

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

November 17, 2025

¹ Contents

² 1 Abstract	¹
³ 2 Introduction	²
⁴ 2.1 Research	²
⁵ 2.1.1 Background of Problem	²
⁶ 2.1.2 Existing Solutions	²
⁷ 2.2 Design Choices	³
⁸ 2.2.1 Past Considerations and Scrapped Plans	³
⁹ 3 Hardware Design and Specifications	⁴
¹⁰ 3.1 First Iteration	⁵
¹¹ 3.1.1 Initial CAD Model and Hand Sketches	⁵
¹² 3.1.2 Commentary	⁷
¹³ 3.2 Second Iteration	⁷
¹⁴ 3.2.1 Updated CAD Model	⁷
¹⁵ 3.2.2 Commentary	⁷
¹⁶ 3.3 Third Iteration	⁷
¹⁷ 3.3.1 Final CAD Model	⁷
¹⁸ 3.3.2 Final Manufacturing Drawings for Custom Parts	⁷
¹⁹ 3.3.3 Commentary	⁷
²⁰ 4 Software Design	⁸
²¹ 4.1 First Iteration	⁸
²² 4.1.1 Preliminary Diagrams	⁸
²³ 4.1.2 User Flow Diagram	¹¹
²⁴ 4.1.3 Techstack	¹¹
²⁵ 4.2 Second Iteration	¹¹
²⁶ 4.2.1 Lo-fi/Mid-fi Designs	¹¹
²⁷ 4.2.2 Basic Logic and Functionality	¹¹
²⁸ 4.2.3 Core Features of Algorithm	¹¹
²⁹ 4.3 Third Iteration	¹¹
³⁰ 4.3.1 Final Product	¹¹
³¹ 5 Electrical Systems Design	¹²
³² 5.1 First Iteration	¹³
³³ 5.1.1 Initial Wiring Diagrams	¹³

34	5.1.2 Initial Power Calculations	14
35	5.2 Second Iteration	14
36	5.2.1 Circuit Building	14
37	5.3 Third Iteration	14
38	5.3.1 Final Wiring Diagrams	14
39	5.3.2 Final Power Calculations	14
40	5.3.3 Circuitry	14
41	6 Final Product	15
42	6.1 Testing Protocol	15
43	6.2 Integration	15
44	6.3 System Performance	15
45	7 Discussion	16
46	7.1 Limitations	16
47	7.2 Future Work	16
48	8 Conclusion	17
49	8.1 Bill of Materials	17
50	References	18

51 List of Figures

52	3.1 First Sketch: Housing	5
53	3.2 First CAD Models: Housing	6
54	3.3 First Sketch: Dispenser Systems	6
55	3.4 First CAD Models: Dispenser Systems	7
56	4.1 Initial Flowchart for Control Logic	8
57	4.2 Pseudocode: Image Processing Algorithms.	9
58	4.3 Pseudocode: Embedded System.	10
59	4.4 UI/UX - Landing page.	10
60	4.5 UI/UX - Loading the foundation shades.	11
61	5.1 Preliminary Wiring Diagram	13

62 List of Tables

63	5.1 Power Calculations	14
64	8.1 Key Hardware Elements.	17

⁶⁵ **Chapter 1**

⁶⁶ **Abstract**

⁶⁷ The Foundation Identifier and Dispenser aims to develop a machine that is able to extract
⁶⁸ samples from a picture of a human to determine the shade of their skin, allowing the machine
⁶⁹ to dispense a corresponding foundation shade. The results produced should be both accurate
⁷⁰ and reproducible. Equipped with computer vision libraries and color correction algorithms,
⁷¹ the system allows the user to take an picture alongside a reference color sheet, which has
⁷² colors of known values. These captured values are processed to correct both camera bias,
⁷³ and lighting correction so that results may remain consistent regardless of lighting conditions
⁷⁴ during image capture. The system converts RGB values to LAB values, which are higher in
⁷⁵ accuracy in physical color mixing, as opposed to RGB, which is used to describe pixel colors.
⁷⁶ The program calculates how much of each color is needed to recreate the user's skin pigment.
⁷⁷ These pigments are then dispensed via a mechanical system comprised of a Raspberry Pi,
⁷⁸ servo motors, and syringes. This project demonstrates an application of computer vision in
⁷⁹ the cosmetic market to alleviate the burden of overconsumption and promote inclusivity.

⁸⁰ **Chapter 2**

⁸¹ **Introduction**

⁸² **2.1 Research**

⁸³ **2.1.1 Background of Problem**

⁸⁴ Testing foundation colors can be a frustrating experience for many, who are unable to find
⁸⁵ the perfect balance. At the end of the day, no line of foundation can realistically provide
⁸⁶ colors that cater to every possible skin tone. The seemingly unresolvable desire for a perfect
⁸⁷ shade leads many makeup users to spend hundreds on shades that are "close enough." This
⁸⁸ leads to lots of waste, not just in money, but in bottles thrown out after purchase because
⁸⁹ the match was ultimately unsatisfying.

⁹⁰
⁹¹ The beauty industry produces 120 billion packaging units per year and 95% of this these
⁹² units are discarded, as opposed to recycled [1]. In addition, traditionally a custom skin
⁹³ matched foundation can cost anywhere from \$60-100 per bottle, which leaves the consumers
⁹⁴ with the dilemma of whether they should take the gamble on the bottle that's just almost
⁹⁵ right versus breaking their wallet on a hand matched bottle. Our custom mixer machine
⁹⁶ offers the same accuracy in skin tone while also promoting cheaper makeup, sustainability,
⁹⁷ and waste-reduction.

⁹⁸
⁹⁹ Our foundation color picker abandons the concept of creating a set of discrete skin tones
¹⁰⁰ to choose from, instead, opting for custom mixed shades depending on the skin color detected
¹⁰¹ by a picture. It also offers the unique ability to sample a shade without having to commit
¹⁰² to a full sized bottle.

¹⁰³ **2.1.2 Existing Solutions**

¹⁰⁴ BoldHue provides an option for an AI powered foundation mixer. It matches skin tone
¹⁰⁵ through a smart wand that detects color. What they promised was millions of shades and
¹⁰⁶ the ability to store user's shades in profiles. However, their color detection methods provide
¹⁰⁷ room for lots of error because color isn't perceived the same depending on the lighting of the
¹⁰⁸ room, as well as the high cost of their machine.

¹¹⁰ Our product will have built in lighting correction that removes biases from the camera
¹¹¹ and uses the Macbeth Color Reference sheet as a set of control colors. The reference sheet
¹¹² is what will allow the machine to recalibrate it's understanding of what colors are being
¹¹³ percieved.

¹¹⁴ **2.2 Design Choices**

¹¹⁵ **2.2.1 Past Considerations and Scrapped Plans**

¹¹⁶ Chapter 3

¹¹⁷ Hardware Design and Specifications

¹¹⁸ 3.1 First Iteration

¹¹⁹ 3.1.1 Initial CAD Model and Hand Sketches

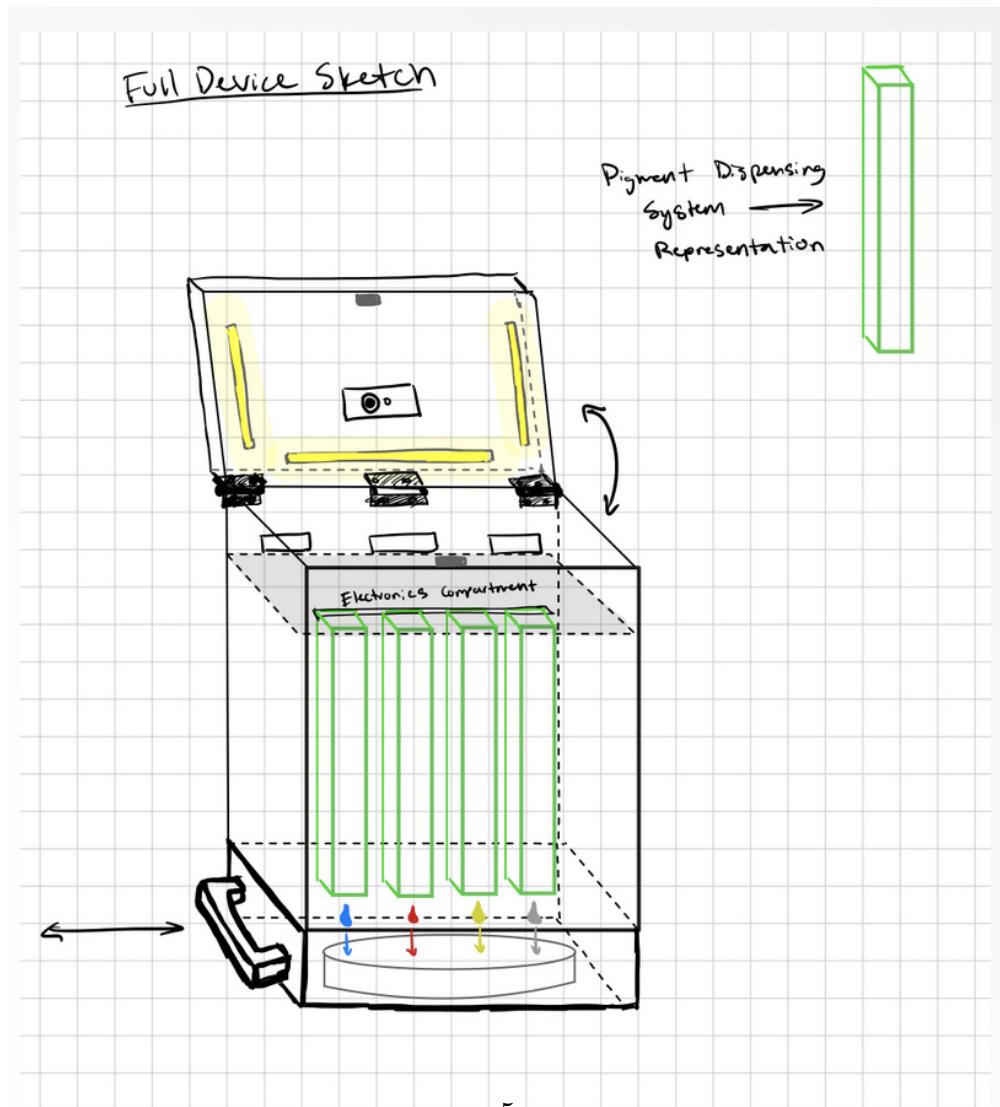


Figure 3.1: First Sketch: Housing

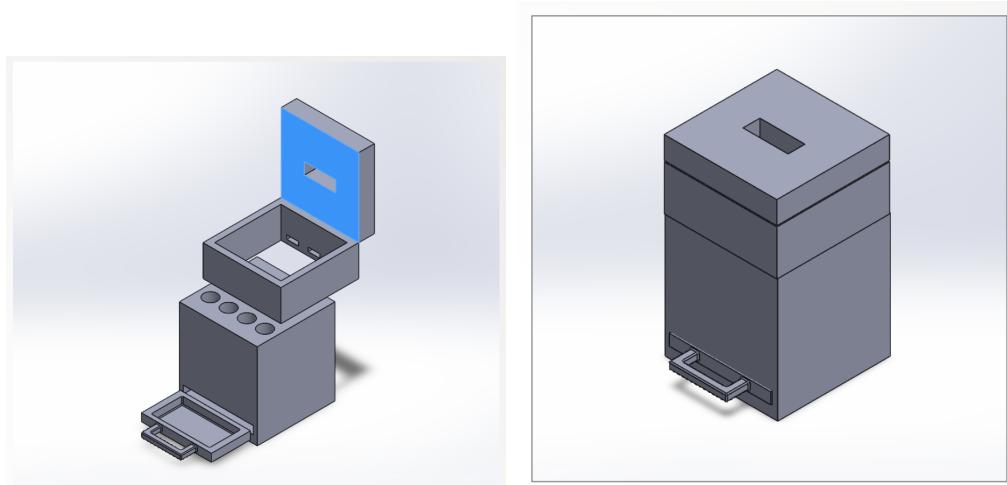


Figure 3.2: First CAD Models: Housing

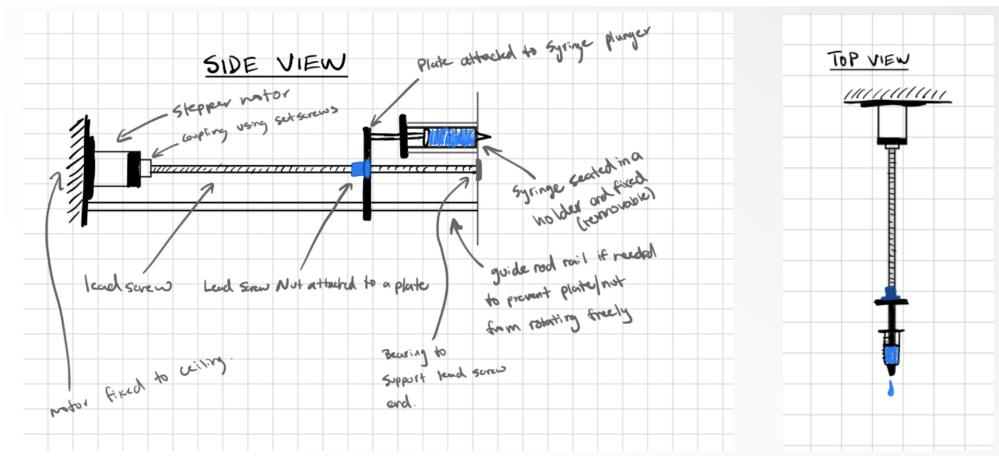


Figure 3.3: First Sketch: Dispenser Systems



Figure 3.4: First CAD Models: Dispenser Systems

¹²⁰ **3.1.2 Commentary**

¹²¹ **3.2 Second Iteration**

¹²² **3.2.1 Updated CAD Model**

¹²³ **3.2.2 Commentary**

¹²⁴ **3.3 Third Iteration**

¹²⁵ **3.3.1 Final CAD Model**

¹²⁶ **3.3.2 Final Manufacturing Drawings for Custom Parts**

¹²⁷ **3.3.3 Commentary**

₁₂₈ Chapter 4

₁₂₉ Software Design

₁₃₀ 4.1 First Iteration

₁₃₁ 4.1.1 Preliminary Diagrams

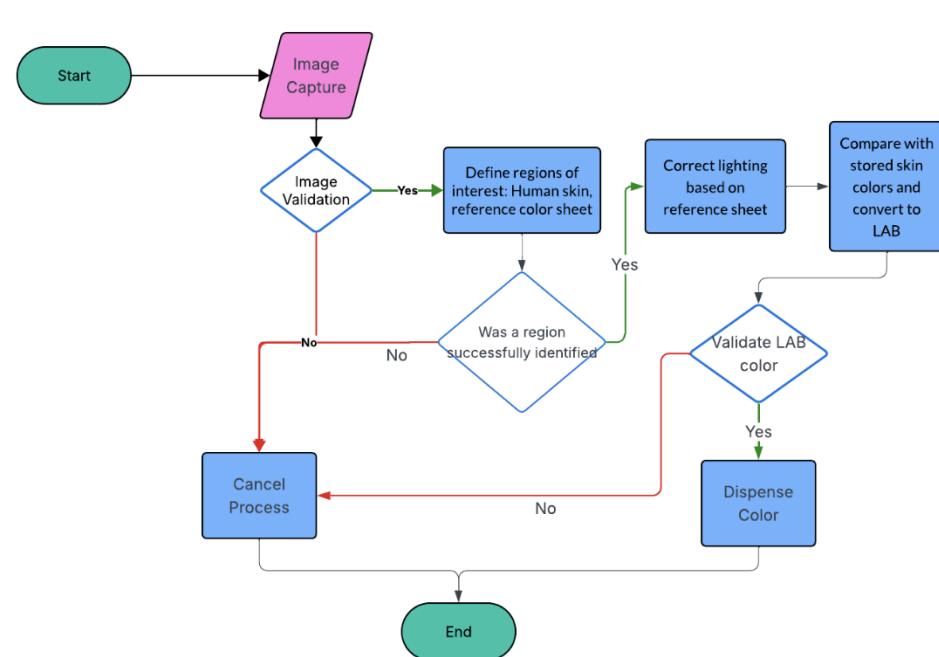


Figure 4.1: Initial Flowchart for Control Logic

```

# 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 "Control 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_TRANSFORM 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 4.2: Pseudocode: Image Processing Algorithms.

```
# 2: Raspberry Pi Code

```python
def EVENT_HANDLER():
 #extract lab values
 SAMPLE_PICTURE = CAMERA_CAPTURE initiated by BUTTON_PRESS
 LAB_VALUES = IMAGE_PROCESSING(SAMPLE_PICTURE)

 #calculate servo turns for desired recipe
 PAINT_QUANTITIES = assign paint quantities indicating how many
 | | | | | times the servo should turn with TARGET_SHADE

 DEPLOY_TO_RASPBERRY_PI(PAINT_QUANTITIES)
 #servo motors move syringes
```

```

Figure 4.3: Pseudocode: Embedded System.

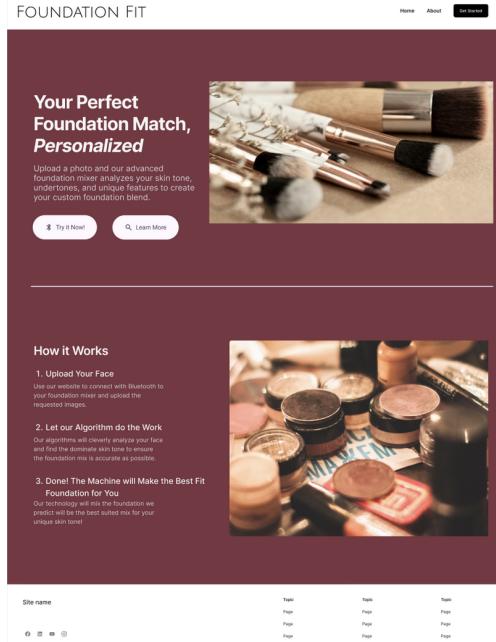


Figure 4.4: UI/UX - Landing page.

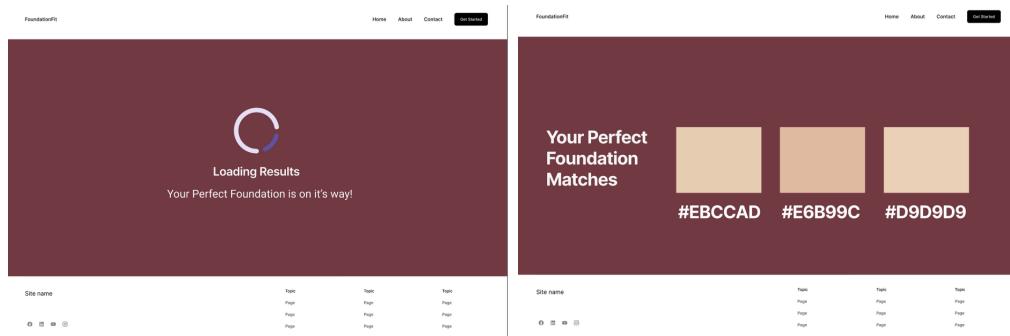


Figure 4.5: UI/UX - Loading the foundation shades.

¹³² 4.1.2 User Flow Diagram

¹³³ 4.1.3 Techstack

¹³⁴ Operating systems: Linux on Raspberry Pi

¹³⁵ Libraries: OpenCV, NumPy, Pandas

¹³⁶ Frameworks: React, Flask

¹³⁷ Languages: Python

¹³⁸

¹³⁹ 4.2 Second Iteration

¹⁴⁰ 4.2.1 Lo-fi/Mid-fi Designs

¹⁴¹ 4.2.2 Basic Logic and Functionality

¹⁴² 4.2.3 Core Features of Algorithm

¹⁴³ 4.3 Third Iteration

¹⁴⁴ 4.3.1 Final Product

¹⁴⁵ Chapter 5

¹⁴⁶ Electrical Systems Design

¹⁴⁷ 5.1 First Iteration

¹⁴⁸ 5.1.1 Initial Wiring Diagrams

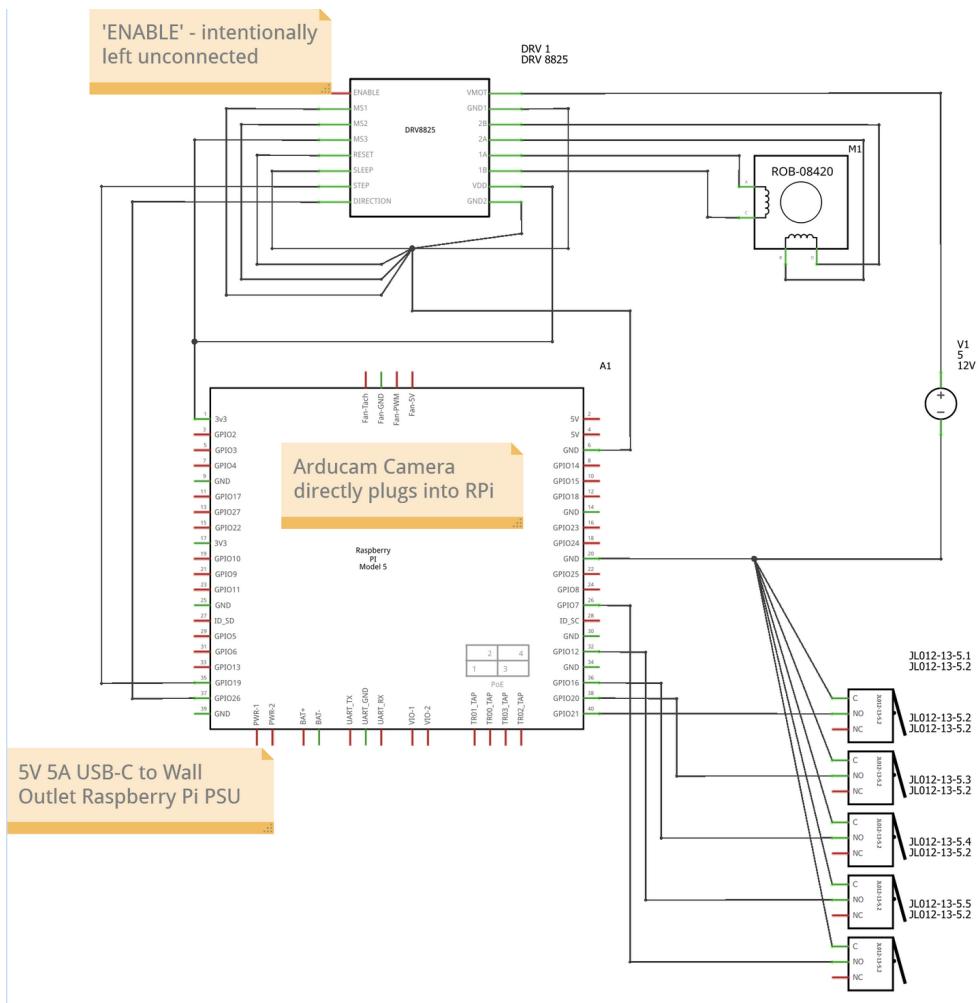


Figure 5.1: Preliminary Wiring Diagram

149 5.1.2 Initial Power Calculations

| Parameter | Symbol | Value |
|------------------|--------|-------|
| Phase Current | I | 0.7 |
| Phase Resistance | R | 4.0 |
| Phases | - | 2 |

Table 5.1: Power Calculations

150 $P_{motor} = 2 * I^2 * R$

151 $P_{motor} = 2 * (0.7)^2 * 4.0 = 3.92 * 5 = 19.6W$

152 $P_{RaspberryPi} = 10W$

152 $P_{total} = 10 + 19.6 = 30W$

153 5.2 Second Iteration
154 5.2.1 Circuit Building
155 5.3 Third Iteration
156 5.3.1 Final Wiring Diagrams
157 5.3.2 Final Power Calculations
158 5.3.3 Circuitry

¹⁵⁹ Chapter 6

¹⁶⁰ Final Product

¹⁶¹ 6.1 Testing Protocol

¹⁶² 6.2 Integration

¹⁶³ 6.3 System Performance

¹⁶⁴ Chapter 7

¹⁶⁵ Discussion

¹⁶⁶ 7.1 Limitations

¹⁶⁷ 7.2 Future Work

¹⁶⁸ **Chapter 8**

¹⁶⁹ **Conclusion**

¹⁷⁰ **8.1 Bill of Materials**

| Component | Purpose | Quantity | Price |
|--|------------------------------|----------|-----------------|
| 2PC T8x8 Lead Screw with Brass Nut | Linear motion conversion | 3 | \$13.99 |
| 5-8mm Lead Screw Coupler 5 pack | Connect motor to screw | 3 | \$8.69 |
| 8x200mm Shaft Guide | Linear guidance | 3 | \$8.69 |
| 8mm Flange Mounted Pillow Block Bearing | Support rotating shaft | 3 | \$8.99 |
| Raspberry Pi 5 (4GB RAM) | Primary controller | 1 | \$66.00 |
| Plastic 30mL Syringes (Pack of 5) | Fluid handling prototype | 1 | \$6.99 |
| Raspberry Pi 5 Power Supply | Power for controller | 1 | \$15.99 |
| Assorted Metric Fasteners (M2–M5) | Mechanical assembly | 1 | \$20.00 |
| Limit Switch (10 pack) | Motion limit detection | 1 | \$5.99 |
| Arducam V3 Camera | Vision and monitoring | 1 | \$25.00 |
| Wiring Kit | Electrical connections | 1 | \$5.00 |
| Breadboard / Perfboard | Circuit prototyping | 1 | \$10.00 |
| 5V 3A DC Power Supply | General power source | 1 | \$7.47 |
| 5 Sets 28BYJ-48 Stepper + ULN2003 Driver | Secondary actuation system | 5 | \$14.99 |
| 1.75mm ABS Filament Reel (Black) | Structural printing material | 1 | \$16.99 |
| Total Estimated Cost | | | \$346.49 |

Table 8.1: Key Hardware Elements.

¹⁷¹ References

¹⁷² Bibliography

- ¹⁷³ [1] Sophie Smith. Billions of beauty packaging goes unrecycled every year - theindustry.beauty. 2024.