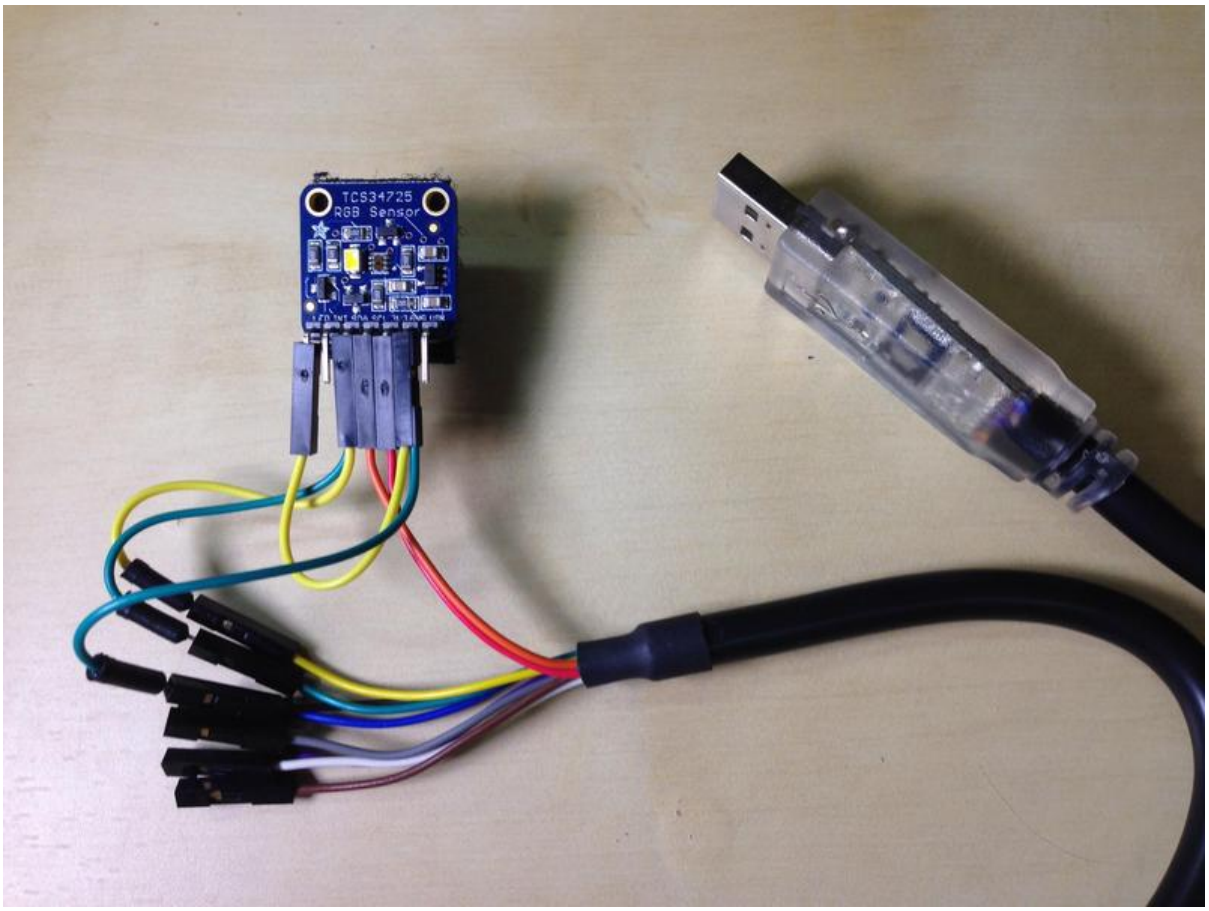
Automatic Monitor Color Temperature Adjustment

Created by Tony DiCola



https://learn.adafruit.com/automatic-monitor-color-temperature-adjustment

Last updated on 2022-12-01 02:09:57 PM EST

Table of Contents

Overview

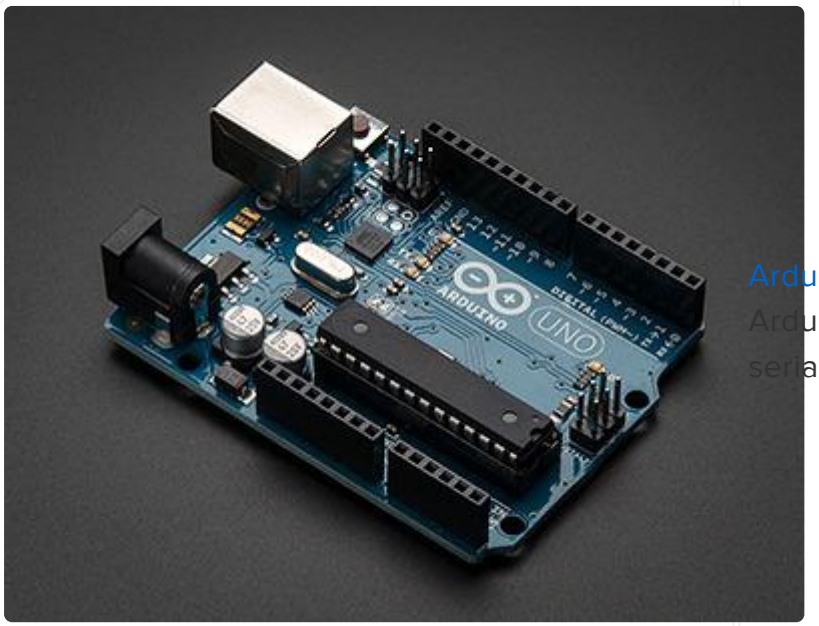
Staring at a computer monitor all day is quite stressful. Moving your gaze from the cold, white light of a monitor to warmer indoor light is especially jarring to your senses. Programs like f.lux () and Redshift () were created to ease this strain by adjusting the color temperature () of your monitor. However these programs don't measure color temperature of light in the environment and instead guess the temperature based on location, time of day, and sunrise/sunset time.

This project will show you how to build hardware that measures the temperature of ambient light and automatically adjusts your monitor color to match. With just a simple RGB color sensor and an Arduino or FT232H-based cable, your computer can easily sense and react to light in its environment.

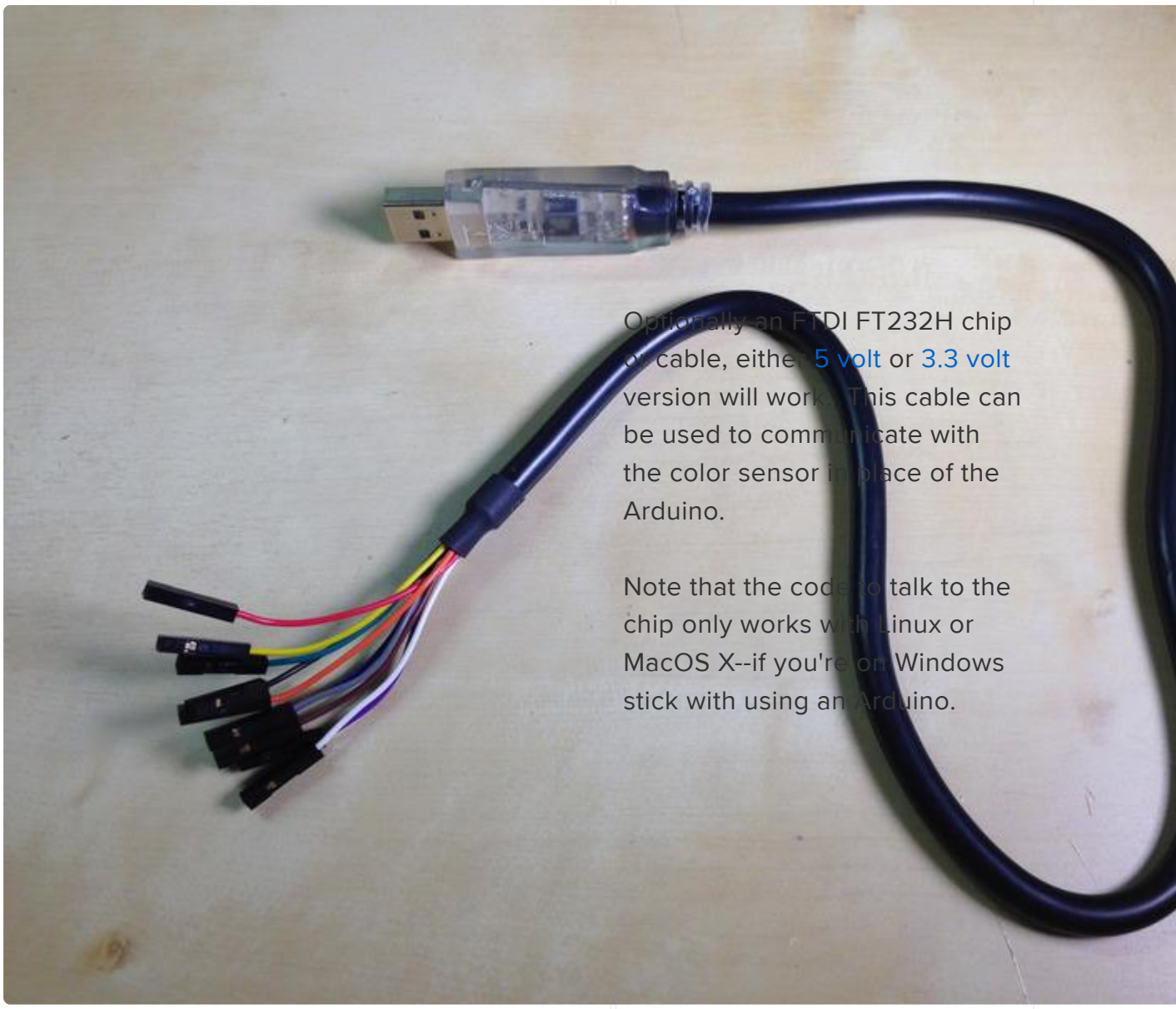
Before building this project, it will be helpful to familiarize yourself with the color sensor guide ().

Hardware

You'll need the following parts to build this project:

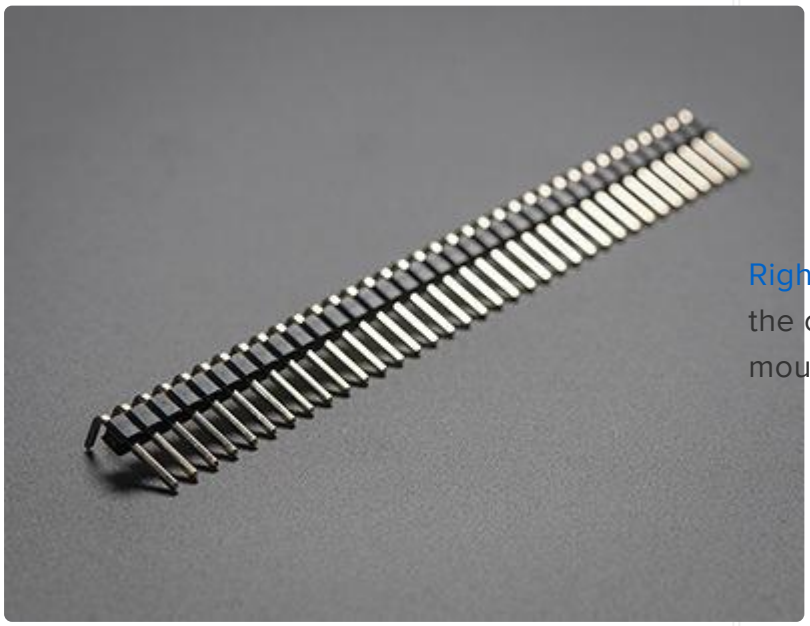


Arduino Uno, Micro, or other Arduino which supports USB serial and I2C communication.



Optionally an FTDI FT232H chip or cable, either 5 volt or 3.3 volt version will work. This cable can be used to communicate with the color sensor in place of the Arduino.

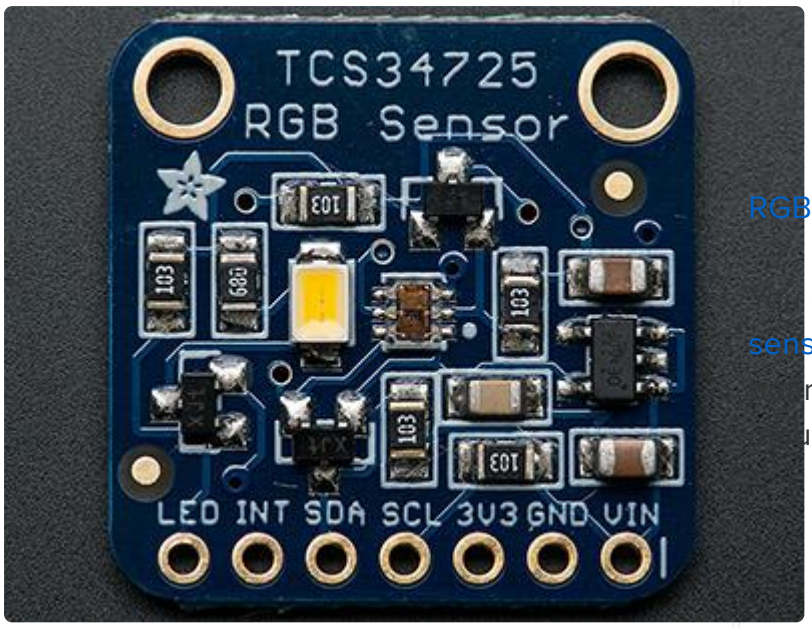
Note that the code to talk to the chip only works with Linux or MacOS X--if you're on Windows stick with using an Arduino.



Right angle header to solder on the color sensor for flush mounting to a monitor.

Female to male extension cables for connecting to the color sensor.

Velcro or other means to attach the sensor to the front of your monitor.



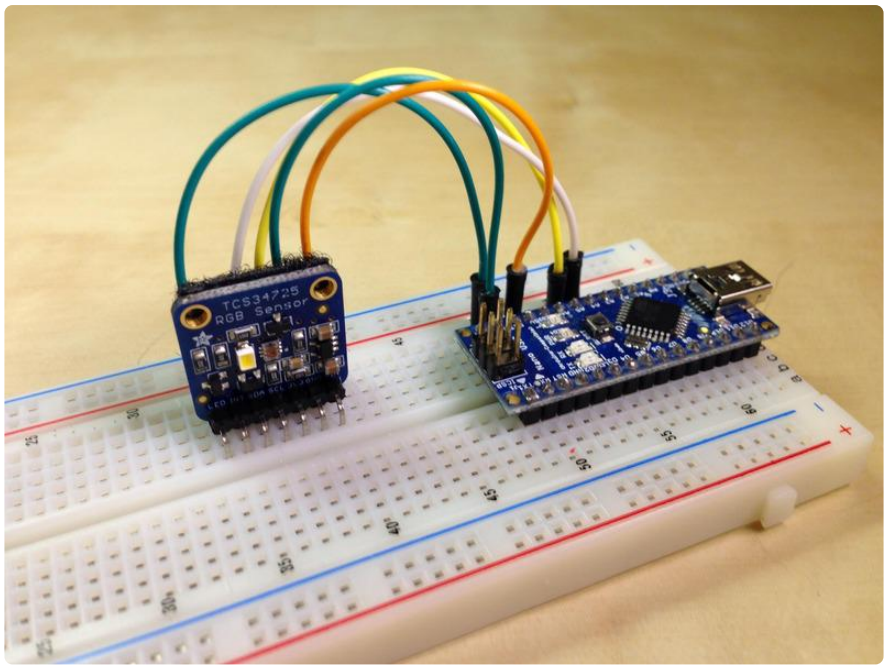
RGB Color Sensor

can use the Flora RGB color sensor too, but the breakout rd sensor will be easier to unt to a right angle header.

Assembly

The assembly of this project is very simple. Cut the right angle header to size and solder it to the color sensor. Connect the color sensor to the Arduino as follows:

- Color sensor 5V to Arduino 5V
- Color sensor GND to Arduino ground
- Color sensor SDA to Arduino SDA (analog 4 on older Arduinos, check your your board pinout () to be sure)
- Color sensor SCL to Arduino SCL (analog 5 on older Arduinos, see above to check others)
- Color sensor LED to Arduino ground to disable the built-in LED.

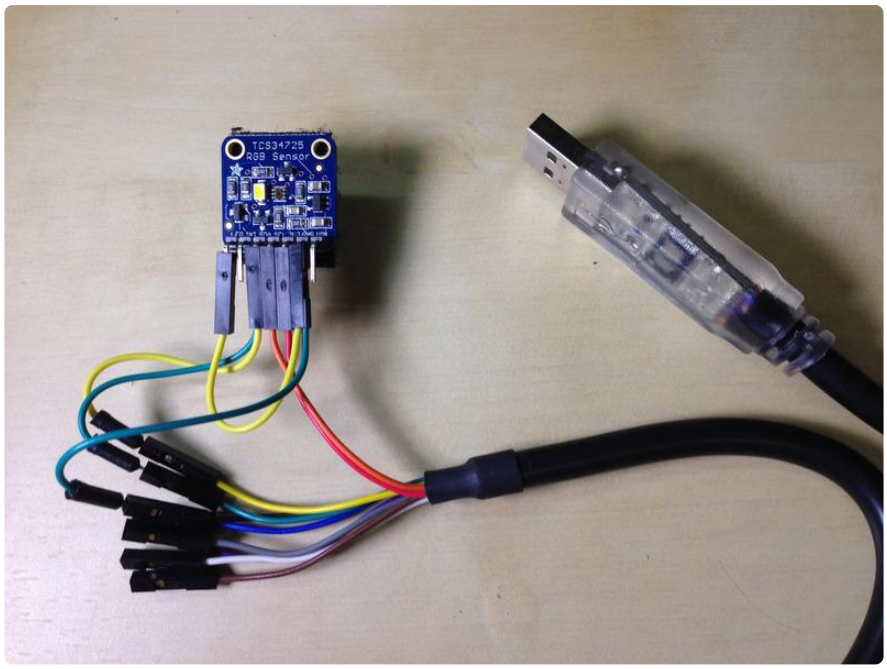


Above you can see a picture of the hardware connected to an Arduino Nano.

If you're using the FT232H cable instead of the Arduino, connect it to the sensor as follows:

- If using the 5V cable: color sensor 5V to cable power (red)
- If using the 3.3V cable: color sensor 3.3V to cable power (red)
- Color sensor GND to cable ground wire (black)
- Color sensor SDA to both the cable green and yellow wires to duplex the sent and received I2C data.
- Color sensor SCL to cable clock wire (orange)
- Color sensor LED to cable ground wire (black)

I found it was easiest to cut a few female to male extension cables up and solder them into 2-into-1 Y adapters to connect the SDA -> green and yellow, and GND & LED -> ground connections.



Above you can see a picture of the hardware connected to a 3.3 volt version of the FT232H cable.

If you have questions about using the FT232H cable, check out its data sheet here () or this application note () on using the cable to talk I2C.

Software

Dependencies

You'll need to download and install the following dependencies for this project's software:

Adafruit TCS34725 Color Sensor Arduino Library ()
If you aren't sure how to install an Arduino library, check out this tutorial ().

Python
The software for this guide is written in python () and should work with either python 2.7 or 3+ (although it was primarily tested against version 2.7).

Once you have python installed, follow these steps depending on your platform:

Mac OS X

Install NumPy () with one of the binary packages here ().

Install the PIP () python package manager by executing the following commands in a terminal:

```
wget https://raw.github.com/pypa/pip/master/contrib/get-pip.py ()
python get-pip.py
```

Finally, install the colormath () and pyserial () library by executing in a terminal:

```
pip install colormath==1.0.9 pyserial
```

If you plan to use the FT232H cable, download and install the X code command line tools by following the steps in the answer here (). You will also want to install the homebrew () package manager too. Once those are installed you can install the required libftdi () and libmpsse () dependencies by executing in a terminal:

```
brew install libftdi
brew install swig
wget https://libmpsse.googlecode.com/files/libmpsse-1.3.tar.gz ()
tar xvfz libmpsse-1.3.tar.gz
cd libmpsse-1.3
./configure
make
sudo make install
```

Linux

These steps are written for Ubuntu, but should be straightforward to adapt to other distributions. In a terminal execute the following commands:

```
sudo apt-get install python-numpy python-pip
pip install colormath pyserial
```

If you plan to use the FT232H cable, execute the following commands to install the required software:

```
sudo apt-get install python-dev libftdi-dev swig
wget https://libmpsse.googlecode.com/files/libmpsse-1.3.tar.gz ()
tar xvfz libmpsse-1.3.tar.gz
cd libmpsse-1.3
./configure
make
sudo make install
```

Windows

Install NumPy () from one of the binary installers here ().

Install PIP by downloading the get-pip.py () file, opening a command window, navigating to the directory with the file, and executing:

```
python get-pip.py
```

Note you might need to add python to your path to be able to run scripts from the command line. This blog post () has a simple set of instructions to add python to your path.

Now execute the following command to install the required colormath and pyserial library:

```
pip install colormath pyserial
```

At this point all the dependencies for Windows are installed. Unfortunately you won't be able to use the FT232H cable with Windows because the libmpsse library doesn't support it. Stick to using an Arduino to communicate with the color sensor on Windows.

Also note that the software for this project has only been tested on a virtual machine running Windows. Everything should work on a real machine, but if you have issues send them to the github repository () for the project.

Download Project Software

Download the software for this project from its github repository () by clicking this button:

Download Software

Unzip the archive and you will find an Arduino sketch, AutoColorTemp_Sketch, and a collection of python scripts.

If you're using an Arduino for this project, load the sketch in Arduino and upload it to your hardware. If you open the serial monitor at 115200 baud you should be able to send a question mark character ? and see the current RGB color (in floating point, with values from 0-1.0) printed.

Software Usage

To run the software for this project, make sure you've assembled the hardware and connected it to your computer. Open a terminal and navigate to the directory with the downloaded python scripts and execute the following command:

```
python run.py --help
```

You should see a display of all the command line parameters for the program. Take note that either the --arduino or --ftdi parameters are required to run the program.

For example to run the program with Arduino hardware connected to /dev/ttyUSB0, execute:

```
python run.py --arduino /dev/ttyUSB0
```

Or to run with an Arduino connected to COM4 on Windows, execute:

```
python run.py --arduino COM4
```

To use the FTDI cable you need to specify the --ftdi option instead of the --arduino option. No port or other parameter is required when using the --ftdi option.

However, before the software is run with the --ftdi option you must disable the built in kernel driver for FTDI devices! Unfortunately the driver built in to recent Linux and Mac OS X kernels is not compatible with the libftdi library and must be temporarily disabled. The provided run_ftdi_linux.sh and run_ftdi_macos.sh shell scripts can be run to automatically disable and re-enable the drivers when running the software. These shells scripts need to be run as root with the sudo command.

For example to run with FTDI hardware on a Mac, execute:

```
sudo ./run_ftdi_macos.sh --ftdi
```

Or on Linux execute:

```
sudo ./run_ftdi_linux.sh --ftdi
```

Be aware when the program is running the FTDI drivers will be disabled so you will not be able to program an Arduino or communicate with other FTDI-based devices!

Once the software is running it will query the color sensor every 10 seconds, compute the temperature of the measured color, and adjust the gamma of the primary display to match the measured color temperature. Make sure to disable or close programs like f.lux before running the software as they might interfere with the gamma

adjustment!

Try shining different colored lights on the sensor to trick it into seeing extreme temperatures (orange, white, or light blue LEDs work best--other colors will be too extreme).

You can press Ctrl-C at any time to quit the application.

Future Work

This project is a great example of using an Arduino or FT232H-based cable to send sensor data to your computer. Using an RGB color sensor you can make your computer aware of the environment and change monitor color temperature based on ambient lighting. Some interesting ways you might extend this project include:

- Flip the sensor around to read color displayed by your monitor and build a calibration tool to adjust colors for perfect display accuracy.
- Use the color sensor to play different music depending on the color of light in the room. Play exciting music to boost productivity in the morning and afternoon, and more relaxed music in the evening and night.
- Build a color matching game for kids where the computer asks to see items of a specific color.
- Extend the Adalight project () to measure ambient light and adjust itself to match the color.

What can you think of to extend the project?