Theta Beta Engineer Project Report

Foundation Color Identifier and Dispenser

University of California, Irvine

Akhil Nandhakumar, Allyson Lay, Dalen Avrin Smith, Elizabeth Yancey, Emma Shin, Harmeet Singh, Ival Momoh, Jay Kim, Richard Tokiyeda, Victoria Sun

December -day-, 2025

Contents

List of Figures

List of Tables

Chapter 1

Abstract

The Foundation Identifier and Dispenser aims to develop a machine that is able to extract samples from a picture of a human to determine the shade of their skin, allowing the machine to dispense a corresponding foundation shade. The results produced should be both accurate and reproducable. Equipped with computer vision libraries and color correction algorithms, the system allows the user to take an picture alongside a reference color sheet, which has colors of known values. These captured values are processed to correct both camera bias, and lighting correction so that results may remain consistent regardless of lighting conditions during image capture. The system converts RGB values to LAB values, which are higher in accuracy in physical color mixing, as opposed to RGB, which is used to describe pixel colors. The program calculates how much of each color is needed to recreate the user's skin pigment. These pigments are then dispensed via a mechanical system comprised of a Raspberry Pi, servo motors, and syringes. This project demonstrates an application of computer vision in the cosmetic market to alleviate the burden of overconsumption and promote inclusivity.

Chapter 2

Introduction

2.1 Background

2.2 Objectives

Chapter 3

System Overview

3.1 Overall Architecture

Chapter 4

Hardware Design and Specifications

4.1 Bill of Materials

Component	Purpose	Quantity	Price
Raspberry Pi 5 (4GB RAM)	Primary controller	1	$66.00
Raspberry Pi 5 Power Supply	Powers Raspberry Pi	1	$15.99
DRV8825 Stepper Driver Module 5PCS	Drives stepper motors	1	$10.56
NEMA 17 Pancake Stepper Motor 5PCS	Pushes lead screws	1	$32.99
Limit Switch 10PCS	Detects plunger	1	$5.99
DC 12V 5A Power Supply	Powers motors	1	$19.99
Arducam V3 Camera	Captures samples	1	$25.00
Wiring	Connects electrical	Bulk	$5.00
Breadboard / Perfboard	Circuit layout	1	$10.00
Lead Screw with T8 Brass Nut 2PCS	Pushes on syringe	3	$13.99
Lead Screw Coupler 5PCS	Connects shaft to screw	1	$15.99
Linear Motion Rod Shaft Guide 2PCS	Syringe alignment	3	$8.69
Lead Screw Pillow Block Bearings 3PCS	Rotational support	2	$8.99
Plastic Syringes 5PCS	Dispensing containers	1	$6.99
Total Estimated Cost			**$315.53**

Table 4.1: Key Hardware Elements.

4.2 Sensors, Actuators, Drivers

4.3 Electrical Connections and Power Distributions

4.4 CAD Models

4.5 Methodology

Chapter 5

Software Design

5.1 Techstack

5.2 Control Logic

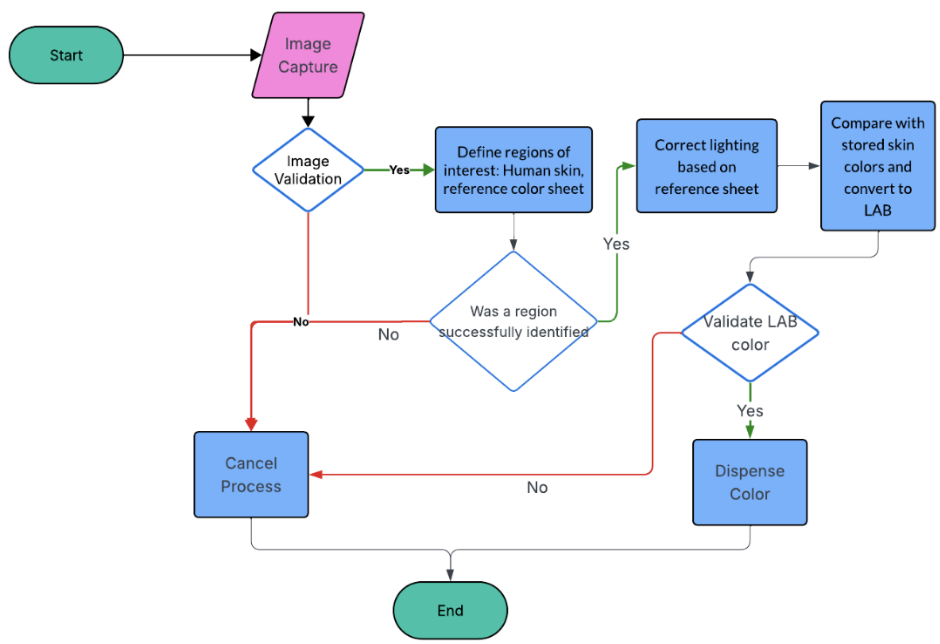

Figure 5.1: User Flow Diagram

5.3 Computer Vision Pipeline

5.4 Color Correction Algorithm

5.5 Foundation Formulation

5.6 Raspberry Pi Integration

5.7 User Interface

5.8 User Manual

Chapter 6

Electrical Systems Design

6.1 Circuit Schematic and Breadboard Design

6.2 Signal Conditioning and Safety Considerations

Chapter 7

Results and Analysis

7.1 Testing Protocol

7.2 Color Accuracy

7.3 System Performance

Chapter 8

Discussion

8.1 Limitations

8.2 Future Work

Chapter 9

Conclusion

References

Appendix A

Pseudocode Listings

````
# 1: Color Bias Adjustment and Sampling
```python
def IMAGE_PROCESSING(SAMPLE_PICTURE) :
    #INPUT: SAMPLE_PICTURE - image captured by camera
    #OUTPUT: LAB_values - color values compatible with paint mixing

    ### Get/Validate image containing skin region
    ### & reference colors/control sheet

    if SAMPLE_PICTURE is EMPTY/NOT_DETECTED :
        RAISE_ERROR "Image captured failed or canceled"
        return NULL

    # initial samples contain gamma-corrected RGB values
    # use OpenCV region of interest rectangle

    SKIN_SAMPLE = list of pixel RGB values that constitute a skin region from SAMPLE_PICTURE
    CONTROL_SAMPLE = list of pixel RGB values that encompass the reference color sheet SAMPLE_PICTURE

    # check whether the reference sheet is present or the image is too dark/light

    if CONTROL_SAMPLE is EMPTY/NOT_DETECTED :
        RAISE_ERROR "Contol sheet not detected. Take picture with color reference sheet in a well-lit room."
        return NULL

    # get average gamma-corrected RGB codes for the skin region and control

    SKIN_RGB_AVG = calculate average of SKIN_SAMPLE
    CONTROL_MEASURED_VALUES = list of averages of CONTROL_SAMPLE

    # get linear RGB codes from gamma-corrected RGB codes
    # allows linear operations to be performed on the codes

    SKIN_LINEAR_RGB = apply reverse gamma-correction to SKIN_RGB_AVG
    CONTROL_LINEAR_RGB = list of linearized RGB codes from CONTROL_MEASURED_VALUES
````

````
    # find difference in the control input RGB codes and stored RGB codes

    CONTROL_KNOWN_VALUES = load("CONTROL_DATABASE.csv") and linearize the codes
    CONTROL_TRANSFORM = find transformation matrix that makes CONTROL_LINEAR_RGB = CONTROL_KNOWN_VALUES * CONTROL_TRANSFORM

    # apply difference to the values found in the skin region for lighting correction

    XYZ = apply CONTROL_TRANFORM to SKIN_LINEAR_RGB

    # convert rgb value to lab for later processing

    LAB_VALUES = convert XYZ color space to LAB(XYZ)

    for element in LAB_VALUES :
        if element is OUT_OF_RANGE :
            RAISE_ERROR "color conversion error. take a new picture."
            return NULL

    return LAB_VALUES
````

Figure A.1: Pseudocode: Image Processing Algorithms.
````

Appendix B

Wiring & Safety

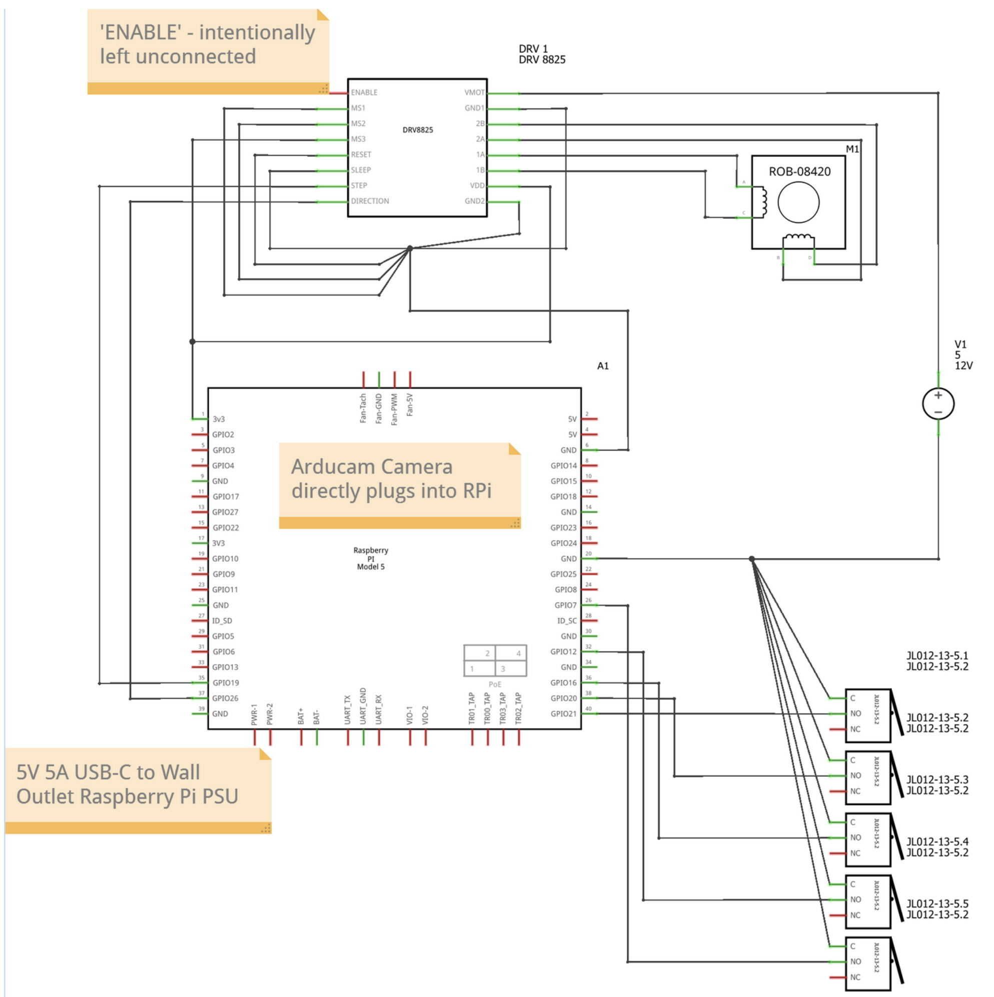

Figure B.1: Wiring Diagram

Appendix C

Data Schemas