

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	⁴
¹¹ 3.1.1 Initial CAD Model and Hand Sketches	⁴
¹² 3.2 Second Iteration	⁷
¹³ 3.2.1 Updated CAD Models	⁷
¹⁴ 3.3 Third Iteration	⁹
¹⁵ 3.3.1 Final CAD Models	⁹
¹⁶ 3.3.2 Final Manufacturing Drawings for Custom Parts	¹⁴
¹⁷ 4 Software Design	²⁰
¹⁸ 4.1 First Iteration	²⁰
¹⁹ 4.1.1 Preliminary Diagrams and Psuedocode	²⁰
²⁰ 4.1.2 User Flow Diagram	²²
²¹ 4.1.3 Techstack	²³
²² 4.2 Second Iteration	²³
²³ 4.2.1 Lo-fi/Mid-fi Designs	²³
²⁴ 4.2.2 Basic Logic and Functionality	²⁴
²⁵ 4.2.3 Core Features of Algorithm	²⁵
²⁶ 4.3 Third Iteration	²⁶
²⁷ 4.3.1 Final Product	²⁶
²⁸ 5 Electrical Systems Design	²⁷
²⁹ 5.1 First Iteration	²⁸
³⁰ 5.1.1 Initial Wiring Diagrams	²⁸
³¹ 5.1.2 Initial Power Calculations	²⁹
³² 5.2 Second Iteration	²⁹
³³ 5.2.1 Circuit Building	²⁹

34	5.3 Third Iteration	30
35	5.3.1 Final Wiring Diagrams	30
36	5.3.2 Final Power Calculations	30
37	5.3.3 Circuitry	31
38	6 Final Product	32
39	6.1 Testing Protocol	34
40	6.2 Integration	34
41	6.3 System Performance	34
42	7 Discussion	35
43	7.1 Limitations	35
44	7.2 Future Work	35
45	8 Conclusion	36
46	8.1 Bill of Materials	36
47	References	37

48 List of Figures

49	3.1 First Sketch: Housing	4
50	3.2 First CAD Models: Housing	5
51	3.3 First Sketch: Dispenser Systems	5
52	3.4 First CAD Models: Dispenser Systems	6
53	3.5 Second Iteration: Full Assembly	7
54	3.6 Second Iteration: Exploded View	7
55	3.7 Second Iteration: Housing Skeleton and Walls	8
56	3.8 Second Iteration: Brackets for Dispenser System	8
57	3.9 Final: Full Assembly <i>Exploded</i>	9
58	3.10 Housing: Walls	10
59	3.11 Housing: Lid	10
60	3.12 Housing: Skeleton	11
61	3.13 Dispenser System : Brackets - for connecting to housing, guide shaft, stabilizing syringe, and pushing down syringe plunger	12
62	3.14 Dispenser System : Full assembly	13
63	3.15 Full Assembly Exploded View	14
64	3.16 Housing: Walls - Back	15
65	3.17 Housing: Walls - Side	16
66	3.18 Housing: Lid	16
67	3.19 Dispenser System : L-Brackets	17

69	3.20 Dispenser System : Syringe Brackets	18
70	3.21 Dispenser System : Full assembly Exploded View	19
71	4.1 Initial Flowchart for Control Logic	20
72	4.2 Pseudocode: Image Processing Algorithms.	21
73	4.3 Pseudocode: Embedded System.	22
74	4.4 User Flow Diagram	22
75	4.5 Initial Figma Mockup: UI/UX - Landing page.	23
76	4.6 Initial Figma Mockup: UI/UX - Loading the foundation shades.	23
77	4.7 Macbeth Color Reference Sheet	25
78	5.1 Preliminary Wiring Diagram	28
79	5.2 Perf Board and Servo Motor to Driver Connection	29
80	5.3 Final Wiring Diagrams	30
81	5.4 Final Circuitry Under Housing Lid	31
82	6.1 Dispenser Systems Fully Assembled	33
83	6.2 Skeleton/Internal Structure	34

84 List of Tables

85	5.1 Power Calculations	29
86	8.1 Updated Bill of Materials.	36

⁸⁷ **Chapter 1**

⁸⁸ **Abstract**

⁸⁹ The Foundation Identifier and Dispenser aims to develop a machine that is able to extract
⁹⁰ samples from a picture of a human to determine the shade of their skin, allowing the machine
⁹¹ to dispense a corresponding foundation shade. The results produced should be both accurate
⁹² and reproducible. Equipped with computer vision libraries and color correction algorithms,
⁹³ the system allows the user to take an picture alongside a reference color sheet, which has
⁹⁴ colors of known values. These captured values are processed to correct both camera bias,
⁹⁵ and lighting correction so that results may remain consistent regardless of lighting conditions
⁹⁶ during image capture. The system converts RGB values to LAB values, which are higher in
⁹⁷ accuracy in physical color mixing, as opposed to RGB, which is used to describe pixel colors.
⁹⁸ The program calculates how much of each color is needed to recreate the user's skin pigment.
⁹⁹ These pigments are then dispensed via a mechanical system comprised of a Raspberry Pi,
¹⁰⁰ servo motors, and syringes. This project demonstrates an application of computer vision in
¹⁰¹ the cosmetic market to alleviate the burden of overconsumption and promote inclusivity.

¹⁰² **Chapter 2**

¹⁰³ **Introduction**

¹⁰⁴ **2.1 Research**

¹⁰⁵ **2.1.1 Background of Problem**

¹⁰⁶ Testing foundation colors can be a frustrating experience for many, who are unable to
¹⁰⁷ find the perfect balance. At the end of the day, no line of foundation can realistically provide
¹⁰⁸ colors that cater to every possible skin tone. The seemingly unresolvable desire for a perfect
¹⁰⁹ shade leads many makeup users to spend hundreds on shades that are "close enough." This
¹¹⁰ leads to lots of waste, not just in money, but in bottles thrown out after purchase because
¹¹¹ the match was ultimately unsatisfying.

¹¹² The beauty industry produces 120 billion packaging units per year and 95% of these units are discarded, as opposed to recycled [1]. In addition, traditionally a custom skin matched foundation can cost anywhere from \$60-100 per bottle, which leaves the consumers with the dilemma of whether they should take the gamble on the bottle that's just almost right versus breaking their wallet on a hand matched bottle. Our custom mixer machine offers the same accuracy in skin tone while also promoting cheaper makeup, sustainability, and waste-reduction.

¹¹⁹ Our foundation color picker abandons the concept of creating a set of discrete skin tones to choose from, instead, opting for custom mixed shades depending on the skin color detected by a picture. It also offers the unique ability to sample a shade without having to commit to a full sized bottle.

¹²³ **2.1.2 Existing Solutions**

¹²⁴ BoldHue provides an option for an AI powered foundation mixer. It matches skin tone through a smart wand that detects color. What they promised was millions of shades and the ability to store user's shades in profiles. However, their color detection methods provide room for lots of error because color isn't perceived the same depending on the lighting of the room, as well as the high cost of their machine.

¹²⁹ Our product will have built in lighting correction that removes biases from the camera and uses the Macbeth Color Reference sheet as a set of control colors. The reference sheet

¹³¹ is what will allow the machine to recalibrate it's understanding of what colors are being
¹³² percieved.

¹³³ 2.2 Design Choices

¹³⁴ 2.2.1 Past Considerations and Scrapped Plans

¹³⁵ Off-the-Shelf Syringe Pump System

¹³⁶ An early design decision we explored was to use commercially available syringe pump
¹³⁷ system available for purchase from a third party. This idea was ultimately abandoned due
¹³⁸ to bulkiness of commercial pumps, as well as high cost. Although designing our own pump
¹³⁹ system required more work from our mechanical team to design the system from scratch,
¹⁴⁰ relying on an existing system would have limited our freedom, increased our costs, and taken
¹⁴¹ away from the educational value of the project. Designing a custom syringe pump offered
¹⁴² hands-on learning experiences in CAD design.

¹⁴³ Quick-Swap Syringe Mounting Attachment

¹⁴⁴ We also considered designing the Syringe housing to allow users to detatch and fill syringes
¹⁴⁵ in between uses. This feature was ultimately unnecessary as the syringes don't need to be
¹⁴⁶ manually pumped. As long as the moters can rotate the opposite direction, the syringes dont
¹⁴⁷ have to be removed to retract. The final design eliminated the syringe-swapping mechanism
¹⁴⁸ in favor of refilling via holding a paint vial under the syringe and letting the machine retract
¹⁴⁹ the pump to draw in fluid.

¹⁵⁰ Single-Piece Printed Housing

¹⁵¹ Our initial housing was a single piece. However, most 3D printers that were available
¹⁵² have a maximum build volume of 256 mm x 256 mm x 256 mm, and the housing dimensions
¹⁵³ we had for the initial design was too close to this limit. Printing the structure in a single
¹⁵⁴ piece also risks expensive failed prints, warping, and geometric constraints. We shifted to
¹⁵⁵ a housing unit comprised of three interlocking sections, with custom brackets to join the
¹⁵⁶ segments securely.

¹⁵⁷ Chapter 3

¹⁵⁸ Hardware Design and Specifications

¹⁵⁹ 3.1 First Iteration

¹⁶⁰ 3.1.1 Initial CAD Model and Hand Sketches

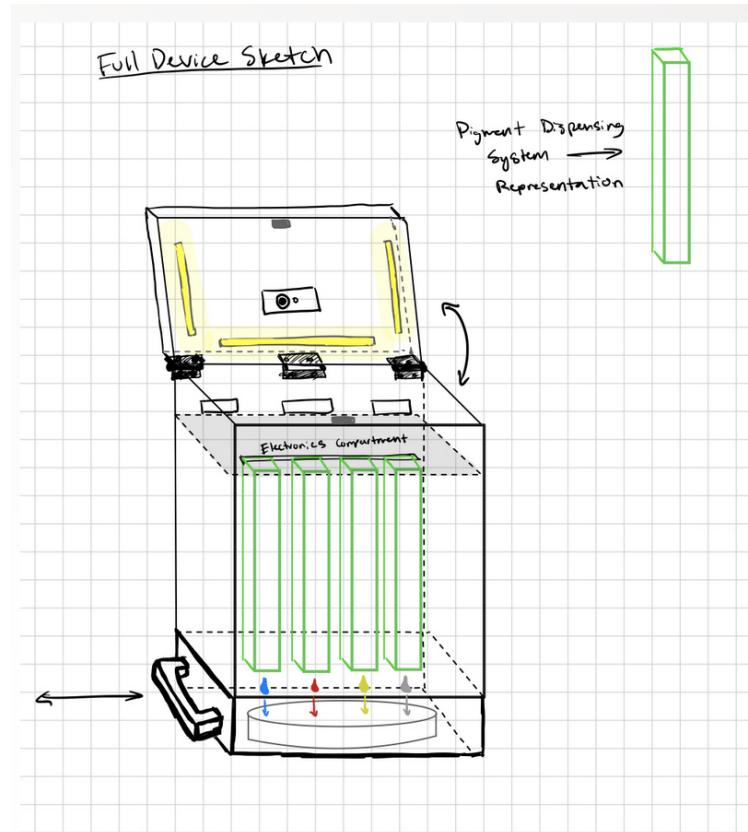


Figure 3.1: First Sketch: Housing

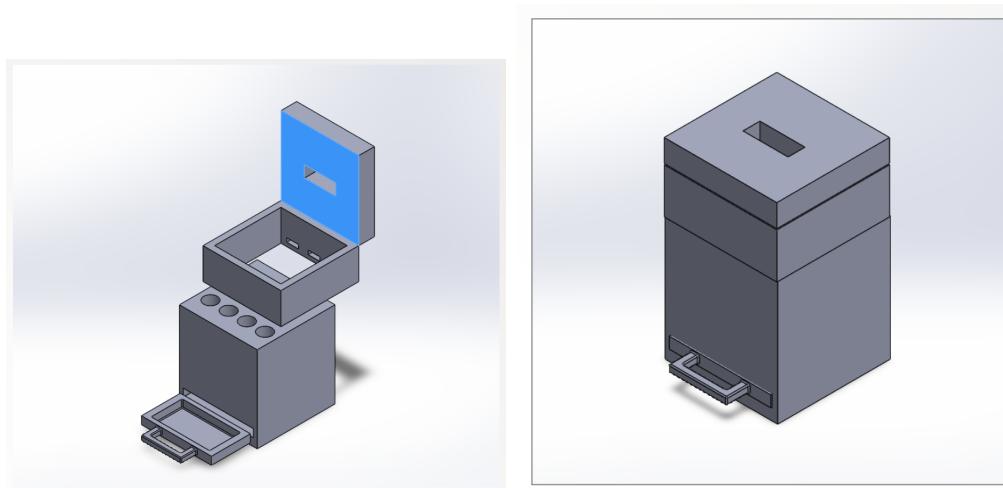


Figure 3.2: First CAD Models: Housing

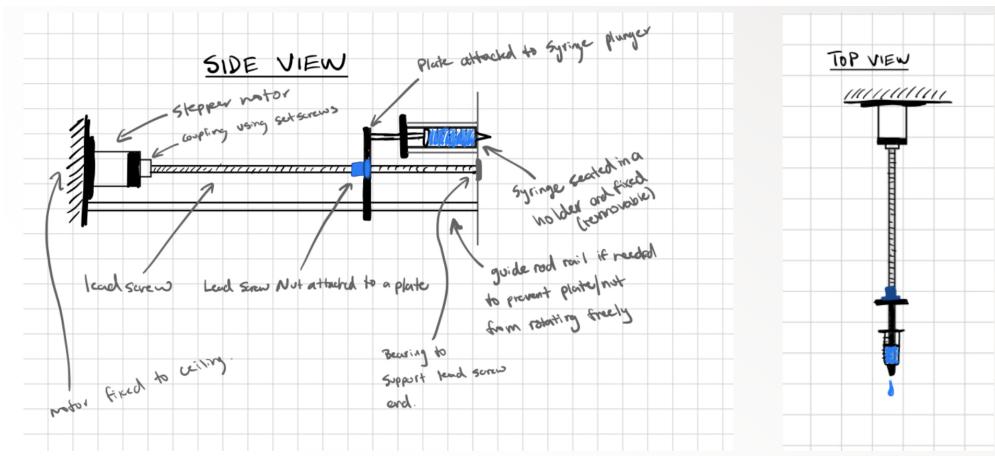


Figure 3.3: First Sketch: Dispenser Systems



Figure 3.4: First CAD Models: Dispenser Systems

¹⁶¹ Our first CAD Models were very simple and provide the most general idea we had at
¹⁶² our projects conception. We made large changes to the design of the housing because the
¹⁶³ dimensions would have been much wider if we lined the syringes up, as shown in Figures 3.1
¹⁶⁴ and 3.2.

¹⁶⁵ The preliminary sketch we have of the dispenser system has stayed consistent throughout
¹⁶⁶ development.

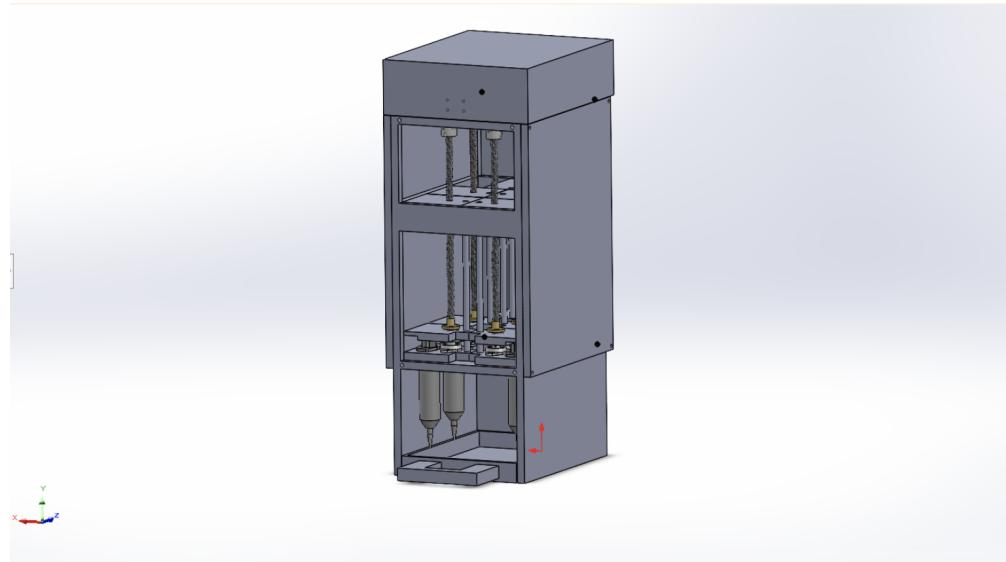
167 3.2 Second Iteration**168 3.2.1 Updated CAD Models****169 Assembly**

Figure 3.5: Second Iteration: Full Assembly

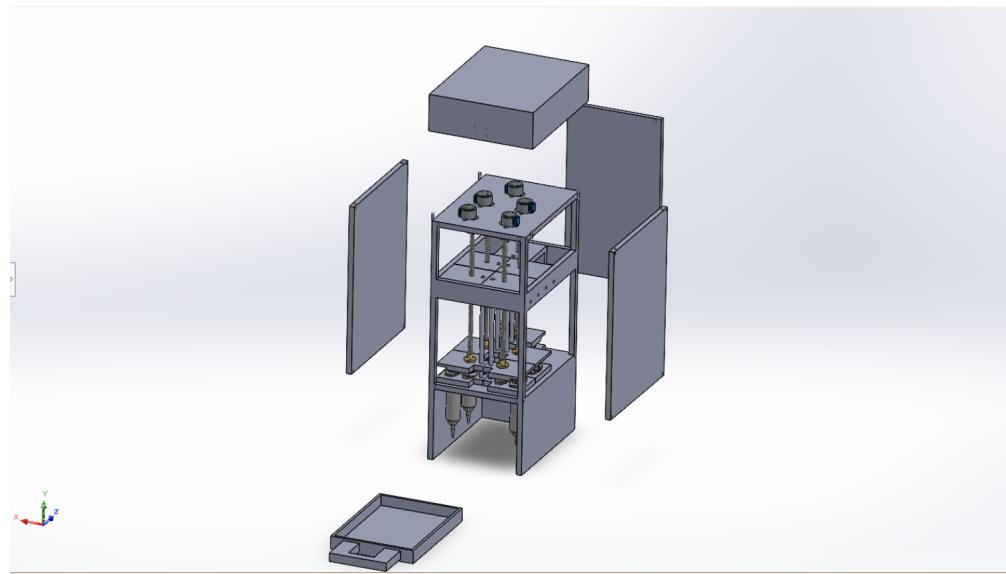


Figure 3.6: Second Iteration: Exploded View

170

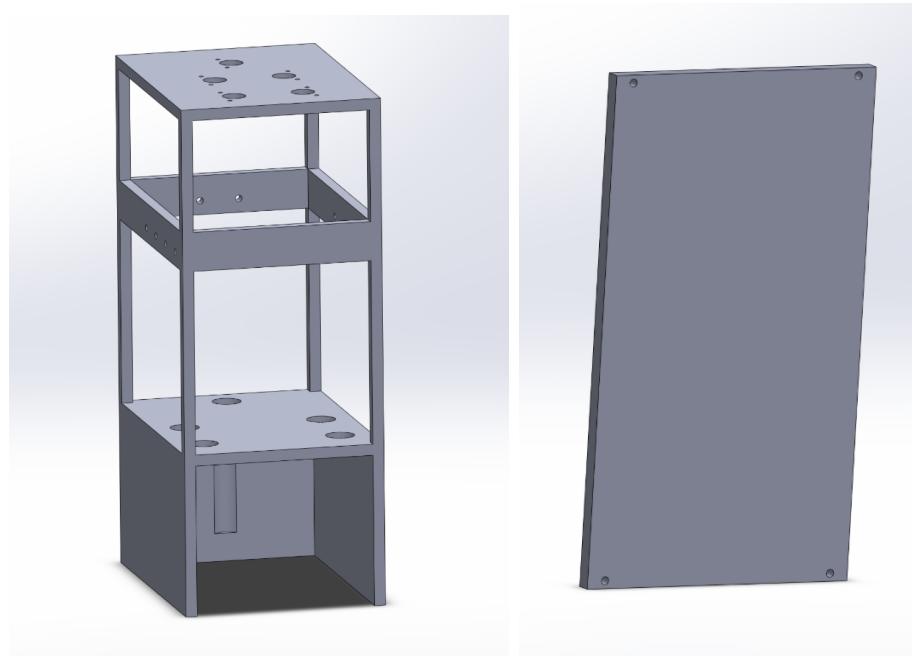
Housing

Figure 3.7: Second Iteration: Housing Skeleton and Walls

171 The housing skeleton is what holds the syringes in place while the walls create an enclosure
172 so that the internal structure may be protected and hidden from users. The main concern
173 with this iterations design was with the housing skeleton's stability. Since we were 3D
174 printing all parts for the demo, the empys spaces could be flimsy, so the final iteration
175 includes a redesign.

176 **Dispenser**

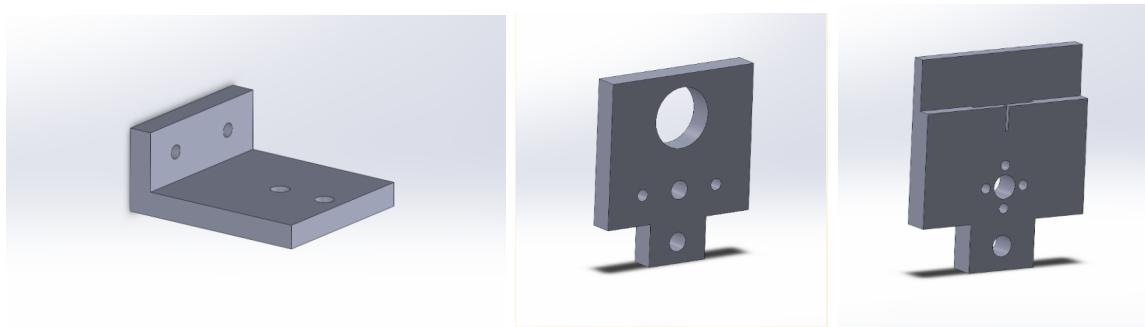


Figure 3.8: Second Iteration: Brackets for Dispenser System

177 These brackets hold and control the dispenser system. The first L-shaped Bracket holds
178 the guide shaft in place. The other two are for stabilizing and pushing the syringe plunger
179 down. This second iteration is missing one of the parts but it will be shown in the next
180 section, along with final drawings of all parts.

¹⁸¹ **3.3 Third Iteration**

¹⁸² **3.3.1 Final CAD Models**

¹⁸³ *Assembly*

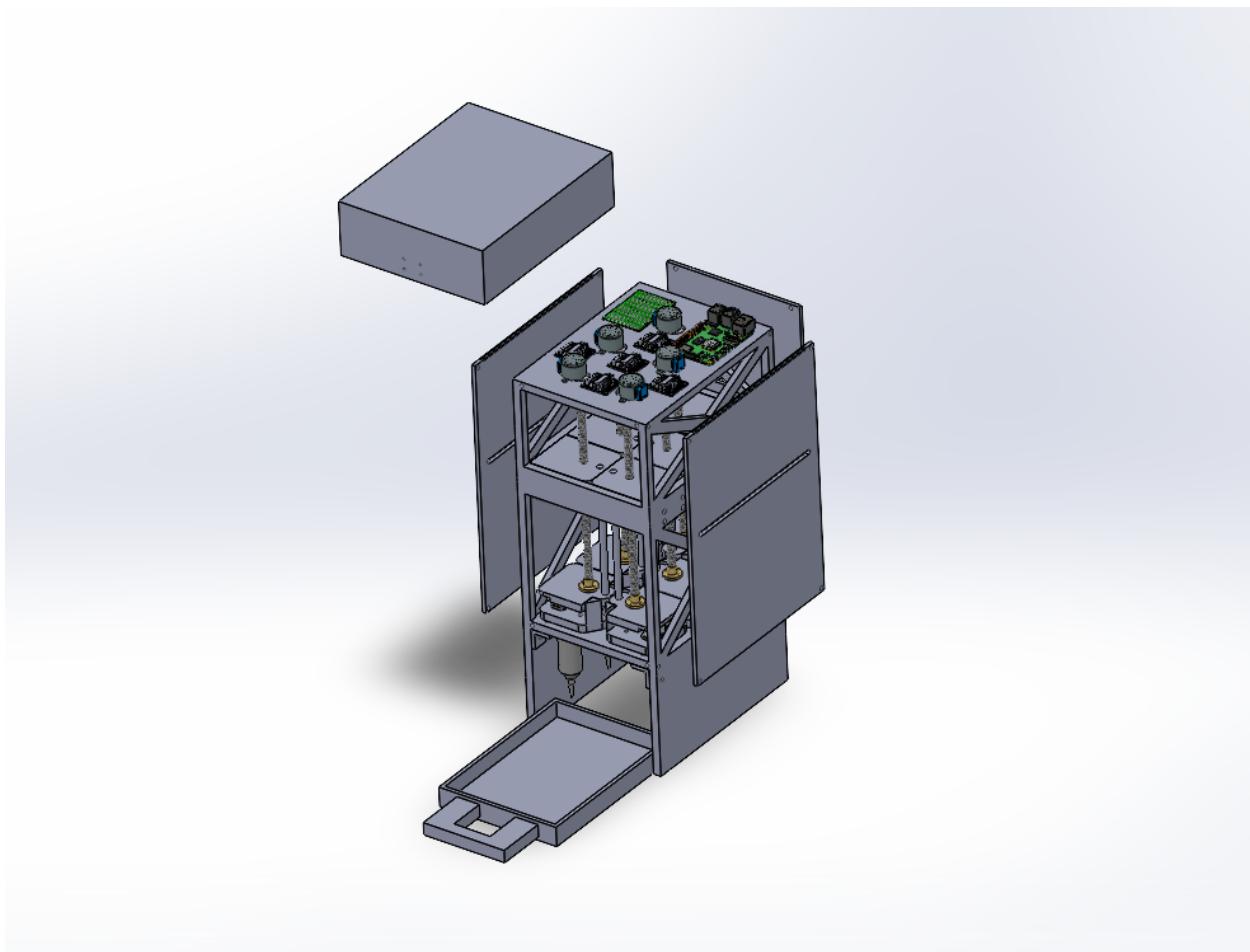


Figure 3.9: Final: Full Assembly *Exploded*

184

Housing

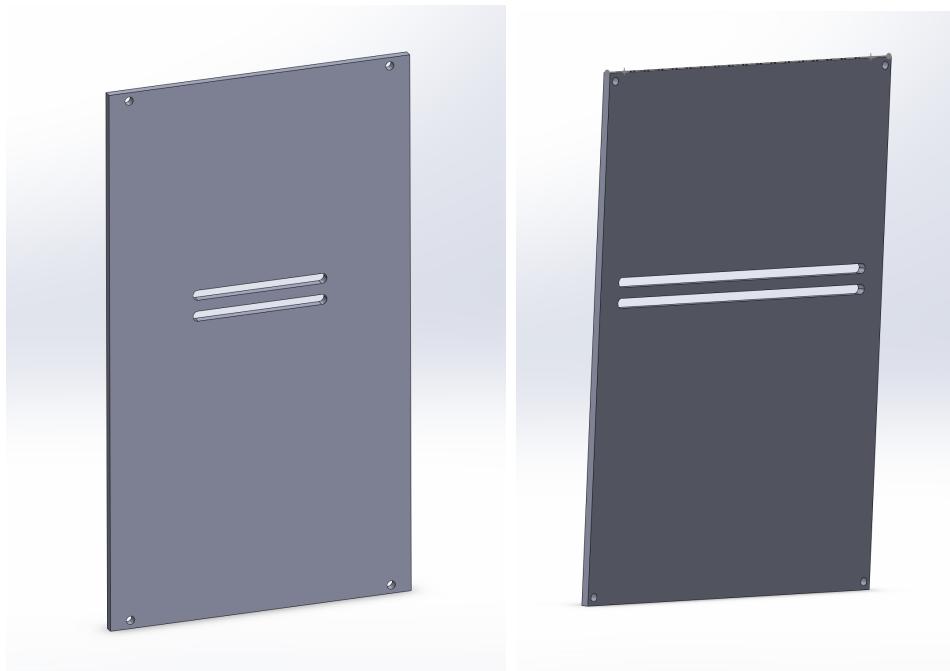


Figure 3.10: Housing: Walls

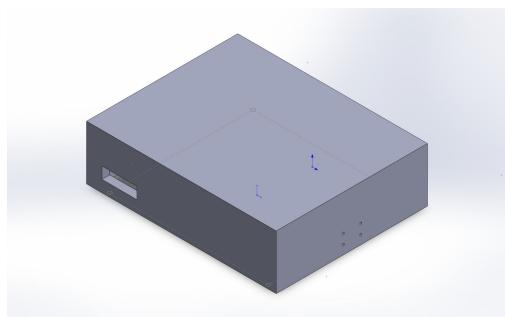


Figure 3.11: Housing: Lid

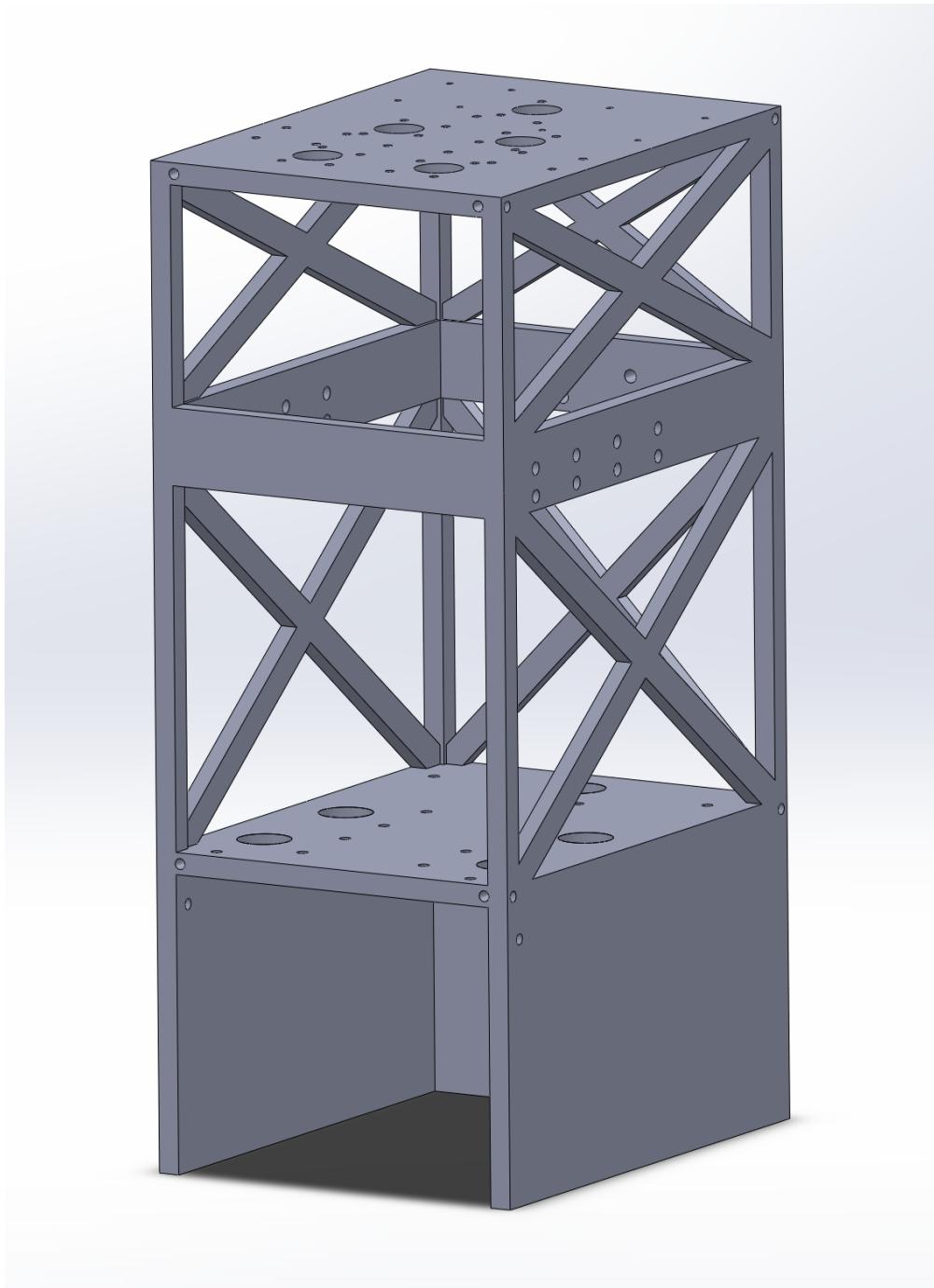


Figure 3.12: Housing: Skeleton

¹⁸⁵ The Housing walls were altered to include a slit because there are some screws in the
¹⁸⁶ skeleton of the product that would press up against the walls if we kept the second iteration's
¹⁸⁷ design. The slit was created to give the screws space and ensure that the components inside
¹⁸⁸ are not putting pressure on each other.

189

Dispenser

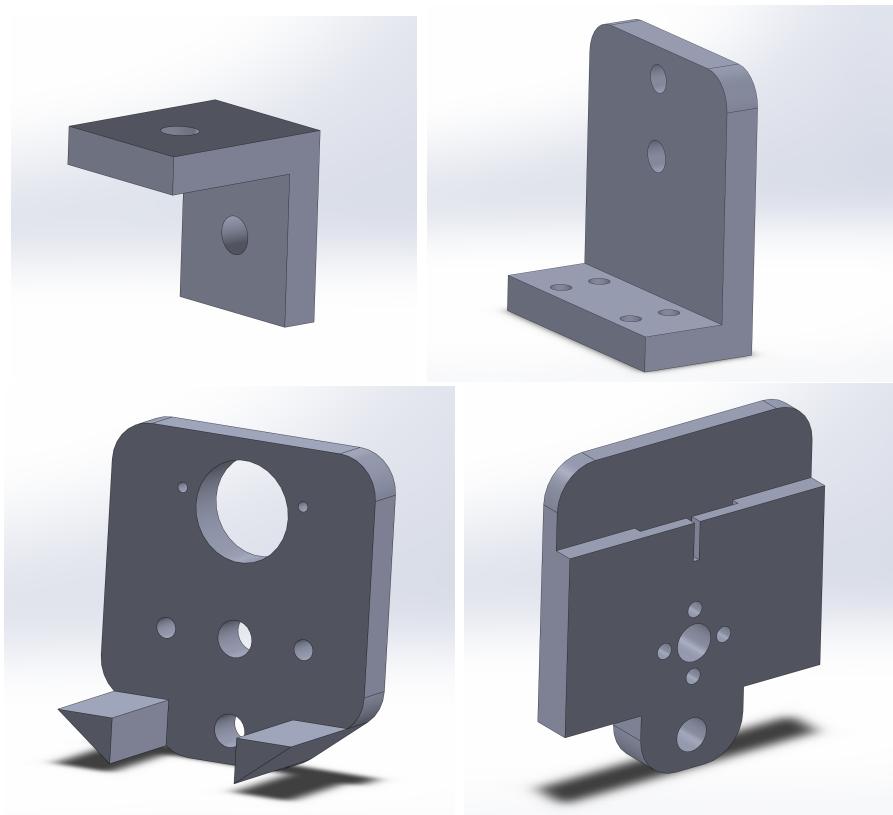


Figure 3.13: Dispenser System : Brackets - for connecting to housing, guide shaft, stabilizing syringe, and pushing down syringe plunger

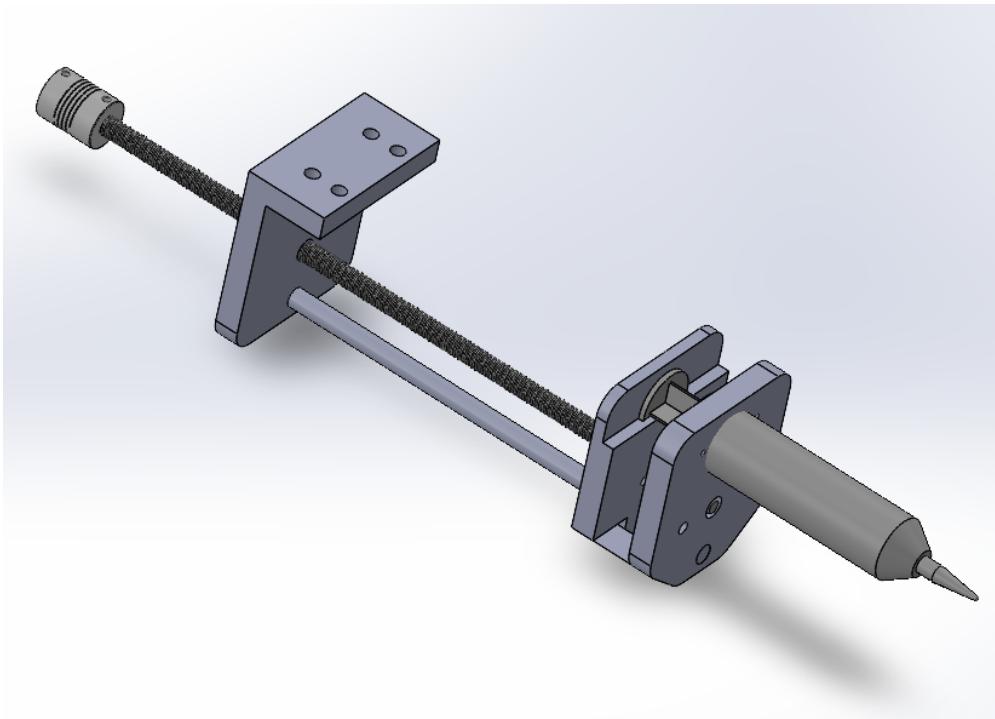


Figure 3.14: Dispenser System : Full assembly

190 The final design consists of four primary components: an L-bracket connecting the dis-
191 penser to the casing, a secondary L-bracket securing the guide rod, a syringe bracket that
192 stabilizes and positions the syringe, and a plunger plate driven by the lead screw to depress
193 the syringe plunger.

194 The final iteration incorporates several refinements across all components. Mounting
195 brackets and L-brackets were redesigned with rounded edges to increase clearance between
196 moving elements. The lower syringe bracket was modified to include two vertical extrusions
197 that act as contact surfaces for limit switches. Additionally, mounting holes for fasteners
198 were added throughout the assembly to improve structural stability and ease of integration.
199

200 3.3.2 Final Manufacturing Drawings for Custom Parts

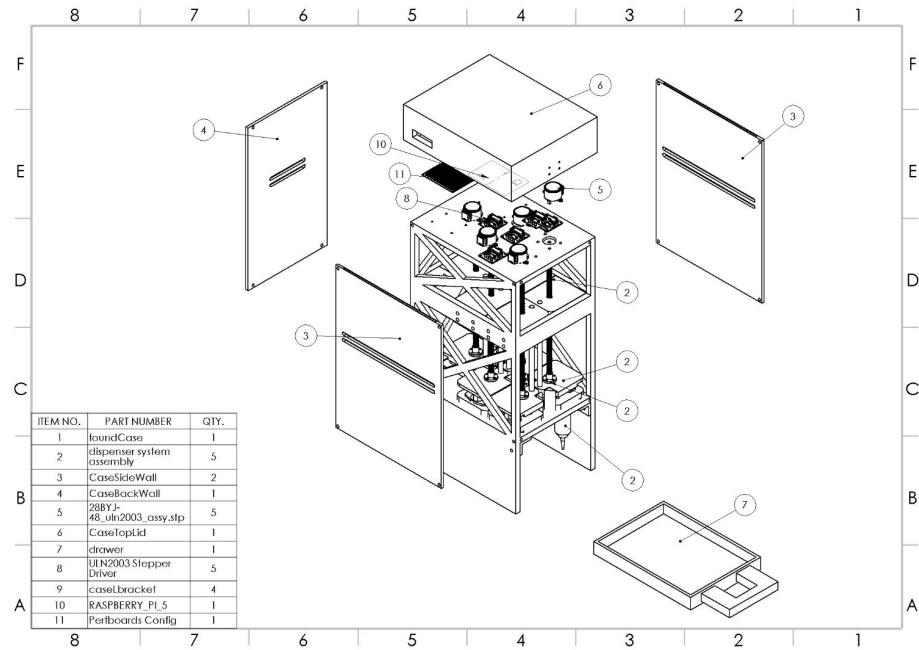


Figure 3.15: Full Assembly Exploded View

201

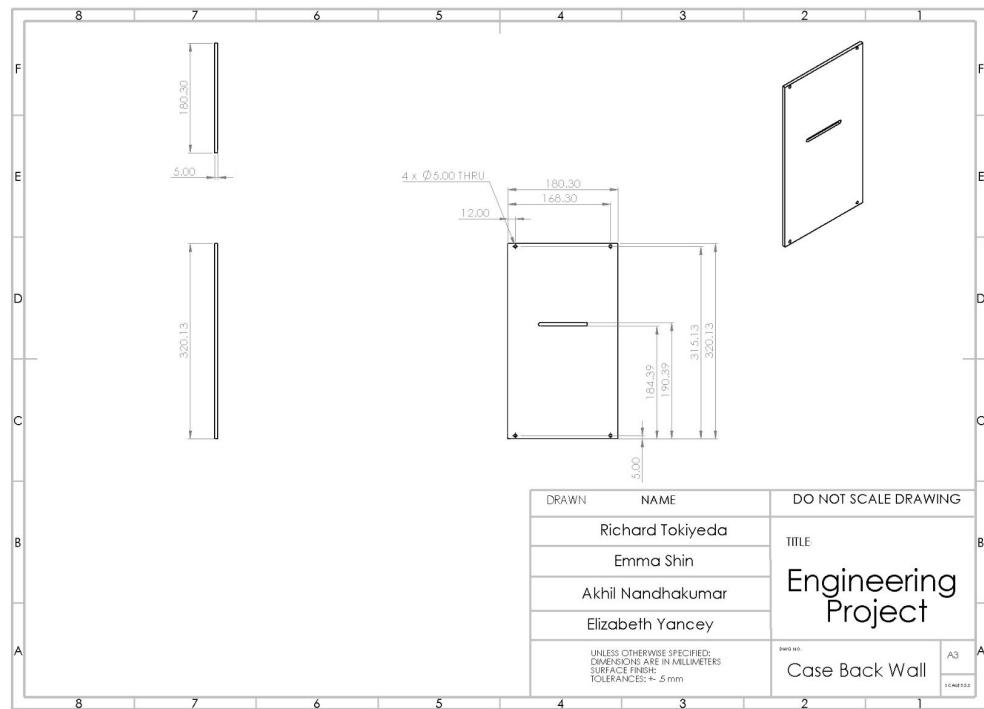
Housing

Figure 3.16: Housing: Walls - Back

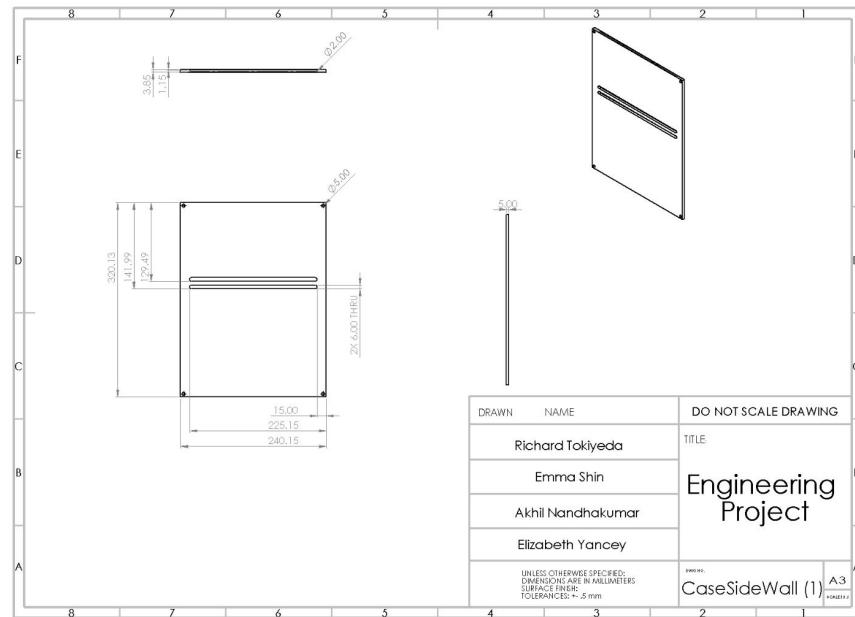


Figure 3.17: Housing: Walls - Side

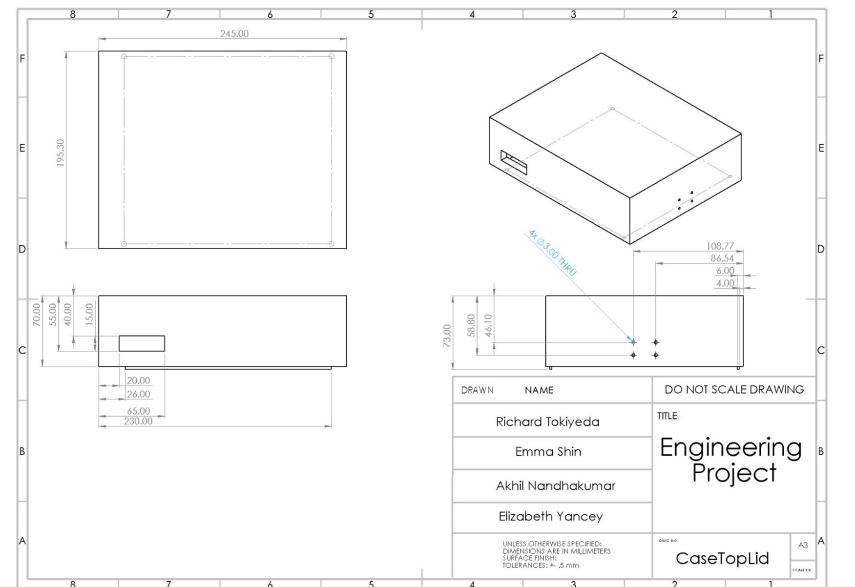


Figure 3.18: Housing: Lid

202

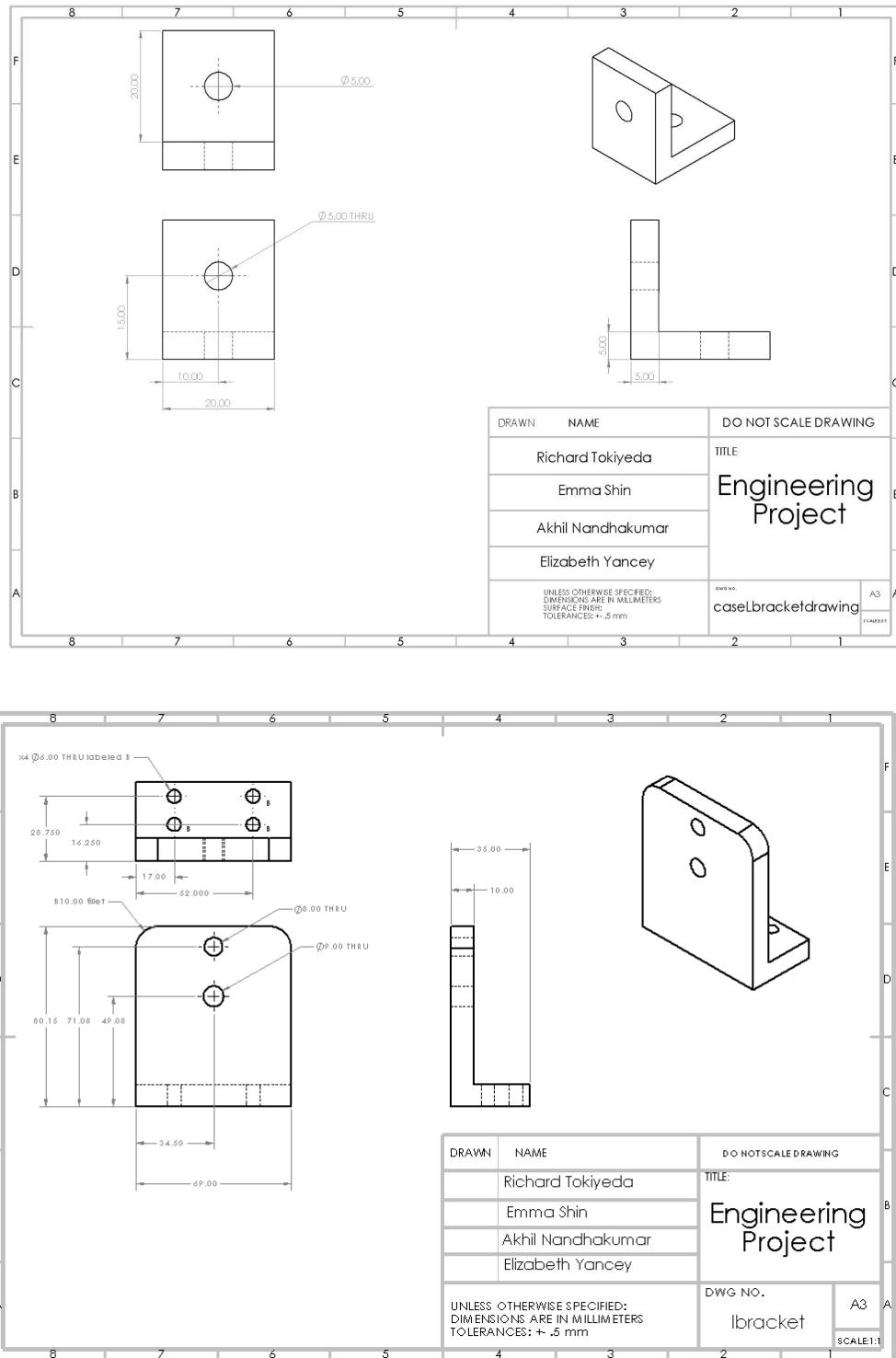
Dispenser

Figure 3.19: Dispenser System : L-Brackets

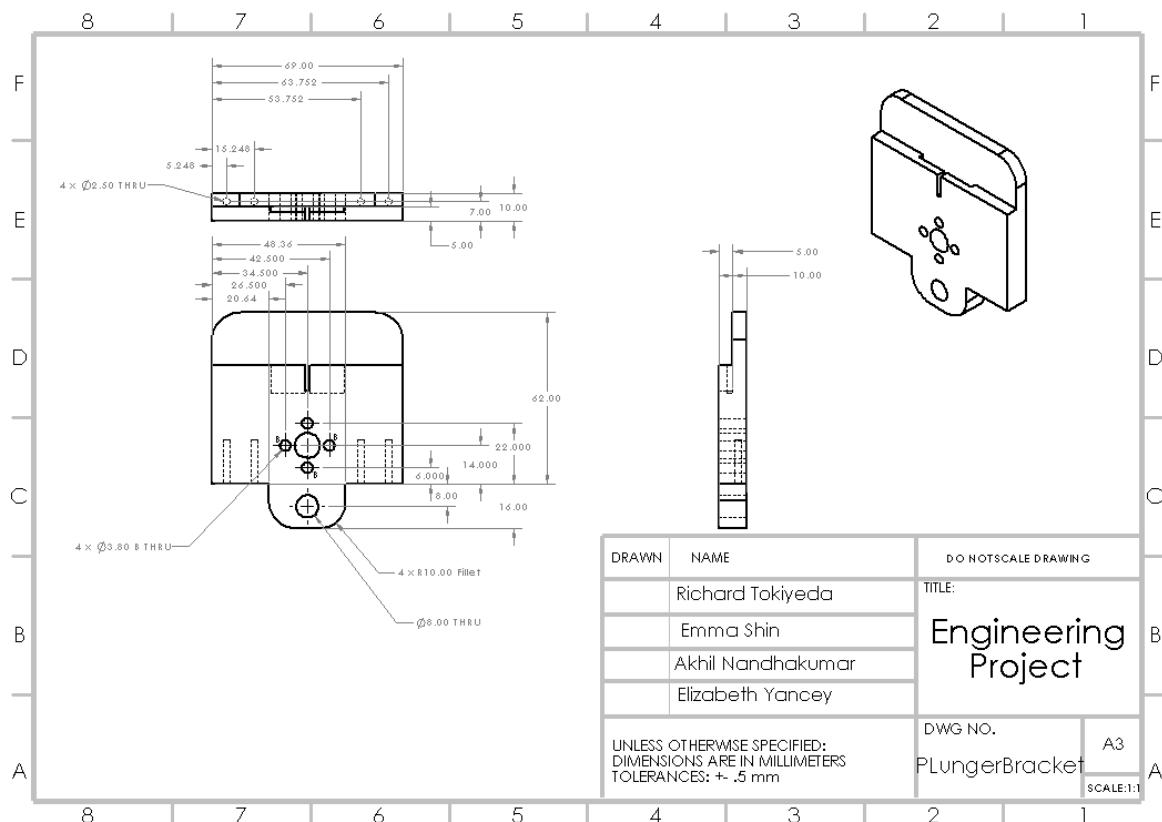
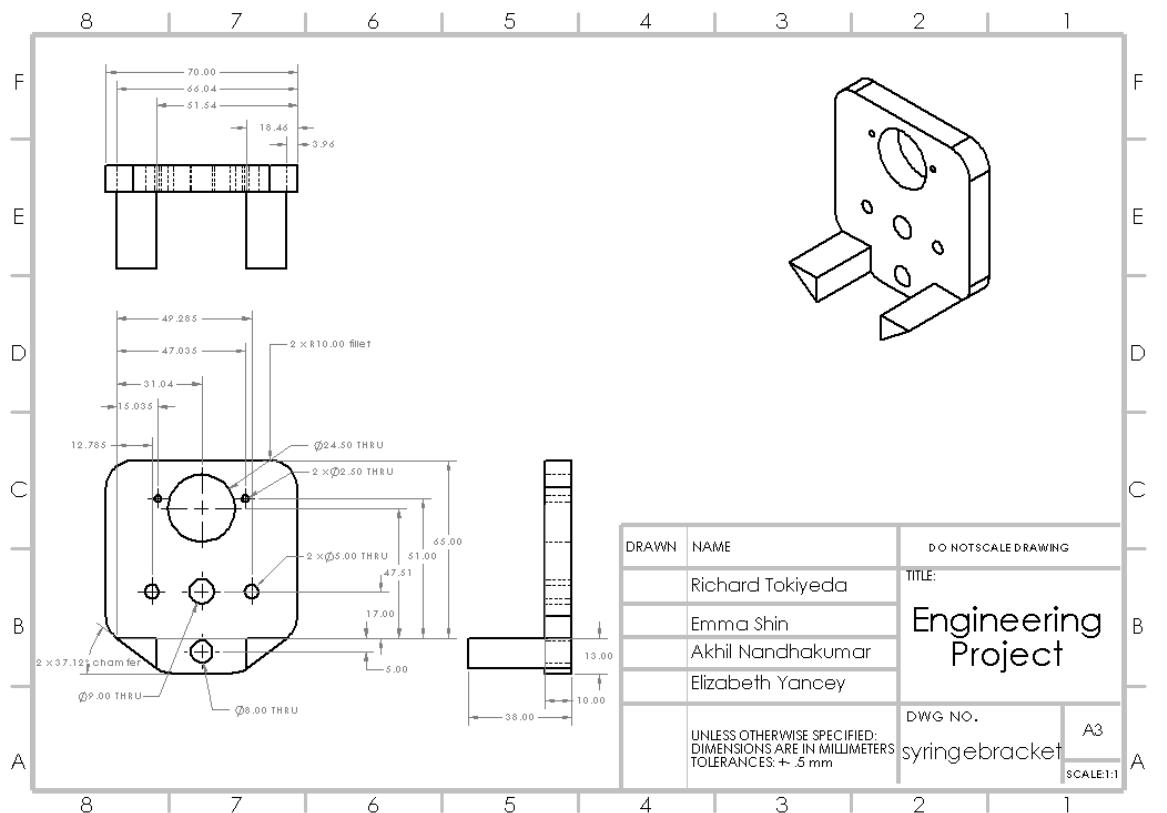


Figure 3.20: Dispenser System : Syringe Brackets

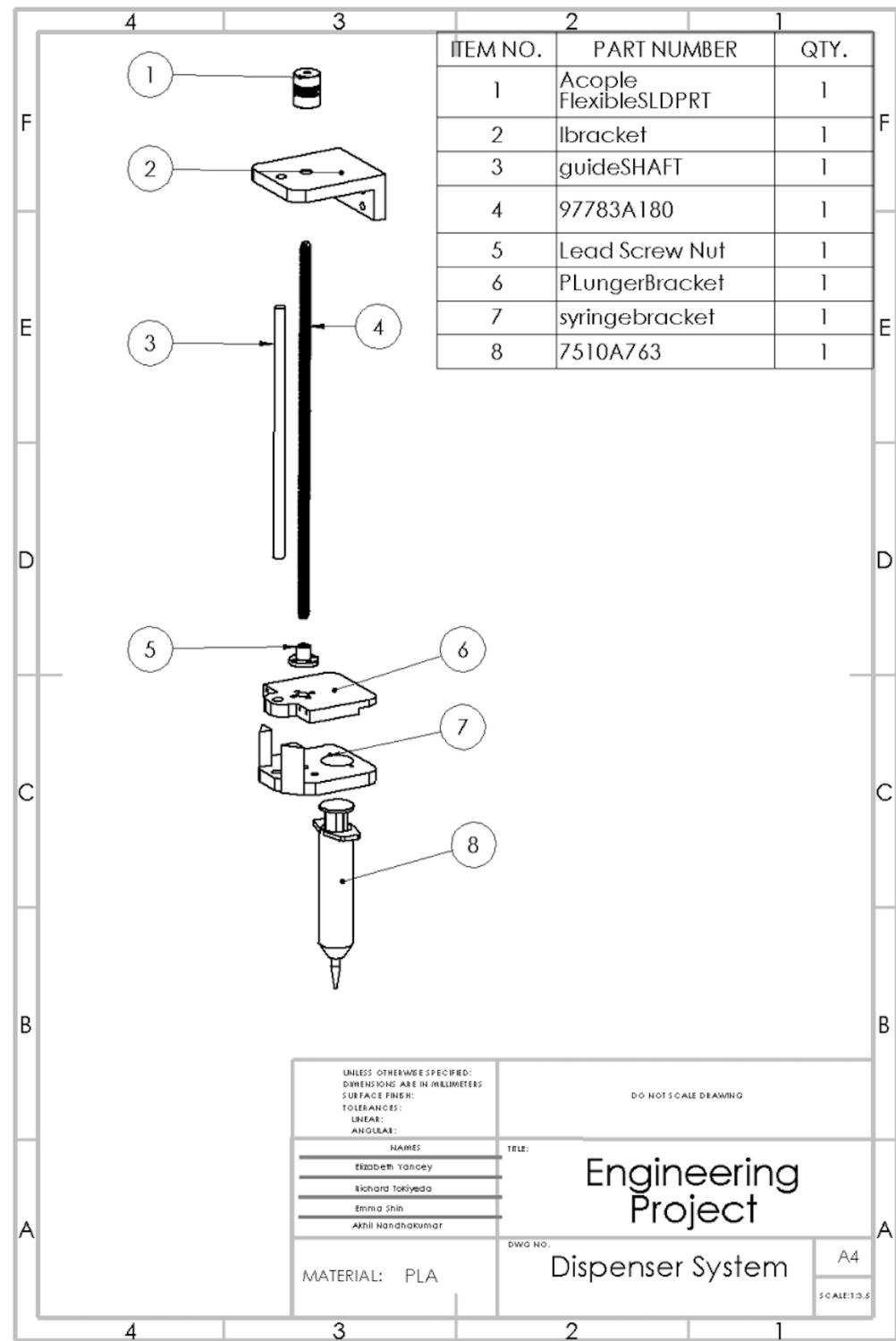


Figure 3.21: Dispenser System : Full assembly Exploded View

203 Chapter 4

204 Software Design

205 4.1 First Iteration

206 4.1.1 Preliminary Diagrams and Psuedocode

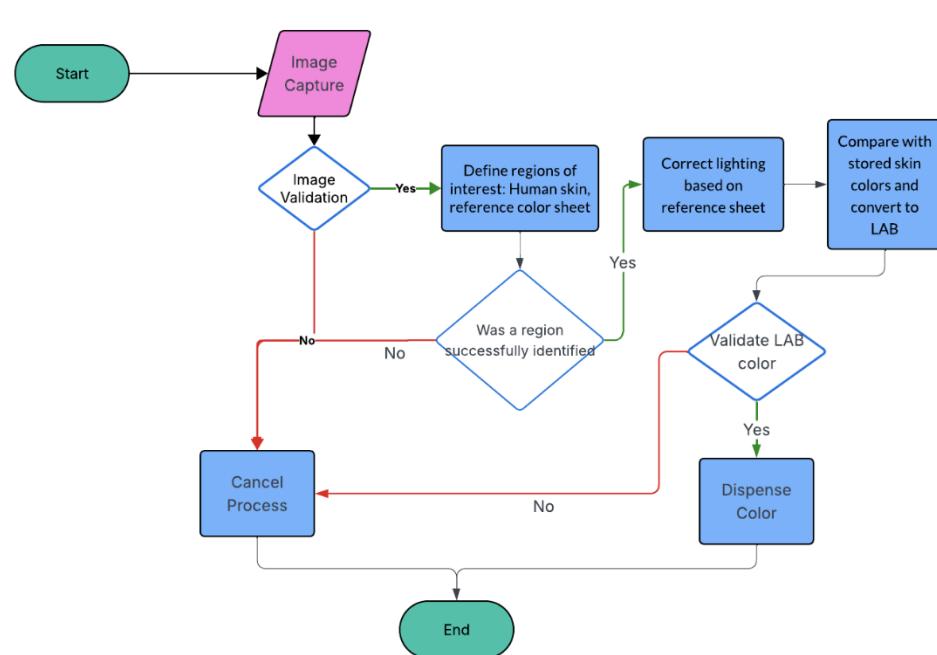


Figure 4.1: Initial Flowchart for Control Logic

```

# 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.

²⁰⁷ 4.1.2 User Flow Diagram

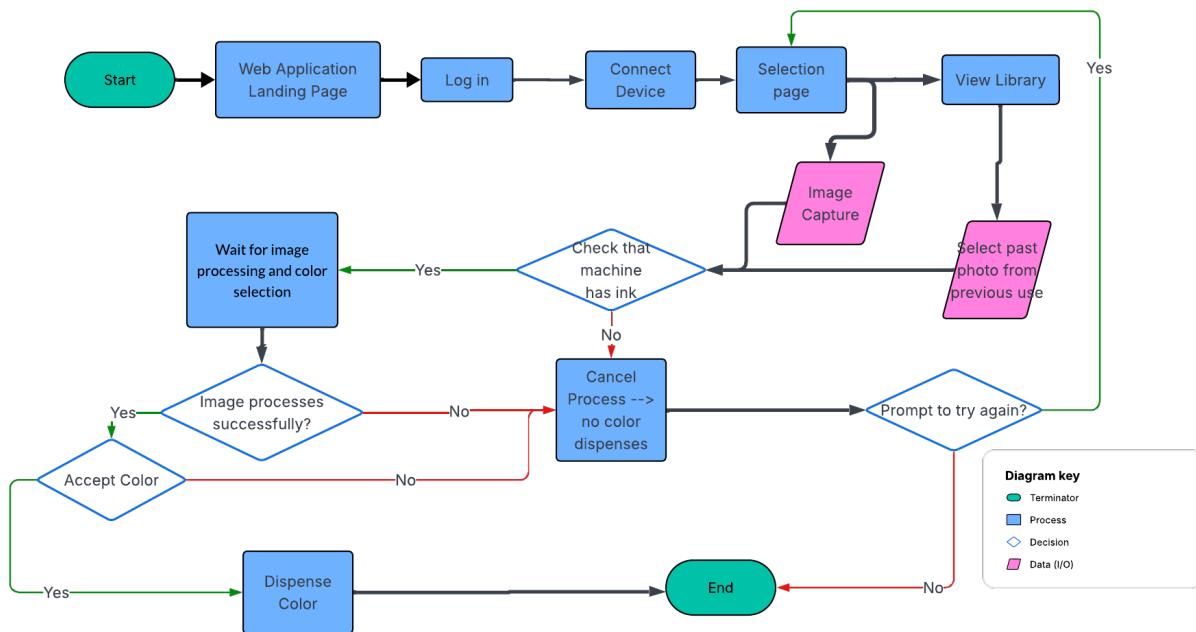


Figure 4.4: User Flow Diagram

208 4.1.3 Techstack

209 Frontend: React, Vite, Typescript, TailwindCSS, Lucide Icons
 210 Storage : IndexedDB (local storage)
 211 Hardware/OS/Hosting : Raspberry Pi (Linux, Pi Local Hosting)
 212 Computer Vision & Data Science : OpenCV, NumPy, Pandas, Haar Cascades
 213 Languages : Typescript, Python
 214 Tooling : Git/Github

215 4.2 Second Iteration

216 4.2.1 Lo-fi/Mid-fi Designs

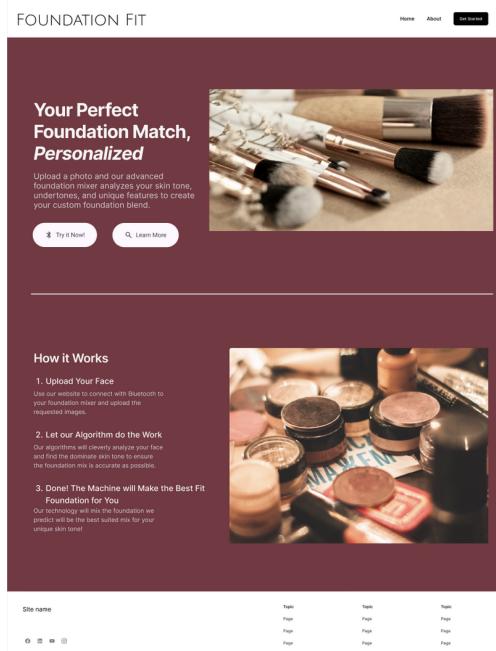


Figure 4.5: Initial Figma Mockup: UI/UX - Landing page.

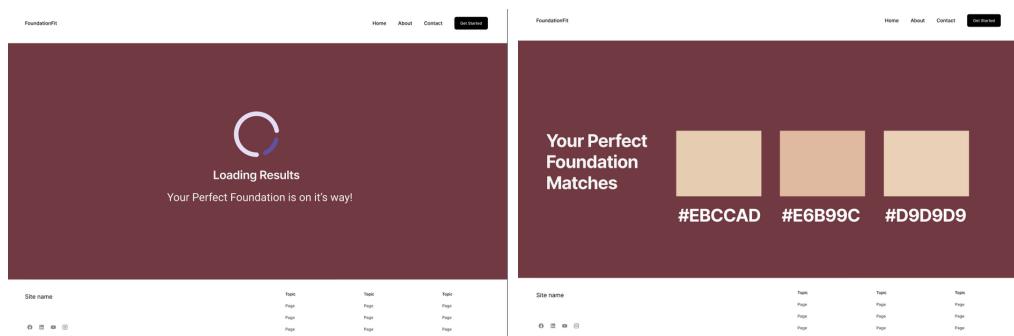


Figure 4.6: Initial Figma Mockup: UI/UX - Loading the foundation shades.

217 4.2.2 Basic Logic and Functionality

218 The backend service is responsible for two core tasks: analyzing a user's skin tone from
219 an input image and triggering the dispenser system for a requested foundation color. It is
220 implemented as a Flask web API with three main endpoints: '/analyze', '/dispense', and
221 '/ping' *app.py*.

222 Skin Tone Analysis Pipeline '/analyze'

223 Image intake (frontend → backend) :

224 Frontend allows the user to view a live feed and takes a picture when it detects the face
225 reference sheet within the yellow box on the screen. The photo is sent to the '/analyze'
226 endpoint. Backend strips, decodes, and loads the data captured into an image object, which
227 is saved for processing.

228 Skin region extraction :

229 The image is converted to a NumPy RGB array and passed to 'define_skin()', which
230 detects and isolates skin pixels. This focuses the analysis on relevant skin regions instead of
231 the background.

232 Color space processing and lighting correction :

233 Skin samples are flattened into a matrix and normalized. 'gamma_to_linear()' converts
234 gamma corrected RGB values to linear RGB. 'lighting_correction()' corrects the for
235 inconsistent lighting and the corrected RGB is converted to XYZ in 'linear_to_xyz()'
236 then to LAB in 'xyz_to_lab()'.

237 Output color encoding :

238 The average LAB color is mapped to an approximate RGB value in 'lab_avg_to_rgb()'
239 and formatted as a hexadecimal color string. This hexadecimal string is returned as '{"re-
240 sult": "<hex_color>"}' to the front end, which uses it for visualization for the user.

241 Dispensing Logic '/dispense'

242 Dispense accepts JSON payload containing the LAB code of the detected color. The
243 endpoint validates that the color is detected and prints a message to let the user know
244 dispensing has started. The endpoint is connected to the Raspberry Pi that drives the
245 stepper motors and pumps.

246 Health Check '/ping'

247 The '/ping' endpoint is a lightweight health-check route that returns status updates. It
248 allows the frontend to verify that the backend service is alive and responsive without invoking
249 the full analysis pipeline.

250 4.2.3 Core Features of Algorithm

251 The color detection algorithm transforms raw camera input into a corrected, device and
 252 lighting independent representation of the user's skin tone. This corrected color is then
 253 mapped to a cosmetic shade.

254 Macbeth Color Chart Reference Calibration

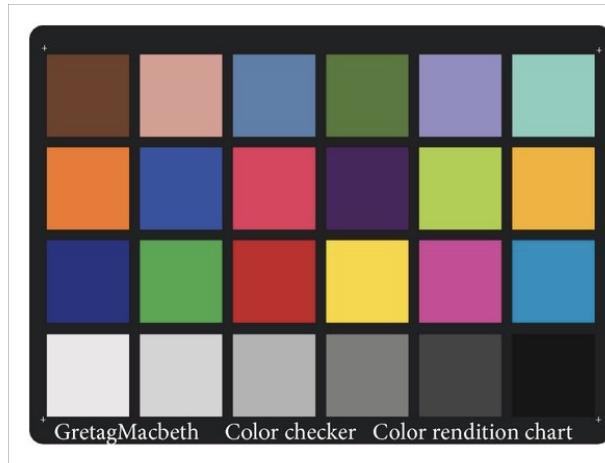


Figure 4.7: Macbeth Color Reference Sheet

255 The algorithm incorporates a Macbeth-style color chart for lighting and camera calibra-
 256 tion. Known RGB or Lab reference values for each chart patch are compared to captured
 257 values:

- 258 • The color chart is detected and segmented from the image.
- 259 • We extract a sample from the middle of the patch.
- 260 • We correct the lighting and distortion

261 Robust Skin Sampling from Facial Regions

262 A Haar-cascade face detector identifies the face region and we extract stable sampling
 263 zones from the mask returned by the Haar-cascade. Regions of interest were defined as the
 264 forehead, nose, and cheeks. Multiple points inside each region are collected to avoid shadows,
 265 edges, or hair interference. These values are aggregated to produce a stable estimate of the
 266 user's skin tone.

267 Gamma Correction and Linear RGB Conversion

268 Since cameras produce gamma-encoded RGB values, the algorithm applies a gamma-to-
 269 linear transformation. Both chart colors and skin samples are linearized before calibration.
 270 Working in linear RGB ensures that subsequent lighting corrections and color transfor-
 271 mations are physically meaningful.

272 Lighting Correction Using Chart Reference Data

273 A lighting correction is computed by comparing captured and reference chart patch col-
274 ors. This correction is applied uniformly to the sampled skin pixels, producing values that
275 approximate standardized, ideal lighting conditions.

276 Transformation into Lab Color Space

277 The corrected linear RGB values are converted into XYZ and then into Lab, a device-
278 independent and perceptually uniform color space. Multiple Lab samples from different
279 regions are averaged to form the final skin-tone vector.

280 Cosmetic Shade Selection and Hex Output

281 The final Lab vector is matched against a database of known cosmetic shades, each
282 precomputed in Lab. The system selects the shade with the smallest perceptual difference
283 (ΔE). The algorithm returns the shade as: a hex code, corrected RGB, and Lab code.

284 Integration with Hardware Dispensing Logic

285 The selected shade is mapped to pigment ratios for the dispenser hardware. The backend
286 provides:

- 287 • `/analyze`: accepts an image and returns the computed shade,
- 288 • `/dispense`: controls Raspberry Pi motors to dispense pigment.
- 289 • `/ping`: returns status updates and process health

290 4.3 Third Iteration**291 4.3.1 Final Product**

292 [Foundation Fix Github](#)

293 Chapter 5

294 Electrical Systems Design

295 5.1 First Iteration

296 5.1.1 Initial Wiring Diagrams

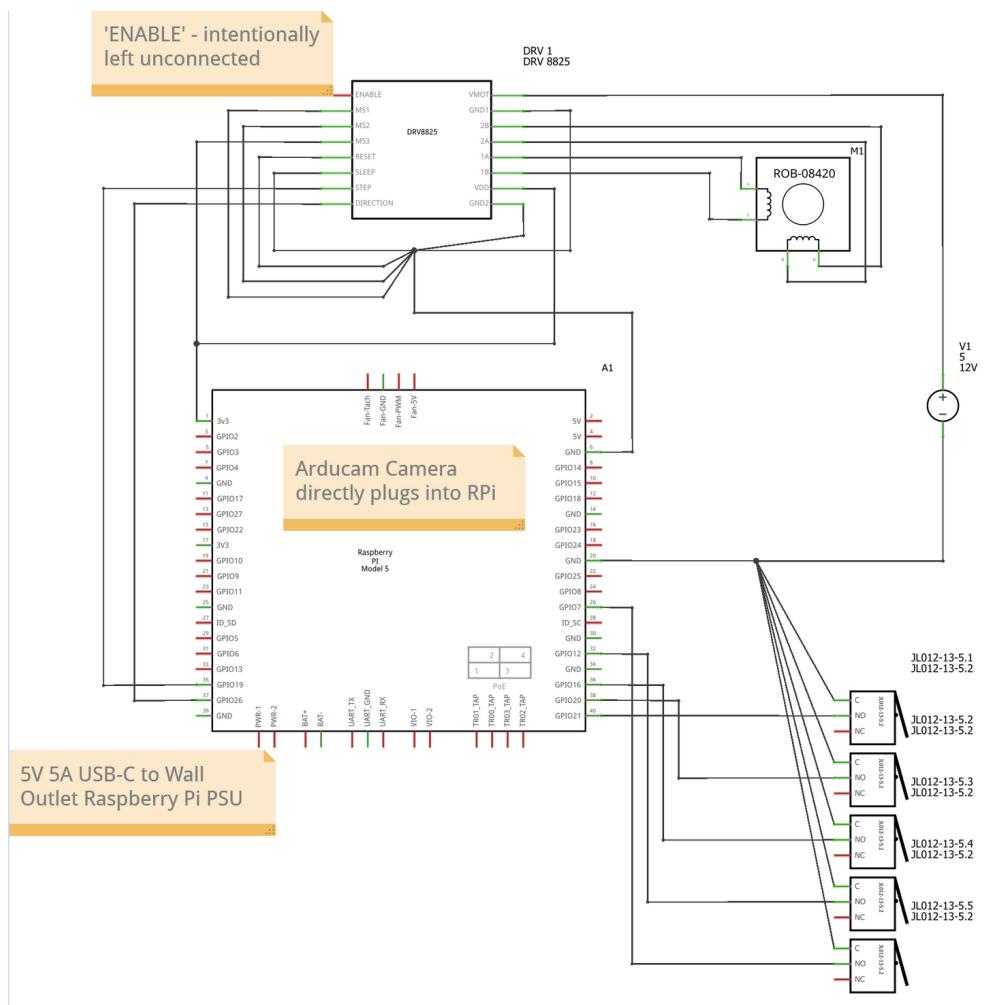


Figure 5.1: Preliminary Wiring Diagram

297 **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

298 $P_{motor} = 2 * I^2 * R$

299 $P_{motor} = 2 * (0.7)^2 * 4.0 = 3.92 * 5 = 19.6W$

300 $P_{RaspberryPi} = 10W$

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

301 **5.2 Second Iteration**

302 **5.2.1 Circuit Building**

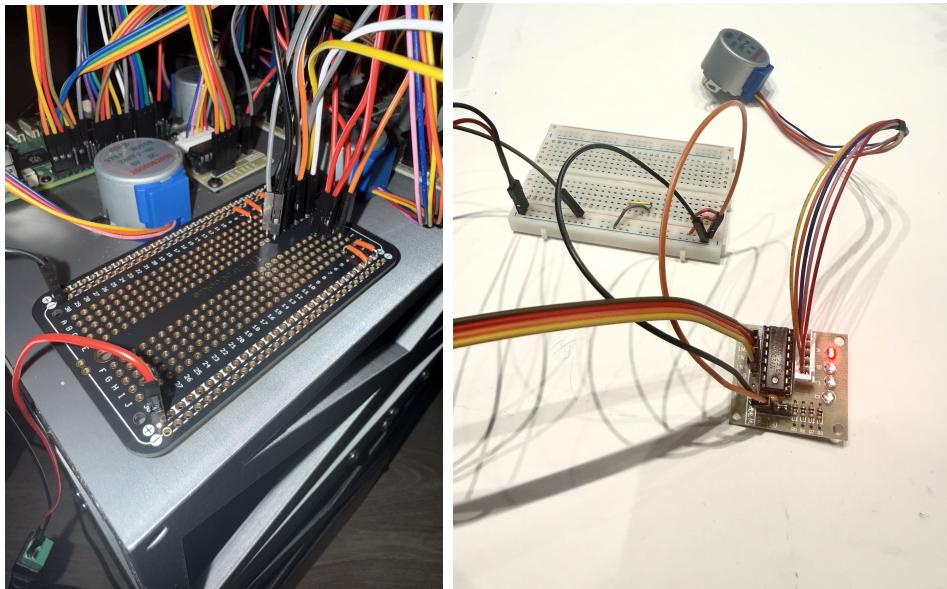


Figure 5.2: Perf Board and Servo Motor to Driver Connection

303 During the second iteration, several key changes were made to the electrical system. The
 304 original 12 V NEMA 17 stepper motors and their corresponding drivers were replaced with
 305 5V 28BYJ-48 stepper motors paired with ULN2003 driver boards. This allowed the removal
 306 of the 12 V power supply and enabled the entire system to operate from a single 5V source.
 307 The wiring diagram was updated accordingly to reflect these component substitutions.

308 5.3 Third Iteration

309 5.3.1 Final Wiring Diagrams

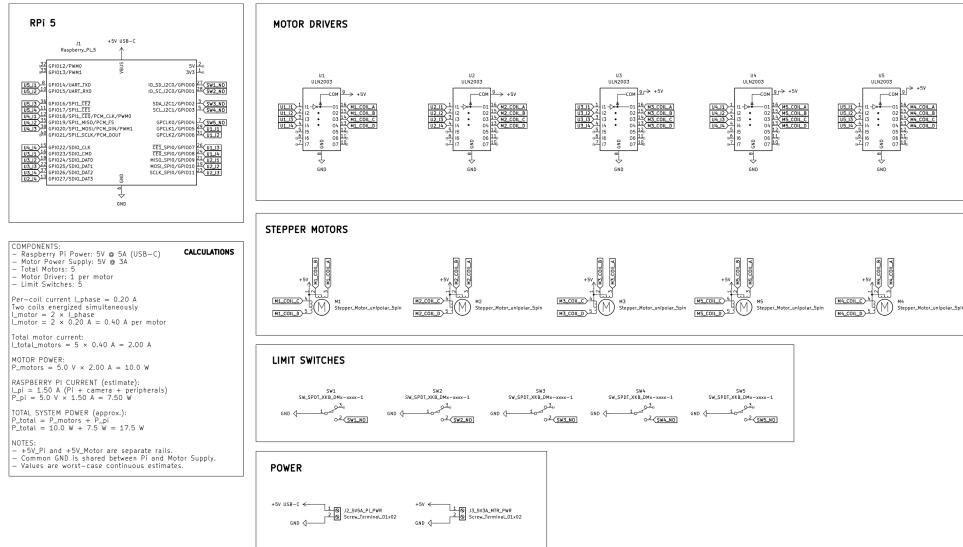


Figure 5.3: Final Wiring Diagrams

310 5.3.2 Final Power Calculations

311 Per Motor:

Approximately 100 mA per coil 2 coils 5 V $\bar{1}$ W

312 Raspberry Pi 5:

Average consumption of 1.2A 5 V $\bar{6}$ W

313 Total System Power:

$$P_{motor} * 5 + P_{pi} = 1W * 5 = 11W$$

₃₁₆ 5.3.3 Circuitry

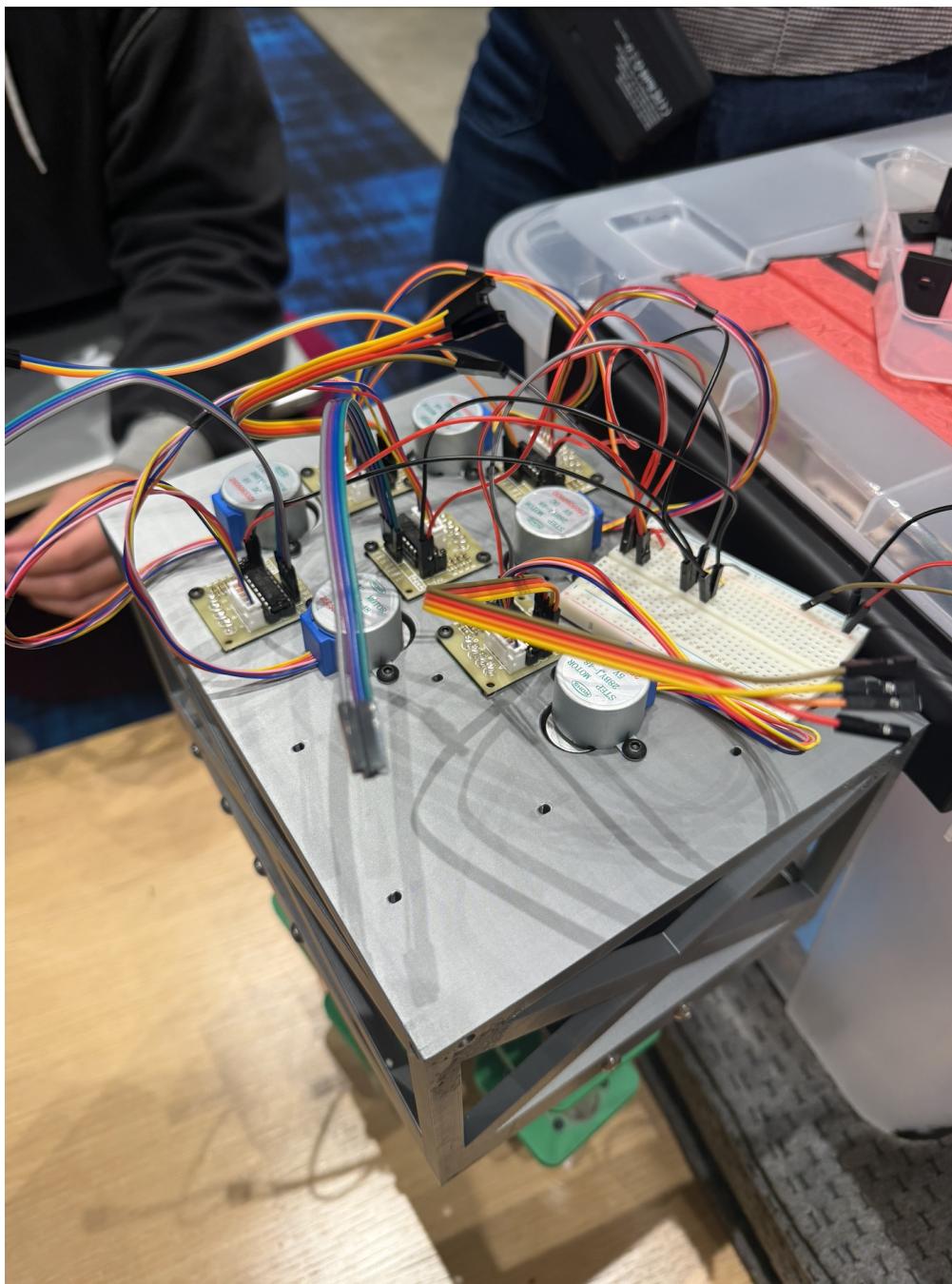


Figure 5.4: Final Circuitry Under Housing Lid

₃₁₇ The only major change from the prototype stage was the transition from breadboard
₃₁₈ wiring to permanent soldered connections on a perfboard. This improved durability, reduced
₃₁₉ wiring instability, and prepared the circuitry for integration into the final enclosure.

³²⁰ Chapter 6

³²¹ Final Product



Figure 6.1: Dispenser Systems Fully Assembled

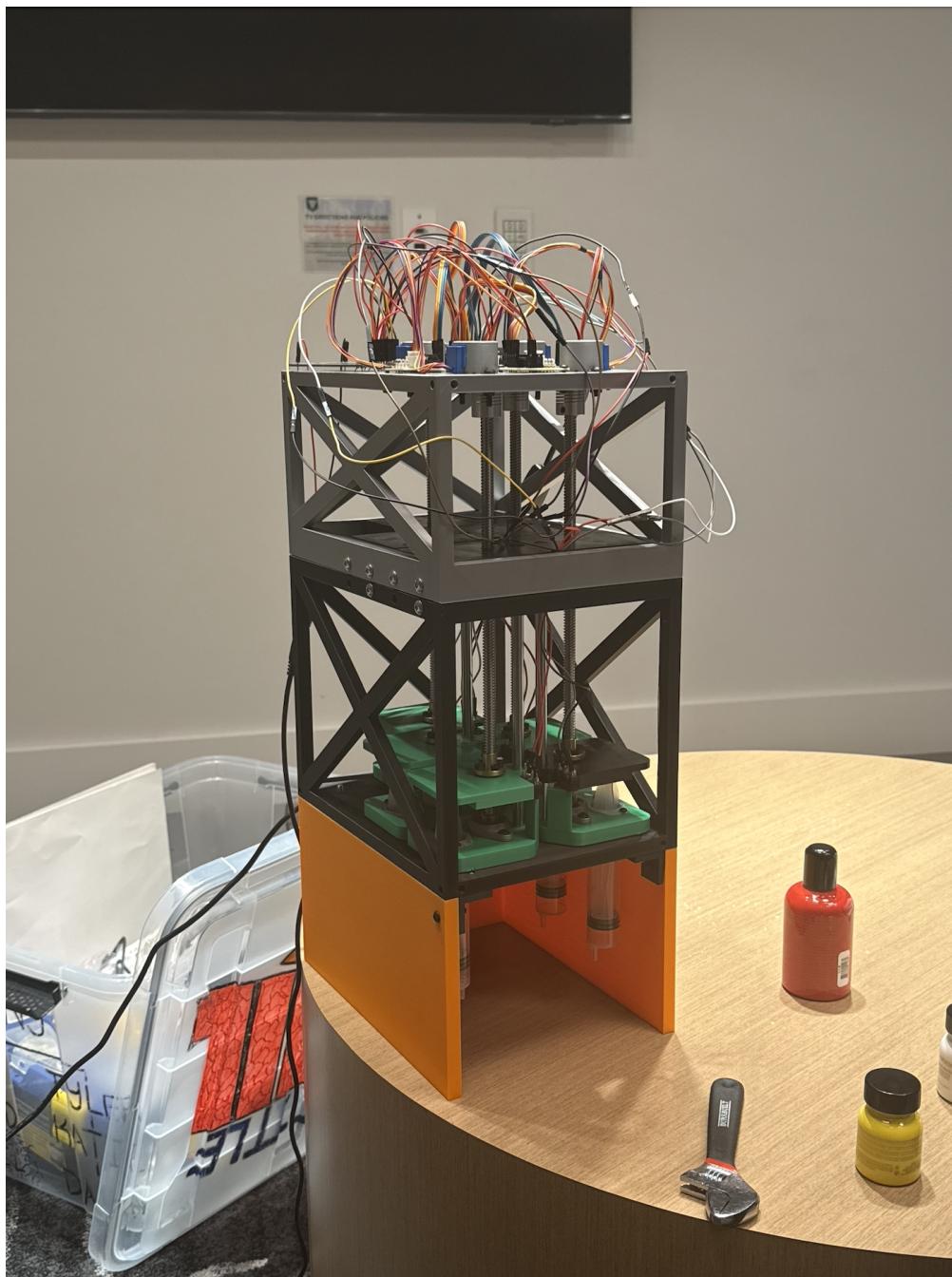


Figure 6.2: Skeleton/Internal Structure

³²² **6.1 Testing Protocol**

³²³ **6.2 Integration**

³²⁴ **6.3 System Performance**

³²⁵ Chapter 7

³²⁶ Discussion

³²⁷ 7.1 Limitations

³²⁸ 7.2 Future Work

³²⁹ **Chapter 8**

³³⁰ **Conclusion**

³³¹ **8.1 Bill of Materials**

| Component | Purpose | Qty | Price |
|---|----------------------------|-----|-----------------|
| 2PCS 300mm Tr8x8 Lead Screw + Brass Nut | Linear motion | 3 | \$13.99 |
| 5mm–8mm Lead Screw Coupler (5 pack) | Motor–screw coupling | 1 | \$9.99 |
| Linear Motion Rod Guide 8mm × 200mm | Linear guidance | 3 | \$8.69 |
| 8mm Flange Pillow Block Bearing | Shaft support | 2 | \$8.99 |
| Raspberry Pi 5 (4GB RAM) | Main controller | 1 | \$66.00 |
| 5 Pack Plastic 30mL Syringes | 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 |
| Wiring Kit | Electrical connections | 1 | \$6.98 |
| 5V 3A DC Power Supply | Low-voltage power | 1 | \$7.47 |
| 5 Sets 28BYJ-48 + ULN2003 Driver | Secondary actuation system | 1 | \$14.99 |
| M2 Spacers | Mechanical spacing | 1 | \$6.69 |
| Perfboard / Breadboard | Circuit prototyping | 1 | \$8.79 |
| Mehron Liquid Makeup Colors | Testing material | 1 | \$50.00 |
| Large 3D Prints (>4 hrs) | Structural components | 1 | \$144.00 |
| Total Estimated Cost | | | \$441.12 |

Table 8.1: Updated Bill of Materials.

³³² References

³³³ Bibliography

- ³³⁴ [1] Sophie Smith. Billions of beauty packaging goes unrecycled every year – theindustry.beauty. 2024. Accessed 2024.
³³⁵