Skin Tone Tracking Device (ChromaBand)

Author: Ziyang Bian
Advisor: Tina Smilkstein

Senior Project
Computer Engineering Department
California Polytechnic State University San Luis Obispo
June, 2017
**Table of Contents**

Chapter 0 - Abstract................................................................................................................. 3

Chapter 1 - Introduction........................................................................................................... 4

Chapter 2 - Background.......................................................................................................... 5

Chapter 3 - Development....................................................................................................... 8

3.1 - Step 1: Determining Design Requirements ................................................................. 8

3.2 - Step 2: Determining Technical Requirements ............................................................. 9

3.3 - Step 3: Design Operation Flow .................................................................................... 12

3.4 - Step 4: Design Individual Components ........................................................................ 13

3.5 - Step 5: Initial Development of Composite Device ....................................................... 19

3.6 - Step 6: Further Development of Composite Device .................................................... 22

3.7 - Step 7: Final Additions to Composite Device .............................................................. 28

Chapter 4 - Testing.................................................................................................................. 29

4.1 - Testing Requirement 1: Basic Data Collection ............................................................. 29

4.2 - Testing Requirement 2 – Operation Longevity .............................................................. 34

4.3 - Testing Requirement 3: Bluetooth Communication Integrity ...................................... 38

4.4 - Testing Requirement 4: Comfort Level ......................................................................... 39

4.5 - Results ............................................................................................................................ 39

Chapter 5 - Conclusion........................................................................................................... 40

Chapter 6 - Appendices.......................................................................................................... 41

Appendix A – Bluetooth Code (Functional Test) .................................................................. 41

Appendix B – Color Sensor Code (Functional Test) .............................................................. 45

Appendix C – Version 1 Code (First iteration code for ChromaBand) ..................................... 47

Appendix D – Version 2 Code (Second iteration code for ChromaBand) ............................... 50

Appendix E – ChromaBand Schematic ............................................................................... 54

Appendix F – Serial Bluetooth Terminal Application .......................................................... 54

Appendix G – Software Flowchart ....................................................................................... 55

Appendix H – Setting up Arduino IDE and Software Libraries ............................................. 56

Appendix I – Uploading Software to the ChromaBand .......................................................... 57

Appendix J – General User Manual ....................................................................................... 57

Appendix K – Installing Serial Bluetooth Terminal Application ........................................... 58

Appendix L – ChromaBand Price Evaluation ....................................................................... 58
Chapter 0 - Abstract

It’s long been said that your skin is a window into your health and with many illnesses the first signs of trouble actually show up in your skin. So if we have this natural warning system, then why isn’t anyone able to effectively use it to predict changes in our health? The problem is that currently there is no reliable way to accurately measure the change in skin tone and how these changes may or may not change with variations in health. This project’s aim is to design and develop a device that will record the changes in a user’s skin tone and log that data through a smartphone application using a low energy Bluetooth protocol.
Chapter 1 - Introduction

A person’s skin tone can change drastically throughout the day. Thus far, there isn’t any user-friendly and reliable way of determining and detailing concrete skin tone changes. Up until now most changes have only been noticed by people. They realized that their skin tone generally seems whiter during the mornings, rosier during the evenings, greener when they’re ill and paler/rosier when they’re nervous. These changes often occur with big changes in health and can even show up before any obvious health changes are evident. In many cases skin can show signs on a serious internal disease long before they advance and actually develop into a serious ailment. Skin tone changes can also change with current moods and situations. For example skin tone can become red upon experiencing embarrassment and during increased nervousness. Skin tone becomes extremely flushed out and pale upon episodes of panic and shock. Skin tone may also become jaundiced (yellowish in complexion) upon liver disease or kidney failure. It’s obvious that a major portion of human health conditions are reflected through our skin but it’s not something that has been proven beyond a reasonable doubt and it is with this concept in mind that Electrical Engineering Professor Tina Harriet Smilkstein proposed the main problem statement for this project.

“Currently, there is no reliable way to accurately measure the change in skin tone and how these changes may or may not change with variations in health.”

With the increasing popularity of wearable digital technology Professor Smilkstein proposed the creation of a small wearable device to sample skin tone data from hour to hour and day to day. The latest trend with wearable technology has peaked rising interest in health apps, heart rate monitors, and sleep tracking software. The problem is that quite often those devices are large, expensive, ugly, and people tend to forget/hate to wear them when they get up in the morning. Thus, she proposed that the color tracking device be outfitted so that it could be attached to underwear so that it would never be forgotten. The basic idea is that everyone wears underwear (or at least should) and that if the device is outfitted for undergarments it will never be forgotten and will also be small enough that the user will never think twice about it.

This senior project’s goal is to design and develop a device that records skin tone data at regular intervals throughout the day and then send that data to an application on a smart phone for it to be logged and saved for later use. The analyzation and use of the data is beyond the scope of this project and will be a future extension.
Chapter 2 - Background

With regard to skin tone changes, one of the main principles behind our skin reacting to health and bodily changes is the principle of Photoplethysmography, more commonly known as a PPG. Photoplethysmography is the idea of taking volumetric measurements of our body by using a light source to illuminate the skin and then measuring the amount of light absorbed and how that absorption changes over time. It’s the basic idea behind how pulse oximeters (e.g. finger clamping heart rate sensors) measure your heart rate. It sends light through the skin into your finger and the existing blood will absorb a certain amount of light. As blood enters and exits the veins and capillaries in your finger that light absorption changes and that data can be extrapolated into a heart rate. The same effect happens in our innermost and outermost layers of skin; changes in our body and state directly influence how much blood and how quickly that blood runs throughout our skin. These changes can thus be tracked by the skin tracking device and later analyzed with any noted changes in the user’s health and mood. One of the advantages of noting minute color changes in skin is that with the right algorithm the collected data could be able to tell us about what our body experiences without us ever knowing what is going on during the day.

With regard to wearable digital technology, they are electronic technologies that are incorporated into items of clothing and accessories which can be worn on the body. Wearables, as they’re called, could have a large and everlasting influence on health and medicine because it could potentially transform many complex medical instruments into miniature sized devices that we can utilize everyday without ever being bothered by a jungle of wires and adhesive pads. In fact a small collection of wearables were already being used in the medical field a few years prior to them being released on the commercial market. The ease of data capturing and portability of wearables yielded themselves as the perfect solution to medical professionals who were attempting to track how new medicine affected/helped people in clinical trials. These wearable monitors can endure day to day activity and normal wear and tear much like a shirt or a pair of jeans. This ease of use coupled with advancements in computing has led wearables to become the next big thing in the world of digital technology. Current advancements have lead to companies choosing different styles and placements for wearable technology as only so much information can be captured from a wrist worn smartwatch. The device for color tracking for example will be worn on the waistband of undergarments.

The device is made up of a list of components, including a battery, battery charger, microcontroller, color sensor, and accompanying smartphone application for use sending data via 2.4 GHz Bluetooth Low Energy (BTLE).

The most vital component is the color sensor (in our case it’s a complex photodiode), which takes the actual readings of skin tone. The sensor first turns on a light source (in our case, it’s a light emitting diode, LED) and illuminates the area for color readings to be taken. The color sensor records and calculates a couple of important data values of which include, Red Filtered, Green Filtered, Blue Filtered, Clear Unfiltered, Color Temperature, and Luminosity. Refer to Figure 1.1 while reading further for clarification. The first three filtered values make up the general RGB values many people have come to know on the scale of 0 to 255 with Red as
(255, 0, 0), Green as (0, 255, 0), and Blue as (0, 0, 255). Further combinations of the three values make up the generic spectrum of the visible light spectrum. This 0 to 255 range is just a generic scale and in the readings taken by the color sensor values can grow larger and are actually in reference to 65536 instead of 255. So instead of using 0 to 255 we are effectively using 0 to 65536 which allows for increased precision per change in color and allows for more accurate data analysis later on. The clear unfiltered value is the value that is recorded initially from the skin tone before any red, green, or blue filters are used. Luminosity is recorded as the intensity of the reflection from the area that is illuminated by the LED. Color Temperature is a measure of the warmness or coolness of a color mix and is often used in TVs and monitors to adjust the preferred quality that an image conveys. It is measured in Kelvin and is based off heating a piece of metal (more ideally, we would use a blackbody radiator) and as it gets hotter it will begin to turn red and graduates to orange, yellow, white, blue, and further shades of blue. This effect can be seen in Figure 1.2.

![Color Sensor (Photodiode) Light Processing Diagram](image-url)

**Figure 1.1: Color Sensor (Photodiode) Light Processing Diagram**
The device heavily relies on a central system to operate the color sensor and redirect the data to be sent to a smartphone application. This is where the brains (in our case, it’s a microcontroller) comes in and must direct the flow of operation for the device. The microcontroller directs the color sensor to capture skin tone data values and then takes that data and sends it to the smartphone application via the onboard Bluetooth module. The most important use of the microcontroller in this project is that it supports functionality of sending data wirelessly to an external device via a safe, secure, and low power Bluetooth protocol. The 2.4GHz Bluetooth Low Energy protocol is extremely efficient and resilient and this helps us save on power when operating off of a lithium ion battery. The Bluetooth runs off of a command set called AT which is short for attention. Each command begins with AT+______, and the blank will have specific characters depending on what it is you are trying to do. One command will be extremely helpful and that is the AT+BLEUARTTX command which when used will transmit anything following the command and will terminate upon hitting a new line character (either, \n or \r). These new line characters (\n for newline and \r for carriage return) are the equivalent for hitting the enter key on a keyboard. So the AT+BLEUARTTX command is essentially completed upon hitting the enter key. The AT+BLEUARTTX command will be utilized over and over again when we capture data from the sensor and send it through the microcontroller’s Bluetooth module. A sample of what a data packet from microcontroller to phone application might look like is shown in Figure 1.3. The smartphone application works off a simple transmit and receive (TX/RX) style communication protocol, hence the TX in the AT+BLEUARTTX command on the microcontroller side.

<table>
<thead>
<tr>
<th>AT+BLEUARTTX</th>
<th>COLOR DATA</th>
<th>\n OR \r OR any newline character</th>
</tr>
</thead>
</table>

- Continuing from here the skin tone sampling device will be referred to as the ChromaBand.
- Reading, sampling, taking, measuring, tracking, recording, and sensing are used interchangeably when referring to the color sensor taking data from skin tone.
Chapter 3 - Development

The process of designing and developing a skin tone tracking and recording device, called the ChromaBand, was broken into the following list of steps:

1. Determine required components for ChromaBand to function on a basic level
2. Determine technical requirements for each component to have for basic functionality
3. Design operation flow for device and create initial operation flowchart
4. Design and develop each component to perform its own task correctly
5. Complete initial development with all components together and test version 1
   a. Version 1 consists of simply sensing skin tone and printing it to the serial monitor while microcontroller is still wired to a computer and terminal
6. Develop version 1 further until wireless communication is achieved in version 2
   a. Version 2 consists of sensing skin tone and sending it through the microcontroller’s Bluetooth module to the application on a smartphone.
   b. There will not be a change in hardware between version 1 and version 2
   c. There will be a change in software between version 1 and version 2
7. Add final touches for the ChromaBand to be easily wearable, concealed, and operated

3.1 - Step 1: Determining Design Requirements

For the ChromaBand to work as desired it needs to be able to sample skin tone on a regular interval for a given amount of time (e.g. taking skin data for 10 seconds, every half hour), last a full workday (at least 8 hours of operation), and be able to send that data to a smartphone application for user’s to see the data and save for later analyzation. Initial brainstorming for creating the ChromaBand led to a couple basic requirements of which include, a central computer to direct flow of data and to command sensors to operate, a color sensor that will do the actual color sampling of skin tone, a rechargeable battery for continued and efficient power, and some form of charger for that rechargeable battery. Listed below is a summary of functionality requirements and basic components required for the ChromaBand.

- Requirements for Successful ChromaBand Operation
  1. Be able to sample skin tone data for a given amount of time on a regular interval of time, such as taking skin tone data for 10 seconds, every half hour
  2. Be able to operate continuously for at least a full workday, so at least 8 hours
  3. Be able to send the skin tone data wirelessly to a smartphone application
  4. Be low profile, comfortable, and almost unnoticeable to the user

- List of Components Needed
  1. Central computer to direct operation (most likely a microcontroller)
  2. Color sensor for skin tone sampling
  3. Rechargeable battery (most likely a lithium ion battery)
  4. Battery charger
3.2 - Step 2: Determining Technical Requirements

For each of the four components listed in Part 1, technical requirements must be determined and matched so that the components will work successfully with each other.

For the color sensor there were a couple of readily available sensors on the market. These included two sensors from Adafruit.com and two from Sparkfun.com. The two Adafruit sensors can be seen in Figures 3.2.1 and 3.2.2, while the two from Sparkfun can be seen in Figures 3.2.3 and 3.2.4. The two from Adafruit use the exact same color filtering process because they use the same TCS34725 photodiode chip. The only difference between these two is breakout board they are mounted onto. The two from Sparkfun are vastly different as they use different photodiode chips and one of the sensors also has an additional, but not needed, gesture sensor. The two sensors from Sparkfun were eliminated because they were more expensive and also didn’t have an onboard LED to illuminate the skin (the color sensor will be on the inside of the waist and will be in a dark environment). The LED was paramount to the sensor collecting accurate data as the environment in which the sensor would be situated would be lacking of light. Thus the final decision was between two Adafruit sensors and the normal color sensor in Figure 3.2.1 was chosen for ease of wire redirection on the board. It could be easily replaced with the Flora sensor from Figure 3.2.2 and the only difference would be the shape, size, and wire placement as the traces for the Flora are placed around the entire board while the normal sensor has all the traces are all on one edge of the board. The color sensor operates off a 3.3V supply and communicates using an I²C Serial communication protocol.

Figure 3.2.1: (LEFT) RGB Color Sensor with IR filter and White LED – TCS34725
Figure 3.2.2: (RIGHT) Flora Color Sensor with IR filter and White LED – TCS34725
For the central computer it was determined that an Arduino compatible microcontroller was most reasonable. Within the Arduino compatible family of microcontrollers there exists a product line named the Feather series that is both lightweight, low profile, and powerful. Researching the Feather series lineup, I came upon the Adafruit Feather 32u4 Bluefruit LE microcontroller. This microcontroller included 8 digital pins, 10 analog pins, a powerful 8.8 MHz ATmega32u4 processor, I^2C serial communication for sensors/peripherals, and most importantly a low energy Bluetooth module for wireless communication with either Apple’s iOS or Google’s Android operating systems. For development an Android platform will be used. Another option was the Adafruit Feather M0 Bluefruit LE microcontroller which boasts a more powerful and faster 48 MHz ATSAMD21G18 processor, which at the time of purchasing was more expensive than the 32u4 so the final decision was to use the 32u4 microcontroller. The microcontroller can communicate with the color sensor via the I^2C serial communication protocol and also has a 2 pin JST connector for an external power supply. It operates on a 3.3V logic making it perfect to power the color sensor indirectly from an external power supply. Most importantly it has an nRF51822 Bluetooth LE chip from Nordic Semiconductors. This Bluetooth module will allow for the wireless transfer of collected data from the device to an external smartphone application. The Adafruit Feather 32u4 Bluefruit LE microcontroller is shown in Figure 3.2.5 below and the Adafruit Feather M0 Bluefruit LE microcontroller is shown in Figure 3.2.6. They look very similar as the only difference between them is the main processor chip.
Figure 3.2.5: (LEFT) Adafruit Feather 32u4 Bluefruit LE microcontroller

Figure 3.2.6: (RIGHT) Adafruit Feather M0 Bluefruit LE microcontroller

For the battery and battery charger it was required that the battery and battery charger support the JST connector that the microcontroller from Figure 3.2.5 had equipped. One such battery was the Adafruit Lithium Ion Polymer Battery that usually operates at 3.7 volts for 1200 milliamp hours or mAh. These lithium ion polymer batteries are much like the batteries that power your cellular devices today. The battery is shown in Figure 3.2.7 below. The battery charger chosen was the Adafruit Micro Lipo Battery with MicroUSB Jack Charger. It charges lithium ion polymer batteries like the one in Figure 3.2.7 at a rate of 100 mA or at 500 mA depending on user preference. It can charge a connected lithium ion polymer battery via a 5 volt MicroUSB source, perhaps from a computer. The charger will display a red light during charging, green light when fully charged, and both red and green when no battery is connected (high impedance case). The battery charger is shown in Figure 3.2.8 below. The best attribute of the battery charger is that when a battery is connected, its voltage output can be extracted from the GND and BAT pins shown onboard. Due to this, the charger can be placed between the battery and the microcontroller and will thus deliver power and charge the battery together. In the final design it can be regarded as part of the microcontroller and not a discrete component. It will then serve as the battery charging portion of the microcontroller.
3.3 - Step 3: Design Operation Flow

This step is a top level look at how the device should operate. For the initial flow of the process the proposed plan is a general overview with a high level view and forgoes any technical vernacular. This would be a flowchart that any average consumer would understand assuming they know why they are using this device. The high level process is shown in Figure 3.3.1 below.

Figure 3.3.1: High Level Overview of Device Operation Process
3.4 - Step 4: Design Individual Components

For developing the microcontroller part the first goal was to get basic Bluetooth transmit and receive communication working. This was done by connecting the microcontroller to a computer and running the Arduino IDE with some basic code. The basic code is simply a messaging system where the serial monitor (on the microcontroller end) could send messages to the application. Then from the application (smartphone end) it could receive and send back messages. A preexisting Bluetooth app called nRF UART 2.0 (by Nordic Semiconductors to accompany their nRF51822 chip) was used to open communication with the microcontroller. This step along with some sent and received messages is shown in Figure 3.4.1 below.

Figure 3.4.1: Communication between Microcontroller and Application via Bluetooth

Once this messaging system worked then the focus shifted to the microcontroller simply sending data and the application receiving. For the purposes of this project the application never actually needs to send anything back, it just needs to worry about receiving data from the microcontroller and saving it. The basic messaging system code was modified so that the microcontroller only sends incrementing numbers to the application. The software relies on the AT command protocol and more specifically the AT+BLEUARTTX command. This code was changed so that the AT+BLEUARTTX was immediately followed by an incrementing integer value. This was sent through the Bluetooth module as a transmit command with a payload of data to be transmitted. This code snippet is shown in Figure 3.4.2 and the data packet sent is shown in Figure 3.4.3. After uploading the code the microcontroller was disconnected and given a power supply (the lithium ion battery). The microcontroller pairing and transmitting data stages are shown in Figures 3.4.4 and 3.4.5 respectively.
1. `void loop(void)
2. {
3.   int x;
4.   while(1){
5.     ble.print("AT+BLEUARTTX=");
6.     ble.print(x);
7.     ble.println("\r\n");
8.     x++;
9.     delay(1000);
10.    if(!ble.waitForOK()){
11.      Serial.println(F("Failed to send??"));
12.    }
13.  }
14. }

Figure 3.4.2: Incrementing Value Code Snippet Sends Incrementing Value Every Second

<table>
<thead>
<tr>
<th>AT+BLEUARTTX</th>
<th>INCREMENTING VALUE</th>
<th>\n OR \r OR any newline character</th>
</tr>
</thead>
</table>

Figure 3.4.3: Data Packet, with Transmit Command and Payload

Figure 3.4.4: (LEFT) Microcontroller with Lithium Ion Battery Pairing with Application
Figure 3.4.5: (RIGHT) Microcontroller Transmitting Incrementing Values to Application
Once the microcontroller Bluetooth modules transmit/receive functionality worked correctly the focus shifted to getting the color sensor working correctly. The first part of setting up the color sensor was wiring the sensor to the microcontroller. In an effort to avoid needless soldering electrical grabbers were utilized for temporary and solder free connections. The connection is set up as shown in Figure 3.4.6. The pins on the microcontroller are not all on one edge, it is simply drawn that for clarity. The red wire connects the microcontroller’s 3 volt supply (coming from attached lithium ion battery) to the V_{IN} pin of the color sensor. The reference on both boards, GND, is connected via the black wire. Together these two wires deliver the power needed for the color sensor to operate. The I^2C serial communication pins SCL and SDA are connected to the respective SCL and SDA on the other board. SCL allows for synchronized timing in order to correctly transmit and receive data packets. SDA allows a highway for the data packets to actually be sent from one board to the other and vice versa. Wiring INT and LED together allows for the software to send an interrupt signal allowing the LED to be turned on or off. This LED interrupt command is sent through SDA with the aid of SCL. The physical connection setup is shown in Figure 3.4.7 below.

After the color sensor was wired to the microcontroller code was developed for the sensor to continuously capture data and print it out on the serial monitor on the connected computer. This developed code commands the color sensor to capture values for Clear, Red, Green, and Blue filters for 100 times resulting in 100 data sets. Then it calculates the color temperature and reflected luminosity of the area. The code Snippet in Figure 3.4.8 shows the initialization for a specific configuration. The name of the sensor is dubbed “tcs” and it is initialized with a 101 millisecond integration time and a gain of 60. The integration time controls how long the sensor takes to sample and produce a single data reading. So it reads for 101 milliseconds and produces the values requested. The gain of 60 is used to boost the value and allow for more precision with larger values. The code snippet for a function that performs these calculations and prints it to a serial monitor is shown in Figure 3.4.9 below. Figure 3.4.10 shows the loop that turns on the LED and calls the captureColorData function 100 times and with a 101ms integration time resulting in a 10.1 second long overall reading. It finishes by turning off the LED and waiting for 5 seconds before capturing another 100 data sets.
/* hardware SPI, use SCK/MOSI/MISO pins, then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

void captureColorData(){
    uint16_t r, g, b, c, colorTemp, lux;
    tcs.getRawData(&r, &g, &b, &c);
    colorTemp = tcs.calculateColorTemperature(r, g, b);
    lux = tcs.calculateLux(r, g, b);
    Serial.print("Color Temp: "); Serial.print(colorTemp, DEC); Serial.print("K ");
    Serial.print("Lux: "); Serial.print(lux, DEC); Serial.print(" ");
    Serial.print("R:"); Serial.print(r, DEC); Serial.print(" ");
    Serial.print("G:"); Serial.print(g, DEC); Serial.print(" ");
    Serial.print("B:"); Serial.print(b, DEC); Serial.print(" ");
    Serial.print("C: "); Serial.print(c, DEC); Serial.print(" ");
    Serial.println();
}
```c
1. void loop(void) {
2.     int i;
3. 
4.     /* turn on color sensor LED */
5.     tcs.setInterrupt(false);
6. 
7.     /* take 100 data sets */
8.     for(i = 0; i < 100; i++){
9.         captureColorData();
10.     }
11.     Serial.println();
12. 
13.     /* turn off color sensor LED */
14.     tcs.setInterrupt(true);
15. 
16.     /* wait for 5 seconds and then loop over taking data again */
17.     delay(5000);
18. }
```

Figure 3.4.10: Loop, Calls Color Sensor Function 100 times, Wait 5 Seconds, Repeat

After uploading the code and starting the serial monitor on the Arduino IDE the color sensor will endlessly capture 100 data sets/10 seconds worth of data and print to the serial monitor as it reads each set. In this case a data set consists of the 5 color values red, green, blue, clear, color temperature, and luminosity. This physical setup is shown in Figure 3.4.11 below.

Figure 3.4.11: Physical Setup of Data Capturing Process with Serial Monitor Output on Right
Moving on to the battery and battery charger assembly, the battery was connected to the charger via the two pin JST connector and grabbers were used to connect GND/BAT from the charger to the 2 pin JST connector on the microcontroller. The schematic for the connection layout is shown in Figure 3.4.12 below.

![Battery to Charger to Microcontroller Wiring Schematic](image)

**Figure 3.4.12: Battery to Charger to Microcontroller Wiring Schematic**

The above setup allows for the battery to still deliver power to the microcontroller via the GND/BAT pins while at the same time allowing for a port to charge the battery in case. For ease of use, a switch should also be placed in line with the red wire between the charger and the microcontroller. Figure 3.4.13 below shows on the left the initial wiring and on the right the inclusion of a switch between the charger and microcontroller. Figure 3.4.14 shows the finished assembly with switch included in wiring.

![Battery to Charger to Microcontroller Wiring Schematic](image)

**Figure 3.4.13: (LEFT) Initial Wiring and (RIGHT) Inclusion of Switch in Wiring**
3.5 - Step 5: Initial Development of Composite Device

Now that all individual component development has been done it was time to assemble them all together and begin soldering together components for good. Brainstorming about the design led to a clip on style device. It would clip on top of the undergarment waistband. The color sensor sits on the inside of the undergarment and should make contact with the skin on a user’s waist. The microcontroller, charger, and battery would then sit on the outside of the undergarment facing away from the body. The connecting wires between the microcontroller and color sensor would then sit in top of the undergarment. This setup is shown in Figure 3.5.1 below. The microcontroller soldered to the color sensor in the clip on style is shown in Figure 3.5.2. Version 1 code that senses only red, green, and blue was then uploaded to the ChromaBand for initial testing. This code serves as a preliminary proof of concept for the ChromaBand. When paired and equipped with a new smartphone application, “Serial Bluetooth Terminal”, the ChromaBand behaved as desired by taking a 10 second reading (100 data sets), sending that data to the application, and then waiting 5 seconds before taking and sending another 100 data sets.
Figure 3.5.1: Schematic of Overall Clip-On Style Design

Figure 3.5.2: Two Aspects of Microcontroller Soldered to Color Sensor in Clip-On Style
Figure 3.5.3a: Completed Version 1 ChromaBand in Connected Mode

Figure 3.5.3b: Completed Version 1 ChromaBand

Figure 3.5.3c: Completed Version 1 ChromaBand

Figure 3.5.3d: Completed Version 1 ChromaBand in Charging Mode
3.6 - Step 6: Further Development of Composite Device

Leaving the hardware alone until further final touches were added, the focus was shifted to further improvement of the software for the ChromaBand. This second version of the software added the two other color values (Color Temperature and Luminosity), battery level readings and charge warnings, and formatting for ease of data extraction. The battery level readings are based off of values read from the microcontroller ranging from right below 5 volts to 4.2 to 3.7 and finally to 3.2. At around 3.7 volts the device will warn the user that the battery is low and should be charged soon. At around 3.3 volts the device will warn the user that the battery is critically low and should be charged immediately. The code for this battery management is shown in Figure 3.6.1 below.

```c
/* print battery level and charge reminders */

void printBatteryStatus(){
  float measuredvbat;
  measuredvbat = analogRead(VBATPIN);
  measuredvbat *= 2; // 22uF capacitor for double 100k resistor in microcontroller
  measuredvbat *= 3.3; // multiply by 3.3V, the reference voltage
  measuredvbat /= 1024; // convert to voltage
  ble.print("AT+BLEUARTTX=");
  ble.print("Battery Voltage: ");
  ble.print(measuredvbat);
  if(measuredvbat < 3.3){
    ble.print("(Critical) . . . Charge Now!");
  } else if(measuredvbat < 3.7){
    ble.print("(Low) . . . Charge Soon!");
  } else if(measuredvbat < 4.2){
    ble.print("(Good) ");
  } else if(measuredvbat < 5.0){
    ble.print("(Full) ");
  }
  ble.println("\r\n");
}
```

Figure 3.6.1: Function to Check Battery Status and Warn User
The additional formatting added is meant to separate data sets (100 sets at a time) as well as formatting the data into different columns for clarity. The code to format data and divide it is shown in Figure 3.6.2 below. The result of this in a text file is shown in Figure 3.6.3 below.

```c
1. /* print a divider for organization and 23luttoo sake */
2. void printDivider()
3. { /* print divider */
4.   ble.print("AT+BLEUARTTX=");
5.   ble.print("=================================
===========================================
=======================");
6.   ble.println("\r\n");
7. }
8.
9. /* print some verbose 23luttoo23tion */
10. void printInitialHeader()
11. { ble.print("AT+BLEUARTTX=");
12.   ble.print("Starting Skin Data Sampling");
13.   ble.println("\r\n");
14. }
15.
16. /* helper function to print columns header */
17. void printDataHeader()
18. { ble.print("AT+BLEUARTTX=");
19.   ble.print("| Color Temp (K) | Red Value | Green Value | Blue Value | Clear Value | Luminosity |");
20.   ble.println("\r\n");
21. }
```

**Figure 3.6.2: Functions Dividing Data, Starting Data Announcement, Different Data Columns**

<table>
<thead>
<tr>
<th>13:45:46</th>
<th>Color Temp (K)</th>
<th>Red Value</th>
<th>Green Value</th>
<th>Blue Value</th>
<th>Clear Value</th>
<th>Luminosity</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:45:46</td>
<td>18683</td>
<td>227</td>
<td>389</td>
<td>438</td>
<td>1014</td>
<td>235</td>
</tr>
<tr>
<td>13:45:47</td>
<td>18683</td>
<td>227</td>
<td>399</td>
<td>438</td>
<td>1014</td>
<td>235</td>
</tr>
<tr>
<td>13:45:47</td>
<td>18683</td>
<td>227</td>
<td>399</td>
<td>438</td>
<td>1014</td>
<td>235</td>
</tr>
<tr>
<td>13:45:47</td>
<td>18683</td>
<td>227</td>
<td>399</td>
<td>438</td>
<td>1014</td>
<td>235</td>
</tr>
</tbody>
</table>

**Figure 3.6.3: Sample Output Showing Formatting in Text File**

13:45:57 Battery Voltage: 4.54 (Full)
The color sensor function was updated to add the inclusion of Color Temperature and Luminosity. These changes are shown in the new function in Figure 3.6.4 below.

```c
1. /* captures color data and sends through 24I Bluetooth module */
2. void captureColorData(){
3.  /* variables for holding values of color data */
4.   uint16_t r;
5.   uint16_t g;
6.   uint16_t b;
7.   uint16_t c;
8.   uint16_t colorTemp;
9.   uint16_t lux;
10. 
11.  /* take a reading from sensor, and calculate color temperature from r, g, and b */
12.  tcs.getRawData(&r, &g, &b, &c);
13.  colorTemp = tcs.calculateColorTemperature(r, g, b);
14.  lux = tcs.calculateLux(r, g, b);
15. 
16.  /* send values through 24I Bluetooth module */
17.  ble.print("AT+BLEUARTTX=");
18.  ble.print("t"); ble.print(colorTemp, DEC); ble.print("t");
19.  ble.print("t"); ble.print(r, DEC); ble.print("t");
20.  ble.print("t"); ble.print(g, DEC); ble.print("t");
21.  ble.print("t"); ble.print(b, DEC); ble.print("t");
22.  ble.print("t"); ble.print(c, DEC); ble.print("t");
23.  ble.print("t"); ble.print(lux, DEC); ble.print("t");
24.  ble.println("\r\n");
25. 
26.  /* wait for delivery ack */
27.  if(!ble.waitForOK()) {
28.    Serial.println(F("Failed to send?"));
29.  }
30. }
```

Figure 3.6.4: Updated Color Sensor Function
The original overall loop must be updated for these changes to be reflected. It checks if a global variable “initialReading” is set to 1 or not. If it is then wait two seconds for initial communication setup to settle, then print the “Starting Skin Data Sampling” announcement. Setting the “initialReading” variable to 0 ensures that we will not enter this if conditional again. Next we print the different data columns header and then turn on the LED for sampling. Continue on to taking 100 data sets and send to application at the same time. Then turn off the LED, print a line divider, and report battery status at the end. Finally, wait 30 minutes before taking the next reading. The new loop that runs this process is shown in Figure 3.6.5 below.

```c
/* main process loop */
void loop(void)
{
  int I;

  /* on setup delay a little for communications to settle */
  if(initialReading == 1){
    delay(2000);

    /* print initial header */
    printDivider();
    printInitialHeader();
    printDivider();

    /* we only need to enter this once */
    initialReading = 0;
  }

  /* print header for different values columns */
  printDataHeader();

  /* turn on color sensor led */
  tcs.setInterrupt(false);

  /* collect 100 data points at 101ms each so total of 10 second reading */
  for(I = 0; I < 100; i++){
    captureColorData();
  }

  /* turn off color sensor led */
  tcs.setInterrupt(true);

  /* print battery status and whetherr you should charge soon or not */
  printDivider();
  printBatteryStatus();
  printDivider();

  /* delay 30 minutes */
  for(I = 0; I < 60; i++){
    delay(30000);
  }
}
```

**Figure 3.6.5: Main Process Controlling Loop**
An output text file that the application produces when saving it is shown in Figure 3.6.6 below. This file does not contain all 100 data sets due to trying to fit data into a single page. The timestamps on the left are in the format of HH:MM:SS so the data collection started at 14:30:50.

<table>
<thead>
<tr>
<th>Time</th>
<th>Color Temp (K)</th>
<th>Red Value</th>
<th>Green Value</th>
<th>Blue Value</th>
<th>Clear Value</th>
<th>Luminosity</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:30:49</td>
<td>3328</td>
<td>1214</td>
<td>898</td>
<td>733</td>
<td>2688</td>
<td>344</td>
</tr>
<tr>
<td>14:30:50</td>
<td>2627</td>
<td>1482</td>
<td>1007</td>
<td>928</td>
<td>3249</td>
<td>429</td>
</tr>
<tr>
<td>14:30:50</td>
<td>3061</td>
<td>1634</td>
<td>1843</td>
<td>960</td>
<td>3484</td>
<td>413</td>
</tr>
<tr>
<td>14:30:50</td>
<td>3088</td>
<td>2073</td>
<td>796</td>
<td>610</td>
<td>2644</td>
<td>157</td>
</tr>
<tr>
<td>14:30:50</td>
<td>1636</td>
<td>1586</td>
<td>773</td>
<td>661</td>
<td>2788</td>
<td>221</td>
</tr>
<tr>
<td>14:30:50</td>
<td>3761</td>
<td>859</td>
<td>587</td>
<td>546</td>
<td>1896</td>
<td>247</td>
</tr>
<tr>
<td>14:30:50</td>
<td>13125</td>
<td>84</td>
<td>69</td>
<td>77</td>
<td>217</td>
<td>25</td>
</tr>
<tr>
<td>14:30:50</td>
<td>11111</td>
<td>76</td>
<td>59</td>
<td>65</td>
<td>185</td>
<td>21</td>
</tr>
<tr>
<td>14:30:50</td>
<td>6387</td>
<td>70</td>
<td>50</td>
<td>52</td>
<td>157</td>
<td>18</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:57</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23699</td>
<td>94</td>
<td>95</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>16236</td>
<td>94</td>
<td>94</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>26236</td>
<td>94</td>
<td>94</td>
<td>111</td>
<td>279</td>
<td>38</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23625</td>
<td>94</td>
<td>94</td>
<td>110</td>
<td>278</td>
<td>37</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23625</td>
<td>94</td>
<td>94</td>
<td>110</td>
<td>278</td>
<td>37</td>
</tr>
<tr>
<td>14:30:58</td>
<td>23625</td>
<td>94</td>
<td>94</td>
<td>110</td>
<td>278</td>
<td>37</td>
</tr>
<tr>
<td>14:30:58</td>
<td>9141</td>
<td>81</td>
<td>66</td>
<td>78</td>
<td>199</td>
<td>26</td>
</tr>
<tr>
<td>14:30:59</td>
<td>3855</td>
<td>688</td>
<td>483</td>
<td>437</td>
<td>1522</td>
<td>221</td>
</tr>
<tr>
<td>14:30:59</td>
<td>1641</td>
<td>3245</td>
<td>1571</td>
<td>1315</td>
<td>5769</td>
<td>463</td>
</tr>
<tr>
<td>14:30:59</td>
<td>2183</td>
<td>3998</td>
<td>1767</td>
<td>1474</td>
<td>6762</td>
<td>415</td>
</tr>
<tr>
<td>14:30:59</td>
<td>1697</td>
<td>3725</td>
<td>1756</td>
<td>1440</td>
<td>6446</td>
<td>588</td>
</tr>
<tr>
<td>14:30:59</td>
<td>1889</td>
<td>4115</td>
<td>1867</td>
<td>1522</td>
<td>6968</td>
<td>496</td>
</tr>
<tr>
<td>14:30:59</td>
<td>1779</td>
<td>4377</td>
<td>2089</td>
<td>1682</td>
<td>7416</td>
<td>577</td>
</tr>
<tr>
<td>14:30:59</td>
<td>1673</td>
<td>4548</td>
<td>2144</td>
<td>1674</td>
<td>7779</td>
<td>682</td>
</tr>
<tr>
<td>14:30:59</td>
<td>1632</td>
<td>4642</td>
<td>2238</td>
<td>1722</td>
<td>7999</td>
<td>764</td>
</tr>
<tr>
<td>14:31:00</td>
<td>1623</td>
<td>4676</td>
<td>2277</td>
<td>1743</td>
<td>8092</td>
<td>889</td>
</tr>
<tr>
<td>14:31:00</td>
<td>1630</td>
<td>4699</td>
<td>2309</td>
<td>1760</td>
<td>8165</td>
<td>839</td>
</tr>
<tr>
<td>14:31:00</td>
<td>1630</td>
<td>4715</td>
<td>2332</td>
<td>1787</td>
<td>8238</td>
<td>842</td>
</tr>
<tr>
<td>14:31:00</td>
<td>1630</td>
<td>4771</td>
<td>2382</td>
<td>1801</td>
<td>8346</td>
<td>892</td>
</tr>
<tr>
<td>14:31:00</td>
<td>1627</td>
<td>4776</td>
<td>2398</td>
<td>1899</td>
<td>8378</td>
<td>910</td>
</tr>
<tr>
<td>14:31:00</td>
<td>2174</td>
<td>3525</td>
<td>2188</td>
<td>1672</td>
<td>6981</td>
<td>957</td>
</tr>
<tr>
<td>14:31:00</td>
<td>6270</td>
<td>288</td>
<td>185</td>
<td>174</td>
<td>546</td>
<td>97</td>
</tr>
<tr>
<td>14:31:00</td>
<td>6061</td>
<td>412</td>
<td>372</td>
<td>344</td>
<td>1093</td>
<td>261</td>
</tr>
<tr>
<td>14:31:00</td>
<td>6546</td>
<td>1231</td>
<td>1158</td>
<td>1000</td>
<td>3365</td>
<td>637</td>
</tr>
<tr>
<td>14:31:01</td>
<td>3864</td>
<td>390</td>
<td>263</td>
<td>249</td>
<td>851</td>
<td>186</td>
</tr>
<tr>
<td>14:31:01</td>
<td>2964</td>
<td>286</td>
<td>175</td>
<td>169</td>
<td>585</td>
<td>59</td>
</tr>
<tr>
<td>14:31:01</td>
<td>3820</td>
<td>264</td>
<td>162</td>
<td>157</td>
<td>542</td>
<td>55</td>
</tr>
<tr>
<td>14:31:01</td>
<td>3063</td>
<td>322</td>
<td>199</td>
<td>192</td>
<td>666</td>
<td>69</td>
</tr>
</tbody>
</table>

14:31:01 Battery Voltage: 4.56 (Full)

---

Figure 3.6.6: Sample Output Text File from Serial Bluetooth Terminal Application
The software flowchart for the ChromaBand is shown in Figure 3.6.7 below. Completely powering off either device or un-pairing the Bluetooth connection at any time breaks the process.
3.7 - Step 7: Final Additions to Composite Device

Final touches for the ChromaBand include adding an elastic band for the microcontroller, charger, and battery and adding Velcro for the ChromaBand to attach to cloth without falling off. The Velcro is semi destructive to some cloths but is generally very well fastening but easily removable. Final touches to the ChromaBand are shown in the figures below.

![Figure 3.7.1: Semi Completed ChromaBand](image1)

![Figure 3.7.2: Fully Completed ChromaBand with Elastic and Velcro](image2)

![Figure 3.7.3: Fully Completed ChromaBand with Elastic and Velcro (Rear View)](image3)
Chapter 4 - Testing

To gauge the success of the ChromaBand it must be determined how many original requirements the final device meets. The original requirements, noted in Step 1 of the Development Section, include:

- Requirements for Successful ChromaBand Operation

<table>
<thead>
<tr>
<th>Requirement #</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Be able to sample skin tone data for a given amount of time on a regular interval of time, such as taking skin tone data for 10 seconds, every half hour</td>
</tr>
<tr>
<td>2</td>
<td>Be able to operate continuously for at least a full workday, so at least 8 hours</td>
</tr>
<tr>
<td>3</td>
<td>Be able to send the skin tone data wirelessly to a smartphone application</td>
</tr>
<tr>
<td>4</td>
<td>Be low profile, comfortable, and almost unnoticeable to the user</td>
</tr>
</tbody>
</table>

For the scope of this project, the ChromaBand must meet all four requirements so that it can be considered a definitive success for a proof of concept style project. If three requirements are met then it is considered a semi successful device but will still require more work. If less than three requirements are met then this device is considered a failure.

4.1 - Testing Requirement 1: Basic Data Collection

To meet the first requirement the ChromaBand must be able to sample skin tone data for an allotted time every regular interval of time. For testing the sampling time was set for 10 seconds and the waiting time interval was set to 30 minutes. Uploading the version 2 code and then operating the device allowed the application to capture the data shown in Figure 4.1.1 below. The readings were taken dry, that is the ChromaBand was not worn and was tested simply for the sake of fulfilling requirement 1. It can be seen in the timestamps on the left that the color sensor samples data from 14:42:04 to 14:42:15 which is 11 seconds. The extra time is accounted for while sending data and printing headers and data to the application’s serial terminal. Also it has the 100 data sets that make up the 10 second reading. It can also be seen in the timestamps that the first data set collection ended at 14:42:15, delayed some time and started data collection for the second data set at around 15:12:22. This delay in time is 30 minutes so the device waiting for 30 minutes in between capturing color data sets works correctly. This correct operation fulfills requirement 1.

<table>
<thead>
<tr>
<th>14:42:04</th>
<th>Color Temp (K)</th>
<th>Red Value</th>
<th>Green Value</th>
<th>Blue Value</th>
<th>Clear Value</th>
<th>Luminosity</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:42:04</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:04</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:04</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:11</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5132</td>
<td>199</td>
<td>166</td>
<td>150</td>
<td>474</td>
<td>87</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5048</td>
<td>199</td>
<td>166</td>
<td>149</td>
<td>474</td>
<td>88</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5297</td>
<td>306</td>
<td>264</td>
<td>237</td>
<td>736</td>
<td>143</td>
</tr>
<tr>
<td>14:42:12</td>
<td>5548</td>
<td>232</td>
<td>198</td>
<td>182</td>
<td>558</td>
<td>103</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5544</td>
<td>228</td>
<td>195</td>
<td>179</td>
<td>551</td>
<td>102</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5531</td>
<td>227</td>
<td>194</td>
<td>178</td>
<td>548</td>
<td>102</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5484</td>
<td>227</td>
<td>193</td>
<td>177</td>
<td>547</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5484</td>
<td>227</td>
<td>193</td>
<td>177</td>
<td>546</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>544</td>
<td>101</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>544</td>
<td>101</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>544</td>
<td>101</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>544</td>
<td>101</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>15:12:22</td>
<td>8890</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15:12:22</td>
<td>14420</td>
<td>161</td>
<td>184</td>
<td>199</td>
<td>524</td>
<td>92</td>
</tr>
<tr>
<td>15:12:22</td>
<td>14414</td>
<td>183</td>
<td>210</td>
<td>227</td>
<td>597</td>
<td>105</td>
</tr>
<tr>
<td>15:12:22</td>
<td>14414</td>
<td>183</td>
<td>210</td>
<td>227</td>
<td>597</td>
<td>105</td>
</tr>
<tr>
<td>15:12:22</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>597</td>
<td>104</td>
</tr>
</tbody>
</table>

14:42:11 Battery Voltage: 4.21 (Full)
<table>
<thead>
<tr>
<th>Time</th>
<th>ID</th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:12:28</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:28</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:28</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:28</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:28</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:29</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:30</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:31</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:33</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:33</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:33</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:33</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:33</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
<tr>
<td>15:12:33</td>
<td>14770</td>
<td>183</td>
<td>209</td>
<td>227</td>
<td>596</td>
<td>104</td>
</tr>
</tbody>
</table>

Figure 4.1.1: Output Showing Two Separate 100 Color Data Sets
4.2 - Testing Requirement 2 – Operation Longevity

To meet the second requirement the ChromaBand must be able to operate for a full workday, so at least 8 hours. This color sensor was operated dry, that is the ChromaBand was not worn and was simply operated continuously for the sake of completing requirement 2. The output shown in Figure 4.1.2 below does not show the entire output because that would take too much report space. Instead the first data set and the last data set are shown to show the time difference between the two. The time difference proves that the device worked properly for that amount of time. The initial data set started at 14:42:04 and the second data set started at 22:45:28 meaning there is a difference of 8 hours between the two data collection times. This shows that the ChromaBand thus can operate for at least 8 hours and thus fulfills requirement 2. Further rigorous testing found that operated continuously the ChromaBand actually lasted 80 hours. The time required to recharge a drained battery to full is about 2 hours.

<table>
<thead>
<tr>
<th>14:42:04</th>
<th>Color Temp (K)</th>
<th>Red Value</th>
<th>Green Value</th>
<th>Blue Value</th>
<th>Clear Value</th>
<th>Luminosity</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:42:04</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:04</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:04</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:05</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:06</td>
<td>5116</td>
<td>198</td>
<td>165</td>
<td>149</td>
<td>472</td>
<td>87</td>
</tr>
<tr>
<td>14:42:13</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>545</td>
<td>101</td>
</tr>
<tr>
<td>14:42:14</td>
<td>5518</td>
<td>226</td>
<td>193</td>
<td>177</td>
<td>544</td>
<td>101</td>
</tr>
<tr>
<td>14:42:15</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:15</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:15</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:15</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:15</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
<tr>
<td>14:42:15</td>
<td>5471</td>
<td>226</td>
<td>192</td>
<td>176</td>
<td>544</td>
<td>100</td>
</tr>
</tbody>
</table>

14:42:15 Battery Voltage: 4.21 (Full)

<table>
<thead>
<tr>
<th>22:45:28</th>
<th>Color Temp (K)</th>
<th>Red Value</th>
<th>Green Value</th>
<th>Blue Value</th>
<th>Clear Value</th>
<th>Luminosity</th>
</tr>
</thead>
<tbody>
<tr>
<td>22:45:28</td>
<td>3785</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22:45:29</td>
<td>7676</td>
<td>245</td>
<td>280</td>
<td>259</td>
<td>745</td>
<td>172</td>
</tr>
<tr>
<td>22:45:29</td>
<td>7642</td>
<td>478</td>
<td>547</td>
<td>505</td>
<td>1455</td>
<td>338</td>
</tr>
<tr>
<td>22:45:29</td>
<td>7642</td>
<td>478</td>
<td>547</td>
<td>505</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7642</td>
<td>478</td>
<td>547</td>
<td>505</td>
<td>1453</td>
<td>338</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7642</td>
<td>478</td>
<td>547</td>
<td>505</td>
<td>1453</td>
<td>338</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7685</td>
<td>478</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>337</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:30</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>Time</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td>Value 4</td>
<td>Value 5</td>
<td>Value 6</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7685</td>
<td>478</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>337</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:31</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7685</td>
<td>478</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>337</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:32</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:33</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:34</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:34</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:34</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:34</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:35</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:35</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:35</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:35</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:36</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:36</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1454</td>
<td>338</td>
</tr>
<tr>
<td>22:45:36</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1455</td>
<td>338</td>
</tr>
<tr>
<td>22:45:36</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1455</td>
<td>338</td>
</tr>
<tr>
<td>22:45:36</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1455</td>
<td>338</td>
</tr>
<tr>
<td>22:45:36</td>
<td>7697</td>
<td>477</td>
<td>547</td>
<td>506</td>
<td>1455</td>
<td>338</td>
</tr>
<tr>
<td>Time</td>
<td>Battery Voltage</td>
<td>LED Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22:45:37</td>
<td>3.94 (Good)</td>
<td>Emitting Blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.1.2: Output Showing Two Separate 100 Color Data Sets Separated by 8 Hours**

### 4.3 - Testing Requirement 3: Bluetooth Communication Integrity

To fulfill the third requirement the ChromaBand must be able to successfully send data to the smartphone application. Requirements 1 and 2 being already fulfilled means that requirement 3 must also be fulfilled. For there to be any output text files like those in Figure 4.1.1 or 4.1.2 then the microcontroller must already be able to send data to the smartphone application. The ChromaBand will only send data assuming the ChromaBand and application are paired and connected. This is shown by the presence of an emitting blue LED on the ChromaBand. If at any time this LED turns off then communication has been broken and the application will not be able to receive any data. Further expanding on requirement 3 it has been noted that sometimes under high smartphone usage the bluetooth communication may be dropped. The dropped communication is intermittent at best and generally very good at maintain a connection. The high smartphone usage usually isn’t consistent issue but whenever the connection was dropped it was under high smartphone usage. For the general case the ChromaBand will have a solid connection to the smartphone. It seldom drops the connection and given these rare circumstances requirement 3 can still be considered fulfilled but more work can be done to ensure a more stable connection in all use cases.
4.4 - Testing Requirement 4: Comfort Level

To fulfill the fourth requirement the ChromaBand must be extremely low profile, comfortable and users should never complain about discomfort. While wearing the ChromaBand myself I unfortunately did notice the ChromaBand at times as it did move awkwardly if I bent down more than 90 degrees while sitting. The ChromaBand is fairly low profile and generally very comfortable, but it is still noticeable at some times. For better testing it would make sense to allow a greater sample size of testers. To do this would require an IRB form for human testing. In this case requirement 4 is not fulfilled and more work can be done to streamline the ChromaBand so that it’s unnoticeable while being worn.

4.5 - Results

Of the four requirements named above only one was not fulfilled. More work can improve its comfort level and noticeability. That being said, the device has fulfilled all of the functional requirements and has met all of the customer’s requirements.

<table>
<thead>
<tr>
<th>Fulfilled</th>
<th>Requirement #</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Be able to sample skin tone data for a given amount of time on a regular interval of time, such as taking skin tone data for 10 seconds, every half hour</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Be able to operate continuously for at least a full workday, so at least 8 hours</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Be able to send the skin tone data wirelessly to a smartphone application</td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>4</td>
<td>Be low profile, comfortable, and almost unnoticeable to the user</td>
</tr>
</tbody>
</table>
Chapter 5 - Conclusion

The ChromaBand is designed as a means for a user to capture and record the changes in a person’s skin tone. It measures no taller than an inch and a half and is short enough to fit comfortably on the waistband of any undergarment. It serves as a means for researchers and scientists to gather day long skin tone data and use it as they deem necessary. This project stands as the initial design and development of a means to capture and save skin tone data. The ChromaBand does just that while utilizing a low power bluetooth protocol to deliver data to any smartphone for an extended time.

Further improvements and extensions of this device can be made in both hardware and software. On the hardware side the ChromaBand can be outfitted with other sensors like a heart rate sensor, temperature sensor, blood pressure sensor, and other vital health related sensors. The ChromaBand could also be minimized and made more comfortable. Updating the hardware will allow a smaller profile as well as increased comfort during normal wear. On the software side is where research and analyzation of the data becomes relevant. This part is considered the evaluation of the data and not a means of collecting the data. New software can be used to scan through the data and see if any changes in skin tone correlate with changes in reported health. The reported health would be a style of survey that users wearing the ChromaBand would fill out throughout the day. They would report what happened to them every hour. These reports used in tandem with analyzed data could very well lead to the supposed correlation between changes in skin tone and changes in health.
Appendix A – Bluetooth Code (Functional Test)

- This code’s purpose is to test that the microcontroller’s Bluetooth module is functional.
- It tests that the microcontroller can send values to a smartphone application.
- The included files are “bluetooth.ino” and “BluefruitConfig.h”

```cpp
1. #include <Arduino.h>
2. #include <SPI.h>
3. #if not defined (_VARIANT_ARDUINO_DUE_X_) && not defined (_VARIANT_ARDUINO_ZERO_)
4. #include <SoftwareSerial.h>
5. #endif
6. #include "Adafruit_BLE.h"
7. #include "Adafruit_BluefruitLE_SPI.h"
8. #include "Adafruit_BluefruitLE_UART.h"
9. #define FACTORYRESET_ENABLE 1
10. #define MINIMUM_FIRMWARE_VERSION "0.6.6"
11. #define MODE_LED_BEHAVIOUR "MODE"
12. Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
13. // A small helper
14. void error(const __FlashStringHelper*err) {
15. Serial.println(err);
16. while(1);
17. }
18. void setup(void)
19. {
20. while (!Serial); //starts up serial monitor, not required
21. Serial.begin(9600);
22. Serial.println(F("Adafruit Bluefruit Command Mode Example");
23. Serial.println(F("-------------------------------");
24. if(!ble.begin(VERBOSE_MODE))
25. {
26. error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?");
27. }
28. Serial.println(F("OK!");
29. if(FACTORYRESET_ENABLE)
30. {
31. /* Perform a factory reset to make sure everything is in a known state */
```
Serial.println(F("Performing a factory reset: "));
if(!ble.factoryReset()){
  error(F("Couldn't factory reset"));
}

/* Disable command echo from Bluefruit */
ble.echo(false);
Serial.println("Requesting Bluefruit info:");
/* Print Bluefruit information */
//ble.info();
Serial.println(F("Please use Adafruit Bluefruit LE app to connect in UART mode");
Serial.println();
ble.verbose(false); // debug info is a little annoying after this point!

/* Wait for connection */
while (!ble.isConnected()) {
  delay(500);
}

void loop(void)
{
  int x = 0;
  while(1){
    /* send 0, 1, 2, 3, and onwards forever */
    ble.print("AT+BLEUARTTX=");
    ble.print(x);
    ble.println("\r\n");
    /* increment integer value for next iteration */
    x++;
    /* wait 1 second and then send again by looping over */
    delay(1000);
    if(!ble.waitForOK()) {
      Serial.println(F("Failed to send?");
    }
  }
}

- BluefruitConfig.h
#include <Arduino.h>
#include <SPI.h>
#if !defined (_VARIANT_ARDUINO_DUE_X_) && !defined (_VARIANT_ARDUINO_ZERO_)
#include <SoftwareSerial.h>
#endif
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"
#include "BluefruitConfig.h"
#define FACTORYRESET_ENABLE 1
#define MINIMUM_FIRMWARE_VERSION "0.6.6"
#define MODE_LED_BEHAVIOUR "MODE"

Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

// A small helper
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while(1);
}

/**************************************************************/
/* @brief Sets up the HW an the BLE module (this function is called automatically on startup) */
/**************************************************************/
void setup(void)
{
  //while (!Serial); //starts up serial monitor, not required
  delay(500);
  Serial.begin(9600);
  Serial.println(F("Adafruit Bluefruit Command Mode Example"));
  Serial.println(F("-------------------"));
  /* Initialise the module */
  Serial.print(F("Initialising the Bluefruit LE module: ");
  if(!ble.begin(VERBOSE_MODE))
  { error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?");
  }
  Serial.println(F("OK!");
  if(FACTORYRESET_ENABLE)
  { /* Perform a factory reset to make sure everything is in a known state */
  Serial.println(F("Performing a factory reset: ");
  if(!ble.factoryReset())
  { error(F("Couldn't factory reset");
  }
  }
  /* Disable command echo from Bluefruit */
ble.echo(false);

Serial.println("Requesting Bluefruit info:");

/* Print Bluefruit information */
//ble.info();

Serial.println(F("Please use Adafruit Bluefruit LE app to connect in UART mode"));
Serial.println();

ble.verbose(false); // debug info is a little annoying after this point!

/* Wait for connection */
while (!ble.isConnected()) {
  delay(500);
}

void loop(void)
{
  int x;
  while(1){
    ble.print("AT+BLEUARTTX=");
    ble.print(x);
    ble.println("\r\n");
    x++;
    delay(1000);
    if(!ble.waitForOK()) {
      Serial.println(F("Failed to send?"));
    } 
  }
}

}
Appendix B – Color Sensor Code (Functional Test)

- This code’s purpose is to test that the color sensor is functional.
- It tests that the color sensor can produce Red, Green, Blue, Clear, Color Temperature, and Luminosity values.
- The included file is “colorsensor.ino”

```
colorsensor.ino
1. #include <Wire.h>
2. #include "Adafruit_TCS34725.h"
3.
4. /* Initialise with specific int time and gain values */
5. Adafruit_TCS34725_tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_101MS, TCS34725_GAIN_60X);
6.
7. void setup(void) {
8.    prepSerialMonitor();
9.    prepColorSensor();
10. }
11.
12. void prepSerialMonitor(){
13.    Serial.begin(9600);
14.    while(!Serial);
15. }
16.
17. void prepColorSensor(){
18.    Serial.print("Prepping Color Sensor...");
19.    if (!tcs.begin()) {
20.        Serial.println("Prep Failed!");
21.        return;
22.    }
23.    Serial.println("Prep Done!");
24. }
25.
26. void captureColorData(){
27.    uint16_t r, g, b, c, colorTemp, lux;
28.    /* take sample and fill red, green, blue, and clear variables, then calculate color temperature and luminosity */
29.    tcs.getRawData(&r, &g, &b, &c);
30.    colorTemp = tcs.calculateColorTemperature(r, g, b);
31.    lux = tcs.calculateLux(r, g, b);
32.    /* print these values to the Arduino IDE serial monitor */
33.    Serial.print("Color Temp: "); Serial.print(colorTemp, DEC); Serial.print("K ");
34.    Serial.print("Lux: "); Serial.print(lux, DEC); Serial.print(" ");
35.    Serial.print("R: "); Serial.print(r, DEC); Serial.print(" ");
36.    Serial.print("G: "); Serial.print(g, DEC); Serial.print(" ");
37.    Serial.print("B: "); Serial.print(b, DEC); Serial.print(" ");
38.    Serial.print("C: "); Serial.print(c, DEC); Serial.print(" ");
39.    Serial.println();
40. }
41.
42. void loop(void) {
43.    int i;
44.    /* turn on color sensor LED */
45.    tcs.setInterrupt(false);
46. ```
/* take 100 data sets */
for(i = 0; i < 100; i++){
captureColorData();
}
Serial.println();

/* turn off color sensor LED */
tcs.setInterrupt(true);

/* wait for 5 seconds and then loop over taking data again */
delay(5000);
Appendix C – Version 1 Code (First iteration code for ChromaBand)

- This code’s purpose is to serve as an initial draft for a completed ChromaBand
- It tests if the device as a whole can take only Red, Green, and Blue values and then send those values to a smartphone application
- It takes 100 data sets with a 101ms integration time so a 10 second reading overall, it then delays 5 seconds and takes another 10 second reading
- The included files are “version1.ino” and the same “BluefruitConfig.h” from Appendix A

**version1.ino**

```cpp
1. #include <Arduino.h>
2. #include <SPI.h>
3. #include <Wire.h>
4. #include "Adafruit_TCS34725.h"
5. #if not defined (_VARIANT_ARDUINO_DUE_X_) && not defined (_VARIANT_ARDUINO_ZERO_)
6. #include <SoftwareSerial.h>
7. #endif
8.
9. #include "Adafruit_BLE.h"
10. #include "Adafruit_BluefruitLE_SPI.h"
11. #include "Adafruit_BluefruitLE_UART.h"
12.
13. #include "BluefruitConfig.h"
14.
15. #define FACTORYRESET_ENABLE 1
16. #define MINIMUM_FIRMWARE_VERSION "0.6.6"
17. #define MODE_LED_BEHAVIOUR "MODE"
18.
19. //* Hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
20. Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
21.
22. //* Initialise with specific int time and gain values */
23. Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_101MS, TCS34725_GAIN_1X);
24.
25. // A small helper
26. void error(const __FlashStringHelper*err) {
27.   Serial.println(err);
28.   while(1);
29. }
30.
31. void setup(void)
32. {
33.   //prepSerialMonitor(); /* dont use this if not connected to computer */
34.   prepColorSensor(); /* preps the color sensor and establishes connection */
35.   delay(500);
36.   Serial.begin(9600);
37.   Serial.println(F("Adafruit Bluefruit Command Mode Example"));
38.   Serial.println(F("---------------------------------------"));
39.   Serial.print(F("Initialising the Bluefruit LE module: "));
40.   if(!ble.begin(VERBOSE_MODE))
41.     { ...
42.     }
43.   
44.   /* Initialise the module */
45.   Serial.print(F("Initialising the Bluefruit LE module: "));
46.   Serial.print(F("---------------------------------------"));
47.   
48.   if(!ble.begin(VERBOSE_MODE))
49.     { ...
50.     }
51.   
52.   /* Collect data */
53.   for(int i = 0; i < 100; i++){
54.     tcs.integrate_and_read_all();
55.     Serial.print("Red: ");
56.     Serial.print(tcs.red);
57.     Serial.print(" Green: ");
58.     Serial.print(tcs.green);
59.     Serial.print(" Blue: ");
60.     Serial.println(tcs.blue);
61.   }
62.   
63.   /* Send data to BLE */
64.   for(int i = 0; i < 10; i++){
65.     ble.begin(BLE_TRANSPORT_UART);...
66.   }
67.
68.   /* Process data */
69.   for(int i = 0; i < 10; i++){
70.     //process data...
71.   }
72.
73.   /* Delay 5 seconds */
74.   delay(5000);
75.
76.   /* Collect data */
77.   for(int i = 0; i < 100; i++){
78.     tcs.integrate_and_read_all();
79.     Serial.print("Red: ");
80.     Serial.print(tcs.red);
81.     Serial.print(" Green: ");
82.     Serial.print(tcs.green);
83.     Serial.print(" Blue: ");
84.     Serial.println(tcs.blue);
85.   }
86.   
87.   /* Send data to BLE */
88.   for(int i = 0; i < 10; i++){
89.     ble.begin(BLE_TRANSPORT_UART);...
90.   }
91.
92.   /* Process data */
93.   for(int i = 0; i < 10; i++){
94.     //process data...
95.   }
96.   
97.   /* Loop back */
98.   delay(5000);
99. }
100. ```
```cpp
if (FACTORYRESET_ENABLE) {
    /* Perform a factory reset to make sure everything is in a known state */
    Serial.println(F("Performing a factory reset: "));
    if (!ble.factoryReset()){
        error(F("Couldn't factory reset"));
    }
}
/* Disable command echo from Bluefruit */
ble.echo(false);
Serial.println("Requesting Bluefruit info:");
/* Print Bluefruit information */
//ble.info();
Serial.println(F("Please use Adafruit Bluefruit LE app to connect in UART mode"));
Serial.println();
ble.verbose(false); // debug info is a little annoying after this point!
/* Wait for connection */
while (!ble.isConnected()) {
    delay(500);
}
void prepSerialMonitor(){
    Serial.begin(9600);
    while (!Serial);
}
void prepColorSensor(){
    Serial.print("Prepping Color Sensor . . . ");
    if (!tcs.begin()) {
        Serial.println("Prep Failed!");
        return;
    }
    Serial.println("Prep Done!");
}
void captureColorData(){
    uint16_t r, g, b, c, colorTemp;
    /* take sample from sensor */
    tcs.getRawData(&r, &g, &b, &c);
    /* send through bluetooth module */
    ble.print("AT+BLEUARTTX-\r\n");
    ble.print("R: "); ble.print(r, DEC); ble.print("\t");
    ble.print("G: "); ble.print(g, DEC); ble.print("\t");
    ble.print("B: "); ble.print(b, DEC); ble.print("\t");
    ble.println("\r\n");
    /* wait for delivery ack */
    if (!ble.waitForOK()) {
```
```
Serial.println(F("Failed to send"));
}

void loop(void)
{
  int i;

  /* turn on color sensor led */
  tcs.setInterrupt(false);

  /* collect 100 data points at 101ms each so total of 10 second reading */
  for(i = 0; i < 100; i++){
    captureColorData();
  }

  /* print divider */
  ble.print("AT+BLEUARTTX=");
  ble.print("==================================================================")
  ble.println("\r\n");

  /* turn off color sensor led */
  tcs.setInterrupt(true);
  delay(5000);
}
Appendix D – Version 2 Code (Second iteration code for ChromaBand)

- This code’s purpose is to serve as an improvement on the initial draft for a completed ChromaBand
- It tests if the device as a whole can take Red, Green, Blue, Clear, Color Temperature, Luminosity values and then send those values to a smartphone application
- It also tests the battery and reports back its status and warnings if needed
- It takes 100 data sets with a 101ms integration time so a 10 second reading overall, it then delays 5 seconds and takes another 10 second reading
- The included files are “version2.ino” and the same “BluefruitConfig.h” from Appendix A

- Version2.ino

```cpp
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include "Adafruit_TCS34725.h"

#if not defined (_VARIANT_ARDUINO_DUE_X_) && not defined (_VARIANT_ARDUINO_ZERO_)

#include <SoftwareSerial.h>
#endif

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"
#include "BluefruitConfig.h"

#define FACTORYRESET_ENABLE 1
#define MINIMUM_FIRMWARE_VERSION "0.6.6"
#define MODE_LED_BEHAVIOUR "MODE"
#define VBATPIN A9

int initialReading = 1;

/* Hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RS T */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

/* Initialise with specific int time and gain values */
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_101MS, TCS34725_GAIN_1X);

/* A small helper */
void error(const __FlashStringHelper*err)
{
    Serial.println(err);
    while(1);
}

/* bluetooth chip setup */
void setup(void)
{
    //prepSerialMonitor(); /* dont use this if not connected to computer */
    prepColorSensor(); /* preps the color sensor and establishes connection */
    delay(500);
    Serial.begin(9600);
    Serial.println(F("Adafruit Bluefruit Command Mode Example"));
    Serial.println(F("---------------------------------------------------"));
}
/* Initialise the module */
Serial.print(F("Initialising the Bluefruit LE module: "));

if(!ble.begin(VERBOSE_MODE)) {
  error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?");
}
Serial.println(F("OK!"));

if(FACTORYRESET_ENABLE) {
  /* Perform a factory reset to make sure everything is in a known state */
  Serial.println(F("Performing a factory reset: "));
  if(!ble.factoryReset()){
    error(F("Couldn't factory reset"));
  }
}

/* Disable command echo from Bluefruit */
ble.echo(false);
Serial.println("Requesting Bluefruit info:");
/* Print Bluefruit information */
ble.info();
Serial.println(F("Please use Adafruit Bluefruit LE app to connect in UART mode"));
Serial.println();
ble.verbose(false); // debug info is a little annoying after this point!

/* Wait for connection */
while (!ble.isConnected()) {
  delay(500);
}

/* setup serial monitor in case we are debugging via arduino IDE serial monitor */
void prepSerialMonitor(){
  Serial.begin(9600);
  while(!Serial);
}

/* setup color sensor so we can use it */
void prepColorSensor(){
  Serial.print("Prepping Color Sensor . . . ");
  if (!tcs.begin()) {
    Serial.println("Prep Failed!");
    return;
  }
  Serial.println("Prep Done!");
}

/* captures color data and sends through bluetooth module */
void captureColorData(){
  /* variables for holding values of color data */
  uint16_t r;
  uint16_t g;
  uint16_t b;
  uint16_t c;
uint16_t colorTemp;
uint16_t lux;

/* take a reading from sensor, and calculate color temperature from r, g, and b */
tcs.getRawData(&r, &g, &b, &c);
colorTemp = tcs.calculateColorTemperature(r, g, b);
lux = tcs.calculateLux(r, g, b);

/* send values through bluetooth module */
ble.print("AT+BLEUARTTX=");
ble.print("t"); ble.print(colorTemp, DEC); ble.print("t");
ble.print("t"); ble.print(r, DEC); ble.print("t");
ble.print("t"); ble.print(g, DEC); ble.print("t");
ble.print("t"); ble.print(b, DEC); ble.print("t");
ble.print("t"); ble.print(lux, DEC); ble.print("t");
ble.println("\r\n");

/* wait for delivery ack */
if(!ble.waitForOK()) {
  Serial.println(F("Failed to send?"));
}

/* print battery level and charge reminders */
void printBatteryStatus(){
  float measuredvbat;
  measuredvbat = analogRead(VBATPIN);
  measuredvbat *= 2; // account for double 100k resistor in microcontroller
  measuredvbat *= 3.3; // multiply by 3.3V, the reference voltage
  measuredvbat /= 1024; // convert to voltage
  ble.print("AT+BLEUARTTX=");
  ble.print("Battery Voltage: ");
  ble.print(measuredvbat);
  if(measuredvbat < 3.3){
    ble.print(" (Critical) . . . Charge Now! ");
  }
  else if(measuredvbat < 3.7){
    ble.print(" (Low) . . . Charge Soon! ");
  }
  else if(measuredvbat < 4.2){
    ble.print(" (Good) ");
  }
  else if(measuredvbat < 5.0){
    ble.print(" (Full) ");
  }
  ble.println("\r\n");

  /* print a divider for organization and clarity sake */
  void printDivider(){
    /* print divider */
    ble.print("AT+BLEUARTTX=");
    ble.print("===================================================================
================================");
    ble.println("\r\n");
  }
}
/* print some verbose initialization */
void printInitialHeader(){
    ble.print("AT+BLEUARTTX=");
    ble.print("Starting Skin Data Sampling");
    ble.println("\r\n");
}

/* helper function to print columns header */
void printDataHeader(){
    ble.print("AT+BLEUARTTX=");
    ble.print("| Color Temp (K) | Red Value | Green Value | Blue Value |
    Clear Value | Luminosity |");
    ble.println("\r\n");
}

/* main process loop */
void loop(void)
{
    int i;
    /* on setup delay a little for communications to settle */
    if(initialReading == 1){
        delay(2000);
    }
    /* print init header */
    printDivider();
    printInitialHeader();
    printDivider();
    /* we only need to enter this once */
    initialReading = 0;
}

/* print header for different values columns */
printDataHeader();

/* turn on color sensor led */
tcs.setInterrupt(false);

/* collect 100 data points at 101ms each so total of 10 second reading */
for(i = 0; i < 100; i++){
    captureColorData();
}

/* turn off color sensor led */
tcs.setInterrupt(true);

/* print battery status and whether you should charge soon or not */
printDivider();
printBatteryStatus();
printDivider();

/* delay 30 minutes */
for(i = 0; i < 60; i++){
    delay(30000);
}
}
Appendix E – ChromaBand Schematic

Appendix F – Serial Bluetooth Terminal Application

The apk file (installer file) for the app is included in the zip file attached with this report in the file path ~/Software_And_Application > app > Serial Bluetooth Terminal_de.kai_morich.serial.bluetooth_terminal

This app is not my creation or my property. All credit for the app goes to Kai Morich, an independent developer who built this app and made it available on the Google Play Store. The link below is for an old version of the app. Included in the zip file as described above is a beta version of the application that works better with the ChromaBand.


The “nRF UART 2.0” and “Adafruit Bluefruit LE Connect” applications also work well with logging data. They can be found on the Google Play Store.


Appendix G – Software Flowchart

1. ChromaBand Powered On
2. Wait for Application to Pair with ChromaBand
3. App and ChromaBand Paired?
   - NO
4. Turn On LED
   - NO
5. Sample Skin Tone Data
   - NO
6. Sent Enough Data Sets?
   - NO
7. Send Data to Application
   - YES
8. Turn Off LED
   - YES
9. Send Battery Status to App
   - NO
10. Wait for 30 Minutes
    - NO
11. Has 30 Minutes Elapsed?
    - YES
Appendix H – Setting up Arduino IDE and Software Libraries

The ChromaBand comes with the software preinstalled. However, if the user wishes to update the software or change reading times/waiting times then the user must setup the Arduino integrated development environment (IDE) and other software libraries with the following:

- **Installing Arduino IDE**
  1. Download the Arduino IDE installer from the link below. Version must be at least 1.6.4
     [https://www.arduino.cc/en/Main/Software](https://www.arduino.cc/en/Main/Software)
  2. Now click on the installer and install the Arduino IDE
  3. After installing the IDE, start up the Arduino IDE program
  4. Now click on **File > Preferences**
  5. Add the following line to lower field titled “**Additional Boards Manager URLs**”
     [https://adafruit.github.io/arduino-board-index/package_adafruit_index.json](https://adafruit.github.io/arduino-board-index/package_adafruit_index.json)
  6. Once entered correctly, click **OK**

- **Adding Adafruit Feather Microcontroller Support to Arduino IDE**
  1. Next, click on **Tools > Board > Board Manager**
  2. In the **Type** drop down menu select **Contributed**
  3. Click on **Adafruit AVR Boards** and install version 1.4.8 or newer
  4. Close the Arduino IDE and restart to make sure changes are kept.
  5. Select **Tools > Board**, and **Adafruit Feather 32u4** should be in the list somewhere

- **Installing Windows Specific Driver**
  1. If installing on windows do the following steps, otherwise skip this section
  2. Download the windows driver by clicking the link below
     [http://adafruit.it/ma1](http://adafruit.it/ma1)
  3. Download and run the installer, make sure **Feather 32u4** is selected and **Install**

- **Adding Bluetooth Support to Arduino IDE**
  1. Download the support library by clicking the link below
     [http://adafruit.it/f4W](http://adafruit.it/f4W)
  2. Rename the uncompressed folder **Adafruit_BluefruitLE_nRF51**
  3. Place the **Adafruit_BluefruitLE_nRF51** library folder to your Documents/Arduino/libraries/ folder.
     a. You may need to create the libraries subfolder. Restart the IDE.

- **Adding Color Sensor Support to Arduino IDE**
  1. Download the support library by clicking the link below
     [http://adafruit.it/cb1](http://adafruit.it/cb1)
  2. Click on Clone or **Download > Download Zip**
  3. Rename the uncompressed folder **Adafruit_TCS34725_master**
  4. Place the **Adafruit_TCS34725_master** library folder to your Documents/Arduino/libraries/ folder.
Appendix I – Uploading Software to the ChromaBand

1. Connect a MicroUSB cable to the microcontroller
2. Start up the Arduino IDE
3. Click on **Tools** and make sure Board says “Adafruit Feather 32u4”
4. Click on **Tools** and make sure Port says **Com# (Adafruit Feather 32u4)**
   a. The # could be any number, it is determined by your computer’s OS
5. Open whatever software you wish to upload and click the upload button
   a. Upload button is the second circular button with a right facing arrow
6. If all is good then you will see Done Uploading in the status bar below.

Appendix J – General User Manual

1. Equip the ChromaBand on undergarment waistband with color sensor facing and touching the waist.
2. Turn on the ChromaBand
3. The ChromaBand will now not have a blue LED emitting.
4. With smartphone in hand, turn on Bluetooth
5. Start the **Serial Bluetooth Terminal** application.
6. Tap the **three bars** on the upper left and then select **Bluetooth Devices**
7. Make sure Bluetooth LE is picked
8. Tap **SCAN** on the upper right and wait
9. The device should be listed as **Adafruit Bluefruit LE**
10. Tap on **Adafruit Bluefruit LE** and the bar should turn **green**
11. Tap the three bars on the upper left and then select **Terminal**
12. To connect with the ChromaBand tap on the icon in the upper right immediately to the right of the “Terminal” title.
13. The ChromaBand will now have a blue LED emitting.
   a. This means the two are connected
   b. The app will also say whether or not a connection is made
14. The ChromaBand will take data until you disconnect them or the battery expires
15. When enough data is logged in the app you can save the data
   a. To save the data, tap on the **three dots** on the upper right and select **Save**
16. The text file is saved in the location shown below on Android fans
    Local > Internal Storage > Android > Data > de.kai_morich.serial_bluetooth_terminal > files
17. The file will be titled in the following format
    **Serial_yyyyymmdd_##.txt**
    a. yyyy is the year
    b. mm is the month
    c. dd is the day
Appendix K – Installing Serial Bluetooth Terminal Application

- There are two ways to install this application

- Via the Google Play Store
  1. Go to the Google Play Store on the smartphone
  2. Search for Serial Bluetooth Terminal
     a. It will be the first entry and its icon will be an adapter with a Bluetooth logo
  3. Install the application

- Via the .apk file
  1. Connect smartphone to computer
  2. Navigate to the smartphone’s files through windows explorer
  3. Navigate to the file path Local > Internal Storage > Download
  4. Place the .apk file into the Download folder
  5. Disconnect the smartphone
  6. Navigate to the same file path listed in step 4 through the File Manager/My Files app
  7. Tap on the .apk file and install the application

Appendix L – ChromaBand Price Evaluation

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit 32u4 Bluetooth LE Microcontroller</td>
<td>1</td>
<td>25.99</td>
</tr>
<tr>
<td>RGB Color Sensor TCS34725</td>
<td>1</td>
<td>7.95</td>
</tr>
<tr>
<td>Lithium Ion Polymer Battery 3.7V 1200mAh</td>
<td>1</td>
<td>9.95</td>
</tr>
<tr>
<td>Adafruit Micro Lipo Charger with MicroUSB Jack</td>
<td>1</td>
<td>6.95</td>
</tr>
<tr>
<td>Wire with JST Connector</td>
<td>1</td>
<td>0.28</td>
</tr>
<tr>
<td>Switch</td>
<td>1</td>
<td>0.39</td>
</tr>
<tr>
<td>8” x 1.5” Elastic Band</td>
<td>1</td>
<td>0.15</td>
</tr>
<tr>
<td>6” Velcro</td>
<td>1</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Total: 52.42