# Pioneer Academics Research Program

# Automatic Clothes Finder for the Colorblind


**Bao To**


**Summer 2021**

**Submitted to:**

**Carlotta A. Berry, Ph.D.**

**September 5th, 2021**

**Executive Summary**

One challenge for people who are colorblind is finding clothes that match or are according to their needs. This system serves as a solution to that problem by providing an automated clothes finder system based on user input of the preferred color. The paper discusses the design of a color-based sensing system using modules to help colorblind people determine the color of their clothes. The paper justifies why each component is selected based on their tested performances. The paper also documents the software and hardware modifications in the design process. The test results show that the device can accurately detect multiple colors and coordinate the mechanical system to present the user with the clothing item with the correct color.

**Table of Contents**

# I. Introduction

According to previous research, 8-12% of all men and 1% of all women in the world are born colorblind (Harwahyu et al., 2013). Ribeiro et al. (2019) states that colorblind people are limited from perceiving colors, which in turn causes them trouble when interacting with other people or examining objects. Unfortunately, this condition cannot be cured by medical surgery or treatment, and as a minority, this group of people is often ignored by society (Lin et al., 2019).

The article "What do color-blind people see?" published by Etudo (n.d.) states: "Dressing up daily can be a chore." It goes on to mention labelling clothing items as a solution to this problem. However, a downside of labelling clothes is that you will have to label every new clothes item that is hung or stored in the wardrobe. Instead of doing so, modern developments might provide a more convenient, compact, and fast solution to this problem: using color-detection technologies.

The use of color sensing techniques to detect the colors of objects has been widely researched. In "Applied color sensor based solution for sorting in food industry processing," Alaya et al. (n.d.) used color sensors to detect and sort food items according to colors, while in "Automated Color Recognition System for Visually Challenged and Achromatopsia People using Arduino and Mobile App," Kumar et al. (2015) used the same color sensors to help colorblind people detect color using a mobile app. Although these systems can detect colors of certain uniformly-colored objects, they often fail to recognize colors accurately under the conditions of changing light levels or even when the objects are moved just slightly away from the sensor modules. With newly developed technologies such as artificial intelligence (AI) and machine learning, recently developed algorithms are able to detect discrete objects in real time, which also means that they are able to detect their colors with a much better accuracy and range of use.

# II. Research and Literature Review

## 2.1. Existing Solutions

While solutions to this particular problem, that is clothes detection, have not yet been widely available, color detection devices and algorithms have been developed and used for

various different reasons. Below are some of the papers that discuss existing solutions to the problem of color detection and their relevance to my project.

Lin, Chen, and Wang (2019) presented an image recoloring technique implemented with the aim of assisting colorblind people in identifying colors that they are not normally able to distinguish. They start off by looking into the types of colorblindness, some of the previous research, and discussing the issues that colorblind people face, and then go into discuss the "color warping" technique that helps converting the colors of an image from a RGB color space to a more suitable color space for colorblind people in real time. While they focus on presenting the conversion algorithm, the paper also provides results of using such techniques through the use of images, as well as human subjective evaluations. This paper's introduction provides valuable data on the problems that color blind people face, which will contribute to the problem statement that I have when writing my paper. In addition, the technique that is being discussed - color warping - is quite similar to what I plan to do for my device to work. I plan to set ranges of wavelengths or RGB values, so that the colors detected are "rounded" to the closest value and will print out as such (for example: a blue shirt may be a light blue or dark blue, but it will still be detected as blue). While generally simpler, my technique will be the same as theirs in terms of analysis and representation in the paper.

Alaya, Tóth & Géczy (n.d.) presented a color sorting system for fruits using an Arduino system. Their system includes the TCS3200 color sensor, stepper motors, and servo: the color sensor gets the color from the object under examination and send it to the Arduino, which translate it to values that can be used to then control the motors and servo to sort the object into the correct color group. This paper uses the same microcontroller and similar parts that I may need when making my project which may be a good reference for me when I use and test the subsystems such as that of the color sensor. They also convert the RGB colors from the sensor into the HSL (hue, saturation, light) values that are sent to the Arduino - similar to what I intend to do. The main difference between my device and theirs is the object being detected: clothing items instead of fruits. This may potentially be more challenging, as clothing items vary more in colors and may be less saturated, making it harder for the color sensor subsystem to detect and identify the color.

Ovchar et al. (2019) discusses the use of color sensors and servos to pick up and sort agricultural objects. Their system uses a camera to capture object images, a computer server that processes that picture and searches for the object using their object detection algorithm, and a servo motor that moves to the object and grabs it. The emphasis in their paper is on the object detection algorithm using neural networks to analyze the images. Their use of sensors and servos that are discussed is a good reference for my project, and their object detection part may be a good reference should I finish the project goals early and want to make it more complex. This paper also discusses an alternative to my solution in terms of color detection methods, which is to use machine vision using a more modern camera with an object detection algorithm instead of solely color sensors, which may make the data collected more accurate and stabilized. While this paper focuses on the algorithm to detect objects, which is not my area of focus for this project, taking into consideration the use of cameras may be useful for me in making decisions on hardware for the device.

Goudet (2009) designed a color sorting system for M&Ms. This system is aimed at providing support in the food industry. This paper has a strong emphasis on the engineering design side of the problem, focusing on designing and constructing the sorter prototype through 3d printing, assembly of the system as a whole, and measuring the accuracy of the TCS3200 color sensor. This system is aimed at providing support in the food industry, but their methodology and presentation is useful for me to review. The strong emphasis on the engineering design side of the problem is a good reference when discussing the hardware in my paper. They also use the same color sensor that I consider using (TCS3200), and also convert the colors detected by the sensor into the HSL color values in order for them to be processed by the program. A difference between this system and mine is that M&Ms are colored uniformly and with colors that can be categorized, while it is not the same situation when it comes to clothes.

### 2.2. Technical References

“Arduino Color Detection” (MJRoBot.orgFollow, n.d.) discusses color sensing with the TCS3200 in great detail. There is a block diagram, circuit layout, and code for implementing the TCS3200 sensor with the Arduino board. They also show how to set up the color sensing

process, from defining frequency scaling, selecting R, G, B colors to be read by the photodiode, and assigning them to the corresponding pins. While this paper is an introduction to using the TCS3200 color sensor, which I decided to not use, it also provides an abundance of information regarding project design and implementation. They also include parameters for a list of colors such as white, black, red, green, ... which will help me when determining the range of values for each color as they are detected by the color sensor. By reading about the use of the LCD display in their project, I am presented with an additional, extended solution that would make my device better suited for supporting colorblind people: including an LCD screen that shows a line of text, either the color that is being used to find clothing items, or the number of clothing items that matches the color input by the user through voice recognition.

## III. Requirements

### 3.1. Design Criteria

In designing the automatic clothes finder system, the ease of use is the top priority, as the system is geared towards aiding colorblind people in order to speed up their time when doing tasks. Similarly, time requirement is also important, with the process set to happen as fast as possible using this system design. In addition, the size and dimension of this device must be reasonable in order to fit in specific places, such as inside or on top of a closet. Lastly, the accuracy and efficiency of this system is also considered an important criterion: the device needs to be able to detect basic colors and provide the user with sufficient information about the clothing items that are being detected.

- Ease of use: Any novice user is able to use the device without too much difficulty. The device needs to have a way to input commands, the ability to detect colors, and mechanical functions that will work together to make the process easier for colorblind people.
- Time Requirements: Clothing items with the correct color must be identified within one minute.
- Size and Dimensions: The size of the device must be of appropriate length to fit in my room closet. The device is less than 30 pounds (~13kg).

- Accuracy/Efficiency: The device should be accurate enough to detect basic colors (yellow, blue, red,...). The arm movement should be accurate every time to a specific angle.
- Low Cost: The parts required to build the device should add up to less than $100.

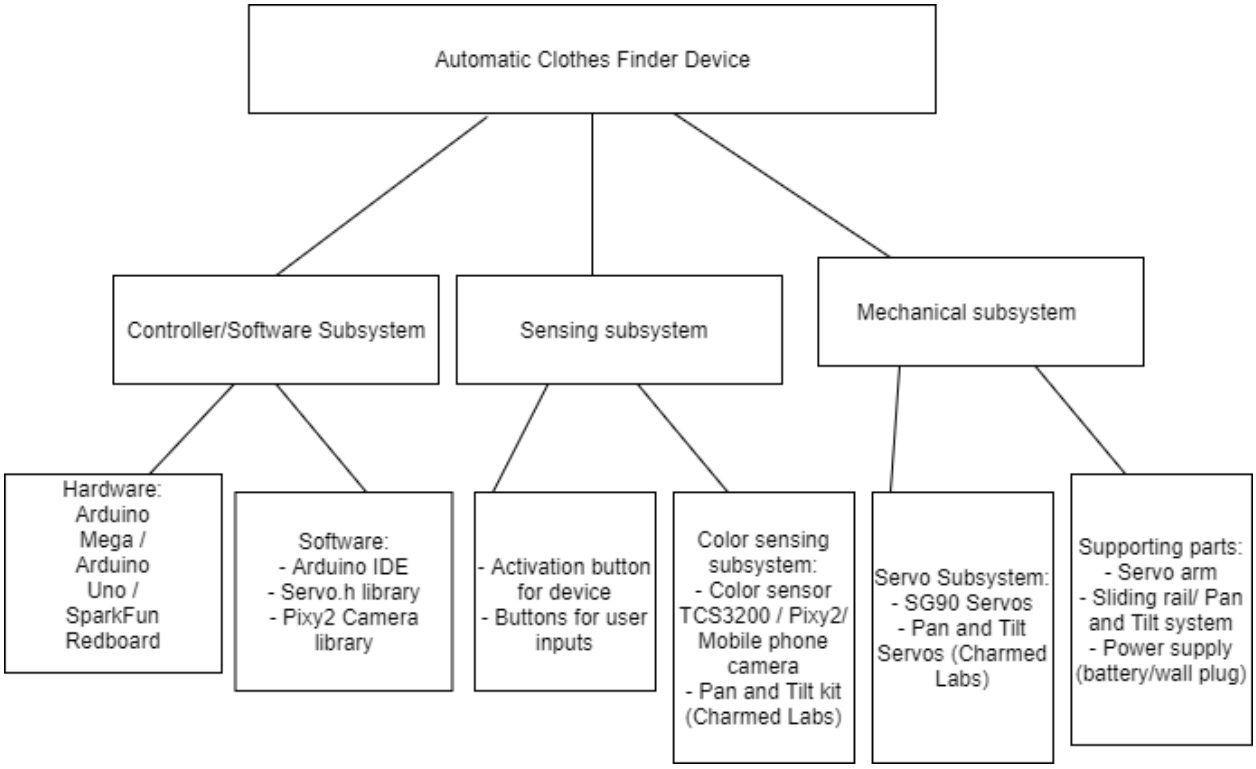
### 3.2. Design Specifications

The proposed system utilizes the technologies of object and color detection, and servo movement, which will interact with each other in a way that will improve the response time and user experience for colorblind people. The system will scan all clothing items that are hung on the arms. Next, it will save the color of each piece of clothing into the system. Finally, when the user inputs a color, the device will move the clothing items with that color to the front. This will allow the user, who are presumably colorblind, to identify clothing items with the colors that they want without much hassle.

## IV. Design Method

In order to design the clothes selection system, it was broken into smaller subsystems. The device must be able to receive information regarding the color needed, detect colors of items, and communicate using its microcontroller to move clothing items and the color detection device. As shown in Diagram 1, there are three main subsystems in the proposed design: the controller and software subsystem which allows for communication between components of the whole integrated system, the sensing subsystem which enables the device to detect colors and receive user inputs, and the mechanical subsystem which allows movement of items.

**Diagram 1: Proposed System Design**




When these subsystems are integrated, the device is able to turn on, detect the colors of items, take in input from the user, and use that input to determine the clothing item with the right color and control the servo to move and show that clothing item. The section below contains information for each subsystem, including their specifications and functions, as well as information for each individual component and their uses.

Figure 1 in Appendix A shows the proposed physical layout for the device. The servos are attached to a base, on which the Pixy2 camera is mounted. This setup along with the angles set by the two pan and tilt servos will allow the camera to see all clothing items when activated. In addition, the Arduino microcontroller and breadboard may also be mounted on the base. Lastly, the user input subsystem, either buttons or voice recognition module, will be mounted onto the breadboard.

### 4.2. Controller/Software Subsystem

The purpose of this subsystem is to communicate with other subsystems and components in the device. The electrical component used in this subsystem is the Arduino Mega 2560

microcontroller, which has 54 digital input/output pins and 16 analog inputs. The software used are: the Arduino IDE and PixyCam's PixyMon software.

The Arduino IDE provides a simple and effective way to write and upload code to the Arduino-based microcontrollers and devices. The microcontroller type and port can be selected and changed inside the IDE. The development environment also allows for effective debugging through the use of the serial monitor and console.

The PixyMon software is used to teach color signatures to the Pixy2 Camera. The camera can be taught by showing an object with a discrete color, then using the camera and application to save the color of the object as a signature. Seven different color signatures can be saved inside the program, and as such the camera is able to detect seven different colors at once. The accuracy that the Pixy2 camera has is due to the object detection algorithm that is running in the background of the PixyMon program, which allows the input to be analysed in real time and the object to be detected in the forms of small pixelated blocks.

### 4.3. Sensing Subsystem

The sensing subsystem consists of two smaller subsystems: the color detection subsystem and the user input subsystem.

#### 4.3.1. Color detection subsystem:

In order to aid the colorblind users, the device needs to be able to identify the color of the items by itself and send data to the microcontroller for further processing. It needs to be accurate enough to be able to detect basic colors. The three possible options for these functions have been analysed in the decision matrix below.

**Table 2: Decision Matrix for Color Detection Sensing Devices**

| **Feature** | **Weight** | **TCS3200 color sensor** | **Pixy2 Camera** | **Mobile Phone (through an app or website)** |
|---|---|---|---|---|
| **Low Cost** | 10% | 5 | 3 | 2 |

| | | | | |
|---|---|---|---|---|
| **Convenience** | 40% | 5 | 5 | 2 |
| **Accuracy** | 50% | 2 | 5 | 5 |
| **Total** | 100% | 3.5 | 4.8 | 3.5 |

In the above decision matrix, each option for color sensing is given a score between 1 and 5 inclusive for three criteria (low cost, convenience, and accuracy), with 1 being the least and 5 being the most suitable.

Convenience is considered based on the time required for colorblind people to start using and receiving data from the device. The systems that use the TCS3200 color sensors or Pixy2 Camera does not require external connections from the user, while the systems that uses a mobile phone as the color sensing device may require the user's phone to be connected to the preexisting system, which takes time and requires more work from the user's end.

Accuracy — the most important feature — is measured based on how well the devices can detect colors. While significantly less expensive to purchase compared to the other two options, the color sensors TCS3200 can only detect basic colors at a very close range, and slight variations or changes in RGB color values at further ranges. On the other hand, the Pixy2 Camera utilises object detection algorithms that allow it to detect discrete objects easily and save them using individual color signatures. Lastly, using the mobile phone may also allow for accurate color detections, with built-in electronic components and functions that enables the users to detect different colors accurately and quickly.

Based on the decision matrix above, the Pixy2 Camera is selected to be used for detecting colors.

The hardware used in this subsystem consists of the Pixy2 camera and the Pan and Tilt servos. The code for this subsystem is written in Arduino IDE and uploaded to the microcontroller.

- **Hardware:**

**Figure 2: Pixy2 Camera**

The Pixy2 Camera is selected because of its user-friendly application interface, well designed code library, and effective color detection program using object detection algorithms. It is more expensive (~$59) compared to some of the alternatives such as the TCS3200 color sensors (~$8), but the feature it brings far exceeds any sensor devices, as it has a camera instead of sensors.

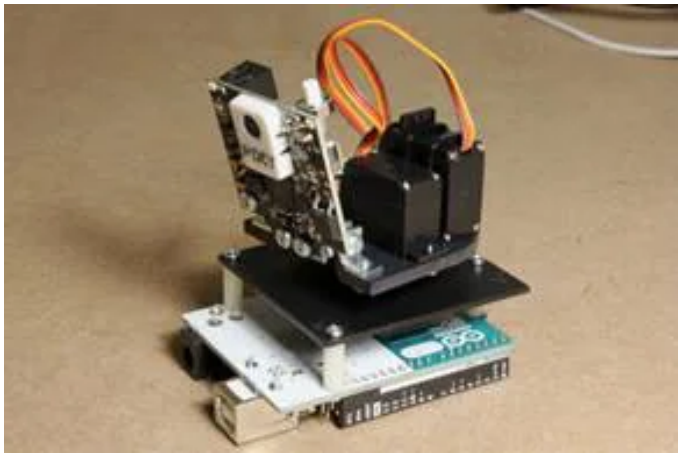


**Figure 3: Pan & Tilt kit assembled with Arduino and Pixy2 Camera**

The Pan and Tilt servos are part of the Pan and Tilt kit for the Pixy2 Camera, which allows for easy integration of the servos as part of the color sensing subsystems. As the name suggests, these two servos work together to move the camera to various angles, allowing the device to scan multiple objects in different places and detect their colors.

- **Code:**

The purpose of this code is to detect the colors of clothing items and save them in an array, allowing for communication with the arduino controller to enable the mechanical system to display the clothing item of the right color to the user. Upon uploading this program to the circuit, the pan servo will begin to move the camera around and scan the objects, saving its color signature to an array. The code written in the loop allows it to select unique values only (sadly, the array data structure is the only one available for arduino, so I could not use other types of data structures that adds only unique value, and instead had to write the code to make this function possible.) and add them into the array. This array stores the data of all uniquely detected colors, in other words all unique "clothing items" or objects. The array will then be printed out to the monitor. Appendix B contains the pseudocode and code for the project.

The array that is saved will be used to decide which servo in the mechanical subsystem may move upon receiving user input.

#### 4.3.2. User input subsystem:

The user input subsystem will enable the user to turn on the device, and choose the color of clothing items that they want. The two choices available for this function are voice recognition modules or buttons. One of the top criteria is that the system must be able to take in information from the user quickly. Below is the decision matrix for this part of the project:

| **Feature** | **Weight** | **Voice Recognition Module** | **Buttons** |
|---|---|---|---|
| **Low Cost** | 20% | 1 | 5 |
| **Quick Response Time** | 40% | 3 | 5 |
| **Accuracy** | 40% | 4 | 5 |
| **Total** | 100% | 3 | 5 |

In the above decision matrix, each option for activation and user input is given a score between 1 and 5 inclusive for three criteria (low cost, convenience, and accuracy), with 1 being the least and 5 being the most suitable. Based on the decision matrix, buttons were chosen for their quick response time, low cost, and high accuracy.

The user input subsystem consists of seven buttons that are connected to the Arduino microcontroller. Each button represents a color that the user may want to choose from. For example, if a user presses the red button, the clothing item with the color red will be selected and shown to them.

**Figure 4: Button for user input**

Most buttons designed to work with Arduino have the same configuration, with one leg for Ground (GND), and another leg for signal. These buttons are not polarized.

### 4.4. Mechanical Subsystem

The mechanical subsystem must be able to aid in the identification of the requested clothing item by showing it separately from other items. The two choices for the movement of clothing items are discussed in this decision matrix:

**Table 3: Decision Matrix for Mechanical Functions**

| Feature | Weight | SG90 Servos | Charmed Labs Pixy2 Pan and Tilt kit |
|---|---|---|---|
| **Low Cost** | **10%** | **5** | **4** |
| **Accuracy** | **50%** | **5** | **5** |
| **Effectiveness** | **40%** | **2** | **5** |
| **Total** | **100%** | **3.8** | **4.9** |

Both choices provide high accuracy, with both servo types moving to the correct position every time the code is uploaded. While the SG90 servos are very cost-effective, the Pan and Tilt kit outclasses the aforementioned option because it is a system designed by Charmed Labs for use with the Pixy2 Camera specifically. The two servos included in the Pan and Tilt kit along with the movement plates work with the Camera together effectively to detect multiple objects at once.

Based on the matrix, the Pixy2 Pan and Tilt kit is selected to control and move the Pixy2 Camera. However, for the mechanical subsystem which is used to move the clothing items, SG90 servos were used.

The mechanical subsystem consists of SG90 servos and the Pan and Tilt kit and servos from Charmed Labs.


**Figure 5: SG90 Servo**

The SG90 servos use pulse width modulations (PWMs) signals to control the horns into a designated angle. This is used to create a moving arm for clothing items, as when the microcontroller is plugged in, the servo horns stay in the designated angle, allowing for reliable use of the device.

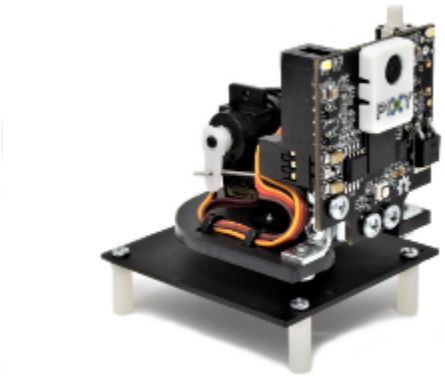


**Figure 6: Pan & Tilt Kit by Charmed Labs**

Charmed Lab's Pan and Tilt kit for Pixy2 Camera is selected due to their compatibility with the Pixy2 Camera and Arduino Mega 2560 microcontroller. The system consists of two servos, pan servo and tilt servo. The pan servo is used when the device is turned on and used by the user, while the tilt servo is used to adjust the level of the camera's field of vision when needed through Arduino code.

### 4.5. Full System Integration

Once each subsystem is built, wires and breadboards are used to integrate all subsystems into one fully functional device. Figure 8 shows the circuit layout for the device.

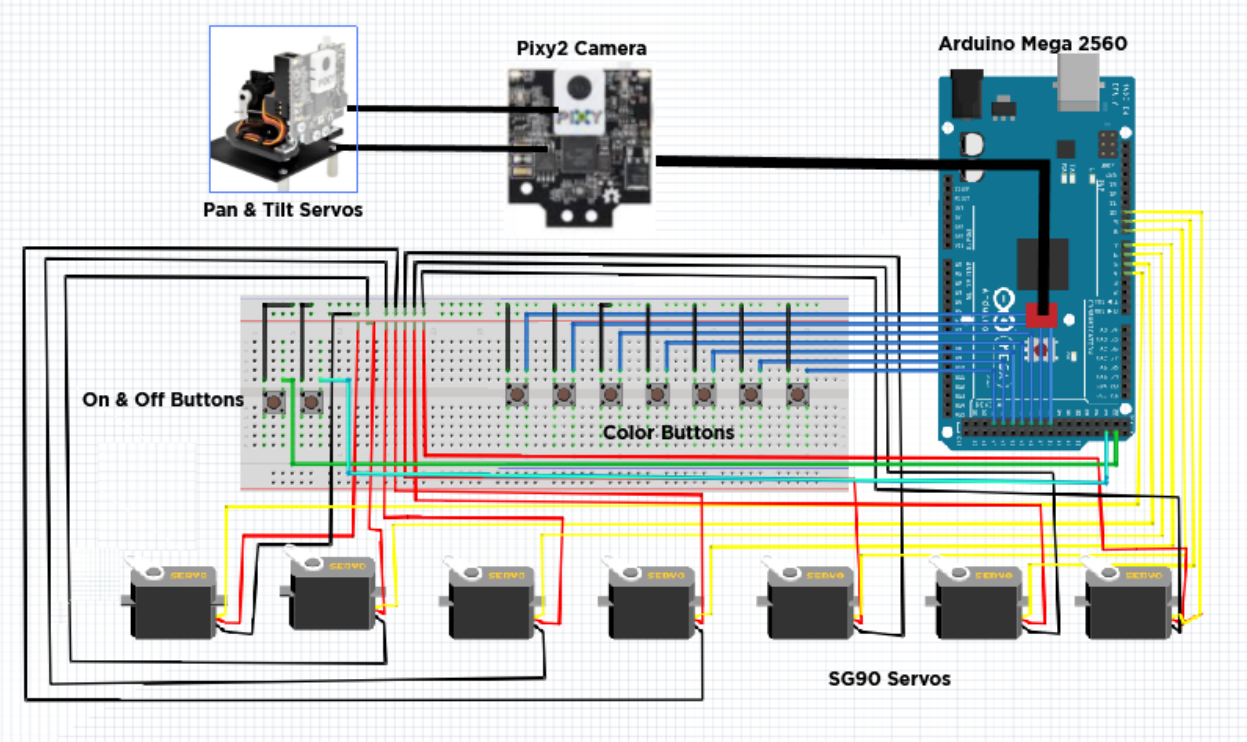



**Figure 8. Full system circuit layout**

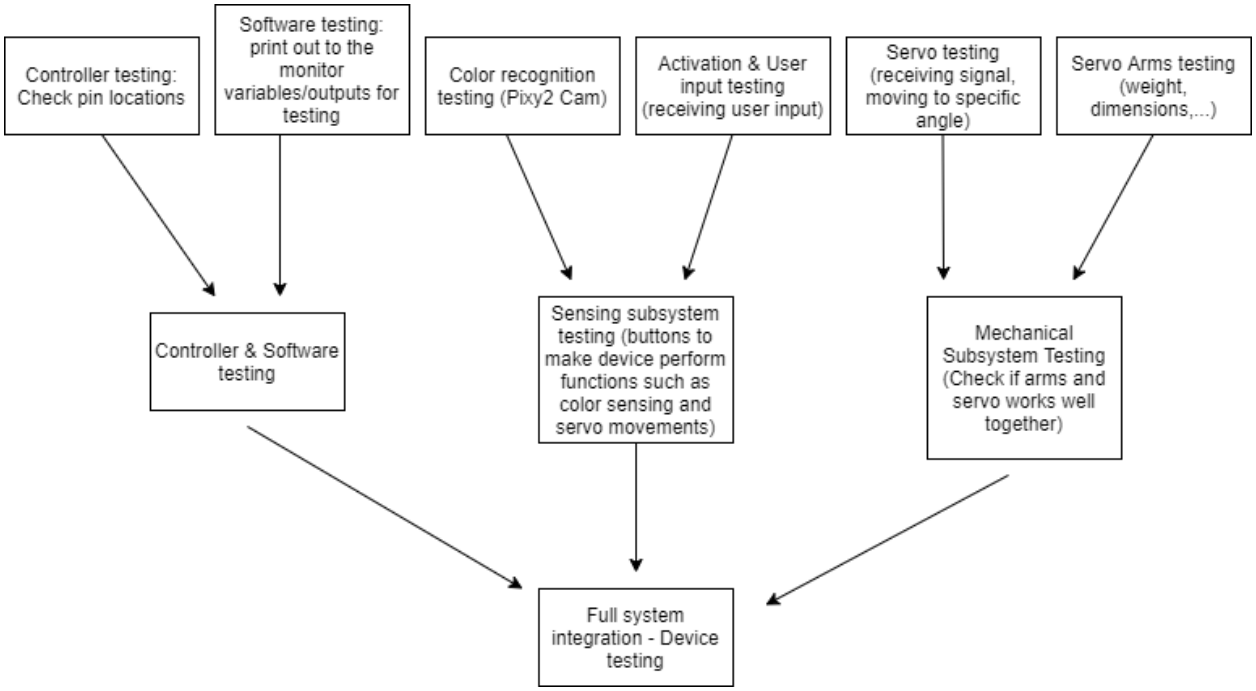
The SG90 servos of the mechanical system are connected to the microcontroller through pins 4 - 10, while the color buttons are connected through pins 36-48 (even-numbered pins). The on and off buttons are on pins 23 and 25 respectively. The Pixy2 Camera is connected to the microcontroller through the six ICSP pins located next to the AT Mega 2560 chip. The Pan and Tilt Servos are connected to the Pixy2, as they are directly compatible for use with the camera. An image of the real device is presented in Figure 9 in Appendix A.

## V. Verification and Results

In order to determine the accuracy and effectiveness of the proposed system, tests were carried out, starting with individual subsystems before full system testing and ending with testing

by multiple users. The following system describes the tests and analyses their results. Diagram 2 shows a block diagram of the testing that was conducted on the system, including component, functional and full system testing. Diagram 2 describes the process of testing, starting with testing individual components to testing subsystems' functions, and ending with testing the device as a whole.

**Diagram 2. Testing Process**




### 5.1. Component testing

#### 5.1.1. Pixy2 Camera Tests

First, the Pixy2 Camera was tested using the PixyMon program. Prior to testing, the camera received training through PixyMon's "Color connected components" (CCC) program. The program uses a connected components algorithm, which considers where each individual object is, its distance from the camera, and its size to tag and send its information to the Arduino microcontroller. Seven different objects with seven distinct colors were brought in front of the camera. Each object provides a color signature that is the same as the color of the LED attached to the Pixy2 Camera: red, orange, yellow, green, cyan, blue, and violet. Each color signature is numbered differently from each other from 1 to 7. While plugged into power, the uploaded program code allows the camera to detect an object's color.

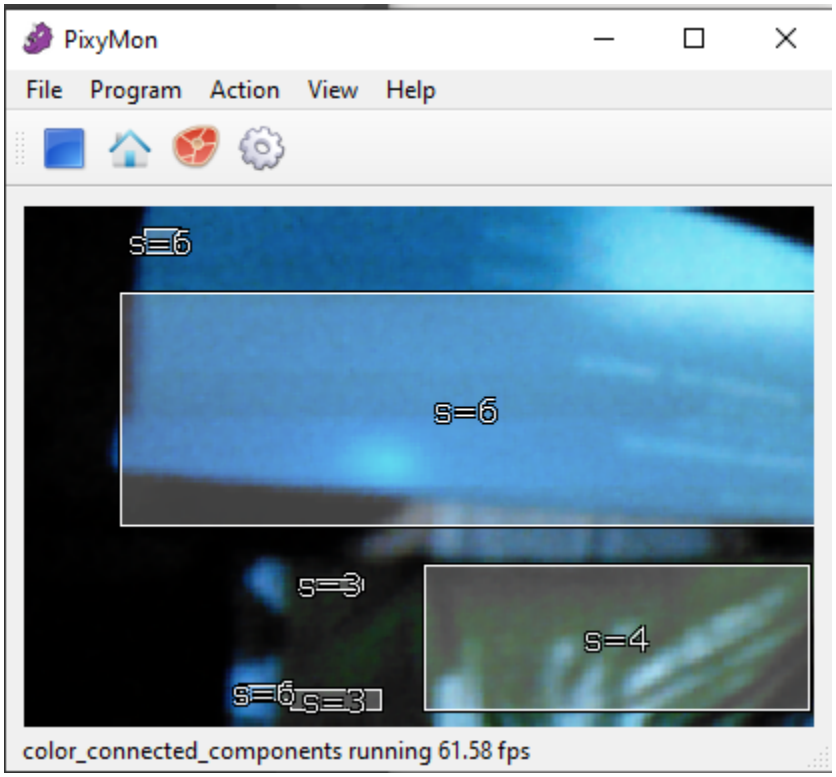



**Figure 10. Color & Object detection with Pixy2 Camera**

Figure 10 shows a testing process in which objects with the colors green and blue are brought in front of the camera. The blue object is marked with signature 6, and the green object is marked with signature 4, which are the correct signatures for the colors according to what was taught to the camera before testing. Observation through the PixyMon program shows that the objects detected have high accuracy with most colors. However, the camera sometimes detects light as the color yellow, and therefore, the light appears as multiple individual blocks with signature number 3 (3= yellow), such as that shown in Figure 6 in Appendix C.

The second test is to determine the accuracy of the camera with and without lights. Objects of colors blue, red, green, yellow, orange, purple (violet), and cyan were placed in front of the camera, and the results of whether the camera could detect each object were recorded in a yes/no format. If the object is detected, the time since the object first appears in front of the camera is recorded in seconds.

**Table 4. Detection of objects' colors with and without lights**

| Object color | With lights | Detected | Time since first appearance |
|---|---|---|---|
| Red | Yes | Yes | 1 second |
| Red | No | Yes | 4 seconds |
| Orange | Yes | Yes | 1 second |
| Orange | No | Yes | 4 seconds |
| Yellow | Yes | Yes | 1 second |
| Yellow | No | Yes | 5 seconds |
| Green | Yes | Yes | 2 seconds |
| Green | No | No | |
| Cyan | Yes | Yes | 1 second |
| Cyan | No | Yes | 5 seconds |
| Blue | Yes | Yes | 1 second |
| Blue | No | Yes | 5 seconds |
| Violet | Yes | Yes | 2 seconds |
| Violet | No | Yes | 5 seconds |

The results demonstrate that the camera takes more time to detect objects in dark environments, and in the case of green-colored items, the camera is likely to be unable to differentiate them from the surroundings.

### 5.1.2. SG90 Servos Test

Each of the seven SG90 servos of the mechanical subsystem is tested using code in Arduino IDE. A few lines of code are written for the servos to move to specific positions. An example line of code to set servos to a specific angle is shown below:

---

```
for (int i=0;i<=6;i++){
```

```
mechServos[i].write(90);
}
```

---

This line of code would set all seven servos to the 90 degree angle as intended. The process is repeated 6 times for each servo and for each angle, and the results are recorded in Table 5.

**Table 5. SG90 Servo Tests**

| Test Number | Requested Position | Average Recorded Position (of 7 servos) | % error |
|---|---|---|---|
| 1 | 0 | 0 | 0.00% |
| 2 | 45 | 44.5 | 1.11% |
| 3 | 90 | 89.7 | 0.33% |
| 4 | 135 | 135.2 | 0.15% |
| 5 | 180 | 179.8 | 0.11% |

Since all 7 servos move to almost exactly the same angles, the averages of their recorded angles are recorded for each test run. The small percent error shows that the servos work with high accuracy and can move their hands to specific locations as specified by the code.

### 5.2. Subsystem (Functional) Testing

The two subsystems that require testing are the sensing and mechanical subsystems.

In order to test the sensing subsystem, the code for the device is uploaded to the Arduino microcontroller, and several “print” statements are used to review the results using the serial monitor. Items of different colors were selected and brought in front of the camera. After all items are displayed, the array containing the color signatures for all detected items are printed out to the serial monitor. The values of the items in the array are then compared to the list of colors of items that were previously shown to the Pixy2 Camera. While the test

Next, the mechanical subsystem is tested using a qualitative test, where each servo is tested to see whether they can communicate with and move according to the microcontroller's signal. Results show that the array of sevens servos are fully functional and can successfully act according to the Arduino code.

### 5.3. Full System & User Testing: tests whether the project works and achieves the criteria and specifications.

Lastly, the device is tested as a fully-integrated combination of subsystems. Tests are designed to consider if the device fulfills all previously stated criteria and specification for the device.

#### 5.3.1. Design Specification

The device is used in 5 separate attempts, and the result of the attempts were recorded in the form of "success" or "failure". In order to be considered a successful attempt, after each button is clicked, the servo that is carrying the item of the correct color must move within two seconds. All 5 out of 5 attempts are successful, and the servo that holds the clothing item with the correct color moves immediately after the button representing that color is pressed. The results indicate that the camera successfully detects the colors of all clothing items, and is able to communicate with the controller to direct the servos to move the correct items.

#### 5.3.2. Design Criteria

##### 5.3.2.1. Time Requirement Testing

The first criteria that require testing is time requirement. As a part of the design specification test stated above, the device is timed from the second a button is pressed until when the servo hand holding the item with the correct color finishes moving. The procedure is repeated several times. Figure 10 shows the result of this test. The vertical axis represents the time recorded in seconds for each attempt.

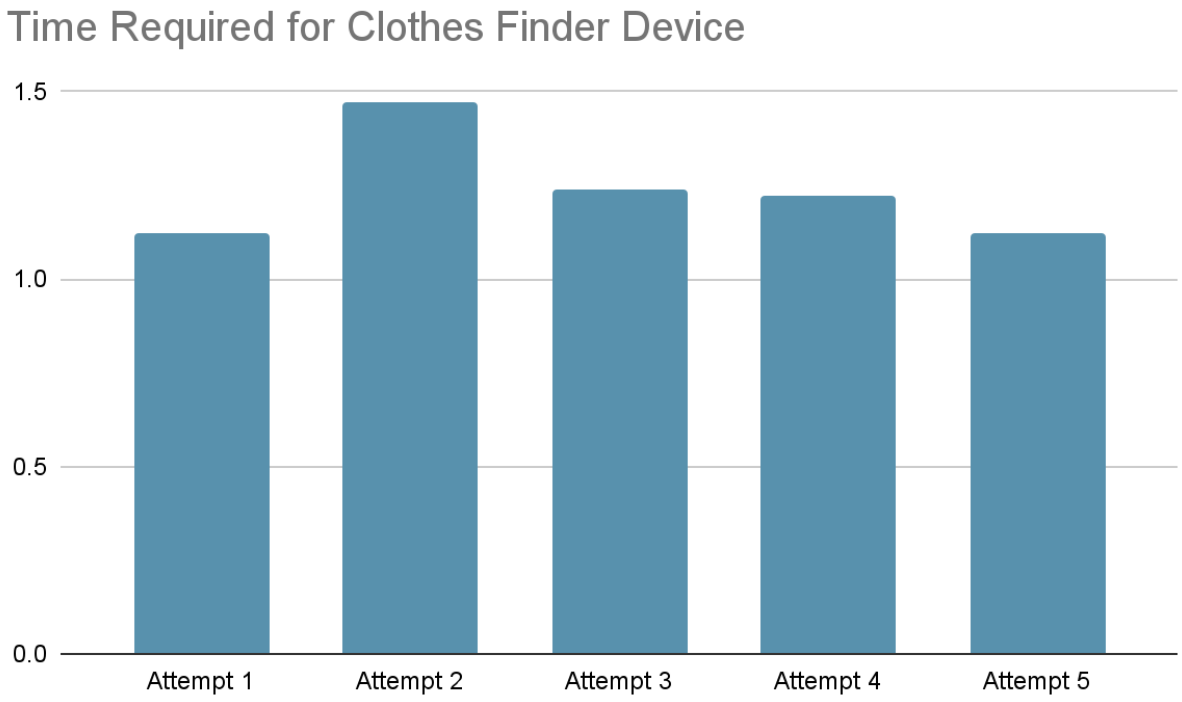



**Figure 10. Time Requirement Testing**

Test values show that the system works almost immediately: within one to two seconds, the item with the detected color was moved by one of the servos. This is a significant result, as it far exceeds the expectations stated in the design criteria - that the response time must be less than a minute.

#### 5.3.2.2. Accuracy testing

The second criteria that requires testing is accuracy. A qualitative test is performed, which includes multiple continuous attempts of pressing individual buttons to determine whether the correct servos moved. Out of 30 attempts, all servos in 30 attempts moved correctly.

## VI. Conclusions and Recommendations

In conclusion, the paper proposes a clothes finder system that is able to provide support for colorblind people when choosing their clothes. The device is able to record the colors of all clothing items in front of its camera, and save the items list so that it can show the user the right item color the next time they use the device. In addition, because of the use of the color detection

camera, the clothing items do not need to be put in the same order every time. The user can simply reset the device for it to rescan all clothing items and save their colors. The proposed system design from the design method passes multiple tests, and is able to perform most functions as stated in the criteria and specifications. While specific in function, the implementation of this device may be quite useful for colorblind people in one of their day-to-day tasks.

Further development of this system may open up to multiple research opportunities. In terms of the device with the specific focus on detecting colors of clothing, the following improvements may be made:

The color detection camera may incorrectly detect light sources as the color yellow, which may mess up the order of the items list and make the device show the user the wrong clothing item. If the algorithm for object detection is improved, the device may be able to ignore light sources when analyzing the live video feed from the camera.

Another problem that is detected during testing is the time it takes for items to be detected when there is a lack of lighting. A system of  LEDs may be installed behind the camera to improve the system's speed for color detection.

The last problem regarding this project is that each clothing item must have a different color from the others in order for the program to work correctly. In order to counter the problem of light sources being detected as a clothing item, the code is written so that no color signature may appear in the array twice, meaning two clothing items of the same color would not both be detected. Similarly to what is stated in the previous paragraphs, the use of a better algorithm for object and color detection may be a solution to look into.

On the other hand, other uses of this system may open up for further possibilities, such as use of this system in factories similar to that developed by Goudet (2009), or for making a mobile color detection system such as that discussed by Chaudhry et al. (2015). Since both of these previous research projects used the TCS3200 color sensors, replacing them with the Pixy2 Camera may provide much more accurate color detection and faster reaction time.

## VII. References

Alaya, M. A., Tóth, Z., & Géczy, A. (n.d.). Applied color sensor based solution for sorting in food industry processing. Periodica Polytechnica Electrical Engineering and Computer Science. https://doi.org/10.3311/PPee.13058.

Chaudhry, S., & Chandra, R. (2015). Design of a mobile face recognition system for visually impaired persons. *arXiv preprint arXiv:1502.00756*.

Etudo, M. (n.d.). *What do color-blind people see?* Verywell Health. https://www.verywellhealth.com/what-do-color-blind-people-see-5092522.

Goudet, L. C. (2009, December 11). *M&M's® Color Sorter Final Project Report*. http://pages.hmc.edu/harris/class/e155/projects09/Goudet.pdf.

Harwahyu, R., Manaf, A. S., Ananto, B. S., Wicaksana, B. A., & Sari, R. F. (2013). Implementation of Color-blind Aid System. *J. Comput. Sci.*, *9*(6), 794-810.

Kumar, M. A., Jilani, S. A. K., Sreenivasulu, U., & Hussain, S. J. (2015). Automated Color Recognition System for Visually Challenged and Achromatopsia People using Arduino and Mobile App. Int. J. Adv. Res. Electron. Commun. Eng, 4(8), 2106-2110.

Lin, H. Y., Chen, L. Q., & Wang, M. L. (2019). Improving discrimination in color vision deficiency by image re-coloring. *Sensors*, *19*(10), 2250.

MJRoBot.orgFollow. (n.d.). *Arduino Color Detection*. Instructables. Retrieved August 9, 2021, from https://www.instructables.com/Arduino-Color-Detection/#:~:text=%20Arduino%20Color%20Detection%20%201%20Step%201%3A

Ovchar, K., Borodin, A., Burlachenko, I., & Krainyk, Y. (2019, November). Automated recognition and sorting of agricultural objects using multi-agent approach. In *Proceedings of the 2nd Student Workshop on Computer Science & Software Engineering (CS&SE@ SW 2019), Kryvyi Rih, Ukraine* (pp. 76-86).

Ribeiro, M., & Gomes, A. J. (2019). Recoloring algorithms for colorblind people. *ACM Computing Surveys*, *52*(4), 1–37. https://doi.org/10.1145/3329118

\

## VIII. Appendices

A. Hardware (CAD layout, circuit layout)

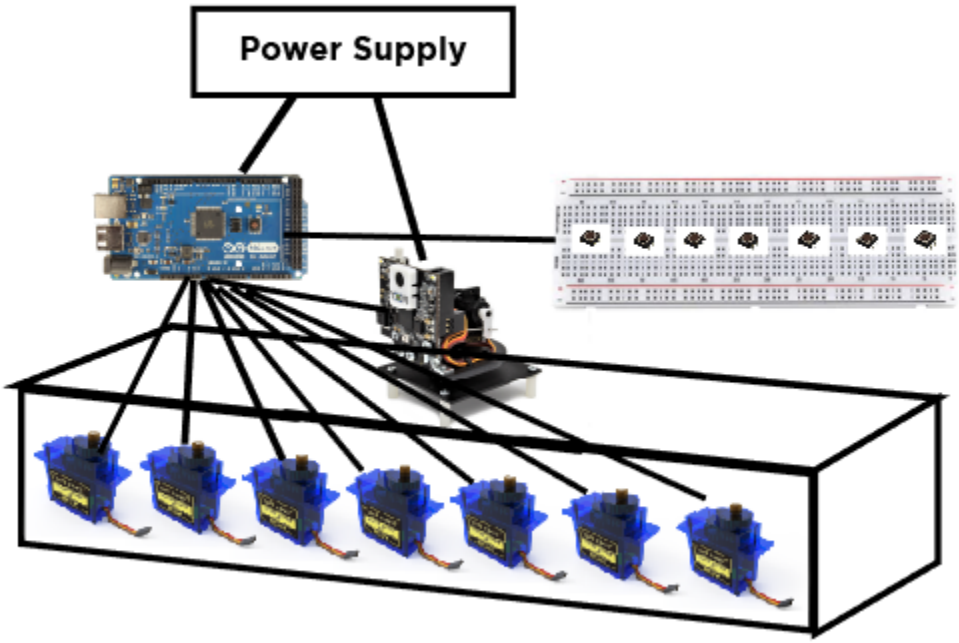



**Figure 1. Proposed Physical Layout**

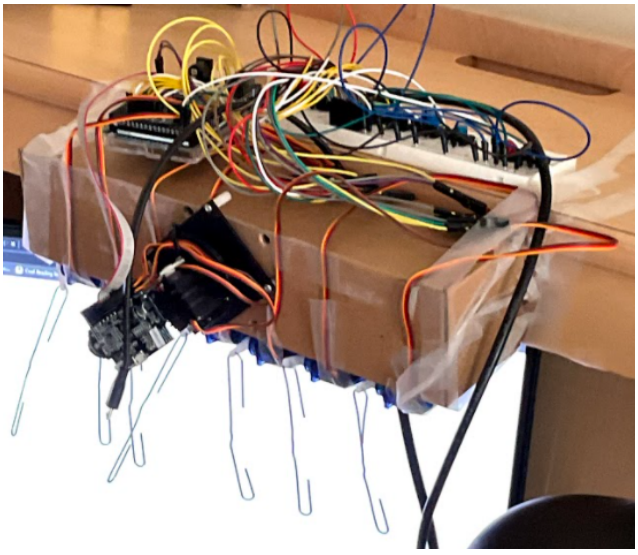


**Figure 8. Photo of actual device**

B. Software

**Pseudocode**

/*

Hardware Connections:

- Mechanical Servo 1 to PWM pin 10
- Mechanical Servo 2 to PWM pin 9
- Mechanical Servo 3 to PWM pin 8
- Mechanical Servo 4 to PWN pin 7
- Mechanical Servo 5 to PWM pin 6
- Mechanical Servo 6 to PWM pin 5
- Mechanical Servo 7 to PWM pin 4
- Button 1 to digital pin 48
- Button 2 to digital pin 46
- Button 3 to digital pin 44
- Button 4 to digital pin 42
- Button 5 to digital pin 40
- Button 6 to digital pin 38
- Button 7 to digital pin 36
- On Button to digital pin 23
- Off Button to digital pin 25

*/

PSEUDOCODE:

1. Initiate all variables and indicate pins for buttons
2. Setup:
- Begin serial connection with the computer
- Start the Pixy Camera
- Call the "colors_connected_components" program for Pixy2 Camera
- Attach Servos to their pins
- Set Button pins as inputs

3. Color Detection:

- If the ON button is pressed:
    - Check for user inputs from buttons
    - Set the "detected" boolean value to false
    - If the camera has not passed the threshold for movement to the left, continue to rotate to the left, else
    move camera back to the starting position on the right
    - If the object has been detected before, set the "detected" boolean to true and does not add it to the list of items, else
    add the color of the newly detected objects to the list of items
    - Goes through the list of items and prints out the color signatures representing colors of all items detected to the serial monitor
- If the OFF button is pressed:
    - Stop the device

4. If the device receives an input from the user:

- Check which button was pressed
- If a button is pressed:
    - Check what color the button represents
    - Check if any of the clothing items match the color needed by the user
    - If a clothing item matches the color, move the servo with the clothing item up; if the clothing item doesn't match the color, the
    servo does not do anything

## Code

This is the code for the automatic clothes finder project. The purpose of this code is to detect the colors of clothing items and save them in an array, allowing for communication with the arduino controller to enable the mechanical system to display the clothing item of the right color to the user. Upon uploading this program to the circuit, the pan servo will begin to move the camera around and scan the objects, saving its color signature to an array. The code written in the loop allows it to select unique values only (sadly, the array data structure is the only one

available for arduino, so I could not use other types of data structures that adds only unique value, and instead had to write the code to make this function possible.) and add them into the array. This array stores the data of all uniquely detected colors, in other words all unique "clothing items" or objects. The array will then be printed out to the monitor. Afterwards, when the user presses a button in the user input subsystem, the microcontroller will receive the signal to send its own signal to the correct servo, which will move the clothing item to show it to the user.

Functions:
- pixy.init(): start the camera
- pixy.changeProg(string): call another program to use for PixyMon app
- pixy.setServos(int,int): set pan and tilt servo values for the camera to move
- pixy.ccc.getBlocks(): detect colors
- checkUserInput(): check which button is pressed
- moveColorServo(int colorSig): go through all the array and check if any of the items in the array has the right color signature as chosen by the user. if yes, move the servo holding that item to show to the user.

Variables:
- pixy.cc.blocks[0].m_signature: the block value is 0 by default of pixy library is set to the signature of the biggest object detected,
which is useful because it allows more accurate and directed color detection of the object that is the closest to the screen,
instead of random objects everywhere in the camera space.

*/

```
//-----------------------------------------
//include all necessary libraries
#include <Pixy2.h>
#include <PIDLoop.h>
#include <Servo.h>

Pixy2 pixy; //create Pixy2 object
```

```
int detectedObjects[10]; //create a detected objects array
int starter = 0; //int to be used in object detection algorithm
int servoMove = 0; //move variable for pan and tilt servos
int mechServoMovement = 90; // move variable for SG90 servos
Servo mechServos[7]; //array used to control mechanical subsystem's (SG90) servos
boolean detected = false; //used to represent whether an object is detected or not
boolean deviceOn = false; //used to represent whether the device is activated or not
//creates all servos objects

int purpleKeyPin = 36; //set all pins for buttons for colors and ON/OFF
int blueKeyPin = 38;
int cyanKeyPin = 40;
int greenKeyPin = 42;
int yellowKeyPin = 44;
int orangeKeyPin = 46;
int redKeyPin = 48;
int onButtonPin = 23;
int offButtonPin = 25;

void setup() {
  //runs once
  Serial.begin(115200); // begin serial connection with the serial monitor
  Serial.print("Starting...\n");

  pixy.init();//start the camera
  pixy.changeProg("color_connected_components"); //call the color detection program



  mechServos[0].attach(4); //attach SG90 servos to the correct signal pins on the microcontroller
  mechServos[1].attach(5);
  mechServos[2].attach(6);
  mechServos[3].attach(7);
  mechServos[4].attach(8);
```

```
  mechServos[5].attach(9);
  mechServos[6].attach(10);

  for (int i=0;i<=6;i++){

    mechServos[i].write(0); //set all SG90 servos' arm location to 0 during set up
  }
  pinMode(purpleKeyPin, INPUT_PULLUP); //set all button pins to input (pullup)
  pinMode(blueKeyPin, INPUT_PULLUP);
  pinMode(cyanKeyPin, INPUT_PULLUP);
  pinMode(greenKeyPin, INPUT_PULLUP);
  pinMode(yellowKeyPin, INPUT_PULLUP);
  pinMode(orangeKeyPin, INPUT_PULLUP);
  pinMode(redKeyPin, INPUT_PULLUP);
  pinMode(onButtonPin, INPUT_PULLUP);
  pinMode(offButtonPin, INPUT_PULLUP);


}

void loop() {


  if (digitalRead(onButtonPin) == LOW)deviceOn = true; //if the on button is pressed, the device
will turn on
  if (digitalRead(offButtonPin) == LOW)deviceOn = false;//if the off button is pressed, the device
will stop working


    if (deviceOn == true) { // only runs if the device is on

    checkUserInput(); //check which button is pressed
    detected = false; //set detected boolean to false
    if (servoMove < 1000)servoMove+=2; //change the horizontal location of the pan servo
```

```
    else servoMove = 0;
    Serial.print("servo moved ");
    Serial.println(servoMove);
    pixy.setServos(servoMove,600); //set the horizontal value of the servo to make it move horizontally

    pixy.ccc.getBlocks();//getBlocks must be called after setServos because setServos reset the getBlocks method, making the camera unable to collect the signature of the biggest object
    for (int i = 0; i <= 9; i++) {
      if (pixy.ccc.blocks[0].m_signature == detectedObjects[i]) //if the object is already detected, it is no longer added to the array as a separate value --> no duplicates
      {
        detected = true;
      }
    }
    Serial.print("block detected is = ");//print out the currently detected object.
    Serial.println(pixy.ccc.blocks[0].m_signature);
    Serial.println(detected);
    Serial.println("starter is = " + starter);
    if (pixy.ccc.blocks[0].m_signature >= 1 && pixy.ccc.blocks[0].m_signature <= 7 && detected == false && starter <= 9)
    //if the color signature is valid (between 1 and 7), it has not been detected before, and the value is within range of the array, set the color signature to be a part of the array
    {
      detectedObjects[starter] = pixy.ccc.blocks[0].m_signature;
      starter++;
      Serial.print("detected block is " );

      Serial.println( detectedObjects[starter]);
      //block 0 is by default the largest block in the picture.
      //Therefore as the camera tilt, that block will be added to the detectedObjects array.

      // delay(100);
    }
```

```
    for (int j = 0; j <= 9; j++) { // prints out the array of uniquely detected objects

      Serial.println(detectedObjects[j]);
      //delay(500);
    }
    Serial.print("\n");

     }
  }


    void checkUserInput() {
    if (digitalRead(purpleKeyPin) == LOW)moveColorServo(7);//if the button is pressed, move the servos that contains the right color signature
    if (digitalRead(blueKeyPin) == LOW)moveColorServo(6);
    if (digitalRead(cyanKeyPin) == LOW)moveColorServo(5);
    if (digitalRead(greenKeyPin) == LOW)moveColorServo(4);
    if (digitalRead(yellowKeyPin) == LOW)moveColorServo(3);
    if (digitalRead(orangeKeyPin) == LOW)moveColorServo(2);
    if (digitalRead(redKeyPin) == LOW)moveColorServo(1);
    }

    void moveColorServo(int colorSig) {
    for (int k = 0; k <= 6; k++) {

      if (detectedObjects[k] == colorSig) mechServos[k].write(mechServoMovement); //go through all the array and check if any of the items in the array has the right color signature as chosen by the user. if yes, move the servo holding that item to show to the user.

    }
    }
```

C. Images and Videos

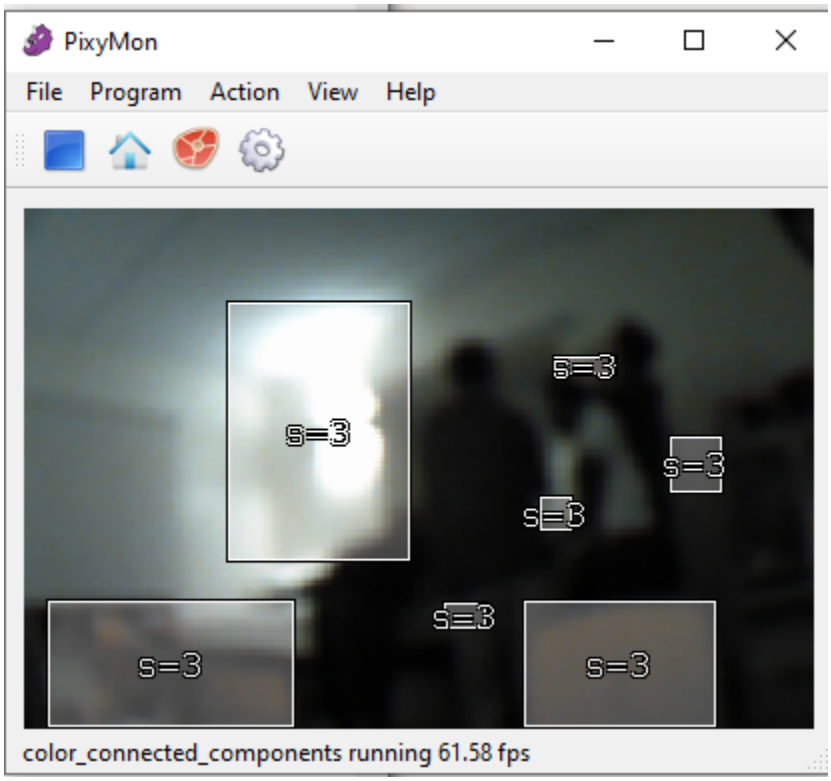



**Figure 6. Detection affected by lights condition**