2015-2016 *FIRST*® Tech Challenge

AdaFruit RGB Sensor Assembly and User's Guide

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	9/12/2015	Initial Release – V. 003

Contents

Introduction

What is the FIRST Tech Challenge?

FIRST Tech Challenge is a student-centered activity that focuses on giving students a unique and stimulating experience. Each year, Teams participate in a new Game that requires them to design, build, test, and program autonomous and driver-operated Robots that must perform a series of tasks.

The Playing Field for the Game consists of the *FIRST* Tech Challenge Game Pieces set up on a foam-mat surface, surrounded by a metal and Lexan Field frame. Each Tournament features Alliances, which are comprised of two Teams, competing against one another on the Playing Field. Teams work to overcome obstacles and meet challenges, while learning from and interacting with their peers and adult Mentors. Students develop a greater appreciation of science and technology and how they might use that knowledge to impact the world around them in a positive manner. They also cultivate life skills such as:

FTC is More Than Robots! While competing, students develop personal and professional skills they will be able to rely on throughout their life.

- Planning, brainstorming, and creative problem-solving.
- Research and technical skills.
- Collaboration and Teamwork.
- Appreciation of differences and respect for the ideas and contributions of others.

To learn more about FTC and other *FIRST* Robotics Competitions, visit www.usfirst.org.

FIRST Tech Challenge (FTC) Core Values

Volunteers are integral to the *FIRST* community. The *FIRST* Tech Challenge relies on Volunteers to run the program at many levels, from managing a region to Mentoring an individual Team. FTC Affiliate Partners coordinate the program in each region or state. These FTC Partners fundraise, run Tournaments, hold workshops and demonstrations, market FTC locally, handle public relations, and recruit Volunteers and Teams. They are a tremendous resource for Mentors and FTC would not exist without them.

FIRST asks everyone who participates in FTC to uphold the following values:

- We display Gracious Professionalism with everyone we engage with and in everything we do.
- We act with integrity.
- We have fun.
- We are a welcoming community of students, mentors, and volunteers.
- What we learn is more important than what we win.
- We respect each other and celebrate our diversity.
- Students and adults work together to find solutions to challenges.
- We honor the spirit of friendly Competition.
- We behave with courtesy and compassion for others at all times.
- We act as ambassadors for *FIRST* and the *FIRST* Tech Challenge.
- We inspire others to adopt these values.

Gracious Professionalism^{TM}

FIRST uses this term to describe the program's intent. This is one of the most important concepts that can be taught to a young person who is learning to get along in the work world. At *FIRST*, Team members help other Team members, but they also help other Teams.

Gracious Professionalism is not clearly defined for a reason. It can and should mean different things to everyone.

Some possible meanings of Gracious Professionalism include:

- Gracious attitudes and behaviors are win-win.
- Gracious folks respect others and let that respect show in their actions.
- Professionals possess special knowledge and are trusted by society to use that knowledge responsibly.
- Gracious Professionals make a valued contribution in a manner pleasing to others and to themselves.

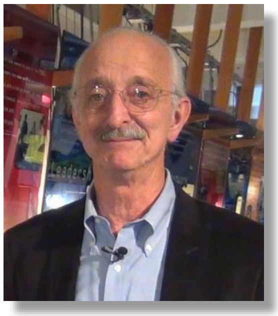
An example of Gracious Professionalism is patiently listening to a Team's question and providing support despite having several pressing things to do on the day of the Event.

In the context of *FIRST*, this means that all Teams and participants should:

- Learn to be strong competitors, but also treat one another with respect and kindness in the process.
- Avoid leaving anyone feeling as if they are excluded or unappreciated.
- Knowledge, pride and empathy should be comfortably and genuinely blended.

In the end, Gracious Professionalism is part of pursuing a meaningful life. When professionals use knowledge in a gracious manner and individuals act with integrity and sensitivity, everyone wins, and society benefits.

Watch Dr. Woodie Flowers explain Gracious Professionalism in this short video.



"The FIRST spirit encourages doing high-quality, well-informed work in a manner that leaves everyone feeling valued. Gracious Professionalism seems to be a good descriptor for part of the ethos of FIRST. It is part of what makes FIRST different and wonderful."

- Dr. Woodie Flowers, National Advisor for ***FIRST***

Youth Protection Program

The purpose of the *FIRST* Youth Protection Program (*FIRST* YPP) is to provide coaches, mentors, volunteers, employees, others working in *FIRST* programs, Team members, parents, and guardians of Team members with information, guidelines, and procedures to create safe environments for everyone participating in *FIRST* programs.

The *FIRST* YPP sets minimum standards recommended for all *FIRST* activities. Adults working in *FIRST* programs must be knowledgeable of the standards set by the *FIRST* YPP, as well as those set by the school or organization hosting their Team.

Youth Protection Expectations and Guidelines

Coaches and Mentors are expected to read and follow elements in the FIRST Youth Protection Program guide that are labeled as required are mandatory in the United States and Canada, and may not be waived without the approval of the *FIRST* Youth Protection Department.

FIRST recommends that the standards set forth in the FIRST Youth Protection Program guide be applied outside of the United States and Canada to the extent possible. At a minimum, local regulations regarding youth protection must be complied with.

Everyone working with FIRST Teams should be familiar with the *FIRST* YPP policies.

Forms are available here: http://www.usfirst.org/aboutus/youth-protection-program

Information on the US Screening process is available here: http://www.usfirst.org/sites/default/files/uploadedFiles/About_Us/US-Youth-Protection-Clearance-Process.pdf

Information on the Canadian Screening process is available here:

http://www.usfirst.org/sites/default/files/uploadedFiles/About_Us/Canadian-Youth-Protection-Clearance.pdf

You can find FAQ and additional information about the *FIRST* Youth Protection Program on the FIRST website at: http://www.usfirst.org/aboutus/youth-protection-program

NOTICE OF NON-DISCRIMINATION

United States Foundation for Inspiration and Recognition of Science and Technology (*FIRST*®) does not discriminate on the basis of race, color, national origin, sex, disability, or age in its programs and activities. The following person has been designated to handle inquiries regarding the non-discrimination policies: Lee Doucette, Youth Protection Program Manager, 200 Bedford Street, Manchester, NH 03101, 603-666-3906, Ext. 250.

The AdaFruit RGB Sensor

For the 2015-2016 season, Teams who participate in the *FIRST* Tech Challenge (FTC) Robot Competition are able to use certain commercially available, off-the-shelf Sensors in conjunction with the Modern Robotics Core Device Interface (CDI) module. One of the Sensors that can be used is the AdaFruit RGB Color Sensor (see http://www.AdaFruit.com/products/1334). The Sensor is sold without a protective shell or connector cables. This document contains instructions on how to print a simple 3D housing for the Sensor and how to create a set of connector cables that can be used to connect the Sensor to the Modern Robotics CDI Module.

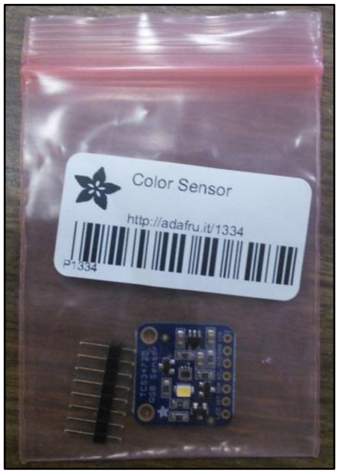


Figure 1 - The AdaFruit RGB Sensor is sold as a board without any protective case or a connector cable.

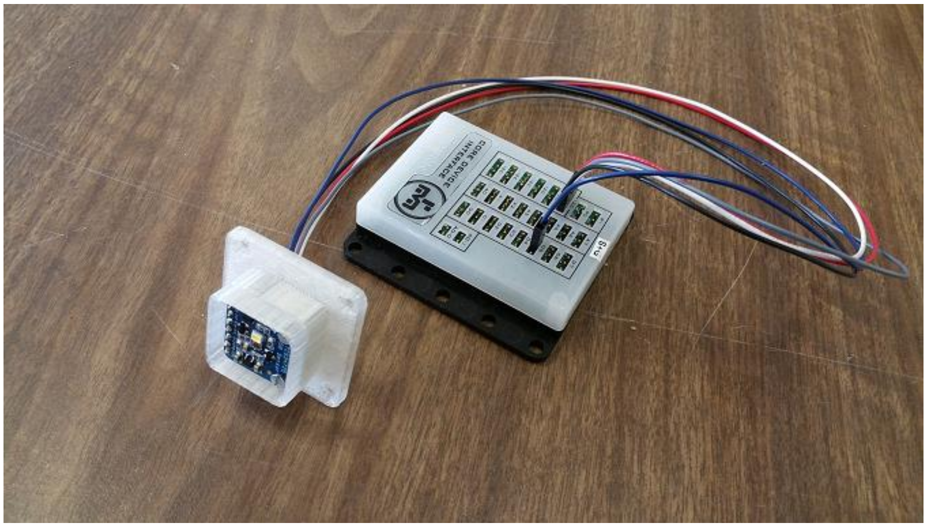


Figure 2 – This guide shows to assemble the Sensor into a mountable housing & how to create a connector cable

Caution!

This assembly guide requires soldering to connect the Sensor to some header pins. Always wear your safety glasses! The soldering iron gets hot – care should be taken to avoid burning yourself or your work area. Do not use the soldering iron without proper adult supervision. Also, some brands of solder contain lead, which can be toxic if ingested. Be sure to clean up your work area and wash your hands after you are done soldering your pieces together.

Tools

The following table lists the tools needed to assemble the Sensor.

Item	Qty	Notes
Safety Glasses	1	Safety FIRST!
Diagonal cutters	1	
Soldering iron & solder	1	Only use with proper adult supervision.
Soldering stand or proto board	1	To hold pins and Sensor while soldering them together.
Wire strippers	1	Needs to be able to strip 24 AWG wire.
Terminal Crimper or Needle nose pliers	1	Required to crimp the terminal connectors onto the ends of wires. The Molex ServiceGrade™ Hand Crimp Tool for Terminals (part # 64016-0201) is recommended, but a pair of pliers will also do the job.
Small screw driver	1	
3D Printer	1	If you would like to print a plastic housing to hold the Sensor, you will need access to a 3D printer.

Table 1 - Tools needed for the job.

Parts

The following table lists the parts needed for the job.

Item	Qty	Notes
AdaFruit RGB Sensor	1	AdaFruit product ID #1334. Visit http://www.AdaFruit.com/products/1334 for details.
24AWG wire (red, black, white, gray, blue).	---	You need red, black, white, gray and blue wires to make a connector cable for the Sensor. The wires should be made only as long as needed to mount the Sensor properly on your Robot.
Crimp terminals	10	Molex SL Crimp Terminal, Series 70058, Female with Tin (Sn) plated Contact, 24-30 AWG. Molex Part # 16-02-0096 is a bag of these terminals. You will need 10 terminals (2 per wire).
7-circuit crimp terminal housing	1	To hold the terminals for the end of the cable that connects to the Sensor. Molex part # 50-57-9007.
4-circuit crimp terminal housing	1	To hold the terminals for the end of the cable that connects to the I2C port on the CDI module. Molex part #50-57-9004.
2-circuit crimp terminal housing	1	To hold the terminals for the end of the cable that connects to the digital port on the CDI module. Molex part #50-57-9002.
3D printed Sensor housing	1	You can download the 3D model for this part from the FTC Team Resources webpage. Go to http://www.usfirst.org/Roboticsprograms/ftc/Team-resources and scroll down to the **Sensor Resources** category and follow the link listed under this category to download a .zip file that contains the 3D model for the Sensor housing.

Table 2 - Parts needed for the job.

Soldering the Pins

Remove Extra Pin

The first step to setting up this Sensor is to solder the pins onto the Sensor. The Sensor has seven holes that will need to have pins soldered into them. A strip of eight pins comes with the Sensor so you will need to cut one pin off of the strip. Use diagonal cutters to remove this pin.

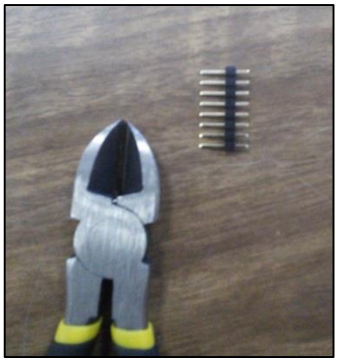


Figure 3 - The Sensor comes with a strip of 8 pins.

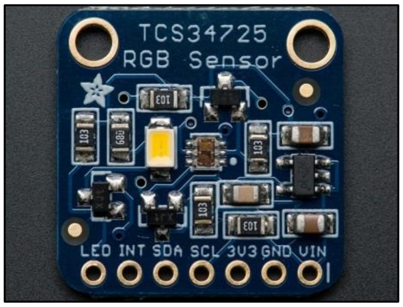


Figure 4 - The Sensor has 7 pins.

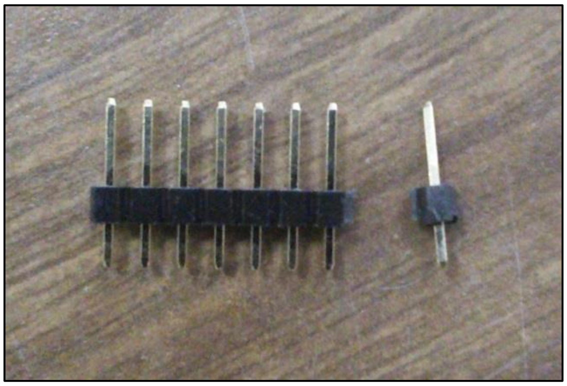


Figure 5 - Use the diagonal cutters to cut off one pin from either end.

Mount Sensor on Pins

Next you will need to place the 7-pin strip into the holes on the Sensor. The shorter side of the pins will come up through the bottom of the Sensor so that the plastic that holds the pins together is touching the bottom side.

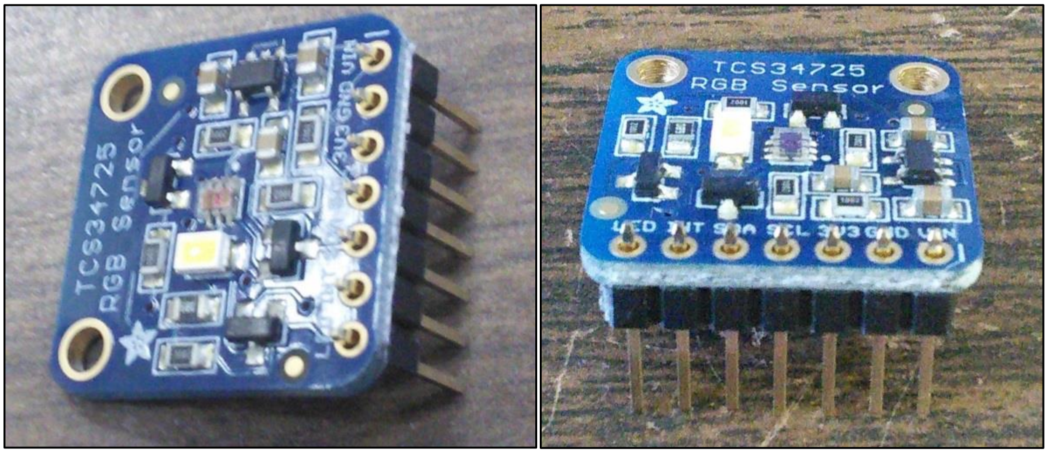


Figure 6 - Insert the shorter side of the pins through the bottom of the holes on the Sensor.

You will want to use a soldering stand (see Figure 7) to hold the pins and Sensor while you solder. The pins **will get very HOT** as you are soldering and should not be held by hand. If you do not have a soldering iron, you can use a proto board to hold the pins and Sensor in place while you solder (see Figure 8).

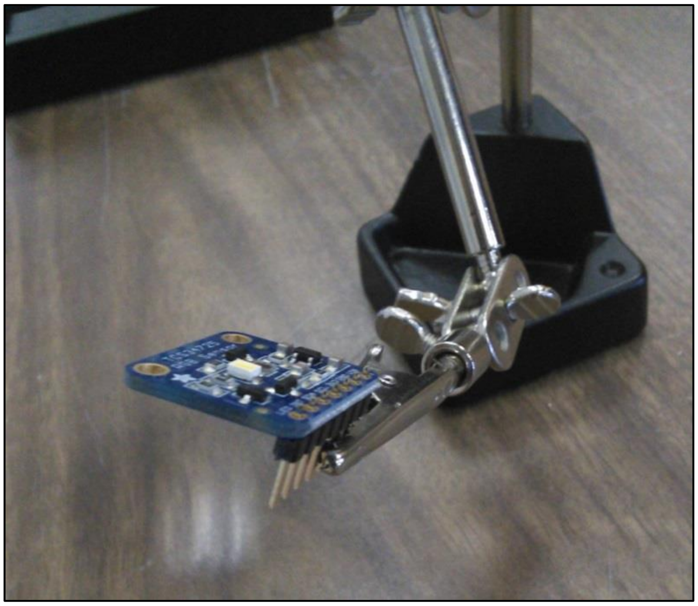


Figure 7 - Use a soldering stand to hold the pins and Sensor while you solder.

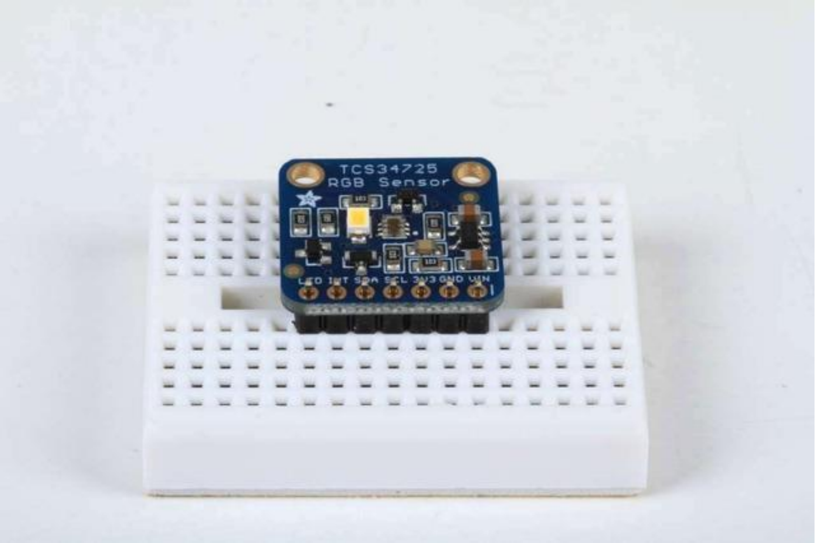


Figure 8 - You can also use a proto board to hold the pins and Sensor in place.[1]

Solder the Pins in Place

Heat up your first pin and add a small drop of solder to secure it into place. Make sure the solder does not flow onto another pin. Also make sure that the pins are now square to the Sensor. If at this point they are not square, just heat up the same pin to melt the solder holding them together, and align them correctly.

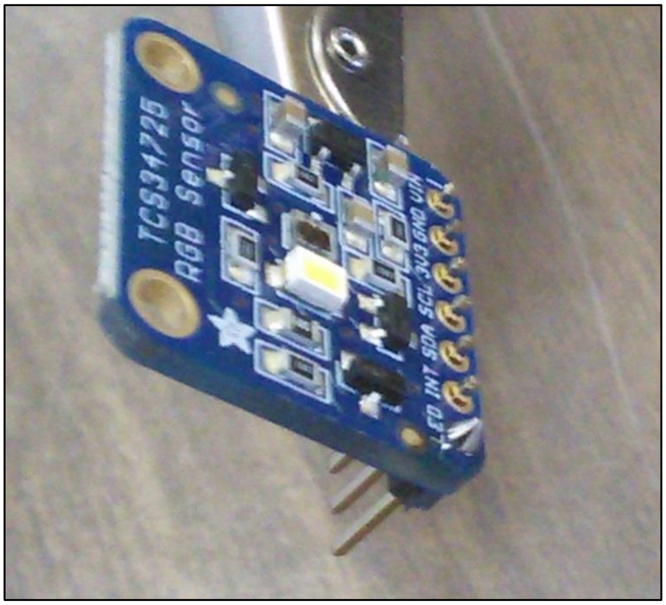


Figure 9 - Solder a pin in place and verify the proper alignment of the pins with respect to the board.

Once you like how the pins are sitting with the Sensor, you can continue to solder the rest of the pins into place. Be careful not to solder two pins together - that will cause a short and can prevent the Sensor from work properly.

[1] Image from https://learn.AdaFruit.com/AdaFruit-color-Sensors/assembly-and-wiring and was downloaded on 9/9/15.

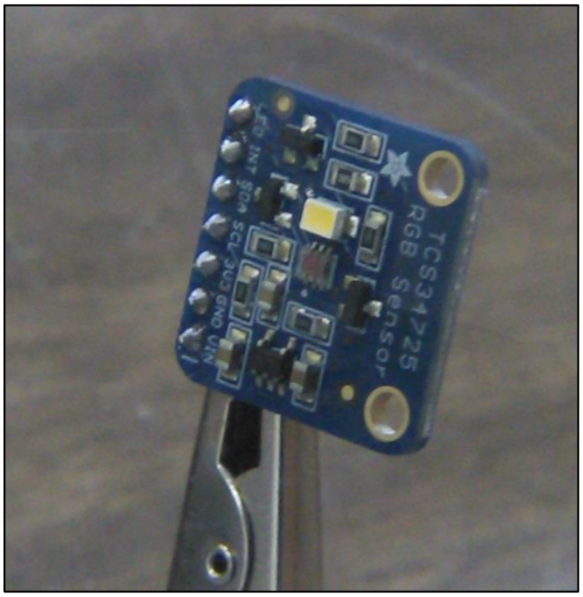


Figure 10 - Solder the remaining pins in place.

Making the Cable

Now that the pins are soldered to the Sensor, it's time to make the cable to connect the Sensor to the Core Device Interface. Let's take a look at the pins on the RGB Sensor board:

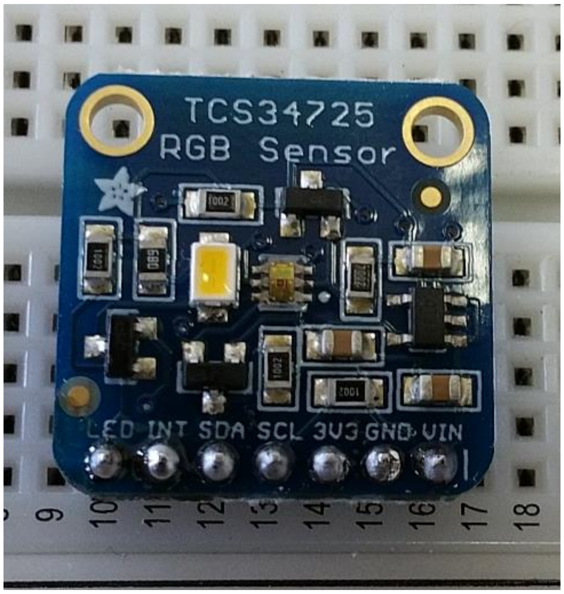


Figure 11 – The Sensor has 7 pins (LED, INT, SDA, SCL, 3V3, GND and VIN).

The RGB Sensor has 7 pins labeled, LED, INT, SDA, SCL, 3V3, GND and VIN. This Sensor is an I2C Sensor[2] which means that it can be connected to an I2C bus, like the one available on the Core Device Interface module. For our application, we are going to use the SDA (serial data), SCL (serial clock), GND (ground) and VIN (input voltage) pins to create a cable that can be used to plug into an I2C port on the CDI module.

[2] See http://www.nxp.com/documents/user_manual/UM10204.pdf for a detailed description of the I2C specifications.

We are also interested in the pin labeled "LED" on the RGB Sensor. This pin is used to control the LED (which is the yellowish rectangle on the Sensor in Figure 11) and turn it on or off. If the pin is "pulled to ground" (i.e., it's voltage is set to the same voltage as ground), the LED will be turned off. Otherwise, the LED will remain on. We are going to create a separate cable/connector for this LED pin that we can use to control the LED.

Figure 12 - Use 24 AWG wires to create your cables.

We will need 5 wires to make the two cables. We used red, black, white, and grey for the I2C cable and blue for the LED cable. We used 24 AWG wires (this is the thickness of the wires). Try to keep the cables as short as you can while still allowing the Sensor to be mounted in a good position on your Robot.

The following table shows which colors to use to connect to the pins on the Sensor:

Pin	Color
VIN	Red
GND	Black
SCL	White
SDA	Gray
LED	Blue

Figure 13 - Match the pin to the color.

Next, strip about an eighth inch from both ends of all five wires.

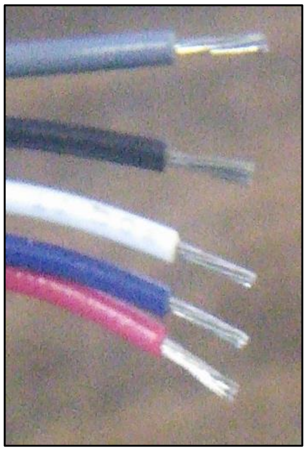


Figure 14 - Strip about 1/8" from both sides of each wire.

We need to crimp metal pieces known as "terminals" onto both ends of each wire.

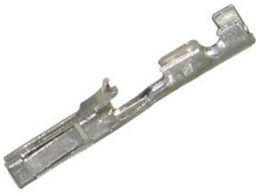


Figure 15 - You need to crimp a terminal onto both ends of each wire.

In Figure 15, the exposed tip of the wire would be slipped into the terminal from the right side of this image. The right side of the terminal in Figure 15 is the part that gets crimped.

Terminals can be found using the following link,

http://www.molex.com/molex/products/datasheet.jsp?part=active/0016020096_CRIMP_TERMINALS.xml

If resources permit, you can use a dedicated crimping tool to do the job. We like to use a tool from a company called Molex to crimp the terminals:

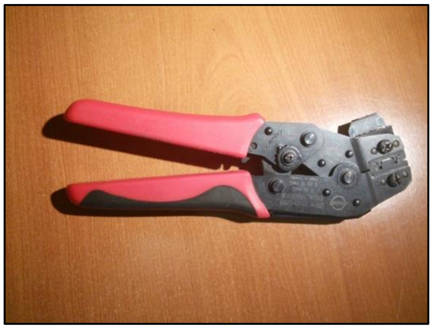


Figure 16 - Molex terminal crimping tool.

Details about the Molex crimping tool are available online:

http://www.molex.com/molex/products/datasheet.jsp?part=active/0640160201_APPLICATION_TOOLIN.xml&channel=Products

If you do not have access to a dedicated crimping tool, you can also use a pair of pliers to crimp the terminals onto the ends of the wire.

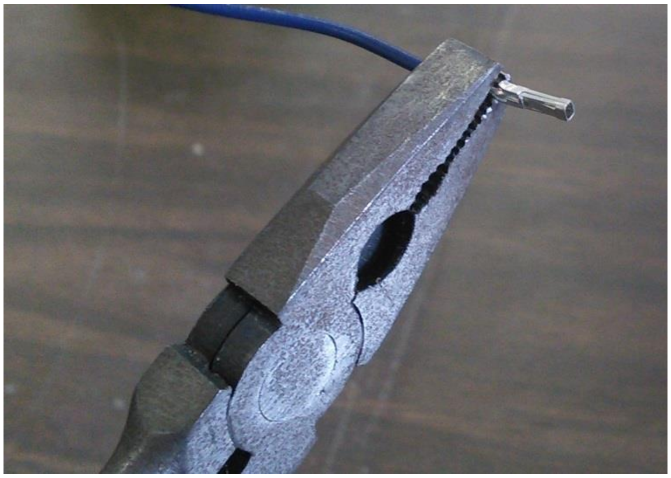


Figure 17 - Hold the terminal with the crimping tool or pliers.

Insert one terminal into the crimp tool or gently hold it with the pliers. Place the end of one of the wires into the terminal and close the crimps. If you are using the pliers you will probably need to reposition the terminal a few times to crimp the terminal properly and to get the terminal to really bite into the exposed end of the wire.

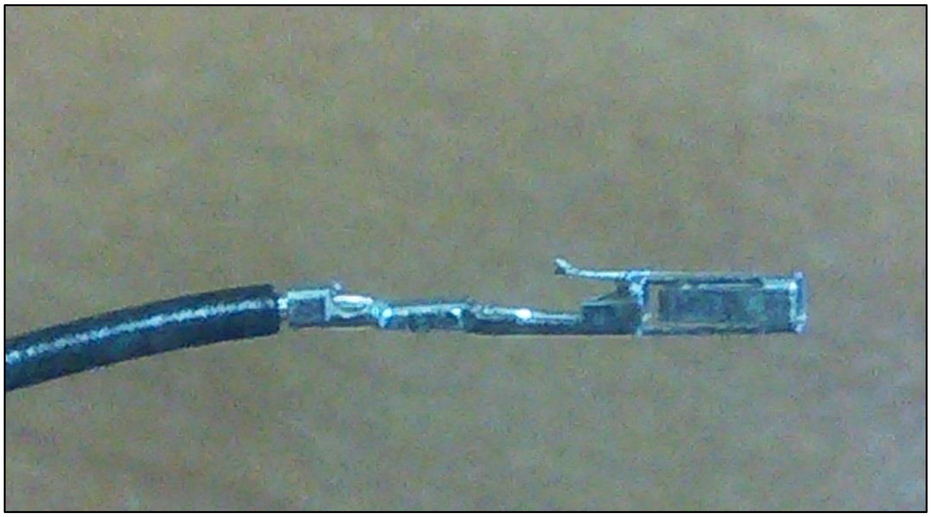


Figure 18 - Crimp a terminal onto one end of the wire.

Repeat this step for both ends of all five wires.

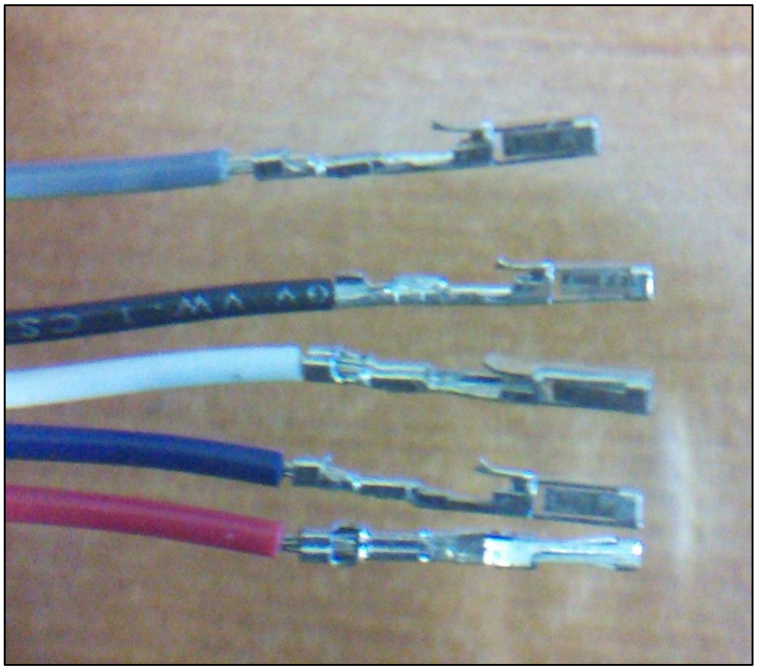


Figure 19 - Repeat the step for all five wires.

After you have crimped terminals onto the ends of each wire, you will need to insert the terminals into plastic housings.

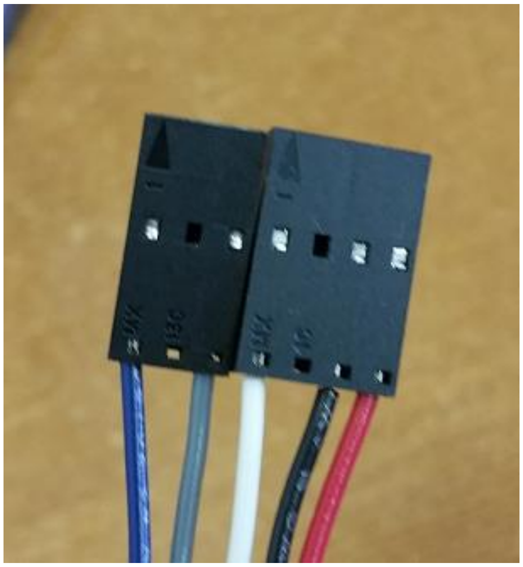


Figure 20 - You will need to slide the terminals into plastic housings

On one side of the wires, it would be ideal if you could use a 7-circuit housing to hold the terminals of all of the cables together. A 7-circuit housing would be used to connect the wires to the 7 pins soldered on the Sensor board. We did not have a 7-circuit housing available, so we used a 3-circuit housing and a 4-circuit housing instead.

The following table lists the housings and a web link to the Molex website for each part.

Item	Web Link
7-circuit	http://www.molex.com/molex/products/datasheet.jsp?part=active/0050579007_CRIMP_HOUSINGS.xml
4-circuit	http://www.molex.com/molex/products/datasheet.jsp?part=active/0050579004_CRIMP_HOUSINGS.xml
3-circuit	http://www.molex.com/molex/products/datasheet.jsp?part=active/0050579003_CRIMP_HOUSINGS.xml
2-circuit	http://www.molex.com/molex/products/datasheet.jsp?part=active/0050579002_CRIMP_HOUSINGS.xml

Table 3 - Terminal housings with the links to their product web pages.

For every Sensor that you make, we recommend that you use the following housings:

- One 7-circuit housing (to use on the side of the cable the plugs into the Sensor)
- One 4-circuit housing (to connect the Sensor to the I2C port on the CDI module)
- One 2-cicuit housing (to connect the LED to either the ground or digital pin of a digital port on the CDI module)

To install a terminal in its housing, first place the terminal housing on a table with the side that has the arrow-shaped symbol on it facing up.

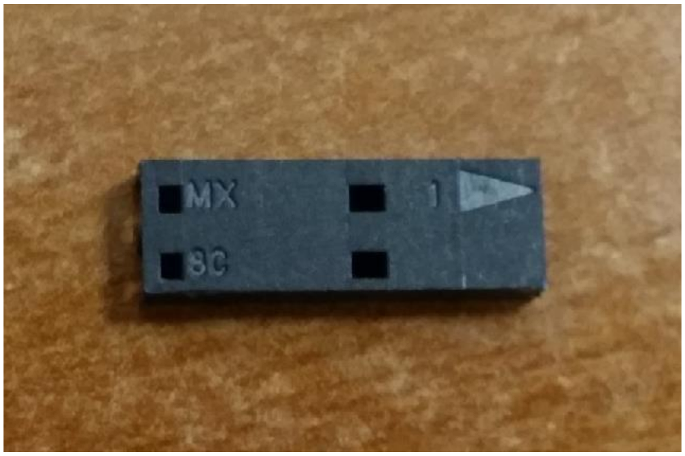
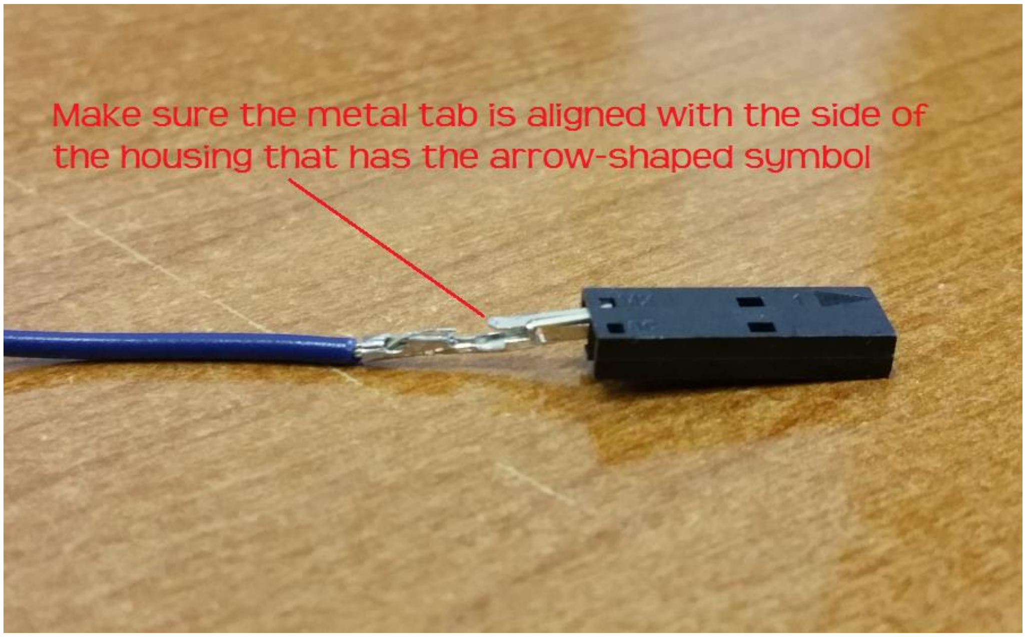


Figure 21 - Place housing so the side with the arrow-shaped symbol is facing up.

Before you slide the terminal into the housing, make sure the part of the terminal that has the metal tab is aligned with the side of the housing that has the arrow symbol.



Make sure the metal tab is aligned with the side of the housing that has the arrow-shaped symbol

Figure 22 - Make sure the metal tab is aligned with side that has arrow-shaped symbol.

Making the I2C Port Housing

Let's start by assembling the housing that will plug into the I2C port on the Core Device Interface Module. If you look on the Modern Robotics website (see http://modernRoboticsinc.com/core-device-interface-module-2) you'll see that there are four pins per I2C port. The leftmost pin provides +5V power to the pin, which should map to the VIN pin on the Sensor (see Figure 11). The next pin from the left is the I2C data pin, which should map to the SDA pin on the Sensor (see Figure 11). The next pin from the left is the I2C clock pin, which should map to the SCL pin on the Sensor (see Figure 11). The rightmost pin is the ground pin, which should map to the GND pin on the Sensor (see Figure 11).

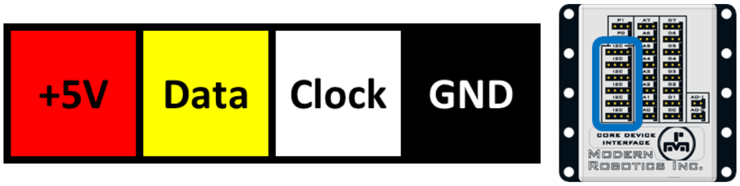


Figure 23 - The pin layout (VIN, SDA, SCL, and GND) for an I2C port on the CDI module.

You want to assemble your 4-pin housing so it will plug into the I2C port as shown in the image below:

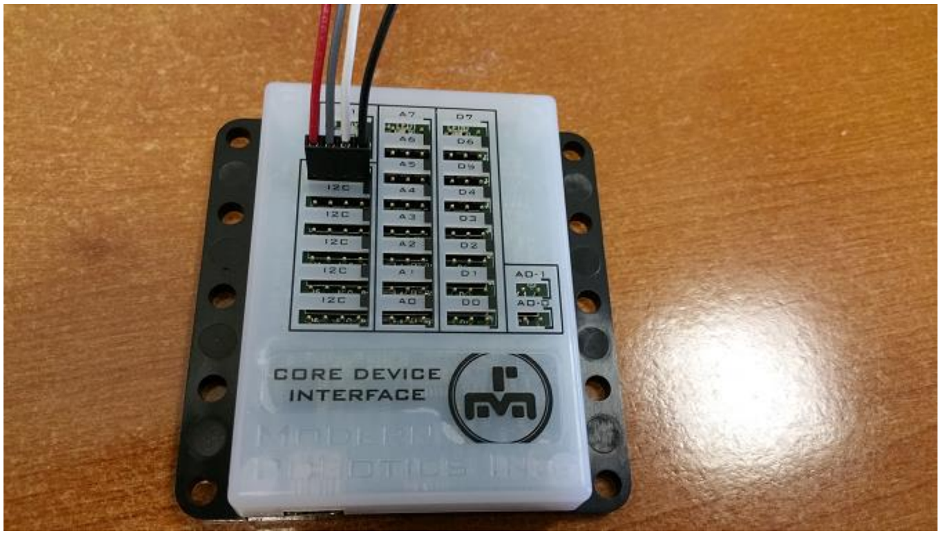


Figure 24 - You'll want to assemble the housing with the VIN, SDA, SCL and GND cables arranged from left to right.

Let's assemble the housing. Start with a 4-circuit housing. If you are looking at the side of the housing with the arrow on it, starting from the right hand side, insert the red wire until you hear it click.

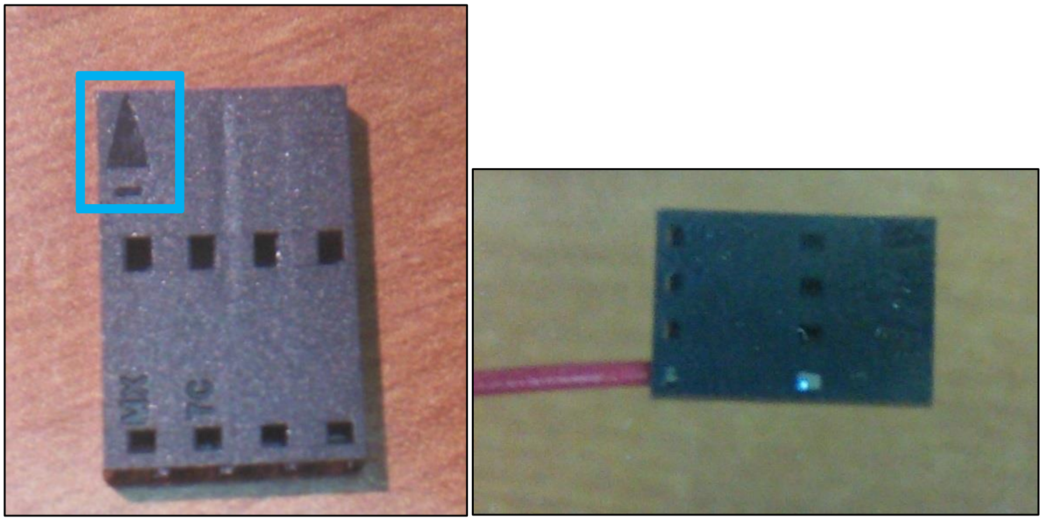


Figure 25 - Start by inserting the red wire into the housing.

After the red wire, continue from the right hand side with the white wire, then the gray wire, and then the black wire. After this is complete, lightly pull on each wire to make sure it is actually in the housing.

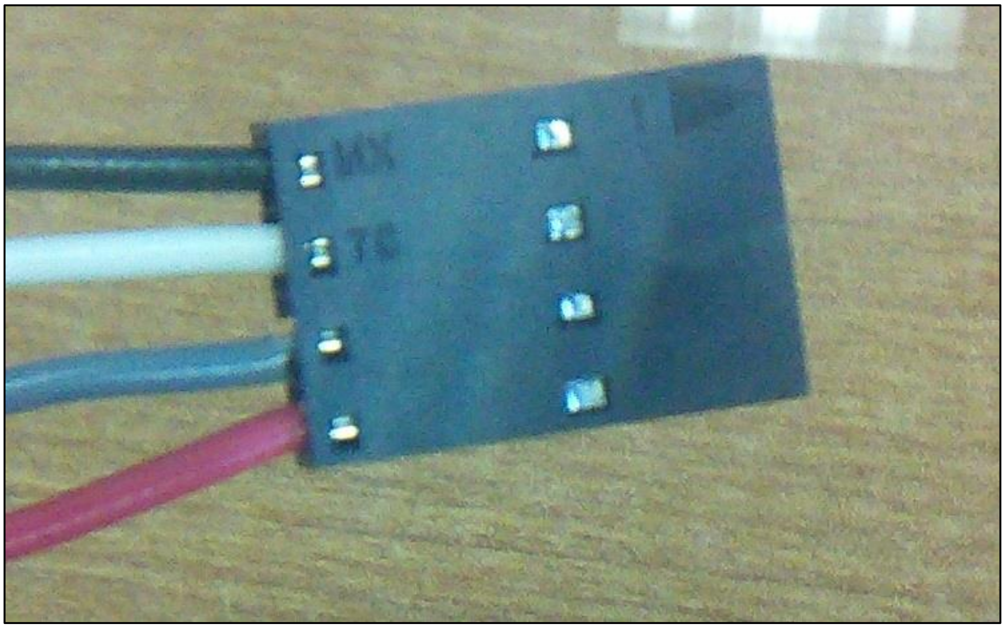


Figure 26 - After inserting the red wire (VIN), do the same for the grey (SDA), white (SCL) and black (GND) wires.

Making the Digital Port Housing

You will need to make a housing that you can use to connect the LED pin to the CDI module. We recommend using a 2 connector housing. With the LED pin connected through the blue wire to the housing, you can use the housing to either connect the LED pin to the signal pin of a digital port (in case you want to use the digital port to turn the LED on or off) or you can switch it around and use the housing to connect the LED pin to the ground pin of a digital port (in case you want to keep the LED off).

The digital port on the Modern Robotics CDI module is arranged as follows,

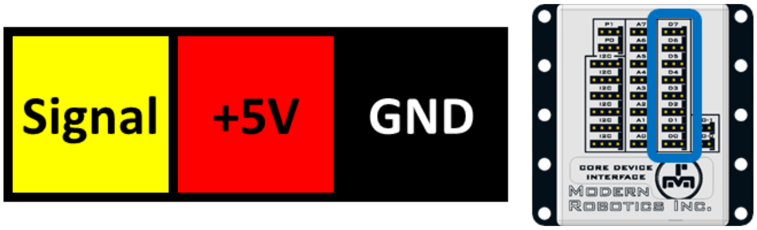


Figure 27 - Layout of digital port on CDI module.

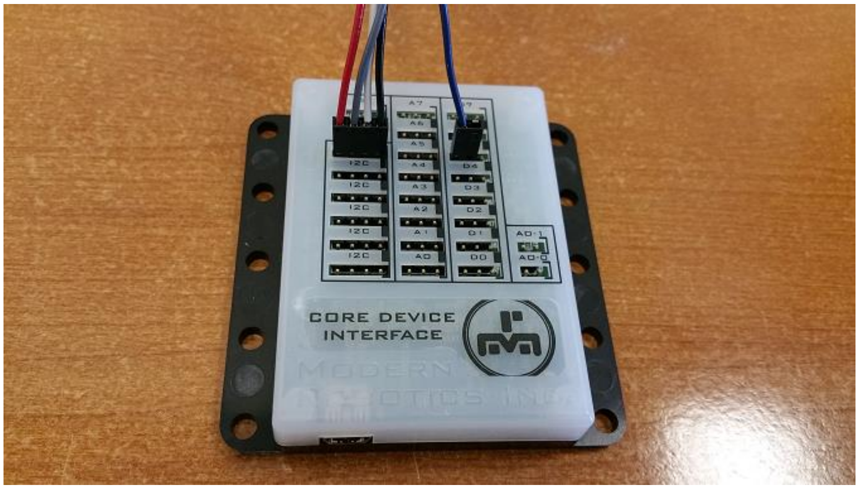


Figure 28 - Connect LED pin to a digital signal pin on the CDI module if you want to control the LED with that port.

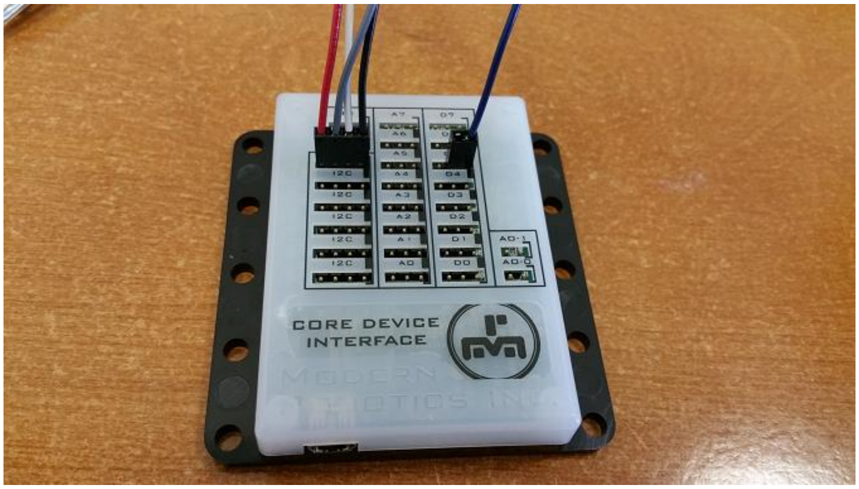


Figure 29 - Connect LED pin to a ground pin on the CDI module if you want to keep the LED turned off.

You will need the 2-circuit housing. Take the blue wire and insert it into right hand port of the housing when the side with the arrow is facing up. Again keep pushing the wire into the housing until you hear a click.

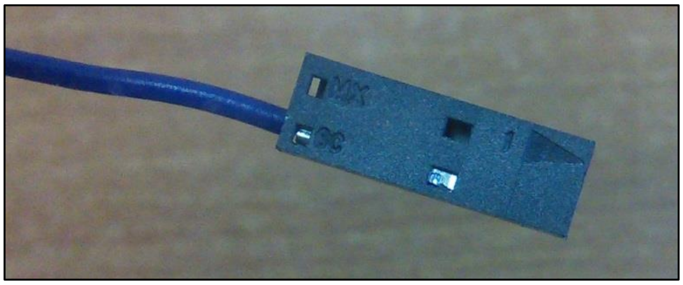


Figure 30 - Insert blue wire till you hear a click.

Making the Sensor Connector Housing

Now you need to assemble a housing that you can use to connect the wires to the pins on the Sensor. Ideally, you would use a 7-circuit housing for this job. However, in our case, we did not have a 7-circuit housing on hand. Instead, we used a 4-circuit housing and a 3-circuit housing to make the connector.

You will want to insert your pins so that blue wire connects to the LED pin, the gray wire connects to the SDA pin, the white wire connects to the SCL pin, the black wire connects to the GND pin, and the red wire connects to the VIN pin. Notice the order of the wires and the gaps (to skip the INT and 3V3 pins) in the image shown in Figure 31.

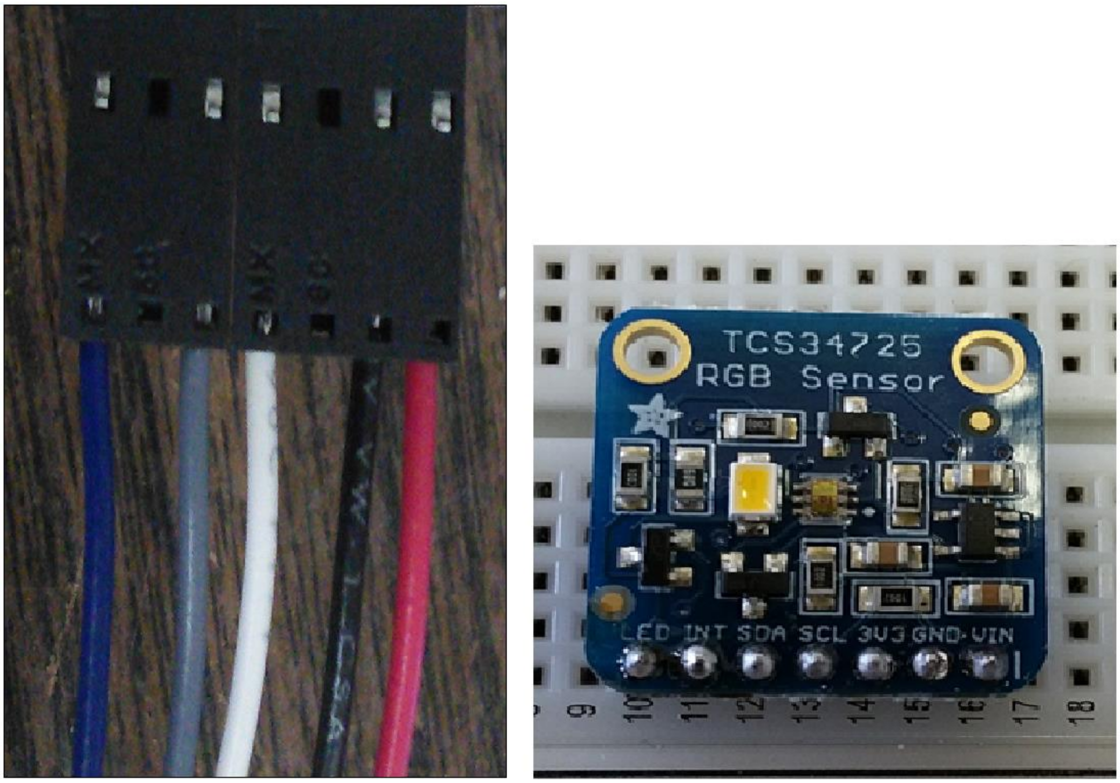


Figure 31 - The blue, gray, white, black and red wires should match the LED, SDA, SCL, GND and VIN pins.

Again, start with the red wire in the right most port when the housing has the side with the arrow facing up.

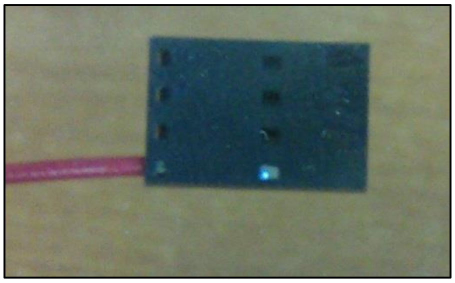


Figure 32 - Start with the red (VIN) wire.

This time, the order of the wires is a little different then last time. The order for this housing will be, starting next to the red wire and moving away from it, black, then skip a port, then white, grey, skip another port, blue. It should look like this:

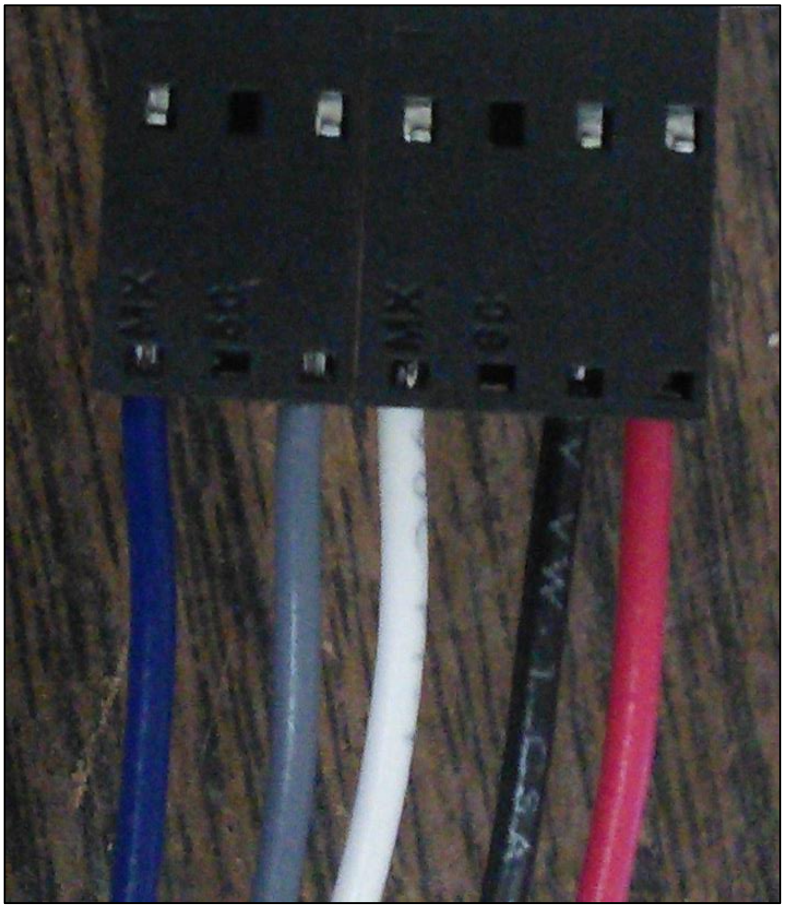


Figure 33 - Your 7-circuit housing should look like this.

Make sure to pull on all the wires a little just to make sure they are really set in the housing.

3D Printed Sensor Housing

3D Model

The RGB Sensor from AdaFruit does not include a protective housing. For Competition use, the Sensor should be placed in some kind of housing that will make it easier to mount the Sensor to the Robot and will offer some protection for the Sensor.

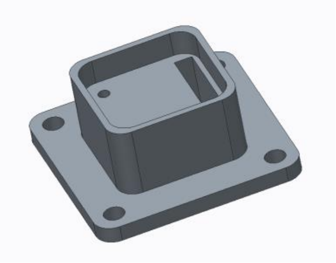


Figure 34 - 3D model of a Sensor housing with a rectangular slot to for the pins.

FIRST has developed a simple model that can be used to 3D print a housing that uses two flat head wood screws (#2 x ½") to secure the Sensor to the housing. The housing has a slot that allows the 7-circuit terminal housing to be connected to the pins on the Sensor.

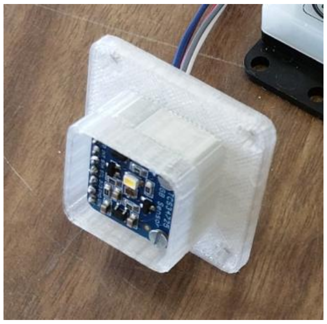


Figure 35 - A simple open-faced housing that secures the Sensor and can be mounted to a Robot.

An open-faced design was used to help make the readings more accurate when the Sensor's LED is turned on. Originally we had a top cover that offered better protection of the Sensor, but gave less accurate readings in active mode (when the LED was turned on) due to reflects from the top cover. The open-faced design provides some protection to the RGB Sensor, but it also makes it easy to determine the color of a surface accurately when the LED is enabled.

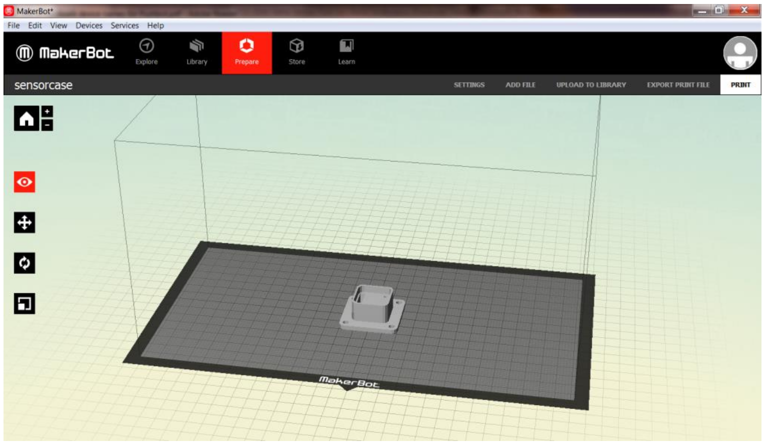


Figure 36 - The model can be used to 3D print a Sensor-housing.

The FTC Team Resources web page has a link that you can follow to download the 3D model files. Visit the following website,

http://www.usfirst.org/Roboticsprograms/ftc/Team-resources

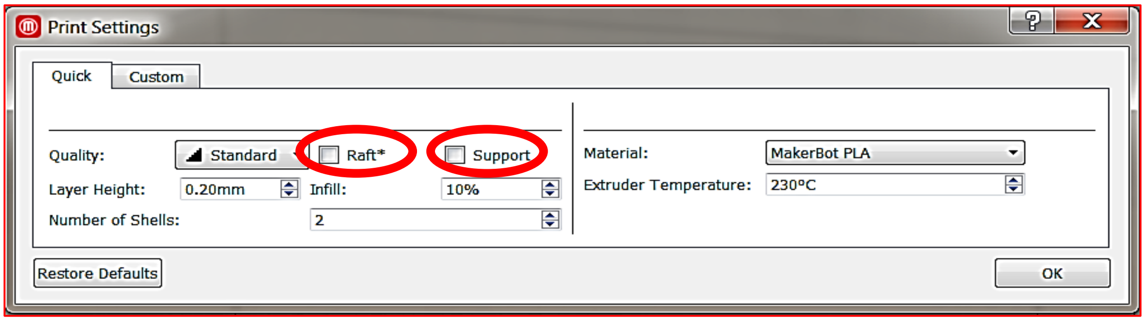
and scroll down the section labeled **Sensor Resources**. Underneath this category you will see a link to a .ZIP file that contains the 3D model files for the RGB Sensor housing.

Note that the model is provided in 3 different file formats:

- .PRT – The file is stored as a PTC Creo part. You can open this file directly using PTC Creo.
- .STEP – If you are using a different CAD program, you can import the model with this .STEP file.
- .STL – If you are using a MakerBot 3D printer you can open this file with the MakerWare software.

If you have access to a 3D printer, you can download the 3D model for the housing and print a housing for your own use. You should also consider printing some spare housings and make them available to members of the community who do not have access to a 3D printer.

If you use the FIRST-provided 3D model, when you print the case, we recommend not using any rafting or support for this specific print. You can turn them off in the Print Settings window of the MakerWare software by unchecking the Raft and Support boxes (see Figure 37).


Print Settings

Quick | Custom

Quality: Standard □ Raft* □ Support

Layer Height: 0.20mm Infill: 10%

Number of Shells: 2

Material: MakerBot PLA

Extruder Temperature: 230ºC

Restore Defaults OK

Figure 37 - Turn off the Raft and Support options for the FIRST-provided 3D housing.

Assembling the Housing

After the print is complete, insert the Sensor into the case with the pins going through the rectangular slot. Next use two number 2 screws (1/2" in length) to secure the Sensor to the case. Screw them through the two holes in the Sensor's printed circuit board.

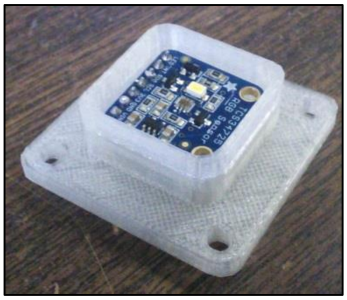


Figure 38 - Place the Sensor in the housing with the pins sticking out (not visible in this image) through the rectangular slot.

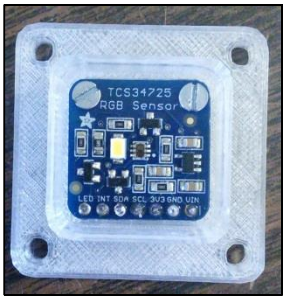


Figure 39 - Secure the Sensor using two #2 wood screws (1/2" in length).

Connecting Your Sensor to the CDI

Now it is time to connect the Sensor to the CDI. You will use the cables and housings that you made earlier to connect the two devices together. Note that on the RGB Sensor side of the cable, the color of the wires should map to the pins in the manner shown in Figure 13 on page 16 of this document.

Take the 7-circuit housing and connect it so that the red wire connects to the pin labeled "VIN" on the Sensor. You can refer to the picture below if needed:

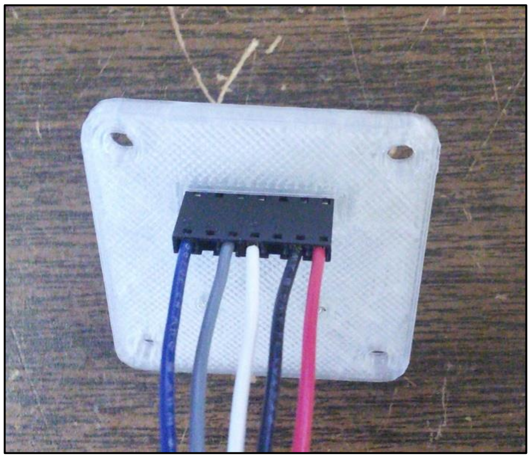


Figure 40 - Connect 7-circuit housing so that the red wire attaches to the VIN pin on the board.

The other end of the cable has two connectors. In effect, the AdaFruit RGB Color Sensor looks like two devices to your Core Device Interface module. The Sensor looks like an I2C device, and it has a 4-circuit housing that can be connected to an I2C port. The senor also looks like a digital device (its LED), and it has a 2-circuit housing (with only one wire connected) that can be connected to the signal or ground pin of a digital port.

Connect the 4-circuit housing to an I2C port as show below in Figure 41. Then connect the 2-circuit housing to a digital port with the wire connected to the digital signal pin (on the left side) if you want to control the LED with a digital port (also see Figure 41).

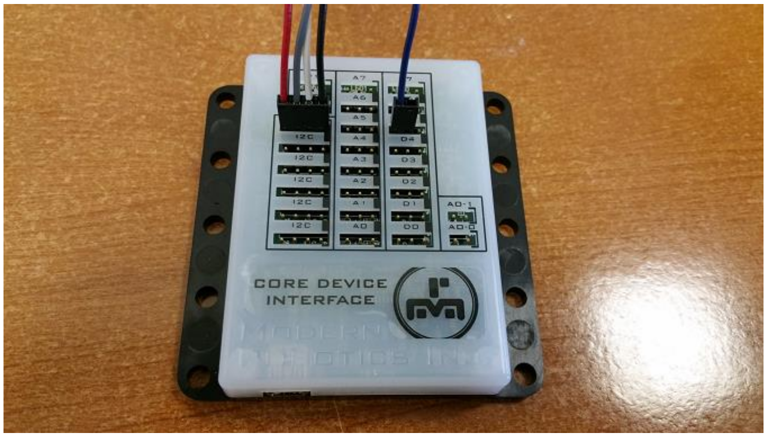


Figure 41 - Plug the 4-circuit and 2-circuit (1-wire) housings into an I2C port and digital port respectively.

Congratulations! You've assembled your AdaFruit RGB Sensor and are ready to begin programming!

Example Op Mode

AdaFruitRGBExample Linear Op Mode

IMPORTANT NOTE: At the time that this document was first released (September 12, 2015) the new ColorSensor class that is required to control the AdaFruit RGB Sensor was not yet available in the publicly released versions of the FTC SDK or the FTC App Inventor appliance file. The ColorSensor class will be made available in the next release after September 12, 2015. This section examines a linear op mode that uses the ColorSensor class to display color info from the AdaFruit Sensor on the driver station. The user can control the LED by pressing or releasing the X button on either gamepad. This example op mode (AdaFruitRGBExample) is included with the version of the SDK that has support for the ColorSensor class.

Let's take a look at the sample linear op mode called AdaFruitRGBExample. This op mode demonstrates how to use the ColorSensor class to retrieve color information from the AdaFruit Sensor. The op mode uses the telemetry mechanism to send the red, green, blue and clear (alpha) color information back to the driver station. The op mode uses an Android method that is associated with the Color class to convert the RGB values into HSV (hue, saturation, and value – HSV is a way to represent "points in an RGB color model[3]). The hue component is sent back to the driver station using the telemetry mechanism and the HSV numbers are used to change the background color of the Robot Controller app to match the detected color from the RGB Sensor.

The op mode also checks for button transitions (button being pressed, button being released) to control the LED on the RGB Sensor. If the op mode detects that the X button on either gamepad has been pressed, it will enable the LED by pulling a digital port HIGH. If the op mode detects that the X button on either gamepad has been released, it will disable the LED by pulling a digital port LOW.

A copy of the op mode is included in this section of the document. You can review the op mode and read the comments to learn how to communicate with and control the AdaFruit color Sensor using the ColorSensor class and by controlling the state of a digital port on the Core Device Interface Module.

Source Code

```
package com.qualcomm.ftcRobotcontroller.opmodes;

import android.app.Activity;
import android.graphics.Color;
import android.view.View;

import com.qualcomm.ftccommon.DbgLog;
import com.qualcomm.ftcRobotcontroller.R;
import com.qualcomm.Robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.Robotcore.hardware.ColorSensor;
import com.qualcomm.Robotcore.hardware.DeviceInterfaceModule;
import com.qualcomm.Robotcore.hardware.DigitalChannelController;

/*
 *
 * This is an example LinearOpMode that shows how to use
 * the AdaFruit RGB Sensor.  It assumes that the I2C
 * cable for the Sensor is connected to an I2C port on the
```

[3] From Wikipedia (https://en.wikipedia.org/wiki/HSL_and_HSV) downloaded on 9/9/15.

```
 * Core Device Interface Module.
 *
 * It also assuems that the LED pin of the Sensor is connected
 * to the digital signal pin of a digital port on the
 * Core Device Interface Module.
 *
 * You can use the digital port to turn the Sensor's onboard
 * LED on or off.
 *
 * The op mode assumes that the Core Device Interface Module
 * is configured with a name of "dim" and that the AdaFruit color Sensor
 * is configured as an I2C device with a name of "color".
 *
 * It also assumes that the LED pin of the RGB Sensor
 * is connected to the signal pin of digital port #5 (zero indexed)
 * of the Core Device Interface Module.
 *
 * You can use the X button on either gamepad to turn the LED on and off.
 *
 */
public class AdaFruitRGBExample extends LinearOpMode {

  ColorSensor SensorRGB;
  DeviceInterfaceModule cdim;

  // we assume that the LED pin of the RGB Sensor is connected to
  // digital port 5 (zero indexed).
  static final int LED_CHANNEL = 5;

  @Override
  public void runOpMode() throws InterruptedException {

    // write some device information (connection info, name and type)
    // to the log file.
    hardwareMap.logDevices();

    // get a reference to our DeviceInterfaceModule object.
    cdim = hardwareMap.deviceInterfaceModule.get("dim");

    // set the digital channel to output mode.
    // remember, the AdaFruit Sensor is actually two devices.
    // It's an I2C Sensor and it's also an LED that can be turned on or off.
    cdim.setDigitalChannelMode(LED_CHANNEL, DigitalChannelController.Mode.OUTPUT);

    // get a reference to our ColorSensor object.
    SensorRGB = hardwareMap.colorSensor.get("color");

    // bEnabled represents the state of the LED.
    boolean bEnabled = true;

    // turn the LED on in the beginning, just so user will know that the Sensor is
active.
    cdim.setDigitalChannelState(LED_CHANNEL, bEnabled);

    // wait one cycle.
    waitOneFullHardwareCycle();

    // wait for the start button to be pressed.
    waitForStart();

    // hsvValues is an array that will hold the hue, saturation, and value information.
    float hsvValues[] = {0F,0F,0F};

    // values is a reference to the hsvValues array.
    final float values[] = hsvValues;

    // get a reference to the RelativeLayout so we can change the background
    // color of the Robot Controller app to match the hue detected by the RGB Sensor.
    final View relativeLayout = ((Activity)
hardwareMap.appContext).findViewById(R.id.RelativeLayout);

    // bPrevState and bCurrState represent the previous and current state of the button.
    boolean bPrevState = false;
    boolean bCurrState = false;

    // while the op mode is active, loop and read the RGB data.
    // Note we use opModeIsActive() as our loop condition because it is an interruptible
method.
    while (opModeIsActive()) {
      // check the status of the x button on either gamepad.
      bCurrState = gamepad1.x || gamepad2.x;

      // check for button state transitions.
      if (bCurrState == true && bCurrState != bPrevState)  {
        // button is transitioning to a pressed state.

        // print a debug statement.
        DbgLog.msg("MY_DEBUG - x button was pressed!");

        // update previous state variable.
        bPrevState = bCurrState;

        // on button press, enable the LED.
        bEnabled = true;

        // turn on the LED.
        cdim.setDigitalChannelState(LED_CHANNEL, bEnabled);

      } else if (bCurrState == false && bCurrState != bPrevState)  {
        // button is transitioning to a released state.

        // print a debug statement.
        DbgLog.msg("MY_DEBUG - x button was released!");

        // update previous state variable.
        bPrevState = bCurrState;

        // on button press, enable the LED.
        bEnabled = false;

        // turn off the LED.
        cdim.setDigitalChannelState(LED_CHANNEL, bEnabled);
      }

      // convert the RGB values to HSV values.
      Color.RGBToHSV((SensorRGB.red() * 255) / 800, (SensorRGB.green() * 255) / 800,
(SensorRGB.blue() * 255) / 800, hsvValues);

      // send the info back to driver station using telemetry function.
      telemetry.addData("Clear", SensorRGB.alpha());
      telemetry.addData("Red  ", SensorRGB.red());
      telemetry.addData("Green", SensorRGB.green());
      telemetry.addData("Blue ", SensorRGB.blue());
      telemetry.addData("Hue", hsvValues[0]);

      // change the background color to match the color detected by the RGB Sensor.
      // pass a reference to the hue, saturation, and value array as an argument
      // to the HSVToColor method.
      relativeLayout.post(new Runnable() {
        public void run() {
          relativeLayout.setBackgroundColor(Color.HSVToColor(0xff, values));
        }
      });

      // wait a hardware cycle before iterating.
      waitOneFullHardwareCycle();
    }
  }
}
```

2015-2016 *FIRST*® Tech Challenge AdaFruit RGB Sensor Assembly and User's Guide

Appendices

Appendix A – 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

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. Thank you!