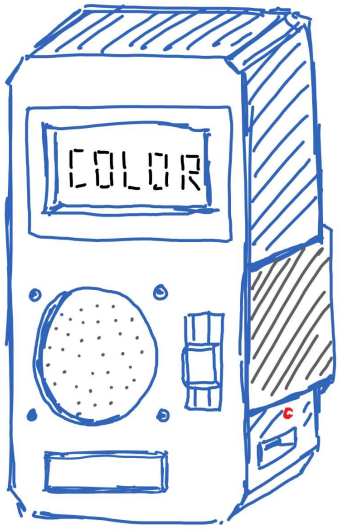
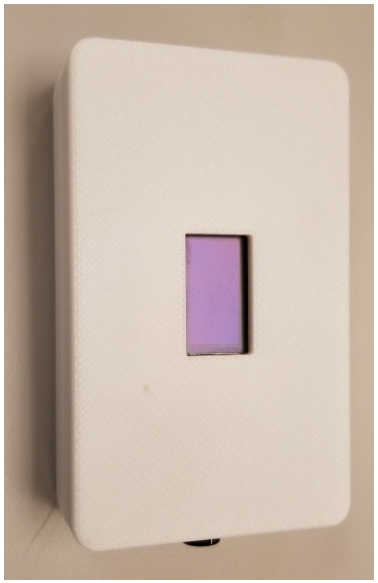
ECE 411 Practicum Project

COLOR DETECTOR





Team 14

James Witcher **Bruce Griffin**

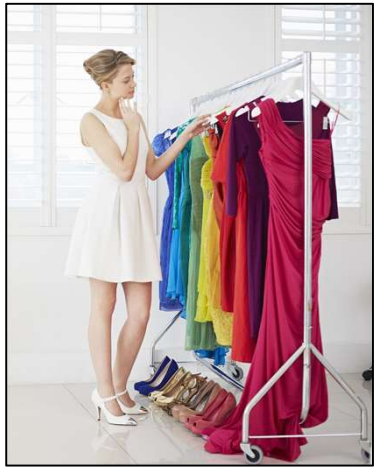
Jeffrey Udall **Matt Fleetwood**

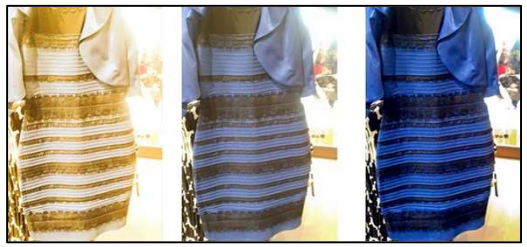
Problem / Need

Finding out the color of something is a real challenge for blind and colorblind people.

Motivation

For a blind or colorblind person something as easy as picking out a matching shirt, shoes, or tie can be an impossible task.





Sometimes color is difficult for everyone!

A relatively cheap and useful way to scan and display colors of objects would make these tasks simple.

Objective

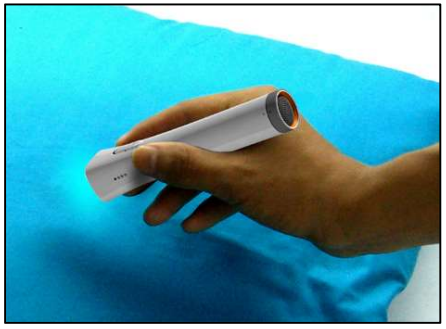
The goal of the project is to develop a working prototype of a color detector which can be used by colorblind people to determine colors of objects like clothes, paints, signs, etc.

This can also be extended to help blind people and robots obtain color information as well.

Existing Alternatives

There are several other devices and algorithms that attempt to perform the same desired function as our design out there:

- **Color Sensor Prototype Using Artificial Neural Networks** - http://www.moretticb.com/blog/color-sensor-prototype-using-neural-networks
- **See Color with Sound Device** - www.yankodesign.com/2008/05/08/see-color-with-sound/
- **Nix Pro Color Sensor** – https://nixsensor.com/



Color With Sound

Nix Color Sensor

Requirements

Our device must meet the following design requirements:

- Use an RGB sensor to get color information about an object
- Deliver the color information on a display
- Include a button to scan/read color
- Work under it's own power
- Have an on/off power switch

This is addition to our class requirements that our device use at least one sensor and at least one actuator.

An acceptable goal would be a working prototype in an enclosure that can detect at least three colors: red, green, and blue.

Requirements

Note : Originally we intended to have an audio sample indicate color as well, but a solution for integrating the audio portion and display subsystems became too difficult to solve in time to complete the prototype before finalizing the schematics.

Our Approach

We follow the guidelines outlined in ECE 411 in order to follow our schedule to design, build, and test our device:

- Use **GitHub** for our subversion control
- Use **Google Docs** to edit and collaborate on documentation
- Use **MS Project** to schedule responsibilities
- Design functional aspects of our device with **modularity** in mind
- Build and test sub-units (e.g. the RGBC sensor + Arduino) independently before integrating parts (e.g. the RBGC sensor + OLED)

Tools

- **Slack** - documentation management and communication
- **GitHub** - documentation management and SVN
- **Google Docs / Slides / Sheets / Drive** - documentation management
- **MS Project / Word** - documentation management and scheduling

Design – Hardware

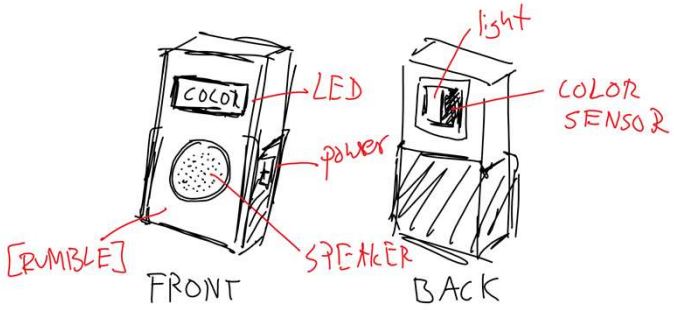
Our design includes both the hardware and software

Hardware pros and cons for our selected parts

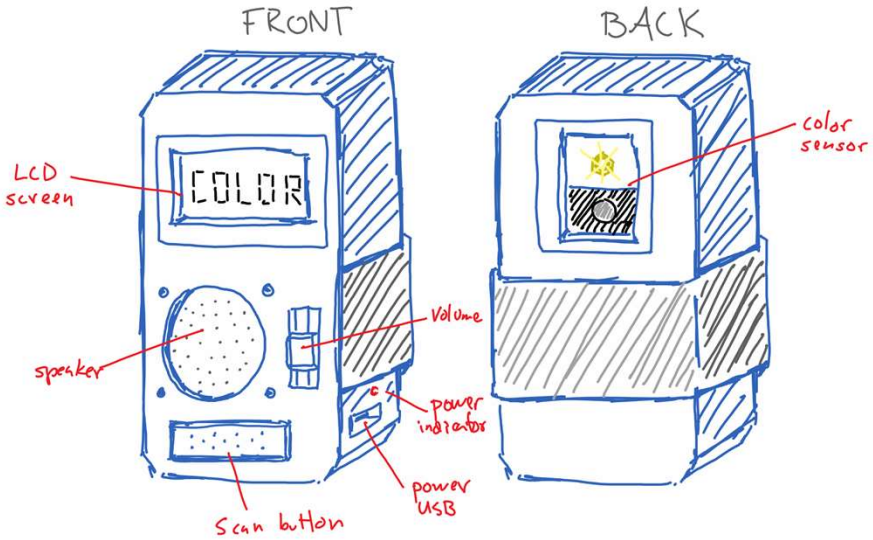
Device chosen	Pros	Cons
RGBC sensor	Cheap, accurate for at least 9 color labels	Specific to using SPI
OLED	Small (helps with modularity)	More expensive than other LEDs

Design – Hardware

Original product design sketches



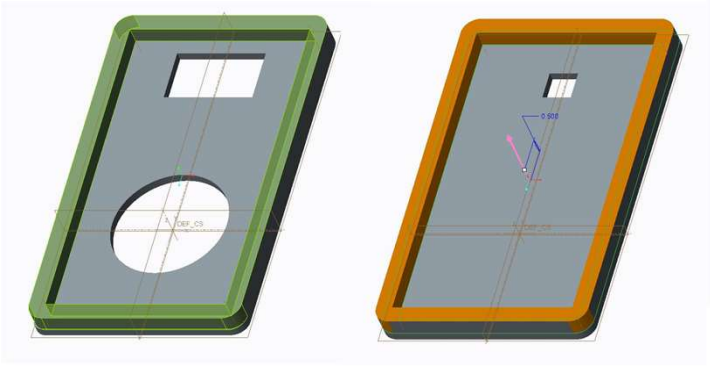
Original idea sketch



Main design sketch

Note the speaker which was later removed in final design.

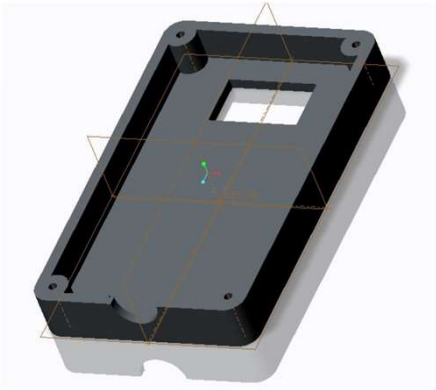
Design – Hardware



Front *Back*

Initial CAD schematic

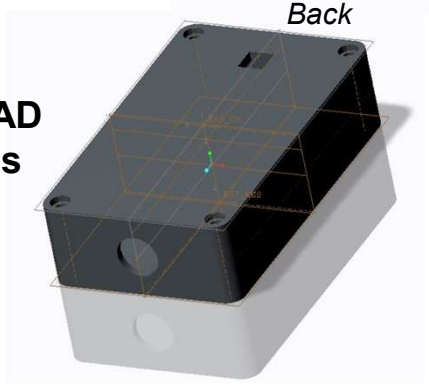
Note the speaker opening later removed in final design.



Front

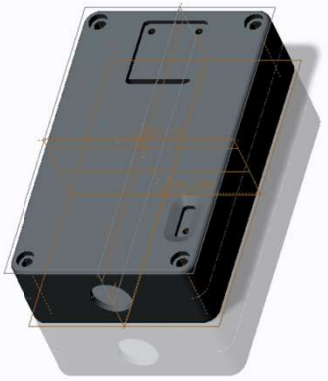
Back

Advance CAD schematics



Assembled

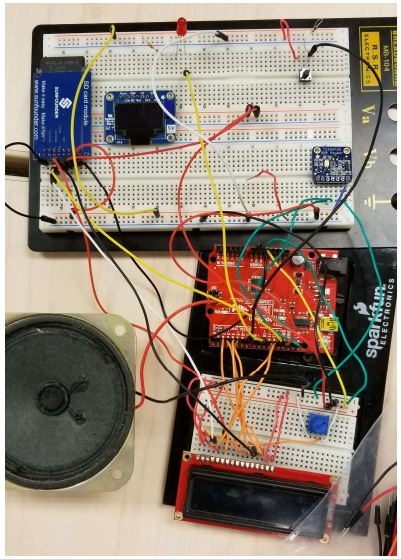
Final CAD schematic

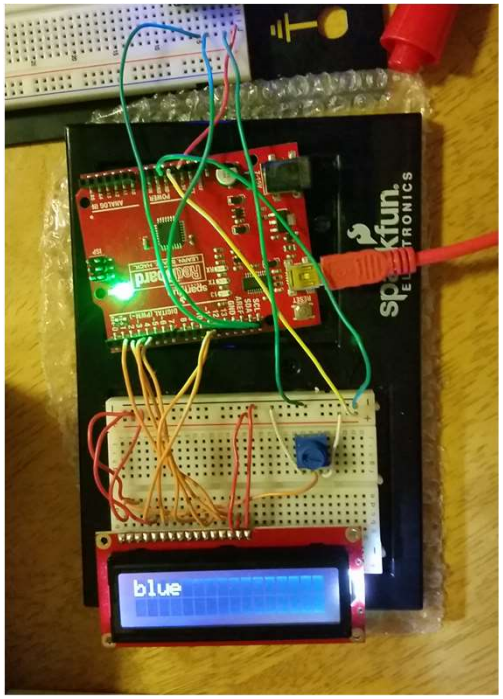


Back

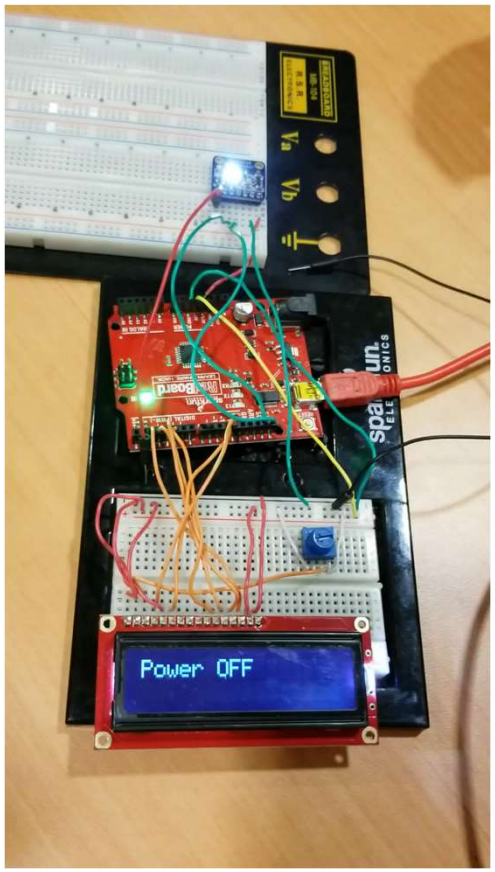
Design – Hardware

Internal components mock-up





Working Prototype Video



Design - Software

Software pros and cons for using Arduino.

Pros	Cons
Uses C / C++, which we already knew	Cryptic error messages require extra time debugging
Arduino IDE is easy to install	Requires knowledge of how the libraries work for Arduino

Implementation - Software

- Algorithm (next slide)
- Entire program (second slide)
- Main line program (third slide - no introductory comments above the `#includes` as shown on second slide).

We created a small library of code to increase abstraction and modularity of our firmware.

Can also be found here:
https://github.com/jeffudall/Team14/blob/master/Code/new_LCD_RGB_v5.ino

High-level algorithm of the Color Detector firmware.

```
Algorithm for the RGB sensor and OLED actuator

_____________________________________________
Get extra (.h or header) files: SPI, Wire, Adafruit_GFX, Adafruit_SSD1306, Adafruit_TCS34725, RGB_OLED_Helper_Functions
Create objects (display object for OLED, tcs object for sensor) to operate on RGB data
Create variables (16-bit unsigned ints) for storing RGB (and clear) data

(Do all other initializations, i.e. code required only once, in the void setup)
void setup:
  IF the baud rate, RGB, and LCD initialization function returns true:
        Print to Serial Monitor "Initialization successful"
  ELSE
        Print to Serial Monitor "Initialization failed"
        Optional: abort execution to void loop
  GO TO: void loop

(Do all repetitive code, i.e. behavior the program repeats forever, in the void loop, which loops forever until power is lost)
void loop:
  Get the RGB sensor data, then print RGB data to the Serial Monitor (using get_and_print_RGB())
  Get the color label and print it to the OLED (using get_and_print_clabel())
  Clear the OLED (using clearDisplay())
  GO TO: void loop
```

Note: We “latch” detected color values by using a while-loop (while detected color doesn’t change, keep displaying the last detected color).

Complete firmware screenshot (with short and long comment descriptions).

```
/*
Group 14
ECE 411
11/11/2017


Short description:
_____________________________________________________________________________________________
                Uses the TCS34725 RGB sensor to detect a color and the 326 OLED actuator to display the color detected.
_____________________________________________________________________________________________


Long description:
_____________________________________________________________________________________________
                The TCS34725 RGB sensor allows for detection of clear, red, green, and blue light as distinct components
                of incident light (to the sensor). Each sensor value (for clear, red, green, and blue) is a base-ten number,
                which can be used to distinguish different colors from each other. There are limitations to the capability
                of this device. For instance, detecting "light green" versus "dark green" could be difficult if not impossible
                because the RGB sensor readings are close to other colors. Also, unless the object is within 1 cm or less of
                the RGB sensor's photodiode array, the sensor's accuracy becomes worse. (Each photodiode in the array is
                responsible for one of the detected values of clear, red, green, and blue, which are controlled internally
                by a Finite State Machine.) The OLED uses Serial Peripheral Interface (SPI) to communicate with the Arduino.

                Expected input range (RGB sensor values): [0, 20,0000+] (dimensionless positive integers)
                Expected output (OLED actuator text): "Ambient", "Black", "Gray", "Blue", "Red", "Yellow", "Green", "Purple",
                                        "Pink", "Orange", "Brown", "Retry" (string of characters)

                If the RGB's LED pin is grounded, then the program will display the last detected label. Once the LED pin
                is ungrounded, the OLED will start to detect new colors again. The program displays the color detected on
                the OLED, which can (among other things) display white font on a black background, or black font on a white
                background.

                This program detects ambient, black, gray, blue, red, yellow, green, purple, brown, pink, and orange,
                using the RGB sensor. If the RGB sensor data does not map to one of these labels, then it displays "Retry"
                until it detects a known label mapping.

                The control logic for this program operates under the following assumption: a solid object is presented
                to the RGB sensor, within (or less than) 1 cm. The OLED's LED pin becomes ungrounded (after previously
                being grounded), and gets the RGBC values of the object. Then, an if-else structure determines which color
                was detected. Multiple solid objects of distinct colors were tested such that the unique colors tested
                were mapped to the following ranges for R, G, and B:

                r_l < R < r_h ;              g_l < G < g_h ;              b_l < B < b_h;

                where the _l and _h subscripts indicate the lower and higher bounds of the detected value, respectively.
                This means that after presenting a red object to the sensor, the values for R, G, and B can be approximated
                using the Serial Monitor. Once approximated, the lower and higher bounds that red could assume were
                implemented in the control structure (shown in the .cpp file for the helper functions).

                Helper functions from the RGB_OLED_Helper_Functions.h file were created by our team to facilitate the
                heavy-lifting for the program.
_____________________________________________________________________________________________


Version 5.0
_____________________________________________________________________________________________
                Detects white (ambient), black, red, orange, yellow, green, blue, brown, purple, and pink.
                Stores the last detected color if the LED pin is grounded.
                Uses functions to increase: abstraction, modularity, and maintainability.
_____________________________________________________________________________________________
*/

//Get libraries of code for functions needed in this program
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "Adafruit_TCS34725.h"
#include "RGB_OLED_Helper_Functions.h"

//Objects for the OLED
Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

//Global variables for storing the RGB sensor data
uint16_t clear, red, green, blue;

//Execute void setup() function only once
void setup() {
  if(init_RGB_OLED())
              Serial.println("Initialization successful\n");
  else
              Serial.println("Initialization failed\n");
}

//Execute void loop() function forever, as long as the circuit has power
void loop() {
  //Check RGB sensor and get the data, then print it to the Serial Monitor
  get_and_print_RGB();
  //Get the color label (clabel) and print it to the OLED
  get_and_print_clabel();
  //Clear the last color detected
  display.clearDisplay();
}
```

Firmware screenshot (no short / long comment descriptions).

```
63  //Get libraries of code for functions needed in this program
64  #include <SPI.h>
65  #include <Wire.h>
66  #include <Adafruit_GFX.h>
67  #include <Adafruit_SSD1306.h>
68  #include "Adafruit_TCS34725.h"
69  #include "RGB_OLED_Helper_Functions.h"
70
71  //Objects for the OLED
72  Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);
73  Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);
74
75  //Global variables for storing the RGB sensor data
76  uint16_t clear, red, green, blue;
77
78  //Execute void setup() function only once
79  void setup() {
80    if(init_RGB_OLED())
81                Serial.println("Initialization successful\n");
82    else
83                Serial.println("Initialization failed\n");
84  }
85
86  //Execute void loop() function forever, as long as the circuit has power
87  void loop() {
88    //Check RGB sensor and get the data, then print it to the Serial Monitor
89    get_and_print_RGB();
90    //Get the color label (clabel) and print it to the OLED
91    get_and_print_clabel();
92    //Clear the last color detected
93    display.clearDisplay();
94  }
```

Implementation - Hardware

Major components:

- **ATmega 328/P** (low power 8-bit CMOS microcontroller)
- **RGB Color Sensor** with IR filter and White LED (TCS34725)
- Monochrome 0.96” 128x64 **OLED graphic display**
- Lithium Ion Polymer **Battery** - 3.7V 1200 mAh
- PowerBoost 500 **Charger** - Rechargeable 5V Lipo USB Boost @ 500 mA+
- **Push button** (Reset)

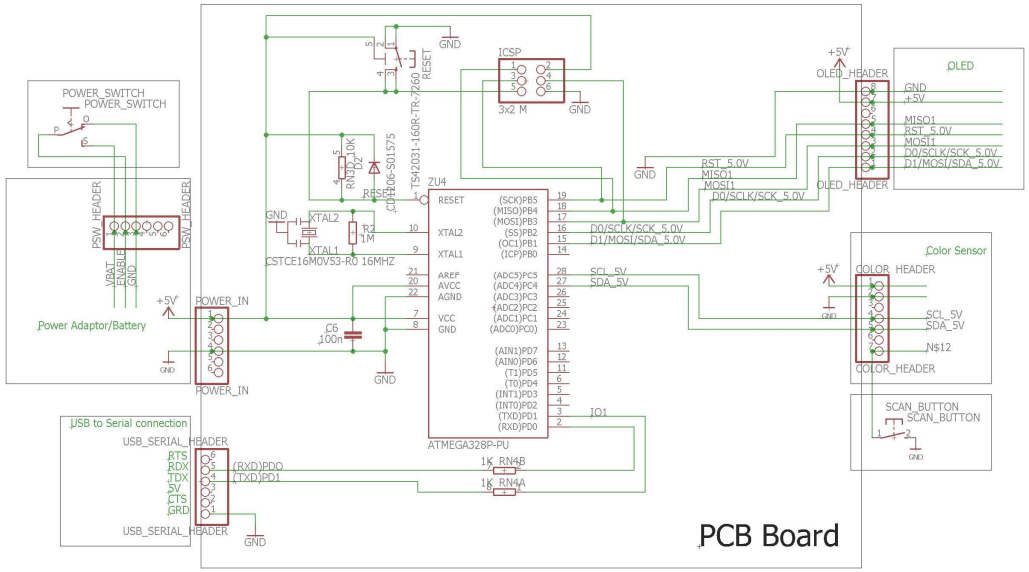
Bill of Materials (BOM)

Color detector BOM

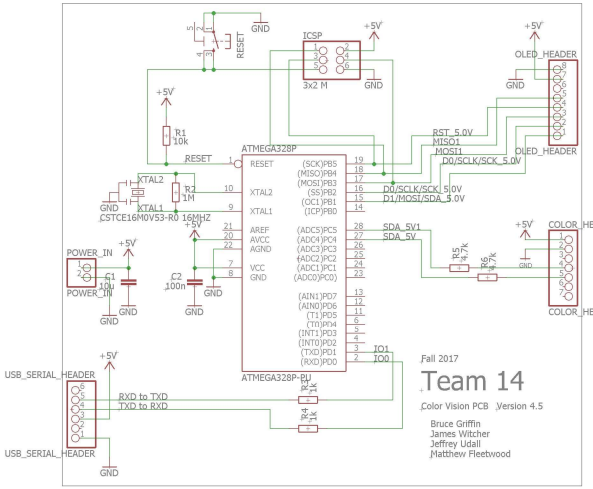
Part	Value	Device	Package	Description	Prices
COLOR_HEADER	COLOR_HEADER	HEADER-1X770MIL	1X07_ROUND_70	PIN HEADER	$0.26
C6	100n	C-EU0603-RND	C0603-ROUND	CAPACITOR, European symbol	$0.06
ICSP	3x2 M	PINHD-2X3	2X03	PIN HEADER	$0.12
OLED_HEADER	OLED_HEADER	HEADER-1X870MIL	1X08_ROUND_70	PIN HEADER	$0.26
POWER_IN	POWER_IN	PINHD-1X6	1X06	PIN HEADER	$0.26
POWER_SWITCH	POWER_SWITCH	EG1218S	EG1218	SLIDING SWITCH	$0.31
PSW_HEADER	PSW_HEADER	PINHD-1X6CB	1X06-CLEANBIG	PIN HEADER	$0.26
R2	1M	R-EU_R0603	R0603-ROUND	RESISTOR, European symbol	$0.06
RESET	TS42031-160R-TR-7260	TS42	TS42	TS42	$0.31
RN3	10K	4R-NCAY16		Resistor	$0.31
RN7 & RN8	4.7K			Resistor	$0.13
OLED Screen					$19.50
RGB Sensor					$7.95
enclosure					$3.73
RN4	1K	4R-NCAY16		Resistor	$0.06
SCAN_BUTTON	SCAN_BUTTON	31-XX	B3F-31XX	OMRON SWITCH	$2.00
USB_SERIAL_HEADER	USB_SERIAL_HEADER	PINHD-1X6	1X06	PIN HEADER	$0.26
Y2	CSTCE16M0V53-R0 16MHZ	RESONATORMU	RESONATOR		$0.42
ZU4	ATMEGA328P-PU	ATMEGA328P-PU	DIL28-3	MICROCONTROLLER	$2.18
				TOTAL	**$38.42**

Implementation - Hardware

Schematics:



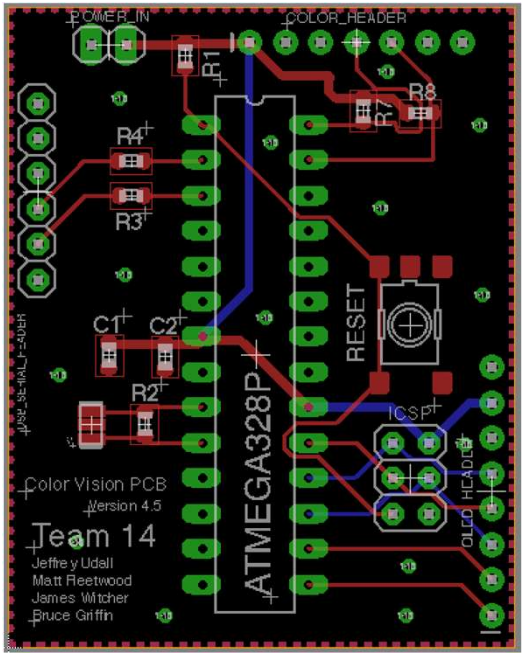
EagleCAD schematic for the Color Detector.



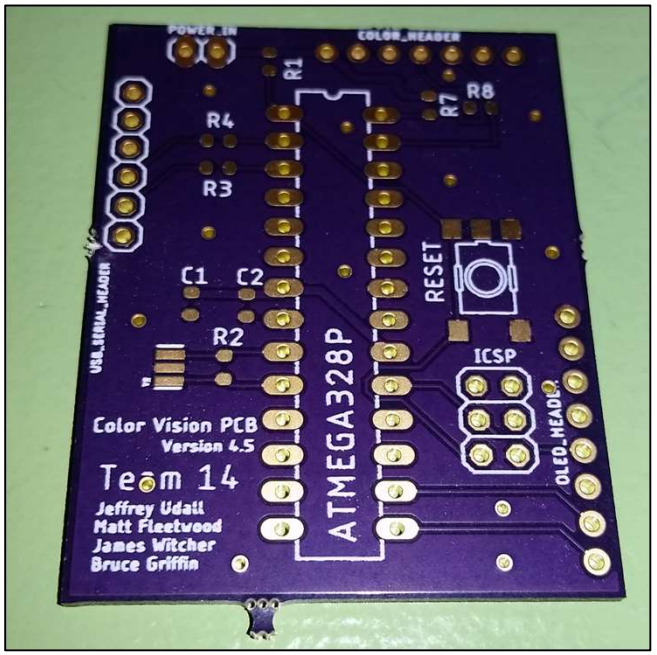
Final EagleCAD schematic for PCB

Implementation - Hardware

Board layout:



EagleCAD board layout

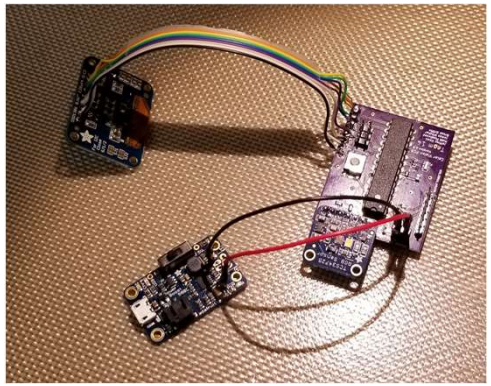


Final printed circuit board

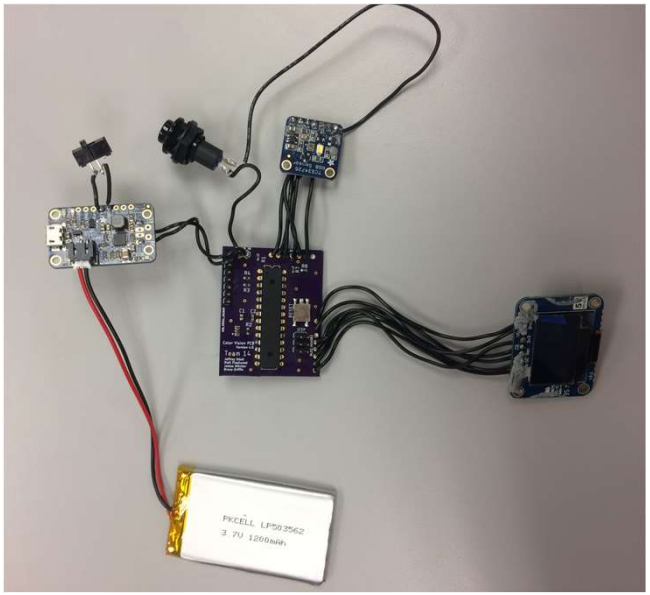
Implementation - Hardware

Assembled device internals:

Reflow Oven Soldered components to PCB



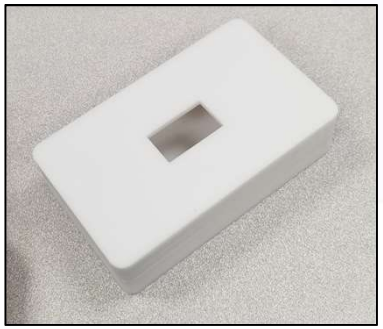
Hand Soldered Components on PCB



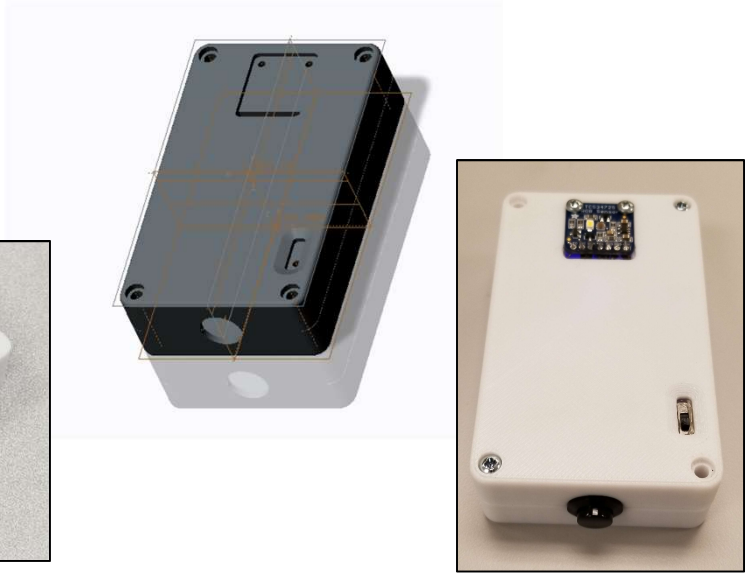
Final device – Internals

Implementation - Hardware

Enclosure:



3D printed enclosure prototypes

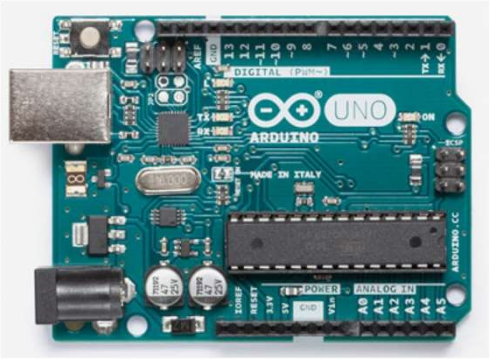


3D printed enclosure
Final design

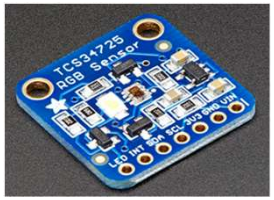
IP and Prior Work

We referenced other IP, tutorials, and prior work to design and build our device such as:

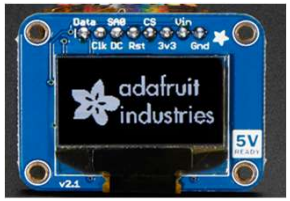
- Arduino open source development board
  https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf
- RGBC sensor tutorial, schematics, software libraries
  https://www.adafruit.com/product/1334
- OLED actuator tutorial, schematics, software libraries
  https://learn.adafruit.com/monochrome-oled-breakouts



Arduino UNO



RGBC sensor



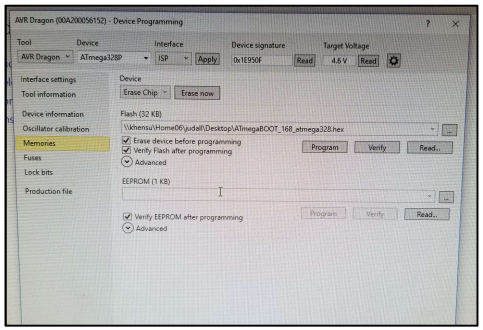
OLED actuator

Testing

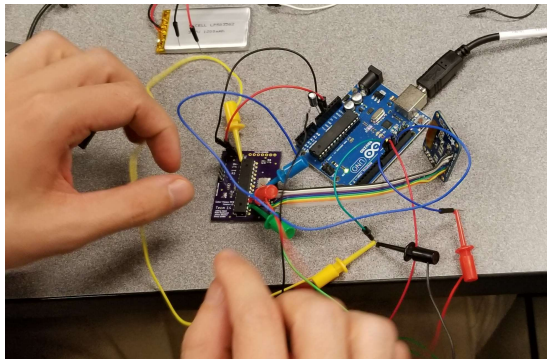
We follow the testing guidelines provided in ECE 411:

- White box test on the system
- Black box testing on individual components (e.g. the RGBC sensor, or the helper functions)

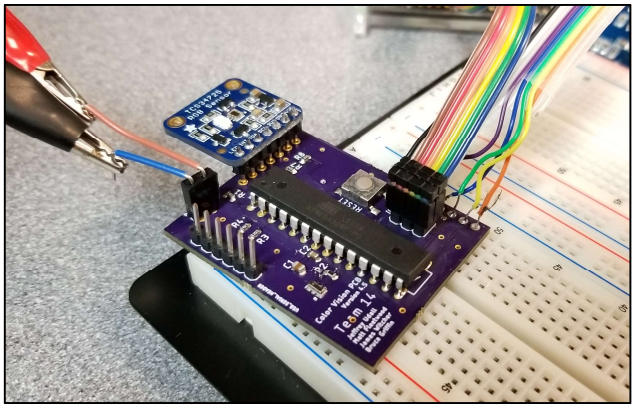
Testing - Troubleshooting

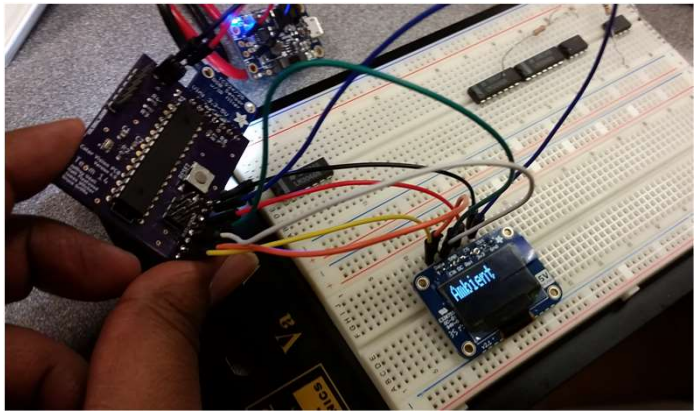


Programing microcontroller

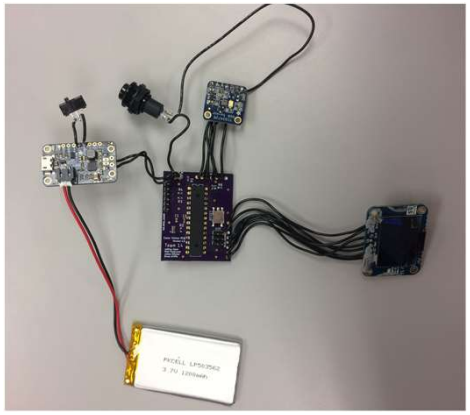


Troubleshooting bootloader program

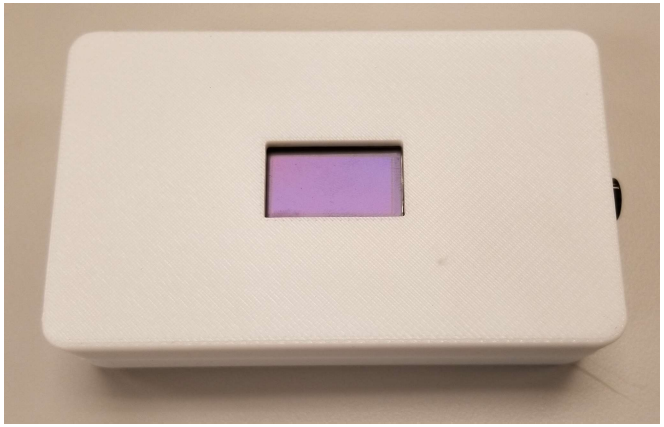




Results



Final device – Internals

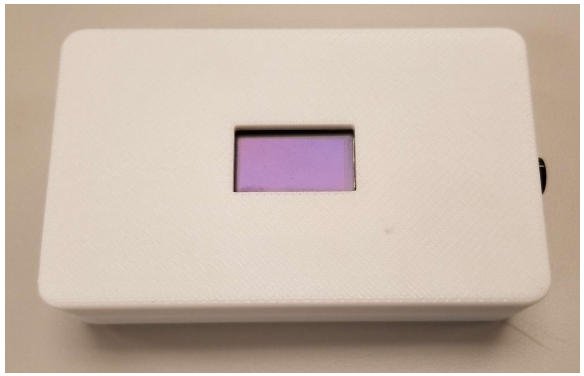


Final device – Front

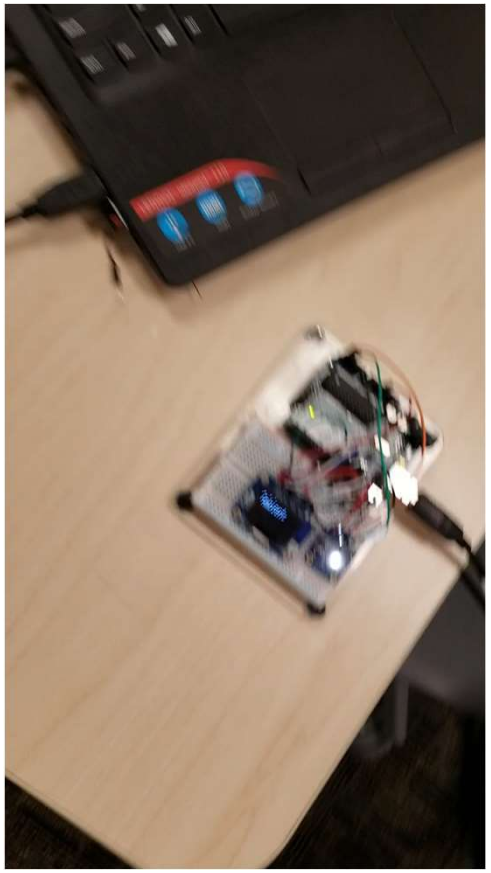
Final device – Back

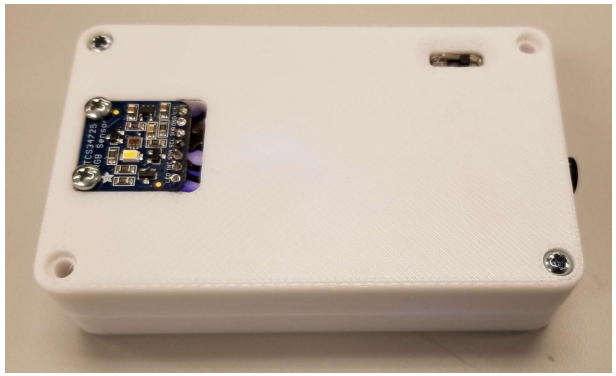
Final device – Scan button

Results



Final device – Front





Final device – Back

Results

What worked:

- Detection of up to 10 “colors”: gray, blue, red, yellow, green, purple, pink, orange, brown, black, ambient, and retry
  - The Color Detector works well but only in the expected environment (room with as much ambient light as our 411 room)

What didn’t work:

- Having enough time to learn about and debug more interesting features (e.g. using a speaker for second actuator, using a NN for the control structure)

Contributions

Team members and their roles

Team member	Role/Contribution
James Witcher	Parts management / ordering; HW design and test
Jeffrey Udall	Concept design, Project design Board design and layout;
Bruce Griffin	Board design and layout; power supply design
Matt Fleetwood	Software design and test; Hardware design/testing

Lessons Learned

We all learned something and would do things differently doing again

- **Matt Fleetwood** – How the bootloader and EagleCAD work; GitHub; making libraries in Arduino. Would have liked to research more sensors before staring project.
- **James Witcher** – EagleCad design; GitHub; Slack; System Design. Were ahead at beginning and then got complacent and fell behind.
- **Bruce Griffin** – Do not disassemble the working prototype, it can be useful to troubleshoot the final product. We learned about this stuff as we went, having these skills before starting would have been great – especially scheduling.
- **Jeffrey Udall** – Schematic design. EagleCAD design process. Design to final product workflow. Team dynamics. The importance of mock up-to-PCB parity. Should have been more proactive about ordering stencil.