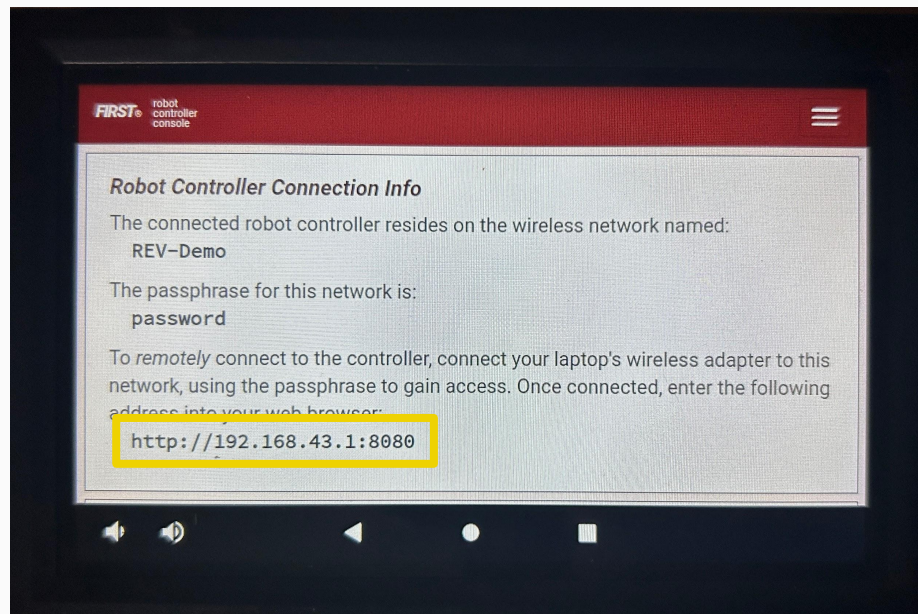
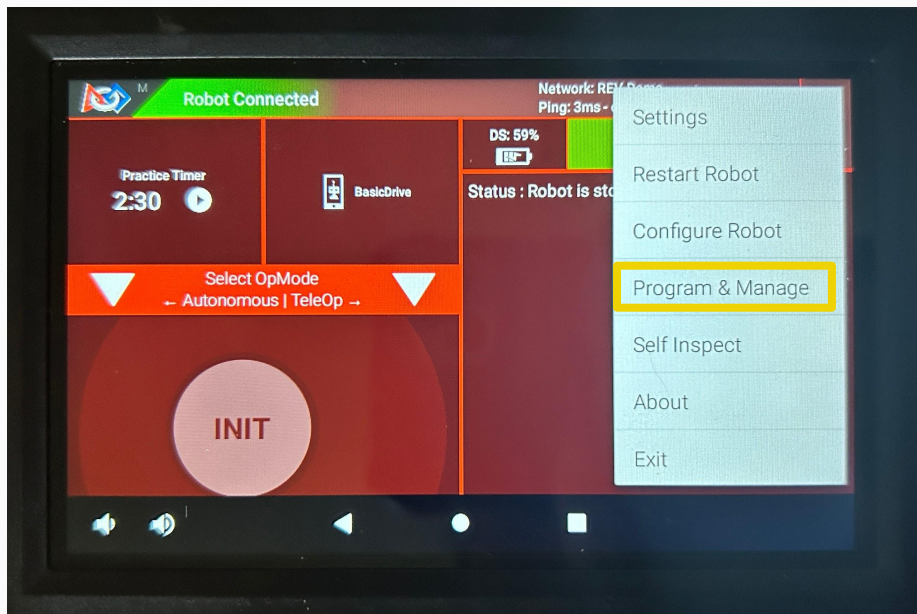


The background is a solid blue color with a repeating pattern of white icons. These icons include various electronic components like gears, resistors, capacitors, and integrated circuits, as well as tools like wrenches, pliers, and screwdrivers. The text is centered and consists of two lines, both underlined.

# REV Electronics Programming Guide

Open the Driver Hub app on the Driver Hub. Click the three dots in the upper right corner and select “Program & Manage.” At the bottom of this screen, you should see an IP address. On your computer, while connected to the WiFi network of your robot, type this into your web browser. This will allow you to connect to the robot’s Control Hub to program it.



You should now see this page with the name of your robot and the network password.  
Click the “Blocks” button at the top of the page.

FIRST<sup>®</sup>  
robot  
controller  
console

Blocks

OnBotJava

Manage

#### ***Robot Controller Connection Info***

The connected robot controller resides on the wireless network named:

REV-Demo

The passphrase for this network is:

password

Robot controller status:

Server OK

Active connections:

REV UI #1    connected

Windows #1    manage.html

Windows #2    connection.html



You are now going to create your first OpMode! Click the “Create New OpMode” button and name the program “BasicDrive.” You will first use this program to make your robot drive straight so you can test if it is wired correctly. Then you will adjust the program to add the ability to drive your robot using your Driver Hub and a controller.

1

**FIRST.** robot controller console **Blocks** OnBotJava Manage

Create New OpMode

Upload OpMode

Download Offline Blocks Editor

Download All OpModes

TensorFlow Lite Models  
Sounds

**My OpModes**

<input type="checkbox"/>	OpMode Name	Date Modified ▼	Enabled
--------------------------	-------------	-----------------	---------

2

Create New OpMode

OpMode Name:

Sample:

Cancel

OK

If done correctly, your screen should now look like this. Uncheck “Show Java” and close the comment by clicking the question mark icon.

The screenshot displays the FIRST robot controller console interface. At the top, there are tabs for "FIRST. robot controller console", "Blocks", "OnBotJava", and "Manage". Below these are buttons for "Save OpMode", "Export to Java", "Download OpMode", and "Download Image of Blocks". The main header shows "OpMode Name: BasicDrive", "TeleOp" selected in a dropdown, "Group:" with an empty field, and a checked "Enabled" checkbox. On the left, a sidebar lists categories: LinearOpMode, Gamepad, Actuators, Sensors, Other Devices, Android, Utilities, Logic, Loop, Math, Text, Lists, Variables, Functions, and Miscellaneous. A yellow callout box labeled "2" points to the "Utilities" category. The main workspace shows a "BasicDrive" OpMode block with a yellow callout box labeled "1" pointing to a question mark icon. The block contains several sub-blocks: "Put initialization blocks here.", "call waitForStart", "if call opModelsActive", "do Put run blocks here.", "repeat while call opModelsActive", "do Put loop blocks here.", and "call Telemetry . update". On the right, the "Java Code:" tab is active, showing the source code for the BasicDrive OpMode. A yellow callout box labeled "1" points to the "Show Java" button in the top right corner of the Java Code tab.

1

2

Java Code:

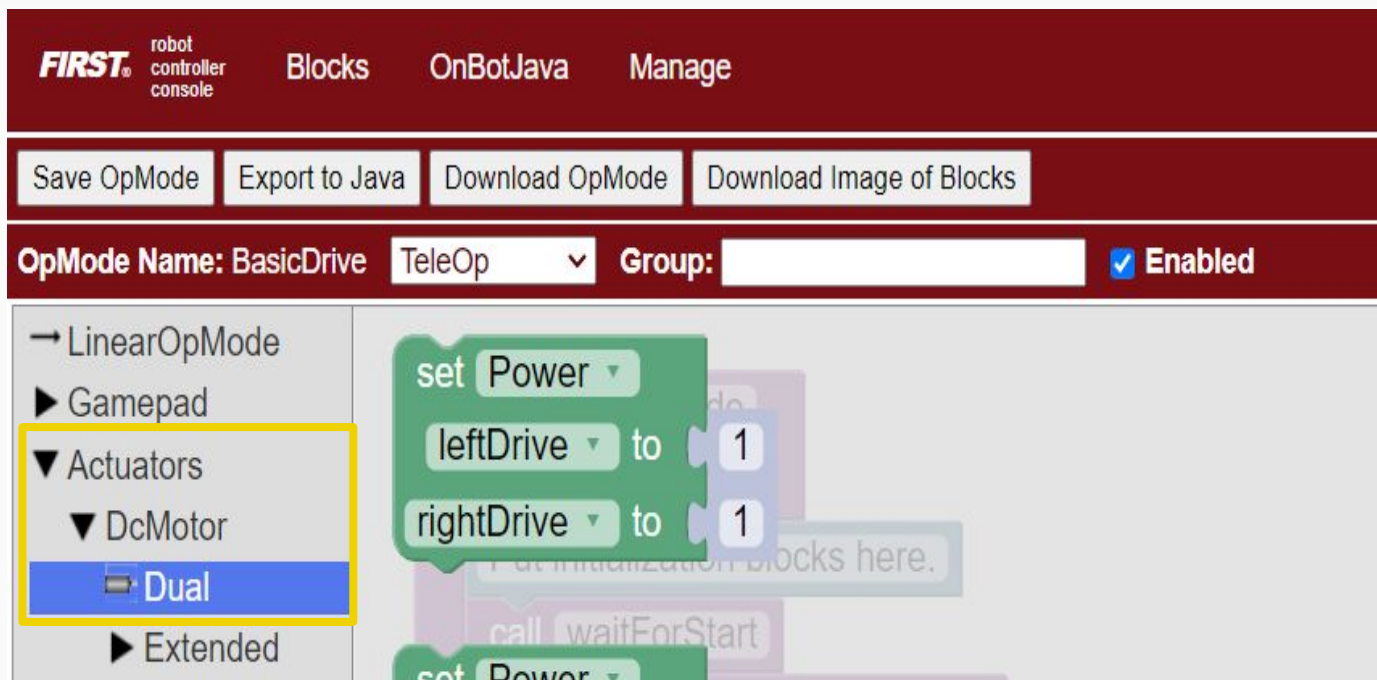
```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.L
import com.qualcomm.robotcore.eventloop.opmode.T

@TeleOp(name = "BasicDrive (Blocks to Java)")
public class BasicDrive extends LinearOpMode {

    /**
     * This sample contains the bare minimum Block
     * Comment Blocks show where to place Initiali
     * DS INIT button, and before touching the DS
     * touching Start), and Loop code (runs repeat
     * Stopped).
     */
    @Override
    public void runOpMode() {
        // Put initialization blocks here.
        waitForStart();
        if (opModeIsActive()) {
            // Put run blocks here.
            while (opModeIsActive()) {
                // Put loop blocks here.
                telemetry.update();
            }
        }
    }
}
```

To start working on your program, click the dropdown for “Actuators,” then “DC Motor.” Finally, click on “Dual.” The blocks in this section are used to control two motors at the same time, which is perfect for your program.



The screenshot displays the FIRST robot controller console interface. At the top, there is a dark red header bar with the text "FIRST. robot controller console" on the left, and "Blocks", "OnBotJava", and "Manage" on the right. Below the header, there is a row of four buttons: "Save OpMode", "Export to Java", "Download OpMode", and "Download Image of Blocks". Underneath these buttons, there is a section for "OpMode Name: BasicDrive", a dropdown menu set to "TeleOp", a "Group:" label followed by an empty text box, and a checked checkbox labeled "Enabled". The main area of the interface is divided into two parts. On the left is a sidebar menu with the following items: "LinearOpMode", "Gamepad", "Actuators" (which is expanded to show "DcMotor" and "Dual", with "Dual" highlighted in blue), and "Extended". On the right is a workspace for block diagrams. It contains several green blocks: a "set Power" block, a "leftDrive" block set to "1", and a "rightDrive" block set to "1". There are also some purple blocks and a text box that says "Put initialization blocks here.".

Drag out the first block and place it in the “repeat while” block as shown. Your program will now command both of your motors to drive at 100% power. For future reference, power is set between -1 and 1, where 1 is 100% power in the forward, or positive direction, and -1 is 100% power in the reverse, or negative direction.

The screenshot shows the FIRST robot controller console interface. At the top, there are tabs for 'Blocks', 'OnBotJava', and 'Manage'. Below these are buttons for 'Save OpMode', 'Export to Java', 'Download OpMode', and 'Download Image of Blocks'. The 'OpMode Name' is set to 'BasicDrive', the mode is 'TeleOp', and the 'Group' is empty. On the left, a sidebar lists categories: LinearOpMode, Gamepad, Actuators, and ServoController. Under 'Actuators', the 'Dual' motor block is selected. In the main workspace, a 'repeat while' loop is visible, and a 'set Power' block for 'leftDrive' to '1' is being dragged into the loop's 'do' section. Another 'set Power' block for 'rightDrive' to '1' is also visible below it.

This screenshot shows the completed 'BasicDrive' OpMode code in the FIRST robot controller console. The 'OpMode Name' is 'BasicDrive', the mode is 'TeleOp', and it is 'Enabled'. The code is structured as follows: a 'to runOpMode' block containing a 'BasicDrive' sub-block. Inside 'BasicDrive', there is a 'Put initialization blocks here.' comment, followed by a 'call waitForStart' block, and an 'if opModelsActive' block. The 'do' section of the 'if' block contains a 'repeat while' loop. The loop's condition is 'call opModelsActive', and its 'do' section contains a 'Put loop blocks here.' comment, followed by 'set Power' blocks for 'leftDrive' to '1' and 'rightDrive' to '1', and a 'call Telemetry.update' block. A yellow box highlights the 'repeat while' loop and its contents.



As an initial test, let's set both motors to run a 20% power. Change both 1's to 0.2's.

The screenshot displays the FIRST robot controller console interface. At the top, there are tabs for 'Blocks', 'OnBotJava', and 'Manage'. Below these are buttons for 'Save OpMode', 'Export to Java', 'Download OpMode', and 'Download Image of Blocks'. The 'OpMode Name' is set to 'BasicDrive', the 'TeleOp' checkbox is checked, and the 'Group' field is empty. The 'Enabled' checkbox is also checked.

The left sidebar shows a tree view of the block categories: LinearOpMode, Gamepad, Actuators (DcMotor, Dual, Extended), ServoController, Sensors, Other Devices, Android, Utilities, Logic, Loops, Math, Text, Lists, Variables, Functions, and Miscellaneous.

The main workspace shows a block diagram for the 'BasicDrive' OpMode. The code structure is as follows:

- to runOpMode** block:
  - BasicDrive** block:
    - Put initialization blocks here.** block
    - call** `waitForStart` block
    - if** `opModelsActive` block:
      - do** block:
        - Put run blocks here.** block
        - repeat while** `opModelsActive` block:
          - do** block:
            - Put loop blocks here.** block
            - set** `Power` block
            - leftDrive** `to` `0.2` block
            - rightDrive** `to` `0.2` block
            - call** `Telemetry.update` block

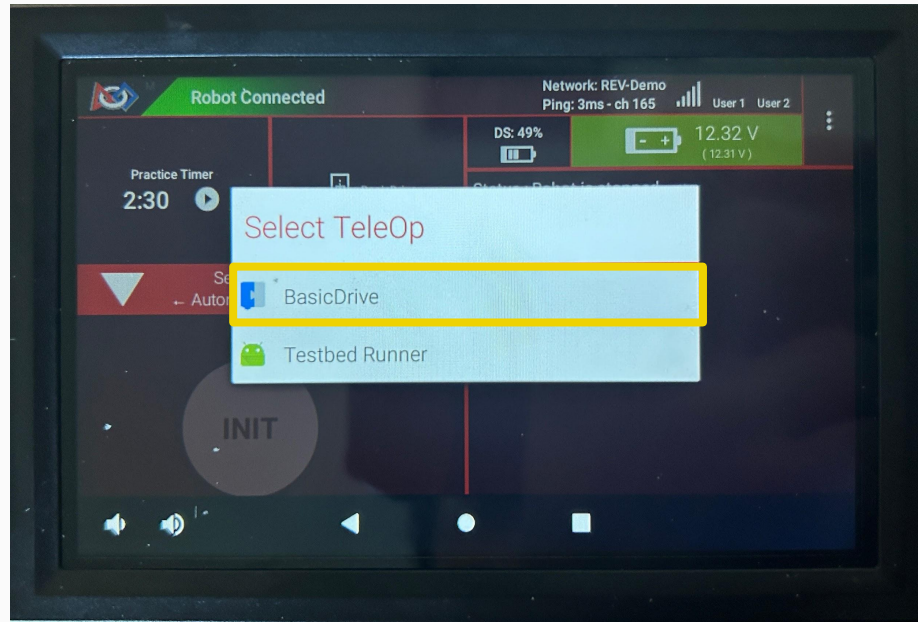
The values `0.2` for `leftDrive` and `rightDrive` are highlighted with a yellow box, indicating the target for the initial test.



Now let's try running your program. To do this, you'll need to save your program. Click "Save OpMode" in the upper left corner of the page.

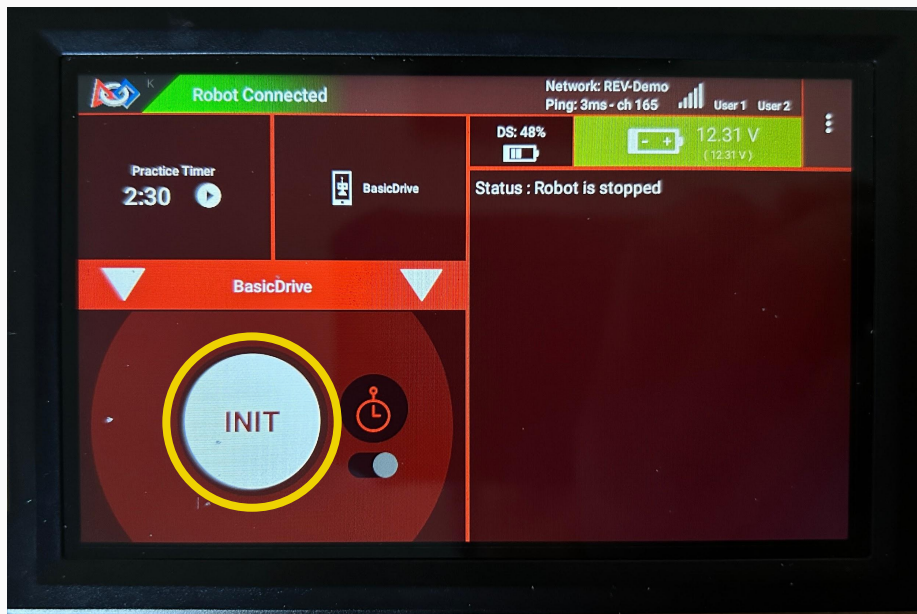
The screenshot displays the FIRST robot controller console interface. At the top, there is a navigation bar with the FIRST logo and tabs for 'Blocks', 'OnBotJava', and 'Manage'. Below this, a row of buttons includes 'Save OpMode' (highlighted with a yellow box), 'Export to Java', 'Download OpMode', and 'Download Image of Blocks'. Underneath the buttons, the 'OpMode Name' is set to 'BasicDrive', the 'TeleOp' dropdown is selected, and the 'Group' field is empty. A checkbox labeled 'Enabled' is checked. On the left side, a sidebar lists various components: LinearOpMode, Gamepad, Actuators (with sub-items DcMotor and ServoController), Sensors, Other Devices, Android, Utilities, Logic, Loops, Math, Text, Lists, Variables, Functions, and Miscellaneous. The main workspace on the right shows a block-based program for 'BasicDrive'. The program starts with a 'to runOpMode' block, followed by a 'Put initialization blocks here.' block containing 'waitForStart'. An 'if' block checks 'opModelsActive', leading to a 'do' block with 'Put run blocks here.' This 'do' block contains a 'repeat while' loop that calls 'opModelsActive' and a 'do' block with 'Put loop blocks here.'. The loop block contains 'set Power' (set to 0.2), 'leftDrive to 0.2', 'rightDrive to 0.2', and 'call Telemetry . update'.

Moving to your Driver Hub, open the FTC Driver Station app. There are two different dropdown arrows on the left side of your screen. Click on the arrow to the right to open the TeleOp program selection menu and select the “BasicDrive” OpMode. If you do this correctly, your FTC Driver Station app should display “BasicDrive” between the two arrows.





Once your “BasicDrive” OpMode has been selected, press the button that reads “INIT” to initialize your program. Upon doing this, the “INIT” button will begin displaying a play symbol. When you’re ready to safely run your robot, press this button again to start the program. Once your program is running, the button will display a stop symbol. When you’re ready to stop running your program, click the button once more.



**If you've done everything correctly, you should notice something interesting: your robot spins! Why does this happen even though we were programming it to drive straight? Aren't both motors set to spin in the same direction?**

**As it turns out, this is due to the orientation and turn direction of the motors. The positive direction of both motors is counterclockwise, so since we gave both motors a positive value, they both spin in that direction. However, because both motors are mounted in opposite directions, one wheel ends up spinning backwards.**



**To fix this, we will need to reverse the direction of the “rightDrive” motor. This can be done on initialization using the set direction block. So essentially, we’d be telling the program to automatically flip the sign of the value given to the right motor as soon as the program begins. This means commanding both motors to spin in the positive direction will tell the robot to drive forward.**

To begin implementing our solution, go to “DC Motor” under “Actuators,” then drag out the block that says “set leftDrive.Direction to Direction.REVERSE.”

The screenshot displays the FIRST robot controller software interface. On the left, a sidebar lists various components: LinearOpMode, Gamemad, Actuators, Dual, Extended, ServoController, Sensors, Other Devices, Android, Utilities, Logic, Loops, Math, Text, Lists, Variables, Functions, and Miscellaneous. The 'Actuators' category is expanded, and the 'DcMotor' block is highlighted with a yellow box, labeled with a '1'. In the main workspace, a block is being dragged from the 'DcMotor' category, labeled with a '2'. This block is 'set leftDrive . Direction to Direction . REVERSE'. The workspace also contains other blocks for 'leftDrive', including 'CurrentPosition', 'Direction', 'Mode', 'Power', 'PowerFloat', and 'TargetPosition'. The top of the interface shows tabs for 'FIRST: robot controller console', 'Blocks', 'OnBotJava', and 'Manage'. Below these are buttons for 'Save OpMode', 'Export to Java', 'Download OpMode', and 'Download Image of Blocks'. The 'OpMode Name' is set to 'BasicDrive', and the 'Group' is set to 'TeleOp'. The 'Enabled' checkbox is checked, and the 'Show Java' button is visible on the right.

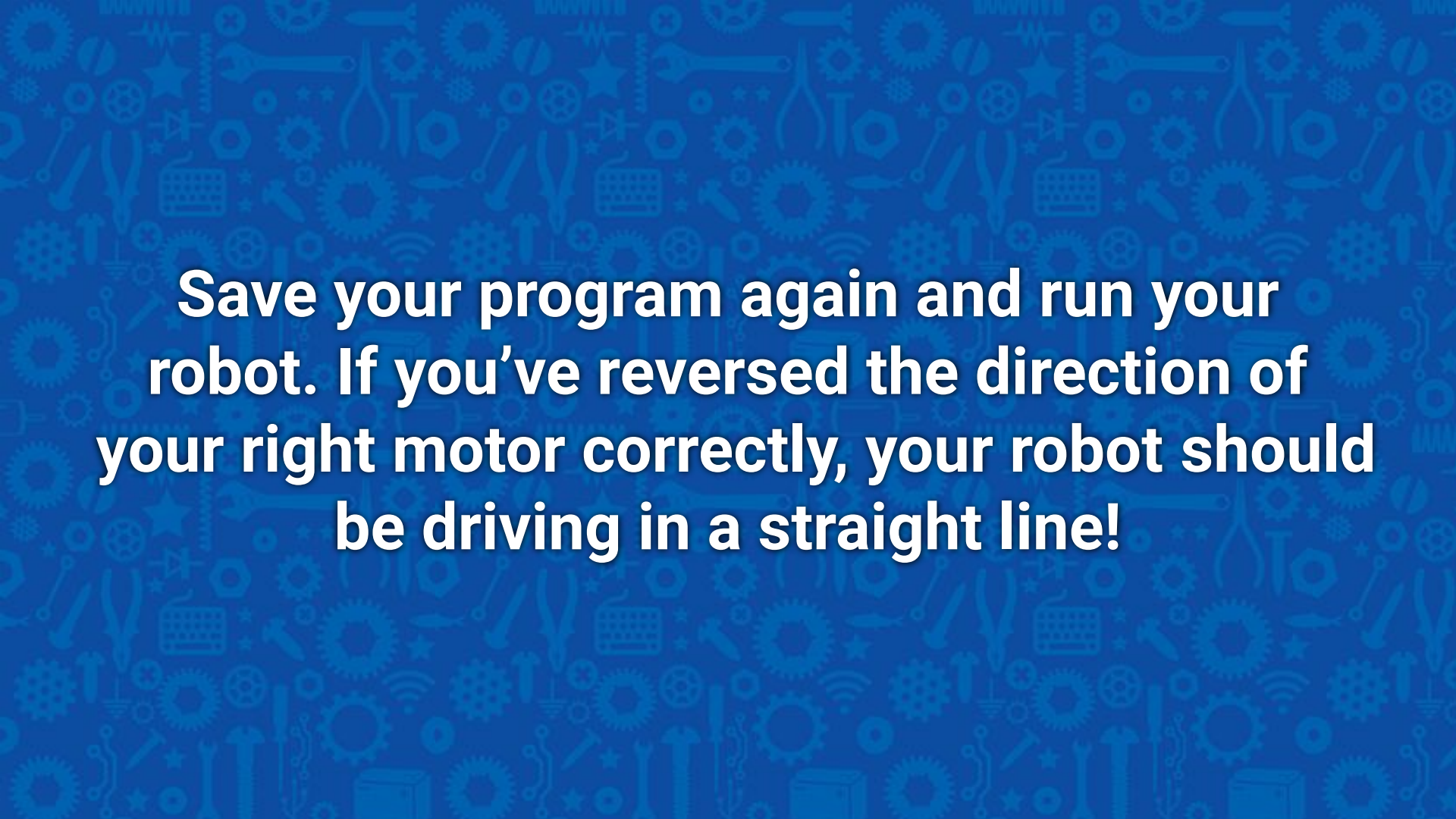
Place the block before the “call waitForStart” block, where all initialization blocks go.

The screenshot shows the FIRST robot controller console interface. The top bar includes the FIRST logo and navigation tabs: Blocks, OnBotJava, and Manage. Below this is a toolbar with buttons for Save OpMode, Export to Java, Download OpMode, and Download Image of Blocks. The main area displays the OpMode Name: BasicDrive, the selected mode: TeleOp, and a status: Enabled. A sidebar on the left lists various components like LinearOpMode, Gamepad, Actuators, Sensors, and Logic. The central workspace shows a block diagram for the BasicDrive OpMode. A yellow box highlights the 'set leftDrive . Direction to Direction REVERSE' block, which is placed before the 'call waitForStart' block. The diagram also includes a 'Put initialization blocks here' block, a 'call opModelsActive' block, and a 'repeat while' loop containing 'Put loop blocks here' and 'call Telemetry . update'.

Change “leftDrive” to “rightDrive”, since that is the motor we want to reverse the direction of.

The screenshot shows the FIRST robot controller console interface. The top bar includes the FIRST logo and tabs for Blocks, OnBotJava, and Manage. Below the top bar are buttons for Save OpMode, Export to Java, Download OpMode, and Download Image of Blocks. The OpMode Name is set to BasicDrive, and the mode is enabled. The left sidebar shows a tree view of the code structure, including LinearOpMode, Gamepad, Actuators (Dual, Extended), ServoController, Sensors, Other Devices, Android, Utilities, Logic, Loops, Math, Text, Lists, Variables, Functions, and Miscellaneous. The main area displays a Blockly script for BasicDrive. The script starts with a 'to runOpMode' block, followed by a 'BasicDrive' block. Inside the BasicDrive block, there is a 'Put initialization blocks here.' block, a 'set rightDrive . Direction to Direction REVERSE' block (highlighted with a yellow box), and a 'call waitOrStart' block. Below these is an 'if call opModelsActive' block with a 'do' loop containing a 'Put run blocks here.' block. Inside the 'do' loop is a 'repeat while call opModelsActive' block, which contains a 'do' loop with 'Put loop blocks here.' containing 'set Power' blocks for 'leftDrive' and 'rightDrive' both set to 0.2, followed by a 'call Telemetry . update' block.

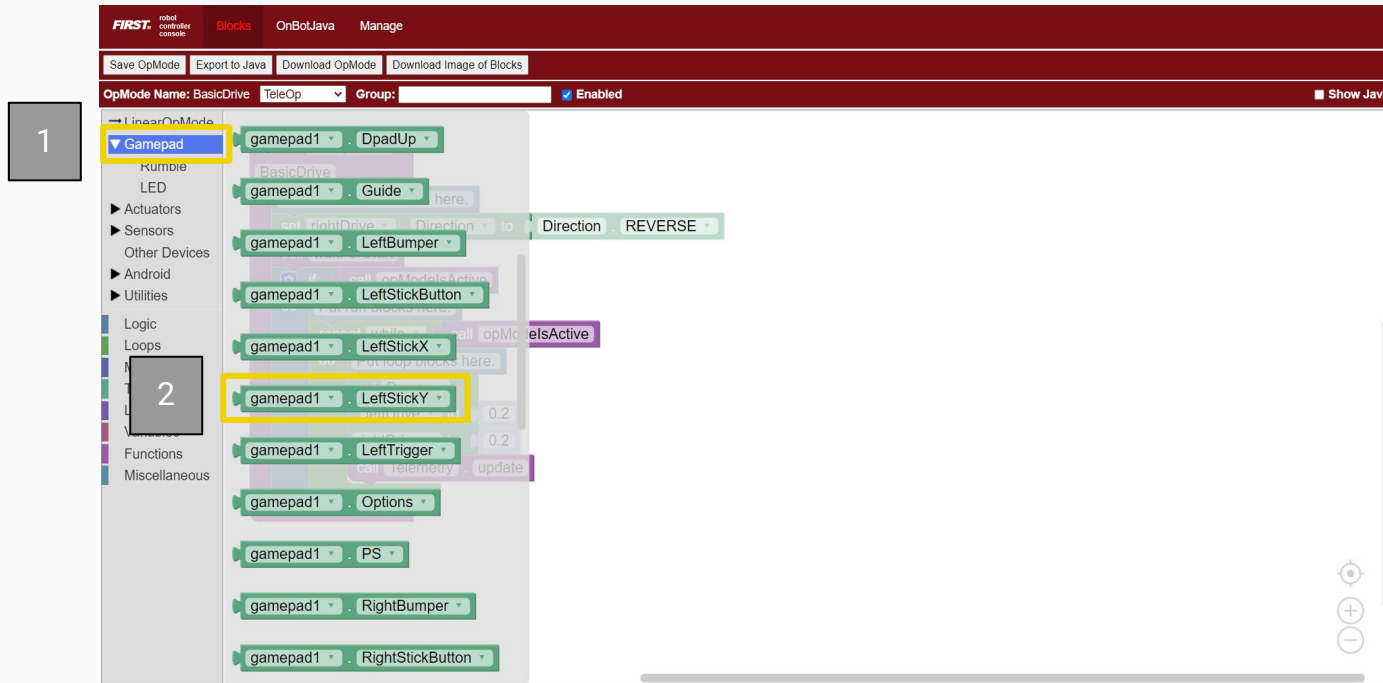




**Save your program again and run your robot. If you've reversed the direction of your right motor correctly, your robot should be driving in a straight line!**

**Now let's make your robot able to be driven with a controller. The control scheme we'll use is called "tank drive," which means the left joystick on your controller will control the robot's left motor and the right joystick will control the right motor. For example, to drive your robot straight, you'll point both joysticks in the same direction, and to turn in place, you'll point both joysticks in opposite directions. You will only need to use the y-axes on both joysticks to control your robot.**

To begin implementing tank drive and controller functionality, locate the “gamepad1.LeftStickY” block in the “Gamepad” folder.



Insert the “gamepad1.LeftStickY” block in place of the constant motor power value block that we were originally using to set the power of your left motor. We can make this change because the joystick axes output values between -1.0 and 1.0, exactly like the values we can use to set motor power. Do the same for the right motor, too.

2

**FIRST** robot controller console Blocks OnBotJava Manage

Save OpMode Export to Java Download OpMode Download Image of Blocks

OpMode Name: BasicDrive TeleOp Group:  Enabled Show Java

- LinearOpMode
  - Gamepad
  - Actuators
  - Sensors
  - Other Devices
  - Android
  - Utilities
- Logic
- Loops
- Math
- Text
- Lists
- Variables
- Functions
- Miscellaneous

to runOpMode

BasicDrive

Put initialization blocks here.

set rightDrive . Direction to Direction REVERSE

call waitForStart

if call opModelsActive

do Put run blocks here.

repeat while call opModelsActive

do Put loop blocks here.

set Power


leftDrive to gamepad1 . LeftStickY

rightDrive to gamepad1 . RightStickY

call Telemetry . update

1





**Save your program again so we can test if  
your robot responds to controller inputs.  
The following slide will show you how to  
connect a controller to your driver hub.**

To connect a controller to your Driver Hub, simply plug your controller into one of the USB 2.0 ports on the Driver Hub and press the “Start” and “A” buttons at the same time to set the controller as User 1. You should now see a gamepad icon in the upper right corner of the Driver Hub’s screen above “User 1”.



**Once you've tested your program, there are two things you may notice:**

- 1. The robot drives very fast. This makes it hard to control.**
- 2. The robot drives backwards. This can be confusing.**

**To fix these issues, we will simply multiply the joystick values by a constant. The constant must be a negative number to make your robot drive forward instead of backward, and it must be a value closer to zero than -1, such as -0.5, so that the motors aren't constantly being told to run at full power.**



To multiply your joystick values by a constant, first find the multiplication block in the “Math” folder and drag it out.

The screenshot displays the FIRST robot controller console interface. The top bar includes the FIRST logo and tabs for Blocks, OnBotJava, and Manage. Below this, there are buttons for Save OpMode, Export to Java, Download OpMode, and Download Image of Blocks. The OpMode Name is set to BasicDrive, and the Group is empty. The console is currently in the Blocks view, showing a list of categories on the left: LinearOpMode, Gamepad, Actuators, Sensors, Other Devices, Android, Utilities, Logic, Loops, Math, Text, Lists, Variables, Functions, and Miscellaneous. The Math category is highlighted with a yellow box and labeled with a '1'. In the main workspace, a multiplication block (1 x 1) is highlighted with a yellow box and labeled with a '2'. The workspace also contains other blocks such as 'to runOpMode', 'Direction to Direction REVERSE', 'Power', 'LeftStickY', 'RightStickY', 'Telemetry update', 'absolute', 'square root', and 'sin'.



Insert the multiplication block in place of the “gamepad1.LeftStickY” block, and move the “gamepad1.LeftStickY” block into the first slot of the multiplication block. Place a constant value block into the second slot of the multiplication block and make its value -0.5. This will multiply the y value of the left stick by -0.5 before applying it to the left motor.

The screenshot shows the FIRST robot controller console interface. The top bar includes tabs for "FIRST robot controller console", "Blocks", "OnBotJava", and "Manage". Below the tabs are buttons for "Save OpMode", "Export to Java", "Download OpMode", and "Download Image of Blocks". The "OpMode Name" is set to "BasicDrive", the "TeleOp" mode is selected, and the "Group" is empty. The "Enabled" checkbox is checked, and a "Show Java" button is visible.

The left sidebar shows a tree view of the code structure, including "LinearOpMode", "Gamepad", "Actuators", "Sensors", "Other Devices", "Android", "Utilities", "Logic", "Loops", "Math", "Text", "Lists", "Variables", "Functions", and "Miscellaneous".

The main workspace displays a Blockly script for the "BasicDrive" OpMode. The script is as follows:

```
graph TD
    Start([to runOpMode]) --> BasicDrive[BasicDrive]
    BasicDrive --> Init[Put initialization blocks here.]
    Init --> SetDir[set rightDrive . Direction to Direction . REVERSE]
    SetDir --> WaitForStart[call waitForStart]
    WaitForStart --> IfActive[if call opModelsActive]
    IfActive --> DoRun[do Put run blocks here.]
    DoRun --> RepeatWhile[repeat while call opModelsActive]
    RepeatWhile --> DoLoop[do Put loop blocks here.]
    DoLoop --> SetPower[set Power to]
    SetPower --> LeftDrive[set leftDrive to]
    LeftDrive --> Mult[gamepad1 . LeftStickY x -0.5]
    Mult --> RightDrive[set rightDrive to gamepad1 . RightStickY]
    RightDrive --> Telemetry[call Telemetry . update]
    Telemetry --> RepeatWhile
```

The multiplication block (gamepad1 . LeftStickY x -0.5) is highlighted with a yellow box. The constant value block is set to -0.5.

Follow the same process to modify the values of your right joystick, as shown.

The screenshot displays the FIRST robot controller console interface. The top navigation bar includes 'FIRST robot controller console', 'Blocks', 'OnBotJava', and 'Manage'. Below this, there are buttons for 'Save OpMode', 'Export to Java', 'Download OpMode', and 'Download Image of Blocks'. The 'OpMode Name' is set to 'BasicDrive', and the 'Group' is empty. The 'Enabled' checkbox is checked. A 'Show Java' button is visible on the right.

The left sidebar shows a tree view of the code structure, including 'LinearOpMode', 'Gamepad', 'Actuators', 'Sensors', 'Other Devices', 'Android', 'Utilities', 'Logic', 'Loops', 'Math', 'Text', 'Lists', 'Variables', 'Functions', and 'Miscellaneous'.

The main workspace shows the 'BasicDrive' OpMode code. The code is written in a block-based format. The 'runOpMode' block contains the following logic:

- Put initialization blocks here.
- set rightDrive . Direction to Direction . REVERSE
- call waitForStart
- if call opModelsActive
- do
- Put run blocks here.
- repeat while call opModelsActive
- do
- Put loop blocks here.
- set Power
- leftDrive to gamepad1 . LeftStickY x -0.5
- rightDrive to gamepad1 . RightStickY x -0.5
- call Telemetry . update

The blocks for 'leftDrive to gamepad1 . RightStickY x -0.5' and 'rightDrive to gamepad1 . RightStickY x -0.5' are highlighted with a yellow box, indicating the process of modifying the right joystick values.

**To explain what we just did, multiplying your joystick values by a negative number inverts them so when they're converted to motor power values, they make your robot's motors spin forward instead of backward. Additionally, multiplying your joystick values by -0.5, means the maximum power of your motors will be 0.5, or 50%. This solves both problems with driving the robot. Save your program and run one final test.**

# That's It!

**If you did everything correctly, your robot should respond to your controller inputs, and everything should be working as intended. The program you just wrote is fairly basic, but it should serve as a good introduction to programming a robot. We'll use it in the next few lessons and you'll learn more about programming in Lesson 13.**