

**2015-2016 *FIRST*® Tech Challenge**  
**PushBot Build Guide**  
**TETRIX Sensors Supplement**



## Volunteer Thank You

Thank you for taking the time to volunteer for a *FIRST* Tech Challenge Event. *FIRST* and FTC rely heavily on Volunteers to ensure Events run smoothly and are a fun experience for Teams and their families, which could not happen without people like you. With over 4,500 Teams competing annually, your dedication and commitment are paramount to the success of each Event and the FTC program. Thank you for your time and effort in supporting the mission of *FIRST*!



## Sponsor Thank You

Thank you to our generous sponsors for your continued support of the *FIRST* Tech Challenge!

*FIRST*® Tech Challenge Official Program Sponsor

**Rockwell  
Collins**

*FIRST*® Tech Challenge  
Official IoT, CAD and Collaboration  
Software Sponsor

**PTC®**

*FIRST*® Tech Challenge  
Official Control System  
Sponsor

**QUALCOMM®**

Revision History		
Revision	Date	Description
1	9/12/2014	Initial Release - Content by former FTC Team #2843, Under the Son

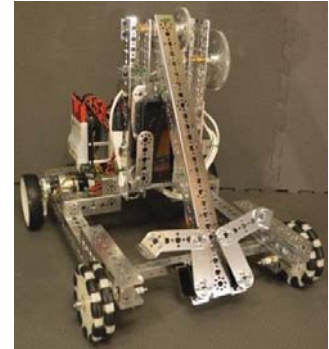
## Contents

Overview .....	4
Adding the Modules and Sensors to the Robot.....	5
Adding the Core Device Interface Module (DIM) .....	5
Adding the Touch Sensor.....	5
Adding the IR Seeker Sensor (V3) .....	14
Optical Distance Sensor (ODS).....	16
Securing and protecting wires .....	21
Installing the PushBot Sensor op-modes .....	23
Determining whether the PushBot Sensor Op-modes are available .....	23
Configuring the Robot Controller .....	25
Integrating the op-modes and electronics .....	46
Testing the Robot .....	47
Testing the touch sensor.....	47
Testing the IR sensor .....	47
Testing the ODS sensor .....	47
Testing the manual operation of the arm.....	47
Testing a multi-sensor autonomous op-mode .....	48
Appendix A: Bill of Materials List for Additions to the PushBot.....	49
Appendix B: Project Team Profiles .....	51

## Overview

The PushBot is a robot that can be built from only the parts in the competition kit: electronics, metal components and technology components (i.e. phones and gamepads). The original PushBot Guide provided step-by-step instructions for building a four-wheeled, one-armed autonomous and manual robot. This PushBot Sensor Guide is an addendum to that document that provides step-by-step instructions for adding a touch sensor, Infra-Red (IR) seeker (V3) sensor and optical distance sensor (ODS) to a PushBot built from the original guide.

In autonomous mode, sensors can be used for detection of an object, guide the robot to a location or move parts to a predefined position. While sensors are used more often in autonomous mode, they can be used in manual mode also. Suppose a game element has an embedded magnet. An on-board magnetic sensor could be used to detect it. Suppose a delivery device needs to be at a predefined height, a touch sensor could be used to move the arm to that position when a user presses a button.



The sensors shown in this guide all connect to the robot's Core Power Distribution Module (PDM) using the Core Device Interface Module (DIM). The DIM connects to the PDM using a provided USB cable. Sensors connect to the DIM using cables attached to the sensors. None of these components require 12 VDC power, so there are no cables for that. The DIM physically connects to the robot using existing parts: the bolts and nuts that are currently holding the power module onto the robot.

The touch sensor, IR seeker and ODS have mounting holes, but those holes don't align to the TETRIX hole pattern. This guide recommends attaching these sensors to the robot using sticky-back Velcro, but Velcro isn't part of the kit. Velcro is widely available and holds the sensors more firmly to the robot. This guide uses zip ties, which are part of the kit, but may have been used to secure wires to the robot from the recommendations in the original guide. Another issue with zip ties is that the sensor is not held as firmly to the robot and may move during competitive play.

The touch sensor and IR sensor require metal parts for it to work as shown in this guide. The parts left from building the original PushBot can be used. These parts attach to the PushBot without any need to modify other parts of the robot.

The IR sensor needs an IR source for it to work as shown in this guide. An IR beacon is not provided in the kit.

The ODS can be tested using a foam tile and white tape. These components are not part of the kit.

The Robot Controller application communicates with the electronics through names assigned to the devices. Part of the process for naming the devices is shown in this document, but this document refers to the original Guide, so it will be needed at times.

The sensor code has been placed into new classes, which derive from the old classes. Class derivation is a large part of object oriented theory and allows code to be added without modifying any of the original PushBot classes. To use the original PushBot code without sensors, use the files listed on the left; the files with added sensor code are listed in the center column and example/test op-modes are listed on the right. The code is part of the FTC SDK and is located at at [https://github.com/ftctechnh/ftc\\_app](https://github.com/ftctechnh/ftc_app).

PushBotAuto.java	PushBotAutoSensors.java	PushBotIrEvent.java
PushBotHardware.java	PushBotHardwareSensors.java	PushBotTouchEvent.java
PushBotManual.java	PushBotManualSensors.java	PushBotOdsDetectEvent.java
PushBotTelemetry.java	PushBotTelemetrySensors.java	PushBotOdsFollowEvent.java

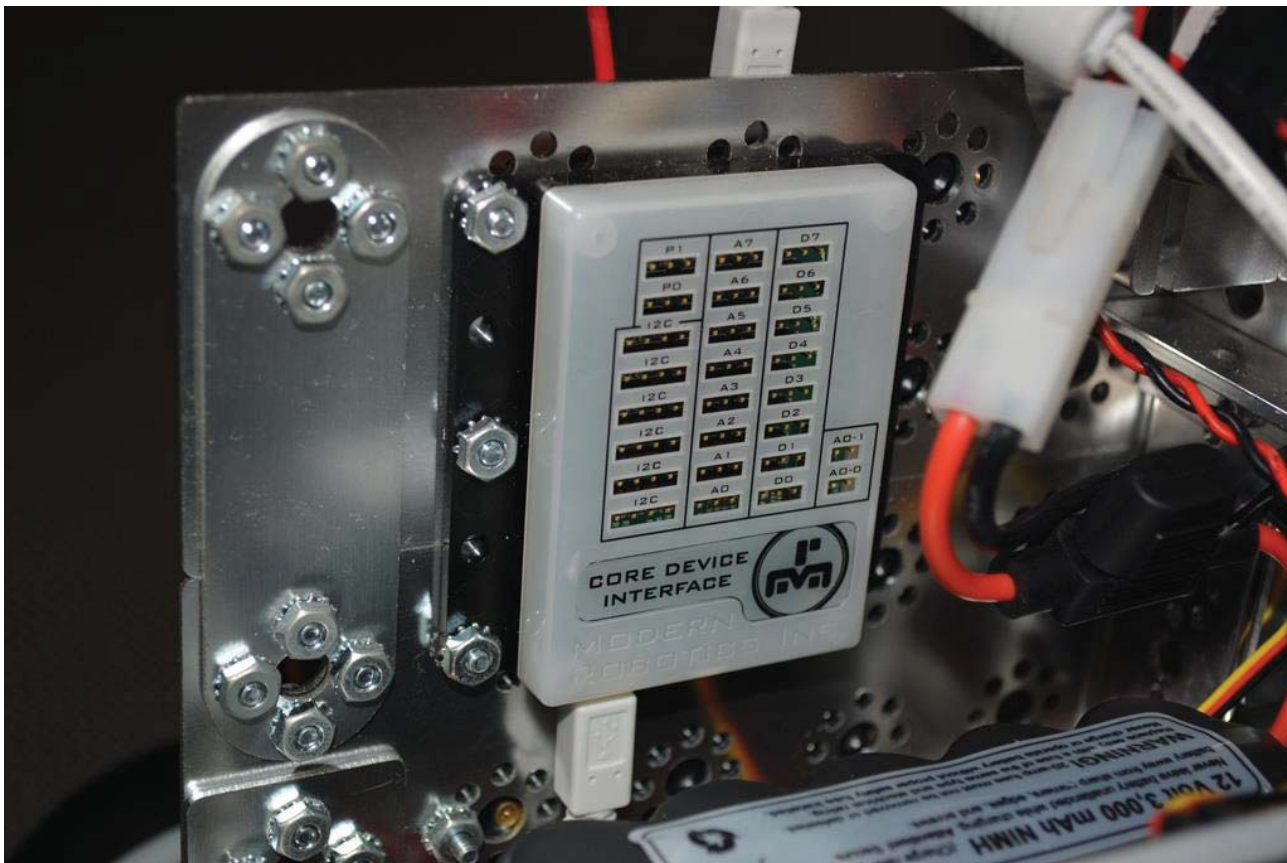


## Adding the Modules and Sensors to the Robot

---

### ***Adding the Core Device Interface Module (DIM)***

This device has digital, analog input, analog output, PWM output and I2C ports. These ports provide for a wide range of sensors. It will physically connect to the robot using three of the screws holding the power module onto the robot. The module will electronically connect to the Core Power Distribution Module using a provided USB cable. It will be logically connected using the Robot Controller application's Configuration activity. The following photo shows the module mounted to the robot. It is mounted in the battery area on the opposite side of the plate that is holding the Core Power Distribution Module. The USB cable has already been plugged into the Core Device Interface Module and the Core Power Distribution Module (not shown).



### ***Adding the Touch Sensor***

The touch sensor can be used for collision detection, weight detection or movement detection. For example, the sensor could be mounted to the front of the robot to detect when the robot has driven into a wall. Or it can be used under a bucket to determine when a certain number of game elements have been added to it. This guide will use it to determine when the arm has reached a predefined position. Suppose the arm must be raised to a certain height to score a ring onto a PVC tube (yep, you guessed it...the Ring It Up! game).

The sensor is mounted to the robot using zip ties to the vertical beam supporting the robot's arm. Using leftover parts from the original guide, a trigger will be built and attached to the arm. When the arm is at a certain position, the trigger will depress the switch on the touch sensor and the sensor will glow. Leftover zip ties can be used to connect the sensor to the robot, but sticky-back Velcro is recommended. Velcro will hold the sensor more firmly to the robot than zip ties, but is not provided in the kit. Another way to hold the sensor on the robot would be to manufacture a mounting device using a 3-D printer. If another attachment method is used, then the trigger may need to be modified to insure that the button on the touch sensor is fully pressed when the arm reaches the desire height.

The following steps will describe how to build the trigger mechanism.

Step 1: 1/2" socket head cap screws (2), keps nut (2), 64mm flat (2), flat spacer (1)





The following photo shows the parts assembled.



Step 2: assembly from step 1, 64mm flat (1), 1/2" socket head cap screws (2).



The following photo shows the parts assembled.





Step 3: The assembly from the prior step, 5/16 socket head cap screws (2), keps nuts (2), flat bracket (1),



The following photo shows the parts assembled.

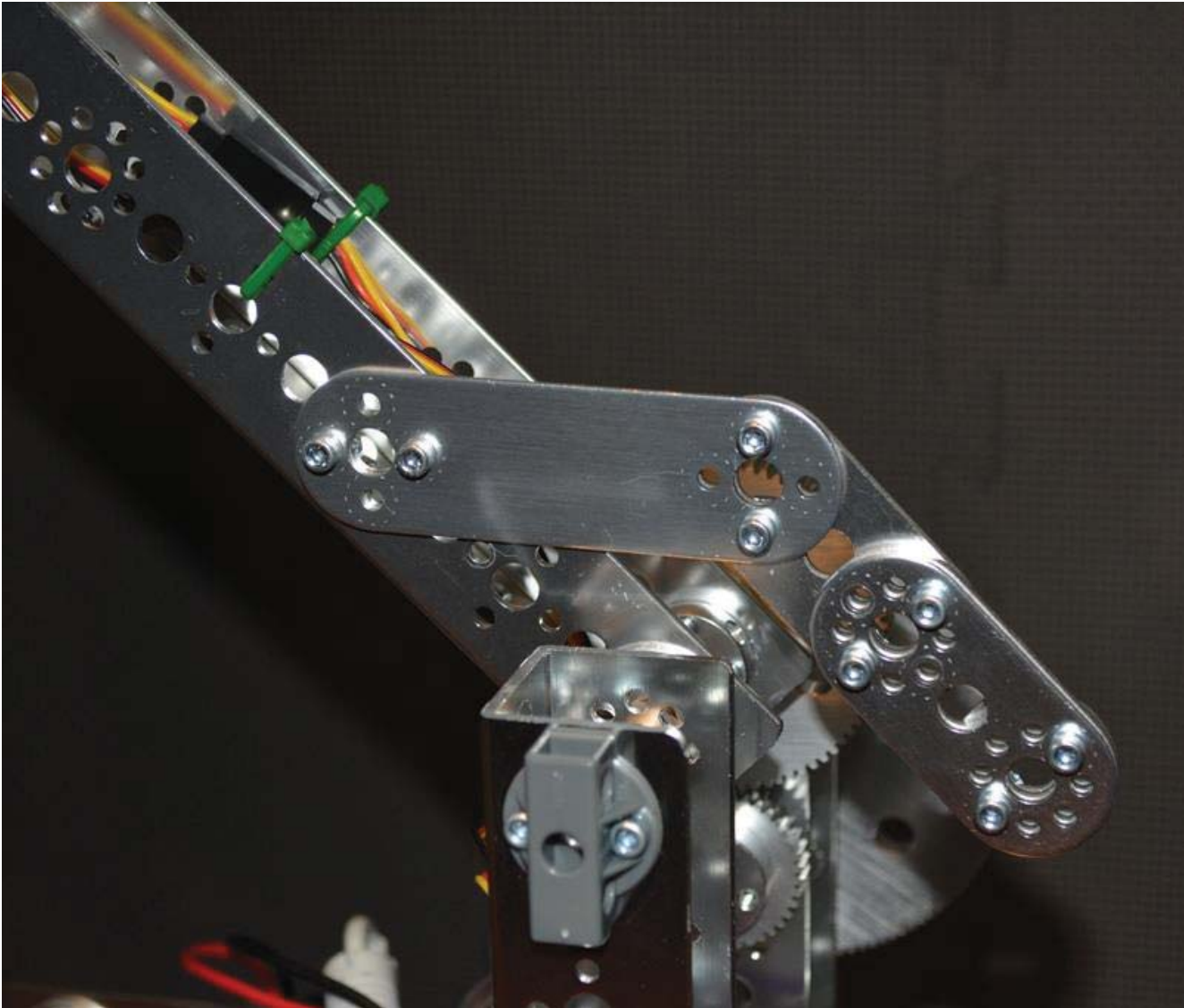


Step 4: The assembly from the prior step, the PushBot, 5/16 socket head cap screws (2), keps nut (2).



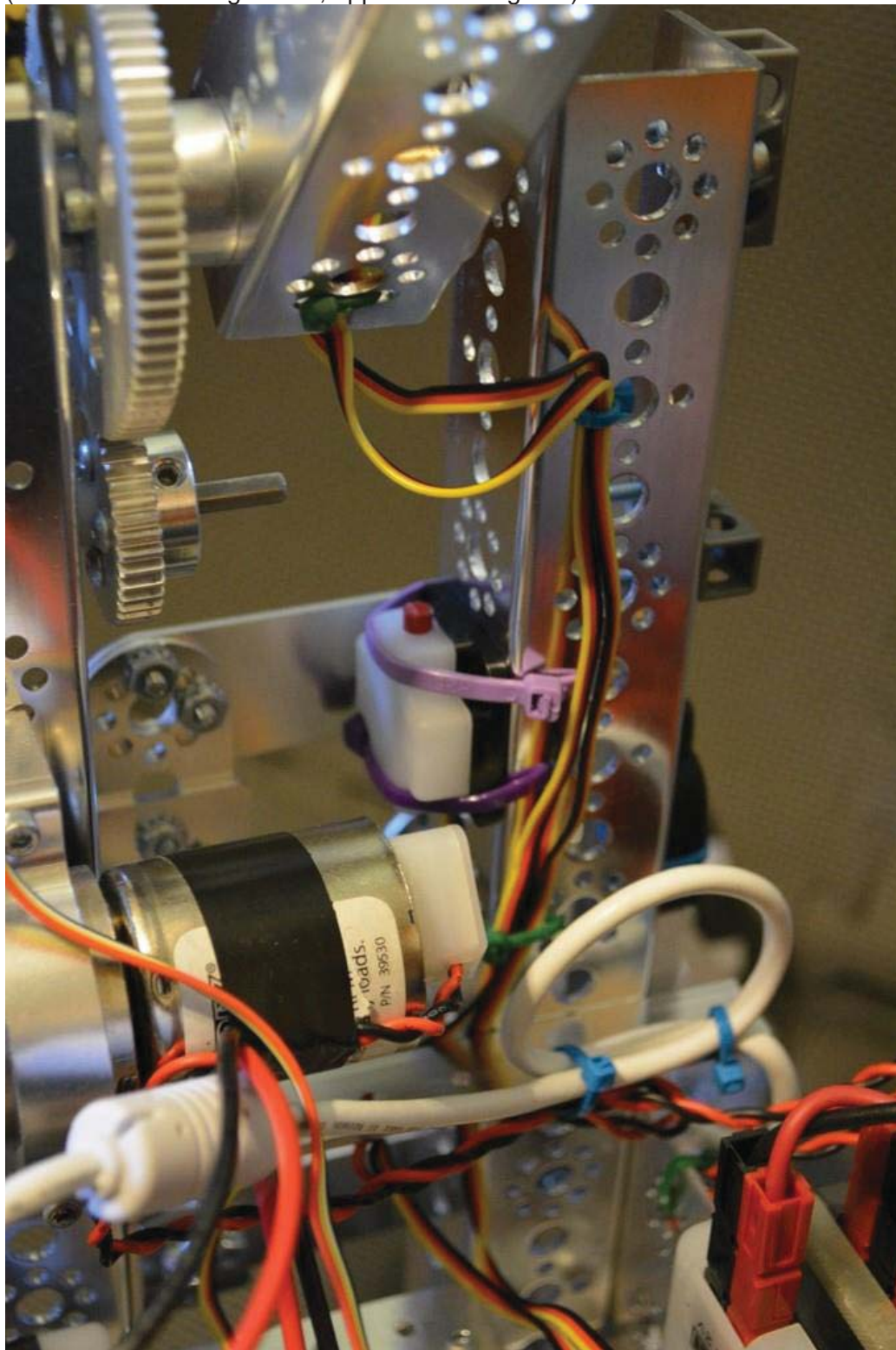


The following photo shows the trigger assembled to the arm. Note, in this photo, the arm has been rotated up past the vertical towards the back of the robot to make the mounting more visible. This photo is of the right side of the robot arm, the side opposite of the gears.

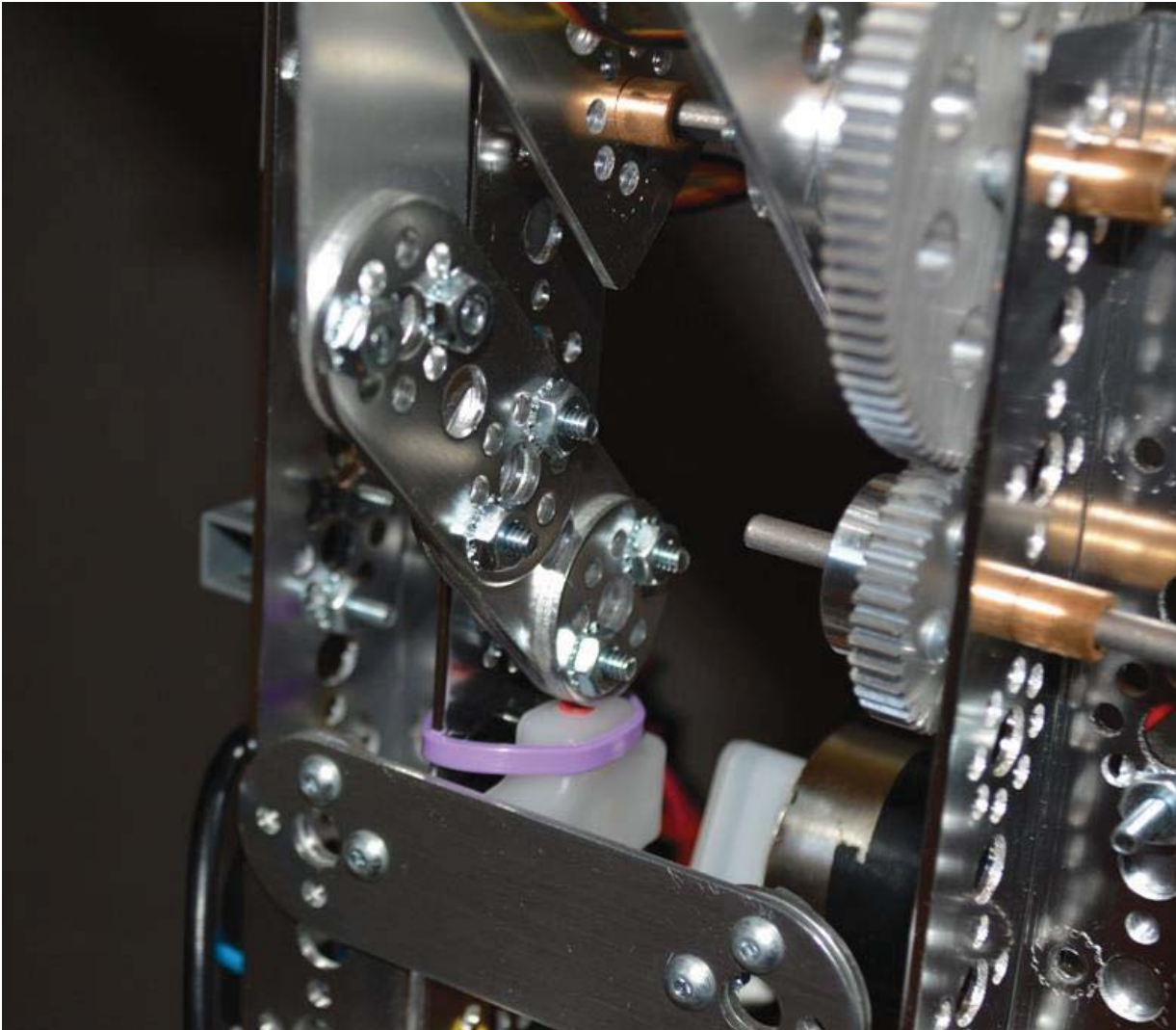




The following photo shows the touch sensor attached to the tower upright. It is mounted to the right upright (the one with the flag holder, opposite of the gears).



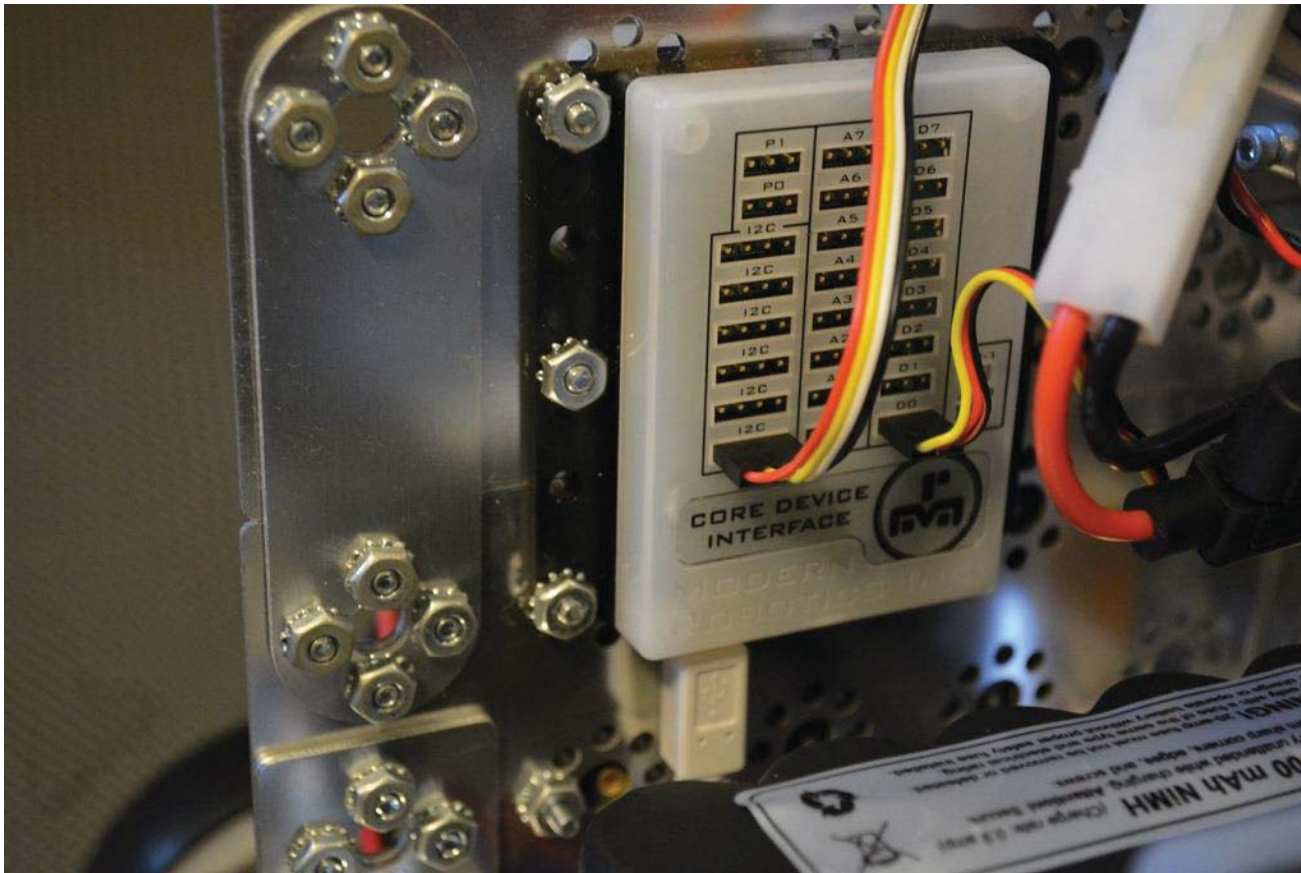
The following photo shows the trigger depressing the button on the touch sensor (photo is from the front of the robot).



The cable attached to the sensor will electronically connect the sensor to the Core Device Interface Module. The cable will be plugged into the D0 connector. This is the first digital I/O port on the Core Device Interface Module. It will be logically connected using the Robot Controller application's Configuration activity.



The following photo shows the sensor plugged into the Core Device Interface Module. Note, in this photo, the IR Seeker Sensor (V3) is already plugged into the first I2C port.



### ***Adding the IR Seeker Sensor (V3)***

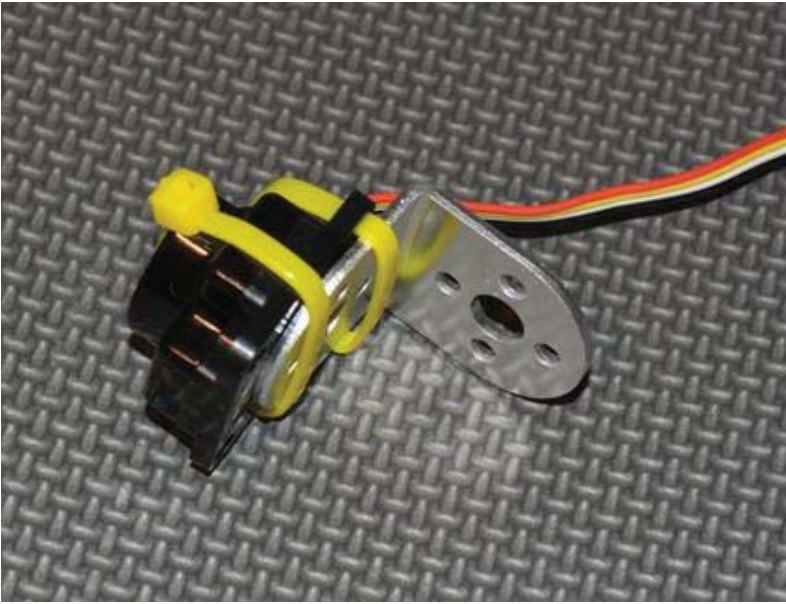
The Infra-Red Seeker Sensor (V3) can be used to detect an IR source. In the Hot Shot! game, an IR beacon was attached to a rotating goal. The robots could drive to the sensor and fire wiffle balls into the goal. The Get Over It! game used a beacon to mark the dispenser location where a doubler baton was placed at the beginning of the autonomous period. This guide will use it to drive the robot toward an IR source. When the sensor is close to the beacon, the robot will stop and open the hand. Note that the beacon's signal strength is used to halt the robot. Signal strength depends on the power level of the battery in the beacon. This strength varies from battery to battery and declines over time. This guide shows the use of the signal strength, but it may cause the robot to stop too soon and open the claw too early. The use of this function should be evaluated before using it in a competitive setting.

The sensor is mounted to the robot using zip ties. It is located at the top of the vertical beam supporting the robot's arm. The following steps show how to mount the sensor to the robot. Zip ties can be used to connect the sensor to the robot, but sticky-back Velcro is recommended. Velcro will hold the sensor more firmly to the robot than zip ties, but is not provided in the kit. Another way to hold the sensor on the robot would be to manufacture a mounting device using a 3-D printer.

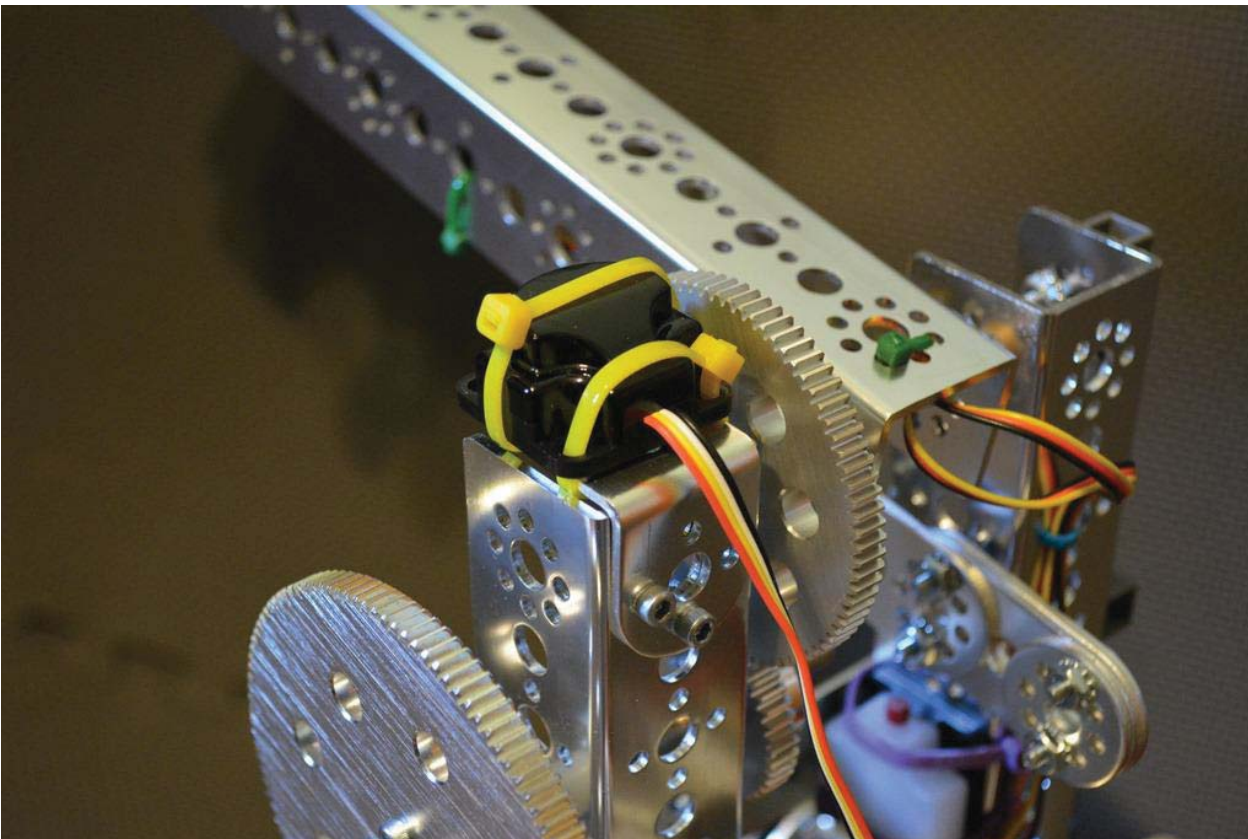


Step 1: L bracket (1), IR Seeker V3 (1), zip ties (2).

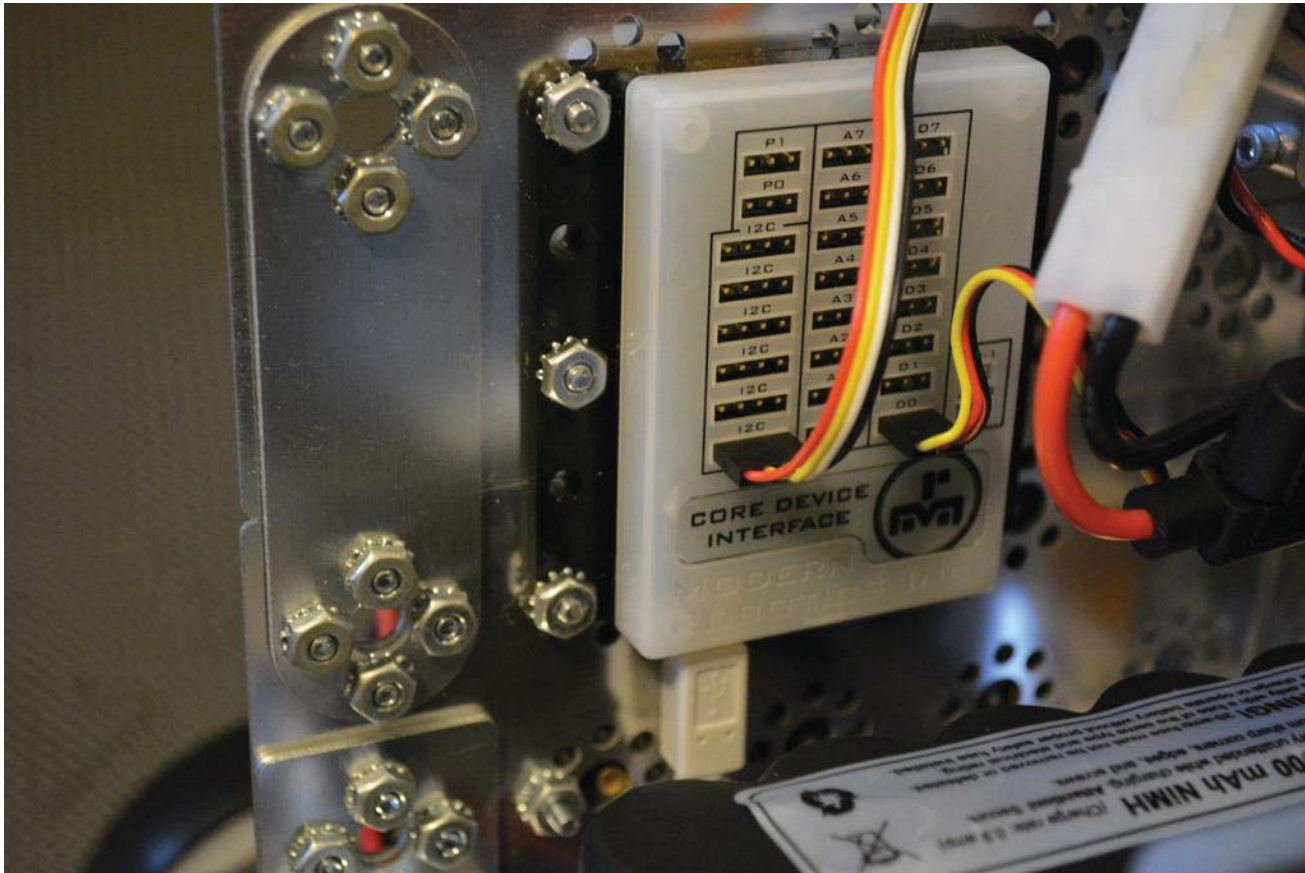
The following photo shows the sensor attached to the L bracket.



Step 2: The assembly from the prior step, 1/2" socket head cap screws (2), keps nuts (2), the Push 'Bot. The bracket is attached to the left upright of the tower. It is attached using 1/2" screws. The following photo shows the sensor attached to the robot (photo is from the back left of the robot).



The cable attached to the sensor will electronically connect the sensor to the Core Device Interface Module. It will be logically connected using the Robot Controller application's Configuration activity. The sensor is plugged into the first I2C connection on the Core Device Interface Module as shown in the following photo. Note, in the photo, the touch sensor is plugged into the first digital I/O port (from the previous section).



### **Optical Distance Sensor (ODS)**

The Optical Distance Sensor can be used to detect a change in the amount of reflected light. That means that it can be used to avoid a collision, measure the distance from the sensor to an object, or measure the distance from the robot's arm from the floor. This guide will use it to detect and follow a line.

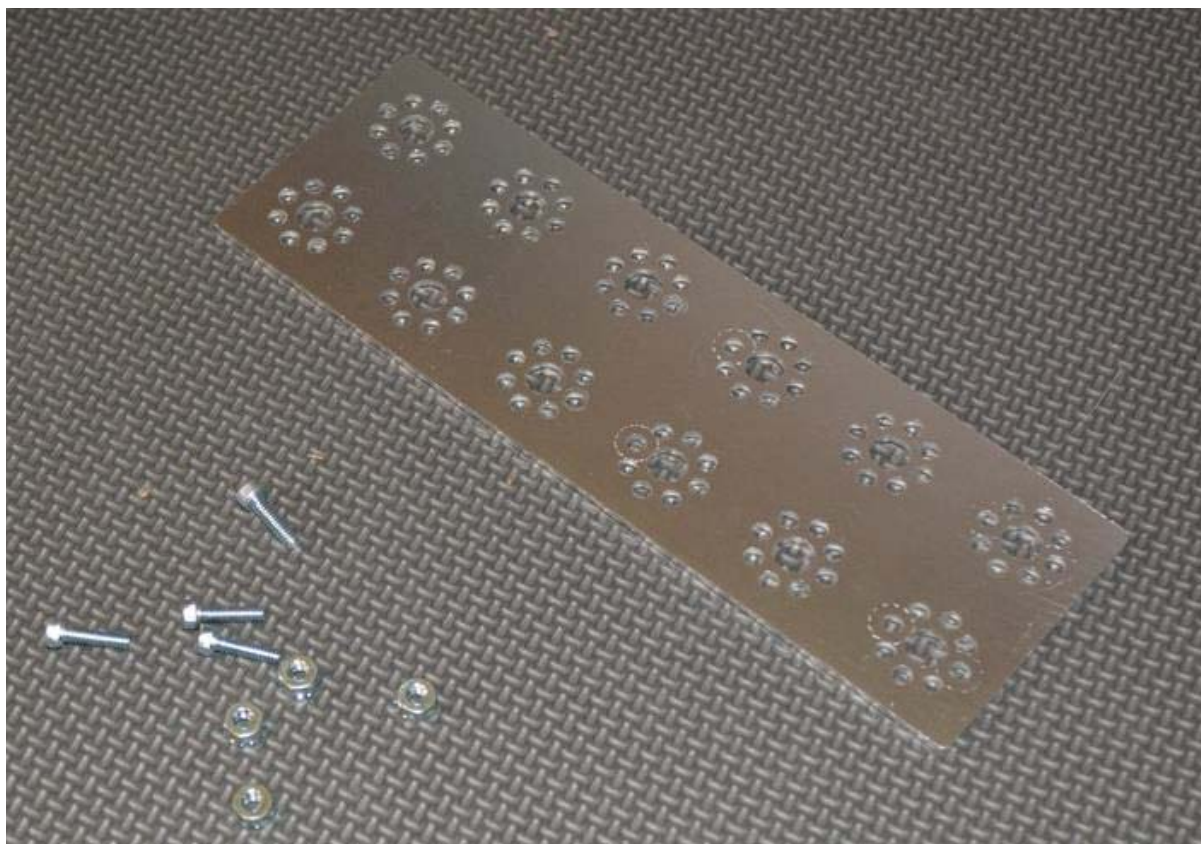
The sensor is mounted to the robot using zip ties. It is located on a plate that fastens to the horizontal beam near the front wheels. Leftover zip ties can be used to connect the sensor to the robot, but sticky-back Velcro is recommended. Velcro will hold the sensor more firmly to the robot than zip ties, but is not provided in the kit. Another way to hold the sensor on the robot would be to manufacture a mounting device using a 3-D printer.

The cable attached to the sensor will electronically connect the sensor to the Core Device Interface Module. It will be logically connected using the Robot Controller application's Configuration activity.



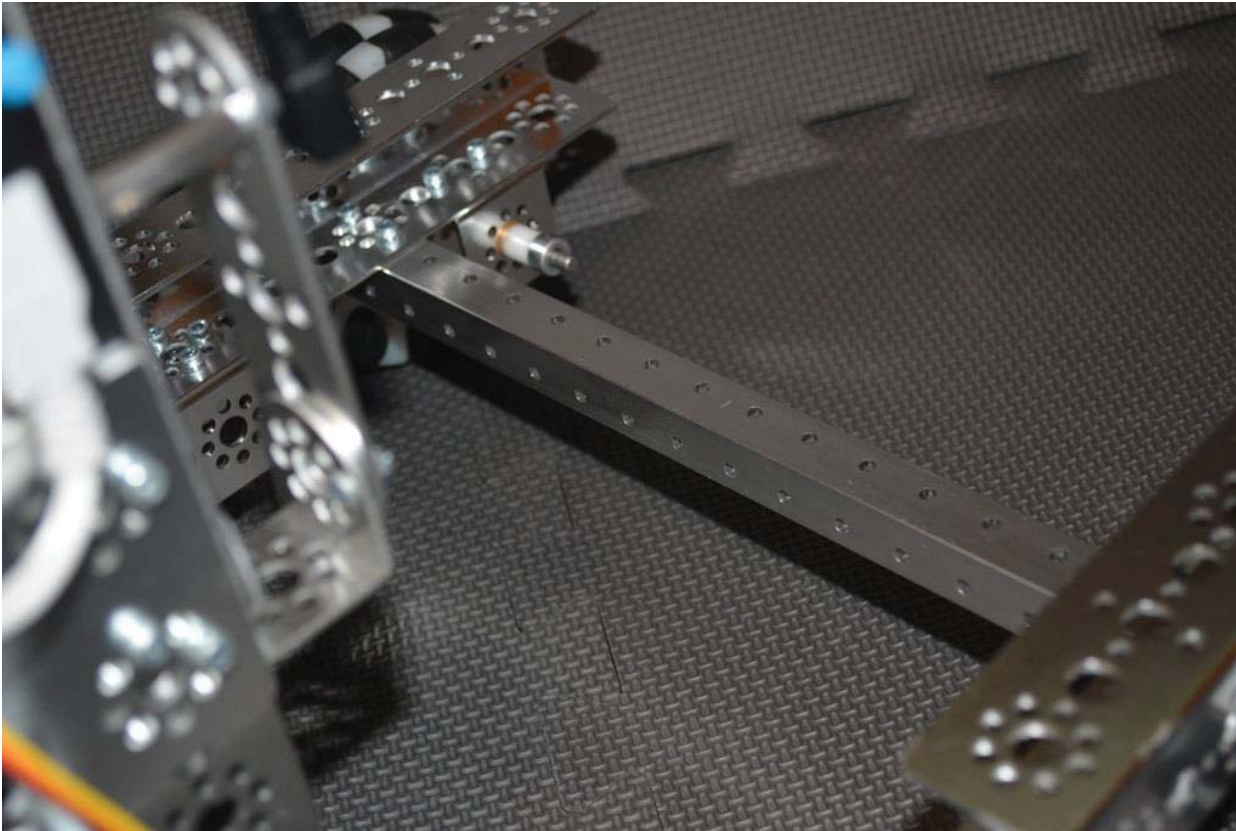
The following steps show how to install the sensor.

Step 1: flat building plate (1), 1/2" socket head cap screws (4), keps nuts (4), pushbot



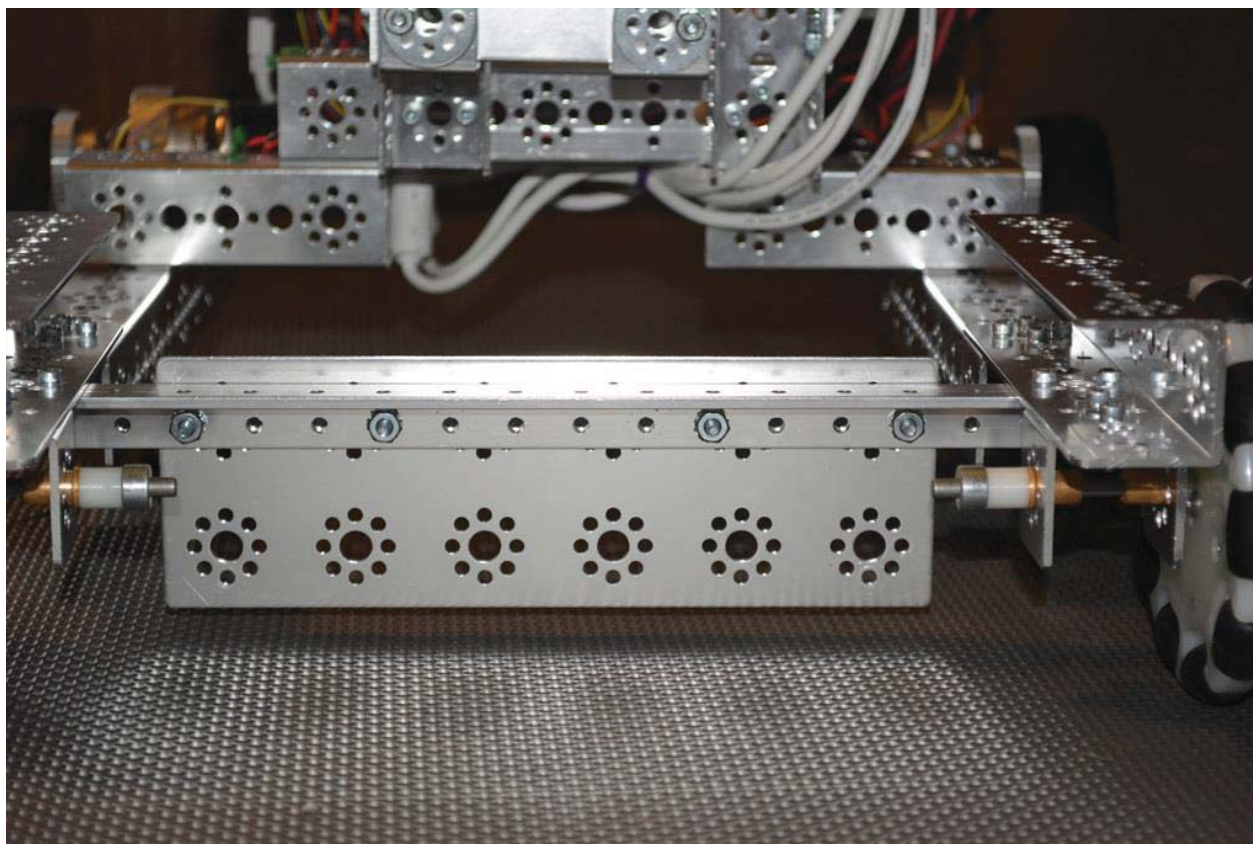


The plate is attached to the angle beam at the front of the robot. Prior to installing, make sure the flat side of the angle beam is towards the back of the robot as shown in the following photo.

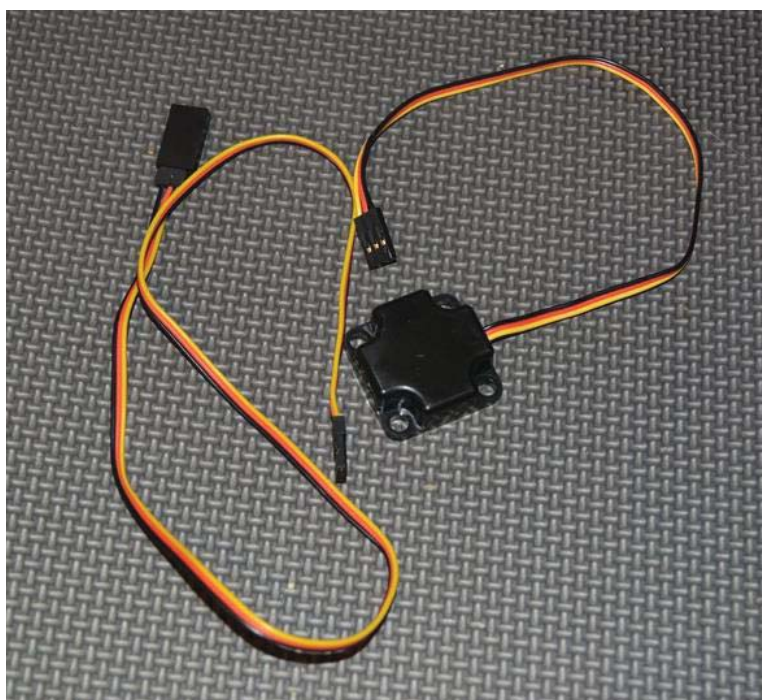


If it is not in this orientation, then remove and re-install it as shown.

The following photo shows the plate attached.



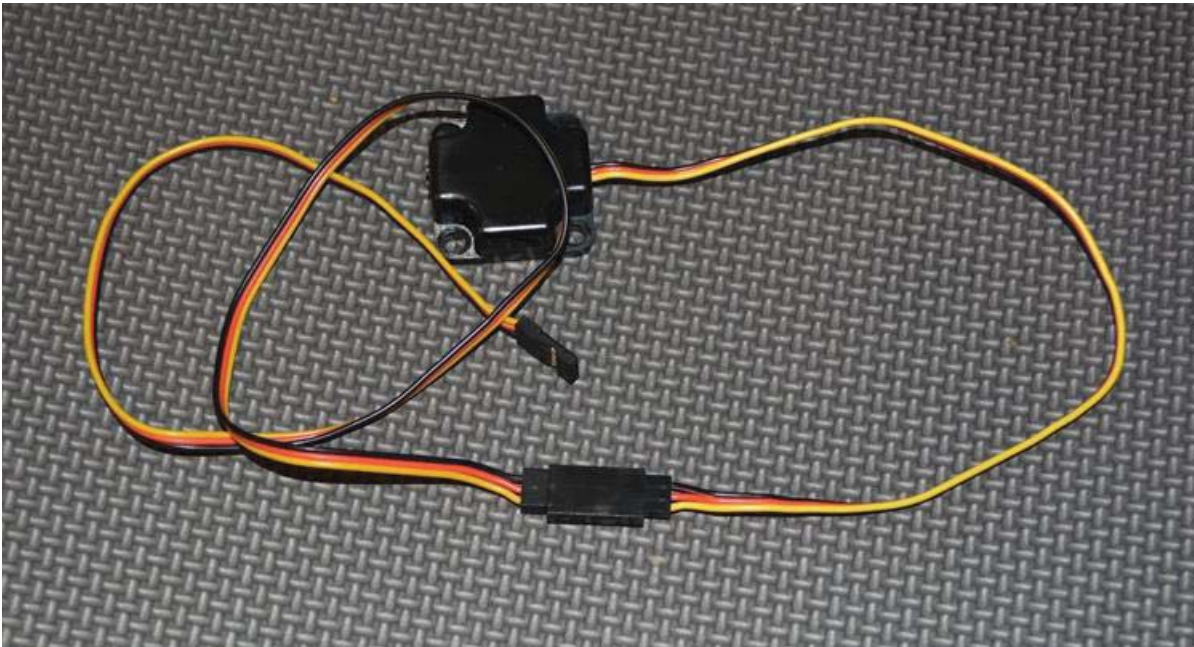
Step 2: sensor (1), servo extension cable (1), zip tie (1) not shown.



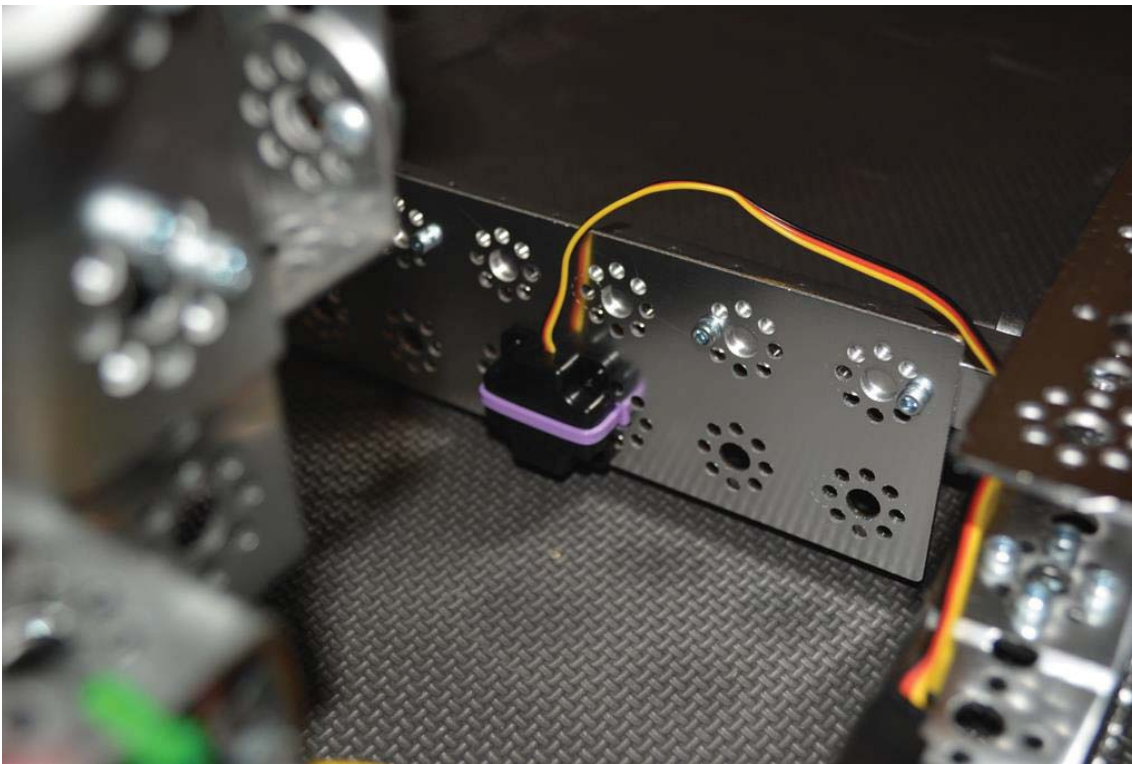
Plug the cable from the sensor into the servo extension cable as shown in the following photo.

**Gracious Professionalism** - *"Doing your best work while treating others with respect and kindness - It's what makes FIRST, first."*



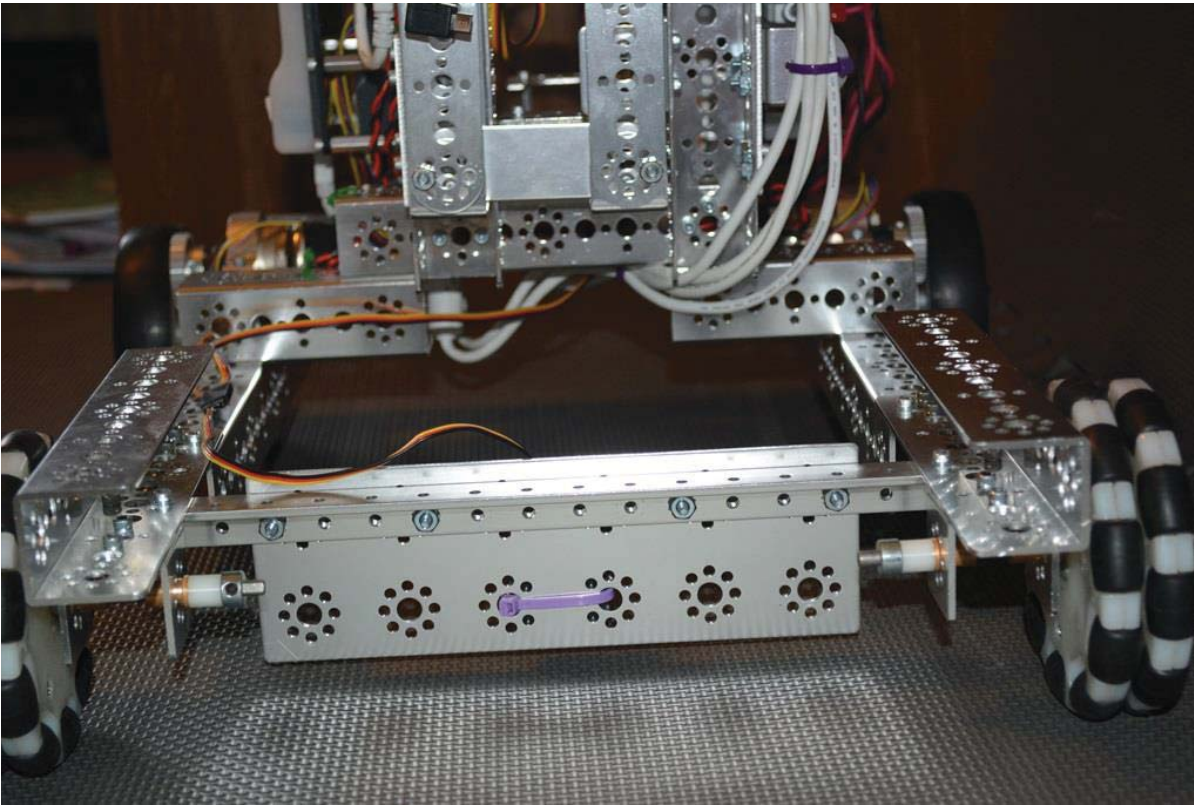


The following photos show the sensor attached to the plate. Note: attach the sensor to the center of the plate so it will be centered on the robot. Also, mount it low to the ground so it will not be as affected by ambient light and will give more accurate readings.

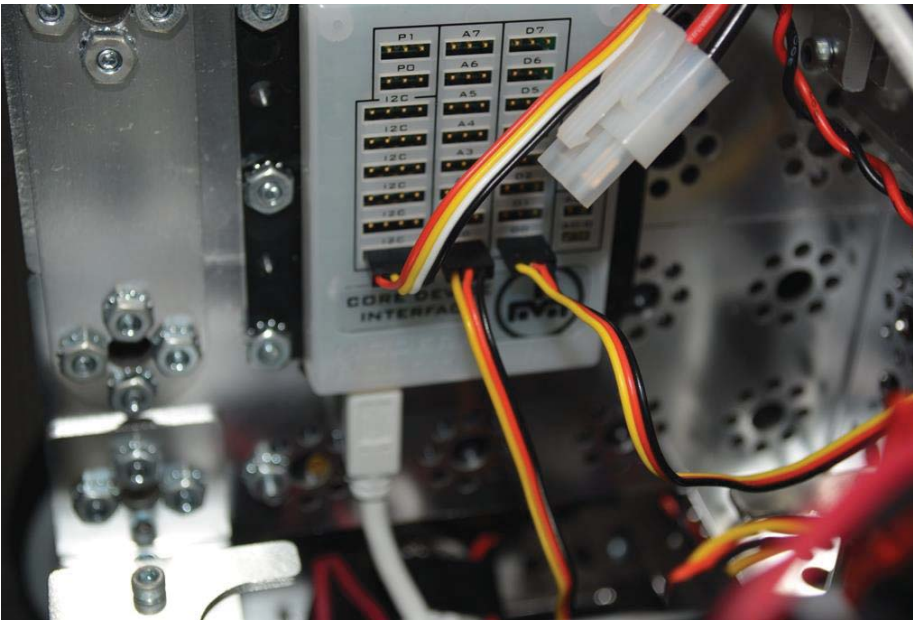


In addition to holding the sensor, the plate give the PushBot a surface which can be used to push various game elements. The elements can include, but are not limited to: hockey pucks, wiffle balls, batons, rolling goals, bowling balls, rings, blocks and the occasional red herring.





The servo extension cable is then plugged into the first analog port (A0) on the Core Device Interface Module as shown in the following photo.

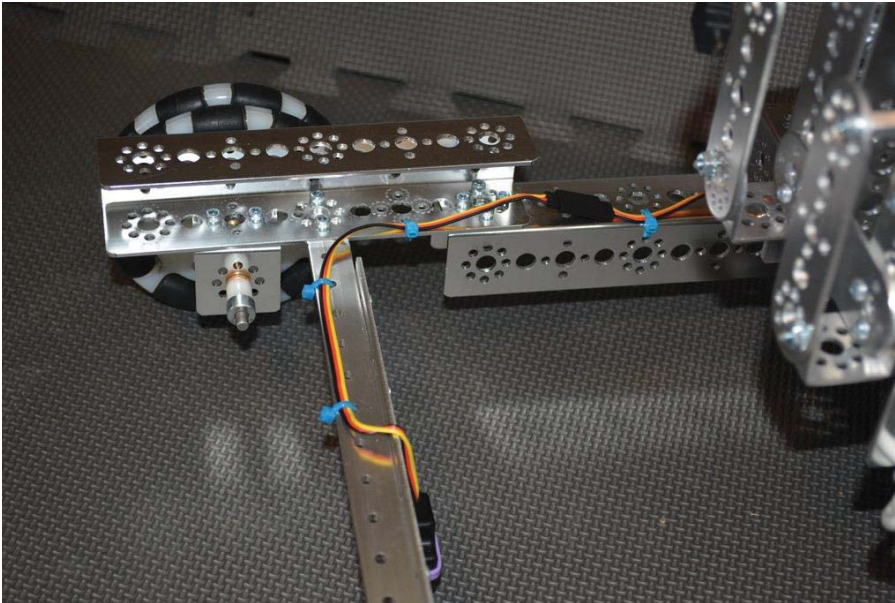


### ***Securing and protecting wires***

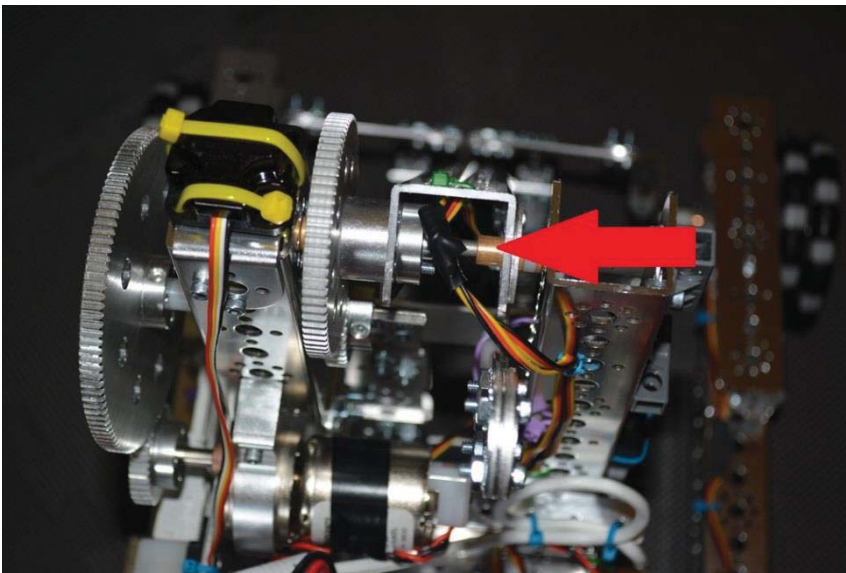
The wires for the sensors need to be tied to the robot frame to insure that they will not be damaged or become snagged on something. Take the time to tie the wires down so that they will be safe. In these photos, 4" zip ties were used. This size is not included in the kit. Many other methods would work just as well. For example, string or tape could be used to attach the wires. Make sure that the method meets rules outlined in the game

**Gracious Professionalism** - *"Doing your best work while treating others with respect and kindness - It's what makes FIRST, first."*

rules. The following photo shows the wires for the optical sensor tied to the robot frame to prevent them from being a snag hazard.



Also, any wires that cannot be completely moved out of places where they will rub and may be damaged should be protected in some way. For example, the servo wires that extend up the arm will rub against the end of the channel. Electrical tape wrapped around the wires will help to protect them as shown in the following photo.



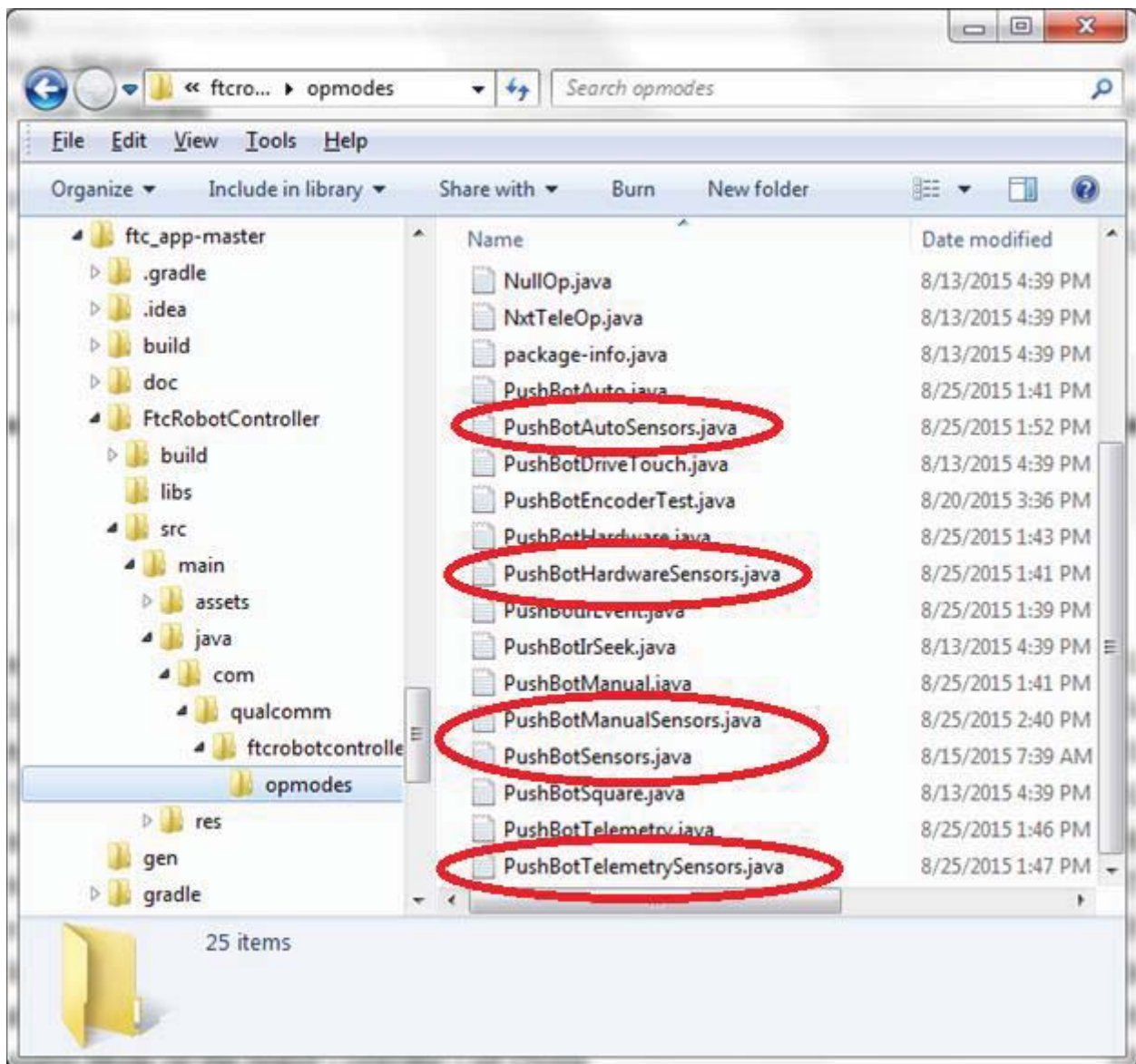


## Installing the PushBot Sensor op-modes

The PushBot op-modes that use sensors are part of the FTC SDK and are available from GitHub. If the installed version has the PushBot[...]Sensors.java files, then this section can be skipped. If a previous version has been downloaded that does not contain the new PushBot[...]Sensors.java files, then a new FTC SDK should be downloaded. If changes have been made to the op-modes, then the op-mode files should be backed-up before the new version is installed and then copied/merged into the new version.

### ***Determining whether the PushBot Sensor Op-modes are available***

Step 1: Using a file manager, open the op-modes folder of the FTC SDK. The image below was generated using Windows 7's Windows Explorer. If the folder contains JAVA files that begin with PushBot and have Sensors in the name, then the installed FTC SDK has the PushBot Sensor files – skip the rest of this section. If not, continue to the next step.





Step 2: If modifications have been made to the original op-modes, then make a back-up of the op-modes before deleting the old FTC SDK.

Step 3: Follow the instructions in the 'Installing the FTC SDK' section from the original PushBot Guide.

Step 4: If modifications were made and a back-up exists, then copy the modified op-modes back into the opmodes folder of the newly installed FTC SDK. Upgrades may have been made to the newly downloaded files. It might be better to merge changes into the newly downloaded opmodes instead of copying over them.

Step 5: Follow the instructions in the 'Deploying the Robot Controller' section from the original PushBot Guide.

Step 6: Follow the instructions in the 'Deploying the Driver Station' section from the original PushBot Guide. The Driver Station and Robot Controller should always match or something unexpected may happen.

## Configuring the Robot Controller

The software won't be able to communicate with the sensors until the sensors are configured using the Robot Controller application. The names assigned to the sensors will be needed by the PushBotHardwareSensors class or program exceptions can result. The names are case sensitive and must match exactly to those listed in the software.

Step 1: Follow the steps 1- 22 in the PushBot Build Guide for "Configuring the USB devices" to restore the drive wheel motors, the arm motor and the servo motors. A new file will be generated that will contain the original contents, but also contain the sensors. With step 22, the configuration will consist of the drive\_controller, arm\_controller and servo\_controller. Don't save the configuration yet; continue with the step below.

Step 2: Select 'Core Interface Module 4'. Note: The number shown here may be different.

**Active Configuration File:** No current file!

Scan

Press this button to scan for attached devices

Devices:

Motor Controller 2

Servo Controller 1

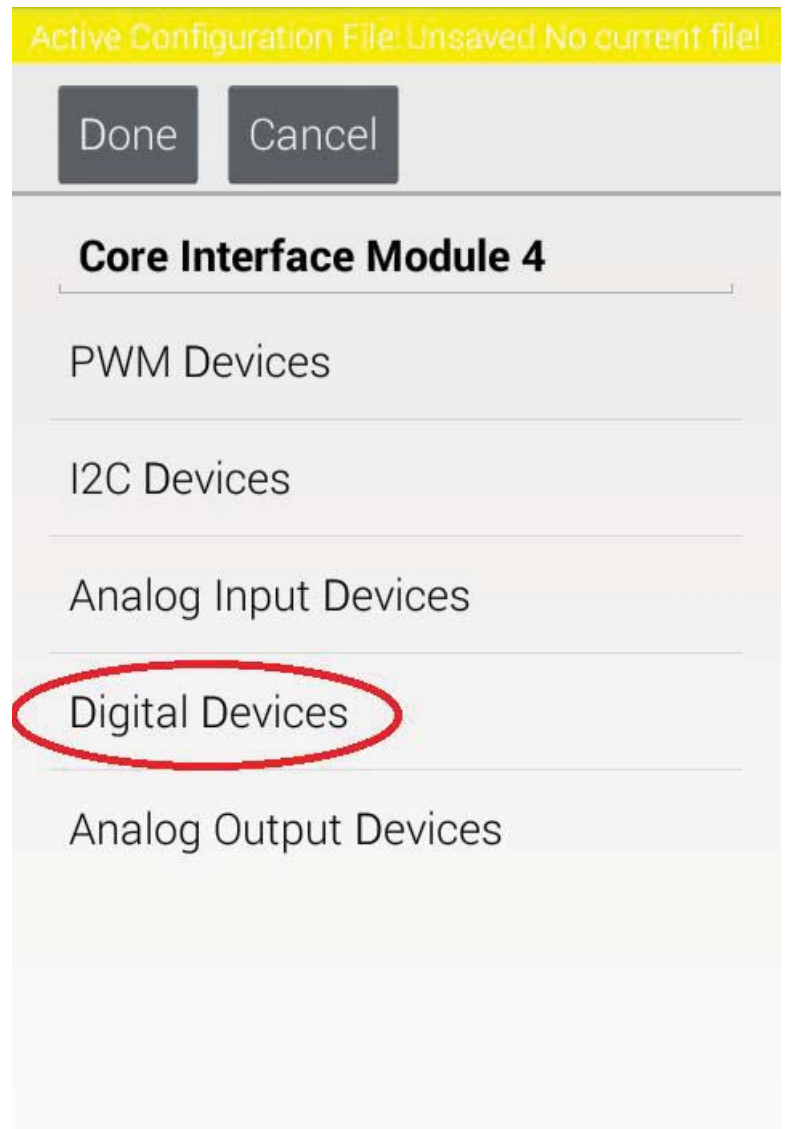
**Core Interface Module 4**

Motor Controller 3

Save Configuration

Press this button to write the current configuration to an XML file

Step 3: Select 'Digital Devices'.





Step 4: Scroll downward until the controls for port zero (0) are visible. Use the drop down control to select 'TOUCH\_SENSOR'.



The screenshot shows the 'Active Configuration File: Unsaved No current file!' window. It has a 'Done' and 'Cancel' button at the top. Below are four port configuration sections, numbered 3, 2, 1, and 0. Each section has a dropdown menu and a text field labeled 'Device name'. The dropdown for port 1 is open, showing options: 'NOTHING', 'TOUCH\_SENSOR' (circled in red), and 'DIGITAL\_DEVICE'. The 'Device name' field for port 1 contains the text 'HED'.

Port	Dropdown Selection	Device name
3	NOTHING	
2	NOTHING	
1	TOUCH_SENSOR	HED
0	NOTHING	

Step 5: Select 'Enter device name here' to change the name to 'sensor\_touch'.



Step 6: Select 'Done' to return to the module page.

Active Configuration File: Unsaved No current file!

Done Cancel

3 NOTHING  
NO DEVICE ATTACHED  
Device name

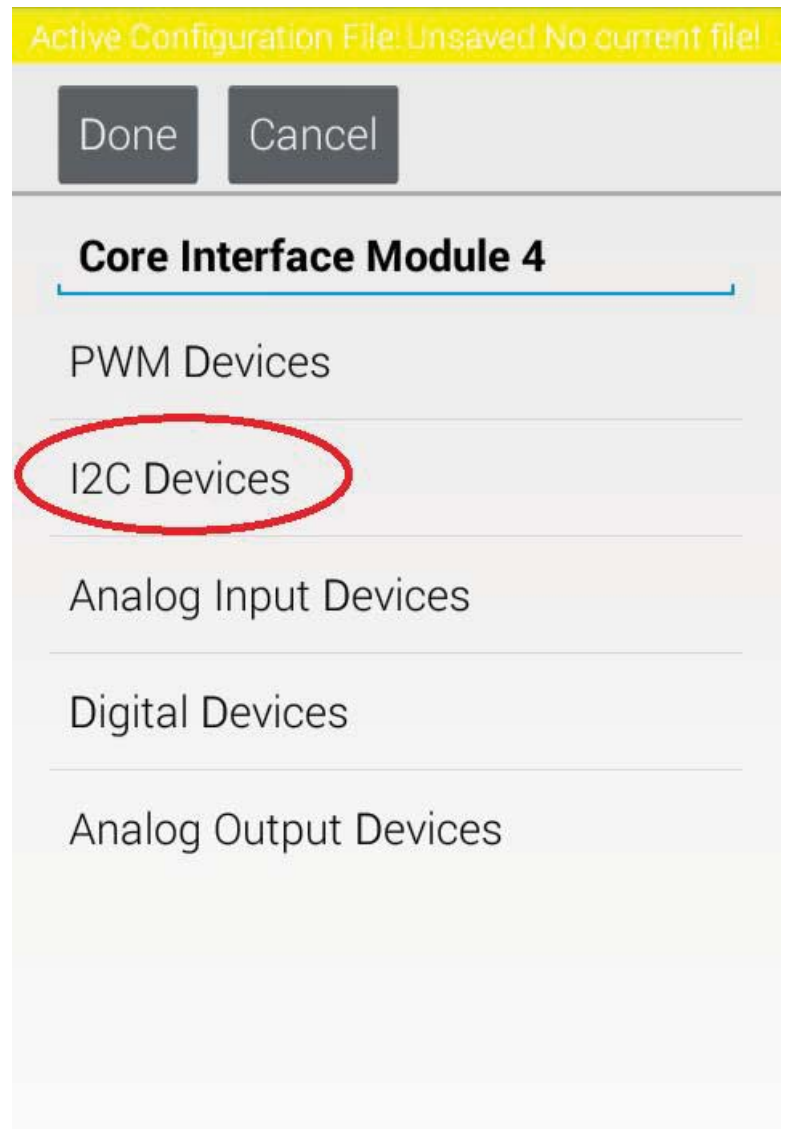
2 NOTHING  
NO DEVICE ATTACHED  
Device name

1 NOTHING  
NO DEVICE ATTACHED  
Device name

0 TOUCH\_SENSOR  
sensor\_touch  
Device name



Step 7: Select 'I2C Devices'.



Step 8: Scroll downward until the controls for port zero (0) are visible. Use the drop down control to select 'IR\_SEEKER\_V3'.

The screenshot shows the 'Active Configuration File: Unsaved No current file!' dialog box. It contains four port configuration sections, numbered 3, 2, 1, and 0 from top to bottom. Each section has a dropdown menu and a text input field labeled 'Device name'. The dropdown for port 1 is open, showing three options: 'NOTHING', 'IR\_SEEKER\_V3' (which is circled in red), and 'I2C\_DEVICE'. The other dropdowns are currently set to 'NOTHING'.

Step 9: Select 'Enter device name here' to change the name to 'sensor\_ir'.





Step 10: Select 'Done' to return to the module page.

Active Configuration File: Unsaved No current file!

Done Cancel

3 NOTHING ▼

NO DEVICE ATTACHED

Device name

2 NOTHING ▼

NO DEVICE ATTACHED

Device name

1 NOTHING ▼

NO DEVICE ATTACHED

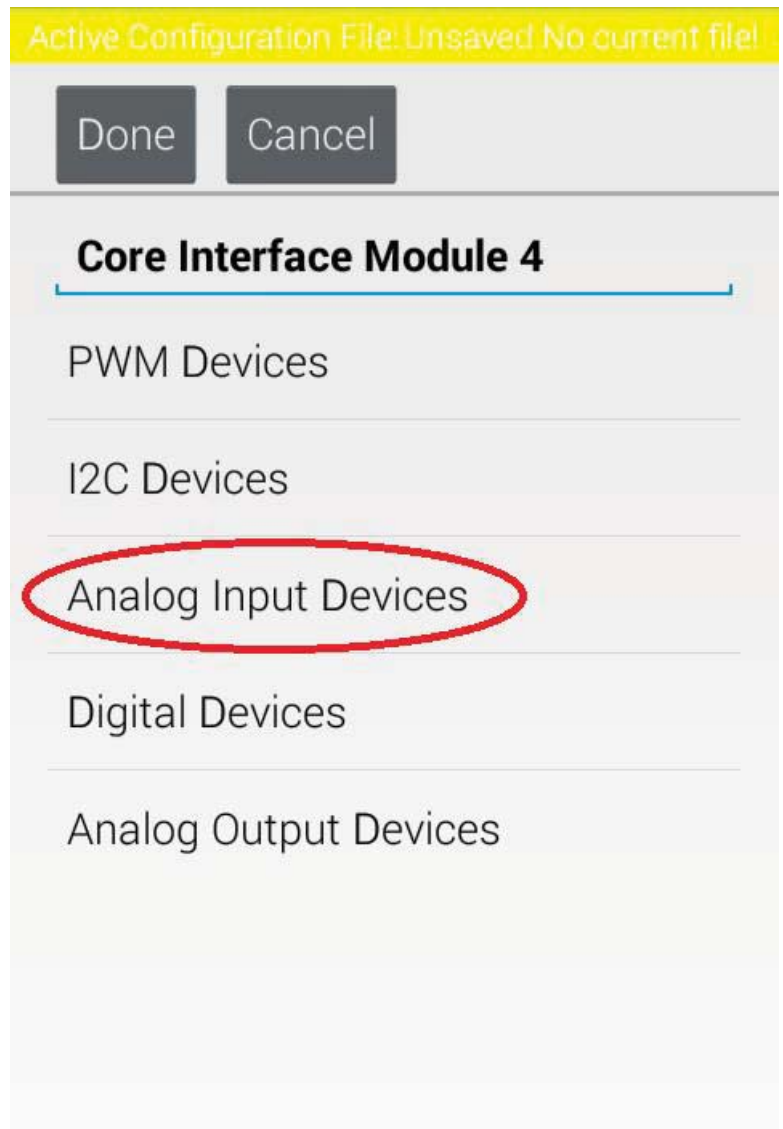
Device name

0 IR\_SEEKER\_V3 ▼

sensor\_ir

Device name

Step 11: Select 'Analog Input Devices'.



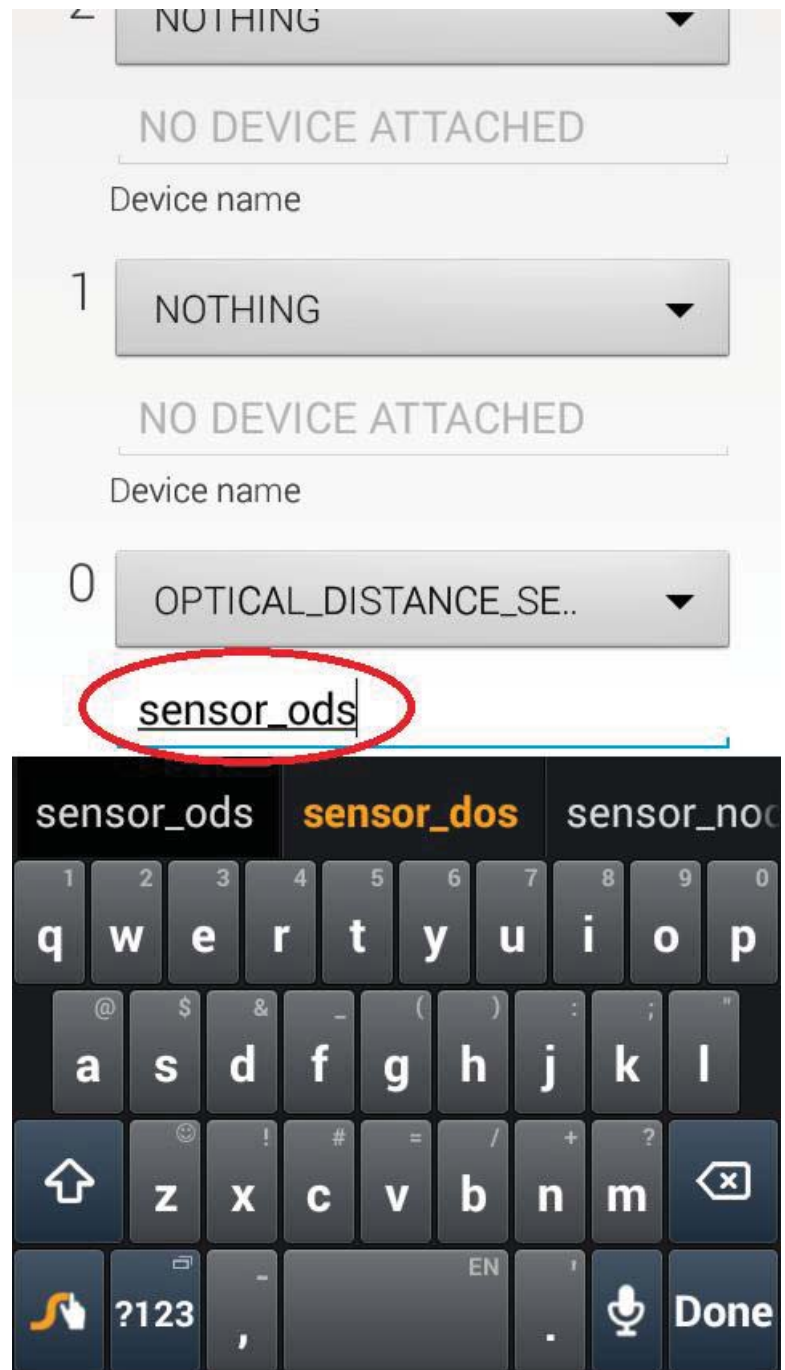
Step 12: Scroll downward until the controls for port zero (0) are visible. Use the drop down control to select 'OPTICAL\_DISTANCE\_SENSOR'.

The screenshot shows a configuration window titled "Active Configuration File: Unsaved No current file!". It contains four rows, each representing a port (3, 2, 1, 0). Each row has a dropdown menu and a text field labeled "Device name". The dropdown for port 1 is open, showing a list of options: "NOTHING", "OPTICAL\_DISTANCE\_SENSOR", and "ANALOG\_INPUT". The option "OPTICAL\_DISTANCE\_SENSOR" is circled in red. The other ports (3, 2, 0) have their dropdowns set to "NOTHING".

Port	Selected Device	Device name
3	NOTHING	
2	NOTHING	
1	OPTICAL_DISTANCE_SENSOR	
0	NOTHING	



Step 13: Select 'Enter device name here' to change the name to 'sensor\_ods'.



Step 14: Select 'Done' to return to the module page.

Active Configuration File: Unsaved No current file!

Done Cancel

3 NOTHING  
NO DEVICE ATTACHED  
Device name

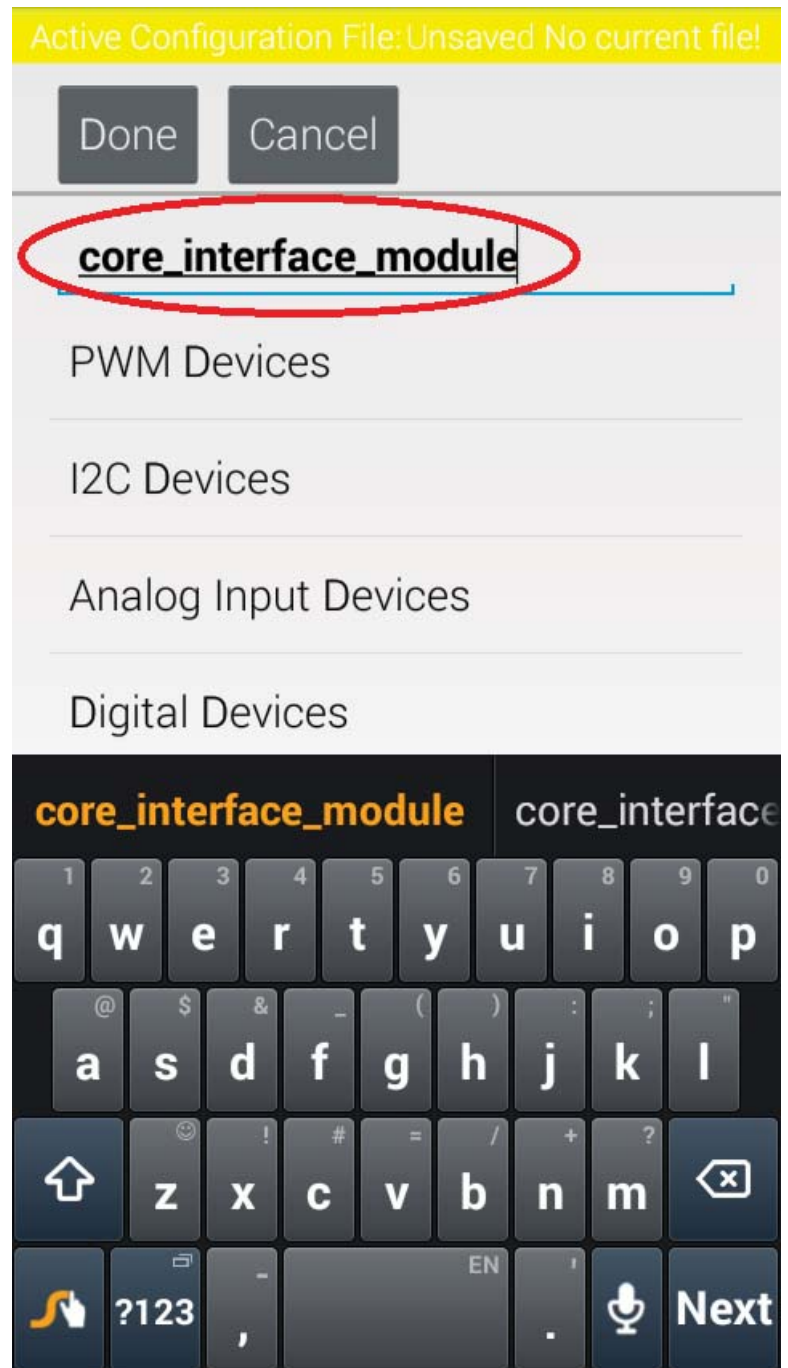
2 NOTHING  
NO DEVICE ATTACHED  
Device name

1 NOTHING  
NO DEVICE ATTACHED  
Device name

0 OPTICAL\_DISTANCE\_SE..  
sensor\_ods  
Device name

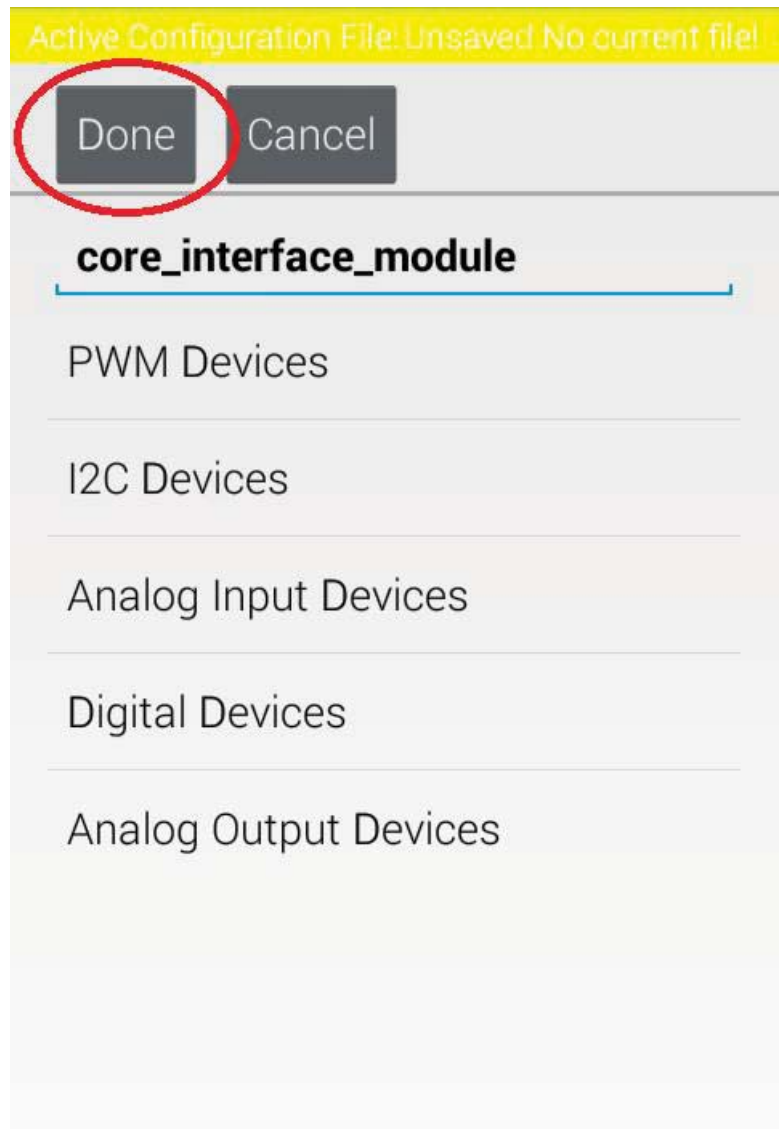
Step 15: Select 'Core Interface Module 4' (just below the Done and Cancel buttons) to change the name to 'core\_interface\_module'.

Note: the number shown ('4') may vary.

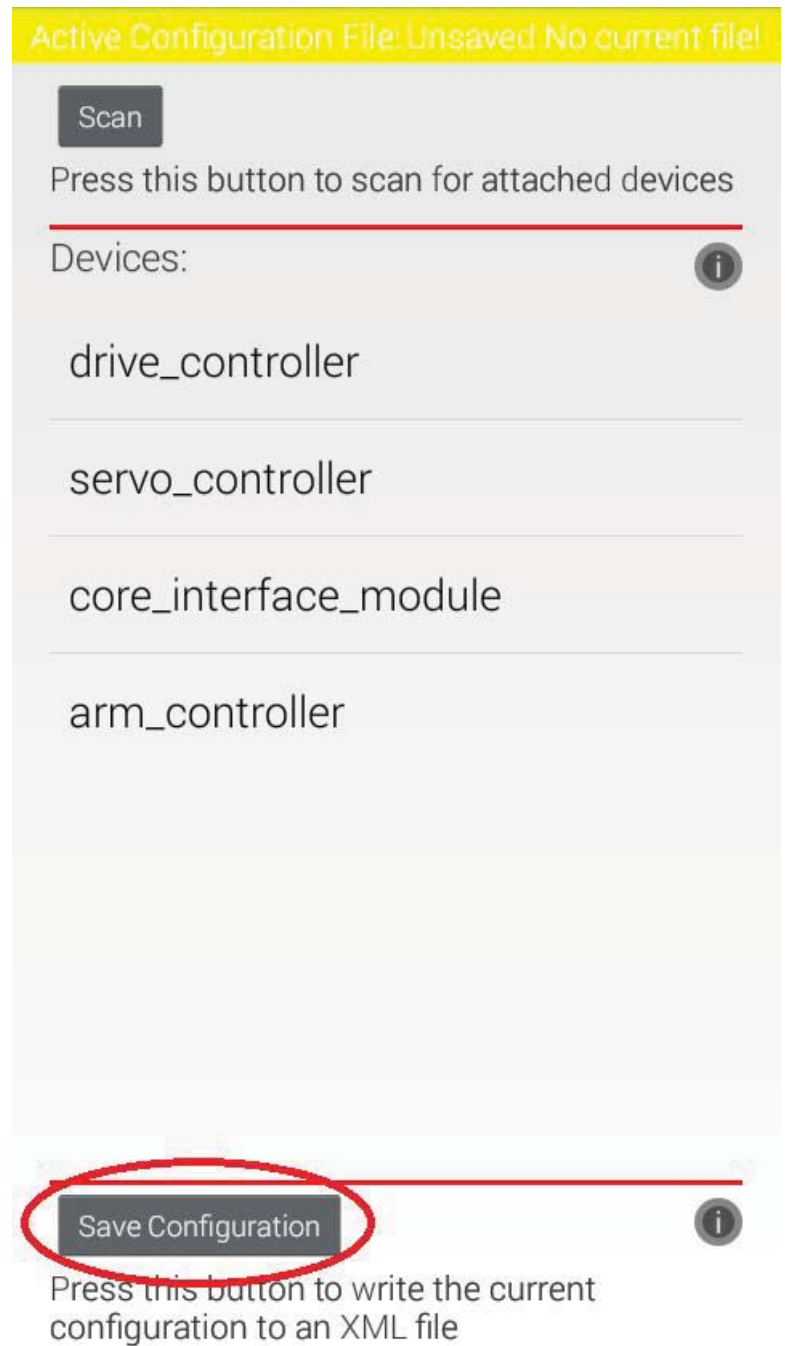




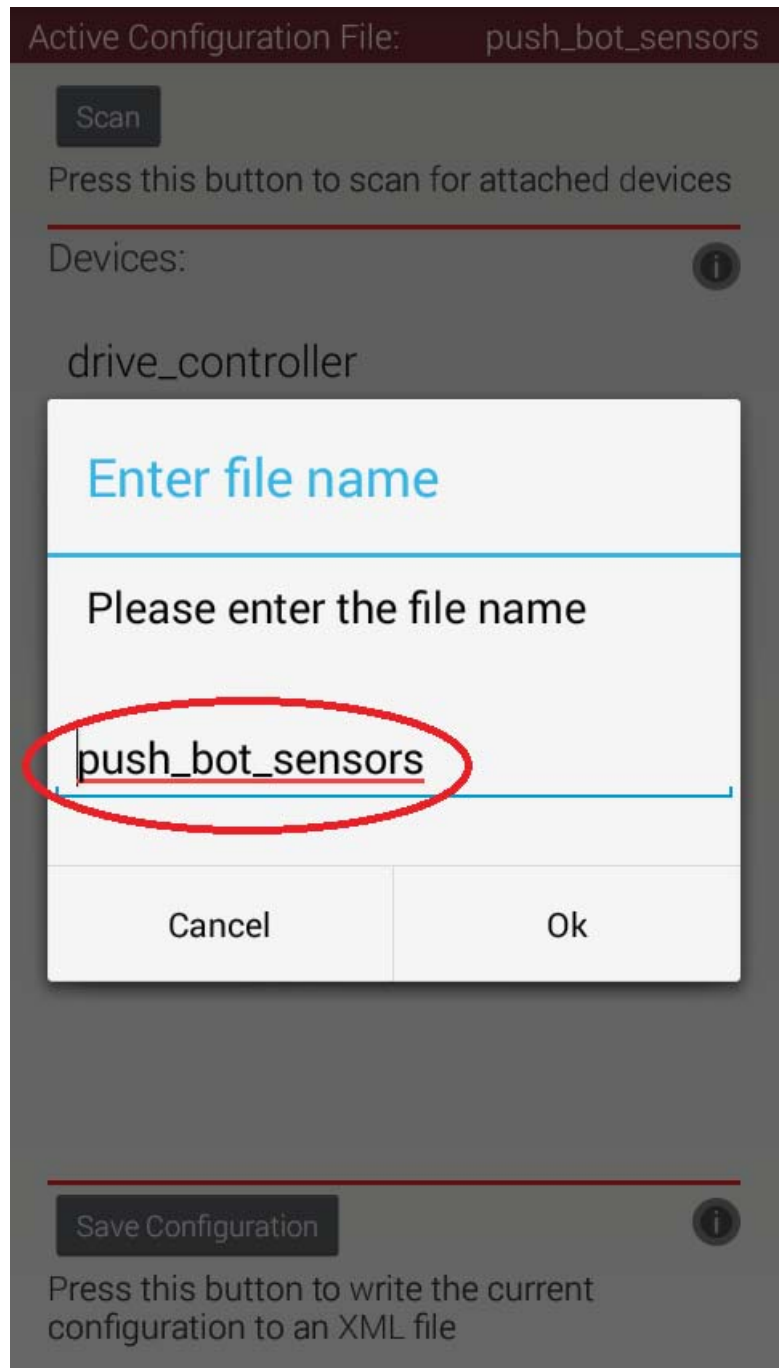
Step 16: Select 'Done' to return to the devices page.



Step 17: Select 'Save Configuration'.



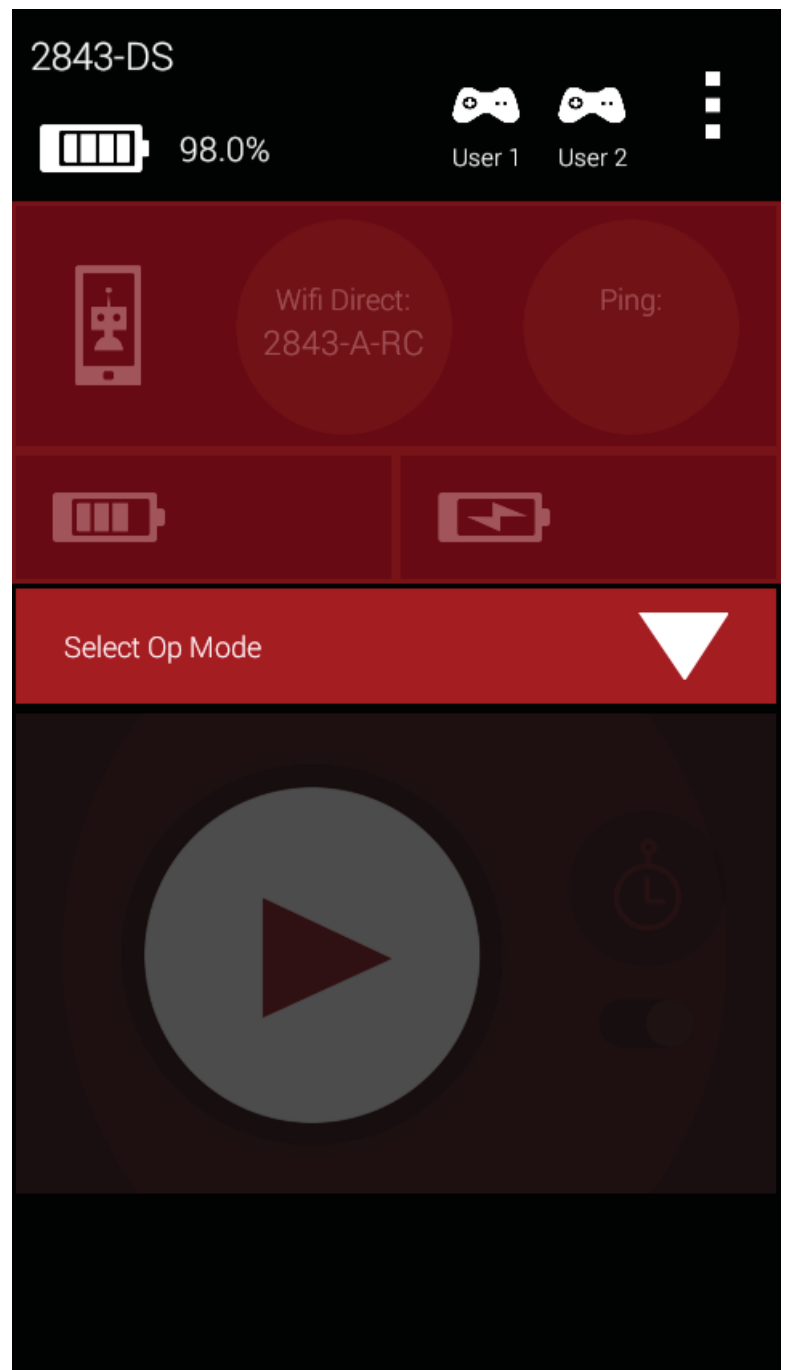
Step 18: Enter 'push\_bot\_sensors' as the configuration's file name. Select 'Ok'. Notice that push\_bot\_sensors is the active configuration.



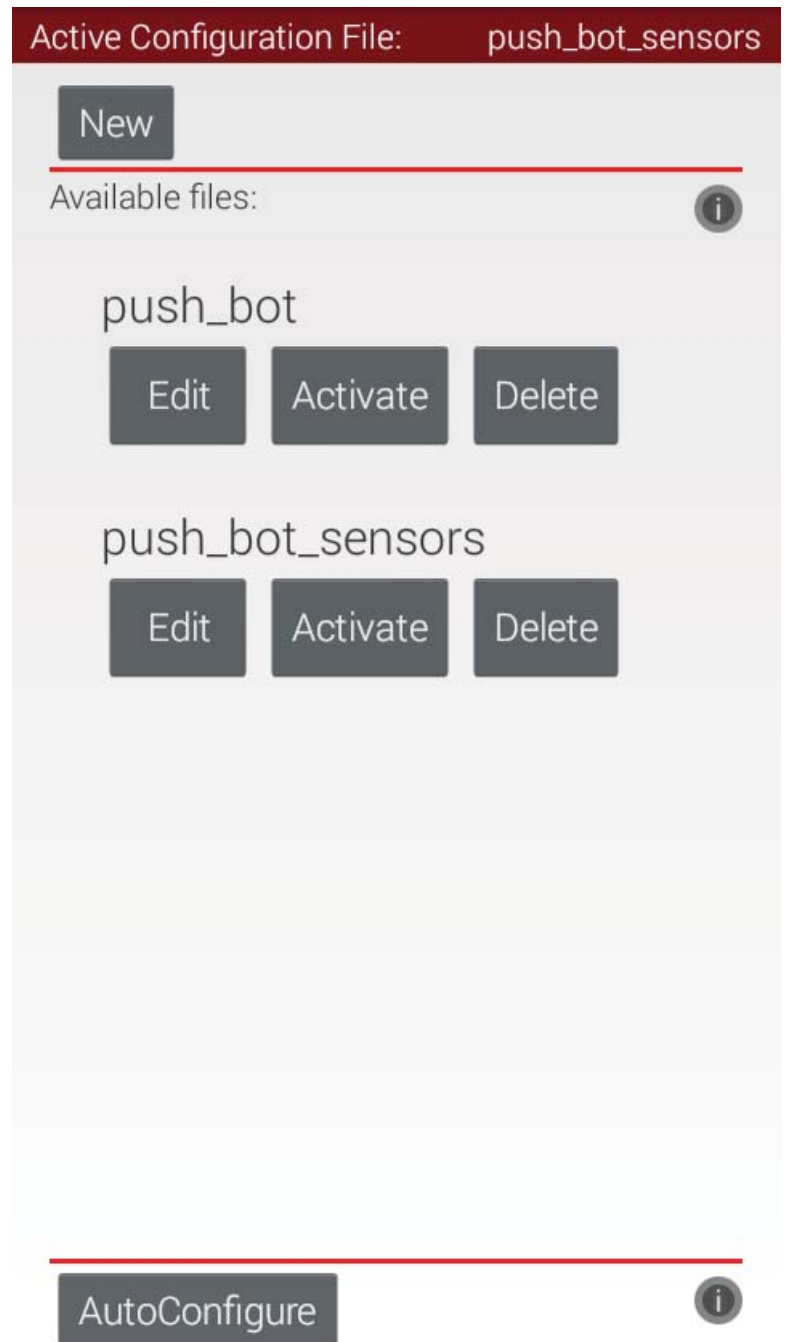


Step 19: If the Driver Station shows that the Robot Controller app is running, but unavailable (i.e. the WiFi Direct and Ping controls show information about the RC, but aren't a bright red), then perform the next three steps.

Note: This image is from the Driver Station phone. The previous and next images are of the Robot Controller phone.



Step 20: Use the phone's back button to return to the page showing a list of available files.



Step 21: Use the phone's back button to return to the Wi-Fi Selection page.

## WIFI CHANNEL SELECTION

---

Change Wifi Channel

## ROBOT CONFIGURATION SETTINGS

---

Configure Robot

---

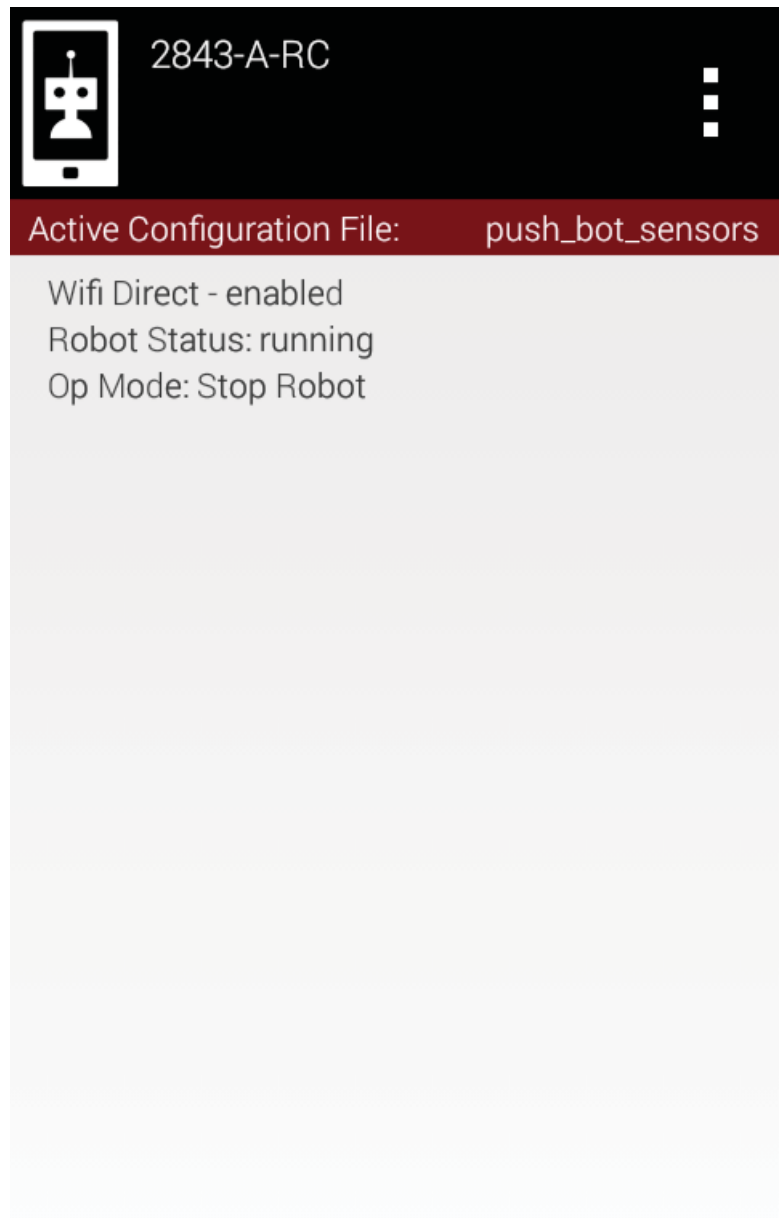
Autoconfigure Robot

---



Step 22: Use the phone's back button to return to the Robot Controller's main page.

This should cause the Driver Station phone to begin showing connection information.



## Integrating the op-modes and electronics

---

When the Robot Controller app configures the USB devices (as was done in the previous section), the devices are assigned randomly – the order depends on which controller responds to the query first. This means that sometimes when the devices are first scanned, motor 'A' is named Motor Controller 1, Motor Controller 2 or Motor Controller 3. This section will determine which motor controller operates the wheels and which operates the arm – there is a chance the motor controller names have been assigned to the wrong serial number. Once the USB devices are named, the serial number of the device is used to control communication, not the order in which the devices are discovered.

Step 1: Raise the PushBot off the floor using a platform. Make sure the robot won't move for the following steps.

Step 2: Disconnect the motor power connectors from the arm motor controller (the wires between the controller and the motor, not the wires from the power distribution module to the motor controller).

Step 3: On the Driver Station cell phone start the Driver Station app. Push the 'Select Op Mode' button.

Step 4: Make sure the gamepads have been assigned (i.e. start-a and start-b).

Step 5: Select 'PushBotManual' from the list of available op-modes.

Step 6: Push the 'Start' button.

Step 7: Push the left joystick of gamepad one up or down. If the motor moves, then the configuration is correct. If the wheel doesn't move, push the left joystick of gamepad two up or down. If the motor moves, then the configuration needs to be corrected – follow the next steps.

Step 8: Change the 'drive\_controller' name to 'new\_arm\_controller' using the configuration pages of the Robot Controller application.

Step 9: Change the 'arm\_controller' name to 'drive\_controller'.

Step 10: Under 'drive\_controller' change 'left\_arm' to 'left\_drive'. Add 'right\_drive' as the other motor's name.

Step 11: Change 'new\_arm\_controller' to 'arm\_controller'.

Step 12: Under 'arm\_controller' change 'left\_drive' to 'left\_arm'. Make the other name blank.

Step 13: Plug the arm motor power connectors back into the motor controller.

Step 14: Test the controls to make sure gamepad 1's joysticks drive the robot's wheels properly, gamepad 2's joystick operates the arm properly, and gamepad 2's buttons operate the hand properly.

## Testing the Robot

---

It is important to test changes in a controlled environment. During the testing of the touch sensor, the first attempts at coding caused the arm to move upwards, touch the sensor, and continue going until it was stopped by the back frame of the robot! If power had not been removed, the motor may have smoked. During the testing of the IR sensor, the robot didn't detect the beacon properly and drove across the floor and into a wall.

When testing the robot, always be prepared to remove power from the system. The PushBot's Power Distribution Module is mounted on the side of the robot. Its power switch is located on the bottom of the module. When the robot is moving, it's hard to turn the power off. Raise the robot's wheels off the surface when testing. A small box, wooden blocks or stack of books might be used to raise the robot. Make sure that the resulting pedestal is stable before using it. Using a pedestal will keep the robot from moving and allow someone to use the Power Module's switch to remove power. Do NOT unplug the battery as its connector is directly in line with the movement of the arm.

### ***Testing the touch sensor***

Step 1: Make sure the arm is in its lowest position (i.e. the hand is between the two omni-wheels).

Step 2: Run the Robot Controller application and Driver Station application (shown in the original PushBot Guide).

Step 3: Be prepared to turn power off using the Power Distribution Module's power switch. Do NOT unplug the battery as its connector is directly in line with the movement of the arm.

Step 4: Select the PushBotTouchEvent op-mode using the Driver Station. If the arm doesn't stop as expected, verify that the light inside the touch sensor comes on when the trigger is depressed. Verify that the sensor is plugged into digital port zero. Further debugging hints can be found on the FTC Forum. If none of the posts match your symptoms, post a question on the forum. The forum and directions on its use can be found at <http://ftcforum.usfirst.org/forum.php>.

### ***Testing the IR sensor***

To fully test the IR sensor, an IR beacon is required, but is not supplied in the kit. The team will need to find one before being able to test. If an IR source is available (and portable), use the PushBotIrrEvent op-mode to perform a simple test. Place the beacon in front of the robot. Make sure that the wheels turn in the proper direction when the sensor is moved from left to right and right to left in front of the robot. Also make sure that the robot stops when the source is near the front wheels of the robot – this may not be useful in a competition setting because the strength depends on the beacon's power level.

Again, debugging hints can be found on the FTC Forum. If none of the posts match your symptoms, post a question on the forum.

### ***Testing the ODS sensor***

To fully test the ODS sensor, a grey foam tile and white tape are recommended, but is not supplied in the kit. Use the PushBotOdsEvent op-mode for testing. Place the foam tile under the sensor without the tape being under the sensor. Make sure that the wheels drive forward. Move the tape under the sensor and the wheels should stop. Use the forum for debugging hints and more ideas on how to use the sensor.

### ***Testing the manual operation of the arm***

Refer to the original PushBot Guide to become familiar with the PushBot's basic controls, minimally repeated below. Run the PushBotManualSensors op-mode to test the touch sensor.



On gamepad 1, PushBoth joysticks away from your body and the Robot drives forward. Pull both joysticks toward your body and the Robot drives backward. Push one joystick up and the other down and the Robot turns.

On gamepad 2, push the left joystick away from the body to raise the arm; toward the body to lower the arm. Button 'X' opens the hand; button 'B' closes the hand. The 'Y' button (red circle) causes the arm to rise until the touch sensor is triggered.



### ***Testing a multi-sensor autonomous op-mode***

Using all these sensors together inside an op-mode is an art. It will require lots of design and testing time. The PushBotAutoSensors op-mode can provide an initial look at the way encoders, a touch sensor, an IR beacon and an ODS can work together using a state machine. Learning to draw state machines is a valuable skill and there are many on-line resources available to learn techniques of managing state machines.

After starting the op-mode, the robot should drive forward at full power until the drive wheel encoders reach 2880 counts. The original PushBot was able to do this using the PushBotAuto op-mode. The new PushBotAutoSensors op-mode introduces more functionality.

After the initial drive forward, the robot should turn left until the left encoder reaches -2880 and the right should reach 2880.

The next task is for the robot to drive until a white line has been detected. When the white line has been detected, the robot will transition to the next state.

The next step will be for the robot to drive towards an IR beacon. If an IR beacon is not available, then this portion will fail. Comment the 'int l\_status...', 'if (l\_status...', 'else if...' (and the rest of the else block) to make the hand open.

The arm is controlled by the update\_arm\_state method when the loop method is first called. This allows the arm to move at the same time the wheels begin the forward drive using encoders. If the robot does something unexpected take note of the state telemetry displayed on the Driver Station. This may help diagnose the code that is causing the problem.

# **2015-2016 *FIRST*® Tech Challenge PushBot Build Guide TETRIX Sensors Supplement**

## **Appendices**

## Appendix A: Bill of Materials List for Additions to the PushBot





This list only includes the items needed to build components shown in this document. All changes relate to adding sensors to the original PushBot.

Quantity	Name	Common Name
1	CORE_DEVICE_INTERFACE_MODULE	Core Device Interface Module
1	IR_SEEKER_V3	IR Seeker V3
1	MRI_TOUCH_SENSOR	Touch Sensor
1	MRI_OPTICAL_DISTANCE_SENSOR	Optical Distance Sensor
1	TETRIX_739061_2012	Flat Bracket
1	TETRIX_739062_2012	L Bracket
14	TETRIX_739094_2013	Kep Nut
10	TETRIX_739097_2012	1/2" Socket Head Cap Screw
4	TETRIX_739098_2012	5/16" Socket Head Cap Screw
3	TETRIX_739274_2013	64mm Flat
1	TETRIX_739387_2013	2mm Spacer
1	TETRIX_739073_2012	Flat Building Plate
1	39081	Servo Extension Wire



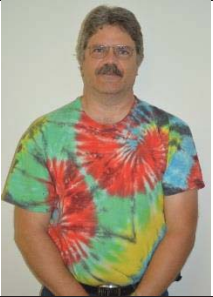

## Appendix B: Project Team Profiles

Our team would like to thank two very special people who spent hours reviewing the team's documentation. One is an alumna of the team, Renee Spangler. The other is a very good friend, Nathaniel Lahn.



	<b>Isa</b> Year 9 student at Great Mills High School. Student for 1 year on Team 2843, Under the Son
	<b>Alanis</b> Year 9 student at Great Mills High School. Student for 1 year on Team 2843, Under the Son
	<b>Nanette</b> Year 7 student at Esperanza Middle School. Student for 1 year on Team 2843, Under the Son
	<b>Laura</b> Year 9 student at Leonardtown High School. Student for 7 years on Team 2843, Under the Son



	<p><b>Mary</b> Year 12 student at Leonardtown High School. Student for 7 years on Team 2843, Under the Son Volunteer for 7 years, MD FIRST</p>
	<p><b>Claudia</b> Mentor for 1 year on Team 2843, Under the Son</p>
	<p><b>David</b> Mentor for 7 years on Team 2843, Under the Son Volunteer for 7 years, MD FIRST <a href="mailto:ssi@the-spanglers.net">ssi@the-spanglers.net</a></p>
	<p><b>Lydean</b> Coach for 7 years on Team 2843, Under the Son Volunteer for 7 years, MD FIRST <a href="mailto:ssi@the-spanglers.net">ssi@the-spanglers.net</a></p>

## Appendix C – Resources

---

**Game Forum Q&A** - <http://ftcforum.usfirst.org/forum.php>

**FTC Game Manuals – Part I and II** - <http://www.usfirst.org/roboticsprograms/ftc/Game>

**FIRST Headquarters Support**

Phone: 603-666-3906

Email: [FTCTeams@usfirst.org](mailto:FTCTeams@usfirst.org)

### **USFIRST.ORG**

[FIRST Tech Challenge \(FTC\) Page](#) – For everything FTC.

[FTC Volunteer Resources](#) – To access public Volunteer Manuals.

[FTC Season Timeline](#) – Find FTC events in your area.

### **FIRST Tech Challenge Social Media**

[FTC Twitter Feed](#) - If you are on Twitter, follow the FTC twitter feed for news updates.

[FTC Facebook page](#) - If you are on Facebook, follow the FTC page for news updates.

[FTC YouTube Channel](#) – Contains training videos, Game animations, news clips, and more.

[FTC Blog](#) – Weekly articles for the FTC community, including Outstanding Volunteer Recognition!

[FTC Team Email Blasts](#) – contain the most recent FTC news for Teams.

[FTC Google+](#) community - If you are on Google+, follow the FTC community for news updates.

## Feedback

---

We strive to create support materials that are the best they can be. If you have feedback regarding this manual, please email [ftcteams@usfirst.org](mailto:ftcteams@usfirst.org). Thank you!