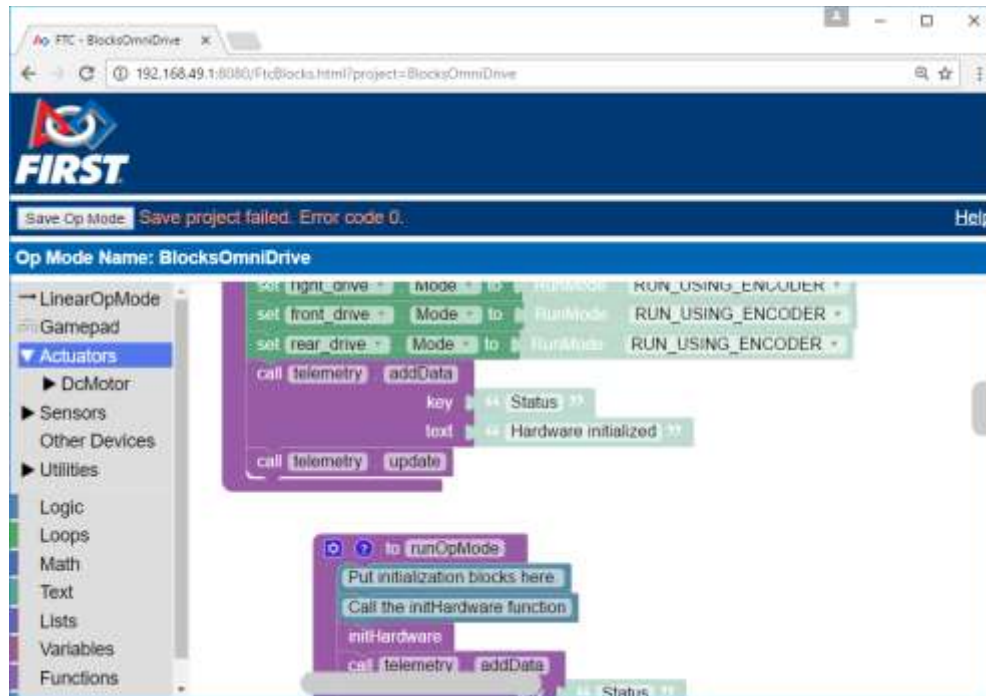


Figure 14: Save Project Failed Error

7.1.3 Op Mode Blocks are Missing

If you have opened an existing op mode to edit it in your Javascript-enabled browser, but the programming blocks are missing, check the following:

1. Did you remember to save the op mode the last time you edited and then exited the op mode? If you did not save the op mode after the last editing session, you might have lost some of your changes.
2. Are the blocks *collapsed* and/or in an area of the design “canvas” (or design pane) that is outside your current browser window? If so, you can use the *expand* and *cleanup* functions of the blocks programming tool to expand all of the blocks on your screen and to organize them in an easy-to-view (and easy-to-find) manner (Figure 15 and Figure 16).

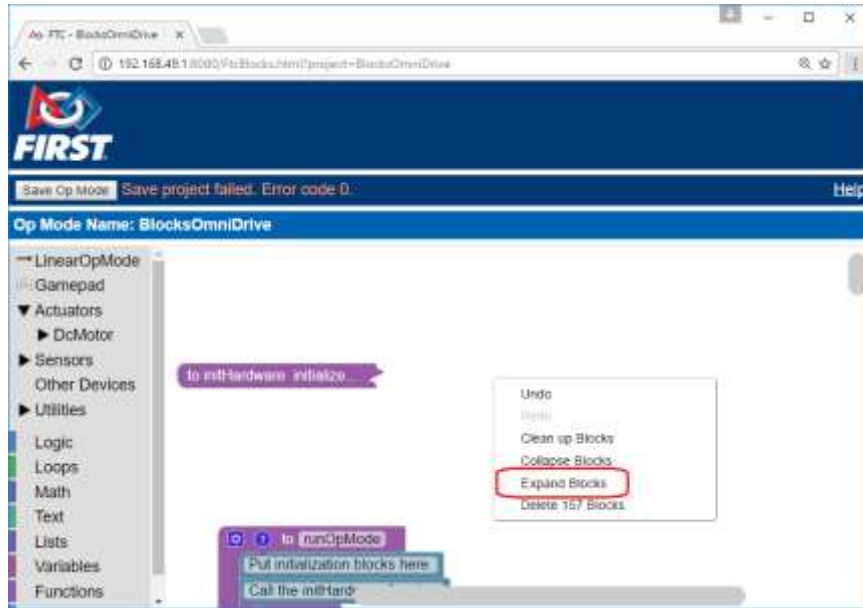


Figure 15: Right click on the canvas to "Expand Blocks"

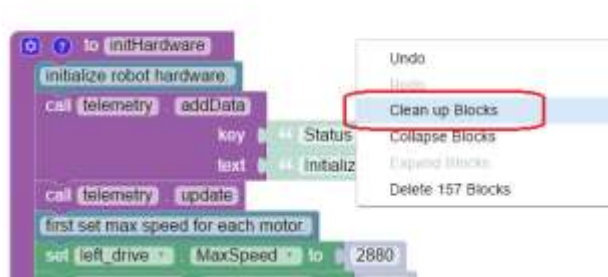


Figure 16: Right click on the canvas to "Clean up Blocks"