ECE 445

Spring 2025
Senior Design - Final Report
Colorimeter for Skin Tone & Makeup Application

By

Team 6
Ashley Herce, Waidat Bada, Shriya Surti
TA: John Li

May 2025

Abstract

This report describes the implementation of our colorimeter for ECE 445, including the logic, details, and designs. We provide descriptions of the project’s features, design, cost, and algorithms used for our project

Table of Contents

1. Introduction

1.1 Problem

The beauty industry faces significant sustainability challenges, primarily stemming from systemic inefficiencies in accurately analyzing and matching skin tones to color products. These inefficiencies arise from several factors, including lighting variations, skin condition differences, and subjective assessment methods. As a result, consumers often end up with mismatched products, leading to high return rates, increased environmental waste, as well as economic losses for brands.

In addition, the beauty industry has long favored lighter skin tones, and many product ranges are sparse or fail to adequately represent olive undertones and deeper shades. While shade ranges are gradually expanding, there is still a notable lack of inclusivity, especially for deeper skin tones. The commonly used Fitzpatrick scale, which classifies skin types based on sensitivity to UV exposure, underrepresents deeper shades, making it difficult for consumers with these skin tones to find products that truly match their complexion [1].

This lack of representation leads to increased frustration among consumers, particularly those with unique undertones and/or deeper complexions. Despite ongoing efforts to expand product ranges, the true distribution of makeup products contributes to a crisis of unmet needs and underrepresentation.

1.2 Solution

To address these challenges, we propose the development of an innovative colorimeter device that enhances beauty by accurately analyzing skin tone with the use of various backlights and recommending suitable products such as skin tints and foundation. Our device integrates a high-precision color sensor that measures the XYZ values of a person's skin across various lighting conditions. Our device also incorporates a replication of various forms of lighting to account for the variations in ambient lighting, melanin distribution, and undertones, providing a "true" and reliable reading of a person's skin tone, regardless of lighting inconsistencies.

By utilizing XYZ color space values, which are more universally consistent than traditional models like RGB or HSV, our device eliminates the inaccuracies often caused by skin condition variations. By also mimicking various lighting, our device can produce a more accurate match to makeup products, which is particularly important for individuals with deeper skin tones or undertones that are commonly overlooked by traditional beauty tools.

Our device is powered by an ESP32 microcontroller, which processes the data from the color sensor and transmits the results to a mobile app. Through the app, users can view personalized foundation matches and receive product recommendations from an extensive database that includes a diverse range of brands, shades, and price points.

Our device aims to challenge the industry's limited approach by mapping out a more comprehensive range of skin tones along the Monk skin tone scale, which emphasizes deeper and more inclusive shades. Overall, this will help reduce consumer frustration and raise awareness around the importance of better representation of skin tones in the beauty industry.

2. Design

Block Diagram

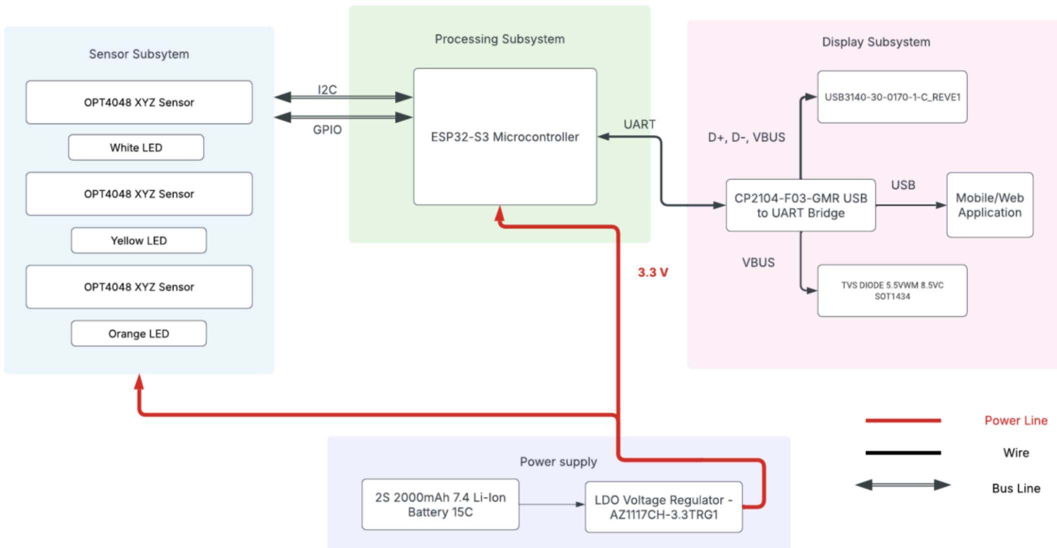

Figure 2.1.1: (ORIGINAL) High-level breakdown of our colorimeter

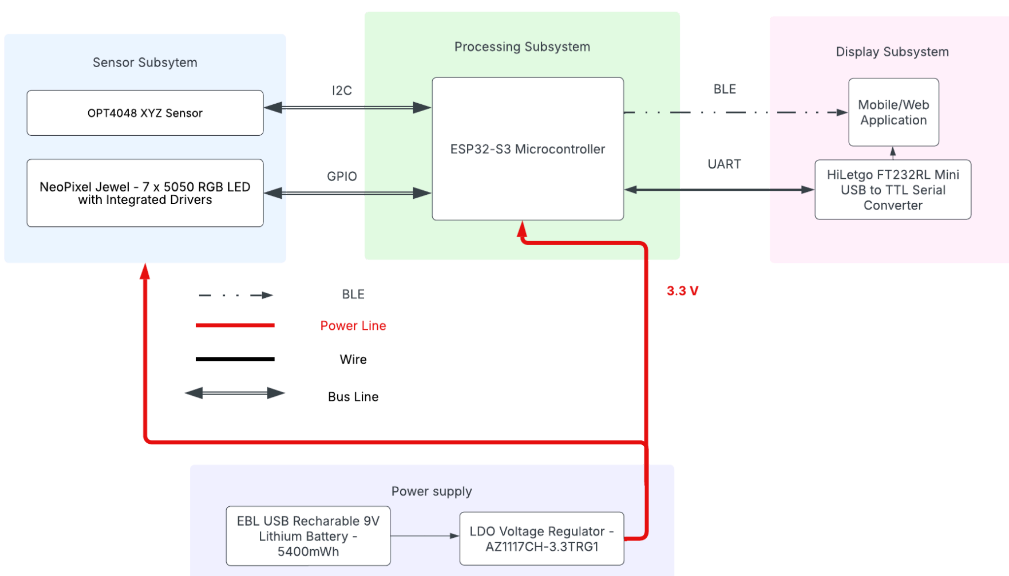

Figure 2.1.2: (FINAL) High-level breakdown of our colorimeter

2.1.1 Block Design Evolution

Our initial system architecture featured three individual color sensors, each designated for testing under distinct lighting conditions. These sensors were to be evaluated both independently and concurrently to enable comparative analysis of real-time data across varied environments. Each sensor was paired with its own discrete LED to simulate specific lighting scenarios.

To facilitate communication with the microcontroller, we originally integrated a CP2104 USB-to-UART bridge directly onto the PCB, allowing seamless data transfer via USB. The system was powered by a 7.4V battery, which was stepped down to 3.3V to safely operate the logic and sensor components.

As development progressed, we identified several areas for simplification and optimization. The use of three separate color sensors and discrete LEDs introduced unnecessary complexity and increased spatial constraints. To address this, we redesigned the system to utilize a single high-accuracy color sensor paired with an Adafruit NeoPixel Jewel. The NeoPixel array provides consistent, controllable illumination while significantly reducing the overall footprint.

In terms of communication, we transitioned from an onboard USB-to-UART interface to an external UART-to-USB adapter, used solely during the firmware flashing stage. For routine operation, we implemented Bluetooth Low Energy (BLE) communication, enabling wireless data transmission to a mobile application and eliminating the need for a persistent USB connection.

Additionally, we updated the power system by replacing the 7.4V battery with a 9V rechargeable battery, regulated down to 3.3V. This change provided a more compact and easily rechargeable solution while maintaining reliable power delivery to all components.

2.1.2 PCB Design and Subsystem Modularity

Our PCB design process closely followed the functional block diagram of the system. Initially, all our subsystems, including the microcontroller, power regulation, color sensor, and lighting components, were consolidated onto a single PCB. While this unified approach simplified early layout, it quickly became evident that the board was too large (80 mm × 90 mm) and not efficient for testing and development.

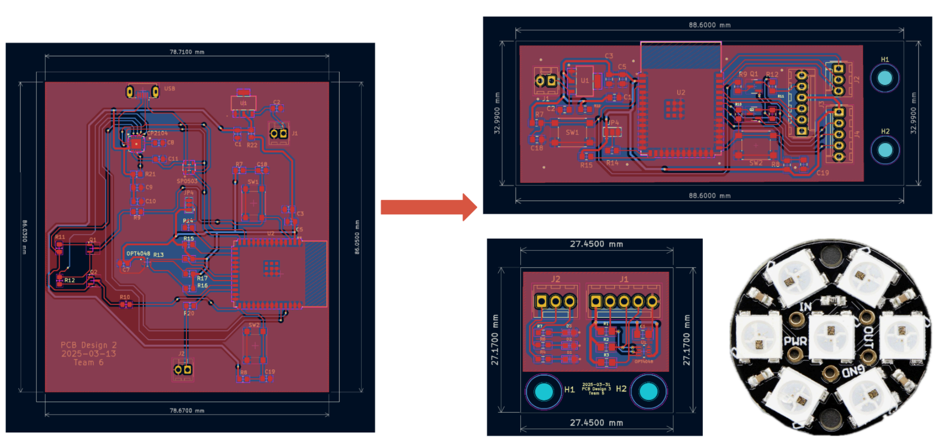

Figure 2.1.3: PCB Design Changes

To address these challenges, we restructured the hardware by partitioning the design into two distinct PCBs where PCB 1 consisted of our Color sensor and LED components and PCB 2 consisted of our microcontroller and power subsystem.

This separation introduced several advantages for us like modular testing, reduced board size and subsystem isolation. Each subsystem could be evaluated independently, improving the efficiency of debugging and development.

Our controller board was significantly reduced in footprint, from 80 mm × 90 mm to 80 mm × 30 mm. In the event of failure or performance issues in one subsystem, the other could still be operated and demonstrated without disruption.

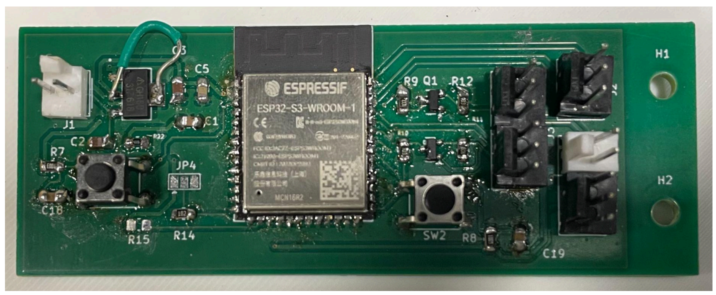

Figure 2.1.4: PCB 2 consisting of our microcontroller and power subsystem

The color sensor and NeoPixel were placed together on PCB 1 due to their functional interdependence since the LEDs serve as the light source during color detection operations. This location ensured optimal alignment and simplified control logic.

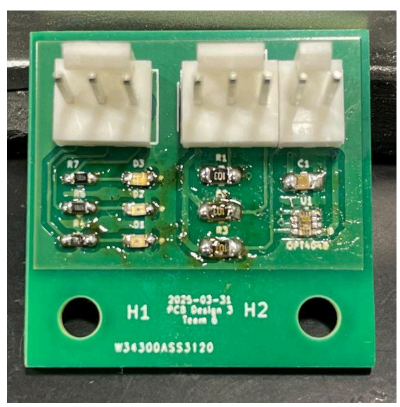

Figure 2.1.4: PCB 1 consisting of our Color sensor and LED components

During initial testing, we discovered that the onboard LEDs were insufficiently bright to serve as a backlight for color scanning. As a result, we repurposed our onboard LEDs as diagnostic indicators for debugging purposes. We used a NeoPixel Jewel in their place to serve as the primary lighting element. It was able to offer brighter and more uniform illumination as well as programmable color output, enabling simulation of various ambient lighting conditions

This refinement significantly improved the reliability and accuracy of color measurements, particularly under controlled lighting scenarios.

2.2 Power Subsystem

2.2.1 Description & Purpose

The power subsystem is responsible for supplying stable and regulated power to all components in the color-matching device, including the ESP32 microcontroller, color sensor, and RGB LEDs. The system operates at 3.3V, and the power subsystem ensures that each peripheral receives the correct voltage and current to function reliably. The integrity of the power delivery directly affects the performance and communication of the entire system.

2.2.2 Cell Choice Justification

Initially, we selected a 2S 2000 mAh 7.4V Li-ion battery with a 15C discharge rating. This battery contains two 3.7V cells in series, providing a total nominal voltage of 7.4V. Given its capacity and C rating, it was capable of delivering up to 30A of current continuously, which was more than sufficient for our system's expected load. However, during testing, the ESP32 only received about 3.0V, leading to unreliable performance and peripheral communication failures. After investigating, we concluded that voltage drops and inefficiencies in regulation contributed to the issue. We then transitioned to using a 9V rechargeable battery, which, after regulation, supplied a consistent 3.3V to the entire system. This change ensured proper logic levels on GPIO pins and restored stable communication with peripherals.

2.2.3 Voltage Regulation Justification

To convert the 9V battery output to the required 3.3V, we implemented a low-dropout (LDO) voltage regulator. An LDO was chosen over a buck converter for its superior noise performance, which is important for sensor accuracy and microcontroller stability. Furthermore, the small voltage difference between input and output made the LDO a more efficient and responsive choice in this scenario. The faster transient response of the LDO also helps minimize power waste during idle periods. To preserve battery life and prevent unnecessary drain, we ensured that the power subsystem could be manually disconnected when the device is not in use.

2.2.4 Interactions

The power subsystem connects directly to all other subsystems, supplying 3.3V regulated power to the ESP32, the color sensor, and the RGB LEDs. It plays a crucial role in maintaining reliable performance by ensuring each component receives consistent voltage and current. Any instability in power delivery would lead to degraded functionality or communication errors across the system.

2.2.5 Requirements

The power subsystem must be capable of providing an average continuous current of up to 30A to satisfy the maximum potential load, even though the actual current demand is significantly lower. Additionally, it must maintain a regulated output of 3.3V with a tolerance of ±5%, ensuring stable operation for all voltage-sensitive components in the circuit.

2.2.6 Verifications

Figure 2.2.1 Power Subsystem output Verification

To verify that the power subsystem met its requirements, we conducted several tests. First, we used a low-resistance shunt resistor to measure the current supplied by the battery and calculated the current based on the voltage drop measured across the resistor. To confirm the voltage output, we used a multimeter to measure the regulated 3.3V output under load and verified that it remained within the required ±5% tolerance. Continuity tests were performed using a multimeter to ensure reliable connections across the power lines. An oscilloscope was also used to measure the output voltage ripple, checking for stability under varying load conditions. During testing, we encountered an issue with the LDO regulator causing a short circuit when connected to the 9V input. This prompted a review of the component footprint and a closer analysis of the LDO's datasheet, which helped us identify the source of the problem and implement corrective measures.

2.3.Processing Subsystem

2.3.1 Design Procedure

The ESP32-S3 microcontroller serves as the central processor for the system. It was chosen for its integrated Bluetooth, robust support for I2C communication, and sufficient computing power to perform sensor data processing and wireless transmission in real time.

The ESP32 interfaces with the color sensor via I2C and controls the Neopixel Jewel RGB LEDs using a GPIO pin with the FastLED library. This allowed us to set the LED colors without dealing with timing conflicts.

It also processes raw color sensor data on-chip, converting it into six-digit HEX values, which are used to classify skin tones and recommend foundation matches. The processed data is transmitted to the display subsystem via Bluetooth, enabling real-time feedback.

2.3.2 Design Details

Sensor Communication: The color sensor is connected using the ESP32's I2C interface (SDA and SCL lines). The sensor is polled periodically to collect XYZ color data.

Data Processing: The ESP32 converts the raw sensor data into six-digit HEX color codes, representing the sensed skin tone. This involves scaling RGB values and formatting them into HEX.

Bluetooth Communication: Processed HEX values are sent wirelessly to the display system via Bluetooth UART, enabling real-time updates.

2.3.3 Verification

Support I2C and efficiently acquire sensor data

To verify I2C communication, we connected a voltmeter to the SDA and SCL lines, then triggered a sensor read operation. Voltage changes confirmed that data was being successfully transmitted between the sensor and the ESP32.

Control RGB LED brightness and color settings

We verified Neopixel LED control by visually observing the color and brightness changes in response to different commands sent from the ESP32. The FastLED library handled timing and updates, and the LEDs reflected the correct output during testing.

Process raw color sensor data

We printed the raw RGB values collected by the sensor via serial output. These values were confirmed to be accurate under different lighting conditions, ensuring the ESP32 was correctly processing sensor data.

```
09:33:31.842 ->  CIEy: 0.34
09:33:32.038 -> CIEx: 0.63
09:33:32.038 ->  CIEy: 0.34
09:33:32.234 -> CIEx: 0.63
09:33:32.234 ->  CIEy: 0.34
09:33:32.463 -> CIEx: 0.63
09:33:32.463 ->  CIEy: 0.34
09:33:32.660 -> CIEx: 0.63
09:33:32.660 ->  CIEy: 0.34
09:33:32.858 -> CIEx: 0.63
09:33:32.858 ->  CIEy: 0.34
09:33:33.056 -> CIEx: 0.63
09:33:33.056 ->  CIEy: 0.34
```

Figure 2.3.1 CIE xy output

Convert raw data into HEX values to determine skin tone

The ESP32 successfully converted RGB values into six-digit HEX codes. We compared these HEX outputs against a reference skin tone chart to validate the accuracy of the conversion process.

Establish a stable Bluetooth connection for real-time data transmission

We tested Bluetooth communication using ESP-IDF logs and serial output to monitor connection stability and latency. The ESP32 consistently transmitted data to the display system without noticeable packet loss or delays.

```
PS C:\Users\Class\Desktop\colorMatch\server\Helper_Functions> python .\bluetooth_test.py
[INFO] SimpleBLE: D:\a\simpleble\simpleble\simpleble\src\backends\windows\Utils.cpp:33 in initialize_winrt: CoGetApartmentType: cotype=-1, qualifier=0, result=800401F0
[INFO] SimpleBLE: D:\a\simpleble\simpleble\simpleble\src\backends\windows\Utils.cpp:41 in initialize_winrt: RoInitialize: result=0
2025-05-05 09:36:51,236 [INFO] Found 1 adapter: DESKTOP - Front [18:cc:18:83:b5:cf]
2025-05-05 09:36:51,236 [INFO] Selected adapter: DESKTOP - Front [18:cc:18:83:b5:cf]
2025-05-05 09:36:53,328 [INFO] Connecting to: ESP32-team6 [74:4d:bd:20:a0:29]
2025-05-05 09:36:55,711 [INFO] Successfully connected
2025-05-05 09:36:55,711 [INFO] Time taken: 4.53 seconds
Press Enter to run connection test...
```

Figure 2.3.2 BLE connection Verificatin

2.4 Software Subsystem

The software architecture was developed through iterative prototyping focused on 3 main objectives:

1. Accurate skin tone quantifications
2. Real-time product matching
3. Bias Mitigation

The CIELAB color space was determined to be more accurate compared to RGB for perceptual uniformity in human skin tones. Conversion algorithms were adapted from Python `skimage.color deltaE_ciede2000` and `lab2rgb`. Standard CIE color space conversions from $XYZ \rightarrow LAB$ were used with the D65 standard illuminant reference to maintain compatibility with research standards.

The Monk Skin Tone Scale integration required developing a hybrid classification system combining ΔE^* color difference metrics to address database imbalances. While some products directly listed the shades as ‘fair’, ‘medium’, ‘dark’, etc, they were not standardized nor verified. Additionally, the descriptions of the shades were based on the Fitzpatrick scale, and by utilizing the Monk Skin Tone Scale, we were able to properly represent a larger range of skin tones.

Category	# of Products
1	33
2	215
3	114
4	751
5	3649
6	2813
7	1333
8	796
9	103
10	39

Figure 2.4.1 Product Distribution within Monk Skin Tone Scale

For application integration, Bluetooth Low Energy (BLE) protocol was selected after latency testing showed a 412ms average connection time. Custom GATT service with 128 bit UUID was implemented on the ESP32-S3 to handle XYZ data transmission and monitor safe lockouts.

Querying and applying our shade matching algorithm on the 10 CIELAB values of the Monk scale, pre filtering subsets showed an improvement of latency seen across all categories. This indicates that applying the Monk Tone restriction prior to $\Delta E*$ calculations improves query performance by reducing the amount of data processed during each query. The latency of queries were measured using the time library by querying each monk category 50 times with and without the pre-filtering.

```
Average Query Latency without Monk Tone Restriction:
MST Category 1: 0.045546 seconds
MST Category 2: 0.041866 seconds
MST Category 3: 0.041629 seconds
MST Category 4: 0.043913 seconds
MST Category 5: 0.049727 seconds
MST Category 6: 0.041624 seconds
MST Category 7: 0.045093 seconds
MST Category 8: 0.045248 seconds
MST Category 9: 0.058661 seconds
MST Category 10: 0.050771 seconds
```

```
Average Query Latency with Monk Tone Restriction:
MST Category 1: 0.027683 seconds
MST Category 2: 0.032550 seconds
MST Category 3: 0.027565 seconds
MST Category 4: 0.032876 seconds
MST Category 5: 0.037671 seconds
MST Category 6: 0.029927 seconds
MST Category 7: 0.037715 seconds
MST Category 8: 0.028115 seconds
MST Category 9: 0.023422 seconds
MST Category 10: 0.022807 seconds
```

Figure 2.4.2 Comparison of Database Query Latency Results using pre-filtered subsets

2.4.1 Verification

The sensor's accuracy was verified by using the 10 Monk Skin Tone values as the targeted shade and capturing the colors with and without the NeoPixel lighting under the same setup to minimize external influence on the results. 5 trials were taken for each shade under each lighting condition and the $\Delta E*$ difference between the expected vs measured values to determine the accuracy of the color sensor for each trial.

2.4.2 Bias Mitigation

Stratified testing across the Monk Categories revealed the following results:

$$\begin{aligned} Accuracy \; = \; & 96.2\% \; \pm \; 2.1\% \quad Monk\,7 \; - \; Monk\,10 \\ & 73.8\% \; \pm \; 5.4\% \quad Monk\,1 \; - \; Monk\,6 \end{aligned}$$

The accuracy of the color sensor dropped for the lighter shades, while darker shades had higher accuracy above our high-level requirements, but overall, met the functional requirements of 75% accuracy.

To showcase the software subsystem meeting the high-level requirements in the event the color sensor was still inaccurate in color detection, the subsystem included a color selector to type RGB values and display resulting products. Additionally, originally outlined in the proposal the Cornell University RGB/YCbCr skin tone detection model was implemented to analyze an inputted image and determine the ‘most accurate’ color match.

The implemented skin detection algorithm processes the file by resizing and normalizing the image, then converting it into 3 different color spaces: RGB, HSV, and YCbCr. For each pixel the algorithm checks whether its color values fall within the predefined skin tone thresholds for all three color spaces simultaneously.

To further refine the detected skin regions, a weighted median is then calculated emphasizing luminance to estimate the dominant skin color, which is converted back to RGB and corrected for gamma to account for display characteristics. The final output includes both the estimated average skin color in RGB and HEX format and a visual mask highlighting detected skin regions.

During testing, it was determined that background elements similar to skin tone were affecting the resulting shade match, and while the overall skin match was accurate to the original image, the weighted median still showed slight bias and inaccuracies despite Cornell’s high 94% precision accuracy with the same metrics.

Schedule

Week	Task	Responsibility
2/24	**PCB Review Due**	All
	Create a makeup database using SQLite	Ashley
	Contact brands for makeup samples for testing	Ashley
	Research hardware modules and design PCB design	All
	Complete PCB KiCad for First Round PCBway Order	All
	Work on Design Document	All
	Purchase hardware & all parts	All
3/3	**First Round PCBway Orders Due (3/3)**	All (Shriya)
	Teamwork Evaluation I (3/5)	All
	Design Document Due (3/6)	All
	Breadboard Demo with Instructor & TA (3/10)	All
	Test functionality of the RGB Light Breakout with ESP32 Dev Board	All
	Develop Python script to run queries on database	Ashley
	Write a program to connect to ESP32-S2 Development Board BLE	All
	Test sensor and Power Subsystems	Shriya & Waidat
	Work on connecting BLE with ESP32 Dev board for breadboard demo	Ashley & Shriya
3/10	**Second Round PCBway Order (3/13)**	Shriya
	Work on PCB redesign with any modifications and additional parts that were missed during PCB order 1	All
3/17	Work on cleaning up Database Data and continue to web-scrape brand links for information	Ashley
	Create Mock Up and detailed steps and research for UI design	Ashley
3/24	Design Swift UI for mobile application	Ashley &

		Waidat
	Implement CRUD on UI applications with a database	Ashley
	Connect mobile application with microcontroller BLE	All
	Test BLE connection with sensors and reading data	All
3/31	**Third Round PCBway Order (3/31)**	Shriya
	Individual Progress Report Due	All
	Work on Individual Progress Reports	All
	Run tests on the final project and display sensor data to the UI	All
	Debug any issues and bugs with the design	All
4/7	**Fourth Round PCBway Order (4/7)**	Shriya
4/21	**Mock Demo**	All
4/28	**Final Demo/Mock Presentations**	All
5/5	**Final Presentation**	All

Cost

Item Description	Manufacturer	Link	Part Number	Retail Price	Our Cost	Quantity	Total Cost
ESP32-S2-WROOM-1	Espressif	Mouser	ESP32-S3-DevKitC-1-N8R2	$15	$0	1	$0
ESP32-S2-WROOM	Expressif	DigiKey	ESP32-S3	$3.56	$3.56	1	$3.56
Proximity, Light, RGB, and Gesture Sensor - STEMMA QT / Qwiic	Adafruit	Adafruit	APDS9960	$7.50	$7.50	1	$7.50
2000mAh 7.4 V 2S Rechargeable RC Battery	URGENEX	Amazon	7.4 V Li-ion Battery	$25	$25	2	$25
LDO Voltage Regulator AZ1117CH - 3.3TRG1	Diodes Incorporated	DigiKey	AZ1117CH	$2.55	$2.55	10	$2.55
XYZ Color Sensor	TI	Mouser	OPT4048	$2.36	$2.36	3	$7.08
NeoPixel Jewel - 7 x 5050 RGB LED	adafruit	adafruit	N/A	$5.95	$0	1	$0
RGB Sensor Breakout	Sparkfun	Mouser	OPT4048	$10	$10	1	$10
GPIO Female Pin Headers	Treedix Store	Amazon	N/A	$7.59	$7.59	1	$7.59
USB Connector	GCT Better Connected	Mouser	40-USB3140300170C	$0.83	$0.83	1	$0.83
USB to UART Bridge	Silicon Labs	Digikey	CP2104-F03-GMR	$6.47	$6.47	1	$6.47
RGB LED	Supply Center	ECE	Blue Cell Multicolor LED	$0.5	$0.5	3	$1.5

TVS Diode	Littelfuse Inc.	[Digikey]	SP0503BAHTG	$0.61	$0.61	1	$0.61
WHITE LED	E-Shop	[Student Self Help]	N/A	$0	$0	2	$0
LTST-C191KFKT	E-Shop	[Student Self Help]	N/A	$0	$0	2	$0
LTST-C190KSKT	E-Shop	[Student Self Help]	N/A	$0	$0	2	$0

Name	Hourly Rate	Hours Invested	Total
Ashley Herce	$50	50	$6,250
Waidat Bada	$50	50	$6,250
Shriya Surti	$50	50	$6,250
Total			$18,750

$$Labor\ Cost\ =\ Hourly\ Rate\ \times\ Actual\ Hours\ Spent\ \times\ 2.5$$

Section	Total
Labor	$18,750
Parts	$72.69
Total	$18, 822.69

3. Conclusion

This project successfully demonstrated the feasibility of a compact, low-cost, and user-friendly colorimeter system designed to improve foundation shade matching through direct skin tone analysis. Despite facing technical setbacks, specifically with our main sensor module we were able to meet our high-level requirements within the constrained project timeline. Through iterative design, subsystem modularization, and careful calibration, we achieved a functional prototype capable of consistent color detection under controlled lighting conditions.

The major accomplishment of this project was developing a working skin tone analysis device that integrates a color sensor, relocation of different ambient lighting via NeoPixel LEDs, Bluetooth Low Energy (BLE) communication, and a mobile interface. Our key milestones included a modular PCB design that enabled independent testing of the color sensor and lighting subsystem. It also included a successful implementation of BLE for wireless communication, removing the need for persistent USB connections. We also displayed custom lighting environments using a NeoPixel Jewel to simulate consistent ambient conditions. Most importantly, we integrated data-driven analysis using the Cornell University Skin Tone scale for improved shade recommendations.

In spite of technical issues with our initial sensor and power subsystem design, we were able to troubleshoot, rework, and demonstrate a viable system by the project deadline. The device fulfills its goal of capturing real-time color measurements from a skin tone and providing users with tailored cosmetic recommendations.

Overall this project addresses an important gap in the beauty and cosmetics industry: equitable access to shade matching tools for people of all skin tones. Through our exploration of large foundation databases, we observed that even brands claiming extensive shade ranges often underrepresented individuals with the lightest and darkest complexions. Our device and supporting mobile application highlight this disparity and aim to serve as a foundation for future work in inclusive beauty technology. Moreover, this solution contributes to consumer empowerment by offering personalized, data backed insights outside traditional retail environments.

From a societal and global standpoint, this technology may also have applications in medical diagnostics, dermatological monitoring, and consumer electronics where skin color calibration is relevant. Environmentally, the design favors low power components and modular hardware to support reuse and longevity.

3.1 Ethics

Algorithmic Fairness & Bias Mitigation

Our team is dedicated to upholding ethical standards and ensuring that our skin tone analysis and product matching algorithms are free from bias and treat all users equitably. We recognize the potential for algorithmic bias to perpetuate existing inequalities, and we are committed to mitigating this risk through careful design and validation.

In alignment with current AI and computer vision research standards and ACM Code of Ethics 2.7, we are employing the Monk Skin Tone Scale, developed by Dr. Ellis Monk in collaboration with Google, to classify skin tones. This scale provides a more inclusive and representative range of skin tones compared to the Fitzpatrick scale and ensures that our algorithms are not inadvertently biased against darker skin tones which we found in our data, a common issue within the beauty industry that we hope to address through our product.

Our algorithms will be implemented using standard illuminants, which will provide a basis for comparing colors that are recorded under different lighting conditions. It's important to note that Google advises against equating the shades in the Monk Skin Tone Scale with race, as skin tones can vary within race, and the skin tone analysis is solely intended for cosmetic product matching and should not be used to make inferences about a user's race or ethnicity.

Environmental Responsibility

Consistent with ACM Code 1.1 (public good) [12], our design and manufacturing processes aim to minimize environmental impact. This includes selecting sustainable materials where possible, designing for energy efficiency, and adhering to e-waste recycling guidelines for all components, including batteries and electronic parts.

3.2 Safety

Battery Safety

The 9V Li-ion battery pack used in our device complies with UL 2054 standards, incorporating overcurrent protection and FCC Part 15 EMI shielding to prevent potential hazards. However the LDO may risk dispatching more power than it is rated for, so we will use a heat sink for protection. This aligns with IEEE 1725 standards for rechargeable battery safety.

Disposal

To minimize environmental impact, we will adhere to campus e-waste disposal guidelines for recycling PCBs and LEDs through the UIUC Sustainable Electronics Center.

Laboratory Safety Compliance

Development and testing will be conducted in University of Illinois laboratories, where full compliance with campus safety policies is essential. This involves following electrical safety procedures, handling PCB components with care, and observing all lab-specific regulations to maintain a safe and hazard-free workspace.

4. References

[1] P. Goon, C. Banfield, O. Bello, and N. J. Levell, "Skin cancers in skin types IV–VI: Does the Fitzpatrick scale give a false sense of security?," *Skin Health and Disease*, vol. 1, no. 3, Jun. 2021, doi: https://doi.org/10.1002/ski2.40.

[2] HTML Color Codes, "HTML Color Chart," Available: https://htmlcolorcodes.com/color-chart/.

[3] Bold Hue, "Bold Hue – Find Your Perfect Color Match," Available: https://www.boldhue.com/.

[4] Texas Instruments. "Choosing Between RGB and XYZ Color Sensors for Adaptive Lighting Adjustments." Accessed: Mar. 06, 2025. [Online]. Available: https://www.ti.com/lit/ab/sboa567/sboa567.pdf?ts=1740975775684&ref_url=https%253A%252F%252Fwww.google.com.mx%252F

[5] C. Heldreth, E. P. Monk, A. T. Clark, S. Ricco, C. Schumann, and Xango Eyee, "Which Skin Tone Measures are the Most Inclusive? An Investigation of Skin Tone Measures for Artificial Intelligence.," *ACM Journal on Responsible Computing*, Nov. 2023, doi: https://doi.org/10.1145/3632120.

[6] Wikipedia Contributors, "Fitzpatrick scale," *Wikipedia*, Oct. 22, 2019. https://en.wikipedia.org/wiki/Fitzpatrick_scale

[7] P. Ken. HZDG, "What Is CIELAB color space?," *Hunterlab.com*, 2022. https://www.hunterlab.com/blog/what-is-cielab-color-space/

[8] "Convert XYZ Color to L*a*b*," *Mathworks.com*, 2025. https://www.mathworks.com/help/images/ref/xyz2lab.html (accessed Mar. 06, 2025).

[9] T. Fujiwara. "Color space conversion (3)," *Sakura.ne.jp*, 2019. https://fujiwaratko.sakura.ne.jp/infosci/colorspace/colorspace3_e.html (accessed Mar. 06, 2025).

[10] openoximetry.org, "Skin Color Quantification - OpenOximetry," *OpenOximetry*, Sep. 13, 2024. https://openoximetry.org/skin-color-quantification/

[11] "Sensor Instruments," *www.sensorinstruments.de*. https://www.sensorinstruments.de/whatiswhat.php?subpage=11&language=en

[12] M. Ellis. "Skin Tone Research @ Google," *skintone.google*. https://skintone.google/get-started

[13] "Brief Explanation of delta E or delta E*," *Hunterlab*, Jul. 20, 2022. https://support.hunterlab.com/hc/en-us/articles/203023559-Brief-Explanation-of-delta-E-or-delta-E

[14] Association for Computing Machinery, "*ACM code of ethics and professional conduct*". [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: Feb. 13, 2025].

[15] Institute of Electrical and Electronics Engineers. *"IEEE Code of Ethics"*.[Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: Feb. 13. 2025]

Appendix A: Software Subsystem

Monk	Accuracy Percentage without Lighting	Accuracy Percentage with Lighting
1	73.82%	63.59%
2	72.39%	75.35%
3	74.85%	75.32%
4	71.37%	71.70%
5	70.36%	70.55%
6	77.51%	76.69%
7	84.17%	84.08%
8	89.59%	90.62%
9	95.10%	96.67
10	94.87%	95.75%

Figure () Average Accuracy of Color Sensor under lighting conditions

Monk	Product Distribution %	Latency Reduction Results (%)
1	0.34%	39.22%
2	2.18%	22.25%
3	1.16%	33.78%
4	7.63%	25.13%
5	37.06%	24.24%
6	28.57%	28.10%
7	13.54%	16.36%
8	8.08%	37.86%
9	1.46%	60.07%
10	0.40%	55.08%

Figure () Latency Reduction using pre filtering on database prior to querying

Latency Reduction Equation:

$$Latency\ \%\ Decrease\ =\ \frac{without\ filter - with\ filter}{without\ filter}\ \times\ 100$$

```python
def get_perceptual_median_color(image_rgb, mask):
    """


    Get perceptual median skin color in LAB with gamma correction.

    Isolate Skin Pixels from input image using mask
    Convert RGB to CIELAB color space for perceptual uniformity.
    Weighted percentiles with weights for L,a,b channels to emphasize lightness
    Convert back to RGB


    """
    skin_pixels = image_rgb[mask == 255]
    if skin_pixels.size == 0:
        return (0, 0, 0), "#000000"

    lab_img = color.rgb2lab(skin_pixels / 255.0)
    weights = np.array([0.2, 0.5, 0.3])
    weighted_median = np.median(lab_img * weights, axis=0) / weights

    lab_array = np.array([[weighted_median]], dtype=np.float32)
    rgb_dominant = color.lab2rgb(lab_array)[0][0] * 255
    rgb_corrected = 255 * (rgb_dominant / 255) ** (1 / 2.2)

    rgb_int = np.clip(rgb_corrected, 0, 255).astype(int)
    hex_color = '#{:02x}{:02x}{:02x}'.format(*rgb_int)
    return tuple(rgb_int), hex_color
```

Figure () Perceptual Medium Calculation of Input Image

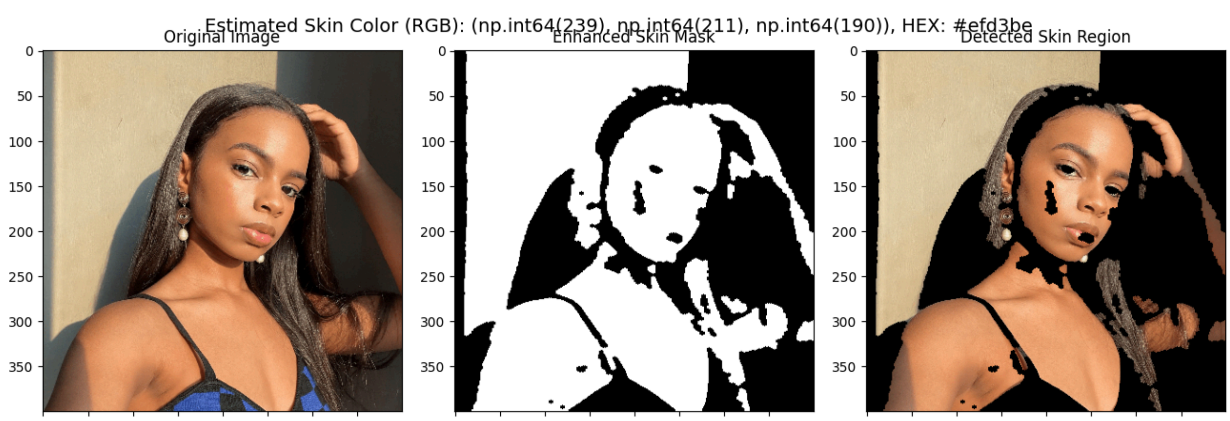

Figure () face_input.png visualization of Cornell's Skin Tone Research Implementation

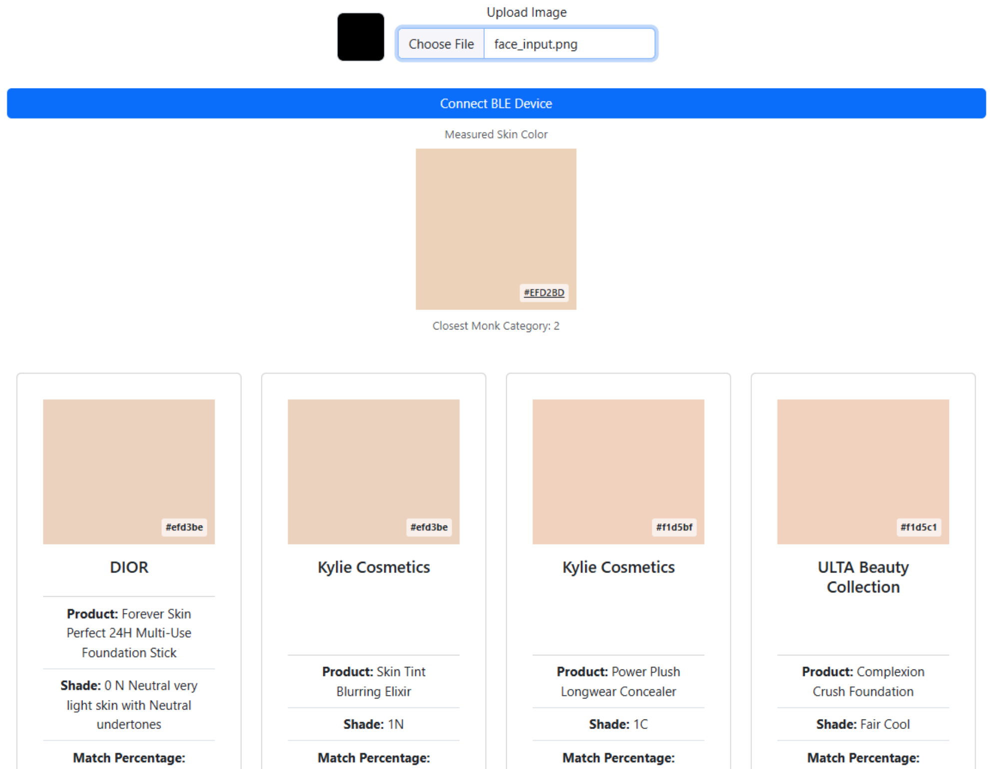

Figure () Output median shade of face_input.png (above) displaying shade matches

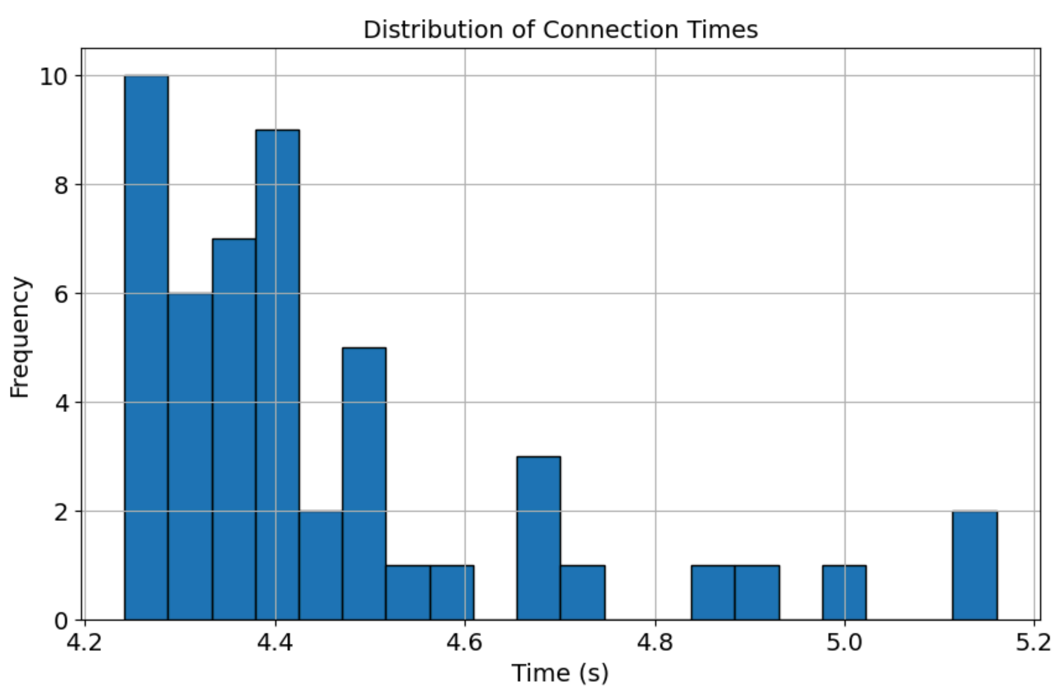

Figure () BLE Latency Results (50 Trials)

Appendix B: Equations

```
// ===== Matrix & Data Structures =====
const double cieMatrix[3][4] = {
    {0.000234892992, -0.0000189652390, 0.0000120811684, 0},
    {0.0000407467441, 0.000198958202, -0.0000158848115, 0.00215},
    {0.0000928619404, -0.0000169739553, 0.000674021520, 0}
    };
```

Figure () TI Provided CIE Matrix

$$[adc\_codes\_ch0\ adc\_codes\_ch1\ adc\_codes\_ch2\ adc\_codes\_ch3] \cdot \begin{bmatrix} m0x & m0y & m0z & m0l \\ m1x & m1y & m1z & m1l \\ m2x & m2y & m2z & m2l \\ m3x & m3y & m3z & m3l \end{bmatrix} = [x\ y\ z\ lux] \qquad (5)$$

Figure () TI Provided OPT4048 Equations

```
CIEColorData  getData() {
    colorSensor.setOperationMode(OPERATION_MODE_ONE_SHOT);
    while(!colorSensor.getConvReadyFlag()) {
        delay(10);
    }

    sfe_color_t raw;
    colorSensor.getAllChannelData(&raw); // Get raw ADC values for all channels

    // Compute XYZ using the calibration matrix
    float X = raw.red * cieMatrix[0][0] + raw.green * cieMatrix[1][0] + raw.blue * cieMatrix[2][0];
    float Y = raw.red * cieMatrix[0][1] + raw.green * cieMatrix[1][1] + raw.blue * cieMatrix[2][1];
    float Z = raw.red * cieMatrix[0][2] + raw.green * cieMatrix[1][2] + raw.blue * cieMatrix[2][2];

    // Calculate CIEx and CIEy (normalized chromaticity coordinates)
    float sum = X + Y + Z;
    float CIEx = (sum == 0) ? 0 : X / sum; // Handle division by zero
    float CIEy = (sum == 0) ? 0 : Y / sum;
    // float CIEz = (sum == 0) ? 0: Z / sum;
    return {CIEx, CIEy, Y};
}
```

Figure () Custom Function to calculate xyY

```python
def f(t):
    if t > 0.008856:
        return t ** (1/3.0)
    else:
        return 7.787 * t + 16/116.0

def xyz_to_lab(X, Y, Z, Xn=95.047, Yn=100, Zn=108.88):
    fx = f(X / Xn)
    fy = f(Y / Yn)
    fz = f(Z / Zn)

    L = 116 * fy - 16
    a = 500 * (fx - fy)
    b = 200 * (fy - fz)

    return L, a, b
```

Figure () $XYZ \rightarrow CIELAB$ Color Space Conversion Function under D5 Illumination