

PUMPED

The Smart Portable System for Weighing, Labeling, and
Tracking Breast Milk

Senior Design 2 Final Report

Group 15

Members:

Electrical Engineers:

Jimmie Rawls

Kevin Fernandez

Computer Engineers:

Agustin Rodriguez

Eric Hernandez



Project Sponsors:

VICTORIA RODRIGUEZ

MATTHEW ABERMAN

Mentors:

Dr. Lei Wei

Dr. Weeks

Reviewer Committee:

Dr. Wei Sun

ECE

Assistant Professor

Dr. Azadeh Vosoughi

ECE

Professor

Mark Maddox

ECE

Adjunct Professor

Dr. Justin Phelps

ECE

Adjunct Professor

List of Tables.....	VI
List of Figures	VII
Chapter 1: Executive Summary.....	1
Chapter 2: Project Description.....	2
2.1 Motivation and Background	2
2.2 Review of Existing Devices	3
2.21 - Smart Milk Bag Labels	3
2.22 - Walz Breast milk Labeling Device System	3
2.23 – Dr. Talbot’s Mom Milkfresh Breastmilk Freshness Sensors	4
2.3 Project Goals and Objectives	5
2.4 Project Features	6
2.5 Requirement Specifications	7
2.6 Prototype Image/Blueprint	8
2.7 House of Quality	12
Chapter 3: Research and Investigation.....	13
3.1 Technologies	13
3.1.1 Microcontroller.....	13
3.1.1.1 Why use an MCU? Why not use an FPGA?	13
3.1.1.2 MSP430FR6989	14
3.1.1.3 Raspberry Pi Pico	14
3.1.1.4 Raspberry Pi Zero 2 W.....	15
3.1.1.5 Arduino Nano	15
3.1.1.6 Arduino Uno R4 Wi-Fi	15
3.1.1.7 Arduino MKR Wi-Fi 1010	16
3.1.1.8 ESP32S 30-pin	16
3.1.1.9 Comparison Between All the MCU’s Named	17
3.1.1.10 MCU of Choice	18
3.1.2 Printers	18
3.1.2.1 Inkjet Printers	19
3.1.2.2 Laser Printers	19
3.1.2.3 Thermal Printers	19
3.1.3 Printer Interfacing Protocols	20
3.1.3.1 Bluetooth	21

3.1.3.2 Wi-Fi.....	21
3.1.3.3 Ethernet	21
3.1.3.4 TTL	22
3.1.4 Display	22
3.1.4.1 Different Types of displays.....	23
3.1.4.2 Honyon ILI9341 2.8-inch Display	23
3.1.4.3 GoDiyModules ST7789 1.6-inch Display	23
3.1.4.4 HiLetgo SSD1306 1.3-inch Display	24
3.1.4.5 HiLetgo SSD1309 2.42 - Inch Display	24
3.1.4.6 Waveshare 4.2 Inch E – Paper Ink Display	24
3.1.4.7 Waveshare 2.13 Inch E – Paper Ink Display	25
3.1.4.8 Display of Choice	25
3.1.5 User Interface Buttons	26
3.1.5.1 Why use physical Buttons	26
3.1.5.2 Button configuration.....	27
3.1.6 Weighing Technologies	27
3.1.6.1 Strain gauge load cells.....	27
3.1.6.2 Capacitive load cells	27
3.1.6.3 Piezoelectric sensor	28
3.1.6.4 Weighing Technology selection.....	28
3.1.6.5 Strain gauge load cell part analysis.....	29
3.1.6.6 ADC	29
3.1.6.7 ADS1256	29
3.1.6.8 HX711	29
3.1.6.9 ESP32 built in ADC	30
3.1.6.10 Our ADC selection.....	30
3.1.6.11 Strain Gauge.....	30
3.1.6.12 Wheatstone Bridge	31
3.1.7 Power Supply	32
3.1.7.1 Expected Power Demands.....	32
3.1.7.1.1 Power for ESP32 Microcontroller (MCU).....	32
3.1.7.1.2 Power for Load Cell (Scale) & HX711 Analog-to-Digital Converter (ADC)	33
3.1.7.1.3 Power for OLED Display	34
3.1.7.1.4 Power for Thermal Printer	34
3.1.7.1.5 Estimated Total Power Consumption	35
3.1.7.2 Rechargeable Battery Options.....	37
3.1.7.2.1 Battery Capacity Requirements	37
3.1.7.2.2 Lithium-Ion (Li-Ion) Batteries	37
3.1.7.2.3 Lithium Polymer (LiPo) Batteries.....	38
3.1.7.2.4 Nickel-Metal Hydride (Ni-MH) Batteries	40
3.1.7.2.5 Sealed Lead-Acid (SLA) Batteries	42
3.1.7.2.6 Portable Battery Banks	44
3.1.7.2.7 Battery Type Comparison and Selection	45
3.1.7.2.8 Battery Bank Comparisons and Selection.....	46
3.1.8 Voltage regulation	48
3.1.8.1 Linear Voltage Regulators	48
3.1.8.2 Switching Voltage Regulators	49
3.1.8.3 AMS1117.....	51
3.1.8.4 TLV767	51

3.1.8.5 LM2576	51
3.1.8.6 AP62300T	52
3.1.8.7 LM2679	52
3.1.8.8 MP2338GTL-P	53
3.1.8.9 Voltage Regulator Selection	54
3.1.9 PCB Software Applications	54
3.1.9.1 Different Types of Software's for Designing PCB	54
3.1.9.2 Eagle CAD	54
3.1.9.3 KiCAD	55
3.1.9.4 Altium	55
3.1.9.5 PCB Software Design of Choice	55
3.1.10 Software Tools and Environments	56
3.1.10.1 GitLab	56
3.1.10.2 GitHub	57
3.1.10.3 Visual Studio Code	57
3.1.10.4 Eclipse	57
3.1.10.5 Arduino IDE	58
3.1.10.6 PlatformIO	58
3.1.10.7 ESP-IDF	58
3.1.11 Android Application Technologies and Frameworks	59
3.1.11.1 HTTP REST API	59
3.1.11.2 Constrained Application Protocol (CoAP)	59
3.1.11.3 MQTT	60
3.1.11.4 Bluetooth Low Energy (BLE)	60
3.1.11.5 Node.js	61
3.1.11.6 Ktor	61
3.1.11.7 Firebase	61
3.1.11.8 Visual Studio Code	62
3.1.11.9 IntelliJ IDEA	62
3.1.11.10 Android Studio	62
3.1.11.11 Java	63
3.1.11.12 Dart	63
3.1.11.13 Kotlin	64
3.1.12 Embedded Thermal Printers	64
3.1.12.1 DFRobot GY-EP204x	64
3.1.12.2 Maikrt MC-EH300	65
3.1.12.3 Maikrt MC206H	65
3.2 Part Selection	66
Chapter 4: Standards and Design Constraints	67
4.1 Industrial Standards	67
4.1.1 PCB Design Standards	67
4.1.1.1 IPC Standards	68
4.1.2 Battery and Power Standards	68
4.2 Practical Constraints	70
4.2.1 Economic	70
4.2.2 Environmental	70

4.2.3 Ethical Constraints	71
4.2.4 Manufacturability Constraints	72
4.2.5 Sustainability	72
4.2.6 Health and Safety	73
4.2.7 Time Constraint	74
4.2.8 Political Constraint	75
Chapter 5: Comparison of ChatGPT/LLM Platforms	77
5.1 Chapter Introduction	77
5.2 Prompt Comparison	77
Chapter 6: Hardware Design	85
6.1 Hardware Overview	85
6.2 Hardware Block Diagram	86
6.3 Power Supply	86
6.3.2 Voltage Regulators	87
6.3.2.1- The 3.3V Voltage Regulator	87
6.3.2.2- The 5-V Voltage Regulator	88
6.4 Main PCB Schematics	89
6.4.1 ESP32-WROOM-32	89
6.4.2 CP2102 USB-to-Serial and ESP32 Flash Interface	91
6.4.2.1 USB-A Connector, Protection, and Auto Reset Circuitry	92
6.4.2.2 CP2102 USB-to-Serial Converter	93
6.4.3 Connectors for Peripherals & Testing	94
6.5 Peripherals	95
6.5.1 Load Cell & HX711	95
6.5.2 Thermal Printer	96
6.5.3 OLED Display	96
6.6 Overall Schematic	97
Chapter 7: Software Design	100
7.1 Device Software	100
7.1.1 State (Activity) Diagram	100
7.1.2 Use Case Diagram	101
7.1.3 Class Diagram	102
7.2 Mobile Application Software	103
7.2.1 Mobile Application Software	104
Chapter 8: System Fabrication/Prototype Construction	110
Chapter 9: System Testing and Evaluation	116
9.1.2.1 ESP32 Microcontroller	118
9.1.2.2 Load Cell/HX711	118

9.1.2.2 OLED Display	119
9.1.2.3 Thermal Printer.....	119
Chapter 10: Administrative Content	121
10.1 Project Budgeting / Cost	122
10.2 Project Milestones	122
10.3 Distribution of Work	125
Chapter 11: Conclusion	126
Appendix A: References and Disclaimer	130
Appendix B: Copyright Permission	132
Appendix C: Data Sheet	132
Appendix D: Software Code	133
Appendix E: ChatGPT Prompts & Outcome.....	136
1. Adafruit Mini Thermal Receipt Printer (Product ID: 597)	136
2. [Gainscha GP-5890 Series / ESC/POS Compatible Thermal Printers (58mm).....	136
3. MUNBYN Embedded Thermal Printer Module (58mm)	136
A few design notes:	137
My Recommendation:.....	137
1. Native Android Development (Java or Kotlin)	137
2. Flutter (by Google).....	138
3. React Native (by Meta).....	138
4. Kotlin Multiplatform Mobile (KMM)	138
5. Xamarin / .NET MAUI (by Microsoft)	139

List of Tables

Table 1 Requirement Specifications	8
Table 2 MCU vs FPGA	14
Table 3 MCU Comparisons	17
Table 4 MCU Comparisons Continued	17
Table 5 Printer Types and Their Capabilities	20
Table 6 Printer interfacing Protocols and Their Capabilities	22
Table 7 Display Comparisons	25
Table 8 Weight Sensor Technology Comparisons.....	28
Table 9 ADC comparison table	30
Table 10 Wheat Stone Bridge - Generated by ChatGPT	31
Table 11 MCU Power Consumption	32
Table 12 Scale/KX711 Power Consumption	33
Table 13 OLED Display Power Consumption.....	34
Table 14 Printer Power Consumption	34
Table 15 Power Consumption Summary by Component.....	35
Table 16 Estimated Total Power Consumption of PUMPED Device	36
Table 17 Technical Properties of Lithium-Ion Batteries	38

Table 18 Technical Properties of Lithium-Polymer Batteries	39
Table 19 Technical Properties of Nickel-Metal Hydride Batteries	41
Table 20 Technical Properties of Sealed-Lead Acid Batteries	43
Table 21 Technical Properties of Portable Battery Banks	44
Table 22 Battery Technology Comparisons.....	45
Table 23 Battery Bank Comparisons	47
Table 24 Linear Voltage Regulator Comparisons.....	49
Table 25 Switching Voltage Regulator Comparisons	50
Table 26 Comparing Voltage Regulator Technologies	51
Table 27 Selecting the Voltage Regulators.....	53
Table 28 Selecting PCB Software	56
Table 29 Selecting IDE	58
Table 30 Wireless Communication Protocols	60
Table 31 Firebase Information	62
Table 32 Android Studio	63
Table 33 Selecting Programming Language	64
Table 34 Maikrt MC206H Information	65
Table 35 Project Part Selection.....	66
Table 36 Bill of Materials	122
Table 37 Project Initialization Milestone	123
Table 38 Project Fabrication Milestone	124
Table 39 Distribution Of Work.....	125

List of Figures

Figure 1 Smart Milk Bag Product.....	3
Figure 2 Walz Breast Milk Labeling System.....	3
Figure 3 Dr. Talbots Smart Milk Fresh Smart Labels	4
Figure 4 Project Sketches	8
Figure 5 Hardware Block Diagram.....	10
Figure 6 Software Block Diagram.....	11
Figure 7 House of Quality	12
Figure 8 Hardware Block Diagram.....	86
Figure 9 3.3 V Voltage Regulator	87
Figure 10 Voltage Regulator Inductor Chart.....	88
Figure 11 5V Voltage Regulator	88
Figure 12 5V Voltage Regulator Inductor Chart	89
Figure 13 MCU Schematic	90
Figure 14 Pullup Resistors.....	91
Figure 15 Boot and Reset Buttons	91
Figure 16 USB-A Connector Schematic	92
Figure 17 UART- USB to Serial Converter Schematic	93
Figure 18 Peripheral Connectors/ Testing Pins	94
Figure 19 Load Cell/HX711 Schematic	96
Figure 20 Overall Schematic	98
Figure 21 Software Activity Diagram	101
Figure 22 Use Case Diagram.....	102
Figure 23 Class Diagram	103

Figure 24 First Time App Login	105
Figure 25 Mobile App General Functionality	106
Figure 26 Mobile App - Main Page & Device Link	107
Figure 27 Mobile App - Data Base & Dietary Info	108
Figure 28 Mobile App - Settings	109
Figure Thermal Printer	116
Figure OLED display.....	117
Figure 10kg load cell	117
Figure Analog to Digital Converter (HX711)	117
Figure ROMOSS Power Bank.....	118
Figure Overall System - Demo	120

Chapter 1: Executive Summary

Breastfeeding mothers have always gone through many difficulties when breastfeeding their children, especially in the middle of the night. It has always been a problem because the only way that mothers are to accurately label their breastmilk bags has always come down to writing on each individual breastmilk bag with a marker. This is an issue because there can sometimes be a mistake in the process. Mothers are going to be tired, and this can cause an issue when writing on the bag. The goal of our device is to alleviate that issue. We want mothers to be certain, to have peace of mind knowing that the next time they want to feed their child a breastmilk bag that they had previously pumped, they will have all the details on the bag labeled with accuracy and without issues. We also want mothers to be able to easily label their breastmilk bags with ease, with a click of a button.

PUMPED is an innovative, portable, easy to use device that is targeting one of the most overlooked but essential parts of breastfeeding. Labeling pumped breastmilk with ease. With a single click of a button, Pumped will print an adhesive label and will apply accurate, ready to read labels that will include date, time, and most importantly the weight of the milk inside the breastmilk bag in ounces. There are many different portable weighing devices that allow people to measure different things that they would choose to and create labels, but none of those devices will be able to compete with PUMPED and its features that it will contain.

What truly sets PUMPED apart from any other breastmilk labeler is its seamless integration with a companion mobile app. This app will contain a very easy and simple user interface that will be soft on the eyes, proper accessibility, and its simple yet effective features that will target all the required key points. Each label will come with a unique QR code that, when scanned, automatically logs vital information into the app and will be stored very neatly so that when a mother or any other person wants to retrieve that information, they will do so without any problem and very quickly. This allows moms to track milk batches, monitor expiration dates, and maintain a digital record of their breastfeeding journey, all from their smartphone.

In this report we are going to include many different types of content that will record a lot of information corresponding to the development of PUMPED. We are going to conduct a lot of information regarding different parts of the build such as MCU, power, scale, software, and hardware aspects. We are going to create many different diagrams that will illustrate the hardware and software designs. We are going to go into standards and design constraints in details, make a comparison of our research against ChatGPT, record our testing procedures and system fabrication, and we will contain all the administrative content such as work distribution, budgeting, bill of material, and project milestones.

With all that being said, our PUMPED device will assist mothers that breastfeed or will do so with ease and will encourage them to have a better time doing so. We want mothers to really enjoy the process, because they are very important.

Chapter 2: Project Description

2.1 Motivation and Background

Taking care of a newborn is not for the weak, but that's why mothers are so strong. Mothers gladly give up their time and attention to make sure their newborn has everything they need. Perhaps one of the most important needs for a newborn is their mother's milk. Babies need constant feeding, and it is up to moms to have breast milk ready for them when they need it.

To keep up with this demand, mothers will painstakingly pump their milk at all hours of the day while also having to meticulously label each and every bag they pump. This process usually entails the mother having to write on the bags with a marker or pen to include information like the date and time the serving was produced, the volume of the serving, allergens, and any dietary considerations a serving may contain. Time is an important factor for pumped breast milk because milk that is pumped at night affects babies differently than milk that is pumped in the morning. Not only that, but mothers also must keep track of their entire stock of breast milk to make sure they can use it before it expires.

What makes matters worse is that if a mother forgets to label, or improperly labels their pumped milk, that serving could end up in the trash, leading to unnecessary waste not only for their baby but for those to whom that milk could have been donated. Thus, labeling and keeping track of breast milk is a critical undertaking. PUMPED, our senior design project aims at eliminating the tediousness of labeling and documenting breast milk servings so that a mother can focus on more important things. When a mother has finished pumping her milk, she can place the filled container on our portable device and it will weigh the milk, convert the weight into fluid ounces, and then print a label from the device's integrated label maker.

The label's information will contain a bag number for ease of tracking, the date and time the milk was pumped, the volume, allergens, and any dietary considerations. Our device will be paired with a mobile app that keeps a running tab on all the milk servings a mother has pumped and will push notifications on servings that are close to expiring. The app will also be able to take input commands so that the labels can be customized to meet any requirements needed for institutions that accept donations.

Our sponsors, Victoria Rodriguez and Matthew Aberman pitched their real-life problem to us, and we jumped at the chance to find a solution. As parents, Victoria and Matthew have had plenty of experience with the tediousness of the labeling process. There is no set schedule to start pumping, an example of this was when Victoria had to wake up at two in the morning to pump breastmilk. She had little to no enthusiasm for filling out the details on the bag much rather opting to wait till the morning. However, as life tends to get in the way, those important details soon find themselves forgotten and a precious serving of milk becomes a mystery.

Another important motive our sponsors had for commissioning this project were the possible implications it could have on the donation side of things. Many hospitals need and accept breast milk donations for newborns whose mothers cannot provide their own. These babies are often critically ill and are in the infant version of the intensive care unit. A breast milk donation can be paramount in saving those babies' lives. Even with a hospital's great

need for breast milk they would still have to reject improperly labeled samples. Hospitals follow specific guidelines that ensure safety and traceability for their breast milk samples. So, with a product like PUMPED in the picture, all the aforementioned specifics that come with a serving of breast milk, could become included in a standardized system where mothers, hospitals, milk banks, and any other entity that deals in breast milk, can meet a baby's needs without the worry of improperly labeled milk.

2.2 Review of Existing Devices

2.21 - Smart Milk Bag Labels

Smart Milk Bag Labels Created by the Cubtale store, being sold on amazon for \$14.99 contains many features that are very important to consumers. This device contains a set of numbered labels that are in the range of 1 to 500. The labels contain a few sets of details. For example, each contains a designated number of four sections corresponding to date, time, volume in ounces, and name. It also contains a QR code that will allow you to connect to the mobile app provided. The physical labels are very simple, they are not self-labeled, so a mother must label them individually each time. The main selling point of this device is that it comes with a mobile app containing many features such as, expiration date alerts, information on CDC guidelines, proper handling instructions, temperature descriptions, caregiving instructions, and comprehensive milk tracking that will allow a mother to track the total amount of bags on hand.



Figure 1 Smart Milk Bag Product

2.22 - Walz Breast milk Labeling Device System



Figure 2 Walz Breast Milk Labeling System

The all-in-one breast milk labeling system developed by Walz offers a portable solution for mothers to label their breastmilk bags. It is advertised as being simple to operate by being plug and go, having mobility due to a stand-alone ergonomic rolling cart, label printer that allows for label customization and is easy to handle. The built-in keyboard allows the consumer to add the baby's name, mother's name, current medications, and pump date. It does not contain a mobile application, so it has its limitations. Due to this product size, it seems to be aimed specifically at the medical industry instead of an average mother. To obtain the price of this device, the consumer is required to request a quote from the manufacturer. This information comes off as the product being on the expensive side.

2.23 – Dr. Talbot's Mom Milkfresh Breastmilk Freshness Sensors

Dr. Talbot's MilkFresh sensors are designed to help mothers keep track of the breastmilk they have stored, and it helps with quality control via the temperature indicators. Each sensor is a peel-and-stick label that has a QR code and a temperature indicator that changes color if the milk has been exposed to potentially unsafe temperatures. The inventory tracking is done via the application called "MilkFresh". When scanned with the MilkFresh App, the sensor's activation date is recorded, and users can manually input the amount of milk stored in each bag. The app's inventory section organizes stored milk by bag number, volume, and activation date. The user can scan the QR code of an activated bag to check the quality of the milk. After scanning the QR code, the app provides the different expiration dates based on whether the milk is kept in the refrigerator or freezer. This product allows mothers to monitor storage conditions and manage their breastmilk supply through the app's tracking features and ensures quality through the temperature indicator on the label.



Figure 3 Dr. Talbots Smart Milk Fresh Smart Labels

Summary:

Looking into existing products on the market right now, there are a lot of key takeaways and small details that are common in both devices. But there are also many differences to each. Both devices are aimed to tackle the same problem of breast milk management for mothers, but each comes with its own limitations.

In the case of our sponsored project, we are taking a different approach from the three devices mentioned above. We are planning on taking all of the smart features that each of these devices offer and bring them into one single small device that is capable of features such as: easy to use, small and compact, user friendly, mobile application ready, abides by the CDC guidelines, prints labels with pre-written information, and much more. One major takeaway from reviewing similar devices and products is that there does not appear to be a readily available, consumer grade device that provides the all-in-one functionality, while still being portable, that we are planning on creating with our own device.

2.3 Project Goals and Objectives

Our overall goal is to create an easy-to-use device that enables mothers to automatically categorize and manage their inventory of breastmilk to ensure feeding their child is as simple as it can be. To achieve this overall goal with our PUMPED project we have numerous smaller goals that we have classified as basic, advanced, or stretch goals. The completion of our basic goals will ensure we have a minimum viable product with our core features successfully implemented.

The advanced goals represent tasks that if successfully completed will add auxiliary features or enhance our existing core features. Our stretch goals are ones in which we acknowledge that we may not have enough time allocated to us in this project to achieve them due to their lower priority relative to other features or due to their overall complexity. We are confident that all of the advanced goals are met and that we had time to complete 1 or 2 stretch goal items, but these goals are of course not guaranteed to be completed or even fulfilled in the exact way as they have been written here.

Basic Goals:

- The device accurately weighs the breastmilk bag placed on the scale portion of the device.
- Device prints a label for bag containing volume, time, date, and allergen/dietary details.
- An integrated app displays current breastmilk stock for user.
- The device is portable and easily carried around.

Advanced Goals:

- The App notifies users that a breastmilk bag is approaching its expiration date.
- The app shows detailed daily/weekly/monthly production stats.
- Users can add detailed allergen information to each bag.
- LCD screen displaying various bag informationo

Stretch Goals:

- Hospital Admin users can send requests to all individual users in the area for donations.
- Individual users can schedule drop-off times with local hospital / milk bank for breastmilk donation.
- Automatic shutdown of device depending on certain conditions.

2.4 Project Features

Entire device

- Portability: The device must meet the required specifications for weight and must be sized so that the average consumer can easily pack their device and use it while traveling, vacations, road trips, etc.
- Rechargeable: Since babies and their needs can be unpredictable, the device needs to be rechargeable so that mothers can use it on the go or wherever they may need it. It can be an inconvenience to have to go buy new batteries when the charge runs out, so that is another issue that a rechargeable battery can solve.
- Supports a variety of breast milk containers: There are hundreds of different breast milk bags that can be bought online, they vary in size, shape, and materials. Our device should be designed so that most breast milk bags / containers can be placed into the scale and weighed.
- Simple to understand user interface: A simple user interface is necessary for the average consumer to easily use the device and application. Time is an essential resource for all mothers and a simple interface will allow for efficient label printing.

Scale

- Measures milk weight: The user will place the breastmilk bag upon the scale, and it will automatically weigh the bag, and that information will be processed by the rest of the device.

Thermal printer

- Prints labels: Our thermal printer will be printing labels that contain some sort of identifier such as a bag number so that the user can distinguish between bags physically. Additionally, things such as date, time, volume, and allergen/dietary sections will also be printed on this label.
- Meeting labeling standards: Hospitals and milk banks which typically take in and process donated breastmilk have certain standards for how they should be annotated so we ensured that these specific standards are met with the format of our labels.

Display

- Provides volume: The LCD screen on our device will depict basic information about the breastmilk bag which has been placed upon the device. Things such as the volume in fluid ounces and other miscellaneous information may also be depicted.
- Options screen: On an option screen users will be able to modify basic settings to make the use of the device more convenient for their own purposes.

User interface buttons

- Scale reset: The scale reset button will reset all peripherals such as the display, scale, and the printer for troubleshooting purposes
- Tares button: The Tares button will be used to zero the scale in the case that the user wants to get an accurate measurement of added weight to a container.

- Mode selection: Mode selection will allow the user to switch units of measurements such as grams, ounces, etc.
- Print Label: After the user places the breastmilk bag onto the device's scale, they will then be prompted to hit the print label button at a time of their own choosing. This will then print out a label with all the necessary information needed to organize the milk bags.

Bluetooth/Wi-Fi capabilities

- Phone connection: The device will be integrated with a mobile app that can deliver certain notifications to a user.
- App connection: The app will be receiving breastmilk bag data from the MCU likely either via Bluetooth or Wi-Fi connections

Application

- Tracks milk inventory: Users will be able to consult the app to view their current inventory of breastmilk bags.
- Tracks milk production statistics: Users will be able to view basic statistics of their breastmilk production while also breaking it down by day/week/month.
- Notifies users of upcoming expiration dates: Users will receive via the app a notification that previously stored breastmilk bags are soon to expire, and they will be also shown this information within the app itself.
- Takes input for dietary/allergen considerations: Users will be able to record detailed notes within the app for individual bags regarding what dietary or allergen concerns they may have regarding particular bags.
- Overview of milk storage
- Label templates

2.5 Requirement Specifications

The specifications and engineering requirements for this project can be found in Table 1. The requirements for this project revolve around making a device that is portable and usable for extended periods of time outside the home. The specifications written here represent our core functionality and the general expectations which we have for them and how they will perform in the field. We anticipated that these listed engineering requirements were feasible and that we have been able to meet all of these engineering requirements without any significant issues.

Table 1 Requirement Specifications

Pumped Device	
Specification	Engineering Requirement
Scale/weight sensor should be accurate	Accurate within +/- 3 grams
Time to print after print button pressed should be almost immediate	5 seconds or less
Time for recorded breastmilk bag data to be sent to mobile app should be short	5 seconds or less
Time for breastmilk bag to be weighed after placed on scale almost immediate	3 seconds or less
Device weight should permit it to be portable	8 pounds or less
Battery Voltage	12V or less
Battery Capacity	At least 10 hours
Battery lifespan	At least 2 years

2.6 Prototype Image/Blueprint

The two images shown below are prototype sketches that were drawn by team members Eric and Kevin, respectively. They both portray similar main functionalities needed to satisfy our requirements such as having a scale to weigh the bags of milk, a display to show the user the ounces weighed, a printer, and a removable container for ease of cleaning in case of milk spills. These drawings were made to portray what our working designs could look like. Of course, the design is subject to change as we gather more information on which printer we will be using, the size of our load cell, battery pack, and PCBs. We expect these to evolve as further information is acquired and as we run into issues.

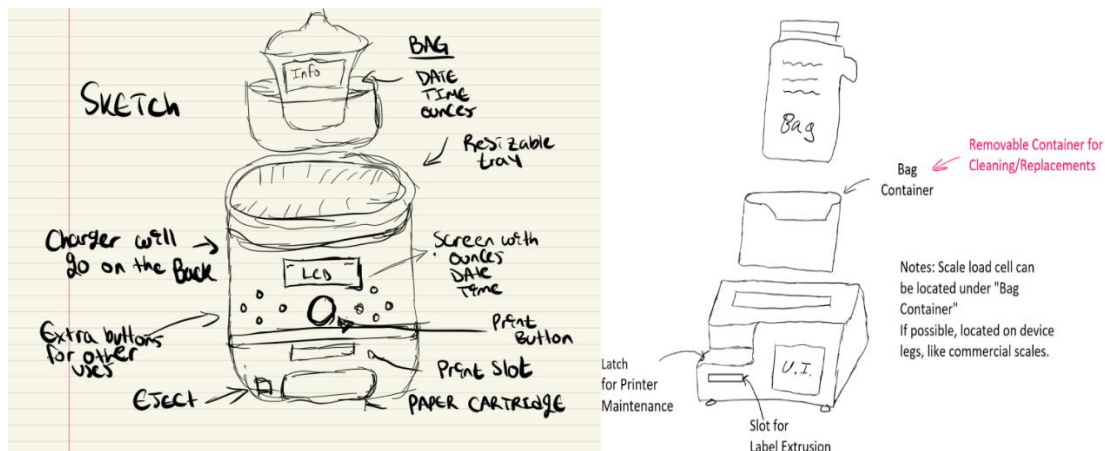
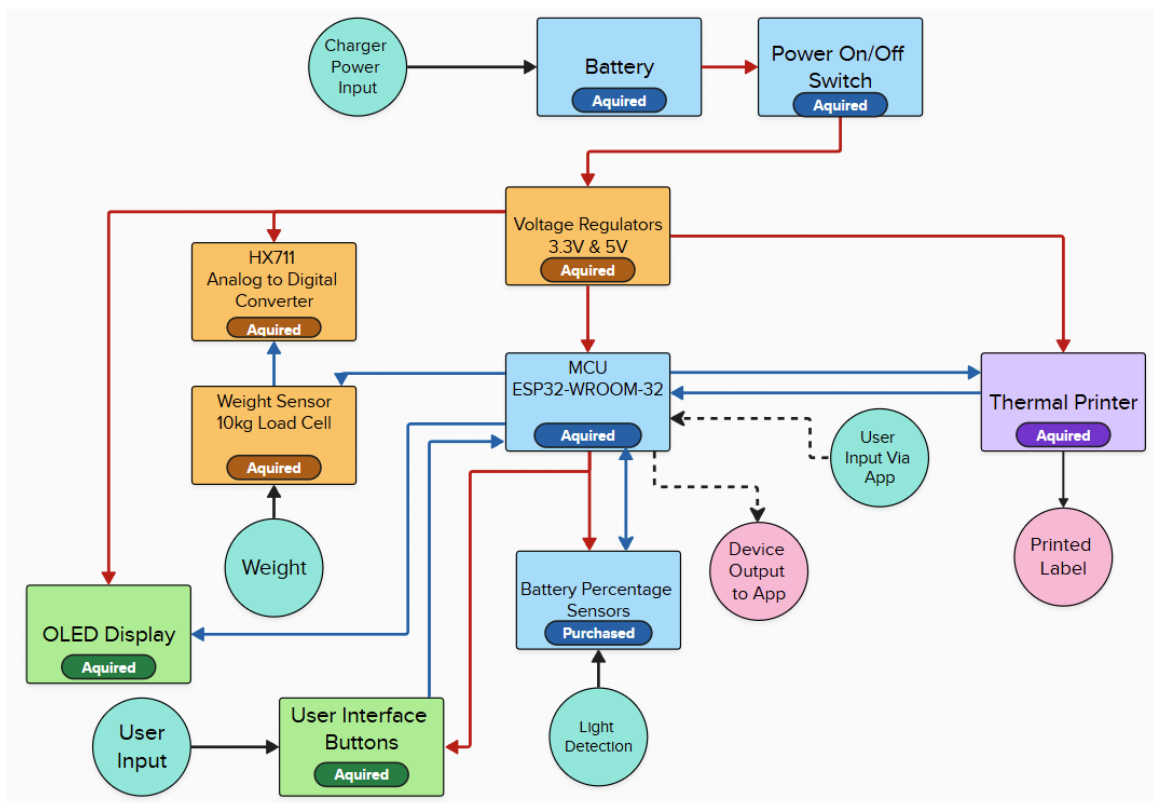


Figure 4 Project Sketches

2.7 Block Diagrams

The diagram pictured below depicts the PUMPED device's hardware connections, and the color shows work distribution by group member. Starting from the top, it shows the connection of the power bank to the power on/off switch. After the switch comes the 3.3 voltage regulator and the 5-volt connection to the thermal printer. The regulator then points to the MCU and all the peripherals. The MCU connects to the thermal printer, OLED display, load cell, ADC, battery percentage indicator, and buttons



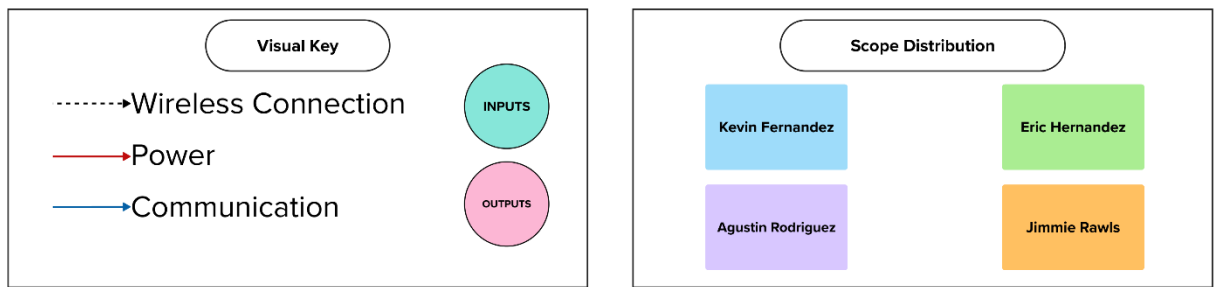


Figure 5 Hardware Block Diagram

The software diagram below shows the PUMPED device's software flow when in operation, and the color shows work distribution by group member. Once the device turns on, the printing and weighing operations can take place following their respective flow paths. The display will continuously display data sent to it by the MCU. The scale will function by sending weight data to the MCU, and the button can operate the tare function. The thermal printer will remain on standby until the print button is pressed. Once a printing operation is started, data is sent to the MCU about the print job, and then that data is transmitted to the mobile app via Bluetooth.

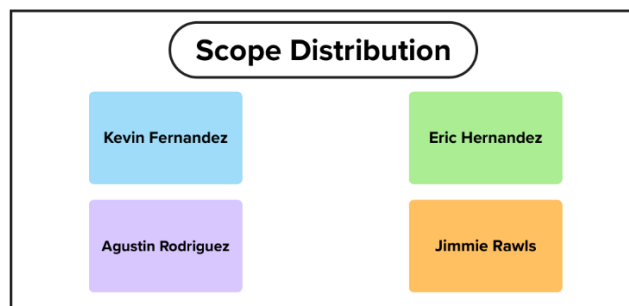
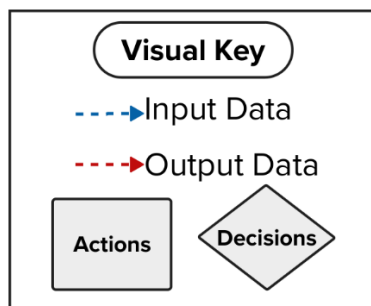
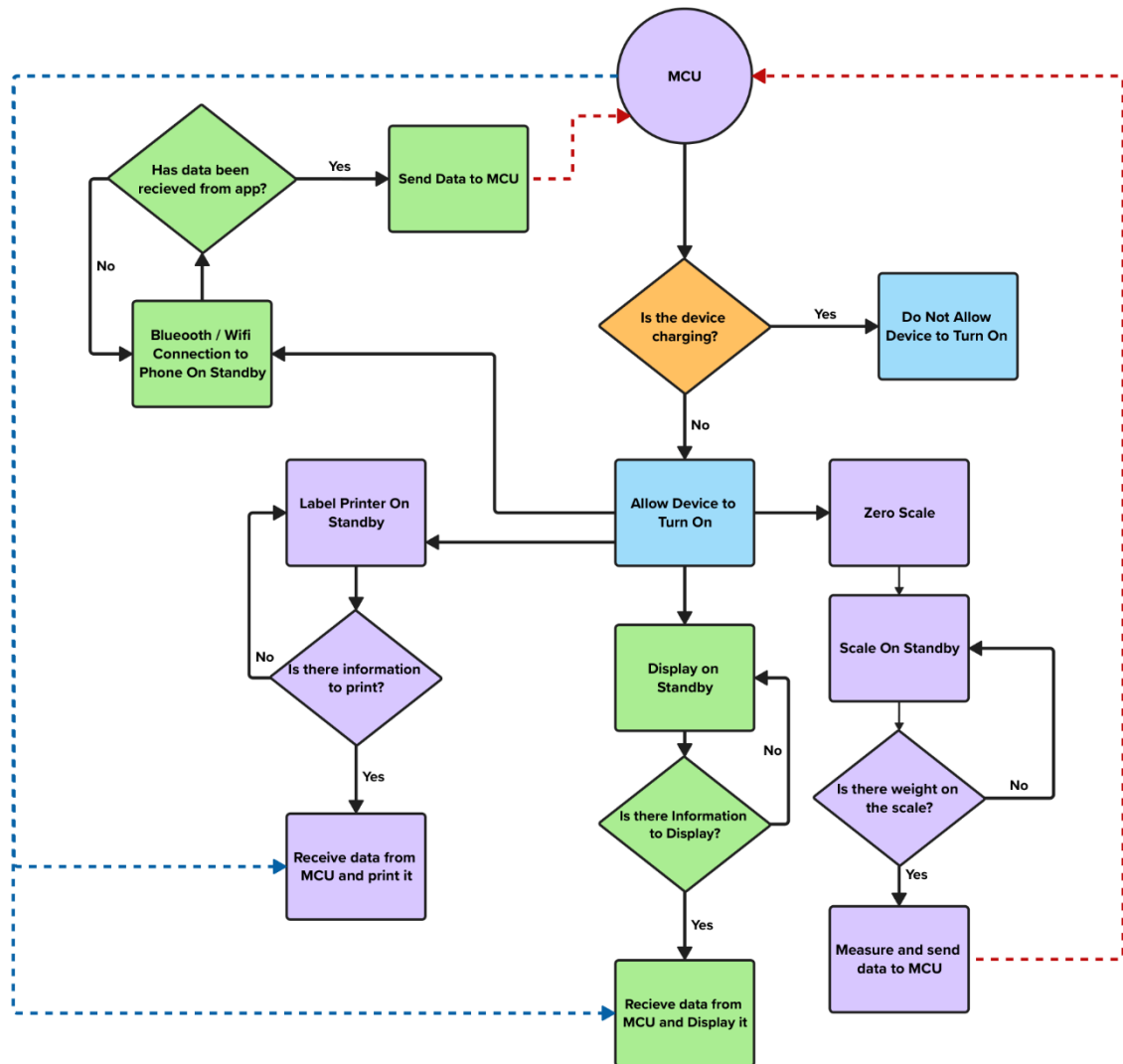


Figure 6 Software Block Diagram

2.7 House of Quality

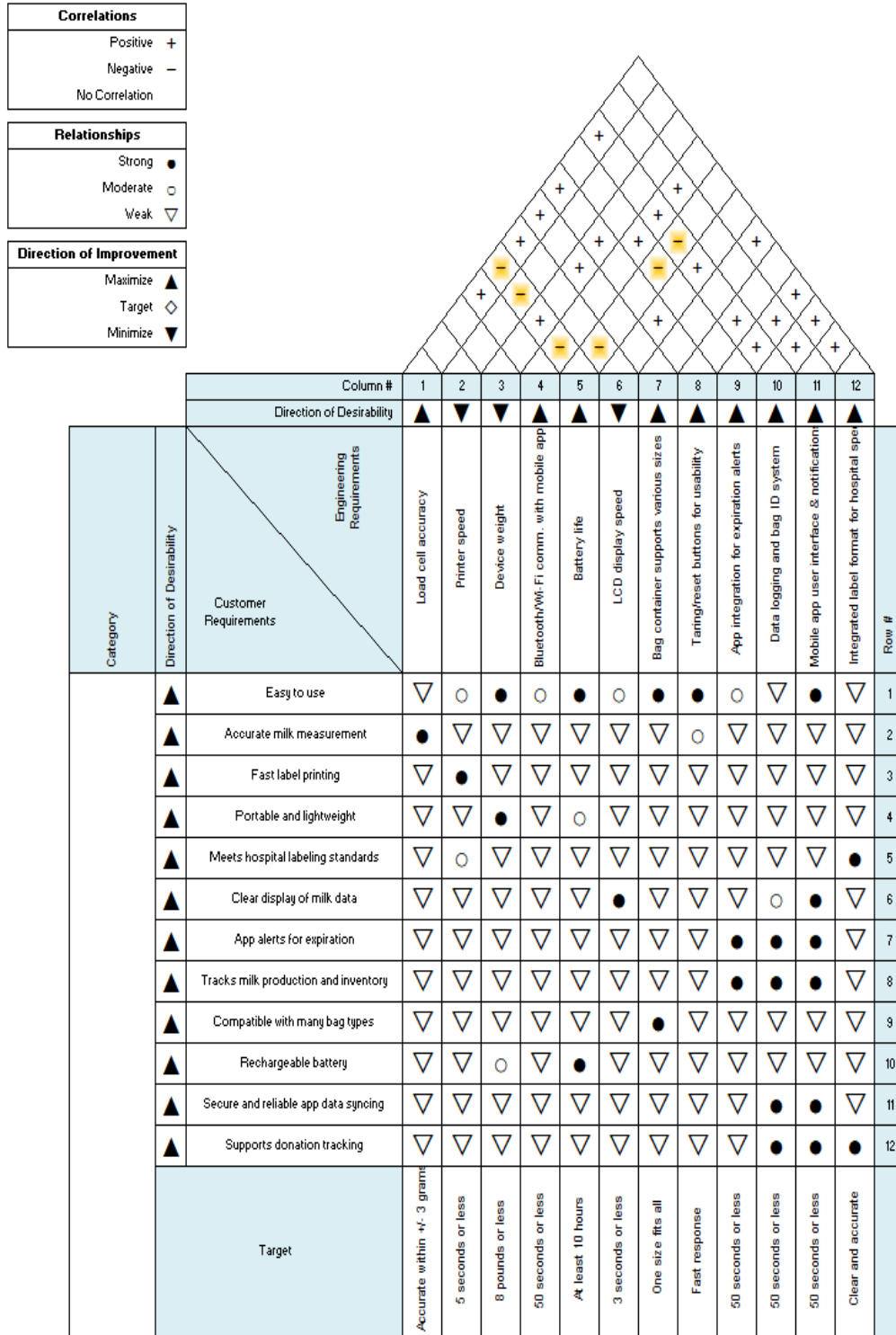


Figure 7 House of Quality

Chapter 3: Research and Investigation

3.1 Technologies

This chapter takes a deep dive into the technologies that make our senior design project, PUMPED, possible, (The smart, portable system for weighing, labeling, and tracking breast milk). We start by looking at the microcontroller, the brain of the operation, before exploring different types of printers and the ways they connect with our system. Next, we break down the essential display and user interface buttons that make the device easy to use. We also examine modern weighing technologies to ensure accurate measurements and discuss the power supply and voltage regulation needed for dependable, on-the-go performance. On the software side, we review the PCB applications, general tools, and specific microcontroller software that bring all these elements together, while also exploring Android application technologies and frameworks that could allow for convenient, remote access. Overall, this chapter not only outlines the technical challenges in choosing the right components that we face but also shows how each component plays a crucial role in bringing PUMPED to life.

3.1.1 Microcontroller

3.1.1.1 Why use an MCU? Why not use an FPGA?

There are two types of components that can be at the center of our project, either an MCU or an FPGA. Both have different targets, and both have a lot of differences that pertain to distinct scenarios. A field Programmable Gate Array (FPGA) is an integrated circuit that is sold separately. It offers the ability to be reconfigured to meet the specific case that it is needed for. FPGA can be updated automatically wirelessly with an update after the FPGA has been sent out by the manufacturer. An FPGA contains many different key components such as a configurable logic block (CLB) which contains LUTs and flip-flops that allow it to perform different operations, IO blocks, Digital signal processing Blocks (DSP), BRAM, Transceivers, Low Latency, High parallel processing, and much more. The Microcontrollers Unit (MCU) on the other hand is a simple computer unit that runs one basic program simultaneously.

We are very familiar with MCU's because we have used them throughout our education at UCF in different classes. There are many benefits to using an MCU such as that they are simpler, they don't require updates, are constant in repeating instructions, they only run one specific code, which means that after the device has left the factory, the specific instruction that was set for it cannot be updated. There are a few simple components that an MCU has, these are, it has a CPU which runs the program counter, ALU, and many more things. When comparing both of the devices listed, there are going to be many advantages and disadvantages to each of these that dictated which device we used for our project.

When comparing one another, an MCU is far superior than an FPGA in the sense that it has a much lower cost, it uses less power to do its tasks, it is easier to develop with because it uses C or C++ which we are all familiar with since we learned throughout our bachelor's degree, it has built in peripherals such as UART, SPI, I2C, and it runs its program continuously. On the other hand, an FPGA has many benefits such as having parallel processing which allows the device to run multiple programs simultaneously, it has low latency and can be reprogrammed with an update once implemented. Both are very

different from one another, but for our project, we went with an MCU because it will be able to satisfy all the requirements that we have while keeping a lower power consumption and having a low-price tag.

Table 2 MCU vs FPGA

Feature	MCU	FPGA
Processing Type	Sequential execution (one task at a time)	Parallel processing (multiple tasks simultaneously)
Reconfigurability	Fixed after manufacturing; cannot be updated remotely	Can be reprogrammed and updated after deployment
Development Complexity	Easier to develop; uses C/C++	More complex; requires HDL (Verilog/VHDL)
Cost	Lower cost	Higher cost
Power Consumption	Lower power usage	Higher power consumption
Performance	Limited by CPU clock speed	High-speed, real-time performance with low latency
Peripherals	Built-in (UART, SPI, I2C, ADC, etc.)	Requires external interfaces for some peripherals
Ideal Use Cases	Simple embedded systems, control applications	High-speed data processing, signal processing, AI acceleration
Flexibility	Fixed function; cannot change logic after deployment	Can be modified post-deployment for different applications

3.1.1.2 MSP430FR6989

This microcontroller was introduced to us in Embedded Systems, we are very familiar with it. It consumes very little power, and the manufacturer is Texas Instrument, its purpose is to be implemented with low power applications. It has a 16 MHz frequency, 16-bit architecture, 16 channels, 4 SPI, 2 I2C's, runs UART, it has 128 KB of embedded FRAM. The reason as to why this MCU is listed here is because we are very familiar with it and know its capabilities. With that being said, it is very simple and although it is an MCU, it might not be the best choice for our project because it does not contain Built-in Wi-Fi or Bluetooth connectivity, and we need our MCU to do so.

3.1.1.3 Raspberry Pi Pico

Raspberry Pi Pico is one of the MCU that is from the Raspberry Pi lineup. It is the very first board in this lineup. Very low cost with a price less than \$10 ranging by seller. It can

easily be reprogrammed when connecting to a computer over a USB using a specific SDK with C/C++. It comes without header pins so those would need to be purchased separately. It is a low-cost, high performance MCU with flexible digital interfaces, and runs on the RP2040 MCU. It has a dual core ARM cortex processor that reaches clock speed up to 133MHz. It has 264KB of SRAM, 2MB of flash memory, 26 GPIO pins, low power sleep and dormant modes, clock and timer chip, temperature sensors, 8 programmable IO state machines. It runs many communication interfaces such as UART, SPI, I2C, and the power that it supports ranges from 1.8V to 5.5V. it does not unfortunately have built-in Wi-Fi connectivity. There is the Pico W version which does contain Wi-Fi connectivity but does not contain Bluetooth connectivity.

3.1.1.4 Raspberry Pi Zero 2 W

This Raspberry Pi is the smallest form factor Raspberry Pi on the market. It is a predecessor to the regular Pi Zero but with wireless connectivity. It is 40% faster than the original Raspberry Pi. It has many features, some of which include the BCM2835 processor 1GHz single core CPU, 512MB of LPDDR2 RAM, HDMI connectivity, USB card reader, it has 40 pin GPIO header, CSI camera interface, and built in wireless connectivity (2.4GHz IEEE 802.11b/g/n wireless LAN, Bluetooth 4.2, BLE, onboard antenna). Its input power is 5V DC and 2.5 A. Its form factor is very small, it only measures 65mm x 30mm. Its architecture is ARM 11, 32-bit. The voltage that it can handle is between 4.75V to 5.25V with its recommended voltage to be 5V. The voltage of this MCU can be a red flag because as we will mention in later topics, there are some components that will not have adequate operating voltages. The price for this MCU ranges between \$10 - \$15 depending on the seller. This MCU is a great option but there can be another that is better for the specific task that we want.

3.1.1.5 Arduino Nano

The Arduino Nano is like the original Arduino but has a much smaller footprint of just about 45mm x 18mm and only weighs 7 grams. It is compatible with the Arduino IDE which is very user friendly and there are many different tutorials that teach you how to use it. The microcontroller that is paired up with it is the ATmega328 which has 16 MHz clock speed, 2KB SRAM, 32KB Flash, and 1KB EEPROM. It has many built in pins, 13 of them are LED pins, 14 are IO pins, 8 Analog input pins, and 6 PWM pins. It can run many different communications such as UART, I2C, and SPI. The power voltage that it runs for its operating voltage is 5V, and its current is 20mA. The recommended input voltage is between 7-12V. It can run and support C/C++ languages which makes it ideal for Embedded Systems. Unfortunately, it does not contain wireless connectivity, which limits its capabilities for our project.

3.1.1.6 Arduino Uno R4 Wi-Fi

The Arduino Uno R4 Wi-fi is an upgraded version of the classic Arduino Uno that features many new and more powerful hardware. This board is designed around the 32-bit MCU RA4M1 and runs an ESP32 module for Wi-Fi and Bluetooth connectivity. It operates with a 5V voltage and has an LED matrix directly on the board which allows users to create visuals. It uses a USB-C programming port, contains many GPIO pins which are 14 digital IO pins, 6 Analog input pins, 1 DAC pin, 6 PWM pin and 2-3 external interrupts. It contains many forms of communication such as UART, I2C, SPI, and CAN. Its input voltage ranges from 6 –24V and the DC current per IO pins is 8mA. It has a pretty good clock speed of

48 MHz for the RA4M1 and the ESP32-S3-Mini-1-N8 runs up to 240 MHz clock speed for wireless connectivity. It is bigger than the previous MCU with a size of 68.85mm x 53.34mm. It has 256KB of flash memory, and 32KB of SRAM. It can run IoT. It is more expensive than the previous MCU's that have been listed before, it ranges in price close to \$20. It also runs on the C/C++ programming language. It seems to be that this MCU could be a great choice to be used for our project.

3.1.1.7 Arduino MKR Wi-Fi 1010

The Arduino MKR Wi-Fi 1010 is a very fast board that allows any user to set up a super easy IoT connection. It also allows people to create a Bluetooth low energy connection, making it ideal for projects that require wireless communication. It can be configured in the Arduino IDE which is very easy to use and with the C/C++ language. This board is paired up with the SAMD21 Cortex-M0 32-bit low power Arm MCU that runs at a 48 MHz clock speed, it contains 256 KB of flash memory and contains 32 KB of SRAM. The board's power supply must be 5V to operate it, which is not significantly large, and it is very well in our range of power. The circuit operating voltage is 3.3V, which is also a good sign because some of our components might operate within this voltage range. It has 13 PWM pins, 8 GPIO Digital I/O Pins, 7 Analog input pins that are ADC, and 1 Analog output DAC pin. This board has a good number of connections, which is a good sign. It has a built-in battery connection of 3.7V. It also includes many different extra features such as a Crypto Chip ATECC508 for secure authentication, a micro-USB port for user interface, LED indicators for Wi-Fi status, and the size of this board is 61.5mm x 25mm. This board seems to be a great choice for our project so far, the only negative aspect of this board is that it is on the expensive side costing \$38.60 on the Arduino website.

3.1.1.8 ESP32S 30-pin

The ESP32 ESP-Wroom-32 30 pin version is a very powerful MCU board developed by the Espressif Systems. This is a very powerful board in a small form factor that can do a lot of different and powerful things. The MCU that is powering this board is the ESP32-D0WDQ6 which is a dual-core Xtensa LX6 processor that has a max clock speed of 240 MHz, it is the fastest processor that has been mentioned from the previous ones. There are many different types of ESP32, the reason for that is because it is a very popular MCU board. Usually each contains 4MB of flash memory, but it varies by module. The board contains 529 KB of SRAM, 448 KB of ROM, and supports external storage support of SPI Flash and PSRAM.

This board has a very good built in wireless connectivity for Wi-Fi and Bluetooth 4.2. The ESP32 has many pins that vary between models, there are models with 30 pins, and others with 38 pins but they all do the same thing, and all have the same necessary pins. The ESP32 contains 25 GPIO pins, up to 18 ADC channels that are 12-bit, 2 8-bit DAC channels, 10 touch sensors, 4 SPI buses, 2 I2C buses, 3 UART ports, a CAN bus that can be supported via an external transceiver. The size of this specific board is very small, only measuring 25.5mm x 18mm. It has many additional features such as an integrated PCB Antenna, a metal shield for RF protection, and security. The power supply needing to power and run this board is close to and less than 3V, but the power to run the pins on the board is 3.3V, which is a great range of voltage because a lot of our other parts run at this voltage. Although this is the ESP32, it can run and being compatible with the Arduino IDE, which

is a detail to have because it is easy-to-use software. It also supports C/C++ and even Python programming languages.

There are many different price ranges that this board comes in through different sellers, but the price is not much higher than \$15. There are multiple packs that come with many ESP32, so if you were to divide the price by the number of ESP32 that you receive in the mail in that purchase, then the cost of each individual ESP32 will range in the price range lower than \$10. This board can be used for many different applications such as IoT, Embedded Systems, wireless networks, and many more. The ESP32 seems to be the MCU of choice for our use of application in our project, with being powerful enough to run our planned tasks, running at a fair voltage rate, having a lot of support and information on the web, and having a cheap price, this MCU might be the one of choice for our project.

3.1.1.9 Comparison Between All the MCU's Named

To summarize all the information listed for each specific MCU, there will be a table down below that serves its purpose to demonstrate each of the MCU's specs and their benefits and key features that can be useful. All the MCU's will be set up in comparison with one another, and the MCU of choice will be highlighted.

Table 3 MCU Comparisons

MCU	Architecture	Clock Speed	Memory (Flash/SRAM)	GPIO Pins	Communication Interfaces
MSP430FR6989	16-bit	16 MHz	128 KB FRAM	16	4 SPI, 2 I2C, UART
Raspberry Pi Pico	ARM Cortex-M0+ (Dual-core)	Up to 133 MHz	2MB Flash / 264KB SRAM	26	UART, SPI, I2C
Raspberry Pi Zero 2 W	ARM11	1 GHz	512MB LPDDR2 RAM	40	USB, CSI, HDMI
Arduino Nano	ATmega328 (8-bit)	16 MHz	32 KB Flash / 2 KB SRAM	14 digital, 8 analog	UART, I2C, SPI
Arduino Uno R4 Wi-fi	ARM Cortex-M4 + ESP32	48 MHz (RA4M1) / 240 MHz (ESP32)	256 KB Flash / 32 KB SRAM	14 digital, 6 analog	UART, I2C, SPI, CAN
Arduino MKR Wi-Fi 1010	SAMD21 Cortex-M0+	48 MHz	256 KB Flash / 32 KB SRAM	8 digital, 7 analog	UART, I2C, SPI
ESP32 (30-pin)	Xtensa LX6 (Dual core)	Up to 240 MHz	4MB Flash / 529 KB SRAM	25	4 SPI, 2 I2C, 3 UART, CAN

Table 4 MCU Comparisons Continued

MCU	Wireless Connectivity	Power Voltage	Size	Cost (\$)	Notes
-----	-----------------------	---------------	------	-----------	-------

MSP430FR69 89	No	1.8V - 3.6V	-	~ \$10-\$15	Low power, not ideal due to lack of Wi-Fi/Bluetooth
Raspberry Pi Pico	No (Pico W has Wi-Fi, no Bluetooth)	1.8V - 3.6V	51 mm x 21 mm	< \$10	Low-cost, flexible, lacks built-in wireless
Raspberry Pi Zero 2 W	Wi-Fi (2.4GHz), Bluetooth 4.2	4.75V - 5.25V	65 mm x 30 mm	\$ 10-\$15	Small, good wireless support but voltage concerns
Arduino Nano	No	5V (Op. Voltage), 7-12V (Input)	45 mm x 18 mm	< \$10	Small, widely supported, lacks wireless connectivity
Arduino Uno R4 Wi-fi	Wi-Fi, Bluetooth	6V - 24V	68.85 mm x 53.34 mm	~ \$20	IoT capable, good wireless support
Arduino MKR Wi-Fi 1010	Wi-Fi, Bluetooth LE	5V (Power), 3.3V (Circuit)	61.5 mm x 25 mm	~ \$38.60	Great for IoT, expensive
ESP32S(30-pin)	Wi-Fi, Bluetooth 4.2	3.3V	25.5mm x 18 mm	< \$15	Powerful, cheap, great for IoT

3.1.1.10 MCU of Choice

The ESP32S 30 pin is the ideal MCU for this project due to its powerful dual-core Xtensa LX6 processor, which can reach clock speeds of up to 240 MHz, providing ample processing power for simple and demanding tasks that we can possibly come across with. It features built-in Wi-Fi and Bluetooth 4.2 connectivity, making it perfect for IoT applications, and since we are planning to create a device that pairs with a mobile app wirelessly, we need an adequate MCU capable of doing such tasks. With 25 GPIO pins, 18 ADC channels, and support for various communication protocols such as SPI, I2C, UART, and CAN, the ESP32 offers excellent flexibility and scalability for interfacing with various sensors and peripherals. Its small form factor (25.5mm x 18mm) and low power consumption (operating at 3.3V) make it highly suitable for embedded systems in power-sensitive environments and can be included in our small planned portable form factor device. Additionally, its compatibility with popular development platforms like the Arduino IDE and support for C/C++ and Python programming languages make it easy to develop and deploy. With its affordability, size, and powerful features, the ESP32 stands out as the best choice for the project's requirements.

3.1.2 Printers

There are three main kinds of paper printers that can be bought off the shelf: inkjet, laser, and thermal printers. Each comes with its own set of pros and cons for a project such as this one. The relevant factors we take into consideration when deciding on a printer are efficiency, reliability, power consumption, size/portability of printer, size of paper printed, print quality, cost of the printer, and cost of the printer materials/cartridges. In addition to the kinds of paper printers there are several ways to connect and interact with them such as Bluetooth, Wi-Fi, or TTL communication. The main areas of concern for the comparison of are printers are:

- Print speed
- Reliability

- Power consumption

3.1.2.1 Inkjet Printers

Inkjet printers, as suggested by their name, use liquid ink in the printhead to print images or text onto paper. Inkjet printers are ideal for printing images or graphics due to their ability to deliver very high-resolution print results. Inkjet printers are also particularly notable for their ability to print things in color rather than black and white. Inkjet printers present potential for use in a built-in device due to their smaller size and overall portability. This also means that you can print things in a variety of sizes depending on the inkjet paper you acquire.

A notable disadvantage though that affects the efficiency and reliability of inkjet printers is the potential for the nozzles in the printhead to get clogged. This means that time needs to be taken to perform maintenance on the printer or risk the printer breaking which of course is a net negative on the overall efficiency and reliability of a printer. The other primary issue with inkjet printers is that they are not a one-time cost. The use of an inkjet printer means you have to continuously buy ink cartridges which presents a significant cost over time if you are doing a large amount of printing.

3.1.2.2 Laser Printers

Laser printers, as suggested by their name, most commonly use varying types of semiconductor lasers in order to print onto paper. The biggest advantage for laser printers with regards to efficiency and reliability is that they have the fastest print speeds of any printer type. This makes laser printers ideal for a device or process that may be engaging in high volume printing. Laser printers, in contrast to inkjet printers, are typically best applied to cases where you are primarily printing black and white text in order to take full advantage of its high efficiency printing. In addition to efficiency laser printers tend to produce the highest print quality for these types of print jobs due to the fact there is no risk of smudging ink like there is for an inkjet printer.

Laser printers do begin to present concerns in the categories of printer size/portability, cost of the printer, and cost of additional materials. Laser printers on average are larger than inkjet printers so it may be more difficult to find a suitable one that is of an appropriate size for use in a built-in device. Another downside of laser printers is that the price of the printers themselves tend to be the most expensive of any printer type. In addition to this already large upfront cost laser printers require you to keep buying toner in order to continue printing, which presents a significant cost if you are engaged in a high volume printing operation. Another concern with regards to power consumption is that laser printers use significantly more electricity and power than other types of printers. In addition to the power consumption problem itself this also means that laser printers also have the potential to create issues with their heat generation if you want to integrate them into an all-in-one device.

3.1.2.3 Thermal Printers

Thermal printers, as indicated by their name, use a heated printhead to create images onto a specific kind of thermal paper. Thermal printers are known to be very reliable compared to other types of printers and the print speed is relatively efficient. The efficiency of thermal printers is further enhanced by the ease of replenishing the printers with thermal paper

compared to laser or inkjet printers. This makes them ideal for use by consumers interacting with a printer integrated into an integrated device due to the ease of use. Thermal printers are also known for their relatively low power consumption compared to say a laser printer which uses significantly more power. Perhaps the biggest advantage of thermal printers is in their size and portability if you are trying to integrate them into a single device. Thermal printers relative to other types of printers can be made extremely small and portable without a degradation of print quality. Thermal printers of this size have an ideal use case for receipts and labels, making them perhaps the best type of printer for our particular needs.

The print quality of thermal printers is good although some concerns are presented with regards to the durability of the print quality over time with potential fading. As mentioned previously these thermal printers use a specific kind of thermal paper that is generally cheaper than laser toner or inkjet cartridges. The aforementioned reliability, portability, and low cost of thermal printers in combination with its excellent print quality make it an ideal choice for use in an integrated, all-in-one device whose primary use case is going to be printing small labels.

Table 5 Printer Types and Their Capabilities

Printer Types	Efficiency	Reliability	Power consumption	Size/portability	Print quality *	Cost of printer	Printer material cost
Inkjet Printer	5 - 15 pages per min.	5 years	30-50W (printing)	Small	1200-5000 DPI	\$50 - \$300	\$0.05 - \$0.30 per page
Laser Printer	20 - 40 pages per min.	5-10 years	300-500W (printing)	Large	600 – 1200 DPI	\$150 - \$1000	\$0.01 - \$0.05 per page
Thermal Printer	20-50 pages per min	5 years	5-20W (printing)	Very small	200 – 300 DPI	\$40 - 100	\$0.01 - \$0.05 per page

3.1.3 Printer Interfacing Protocols

Every printer has unique ways to interface with it in order to send print jobs. Some printers are using wired technologies such as transistor-transistor logic (TTL), universal serial bus (USB), or ethernet. Other printers use wireless technologies such as Bluetooth or Wi-Fi. These protocols are all generally evaluated on the basis of reliability, ease of use, range, data transfer speed, and security. The MCU to be used for this project will partially dictate which of these interfacing protocols will be compatible but generally speaking our selected MCU should be able to utilize most, if not all, of the aforementioned technologies. The areas of concern for our printer interfacing protocols are:

- Power consumption
- Wired vs Wireless
- Data transmission speed

3.1.3.1 Bluetooth

Bluetooth technology enables the exchange of data over a short range in a wireless fashion. Printers that have Bluetooth capabilities, the ability to receive a print job via Bluetooth, are very common. This means there will be no shortage of options in terms of finding a printer that fits the specific needs we have to integrate it into our device. In terms of integration a Bluetooth printer offers some convenience since there is no need for physical cables. Another positive of a Bluetooth printer is that it is a low power consumption technology which fits our needs if it is operating on an MCU. Overall MCUs that have Bluetooth technology are known to be generally reliable. Concerns with Bluetooth arise though when we observe how exactly we are pairing the connection of an MCU with a printer via Bluetooth. This pairing would require some kind of printer control software that if written from scratch could be particularly complex and time-consuming to create.

The alternative would be the use of an open-source library but then this severely limits our choice of printers to strictly ones supported by these hypothetical libraries. Another pair of issues with Bluetooth is its security and data exchange speeds. If we have a wireless connection such as Bluetooth we expose ourselves to security concerns with unauthorized connections by bad actors, denial of service attacks, or any kind of malware sent using the protocol. In terms of data exchange speeds almost all wireless protocols are going to be slower at exchanging data in comparison to their wired counterparts.

3.1.3.2 Wi-Fi

Wi-Fi technology enables the wireless exchange of data via the use of radio waves. Printers that are Wi-Fi enabled are very common which means we should have less difficulty in terms of finding a Wi-Fi printer that fits our specific needs. Integrating a Wi-Fi enabled printer should present a higher degree of convenience since there is no need for physical cables. MCUs that have Wi-Fi technologies are very reliable and an extra degree of reliability arises from the fact that portability is increased due to the device working over an entire Wi-Fi network. Concerns with Wi-Fi technology arise though when we consider how exactly the connection between the MCU and printer is established via Wi-Fi. The process would be somewhat complicated with having to write code that adheres to a specific communication protocol (MQTT, TCP, HTTP, etc.) and also format the print data sent to the printer according to the printer's specific requirements. The process is different compared to Bluetooth due to only having to connect to the same Wi-Fi network and not having to arrange a connection specifically between two devices but it still presents a certain degree of additional complexity.

3.1.3.3 Ethernet

Ethernet printers, while less common than other types of printers, are readily available commercially and present unique advantages compared to their wireless counterparts. Ethernet enabled printers can be more reliable due to the wired connection being more stable compared to wireless methods. Additionally the wired connection means the data exchange is of a higher speed. If we are engaged in high volume printing these two aforementioned factors regarding reliability and efficiency would make an ethernet printer

significantly better than wireless printers. The chief issue with using an ethernet printer despite these advantages is the lack of native support among many MCUs for ethernet. While there are external modules we could use it would ultimately be complex and time consuming while also possibly degrading the reliability of the ethernet printers that attracted us in the first place. The lack of support for ethernet among many of our preferred MCUs make it difficult to justify its use.

3.1.3.4 TTL

TTL UART printers are certainly not as common commercially as other wireless or wired printers but they present several unique advantages for use in a project such as this. Much like Ethernet the biggest advantages come from the wired nature of these printers. The wired connections mean we have a higher degree of reliability due to not having to worry about interference or loss of connection like we have to with other wireless technologies. There is an increase in efficiency as well that comes with the unmatched fast data transfers of a physically wired printer via a wireless printer. This high degree of reliability and efficiency is very important if we are engaged in high volume printing. TTL UART printers are also ideal because there is significant native support among a wide range of MCUs with most MCUs typically having multiple UART ports and the other few connections needed for this type of printer to work. The wiring for a TTL UART printer should also be straightforward with only 3 connections: TX, RX, and GND.

The low power consumption of this communication protocol compared to a wireless technology such as Wi-Fi also makes the use of this method very attractive. The primary concern with the use of TTL comes from ensuring the TTL compatibility between printers and the selected MCU. Many TTL printers operate at a 5V TTL level while many MCUs operate at a 3.3V TTL level. This can be addressed via the use of a level shifter in order to ensure compatibility between the printer and the MCU. Aside from ensuring voltage compatibility there is a significant upside to using a TTL printer with limited concerns. Due to the reliability, efficiency, and simplicity of a wired setup compared to a wireless one the TTL printer is ideal for this integrated device.

Table 6 Printer interfacing Protocols and Their Capabilities

Printer Type	Reliability	Range	Data transfer speed	Power consumption	Wired
Bluetooth	Moderate	10 meters	1 – 3 Mbps	~2.5 W	No
Wi-Fi	Moderate	30 - 100 meters	50 – 600 Mbps	~5 W	No
Ethernet	High	100 meters	100 – 1000 Mbps	~2.5 W	Yes
TTL	High	<1 meter	115200 bps	~1 W	Yes

3.1.4 Display

3.1.4.1 Different Types of displays

When choosing the right display for an ESP32S 30-pin microcontroller project, it's important to consider both compatibility and functionality. Since the goal is to display date, time, weight, and battery percentage, a screen size of around 1-3 inches offers a great balance between readability and device compactness. There are several display types to choose from, including TFT LCDs, OLEDs, and e-paper screens, all with unique benefits. TFT LCDs provide vibrant colors and fast refresh rates, ideal for dynamic data updates. OLEDs offer excellent contrast and wide viewing angles, displaying clear visibility even in low-light environments. E-paper displays, while slower to refresh, are very power-efficient and can hold an image without consuming power, making them perfect for low power projects. To ensure compatibility with the ESP32S 30-pin MCU, the display should support either SPI or I2C communication, both of which are well suited for efficient data transfer. Additionally, the display should operate at 3.3V, as the ESP32's GPIOs are not 5V capable. If a 5V display is used, a logic level shifter will be required to prevent damage to the microcontroller. Power-wise, many small OLED and e-paper displays can run directly from the ESP32's 3.3V supply, while larger TFT LCDs may require a separate 5V power source. Below are some examples of displays that can possibly be used for our project.

3.1.4.2 Honyond ILI9341 2.8-inch Display

The TFT LCD Touchscreen display contains the ILI9341 driver and has a resolution of 320 x 240 pixels. It has a supported voltage of 3.3V which is compatible with the ESP32 pin input voltage making it ideal for pairing with one another. This display supports the SPI interface which is an ideal high speed synchronous serial communication protocol used to transfer data between MCU and peripherals. There are many pros to this display, one of which is the size. The size of this display is 2.8 inches, but in the product dimensions the overall size of this display is 3.39 x 1.7 x 3.28 inches. The color depth of this display is 265k/65k, has a white LED backlight. One more benefit to this display is that it is also compatible with a 5V logic pin, both of which the ESP32 is able to support with level shifting. A few more benefits to this display are that it has a fast refresh rate, and good library support, which is beneficial to us if we end up using this display in our PUMPED project. One major disadvantage of using this display for our project is that it can require a higher consumption of power than other displays. Since we are making a portable device, we must make sure that there is enough power to support all our devices for a certain amount of time. One secret feature of the display is that it is also a touchscreen panel, it is definitely nice to have but we are not going to be using the touchscreen feature for our PUMPED project, so it is neither pro or con, and we can disregard this feature.

3.1.4.3 GoDiyModules ST7789 1.6-inch Display

This is another TFT display. This display contains the ST7789 chipset driver and has a resolution of 240 x 280 pixels. It has a supported voltage of 3.3V which is ideal when pairing with the ESP32 MCU. It is an IPS full panel display that has a 178-degree viewing angle range and measures 1.69 inches, one cool feature about this display is that it has rounded edges giving it a bit of character. This display also supports the SPI interface which again is ideal for high-speed synchronous communication used to transfer data between the MCU and peripherals. The dimensions of this display are 3.94 x 1.97 x 0.39 inches, which is small but can work for our PUMPED project. It is definitely a little bit smaller than our anticipated size display, but it can be an option to be used if we prefer to use a TFT display.

This display contains 8 pins, each of which are illustrated as being: pin1 = GND, pin2 = VCC, pin3 = SCL, pin4 = SDA, pin5 = RES, pin6 = DC, pin7 = CS, and pin8 = BLK. This is a pretty straightforward display that contains the basics and the necessary features that we can use for our PUMPED project, that size is on the smaller size, but it can accommodate displaying the necessary information that we want our device to illustrate to consumers. One of the major cons for this display is that it contains a higher current draw, so we would need to keep an eye out on that because we are making a portable device.

3.1.4.4 HiLetgo SSD1306 1.3-inch Display

This display is different from the previous 2 that have been listed. This is an OLED display that contains the SSH1106 chipset and has a resolution of 128 x 64 pixels. It has a supported voltage of 3.3V which is great because it works well with the ESP32, and we would not need an adapter. The display supports I2C communication which is a multi-serial communication protocol used to connect low – speed peripheral to MCUs heavily used in embedded systems. The screen size of this peripheral is 1.3 inches which is fairly small compared to the previous displays that we have talked about. It has a special feature that is low light.

This display is compatible with many different MCU, and it has a large library selection that can be used. A big benefit to this display is that on the website where it was found, the schematic of the display is attached and shows all the pins and configurations that it contains. A few pros to this display are that it supports low power consumption, high image contrast, and wide viewing angles. Some cons to this display are that it is monochrome only, and that it is a small display which can limit data visibility. Overall, this display is a decent peripheral that can possibly be used in our PUMPED project, it has very low power consumption, but it is small.

3.1.4.5 HiLetgo SSD1309 2.42 - Inch Display

Another OLED Display module which contains the SSD1309 Driver. This display is a little different from the other ones because you can purchase it with 2 variations. You have the option to purchase it with an I2C serial communication which we are all familiar with in our group because we used this communication method in our embedded systems class, and you can also purchase this display with an SPI serial communication. It all comes down to preference. The size of this display is 2.42 inches, which is very good in size and could possibly be the best size that we want for our PUMPED project. It has a 128 x 64 pixels resolution size. It can support different voltages ranging from 3V – 5V which is perfect since our ESP32 supports 3.3V and that falls in between the threshold voltage. This display is also good for low power consumption tasks which we want it to be because we are making a portable device. The I2C version of this display contains a 4-pin configuration which are GND, VDD, SCL, and SDA, the SPI version of this display contains a 7-pin configuration which are GND, VCC, SCK, SDA, RES, DC, and CS. The display is monochromatic, it doesn't require backlighting, has wide viewing angles, and overall, a great display to use. This display seems to be a great option for our PUMPED project.

3.1.4.6 Waveshare 4.2 Inch E – Paper Ink Display

This is an E – Paper Ink display that is very different from the other two types of displays that have been talked about previously. It is a very interesting display that I have never worked with before, so it is new to me. The display has a size of 4.2 inches which is by far

the biggest one so far. It has pretty high resolution from the previous ones which comes at 400 x 300 pixels. The color contrasts of this display are only in black and white but has 4 gray scale colors. It has the 3-wire and 4-wire SPI interface protocol which is very good and is capable of pairing with the ESP32. It has very low power consumption, a key feature is that the last thing that was displayed can stay on the screen even after powering off the device.

It has wide viewed angles and no backlighting. One of the main key features of this display is that it is compatible with both 3.3V and 5V MCUs. It comes with 7 pins, GND, DIN, CLK, CS, DC, RST, and BUSY. The product dimensions are 7.09 x 3.94 x 3.54 mm, it is a large display that might be a little bit too big for our PUMPED project and might have a lot of negative space, which we can possibly fill up with an image that corresponds to the project's usage. Some of the disadvantages of this display are that it has a very slow refresh rate, requires full refresh to prevent ghosting, and limited colors. Since we are going to be displaying measurements that are going to be updated frequently, this might not be the best choice for our PUMPED project.

3.1.4.7 Waveshare 2.13 Inch E – Paper Ink Display

This is another E – Paper Ink Display that is very similar to the previous display, the main difference is its size and resolution. This display is a small display that comes in at 2.13 inches in length. The resolution of this display is 250 x 122 pixels, and the dimensions of the peripheral are 65 x 30.2 x 0.194 mm. This display has a 3-wire and 4 -wire SPI communication protocol like the previous one, it has a paper like feel, displays only black and white colors, and has a 2 grey scale. Since it has a paper-like feel to it, it contains a high contrast imaging capable of being easily readable in harsh sunlight. One key feature that this type of display has is that it has image retention without power, which means that no power is needed to maintain the image content.

This display operates at a 3.3V input, which is ideal when pairing it up with the ESP32, but you can pair it up with 5V systems because the display has an onboard voltage translator. The power consumption is very low, and it contains all of the necessary pins that will allow you to pair with the ESP32. Some of the disadvantages of this display is that it has a slow refresh rate which is not ideal because our device will need a decently fast refresh rate, it has ghosting if partial refresh is not done properly and has limited color options.

3.1.4.8 Display of Choice

When choosing the right display out of all the above options that were listed, we want a display that can check off all the boxes that we need to accurately show all our features that we want out of our PUMPED project. We also want the users to be able to successfully navigate the information with ease. We are looking for a good size display that is not so big, not so small, has a good refresh rate, displays information very clearly, and is capable of pairing up with the ESP32 easily and efficiently. Below is a table comparing all the features of each of the displays that were listed as possible options to use:

Table 7 Display Comparisons

Display	Type	Size	Resolution (pixels)	Voltage	Communication
---------	------	------	---------------------	---------	---------------

ILI9341	TFT LCD	2.8 Inches	320 x 240	3.3V (supports 5V with level shifting)	SPI
ST7789	TFT LCD	1.69 Inches	240 x 280	3.3V	SPI
HiLetgo SSD1306	OLED	1.3 Inches	128 x 64	3.3V	I2C
HiLetgo SSD1309	OLED	2.42 Inches	128 x 64	3V - 5V	I2C or SPI (Both Options)
Waveshare Ink Display	E-Paper	4.2 Inches	400 x 300	3.3V (also supports 5V)	SPI
Waveshare Ink Display	E-Paper	2.13 Inches	250 x 122	3.3V (onboard translator for 5V)	SPI

After comparing each of the displays and taking a look at the pros and cons, the most suitable display to use for our PUMPED project is going to be the HiLetgo 2.42-inch SSD1309 OLED display. The reason for this is because it offers a balanced screen size that's easy to read, you are able to get a different version whether you want to use I2C or SPI communication protocol, it offers wide viewing angles, it is monochromatic which doesn't require any backlight (it is not an issue since we want it to be used for basic tasks), it has a low power consumption which allows for our battery to last longer, flexible voltage options that work well with the ESP32S, and the refresh rate although not as fast as the TFT displays, is perfect for what we need it to do.

3.1.5 User Interface Buttons

3.1.5.1 Why use physical Buttons

There are many ways in which a consumer can utilize different technologies, there are touchscreen displays where someone can change something directly from the screen, there are voice commands that someone can use to make a device do a certain thing, and there are also old school physical buttons that each are used for different tasks. In the case of our PUMPED project, we have decided to go with the physical buttons. The reason for this is because of the targeted consumer that we are trying to help. Our PUMPED project device will be directly connected to recent mothers that are in need of an easy-to-use portable device that will allow them to weigh and label their breastmilk bags.

We want to have physical buttons because mothers do breastfeed at night, and can be sleepy at times, so instead of accidentally pressing the wrong option on a touchscreen, we want them to be able to physically feel the different buttons which will have labels on them illustrating what each of them are going to be used for. The main key points that we want to target in our device's user interface is for it to have reliable tactile feedback which is essential for users who will operate the device while tired or in low-light conditions, simple and intuitive operation which will reduce the chances of errors when doing the tasks of weighing and labeling breastmilk, and have a good performance no matter the conditions which can interfere if we ended up going with a touchscreen.

3.1.5.2 Button configuration

For our button configuration, we are trying to make it minimalist, simple to use configuration. Since the device will be paired with a mobile application, we decided to keep the number of buttons to < 8 buttons. Each button will have a specific task that they will do, and that task is not going to change, it will always stay the same. Button 1 is going to allow us to open the printer cartridge so that we can refill the printer paper when it is needed to, this button is going to be alone separated from the rest. The next 3 buttons are going to be together arranged into a soft triangle on the front left side of the device. Button 2 will allow us to reset the device in case of an issue with the software or hardware, this will be very similar to dev boards that contain a reset switch.

Button 3 will allow us to establish a wireless connection between the device and the phone that's being paired with. Button 4 will give us the choice of changing the date and time that is shown on the OLED display. The next 3 buttons are going to be together also in a triangle shaped on the front right side of the device. Button 5 will allow us to move right when changing time and date. Button 6 will allow us to move left when changing time and date. Button 7 will let us increase the time and date number and will reset back to zero when reaching a specific number. Lastly, Button 8 is going to be placed in the middle between the other 6 buttons. This button is going to be much larger than the rest of the buttons, and it will allow the user to print the label after weighing the breastmilk bag. All buttons have a purpose and will work properly.

3.1.6 Weighing Technologies

One of the important pieces of our PUMPED design is the scale that will weigh bags of breast milk and send that information to the printer. The scale will need to be fast in calculating the weight and provide sufficient accuracy so that the user can have confidence in the exact amount of milk in their inventory. To accomplish these tasks, we have found three technologies that could possibly make it into the final product

3.1.6.1 Strain gauge load cells

A strain gauge load cell is an instrument that measures weight or force by measuring the resistance changes in a wire compared with other resistances connected to a surface. The surface is called a load cell and when acted upon by a force is deformed enough to cause the resistance change. Functioning as sensors, strain gauges mounted to a surface are connected in such a way that creates a Wheatstone bridge and when acted upon by a force creates a new electrical signal. The combination of the load cell and the strain gauges creates a transducer that converts force into an electrical signal. The electrical signal created can be interpreted in a way to give a numerical value to the force or weight applied. Strain gauge load cells provide high accuracy due to their ability to detect small changes in force. They are also reliable because their performance over time shows little degradation. Strain gauge load cells have applications in many different industries, such as medical, automotive and aerospace. Strain gauge load cells are budget friendly. Lots of support due to their wide use.

3.1.6.2 Capacitive load cells

A capacitive load cell uses capacitors to measure weight or force across a load cell. In the capacitor there are two charged metal plates that contain an uncharged dielectric material

between them. When a load is placed on the capacitor the two plates are pressed together creating an electric current that builds up a charge in the system. The charge created is stored in the dielectric and is calculated as voltage that can be read as weight. Capacitive load cells don't come in contact with what they're measuring so they can last longer. Capacitive load cells are more accurate than their strain gauge counterpart. Compact and light weight. Resistant to environmental factors.

3.1.6.3 Piezoelectric sensor

These types of sensors are made of piezoelectric materials that when put under pressure create an electric charge. The weight of an object can be determined because the amount of pressure that is applied to the material is proportional to the electric charge that is created. A piezoelectric sensor is typically made up of quartz crystals that generate charge when stressed and zero charge when not stressed

Table 8 Weight Sensor Technology Comparisons

Features	Strain gauge Load Cell	Capacitor load cell	Piezoelectric sensor
Accuracy	Sufficient	Sufficient	Sufficient
Uses	Static weight measurement	Static weight measurement	Dynamic force measurement
Complexity	signal conditioning required	Signal processing required	charge leakage
Cost	~\$8.00	~\$100.00	~\$100.00
Power Consumption	Negligible	Negligible	Negligible
Stability Over Time	Very stable	Moderately stable	Poor (due to voltage decay)
Temperature Sensitivity	Affected	Affected	Affected

3.1.6.4 Weighing Technology selection

From the technologies mentioned our group has decided to use the Strain Gauge Load Cell due to its ease of use, affordability, accuracy, reliability, and low power consumption. This device can be integrated into our design because of its compatibility with the ESP32.

3.1.6.5 Strain gauge load cell part analysis

The strain gauge load cell that we decided to go with is the WWZMDiB 1KG Load Cell. This load cell can be used to measure force, pressure, displacement, strain, torque, and acceleration. Our focus will be its force functionality. We have found that the average capacity of breast milk storage bags is in the range of 5 to 9 ounces, or approximately 141.75 to 255.15 grams, so the 1kg max capacity is well within the scope of how it will be used. The recommended voltage to power this cell is in the range from 2.6 to 5.5 volts. This particular load cell will require an Analog to Digital Converter to facilitate communication from the cell to our microcontroller of choice. Further discussion of the ADC will be mentioned below. This load cell has 4 strain gauges bonded to its surface to monitor the deformation of the cell. An in-depth discussion will be mentioned below.

3.1.6.6 ADC

The analog-to-digital converter is a device that converts an analog signal into a digital signal. An analog signal is described as continuous in time and amplitude while a discrete signal is discrete in time and amplitude. The process of this conversion can result in an error that loses output data, so resolution is a key aspect to the ADC. The resolution of an ADC is the number of different discrete values that can be produced from analog inputs in an instance of time. This essentially means that the greater the resolution the better the analog signal has been translated into a digital signal. Now we will discuss three different ADC's that we could possibly use with our load cell selection.

3.1.6.7 ADS1256

The ADS1256 is a high precision, low noise 24-bit ADC. This ADC provides high resolution for demanding measurement applications. It contains a delta signal modulator with a programmable digital filter, with support for different types of inputs. This device has a programmable gain amplifier (PGA) which offers gain from 1 to 64, while a selectable input buffer increases impedance. It has sampling rate up to 30k SPS, which can allow for fast channel cycling for measuring multiplexed inputs and one-shot conversions. It utilizes Serial Peripheral Interface communication that can operate with a two-wire connection.

3.1.6.8 HX711

The HX711 is a 24-bit precision ADC designed for weigh scales and industrial control applications, allowing direct interfacing with a bridge sensor. It features a low-noise programmable gain amplifier (PGA) with two differential input channels: Channel A supports gains of 128 or 64, allowing full-scale input voltages of $\pm 20\text{mV}$ or $\pm 40\text{mV}$ (with a 5V supply). While Channel B has a fixed gain of 32. An on-chip voltage regulator eliminates the need for an external power supply for the ADC and sensor. The clock source is flexible and can come from an external clock, crystal, or built-in oscillator, which requires no extra components. A power-on reset (POR) circuit simplifies digital

initialization. There is no programming needed for the internal registers. All controls to the HX711 are handled through the hardware pins.

3.1.6.9 ESP32 built in ADC

The ESP32 features 12-bit SAR ADCs capable of measuring voltage on up to 18 analog-enabled pins. It includes a ULP (Ultra-Low-Power) coprocessor that can measure voltage while in sleep mode, enabling low power consumption. The ADC can wake the CPU based on threshold settings or other triggers. The sampling rate is provided by two controllers: The RTC controller with a max rate of 200 Ksps and the DIG controller with a max rate of 2 Msps.

Table 9 ADC comparison table

Features	ADS1256	HX711	ESP32 built in ADC
Resolution	24-bit	24-bit	12-bit
Ease of Use	Further programming required	Works with simple implementation	Further programing required
Best Used For	High-speed applications	Load Cells	Embedded applications
Cost	~\$11.00	~\$1.00	Tied to the esp32 price

3.1.6.10 Our ADC selection

Our selection for an ADC was an easy decision. The HX711 is specifically designed for the load cell we decided upon. Along with its cheap cost and high resolution it's the perfect choice to get an accurate weight reading from the strain gauge load cell. Also, there is plenty of support available online that helps with interfacing this ADC with our microcontroller.

3.1.6.11 Strain Gauge

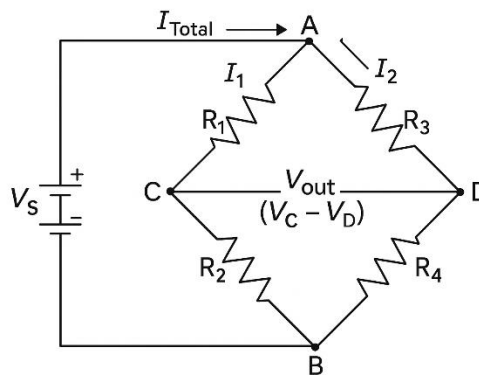
A strain gauge is used to measure the strain of an object it is bonded to. When the gauge is powered and strain is applied the electrical resistance of the material, usually called a conductive conductor, will undergo a change. The electrical conductor is a strip that contains parallel lines that can be stretched which in turn will increase the resistance of the

material. To make use of the change of resistance a Wheatstone bridge is used to measure these small changes.

3.1.6.12 Wheatstone Bridge

A Wheatstone Bridge is a type of circuit that is used to measure unknown resistance values. The circuit is typically shaped like a diamond with the voltage source applied on to opposite ends of the diamond and the voltage output measured across the other two opposite ends. The bridge contains four resistors that can have known or unknown values depending on the type of bridge being used. The simplest design for a Wheatstone Bridge would contain one variable resistor value and 3 known resistor values. The types of bridges are as follows: The Quarter bridge, as previously mentioned, has one variable resistor and three known resistance values. This configuration is the simplest but also least sensitive to strain and the least accurate due to only one variable change. The logic behind why that is will be mentioned below. The second type of Wheatstone Bridge is called the half bridge. The half bridge contains two variable resistors and two known static resistor values. This bridge is a step above the quarter bridge. The last bridge type that has the best sensitivity and accuracy of the three types is the full bridge. The full bridge has four variable resistance values that provide the best performance when providing an output voltage. To go more in depth into how the Wheatstone bridge produces an output voltage an image of a bridge is pictured below:

Table 10 Wheat Stone Bridge - Generated by ChatGPT



To produce an output voltage Wheatstone bridge forms a ratio of resistances in its branches.

When the ratio of resistances at the output nodes (C and D) are equal, the Voltage output will be zero. V-out equaling zero means the bridge is balanced. A balanced bridge is the normal state of the bridge before any change is made. The voltages at C and D are calculated by voltage divider rules, where $V_c = (R_2 / (R_1 + R_2)) * V_{in}$, V_d is calculated in a similar fashion. Subtracting V_c and V_d will provide V-out. Now let's say one of the resistors is a variable resistor, if the bridge is calibrated to be zero in a normal state with the variable resistor and then that resistor is manipulated to unbalance the bridge V-out will result in a non-zero value.

The output voltage can be measured to calculate the change. In the context of this project the variable resistors are the strain gauges, and when the gauges are manipulated, their

resistances will change and unbalance the bridge. So, since the load cell we chose has four strain gauges connected in such a fashion to create a full bridge circuit, the sensitivity of the cell is greatly increased since there are four variable resistance values that will greatly increase the value of V-out. A greater output voltage allows for less noise to be picked up when the voltage signal is being converted to a digital value.

3.1.7 Power Supply

When it comes to powering the system, one of the top priorities is to make sure that it meets the devices requirement of being easily portable for mothers to carry around with them. The PUMPED device is meant to facilitate the mother's ability to label their bags whenever, or wherever they are. For these reasons, we have decided to use rechargeable batteries to power the system. The use of rechargeable batteries will ensure portability, convenience, and it will reduce long term costs compared to using single-use batteries. This power system will need to supply energy to multiple subsystems including the ESP32 MCU, the load cell (scale), an OLED display, and the thermal printer. Each of these components have different voltage and current requirements, which means that proper power distribution is essential for a functional design.

3.1.7.1 Expected Power Demands

Each subsystem in the PUMPED device calls for different amounts of power. Making sure that we understand these is essential for our power supply design because we need to make sure that the need of each component is met. These various power requirements must be taken into consideration to ensure proper power distribution. Below is a breakdown of each subsystem in terms of the power they require for optimal efficiency and functionality.

3.1.7.1.1 Power for ESP32 Microcontroller (MCU)

The ESP32 Microcontroller is designed for embedded systems and uses a relatively low amount of power. In the case of our PUMPED scale, it will be the core component managing all the information received from the user via the scale, application, or the devices user interface, it will be responsible for processing data received from peripherals, and it will be sending data to the thermal printer. The microcontroller is essential for the device's functionality and will require a stable power input to function properly.

Table 11 MCU Power Consumption

ESP32 Power Consumption by Mode		
Mode	Minimum	Maximum
Active Mode	95 mA	240 mA
Modem Sleep Mode @2MHz	2 mA	4 mA
Modem Sleep Mode @80MHz	20 mA	25 mA
Modem Sleep Mode @240MHz	30 mA	50 mA
Light Sleep Mode	0.8 mA	0.8mA
Deep Sleep Mode	10 μ A	150 μ A

Hibernation Mode	5 μ A	5 μ A
Power Off	0.1 μ A	0.1 μ A

The operating voltage of this microcontroller is 3.3 volts. To assure that we have a steady source of power coming from the battery to the MCU, we are using voltage regulators. Now, as far as power consumption goes, since the amount of power consumed by the ESP32 heavily relies on which mode it is on, we only want the MCU to be in active mode for the time that the user will be weighing the bag and printing the label. Implementing an idle case for the device where the MCU can enter one of its sleep modes will allow for an extended battery life. Transitioning between these two states efficiently will require a balance between performance and saving power.

3.1.7.1.2 Power for Load Cell (Scale) & HX711 Analog-to-Digital Converter (ADC)

The load sensor we used will be responsible for measuring the weight of the breast milk bags and the HX711 ADC will be responsible for converting the signal transmitted from the load cell from a small voltage into digital data which will then be sent to the ESP32 for processing.

The 1kg load cell sensor that we are using for the PUMPED device uses a half Wheatstone bridge circuit that is made of 2 fixed resistors and 2 variable resistors. The resistance of the variable resistors is changed depending on the load that is applied to the load cell, and in turn, it produces a differential voltage output between 2mV and 100mV. According to the device's datasheet, the sensor requires an input voltage anywhere between 5V and 10V to function. Now, the datasheet did not specify the current that the sensor will draw, so I cannot calculate the power consumption for the load cell. However, the output voltage is relatively small, and it will serve as the data input into our ADC converter, so, based on this and the fact that they did not add the operating current to the datasheet we are assuming that the power consumption for the load cell is negligible.

The HX711 is a precision 24-bit analog-to-digital converter (ADC) that is specifically designed to be used for weigh scales. This ADC is packaged with a low-noise programmable gain amplifier (PGA) that gives us the option to boost the signal provided by the load cell by 32, 64, or 128 before it gets converted to a digital signal. The gain we use will be determined by the signal we receive from the load cell. The ADC does not need much power to function, as it draws less than 1.5 mA while in use, and less than 1 μ A while powered down.

Table 12 Scale/KX711 Power Consumption

Scale Components Power Consumption			
	Operating Voltage	Operating Current	Power Consumption
1 kg Load Cell	~5-10V	Negligible	Negligible
HX711 ADC	~2.7-5V	<1.5mA	~4mW-~7.5mW

Since the operating voltage for these components are anywhere between 2.7V and 5.5V for the HX711 ADC. The load cell operates at 5V to 10V, which means we have a range of voltage where both components can operate in. We are bypassing the ADCs internal voltage regulator and use an external voltage regulator instead; the voltage will be regulated to 5V which will meet both components operating voltage demands. This provides simplicity to the power supply design.

3.1.7.1.3 Power for OLED Display

The LCD display will serve as the primary interface for the user by showing the weight of the breast milk bags in ounces, the date and time, and possibly other system notifications such as a low battery level message, printer maintenance message, errors etc. The display will remain on as long as the PUMPED device is in use, so it will require continuous power.

Table 13 OLED Display Power Consumption

OLED-Display Power Consumption		
Operating Voltage	Operating Current	Power Consumption
3.3V	~38mA (Full off)	~125.4 mW
	~196 mA (Full on)	~646.8 mW

To power this display, we are using the same regulated 3.3V lines that will power the MCU. While the LCD screen will subject the system to continuous power consumption, there are ways that we can maximize battery life by implementing a sort of idle mode to the screen when no new info has been entered for a certain amount of time.

3.1.7.1.4 Power for Thermal Printer

The thermal printer is an essential component in the PUMPED device, it is responsible for printing the labels for the user to attach onto the breast milk bags. When compared to the previously discussed components, the printer will be the most power intensive by far because it requires high current draw while it is printing the labels. Designing the power system to handle these high demands will be essential to avoid voltage drops or insufficient power to other subsystems.

According to the available information on the printer, it will operate at a voltage between 5-9V drawing between 0.8-1.5A while printing. The website also mentions that it draws an instantaneous current of around 2A. This means that our system must be designed to handle at least 2A so that the peak current does not lead to any issues. The website also mentions that the operating voltage of the thermal printer is between 5-9V and specifies that the voltage directly affects the speed and print quality of the printer.

Table 14 Printer Power Consumption

Thermal Printer Power Consumption		
Operating Voltage	Operating Current	Power Consumption
9V (Max)	~2A (Peak)	~18W (Peak)
	~0.8A~1.5A (Normal)	~7.2W~13.5W

7V	~2A (Peak)	~14W(Peak)
	~0.8A~1.5A (Normal)	~5.6W~10.5W
5V (Min)	~2A (Peak)	~10W(Peak)
	~0.8A~1.5A (Normal)	~4W~7.5W

The operating voltage that we choose for the thermal printer will be essential for our system to function properly because it will directly affect both the quality of the printed label and the power consumption of the printer. If we supply the printer with a voltage regulated input of 5V, we may save substantial power, but the print quality and speed may be poor. However, if we supply the printer with a 9V input, then the opposite could occur, where the print quality is amazing, but our power consumption is high and significantly lowers our battery life. It is essential for our design that we find the optimal voltage level where we have a balance between printer speed/quality and power consumption.

When designing the power supply, we may need can try to isolate the printers power input so that the current draw does not affect any of our other systems. This means that we may need a dedicated voltage regulator just for the thermal printer. Another challenge we faced was handling the peak current of 2A. A potential solution could be adding capacitors to help reduce voltage drop, possibly leading to less strain on the battery. The high current demand from the thermal printer also means that our battery must be capable of sustaining its peak loads. Therefore, one of the main influences on our battery choice will be the power consumption of this printer.

3.1.7.1.5 Estimated Total Power Consumption

Now that we have defined each individual subsystems power requirements, we can move on to estimating the total power consumption for the PUMPED system. This estimate of power consumption will directly influence our choice of battery and the design of our power supply system. Here we considered what the normal use of the device could look like and what power requirements must be met, and we also estimated what consistent peak power demands could look like. Below, we have a table with a summary of all the subsystems and their power consumption.

Table 15 Power Consumption Summary by Component

Power Consumption Summary by Component			
Component	Operating Voltage	Current Draw	Power Consumption
ESP32 (Active Mode)	3.3V	~95mA - ~240mA	~313.5mW - ~792mW
ESP32 (Sleep Modes)	3.3V	~10 μ A - ~50mA	Negligible - ~165mW
1 kg Load Cell	5V	(TBD, assumed Negligible)	(TBD, assumed Negligible)
HX711 ADC	5V	<~1.5mA	<~7.5W

OLED Display	3.3V	~38mA- ~196mA	~125.4mW- ~646.8mW
Thermal Printer (5V)	5V	~0.8A - ~2A	~4W - ~10W
Thermal Printer (7V)	7V	~0.8A - ~2A	~5.6W - ~14W
Thermal Printer (9V)	9V	~0.8A - ~2A	~7.2W - ~18W

Since the power consumption varies depending on usage, we can consider two scenarios. In the first scenario we considered the normal usage of the device, let's assume that:

- The ESP32 functions consistently at 3.3V and draws around 170mA , which is the midrange between 95mA and 240mA.
 - Power consumption: 561mW
- The scale and load cell function at 5V and draw around 1.5mA.
 - Power Consumption: 7.5mW (Negligible)
- The LCD functions consistently at 3.3V and we assumed that on average, it will draw 120 mA. This is the midrange between 38mA and 196mA.
 - Power Consumption: 396 mW
- And the thermal printer is operating at 7V and draws around 1.15A, which is the midpoint between the operating currents of 0.8A and 1.5A.
 - Power Consumption: 8.05W

Under these conditions, the total estimated power consumption could be around 9W. Since these conditions are assuming normal usage of the device, this power rating will be considered the base line for the device. Now, we can consider the second scenario where the power consumption for the PUMPED device is constantly working at peak power levels. In this case, we can assume that:

- The ESP32 functions consistently at 3.3V and draws around 250mA.
 - Power consumption: 825mW
- The scale and load cell function at 5V and draw around 1.5mA.
 - Power Consumption: 7.5mW (Negligible)
- The OLED display functions at 3.3V and draws 196mA.
 - 646.8mW
- And the thermal printer is operating at 9V and draws around 2A consistently and not just instantaneous.
 - Power Consumption: 18W

Under these conditions, the total estimated power consumption for the system would be around 19.4793W. Since these conditions assume that the device is always functioning at peak performance, we used it as a max rating for the power consumption of the system.

Table 16 Estimated Total Power Consumption of PUMPED Device

Estimated Device Power Consumption	
Scenario	Power (W)

Typical Operation (Baseline)	~9W
Constant Peak Load (Maximum)	~19.4793W

Now that we have estimates for both the base line and max rating for the power consumption that the PUMPED device could require, we can start deciding on what kind of rechargeable battery would best suit our needs.

3.1.7.2 Rechargeable Battery Options

As mentioned earlier, the PUMPED system requires a rechargeable battery to provide portability and reliable functionality to the user. Choosing the appropriate battery technology is going to be essential to ensure that all systems are powered properly. Determining the necessary capacity needed to run the device for at least 10 hours before needing to be charged again will also be crucial to meeting our battery life requirement. Now that we have an estimate for the power consumption, we can calculate the battery capacity, we also evaluated different technologies and determined which would be the best fit for our system.

3.1.7.2.1 Battery Capacity Requirements

To guarantee that the PUMPED system can be used for a total of 10 hours before requiring a recharge, we have to calculate the minimum battery capacity needed. This calculation will be done based on the power consumption estimates we have found. However, the actual battery selection will also depend on other factors such as output voltage levels, energy efficiency, and other battery specifications such as size, weight, etc. The goal is to find the optimal battery for the device.

3.1.7.2.2 Lithium-Ion (Li-Ion) Batteries

Lithium-ion (Li-ion) batteries are a kind of rechargeable battery that uses the transferring of lithium ions between electrodes to produce a charge difference. When the battery is being used, the lithium ions move from the negative electrode, which is the anode, to the positive electrode, which is referred to as the cathode. This produces a release of energy. When the battery is being recharged, the process is reversed. Meaning that the lithium ion is now going from the cathode to the anode and being stored there for later use. Lithium-ion batteries usually operate at around 3.6 to 3.7 volts per cell with the ability of combining cells to achieve higher voltages.

Some advantages that come with lithium-ion batteries is that they offer a high energy density at around 150-250 Wh/kg, meaning that for their size, they can hold a good amount of energy. This is why they are widely used across our portable devices with high power demands like cellphones and electric vehicles. Moreover, they also have the advantage of having a low discharge rate when they are not in use. The average lithium-ion battery will lose around 0.35% to 2.5% every month depending on how much charge it has at the start. This ensures that even if the device that is powered by this kind of battery can hold its charge for a substantial amount of time.

However, Li-ion batteries do come with some downsides. The most significant one being that they can pose some safety risks. The electrolytes used in a lithium-ion battery are flammable and if the batteries protective case breaks due to overheating causing it to expand and puncture the case then this could lead to an explosion of the battery or a fire wherever the incident occurs.

Table 17 Technical Properties of Lithium-Ion Batteries

Technical Properties of Lithium-Ion (Li-ion) Batteries		
Property	Value / Range	Additional Info
Nominal Voltage	3.6V - 3.7V per Cell	Cells can be connected in series to achieve higher voltages.
Charge Voltage	3.7-4.2V	Anything over 4.2 volts could damage the cells.
Current Limit	Depends on C-Rating of Battery	A higher C-rating means a higher maximum current. (A 10C rating means the battery can discharge 10x its capacity in amps)
Energy Density	150-250 Wh/kg	High Energy density. Helps with lower weight and size of battery needed.
Recharge Cycles	300-500 Charge Cycles	Number of times the battery can be discharged and recharged before substantial degradation.
Safety Risks	Fire/Explosions	A charge management system is needed to ensure the cells do not get overcharged or receive too much current.

In the case of our PUMPED device, Li-ion batteries could be a suitable power source considering that these batteries have a high energy density and a low discharge rate. Since they have a high energy density, we could opt for a smaller battery pack, making the device more portable and lighter. Moreover, its low discharge rate when not in use could be convenient for the user because they would not have to worry about consistent charging when it hasn't been used. However, we still must consider the safety concerns that these batteries come with. If we do implement them into our design, we must ensure that they are handled safely, with proper charge management, and ensure that the batteries voltage and current outputs can match our components' needs.

3.1.7.2.3 Lithium Polymer (LiPo) Batteries

Lithium Polymer (Li-Po) batteries are another type of rechargeable battery, and it is very similar to the lithium-ion batteries we previously discussed. While the two batteries use the same fundamental technology of exchanging lithium ions between two electrodes to produce a charge, their main difference comes from the kinds of electrolytes inside the

battery. The Li-ion battery uses a liquid electrolyte, while the Li-Po battery uses a gel like substance, it is a high conductive semi solid polymer. The nominal voltage levels for a Li-Po battery is around 3.6 to 3.7 volts, however, the voltage can vary depending on the charge level of the battery. When it is fully charged, the voltage is around 4.2V and when it is completely discharged, it can reach lower levels of 1.8 to 2.0 volts.

Some advantages of using Li-Po batteries are that like Li-ion batteries, they have a relatively high energy density of around 100 to 200 Wh/kg. This is slightly lower than the Li-ion battery but still high compared to other kinds of rechargeable batteries. Another advantage to using a Li-Po battery is that it can be discharged and recharged a high number of times. They usually last around 400-600 charge cycles before significant battery deterioration takes place. However, this can be affected by how the user uses the battery. For instance, things like constantly charging it to its max, or constantly letting it discharge to its lower voltage limits could cause it to degrade and lead a shorter life cycle. Li-Po batteries are manufactured differently from Li-ion batteries allowing for smaller and more flexible designs for the shapes and sizes of their cases. This is why they are widely used in small devices like drones, smartphones, and smartwatches.

However, this kind of battery does come with its disadvantages. For example, from an industry standpoint, these batteries are much more expensive to produce than other kinds of Lithium-Ion batteries. Another big disadvantage that these batteries pose is that they are very sensitive to overcharging, or over discharging. If the battery is overly discharged the battery life could be damaged and work a lot less effectively. The big danger comes from these batteries being overcharged, if a lithium polymer battery is overcharged, the gel inside of it heats up and its battery case can expand. If this case is punctured, the gel could leak out and the lithium could cause a fire or explosion. This means that the use of lithium polymer batteries must be done cautiously, and it must have a proper way to ensure that its charge is being controlled to limit these dangerous situations.

Table 18 Technical Properties of Lithium-Polymer Batteries

Technical Properties of Lithium-Polymer (Li-Po) Batteries		
Property	Value / Range	Additional Info
Nominal Voltage	3.6V - 3.7V per Cell	Cells can be connected in series to achieve higher voltages.
Charge Voltage	3.7-4.2V	Anything over 4.2 volts could damage the cells.
Current Limit	Depends on C-Rating of Battery	A higher C-rating means a higher maximum current. (A 10C rating means the battery can discharge 10x its capacity in amps)
Energy Density	100-200 Wh/kg	High Energy density. Helps with lower weight and size of battery needed.

Recharge Cycles	400-600 Charge Cycles	Number of times the battery can be discharged and recharged before substantial degradation.
Safety Risks	Fire/Explosions	A charge management system is needed to ensure the cells do not get overcharged or receive too much current.

Considering the PUMPED devices requirements for portability and long battery life, these Li-Po batteries could be a possible option since it has a high energy density and is lightweight. These batteries are usually flatter in design compared to Li-Ion batteries which could be useful for fitting it into a compact area in our device. However, we must take into consideration the safety concerns that come with the use of Lithium-Polymer batteries. They are highly flammable and explosive, and a robust battery management system must be implemented to safely use the battery.

3.1.7.2.4 Nickel-Metal Hydride (Ni-MH) Batteries

Nickel-metal hydride batteries (Ni-MH) are a kind of rechargeable battery that uses a chemical reaction between two electrodes to produce electricity. The positive electrode plate (cathode) is made of a nickel oxide-hydroxide, and the negative electrode plate (Anode) is made of a metal hydride (a hydrogen absorbing alloy). These two electrodes are separated by an alkaline electrolyte such as potassium hydroxide and a separator made from a polymer film. When the battery is being used (discharged), the hydrogen inside of the anode is oxidized to a proton. This chemical reaction releases the electrons stored in the metal hydride this produces a voltage of around -0.89V.

The separator prevents direct contact between the positive electrode and the negative electrode. On the other side of the separator, the cathode goes through a different chemical reaction where the nickel oxide-hydroxide is reduced to nickel hydroxide and produces a voltage of around +0.49V. In the end, both of these reactions leave us with a battery that produces around 1.2 to 1.4V. When we supply power to these batteries (Recharge), the chemical reaction is reversed, though not perfectly. For this reason, the battery loses its capacity over time. Depending on the batteries usage patterns and charging habits, they can go through anywhere between 500 to 200 charge cycles before significant capacity loss. These batteries have a discharge rate of around 50 to 80 Wh/kg, which is significantly lower than the lithium-ion, and lithium polymer batteries we previously discussed.

Some advantages that come with the use of these Ni-MH batteries include the fact that they are very good conductors of electricity and therefore allows them to adapt adequately to high power discharge applications. This is why these Ni-MH batteries are typically used in devices like digital cameras and flashlights. Another advantage these batteries have is that they output a similar voltage to regular alkaline AA sized batteries and are the same size. This means that they are widely used to replace these non-rechargeable batteries. One very big advantage with these batteries is that they are a lot safer than other kinds of batteries such as Li-Po and Li-ion. If a Ni-MH battery is overcharged, it could lead to overheating which would cause hydrogen gas to leak out of the battery. However, the battery will not explode and catch on fire, it will just vent the excess pressure. This would, however, damage the battery and diminish its life cycle.

However, these batteries do have some downsides, one of the main downsides is that they do not have high energy density, which means that to power a device for an extended period of time we would need significantly more batteries to do so. This would increase weight and space required for the battery pack. Moreover, they are not very efficient when charging. This would mean that a device using these batteries would take a lot longer to fully charge than if it used a Li-Po or Li-ion battery.

Table 19 Technical Properties of Nickel-Metal Hydride Batteries

Technical Properties of Nickel-Metal Hydride (Ni-MH) Batteries		
Property	Value / Range	Additional Info
Nominal Voltage	1.2 to 1.3V	Cells can be connected in series to achieve higher voltages.
Charge Voltage	1.4V to 1.5V	Anything over 1.5 volts could damage the cells.
Current Limit	Depends on C-Rating of Battery	A higher C-rating means higher maximum current. (A 10C rating means the battery can discharge 10x its capacity in amps)
Energy Density	50-80 Wh/kg	Low Energy density compared to Li-ion and Li-Po. This means more batteries would be needed, increasing the size and weight of the device
Recharge Cycles	500-2000 Charge Cycles	Number of times the battery can be discharged and recharged before substantial degradation.
Safety Risks	Overheating, venting	Ni-MH batteries are a lot safer than Li-Po and Li-ion batteries. Charge must be managed but they will not explode or burst into flames. If the battery overheats, it creates hydrogen and the pressure is vented.

In the case of our PUMPED device, these NI-MH batteries could be a viable solution to our needs of portability and extended battery life. They have the ability to withstand high currents that we may need for our thermal printer, and they can be recharged a lot of times. Another good thing these batteries have over the Lithium ion and lithium polymer batteries is that they are much safer and easier to handle. Since our device is portable and will be used around mothers with babies, safety is a huge concern. However, it does not come without its downsides. These batteries offer much lower voltage levels and much lower

energy densities than other options. This means that the overall weight and size of the device would be affected, and the batteries will need to be charged more frequently. While these batteries are safer, we would still need a proper battery charge system that would keep these batteries from being overcharged so that we can ensure that they last as long as possible before needing to be replaced.

3.1.7.2.5 Sealed Lead-Acid (SLA) Batteries

A sealed lead-acid battery is a kind of rechargeable battery that uses a chemical reaction to produce electricity. The negative electrode (cathode) is made of lead, and the positive electrode (Anode) is made of lead dioxide. These two metals are suspended in an electrolyte that consists of sulfuric acid. When a cell is discharged the lead and diluted sulfuric acid produces lead sulfate and water along with a charge. When the battery is recharged, this chemical reaction is reversed, and the lead sulfate and water are turned back into lead and acid. The sealed design of these lead batteries incorporates a layer of safety where it allows internal pressure build ups due to charging to be expelled through safety valves that open when a certain pressure is reached. These individual cells can output a voltage of around 2V, but they are usually sold in sealed battery packs that output a voltage of around 6 or 12V. SLA batteries usually have an energy density of around 30-40 Wh/kg. This is very low when we compare it to the other options we mentioned. These batteries, however, are very good at outputting large amounts of current in very little time, this is why they are used in cars to start motors, and they are even used in things such as medical devices like defibrillators.

Some of the advantages that these kinds of batteries have is that since they are sealed, the electrolyte mixed with sulfuric acid is contained which prevents any spilling of dangerous chemicals. Moreover, if the battery is overcharged, it is equipped with safety valves that allow it to relieve pressure in a safe way, preventing the battery from expanding and leaking. This means that we do not have to worry about one of these batteries exploding or causing a fire such as the lithium-ion or lithium polymer batteries. Another advantage to the sealed design is these batteries can deliver significant power when needed over short periods of time.

However, some of the negative things that come with using these batteries is that they are very bulky and heavy. Not only are they bulky, but they have a significantly low energy density, which means that compared to other battery types, this one would take up a lot more room. Last but not least, these batteries are made of lead and sulfuric acid. These are toxic materials and when the time comes to dispose of the battery, it must be taken to a proper disposal place where they will correctly handle the toxic materials and prevent contamination of the environment.

Table 20 Technical Properties of Sealed-Lead Acid Batteries

Technical Properties of Sealed Lead-Acid (SLA) Batteries		
Property	Value / Range	Additional Info
Nominal Voltage	2V	Typically sold in packs where cells are in series, producing 6V or 12V.
Charge Voltage	2.3-2.4V	Each battery is different and depends on the specific design of the manufacturer. Anything over the recommended charge voltage could damage the battery.
Current Limit	High current limits	These SLA batteries are typically capable of handling high currents over short periods of time.
Energy Density	30-40 Wh/kg	very low energy density which means it adds a lot of weight, and a bigger space is needed to store the battery.
Recharge Cycles	<350 Charge Cycles	Number of times the battery can be discharged and recharged before substantial degradation.
Safety Risks	Gas buildup, Gas venting	These SLA batteries are much safer than options like Lithium-ion and Lithium-Polymer because they are sealed and have safety valves to allow pressure buildups to escape if the battery is overcharged. These safety precautions avoid any potential dangers of explosions due to pressure buildup or acid leaks.

Considering the PUMPED device and its requirements, an SLA battery would be a viable option for managing the peak current draw from our thermal printer. They are also very safe since the lead and acid are completely sealed with proper venting for possible built-up pressure release. However, we have concerns with the size and weight that an SLA battery could add to the PUMPED device. The device is supposed to be as portable as possible, and these batteries run on the heavier side. Another issue with using these batteries, is that the system is being designed to be in very close proximity with mothers and their children and will be in direct contact with milk intended for the consumption of an infant. Because of this, we would like to refrain from using these since they contain lead and sulfuric acid.

While they are sealed, it is best to avoid any accidental spillage of acids due to the battery releasing built up pressure via its safety valves.

3.1.7.2.6 Portable Battery Banks

Portable battery banks are a widely used way of storing electrical energy for later use. They typically use multiple lithium-ion battery cells sealed inside a container that protects the cells from being exposed to the elements. Moreover, they typically consist of charging circuits to protect the cells from being overcharged, and sometimes they can deliver different voltage levels via different outputs. Battery banks come in many different compact designs and are widely used by consumers to power things such as cell phones, tablets and the more powerful ones can even power laptops. The power banks capacity is usually listed by the manufacturer in terms of milliamp-hours (mAh) or in terms of watt-hours (Wh). This capacity indicates the total charge that the device can hold, meaning that a higher capacity can power devices for longer periods of time.

Choosing the portable battery bank option over other battery types comes with its own advantages. One of the biggest advantages is that they already come prepared with their own battery management systems. Meaning that we would not have to worry about overcharging or discharging the power bank to the point that the battery cells get damaged. Another advantage is that a lot of these power banks come with different power output options, with some having multiple output ports. This gives us versatility in our design,

However, they do come with certain disadvantages. Power banks vary in size and power output, so it is essential that a bank powerful enough to handle the thermal printers' peak currents is properly found.

Table 21 Technical Properties of Portable Battery Banks

Technical Properties of Portable Battery Banks		
Property	Value / Range	Additional Info
Nominal Voltage	Varies, typically 5V (USB Standard), some offer 9V, 12V or Higher	Higher voltages like 9V or 12V could be useful for powering our subsystems. (E.g., Thermal Printer:9V max)
Charge Voltage	Typically charged via USB-C, Micro-USB, etc.	Typical chargers used for cell phones could be used to recharge the power bank
Current Limit	Varies between Battery bank	Higher current battery banks are typically used for larger devices like laptops.
Energy Density	150-250 Wh/kg	Battery banks typically use Li-ion battery cells; therefore, they have a high energy density
Recharge Cycles	300-500 Charge Cycles	Number of times the battery can be discharged and recharged before substantial degradation.

Safety Risks	Minimal risk. Power limits must not be exceeded	While power banks typically have their own battery management systems to avoid overcharging of Li-Ion cells, their limits must still be considered. We need to ensure that if we are to use a battery bank in our device, that it can manage the peak currents drawn by the thermal printer along with the current from the other subsystems. Besides this, they are the safest choice by far compared to other batteries.
--------------	--	--

In the case of our PUMPED system, a battery bank could be a workable solution to fit our design. The main advantage it has over other battery types is that they are capable of delivering power via the use of lithium-ion cells without the dangers that using the raw cells comes with. A power bank is already designed to protect and ensure the Li-ion cells inside of it are not subjected to any dangerous voltages or currents that could lead to fires. This would be a compact, safe, and effective battery for our system. We just have to ensure that the battery banks capacity, voltage output, and max current ratings match our needs.

3.1.7.2.7 Battery Type Comparison and Selection

Selecting the optimal battery type for our PUMPED system is essential to ensure that the battery meets all operating requirements such as being able to meet the peak current demands from our thermal printer of 2A, voltage levels between 3.3V and 9V for all components, and it must have a capacity to last at least 10 hours of constant usage. From our previous calculations, the device will have a typical usage of around 8.62W, and a peak load of around 18.83W. This means that the minimum battery capacity needed for the device to last 10 hours is 86.2Wh, and the maximum battery capacity is 188.3Wh. The device operating at max load is a highly unlikely scenario in real use cases, so a battery of around 100Wh should be sufficient. In our decision, we took into account the portability (Size and weight) of each battery type, and we considered how safe they are to use in our device.

Table 22 Battery Technology Comparisons

Battery Comparison Table					
Battery Type	Nominal Voltage	Charge Voltage	Energy Density	Current Limit	Note
Lithium-Ion (Li-Ion)	3.6-3.7V per cell	3.7-4.2V	150-250 Wh/kg	Depends on C-Rating of Battery	1
Lithium-Polymer (Li-Po)	3.6-3.7V per Cell	3.7-4.2V	100-200 Wh/Kg	Depends on C-Rating of Battery	1
Nickel-Metal Hydride (NiMh)	1.2V per Cell	1.4-1.5V	50-80 Wh/kg	Depends on C-Rating of Battery	2
Sealed Lead-Acid (SLA)	Typically sold in sealed 6V or 12V packs	2.3-2.4V	30-40 Wh/kg	Typically, High current limits	3

Portable Battery Bank	Typically found in 5V, 9V, or 12V	Typically charged via USB 5V	100-250 Wh/kg	Varies between Battery bank	4
-----------------------	-----------------------------------	------------------------------	---------------	-----------------------------	---

Notes
¹ High risk of Fire/Explosions without proper Battery management system. ² Low Risk has safety valves for pressure relief of hydrogen gas buildup. Battery management system needed. ³ Low risk of catastrophic failure, sealed container has safety valves to vent built up pressure. Made of toxic materials like lead and sulfuric acid. Battery management system is needed. ⁴ Low risk, equipped with its own battery management system. Power limits must not be exceeded.

After taking into account all of the different battery technologies and contemplating their strengths and weaknesses, the portable battery bank stood out as the most practical and user-friendly solution. One of the strongest deciding factors in our selection was that the power banks offered the safest circumstances to our design. Battery banks offer the high energy density of lithium-ion/Polymer cells without the risk of fires or explosions because they come pre-equipped with their own battery management circuits. The other benefit to battery banks is that they are sold in a variety of compact and lightweight designs, which aligns perfectly with our need for a compact and portable design. After researching and comparing various options online, we found that the best fit for our voltage, current, and power demands were best met by higher capacity power banks.

3.1.7.2.8 Battery Bank Comparisons and Selection

In order to ensure that the PUMPED system meets its design requirements, particularly having a runtime of at least 10 hours. In our power consumption calculations, we determined that the system will use a baseline of 9W to operate. If we multiply these 9W by 10 hours of usage, we see that battery bank must have a minimum capacity of at least 90 Wh. Multiple high-capacity battery banks were considered and evaluated based on their total energy capacity (Wh), available output voltage levels and current ratings that meet the high demands of our thermal printer, and size and weight of the battery bank.

Table 23 Battery Bank Comparisons

Battery Comparison Table					
Battery Type	I/O Ports	Energy Capacity (Wh)	Size (Inches)	Weight (lbs.)	Cost
Talent Cell YB1208300-USB	•(Input/Output) DC5521:12V/6A •(Output) USB-A: 5V/2A •(Output) DC5525: 9V/1A	9Ah @ 11.1V ≈ 99.9Wh	5.39"x 3.11"x 1.54"	1.57	\$47.99
ROMOSS PHP30 Pro	•(Input)Micro-USB: 5V/2A, 9V/2A •(Input) Lightning: 5V/2.1A •(Input/Output) USB-Type-C/B: 5V/3A, 9V/2A, 12V/1.5A •(Output) USB-A#1: 5V/3A, 9V/2A, 12V/1.5A •(Output) USB-A#2: 5V/2.1A	30Ah @3.7V ≈ 111Wh	6.6"x 3.1"x 1.3"	1.48	\$26.99
ROMOSS PEA40-282	•(Input)Micro-USB: 5V/2A, 9V/2A •(Input) Lightning: 5V/2A, 9V/2A •(Input/Output) USB-Type-C: 5V/3A, 9V/3A, 12V/3A, 15V/3A, 20V/3.25A •(2x Outputs) USB-A: 4.5V/5A, 5V/4.5A, 9V/2A, 12V/1.5A	40Ah @3.7V ≈ 148Wh	6.67" x 3.21" x1.73"	2.1	\$54.99

When comparing these 3 battery banks, we determined that the Talent Cell has sufficient power to reach our goal of 10 hours of runtime. However, to power our thermal printer, which needs at least 5V/2A, we would need to use the DC5521 port for power. This would mean that to recharge the bank, we would have to unplug a cable from our system. This is not ideal and not user friendly, so even though it is the smallest of the three options, and has just enough capacity to match our needs, we did not pick this battery bank.

Moving on to the ROMOSS PEA40-282, while this battery bank has the highest capacity, and output power capabilities of the three, we decided that it was just too much for what we needed. It is the largest and heaviest of the three, which means it would make our overall system heavier and larger than necessary. It is also the most expensive of the three, so, overall, these are the reasons that we did not go with this battery bank.

In the end, our final choice of battery was the ROMOSS PHP30 Pro Power Bank with a capacity of 30,000mAh. This battery bank offers us with 2 capable output ports to power our system with a USB-A output capable of delivering 5V at 3A, and another capable of delivering 5V at 2 A, along with a 111Wh of capacity which is well above the 90Wh we were looking for. This will ensure that even under higher-than-expected loads, the device will still be able to function for more than 10 hours on a single charge. Another good thing

about this battery bank is that it is also the cheapest of the three options while being the lightest of the three. Overall, this seemed like the best choice with a great balance in size, power capabilities, weight, and price.

3.1.8 Voltage regulation

Another key aspect of our PUMPED design is voltage regulation. Voltage regulators are an integral part of the power supply system that serve as an intermediary between the power supply and the rest of the electronic components. Our power supply system is tasked with providing constant, stable, and regulated power so that all the electronic components can work in their respective operating voltage ranges, so that each component can function properly. Our design will be powered by batteries, which will provide direct current (DC) to all the components. A voltage regulator is defined as a device or a circuit that is responsible for providing a steady DC voltage to an electronic load. To accomplish this, all voltage regulators are generally comprised of three parts. The first part being the control circuit, which has the responsibility of checking the magnitude of unregulated voltage from the power source and passing regulated voltage as output. The second part is the sampling circuit. Its function is to sample a proportion of the output voltage and then compare it to the reference voltage.

The third part is the comparator circuit, its function is to compare the feedback signal with a fixed reference voltage so that it can generate a control signal for the control element. Now we will go into the different types of voltage regulators there are and discuss which regulator is the most suitable for our needs. There are two types of voltage regulators: Linear Voltage Regulators and Switching Voltage Regulators. These voltage regulators work off the principles of how a transistor operates in their control circuits. The transistor can pass regulated output voltage in different ways according to its operating modes. The modes include saturation mode, cutoff mode, active mode, and reverse active mode. The modes that describe how a voltage regulator functions are the saturation mode, cutoff mode and the active mode. In active mode the transistor creates an output that is proportional to the input of the transistor. The cutoff mode is when the transistor functions as an open circuit and no output is produced. Saturation mode for a transistor is when it acts like an open circuit where output isn't restricted.

3.1.8.1 Linear Voltage Regulators

When a transistor stays in the active mode while operating within a voltage regulator it is a linear voltage regulator. When a load is connected to a linear voltage regulator, the input will vary the current through the transistor so that the output is maintained constant. For this operation to work, scaling down the input to the output voltage, the regulator dissipates the lost power as heat. There are three types of linear voltage regulators.

1. **Standard Regulator-** This type of regulator typically uses a BJT or MOSFET as a pass transistor. This type can be further divided into different categories:
 - Positive voltage regulators- Output produced is positive.
 - Negative voltage regulators- Output produced is negative.
 - Adjustable voltage regulators- Output can be adjusted to a selected voltage within the range that the voltage regulator is designed for.

Standard Regulators Characteristics:

- Require higher input voltage compared to the output voltage
- Does not require many components
- Regulates voltage well but are less efficient due to heat dissipation
- Limited in their current capacity

2. Low Dropout (LDO) Regulator- This type of regulator can operate when the supply voltage has a small difference to the output voltage.

LDO Regulators Characteristics:

- The LDO regulator has the best dropout voltage of the three regulator types making it great for battery powered operations.
- They have high efficiency since they can maximize the utilization of the available input voltage
- Higher complexity due to the amount of components

3. Quasi Low-Dropout (quasi-LDO) Regulator- The dropout voltage for the quasi-LDO is higher than the LDO but lower than the standard regulator.

- Best used when a combination of Standard and LDO characteristics is needed
- Moderate efficiency

Table 24 Linear Voltage Regulator Comparisons

Type	Efficiency	Dropout Voltage	Complexity	Best suited for
Standard Linear Regulator	Depends on dropout voltage	Typically, >2V	Simple internal structure	AC and DC low power applications
LDO Regulator	Improved due to low dropout voltage	Typically, <0.5V	Requires pass elements or MOSFETs	Battery Powered applications
Quasi-LDO Regulator	Better than LDO	0.1V-0.4V	Uses advanced control transistors	Higher current applications

3.1.8.2 Switching Voltage Regulators

The other type of voltage regulator is called the switching voltage regulator. The switching voltage regulator differs from the linear version in numerous ways. This type of regulator is greatly more efficient than a linear voltage regulator due to its ability to store and transfer excess energy through inductors and capacitors, unlike a linear voltage regulator that dissipates excess energy as heat. The standard application for switching regulators is through battery powered systems. And unlike linear regulators, switching regulators can increase the voltage it regulates. Switching regulators are more complex in design because it converts DC voltage using a high frequency switching element. Though the switching

element makes the switching regulator more efficient there are drawbacks. One being the increase in price for the complex components and second being the noise the switching element creates. There are several types of switching regulators that can be useful in different types of applications:

1. Buck Regulators- This is the most common type of switching voltage regulator whose function is to convert a higher DC voltage into a lower one at the same polarity.
2. Boost Regulators- This type of regulator takes a DC input voltage and produces a DC output voltage that is higher than the input voltage at the same polarity.
3. Buck-Boost Inverting Regulators- This type of regulator will take an input DC voltage and invert the polarity to make it opposite of the input and either increase or decrease the output voltage depending on the application.
4. Flyback Regulators- This type of regulator is the most versatile regulator and has become a popular choice in battery powered systems due to its ability to produce one or more output voltages of differing polarities. This type of regulator also has high efficiency like the rest of the switching regulators.

There are more types of switching voltage regulators, but the ones listed above are more sufficient for low power operations compared to the ones not listed.

Table 25 Switching Voltage Regulator Comparisons

Type	Function	Best suited for
Buck Regulator	Reduces input voltage	Microcontrollers, Processors
Boost Regulator	Increases input voltage	Battery powered devices, LED drivers
Buck Boost Inverting Regulator	Increases or Decreases input voltage along with polarity	Battery powered devices with fluctuating voltage
Flyback Regulator	Can produce multiple different output voltages	Small power adaptors, Isolated power supplies

Now after stating the facts about the two possible types of voltage regulators that we could have used in our PUMPED design, we will discuss voltage regulators that are in the market and the regulator we chose to be the best suited for our design. When choosing a voltage regulator, we keep in mind the differences between linear and switching regulators:

Table 26 Comparing Voltage Regulator Technologies

Characteristics	Linear Regulator	Switching Regulator
Efficiency	Low due to heat loss	Great due to advanced energy management
Complexity	Simple (fewer components)	More complex
Power Loss	High due to heat loss	Negligible compared to linear regulators
Noise	Low due to simple components	High due to switching element
Initial cost	~\$1.00-\$5.00	~\$3.00-\$15.00
Best suited for	Low-power, Noise sensitive circuits	Battery efficiency, Precise voltage control

3.1.8.3 AMS1117

The AMS1117 is a low dropout voltage regulator designed to provide stable voltage outputs with minimal voltage loss. It offers both fixed voltage options, including 1.2V, 1.5V, 1.8V, 2.5V, 3.3V, and 5.0V, as well as an adjustable version, with output accuracy of $\pm 1.5\%$ for most fixed versions and $\pm 2\%$ for the 1.2V variant. With a dropout voltage of 1.2V at 1A, it is ideal for battery-powered systems and other electronic applications that require efficient voltage regulation. The AMS1117 also features built-in current limiting and thermal shutdown for added protection, ensuring reliable operation across a wide temperature range from -40°C to 125°C . Its combination of accuracy, protection, and efficiency makes it a reliable choice for powering microcontrollers, sensors, and other low-voltage circuits.

3.1.8.4 TLV767

The TLV767 is a wide input linear voltage regulator that supports an input voltage range from 2.5V to 16V and delivers up to 1A of output current. It offers both fixed (0.8V to 6.6V) and adjustable (0.8V to 14.6V) output versions, with 1% output accuracy, making it suitable for low-voltage microcontrollers and processors. Designed for high efficiency in standby mode, it has a low quiescent current (50 μA typical, 1.5 μA in shutdown mode) to meet stringent low-power requirements. The high-power supply rejection ratio (PSRR) of 70dB at 1kHz and 46dB at 1MHz makes it effective in minimizing noise from switching regulators. Additional features include internal soft start, foldback current limiting, and thermal protection for improved reliability. The TLV767 is available in compact WSON, HVSSOP, and SOT-23 packages, making it ideal for appliances, TVs, motor drives, printers, routers, and other consumer electronics applications.

3.1.8.5 LM2576

The LM2576 series is a monolithic step-down (buck) switching regulator capable of delivering a 3A load with excellent line and load regulation. Available in fixed output voltages of 3.3V, 5V, 12V, and 15V. With a fixed 52-kHz internal oscillator, the LM2576 requires only four external components, making it simple to use while including built-in

fault protection, low-power standby mode, and external shutdown with a standby current of just 50 μ A. Designed as a high-efficiency replacement for traditional three-terminal linear regulators, it significantly reduces heat sink size, and in some applications, eliminates the need for one entirely. The series supports standard inductors available from multiple manufacturers, streamlining the design of switch-mode power supplies. Additional features include a $\pm 4\%$ tolerance on output voltage under specified conditions, $\pm 10\%$ oscillator frequency accuracy, cycle-by-cycle current limiting, and thermal shutdown for robust fault protection. Applications range from motor drives, network and server power supplies, and appliances to test and measurement equipment.

3.1.8.6 AP62300T

AP62300T is a 3A synchronous buck converter designed for high-efficiency step-down DC-DC conversion across a wide input voltage range of 4.2V to 18V. It integrates a 75m Ω high-side power MOSFET and a 45m Ω low-side power MOSFET, minimizing power losses and improving efficiency. Utilizing Constant On-Time (COT) control, the device ensures fast transient response, easy loop stabilization, and low output-voltage ripple, reducing the need for external components and simplifying design. Engineered for low Electromagnetic Interference (EMI), the AP62300 series incorporates a proprietary gate driver scheme that minimizes switching node ringing while maintaining optimal MOSFET switching performance.

This results in reduced high frequency radiated EMI noise, making it well-suited for applications requiring strict EMI compliance. The AP62300/AP62301 offers an adjustable output voltage from 0.8V to 7V, a low quiescent current of 155 μ A, and a 750kHz switching frequency when operating at 12V input and 5V output. The device achieves up to 83% efficiency at light loads (5mA), making it ideal for power-sensitive applications. Comprehensive protection features, including undervoltage lockout (UVLO), cycle-by-cycle valley current limit, output undervoltage protection, and thermal shutdown, ensure reliable operation under fault conditions.

3.1.8.7 LM2679

The LM2679 series is a monolithic step-down (buck) switching regulator designed to drive loads up to 5A while maintaining excellent line and load regulation. With efficiencies exceeding 90%, it utilizes a low ON-resistance DMOS power switch to minimize power loss. The series includes fixed output voltage options of 3.3V, 5V, and 12V, as well as an adjustable version ranging from 1.2V to 37V. Operating within an input voltage range of 8V to 40V, the regulator features a high-frequency 260-kHz internal oscillator, allowing for the use of smaller external components. The SIMPLE SWITCHER® design simplifies implementation by requiring only a minimal number of external components, and a selection of standard inductors from multiple manufacturers further streamlines the design process.

Additional features include soft-start capability to reduce inrush current at power-on, a resistor-programmable peak current limit adjustable between 3A and 7A, and built-in thermal shutdown protection. The output voltage maintains a maximum tolerance of $\pm 2\%$

under full line and load conditions, while the oscillator frequency remains stable within a $\pm 11\%$ range. With an operating junction temperature range of -40°C to 125°C , the LM2679 is well-suited for high-efficiency step-down regulators, system pre-regulators for linear voltage regulators, and battery charger applications.

3.1.8.8 MP2338GTL-P

The MP2338 is a fully integrated, high-frequency, synchronous step-down switch-mode converter designed to provide a compact and efficient solution for delivering up to 3A of continuous output current. With an operating input voltage range of 4.5V to 28V and an adjustable output voltage range from 0.5V to 16V, it ensures excellent load and line regulation. Featuring a constant-on-time (COT) control scheme, the MP2338 enables fast transient response, simplified loop design, and tight output voltage regulation. The device incorporates low RDS(ON) internal power MOSFETs ($115\text{m}\Omega$ high-side / $55\text{m}\Omega$ low-side) to achieve high efficiency, exceeding 90% under a 24V to 5V/3A load condition. It also supports power-save mode at light loads to further enhance efficiency. Built-in protection features include short-circuit protection (SCP), over-current protection (OCP) with hiccup mode, under-voltage protection (UVP), and thermal shutdown, ensuring robust system reliability. The MP2338 operates at a 450kHz switching frequency and includes a configurable soft-start time, power-good indication, and t-on extension to improve dropout performance. Requiring only a minimal number of external components, it is available in a compact SOT583 package, making it ideal for space-constrained applications. The MP2338 is well-suited for multi-function printers.

Table 27 Selecting the Voltage Regulators

Feature	AMS1117	TLV767	LM2576	AP62300T	LM2679	MP2338
Type	Linear (LDO)	Linear (LDO)	Switching (Buck)	Switching (Buck)	Switching (Buck)	Switching (Buck)
Input Voltage	4.75V-10V	2.5V-16V	7V-40V	4.2V-18V	8V-40V	4.5V-28V
Output Voltage	3.3V or 5.0V	0.8V-6.6V (fixed), 0.8V-14.6V (adjustable)	3.3V, 5V, 12V, 15V, Adjustable (1.23V-37V)	0.8V-7V	3.3V, 5V, 12V, Adjustable (1.2V-37V)	0.5V-16V
Max Output Current	1A	1A	3A	3A	5A	3A
Efficiency	Low	Moderate	High	High	High	High
Switching Frequency	N/A	N/A	52kHz	750kHz	260kHz	450kHz
Dropout voltage	1.2V	0.9V	N/A	N/A	N/A	N/A
Protection Features	Thermal Shutdown	Foldback Current Limit, Thermal Protection	Over-Temp, Over-Current, UVLO	Over-Current, UVLO, Thermal Shutdown	Over-Temp, Current Limit, Soft-Start	Over-Current, Thermal Shutdown, Soft-Start

Best Applications	Battery Powered Systems	Printers, Wi-Fi access points	Test and Measurement equipment	General purpose point of loads	Simple designs	Multi-functional Printers
-------------------	-------------------------	-------------------------------	--------------------------------	--------------------------------	----------------	---------------------------

3.1.8.9 Voltage Regulator Selection

In senior design 2, we determined that voltage regulation to 5 volts was not necessary, due to the power banks voltage capabilities, the thermal printer can be directly powered by the battery. Since the PUMPED device will incorporate multiple different systems there will be a need for multiple voltage regulators to compensate for the different voltage requirements. The ESP32 requires 3.3 volts and about 95 milliamps to 240 milliamps to operate, as well as the display, load cell, and ADC. The thermal printer on the other hand, can operate at a range of 5 to 9 volts. The ideal choice for a voltage regulator will be the LM2576. The LM2576's are available in fixed output voltages of 3.3 volts, 5 volts, and an adjustable output voltage version, so all of our system's voltages can be accommodated by this device. The LM2576 is also capable of driving 3 amps, which is well within the requirement. Three of these devices incorporated into our design will ensure that the PUMPED will work properly and since this regulator is a switching regulator there will be high efficiency in power usage.

3.1.9 PCB Software Applications

3.1.9.1 Different Types of Software's for Designing PCB

There are a lot of PCB software design tools that exist in the real world, many are free to use, and many require a membership purchase which usually costs a lot of money each month. When trying to choose the right software tool that will allow us to successfully and easily design our PCB, we must set up a list of the requirements that we want to fulfill when choosing. If we were to write about all of the PCB software design tools that exist, this chapter would be multiple pages long. What we decided to do is we picked 3 out of the rest that seemed to align to our requirements and we will talk about their pros, cons, and compare them with one another. The main key details that we are looking for in our PCB design software are for it to be straightforward and not complicated to use, have the most up-to-date standards and details, and most importantly be of low cost. Because PUMPED is a sponsor project, we are limited in our spending due to our budget, so we have to be aware of our expenses and make smart purchases.

3.1.9.2 Eagle CAD

The first PCB software design that we chose for a potential option to use is EAGLE CAD. The main reason as to why this software is one of our options is because we have a lot of experience using this software because this was the software that we used in our junior design project when we were designing our first PCB in our curriculum. EAGLE CAD used to be its own individual software but is now part of Autodesk Fusion. It is \$85 a month but we are able to acquire the license for it and use it at no cost when using our UCF email. This software is an electronic design automation software that allows the user to seamlessly connect schematic diagrams, component placement, PCB routing, and contains comprehensive library content. There are many pros to this software such as being great for designing small to medium size PCB designs, it has extensive library support, Fusion 360 integration 3D modeling, and is cross platform supported. Some cons are that you are

required to purchase Autodesk fusion because it is not offered alone anymore, and the software's auto router is not as powerful as other PCB designers. This software is also capable of exporting BOM and Gerber files so that you can use an ordering website such as JLCPCB to order your PCB design fast and easily.

3.1.9.3 KiCAD

KiCad is another PCB software design that is a cross platform and open-source electronic design automation suite. This is a free software for everyone that is used by people casually and professionally. Kicad schematic editor supports everything from the most basic schematics to the most complex. The supported library is very extensive, which allows you to create your own custom symbols or use thousands of existing ones. It has integrated SPICE simulator, the layout of the program is easy to use and intuitive, it contains improved visualization and selection of tools, and the most important key aspect of this program is that it contains a 3D viewer that allows you to easily inspect your design from all angles. This software contains automatic and manual trace routing for PCB layout support, it has a built in electrical and design rule check that helps catch design errors at an early stage, it supports BOM and Gerber file exports like EAGLE CAD, it is multi-platform supported, and lastly it has no board size limits for a free PCB software design tool. This is a great PCB software design tool that we can use for our PUMPED project.

3.1.9.4 Altium

Altium is a high-end PCB design software used mainly by professionals and industrial applications. This software is very powerful software that is the main choice for professionals because it is fast, easy to use, and has a lot of support. It is capable of being integrated with other engineering tools, it has seamless workflow, it has a powerful schematic editor that has a real time rule checking. It has Advanced routing, multi-layer design, high speed signal integrity analysis for its PCB layout. It also supports 3D visualization like the previous listed CAD, it supports the export of many different types of files such as Gerber files, ODB++, and many more.

The main benefit of this software is that it can support collaboration and cloud integration which comes from Altium 365 for real time teamwork collaboration which is in a way like GitHub. Those are all benefits to this software and does in fact contain everything that someone would need to design a PCB and more. The main negative aspect of this software is that it is very expensive to purchase, costing thousands of dollars for group collaboration, it requires powerful computers for designing large PCBs, and has a steep learning curve for beginners. The good thing is that we messaged Altium's customer service, and we were able to obtain some information that was very beneficial. We were informed that we could get an Altium license for a time by using our UCF email. After hearing that information, we quickly jumped to it and obtained Altium.

3.1.9.5 PCB Software Design of Choice

There were many key points that determined what PCB software tool we were going to use. For us to efficiently choose the right software for our PUMPED PCB design, we decided to make a table that contained information on the three software's comparing the pros and cons of each.

Table 28 Selecting PCB Software

Feature	EAGLE CAD	KiCad	Altium
Ease of Use	User-friendly for small to medium PCBs	Intuitive layout, good for all levels	Steep learning curve, best for professionals
Library Support	Extensive component libraries	Large library with custom symbol support	Comprehensive library & cloud integration
3D Modeling	Integrated with Fusion 360	Built-in 3D viewer	Advanced 3D visualization
Routing Capabilities	Integrated with Fusion 360	Manual and automatic routing	Advanced routing with high-speed signal integrity
Design Rule Checks	Basic rule checking	Built-in electrical & design rule checks	Real-time rule checking
Collaboration	Limited collaboration	No cloud collaboration	Altium 365 supports real-time teamwork
Export Options	Supports BOM & Gerber file export	Supports BOM & Gerber file export	Supports Gerber, ODB++, and more

The best software that we decided that we are going to use for our PUMPED project is going to be Altium. The main reason for that is because we were able to get it for free using our UCF email. Doing so will allow us to have all the tools and benefits that are up to standard in one single program.

3.1.10 Software Tools and Environments

Due to our need for a significant amount of embedded programming and other software development for this project it is important we select the appropriate tools that make a complicated project as simple as possible. A key factor with selecting the tools that are going to support our software development is a shared knowledge or understanding of the tools being used across the team so that anyone can reference our code and understand it.

In addition to having a sense of familiarity with our tools though we also need to ensure that they are the ones that provide us with the most straightforward path to turning a desired logical flow into working code. In this case it should be prioritized to a certain extent that the best tool is selected if it is far and away the best to use, even if there is a reduced shared knowledge base among the group or a lack of familiarity. The aforementioned factors apply to everything from where our code is stored, the coding language, the environments we write the code in, and any frameworks or libraries we insert into the project. The areas in which we need to consider for our tools and environments are:

- Version control systems
- Integrated Development Environments
- Embedded frameworks

3.1.10.1 GitLab

Before any code is written it is important to establish a way for the team's developers to collaborate and store our code in a shared environment. Additionally, it is paramount that a selected storage environment enables us to track changes to our code over time and

potentially revert back to older versions if necessary. Perhaps the most popular way to do this is to use Git to keep track of changes and more specifically use an online platform that keeps a remote repository for this Git data. There are several such online platforms and one of them is GitLab. GitLab is a platform that can host our code and help us track changes or potential issues in the code over time. The platform is straightforward, but the primary drawback is that no developers in our group have experience with using the platform. In order to prioritize ease of development it would likely be preferable to go with a platform the entire group is somewhat familiar with.

3.1.10.2 GitHub

In order to meet the needs for our coding collaboration it is likely that the git-based platform GitHub is going to be ideal for us. GitHub allows us to store our code and any other supporting files while also maintaining a file revision history. The file revision history is a key reason for using GitHub as we are able to easily revert to an older state if necessary, on their platform. The biggest supporting reason for selecting GitHub as our code storage solution is that the entire team is either somewhat or very familiar with GitHub and its features.

3.1.10.3 Visual Studio Code

Visual Studio Code (VS Code) is a powerful and well supported integrated development environment that is capable of facilitating the development of all kinds of programs. For a project such as this one that requires some embedded programming on a microcontroller VS Code is a valid choice that has various benefits. The foremost benefit and reason to use Visual Studio Code for embedded programming on this project is that it is compatible with ESP-IDF which is the officially supported development framework for developing programs on the ESP32 microcontroller. If we end up choosing to use an ESP32 microcontroller for this project it would be a logical pairing to do our embedded development in Visual Studio Code.

Another benefit of using Visual Studio Code for this project is that it is an integrated development environment that the developers on this project are familiar with making development more efficient. In VS Code is well supported by a large degree of plugins that could make development more streamlined. Overall Visual Studio Code is a fairly easy to navigate integrated development environment compared to some other choices of integrated development environments.

3.1.10.4 Eclipse

Eclipse is a well-known integrated development environment that can support the development of many software projects. Eclipse, like VS Code, is also compatible with ESP-IDF which is the officially supported development framework for developing programs on the ESP32 microcontroller. If we do decide to use the ESP32 microcontroller for this project it would be a strong pairing. Some drawbacks of using Eclipse for this project is that the developers on this project are less familiar with it compared to other integrated development environments. The main drawback with using Eclipse is that it is not as widely supported as other integrated development environments and therefore has less plugins to support development and the development environment itself. Additionally, Eclipse is not as lightweight as some other integrated development environments which has its own set of benefits and drawbacks. Ultimately Eclipse is a valid potential choice of

integrated development environment but does lack the customization that many others do have.

3.1.10.5 Arduino IDE

The Arduino integrated development environment is a very easy to use integrated development environment that is welcoming to beginners. While the Arduino integrated development environment may be easy to use it is not particularly powerful and very limited in the types of projects it can support. The Arduino integrated development environment is well suited to very small and simple projects, not necessarily larger and more complex projects like this one. It has limited support for customization for various things such as debugging and lower level support for many types of microcontrollers. Overall the Arduino IDE is very limited in its usefulness and depending on how complex the project becomes it could potentially fall short of what we need.

Table 29 Selecting IDE

Embedded IDE	Resource Usage (RAM)	Resource usage (CPU usage)	Familiarity	Customizable	ESP-IDF support
Arduino	200 – 400 MB	10 – 30%	Low	Low	No
Eclipse	1 – 2 GB	50 – 90%	Moderate	Moderate	Yes
Visual Studio Code	300 – 600 MB	20 – 50%	High	High	Yes

3.1.10.6 PlatformIO

PlatformIO is essentially an open-source integrated development environment built on top of the existing VS Code integrated development environment. It facilitates embedded development for a variety of microcontrollers. The selling point of PlatformIO is that it automatically manages libraries and dependencies for you theoretically making development more streamlined. A drawback is that as developers we lack any familiarity with PlatformIO and how to configure it meaning there will be a degree of setup time before we can even begin programming. Another drawback if we decide to use an ESP32 microcontroller is that while PlatformIO is technically compatible with ESP-IDF there is some lag time and it may not be fully up to date depending on what we are trying to do. At the end of the day PlatformIO is a potential choice for our embedded programming integrated development environment but there are potential issues at hand.

3.1.10.7 ESP-IDF

ESP-IDF is the officially supported development framework for developing programs on the ESP32 microcontroller. It can be used with the integrated development environments VS Code or Eclipse. If we decided to select an ESP32 microcontroller for this project using ESP-IDF would be a very powerful control because it allows for full control of the microcontroller at a low level. This would be an important aspect as it would also help us manage the power consumption of the microcontroller, which is an area of concern. A drawback is that there is some set up required to make this workflow possible, but it could be worth it considering the usefulness of the framework. Furthermore, as developers we

lack familiarity with ESP-IDF, but it may indeed be worth taking time to learn it to unlock the full power of our microcontroller.

3.1.11 Android Application Technologies and Frameworks

A significant portion of this project revolves around the use of an associated app with our device that receives the breast milk bag data and stores it. Aside from just storing the data we want to create a user-friendly interface that displays the data collected regarding the bags in a sophisticated manner, with the option to also show statistical trends regarding the collected data. In addition to the statistics themselves we want to keep track of our breast milk bag inventory and have the capability to send notifications regarding the current state of the inventory. This app will have several complicated components that make it up including: MCU communication, a backend for storing data, app development framework for the interface and interaction with the backend, and likely many more supporting frameworks or technologies to facilitate the success and reliability of the aforementioned technologies.

3.1.11.1 HTTP REST API

One of the two methods for communication between the microcontroller we end up using and the eventual mobile app is an HTTP REST API. If we use it the flow of information will likely be: the weight sensor detects and weighs an item, the microcontroller attached to the weight sensor then sends the weight data to a backend server/database, the data is processed on the backend, and then finally the mobile app that is periodically polling the backend for new weight data finds and receives it onto the app side. One notable step in this process is the “periodic polling.” While there are possible ways around it depending on how we configure things on the backend, generally speaking the HTTP REST API may not be ideal for tracking real time changes.

That is to say there could be a possible delay (in the range of seconds) in the time it takes from the weighing of a bag to that bag and weight data showing up on the user's mobile application. This delay might not be too great and could even be deemed acceptable, but it is possible that nonetheless it could end up being noticeable depending on how well backend implementation goes. Additionally for an integrated device such as the one we are planning to build HTTP communication protocols are generally speaking not the ideal match although they are certainly compatible and due to HTTPs ubiquitous nature a developer might be more familiar with it than other potential protocols. The other drawback though if a polling method is used, as is typical with an HTTP REST API, is that there would be higher usage with the constant polling making it inefficient if it keeps polling without anything to receive. Ultimately an HTTP REST API would likely be one of the simpler, if not the simplest protocols to implement but it comes at the cost of flexibility and inefficiency.

3.1.11.2 Constrained Application Protocol (CoAP)

Constrained Application protocol is an especially optimized communication protocol that is intended to be used with integrated devices that have resource constraint concerns. It shares some similarities with HTTP REST communication methods, but its efficiency mainly comes from its use of UDP. UDP is more efficient than TCP which is used by most other protocols like MQTT and HTML. While UDP is efficient in the sense that data can be transmitted faster it is not as reliable due to its connectionless nature at the packet level.

The other main drawback is that there is far less support for the use of CoAP with additional libraries for this sort of project so it would require a lot more effort and development time to implement.

3.1.11.3 MQTT

The second method we could use for communication between the microcontroller in our integrated device and the mobile application associated with it is Messaging Queuing Telemetry Transport (MQTT). This method uses a publisher/subscriber model and the flow of information would likely look like: the microcontroller would be setup to be connected to an MQTT broker, the weight sensor detects and weighs an item then sends and publishes the weight data to a topic, the mobile app which is configured as a subscriber to the aforementioned topic has the data sent to it by the MQTT broker. The important thing to note from the aforementioned process is that as soon as the broker receives the weight data it is sending that data to the mobile application in real-time. This is efficient because there is no constant polling in the publisher/subscriber method so that makes MQTT an efficient protocol compared to others. MQTT has a fair latency and overhead use that makes the protocol relatively efficient. Compared to HTTP rest for example MQTT is much more lightweight.

This makes sense as MQTT in general is known as a lightweight communication protocol. A lightweight communication protocol would be well suited to this type of project where real time updates are needed but we don't want constant polling. Generally speaking, MQTT is the preferred communication protocol for an integrated, internet of things adjacent device such as the one we are creating. The main drawback with the MQTT protocol is that it requires an extra degree of setup and complexity, but it may well be worth it due to the protocol being more efficient and more suited to the needs of this device.

3.1.11.4 Bluetooth Low Energy (BLE)

A possible method that could be used for communication between the microcontroller (and its associated sensors) and the mobile app is Bluetooth Low Energy (BLE). BLE is a version of Bluetooth that is known for its low power consumption which makes it ideal for IoT devices. With BLE there would be a client (mobile app) and server (ESP32). With BLE there is minimal setup required by the user as the client will scan for the connection to the server and connect directly automatically. This enhances the portability of the device as there is no inter connection required and it is easy to use since a user does not have to enter any Wi-Fi credentials. It is a lightweight, low latency protocol that can facilitate real-time updates like we need. Some drawbacks of using BLE include its limited range since it is dependent on the range between the phone and the device. Another drawback is that only one phone can be connected to our microcontroller at a time, but this may not be too much of an issue considering it was part of our design anyways. Overall BLE is a viable choice and well suited to IoT devices such as the one we are creating.

Table 30 Wireless Communication Protocols

Communication Protocol	Packet overhead	Latency	Real-time updates

HTTP REST	800 - 1500 bytes	100 – 500 ms	No*
CoAP	10 – 50 bytes	20 – 200 ms	Yes
MQTT	20 – 100 bytes	50 – 300 ms	Yes

3.1.11.5 Node.js

This is one of two environments/platforms we can consider using for the backend for this android mobile application. It is important to note that Node.js is almost always paired with the Express.js framework to simplify things by on the backend by providing more built-in features. The main benefit of using Node.js for our backend is that we have full flexibility to build our backend, however we want and choose to integrate it with any database of our choosing. The obvious drawback of course is that we are then responsible for developing and setting up the entirety of the backends: servers, databases, and the APIs to connect it all. Furthermore, there could be additional setup and complexity depending on if we want real-time updates from the weighing to info being sent to the app. Overall Node.js as our backend offers unmatched flexibility but it may come at the cost of significant development overhead and responsibility with setting up every step of our backend (self-hosting servers, database, API, etc.).

3.1.11.6 Ktor

Ktor is a lightweight framework that can be used to build the backend and interestingly it is built in Kotlin so there is some potential synergy if we choose to build our app with Kotlin as our language of choice. Ktor is a modern framework and therefore has a lot of quality-of-life benefits with regards to its ease of use with Kotlin. The main issue with Ktor is that its an immature framework and therefore lacks a lot of support. Furthermore, its performance is not ideal when compared to more popular frameworks such as Node.js. Additionally, its modern nature means there is less information out there for unfamiliar developers to utilize to learn the framework which can put a negative impact on the development time.

3.1.11.7 Firebase

Firebase is the second platform we can consider using for the backend of this android mobile application. The main benefit of using Firebase is that there is little setup needed with Firebase being a managed backend service so we would not have to setup any servers, databases, etc. For our android app Firebase can handle everything from the database itself to sending notifications to the android mobile application. Real-time updating, which we may end up needing for this device and its mobile application, is built-in to Firebase with its Realtime database. Firebase is well suited to be used to build the backend for an android mobile application as both our Google products meant to be used together. The two main drawbacks for the use of Firebase are: cost and lack of flexibility. Firebase has a free tier that can be utilized to a good extent for a relatively lightweight app such as this, but it should be ensured that the backend usage either falls within that free tier or not too far over to avoid incurring overly significant costs. The other drawback is the lack of flexibility

since we are using a backend service that we do not have a full degree of control over. Ultimately Firebase has strong synergy of android mobile applications and offers perhaps the simplest implementation of the backend while coming at the cost of any flexibility.

Table 31 Firebase Information

Backend	Built-in Android Studio Compatibility	Previous Experience	Cost
Node.js + Express.js	No	Yes	Self-hosted
Ktor	No	No	Self-hosted
Firebase	Yes	No	Free* (under certain level of usage)

3.1.11.8 Visual Studio Code

While Visual Studio Code is not the official or preferred integrated development environment for Android mobile applications it has some benefits. The main upside to Visual Studio Code over Android Studio is that it is much less resource intensive. This is something to take note of if for some reason development must be done on a particularly low-end system. Visual Studio Code also has some supporting libraries that can be used in this project for potential communication protocols. Visual Studio Code has significant drawbacks with its lack of official Android support, so it completely lacks things like emulation of the app we are building on a variety of Android phones or versions. This is important to note because we do not have a variety of Android phones in or possession nor the funding to acquire them. Overall, the experience of using Visual Studio Code for android mobile application development will likely not be particularly seamless. The main use case for Visual Studio Code in this project would be any potential development of backend APIs.

3.1.11.9 IntelliJ IDEA

As an integrated development environment IntelliJ is a very powerful tool. IntelliJ has lots of built in tools for various things such as version control and API testing. It has a wide variety of plugins that give additional support for almost anything you need. The main issue with IntelliJ is that it lacks android specific tools that other integrated development environments have. This includes the IntelliJ integrated development environment not having an Android emulator. Not have an android emulator would mean that we would have to have physical access to a massive number of Androids running different versions, and that is not something we have the funding for in this project. Another possible issue is that the integrated development environment is quite resource intensive.

3.1.11.10 Android Studio

The likely development environment for this android mobile application is Android Studio. Android Studio is the official integrated development environment for the Android OS. This is important as it means there will be a plethora of supporting libraries and helpful information available to us as we are building the mobile application. For example, Android Studio has UI libraries that will allow us to display the bag and weight data in sophisticated charts or inventories. The other big benefit of using Android Studio to develop an Android mobile application is that it has an emulator that allows us to test the app as we are building it on multiple types of Android devices without even having to physically possess them. In addition, it is capable of simulating sensor data which is exactly what we needed developing a mobile app for this integrated device. Furthermore, the integrated development environment is supported by Google and has support for integrating very seamlessly into Firebase if we decide to use that for our backend. The only real drawback to using Android Studio is a lack of experience among our developers with using it but this is something that is likely outweighed by the amount of support this integrated development has.

Table 32 Android Studio

Mobile IDE	Resource usage (CPU)	Resource usage (RAM)	Familiarity	Cross-platform	Official support
Visual Studio Code	20 – 50%	300 – 600 MB	High	Yes	No
IntelliJ	40 – 70%	800 MB – 1.5 GB	Low	Yes	No
Android Studio	70 – 90%	1.2 - 2.5 GB	Low	No	Yes

3.1.11.1 Java

Java is more of a legacy language in android mobile application development, so it is good to use with older Android projects. Its longtime use means that there are lot of resources available to assist with development issues but some of these may be increasingly outdated. The biggest benefit of using Java is that like many developers the developers on this project are already familiar with the language. While Java does offer some simplicity thanks to us already being familiar with it that may not be enough to overcome the fact that it is no longer the preferred language for android mobile app development and use with Android Studio.

3.1.11.11 Dart

Dart is another possible choice of language to develop this android mobile application with due to its modern nature. Dart has all the features you would expect from a modern language and additionally has strong support in the form of a supporting ecosystem with helpful libraries. The main drawback with Dart is that its primarily limited to use with Flutter, an opensource framework for mobile application development, and so if we are not using Flutter than the usefulness of Dart will be reduced significantly. Dart is at its most

useful when writing cross-platform mobile applications, but we are more centered around writing code for Android applications. In this case Dart may not be as practical to use if we are only going to develop a mobile application for Android.

3.1.11.12 Kotlin

Kotlin is quickly becoming the new standard for android mobile application development. Kotlin is generally speaking the modern choice for mobile app development and has quick development thanks to its more concise syntax. Kotlin's modern nature is displayed by things like its null safety where it finds null related issues at compile time instead of run time. Since Kotlin is the preferred language in the Android Studio ecosystem it has the best support and compatibility with all of Android Studio's libraries for mobile app development. The obvious drawback with Kotlin for our use on this project is that none of us are familiar with Kotlin but simply learning it may make things easier for us in the long run.

Table 33 Selecting Programming Language

Language	Memory Usage (avg)	Familiarity	Overall Support	Null Safety
Java	80 – 150 MB	Yes	Some support	No
Dart	50 - 120 MB	No	Some support	Yes
Kotlin	70 – 130 MB	No	Well supported	Yes

3.1.12 Embedded Thermal Printers

For this project an embedded thermal printer will likely be ideal compared to various other types of printers. It is important that we identify the right embedded thermal printer that meets our specifications as in some ways this printer is the engine of this overall project. The ideal embedded thermal printer will have to meet specifications with regards to size, operating voltage, operating current, compatibility with MCU, baud rate, communication type, and more.

3.1.12.1 DFRobot GY-EP204x

The DFRobot embedded thermal printer was considered for this project. This printer has the capability to print onto both receipt style thermal paper and sticky labels. It utilizes a RS232 or TTL serial connection with an associated microcontroller. It is a fairly compact printer and printed at a width of 48mm with a thermal paper roll. The most important

attribute is that it has an operating voltage range of 9-24V drawing a current from 0.5 - 2.5 amps. We found that in practice though that this printer operates ideally with the best print quality at a voltage of 20 volts. This is a much higher operating voltage than we would like to use in order to avoid unnecessary power consumption in the overall device. The upside of using this much voltage was that the print speed was fairly fast, but it does not seem to be worth the tradeoff of power consumption.

3.1.12.2 Maikrt MC-EH300

The MC-EH300 embedded thermal printer is a printer under consideration for use in our IoT device. This printer is compact and prints at a width of 58mm on a thermal paper roll. It connects to the microcontroller using a RS232 or TTL serial connection. The key aspect of this printer that disqualifies it from use in our project is that it has an operating voltage range of 9-24V with it only delivering high quality printers at the higher range. We want to conserve the amount of power we are consuming with our printer since it is going to be the most demanding component from a power perspective. If our operating voltage is too high it will result in us having reduced battery life or having to compensate by buying a larger battery that could reduce the overall portability of our device

3.1.12.3 Maikrt MC206H

The MC206H embedded thermal printer is a viable printer that would be a good choice for this project. The printer utilizes a RS232 or TTL serial connection with an associated microcontroller, such as the ESP32. It can print onto receipt style thermal paper or sticky labels. It is a compact printer printing at a width of 58mm with a thermal paper roll. The most important part is that the printer has an operating voltage range of 5-9v and draws a current of about 2 amps. This is a good operating voltage for us as it delivers high quality prints with mild power consumption which will help prolong the battery life of our overall device.

Table 34 Maikrt MC206H Information

Thermal Printer	Size	Operating Voltage	Operating Current	Baud Rate
GY-EP204x	77 mm x 53 mm x 42 mm	9 - 24 V	1.5 amps	115200
MC-EH300	75 mm x 49 mm x 49 mm	9 – 24 V	1.5 amps	9600
MC206H	81 mm x 45 mm x 56 mm	5 – 9 V	2 amps	9600

3.2 Part Selection

There were many different parts that were an option to be used that would have allowed us to create our project. Our research has helped us be able to obtain the best parts and software applications that will allow us to successfully create our project without cutting any corners, and without overspending. This was a very tedious and lengthy chapter because there are an extreme number of different components that are out there that can serve a similar purpose, but we did all our research and compared most of the parts that we thought would be adequate to be included in our project. After comparing them, we came up with the following parts and software programs that we used in our PUMPED senior design project:

Table 35 Project Part Selection

Name	Description	Usage
Microcontroller: ESP32-Wroom-32	• Processor: Xtensa LX6 (Dual-core) • Clock Speed: Up to 240 MHz • Memory: 4MB Flash / 529 KB SRAM • GPIO Pins: 25 • Interfaces: 4x SPI, 2x I2C, 3x UART • Connectivity: Wi-Fi, Bluetooth 4.2 • Operating Voltage: 3.3V • Price: Under \$15 • Key Features: Powerful, affordable, ideal for IoT applications	This is the MCU that will allow us to sync and use everything together, without it and its capabilities, the device will not work in all aspects, and will not be able to be paired with our mobile app.
ROMOSS PHP30 Pro	• 111 Wh Capacity • Multiple Output ports capable of delivering current of up to 3 Amps at 5V • Rechargeable	This battery bank has a high capacity capable of powering the PUMPED system for days at a time. It also has the power delivery capabilities of powering all our subsystems without issue.
Weighing Technology: 10kg strain gauge load cell	• Accurate up to two decimal places • Cheapest compared to alternatives • Stable over time • Signal conditioning via HX711	This load cell will be sufficient for our needs due to it being easily integrated into the entire system.
ADC: HX711	• 24-bit resolution • Made for the 10kg strain gauge load cell • Price: ~\$2.00	This ADC will work nicely in our system because of its accuracy and ease of implementation
Display: HiLetgo SSD 1309	• Display Type: OLED	The display is going to be an average-sized screen that will give us the

	• Screen Size: 2.42 inches • Resolution: 128 x 64 pixels • Operating Voltage: 3V – 5V • Interface Options: I2C or SPI (both supported) • Price: < \$20	capability to display date, time, and weight in ounces and can possibly allow us to add more. It is cheap and by far the best option for us to go with.
User Interface: Buttons	• Reliable Input: Less prone to misreads compared to touchscreens. • Works in Harsh Environments: Function well with wet hands, or in dirty/dusty conditions. • Low Power Consumption: They don't need constant sensing. • Simple and Intuitive: Familiar and easy to use.	Using physical buttons will make the overall feel of our device easy to use and easy to troubleshoot in case of any problems. Each button that is on the device will serve its own individual role.
Voltage Regulation: LM2576: 3.3V	• Highly efficient since it is a switching (buck) regulator • Equipped with voltage and current protection • Price: ~\$5.00	This regulator is sufficient for our system. The 3.3V is the typical voltage requirements for the other components

Chapter 4: Standards and Design Constraints

This chapter will touch on key industrial standards and design constraints that will shape the design and development of our PUMPED project. This chapter begins by addressing critical industrial standards such as PCB design standards that will ensure our prototype meets all of the technical and safety requirements. The chapter then shifts to the practical constraints that will determine all the design decisions starting with the economic constraint, since we are working with a strict budget of \$500, we are forced to prioritize essential components that will satisfy all our requirements within a low cost. Additionally, we considered environmental factors, such as energy consumption and waste management, alongside ethical concerns regarding user safety and product reliability. Manufacturability challenges, sustainability requirements, health and safety protocols, and time management pressures further affect our approach, while political constraints remind us to respect regional sourcing and sponsor expectations. Together, these standards and constraints provide a comprehensive framework that guides our decision-making process throughout the project.

4.1 Industrial Standards

4.1.1 PCB Design Standards

For our PUMPED device we designed a PCB to interconnect all our components which will require the PCB to be designed in accordance with industry standards to ensure safety,

proper functionality, and manufacturability. To do this, we followed ICP standards that apply to our device.

4.1.1.1 IPC Standards

IPC 2221: Establishes some generic design and performance requirements in PCBs. This includes requirements on PCB trace width, spacing, thermal relief, and via placement requirements

IPC 7351: Specifies footprint and land pattern requirements for SMT components. This ensures that our design's solder pad geometries meet manufacturing and soldering process requirements.

IPC-A-600: Provides criteria for physical and cosmetic acceptability during inspection and quality control

IPC-A-610: Ensures acceptability of electronic assemblies, which provide criteria for the workmanship quality of electronic assemblies.

Following these standards will ensure our PCB can be produced correctly.

To go more in depth within the standards required we will discuss some important considerations to take into account. When designing our PCB, we have to make sure our different subsystems within our device have electromagnetic compatibility. We have both analog and digital components, so we needed to minimize electromagnetic interference. The noise that is produced by the switching regulators is an important aspect to keep in mind when placing components on the PCB because it can interfere with noise sensitive components such as the Wi-Fi/Bluetooth antenna mounted on the MCU. Electrostatic discharge protection also needs to be considered since there will be user interactive components such as buttons and USB interfaces, so protection is important. We needed to have effective grounding planes to safely discharge static buildup. This will improve device reliability and protect ICs from damage. Safety and Material standards will help prevent unexpected voltage production from the battery so input capacitors and voltage regulators will help in damping possible overvoltage.

4.1.2 Battery and Power Standards

Our pumped device will be powered by a rechargeable battery bank, so there must be adherence to battery safety and power management standards. This will ensure safe operation, charging, and storage for the battery system. Standards to follow include:

IEC 62133: This is an international standard that outlines safety requirements for portable sealed secondary cells and batteries, particularly lithium-ion cells. It addresses issues such as overcharging, thermal protection, short-circuit mitigation, and mechanical abuse. While PUMPED does not directly design the battery pack, the selected power bank is chosen to be compliant with this standard to ensure safe and reliable performance during everyday use.

UL 2054: This standard provides additional guidance for household and commercial battery systems, including enclosures, protective circuitry, and safety-critical fault

conditions. This standard goes along with IEC 62133 by addressing use case specific risks, particularly relevant given the user-facing and portable nature of PUMPED.

4.1.3 Wireless Communication Standards

The Pumped device will make use of wireless communication through the Wi-Fi/Bluetooth capabilities of the ESP32 microcontroller to connect with a mobile app that will allow real time data transfer and device control. This will require the need to comply with radio frequency regulations to ensure interference free and the ability to be in the vicinity of other wireless technologies.

FCC Part 15, Subpart C: In the United States, unlicensed radio-frequency devices must comply with FCC Part 15 regulations. The ESP32 module used in PUMPED is FCC-certified under Subpart C, which governs intentional radiators such as Bluetooth transmitters. This ensures that the device operates within allowable emission limits and does not cause harmful interference.

ETSI EN 300 328: For broader global compliance this standard ensures proper operation of data transmission equipment operating in the 2.4 GHz band, including Bluetooth. It covers spectral efficiency, power control, and adaptive frequency hopping all of which are supported by the ESP32.

4.1.4 User Safety Standards

As a portable, user-facing device designed to assist parents in weighing and labeling breast milk, our PUMPED design must prioritize user safety and ergonomic interaction. The design incorporates principles from established human centered and risk-based design standards to reduce user error, enhance comfort, and ensure safe operation.

ISO 9241-210: Is the Ergonomics of Human-System Interaction standard that outlines principles of human-centered design for interactive systems, emphasizing usability, accessibility, and intuitive operation. PUMPED will integrate a simple button-based interface with clear labeling, a readable display, and structured workflows to ensure users can operate the device effectively without requiring technical knowledge. The mobile app interface will also be guided by these principles to enhance accessibility and reduce cognitive load during operation.

ISO 14971: This medical device standard could apply to our device since we are intending it for use in medical establishments, so the principles in this standard can be used to identify and mitigate potential hazards such as mislabeling, weighing inaccuracies, or improper storage instructions. This includes built-in prompts, calibration validation, and label clarity to prevent data misinterpretation or user mistakes.

4.1.5 Environmental Standards

To align with modern engineering practices, the PUMPED device will integrate environmental and sustainability considerations throughout its lifecycle. So, standards that deal with electronic waste will need to be accounted for.

RoHS (Restriction of Hazardous Substances Directive): All electronic components and PCB materials used in the PUMPED device will be selected to be RoHS-compliant, ensuring the exclusion of hazardous substances such as lead, cadmium, mercury, and hexavalent chromium. This promotes safer manufacturing, use, and disposal practices, particularly important for a device intended to interact with child-care products.

4.2 Practical Constraints

4.2.1 Economic

The primary constraint that this project is facing is an economic constraint. We have a budget of just \$500 to make the prototype for this integrated device and its associated mobile application. This budget is going to force us to make decisions at various junctures between performance or cost savings.

When we purchased components, we had to decide which components are a higher priority and make an effort to spend the limited money we had on those first and foremost. Additionally, we had to find the components that could be purchased at the lowest cost while not adversely affecting the quality of our prototype to a significant degree.

We also had to take note of which components are the most likely to experience issues or otherwise are so important that we can't afford for them to have issues. An example of this in our project may be the printer itself in our integrated device. For the printer, and any other components of its same level of importance, we had to consider buying backup parts for them as well. This is going to further put pressure on us to make calculated choices with regard to our limited budget of \$500.

In our evaluation we decided the printer was one of the highest priority items, if not the highest priority, and so we bought multiple printers. This decision was made ahead of time because if we end up experiencing breakage and need to suddenly get another printer the expedited shipping costs could be extreme, especially if the shipping is international. To maintain our budget to the best of our ability, we have decided to just buy multiple printers ahead of time so that we are not possibly adversely affected by such an issue with having to deal with expedited shipping costs.

Another notable economic impact of the printer is that it has a maintenance cost in the form of having to buy thermal paper. This cost of maintenance is going to impact us in the testing phase where our budget could be impacted by having to keep buying thermal paper depending on how intensive our testing is. This cost of maintenance is also going to impact the end user who is going to have to also keep buying thermal paper for themselves in order to keep using the product.

4.2.2 Environmental

This integrated device and its associated mobile application will be used ideally in a variety of locations. It's important to take this into consideration as we try to assess any possible environmental impacts this prototype product can have. This could come in the form of any waste this product produces, its energy usage, or any other relevant factors.

Something to take into consideration with the possible impact our product has on its external environments is the thermal paper waste we are producing. The device will print out bag details on thermal paper and it will be useful for a period of days, weeks, or months

until it is used. Once the bag label is no longer useful it is at risk of becoming waste or litter in its external environment. It is important to see if there is a way we can encourage end users to properly recycle the thermal paper and ensure the thermal paper itself is compatible with the recycling process.

An obvious point of concern with any device like this is the energy usage. Energy usage for this device will be significant as it will have a printer integrated into it which will dramatically raise the baseline of energy consumption. Additionally, our selected microcontroller and how we use it can present large energy usage concerns. It is important we take steps to ensure we are using our microcontroller in an efficient manner and in a way that is not wasting energy.

With regards to our microcontroller, it is important that we take steps to make sure that it is not on any more than it needs to be. Furthermore, many microcontrollers have low power modes that can be enabled at certain points that we can take advantage of. The microcontroller energy usage can be further optimized even down to the communication protocol we select for handling data transfers between the microcontroller and the android mobile application.

Some communication protocols use inefficient methods like polling which regularly checks for updates nonstop while other protocols, such as MQTT, use a publisher-subscriber model that only sends and receives data when it is necessary. These choices and more with regards to what we do with our microcontroller can have a positive impact on reducing the environmental impact of our device via curbing energy usage.

4.2.3 Ethical Constraints

When designing this device, certain ethical considerations must be taken into account so that we can make sure that the system is safe, reliable, and accessible to its users. We must also make sure that we minimize any potential harm to the users. One of our main concerns with safety for this project revolves around the battery that the device will use. Since the device must be portable and rechargeable, there are safety risks associated with a potential battery failure due to overcharging or overheating.

A robust battery charging system must be implemented in the case that we use batteries susceptible to explosion or creating a fire such as Lithium-Ion or Lithium-Polymer batteries. Another concern for safety revolving around the battery is that the substances that some batteries are made of could be toxic. We must avoid the use of hazardous chemicals such as lead and acid to ensure the users' safety.

Another ethical concern is that our device must be reliable and accurate at measuring the breast milk. The PUMPED system must provide accurate weight measurements to prevent mislabeling. Inaccurate readings could lead mothers to overfeed or underfeed their babies, or it could potentially lead to unnecessary waste of breast milk.

The app must also be free of these inaccuracies. Any errors in the time keeping of these milk bags could lead to mothers discarding usable milk, or even worse, feeding expired milk to their child. Expired milk could potentially lead to sickness and pose health risks so, we must ensure our system is accurate and reliable for the safety of the user's child.

Finally, while the device that we are making will only be a prototype, we can also talk about the ethical concern of affordability. Since the PUMPED system will be targeted at

mothers; unnecessary costs should be avoided so that the device can be accessible by lower income users.

4.2.4 Manufacturability Constraints

The PUMPED system is being developed as a senior design prototype rather than a large-scale product that must be mass produced. However, there are still some prototype manufacturability constraints that can impact our development and our project. This includes things like the sourcing of our components, the complexity of assembling everything, lead time on PCB deliveries, and things like lab equipment accessibility.

In the case of component availability and sourcing parts, since this is a prototype, we did not get bulk discounts because we did not buy things in bulk. Moreover, lead times when ordering components could delay our project and impact our timelines.

When it comes to PCB designs, we designed our own custom PCBs and will need to account for delivery times. If a PCB fails to function, this could cause us to have a redesign and delay our project even more. Not only that, but the components will also not be soldered onto the PCB using automated manufacturing processes. We manually soldered these components which substantially increased the amount of time needed for each PCB.

In the case of the device itself, we have 3D printed the enclosure, making sure that all the components fit inside of it in a compact fashion to ensure maximum portability. If we were to be going to mass manufacturing, it would be much better to go with some sort of injection mold for faster and more reliable parts.

One of the main manufacturing constraints this project has is the limited budget of \$500. This means that more expensive, high precision parts are not an option, and repairs or PCB redesigns must be kept to a minimum to reduce unnecessary costs.

4.2.5 Sustainability

Sustainability is an important constraint for this product because of the consumers this device is marketed for. Our PUMPED device will be portable since it will have a rechargeable battery source, because of this there are environmental factors that will have to be taken into consideration.

The battery must be long lasting not only on a single charge but also for the device's life cycle itself. Since this device will be used numerous times a day by mothers, long battery life is paramount even with all the components housed in this device. If the battery only lasted for a couple of uses, then the environmental impact would be large due to the multiple recharges it would need a day. And for the device life cycle a rechargeable battery loses its capacity to keep a charge over time, so the battery within the device should have a long life.

Longer than a year of use before battery degradation would be preferred so that the need for a new device can be pushed further down the line since our proposed battery will not be designed to be easily replaced. Having a sufficient battery is not the only parameter to keep power management within sustainable environmental constraints but also the components themselves within the device. Power conversion by the PUMPED device's voltage regulators should be sufficiently efficient so power won't be wasted during use, so

the use of switching voltage regulators over linear voltage regulators on critical components will create much better battery usage since the switching regulators are considerably more efficient than their linear counterparts.

The microcontroller is another component that can help combat power mismanagement. The ESP32 microcontroller that we have chosen can in fact implement low power mode so that it will not draw considerable power while not in use and only activate systems that need power when requested by the user. This leads to the next way power can be conserved, which is how the software can be optimized enough so that the weigh sensor does not take unnecessary readings, the LCD screen does not update constantly, and the printer functions in a consistent manner. Maintaining focus on power consumption will ensure sustainability with the device.

Another topic within the realm of sustainability to keep in mind is product durability. The PUMPED device is designed to be used on the go so ideal conditions such as being indoors at room temperature cannot be guaranteed. So, the device must be built with components that can function as intended within a wide range of possible stress factors. Factors that could include extreme ambient temperatures. Another factor is seen due to the nature of the PUMPED device. It is designed to measure liquids so there is always a risk that the device might get wet. The components within the device must be able to function after such a stressor so the device's life cycle will not be shortened. This can be accomplished by using components that are invulnerable to liquids or by housing the components in such a way that they are protected from such a scenario.

Another constraint that falls within sustainability is making the PUMPED device affordable. To keep the device affordable the components chosen must not be expensive but at the same time are made of high quality so that they are reliable, accurate, and most importantly safe so the device will still be a realistic option in the market. If the product was made with only expensive parts, it would make it costly to produce which in turn would make the device have to be sold at high prices. PUMPED would not have a sustainable market outlook. The same can be said if PUMPED was made with the cheapest components to cut costs and maximize profits, safety would become a real concern.

4.2.6 Health and Safety

One of the most if not the most important design constraints to take into consideration for the PUMPED design is how it relates to health and safety. This device will not only be in close proximity to parents and infant children but also to a food that is intended for infants. This makes health and safety paramount because of the vulnerability of its target audience. The PUMPED device needs to be made of materials that are non-toxic and food safe so that there is no contamination of the breast milk it will be used for. The labels that will be printed on thermal paper should be non-toxic as well because they will be transferred by hand to a bag of milk. The label itself must also be tampering proof to prevent the loss of adhesiveness when breast milk is shuttled to and from freezers and refrigerators and out to ambient temperatures. The design needs to have surfaces that are easily cleanable and made in such a way that bacteria has nowhere to grow after sterilization. This can be accomplished by heavily researching the regulations and guidelines imposed by the Food

and Drug Administration (FDA), so unsafe practices can be avoided and consumer confidence in the device can be ensured.

Another aspect of health and safety that is important for this design is tied to the electrical component. Since PUMPED will be a portable device that will runoff a rechargeable battery, considerations need to be taken into account of how this device will run smoothly without the worry of potential hazards like fire or electric shock. The design will contain multiple systems that will work together on a PCB with the potential of having exposed wires, so it will be important to have an enclosure that will properly shield the components to prevent interference that could cause harm to a person or the device. Voltage regulation from the battery is also a big consideration due to the fact that the components will have specific operating ranges.

If the components were to ever fall out of their recommended ranges, the device could no longer function or become a hazard. Most voltage regulators come with some sort of circuit protection from things like temperature spikes and over current situations. The choice of battery is also an important factor to consider because of a batteries ability to catch fire or worse. The battery chosen must have protections that prevent overcharging and overheating. The battery must also be properly placed within the device so that it can operate properly, that means giving it proper ventilation.

Since the PUMPED device will be portable and on the move with the parents there is possibility that the device will experience situations where it will get knocked around or even dropped. The device needs to be able to function properly even after some sort of shock to the system, so sufficiently mounted components and a durable outer design is needed. Since this design is meant for precision weight measurements the base of the device needs to be stable so that an accurate measurement can be taken. Also, since liquids will be involved the base should be wide enough to prevent tipping and spillage. The load cell that is used to weigh the bags of breast milk should also have a maximum capacity above the normal rated bag limit so there won't be a risk of overloading the cell and breaking it.

The PUMPED device will also have an app tied to it so it can seamlessly export information about specific breast milk samples that have been weighed and labeled. It will maintain a running catalog, so privacy protection is necessary in the form of encryptions so data cannot be stolen. The software of the device should also inform its users of any errors or irregularities. In the event of an error the users can be notified so that there won't be continued use of the device until a fix can be made. For example, if the weigh sensor was producing incorrect readings, there would be a notification that the sensor will need to be reset. Keeping these constraints in mind helps ensure the health and safety of the PUMPED device.

4.2.7 Time Constraint

Time is always valuable no matter what you are doing or what you are going to do. Time is the only thing that we can never get back once it passes through so in the case of our time constraint with our PUMPED project, we definitely need to keep ourselves organized for everything because this project is the biggest and most time consuming thing that have

come across in our education. Time constraints are one of the main key constraints that we had when going through our Senior Design journey. We had a limited amount of time in senior design 2 summer semester compared to if we did senior design 1 in the spring and senior design 2 in the fall, but since we did not do that, our time constraint is much more fixed.

We are taking senior design 1 in the spring of 2025, but we are quickly going to be jumping into senior design 2 in the summer of 2025, which drastically shortens the time that we had to work on our project. With that being said, we had to make adequate adaptations in order for us to properly and efficiently get our work completed in a timely manner.

One of the main issues with our time constraint is that we are going to be doing a lot of testing and debugging. That itself takes a lot of time to do so, no one is lucky enough to get everything working properly on the first try. There is going to be delays, problems, issues, and situations that are going to arise as we are working on our project and trying to fix things that come up, so we have to schedule a good amount of time and expect that this will happen so that we don't get hit by it blindly.

Another issue that could possibly come up that could cause a time constraint is that of ordering our PCB board. This can be an extremely long inducive issue if we design our board incorrectly, and we don't realize it until we receive it back. The reason as to why this is a very important thing that we must pay attention to is because PCB fabricators like JLPCB take a couple of weeks for them to ship our requests. The reason for this is because they must fabricate them and send them back, a process that takes a while. We cannot afford to mess up on our PCB design because that will waste a couple of weeks that we cannot lose.

Lastly, another issue that can arise is the availability of all the products and tools that we are going to need. There are many groups that are going to be working on their project simultaneously, there are a limited number of tools that we can use such as the 3d printer. We need to be able to accommodate a scenario in which it will take us a long time to be able to utilize the equipment that has been given to us by UCF, so in order to approach this and manage our time, we have to be able to make schedules for each of us to make progress on a daily basis even if it is just a little at a time.

As much as we wanted everything to flow smoothly, we came across some unexpected issues that took a toll on the time that was given to us to complete our project. It is inevitable. What we can do is to be prepared, to act accordingly and efficiently when these issues arise. We need to be able to have a good sense of time management, be accountable to each other, communicate with one another, and never ghost one another. The professors have stressed this a lot since the beginning of senior design 1. We need to stay and plan of time, we need to be involved and try our best to avoid procrastination. One of the key ideas that we proposed to ourselves is that we need to set earlier deadlines from what they are in case any issue occurs. Since we followed our well thought out plan that we had proposed, we successfully completed the PUMPED project.

4.2.8 Political Constraint

This constraint is not as big as the other constraints because the main thing that we must keep an eye on is where we are getting our parts that we are going to use in our project from. We need to get our parts from a relevant store that is able to sell in the US. We don't

want to get into any trouble or cause any issues regarding that. In the case of our project, we are being sponsored by Victoria and Matthew, both of whom work for Microsoft and Lockheed Martin respectively. We must take accountability that both companies that they work for are American companies and will prefer for our product to stay that way. We definitely don't want to cause any issues that would make us lose our sponsorship.

For us to be good and comfortable regarding political constraints, we are keeping in communication with our sponsors and updating them with any questions or concerns we come across. Since we already have sponsorship, we cannot look for outside sponsorships and we cannot share any of our research information that we have gathered because that will invalidate the request that was made by our sponsor. All our research will be kept confidential and be given to our sponsor, which they will have the rights to all the information and project design we collect.

Following this process, and all of the regulations regarding political constraints, will allow us to efficiently and easily work through our project smoothly without any issues, which can possibly also fall into the time constraint since if there is an issue we would have to rework work that did not satisfy the requirements and sponsorship.

Chapter 5: Comparison of ChatGPT/LLM Platforms

5.1 Chapter Introduction

This chapter explores the role of ChatGPT as a research assistant in our PUMPED project. In this chapter, we go into how using this particular AI model can both help us in our research and also lead us to a dead end by not offering useful information. We will begin by comparing different prompts that were used to investigate various components in our Chapter 3, from the MCU options, Printers, power supplies, regulators, scales and software Design tools and implementation. When going through the process of using ChatGPT we discovered that there are many strengths and weaknesses to it. On the one hand, this AI has the ability to generate a wide range of ideas and suggestions that give us many different approaches that we might not have considered. On the other hand, the information that ChatGPT would give us could be a little bit outdated or not useful. By looking at specific examples that will be listed below, we determine how useful this AI is in regard to Engineering in the real world. Overall, this chapter shows the process and attempt we each took to gain some information on how to approach our PUMPED project, some of the information helped, and some of the information set us back a little bit.

5.2 Prompt Comparison

Using a large language model, like ChatGPT, to assist with research for a project can have some very mixed results. On the one hand your prompt for a particular topic or area of research could present information or issues you had never even considered. The big issue though is that large language models like ChatGPT can hallucinate; that is present information to you as if it is correct, and even tell you it's correct explicitly, and yet the information is absolutely incorrect. Until large language models have the capability to consistently tell you whether or not the information they have presented is 100% correct or presented with some fair degree of uncertainty they will always have to have their responses double checked. Along with ChatGPT there are several other LLM platforms including Google, Claude, Deepseek, etc. that we will compare in terms of usefulness.

Prompt #1 regarding printers: “What is a good thermal printer for a project like this...”

When ChatGPT was prompted regarding the appropriate thermal printer to use for this sort of project its answer took into consideration the following properties and areas.

- Specific printers that could be used
- Voltage compatibility with regards to the printer's operating voltage and the operating voltage of a potential microcontroller
- The power supply needed for the printers
- The size of the thermal paper rolls for the printers

Of the aforementioned elements of ChatGPT response some were more helpful than others. For example, ChatGPT's suggestion of which specific printers to use was outdated because the model is not updated enough to be aware of whether the printers it recommends are in stock or even still produced. One area of concern that ChatGPT brought to my attention was the voltage logic level, particularly the TTL logic level, for potential printers. With this in mind I went on further in my own personal research to look for documentation on printer's TTL logic level. The last two points ChatGPT made were fairly obvious in that it

briefly went over power supplies and thermal paper rolls for the corresponding printers but I think for the power supply ChatGPT can't really give a fully accurate assessment without knowing all the details of our components, if at all.

Prompt #2 regarding mobile app technology: "Which technologies or frameworks would work best for an android mobile app like this..."

When ChatGPT was prompted regarding the appropriate thermal printer to use for this sort of project its answer took into consideration the following properties and areas.

- Programming language to use
- Development frameworks
- Communication methods between the ESP32 and mobile app
- Backend platforms to use for data
- Mobile app UI libraries

This prompt received some useful responses in the sense that it gave a wide variety of suggestions for each of the above elements instead of just narrowing it down to one. I was able to use these various suggestions to then do my own independent research to further decide which I thought were best. I think an aspect of ChatGPT's response where it obviously fell short and could have been arguably harmful to the research experience was it was giving suggestions for cross platform technologies and frameworks to use despite us specifying that we wanted to develop for android.

Prompt #3 regarding batteries: "What kind of battery would you recommend to power a thermal printer, an LCD screen, a load cell, and an MCU?"

When ChatGPT was asked regarding which kind of battery would be best for powering the components in our device, it first described the main power concerns it had, then it presented three different types of batteries it thinks could work and mentioned their main characteristics.

- It accurately mentioned that thermal printers draw high peak currents
- Considered stable voltages for powering peripherals and MCU

The three batteries it mentioned along with their voltage levels, pros and cons were:

- Lithium Polymer (LiPo) Battery
 - Typically, 3.7V per cell
 - Pros: High discharge rate, lightweight, rechargeable.
 - Cons: Requires a proper battery management system (BMS) to prevent over-discharge.
- Lithium-Ion (Li-Ion) Battery (18650 Cells)
 - 3.7V per cell
 - Pros: More stable than LiPo, good capacity-to-weight ratio.
 - Cons: Lower discharge rate than LiPo, needs a BMS.
- LiFePO₄ (Lithium Iron Phosphate) Battery
 - 3.2V per cell
 - Pros: Safe, long cycle life, stable voltage.
 - Cons: Lower energy density than LiPo/Li-Ion.

This prompt gave some useful information since it gave us 3 different options, two of which, we were already considering. The information revolving around the voltage levels, pros and cons and even mentioned that the batteries needed to have a battery management system to prevent overcharging. However, it failed to mention the safety hazards of using these kinds of batteries such as the possibility of explosions or fires if mishandled. Overall, it gave use some decent information.

Prompt #4 regarding charge controllers: “What kind of charge controllers could be used to prevent overcharging these batteries?”

After prompting ChatGPT with this question, it took into consideration the amount of current the system will be drawing and it considered choices that would help prevent overcharging, over-discharging, and overcurrent in the thermal printer, the LCD screen, load cell, and MCU system. Then, it gave information on different options depending on the different battery choices that it recommended in prompt #3. Since this prompt was asked right after Prompt #3, it did benefit from the fact that it had the previous information regarding possible battery options.

- Recommended Charge Controllers:
 - TP4056 (for single-cell 3.7V LiPo)
 - Ideal for small-scale applications.
 - Includes overcharge, over-discharge, and short-circuit protection.
 - BMS (Battery Management System) for 2S/3S LiPo (e.g., 2S 7.4V or 3S 11.1V)
 - Ensures balanced charging & protection for multi-cell packs.
 - Example: 3S 12V 25A LiPo BMS Module.

Then it gave options for the Lithium-Ion Battery cells specifically 18650 cells,

- Recommended Charge Controllers:
 - TP4056 with Protection Circuit (for single 18650 cells)
 - BMS for multi-cell packs (e.g., 2S, 3S, 4S Li-Ion)
 - Example: BMS 3S 12.6V 10A Protection Board.

Finally, it mentioned possible choices for the Lithium Iron Phosphate battery cells,

- Recommended Charge Controllers:
 - LiFePO4-Specific BMS (e.g., Jikong 4S 12.8V BMS)
 - MPPT Solar Charge Controllers (for renewable energy applications)
 - Example: EPEVER MPPT 10A-20A for LiFePO4.

In the end, ChatGPT’s response ended up being a little helpful in the way that it is very quick to recommend certain parts and explain what it thinks is best for the project. However, it does come with its limitations when asked about parts that we can use in our project. For instance, it does not have the full context of what our project is, the exact parts we will be using, and it doesn’t have real time information on components and their availability. Without having the full context of what we are designing, it mainly prompts us with very generalized answers rather than a specifically tailored response or recommendation to satisfy our design. Even with the generalized answers, it does not have

real time access to manufacturers sites with their datasheets and it could very well be giving us outdated or incorrect technical advice. From a general standpoint, it did provide us with very simple options to start off, this could be used as a starting point to then research these parts and determine if they could be useful in our design or not.

Prompt #5 regarding weight sensors: “The intention is to design a device that will assist parents with weighing and labeling breast milk inside bags. The device will be portable and will include a rechargeable battery, a weigh sensor, thermal printer, ESP32, and will incorporate an app that will receive information from the scale and other information. What types of weight sensors would you recommend for this project?”

Strain Gauge Load Cell (with HX711) – Recommended

Measures strain in a metal bar under weight. Offers high precision (gram-level), low power use, and easy integration with ESP32 via HX711. Requires calibration and may be affected by temperature but is ideal for accurate, affordable, and reliable weight measurement in your application.

Capacitive Weight Sensor

Detects changes in capacitance due to force. Highly sensitive and fast, but more expensive and needs a dedicated capacitance circuit. Best for compact designs needing high precision and quick response.

Piezoelectric Sensor

Generates voltage when compressed. Very sensitive and compact but unstable for continuous/static loads and needs signal conditioning. Suitable for detecting quick force changes rather than consistent weight.

Digital Pressure Sensors (MEMS-Based)

Uses air pressure in a micro chamber to measure force. Very compact and easy to interface with ESP32 via I2C/SPI but has limited weight capacity and needs careful mechanical design. Good for ultra-small applications.

Resistive Force Sensors

Change resistance under pressure. Thin, flexible, and inexpensive, but less accurate and not suitable for precise or continuous weight readings. Better for presence detection or rough estimates.

ChatGPT’s response to this question was considerably accurate with its recommendations on how they would benefit our design. It laid out several technology types that we considered and some new ones that were not considered as options. Pros and cons were laid out for each option.

From ChatGPT's recommendations for weight sensors, its best recommendation was the load cell and HX711 combination, which is the technology we intend to use. It suggests that this is the best choice because it provides high accuracy, easy integration, and is cost-effective. It also highlights its ability to integrate with an ESP32.

Prompt #6 regarding voltage regulators: "The intention is to design a device that will assist parents with weighing and labeling breast milk inside bags. The device will be portable and will include a rechargeable battery, a weigh sensor, thermal printer, ESP32, and will incorporate an app that will receive information from the scale and other information. What types of voltage regulators would you recommend for this project?"

When probed with this question ChatGPT's response was an informative one. It not only laid out its recommendations for regulators but also explained how each type of regulator could work with the different parts in our design. Though ChatGPT wasn't told the specific parts we chose, it provided possible voltage requirements for each part and why a certain type of regulator was the right choice for it.

Voltage Requirements:

The system has different voltage needs:

- ESP32 -3.3V
- HX711 Load Cell Amplifier - 3.3V or 5V
- Thermal Printer - 5V (or higher, depending on model)
- LCD Display - Typically 3.3V or 5V
- Rechargeable Battery (Li-ion/Li-Po) - 3.7V (nominal), 4.2V (fully charged)

Because of these varying voltages, you need both linear and switching regulators.

Recommended Voltage Regulators:

1. AMS1117-3.3V (LDO)
 - a. Input: 4.5V - 12V
 - b. Output: 3.3V @ up to 1A
 - c. Use Case: ESP32, HX711
 - d. Pros: Cheap, easy to use
 - e. Cons: Not efficient for battery-powered applications
2. LP2985-3.3V (LDO)
 - a. Input: 3.7V - 16V
 - b. Output: 3.3V @ up to 150mA
 - c. Use Case: If the ESP32 is in deep sleep mode and you want ultra-low power draw
 - d. Pros: Low quiescent current (saves battery)
 - e. Cons: Limited current output

Linear Regulators (LDOs) - For Low Power Components

Linear regulators are simple and produce clean power, but they waste energy as heat. They are ideal for low-current components.

Switching regulators (DC-DC converters) are much more efficient (80-90%) compared to linear regulators, making them ideal for battery-powered designs.

Even though our design requirements were not explicitly stated, ChatGPT's recommendations are helpful when planning on which type of regulator is best. But it would be better to state the specifics because some of the regulator options are not recommended for integrated circuits.

Prompt #7 regarding Microcontrollers and Field Programmable Gate Arrays:

When prompting ChatGPT about MCUs and FPGAs, the goal was for it to determine the difference between both, it will allow us to determine which of them will allow us to accomplish what we are looking for, and what could possibly be our options when choosing a specific one.

Case 1: what are the differences between a microcontroller and a Field Programmable Gate Array?

Microcontrollers and Field Programmable Gate Arrays (FPGAs) are both used in embedded systems, but they differ significantly in architecture, programming, and typical applications. Here's an overview of the key differences:

Architecture & Programming Model:

Microcontrollers are compact computing systems with a fixed CPU, memory, and peripherals all on a single chip, designed for executing sequential programs written in languages like C or C++. They are ideal for tasks with predictable control flow and offer a straightforward development process. In contrast, FPGAs are made up of configurable logic blocks and programmable interconnects, functioning as blank slates that allow developers to design custom digital circuits using hardware description languages like Verilog or VHDL. This allows FPGAs to execute multiple tasks in parallel and perform complex hardware-level operations.

Performance, Flexibility & Cost:

While microcontrollers are cost-effective, energy-efficient, and simpler to develop for, their performance is limited by their sequential execution model and fixed hardware. They are best suited for general-purpose or control-based applications. FPGAs, on the other hand, offer significant advantages in speed and parallel processing, making them ideal for high-performance or real-time applications. Their reconfigurable architecture allows for hardware-level customization even after deployment, though this comes at the expense of higher development complexity, cost, and power consumption.

- Microcontrollers are best suited for applications that require straightforward control, low cost, and low power consumption, with a fixed hardware architecture managed through software.

- FPGAs are ideal when high-speed parallel processing or custom hardware configurations are needed, offering flexibility and performance at the cost of increased design complexity and typically higher power consumption.

The experience when using ChatGPT to determine the differences between MCUs and FPGAs was not bad. Even though the prompt that was offered to AI was not super concise and was generalized. It gave us the main differences between MCUs and FPGAs. We already knew what the main differences were between them, but we also learned some new things that we didn't know about them. Overall, the experience was great when prompting this and gave us a much clearer understanding of why we decided to go with an MCU for our PUMPED project.

Case 2: **what are some great MCUs to use that are capable of wireless communication?**

There are many microcontrollers that offer robust wireless capabilities for IoT, sensor networks, and other connected applications.

- ESP32 Series
- ESP8266
- Raspberry Pi Pico W
- Nordic nRF52 / nRF52840
- TI SimpleLink (e.g., CC2640, CC2650)
- STM32WB Series

Choosing Tips:

- Match wireless protocol to your app (Wi-Fi, BLE, Zigbee, etc.)
- Prioritize low power for battery-based designs
- Ensure the MCU has enough processing power and memory
- Look for strong development tools and community support
- Factor in cost and availability for production scaling

When prompting ChatGPT on case 2, it gave us a few different types of MCUs that we could use. A lot of the options offered are very popular, but also there was a limited list that it gave us. It is also great that it gave us some pros and cons for each series so that we could compare within the list. One thing that was very nice that ChatGPT offered was some considerations when choosing an MCU, which is very important in every project. It wasn't useful for us because we had already done our research and knew exactly what MCU we were going to use.

Prompt #8 regarding Displays:

In the case when choosing a display for our project, our prompt for ChatGPT was targeted towards trying to know the different types of displays that exist that can be paired with our

MCU of choice. We made sure that the details of our prompt were accurate so that there wouldn't be any display that was incompatible with the ESP32.

The ESP32 is a versatile microcontroller with built-in Wi-Fi and Bluetooth, making it great for display applications. You can pair it with various types of displays depending on your project needs. Here are some common display types that work well with the ESP32:

What are the different types of displays that can be paired up with the ESP32 microcontroller?

- OLED Displays
- LCD Displays (Character & Graphic LCDs)
- TFT LCD Displays
- E-Paper (E-Ink) Displays
- RGB LED Matrix Displays
- Touchscreen Displays

ChatGPT gave us a very large list of different types of displays that are compatible with the ESP32. There were some displays that we didn't think about because they were not really worth looking into.

Prompt #9 regarding User interface:

User interface was pretty easy to figure out with what we wanted to do, we already had a plan going into it regarding how the user would navigate through the device when printing and doing other key details and changes to the device. We were stuck between having every user option on a touchscreen, but we decided to go with physical buttons, but aside from that, this is what we prompted ChatGPT:

What are the advantages and disadvantages of using a touchscreen or physical buttons?

Advantages and Disadvantages of Touchscreens vs. Physical Buttons in summary:

Touchscreens offer a versatile and minimalistic interface with dynamic functionality, easy software updates, and multi-touch gesture support, making them ideal for modern devices like smartphones and kiosks. However, they lack tactile feedback, consume more power, are more expensive to repair, and can be difficult to use with gloves or in wet conditions. On the other hand, physical buttons provide reliable tactile feedback, work well in harsh environments, and consume less power, making them suitable for tasks requiring precision or muscle memory. Their drawbacks include limited functionality, potential wear over time, added bulk, and increased hardware complexity.

The information that ChatGPT was in-depth, although it was measuring even the most extreme cases, it was very informative and allowed us to rethink our decision for a short time. We still decided to go with our initial decision, but it was very beneficial for ChatGPT to have given us this information.

Prompt #10 regarding PCB Software Programs:

There are many different types of PCB design software that exist, we wanted ChatGPT to be able to list all the existing software and determine if they were still managed by their companies, if they were free or if they cost money.

What are some different PCB software design applications that are available to use? Are they free or do they cost money?

There are many PCB design software applications available, ranging from free, open-source tools to premium, professional-grade applications.

PCB design software comes in both free and paid options, catering to a range of users from beginners to professionals. Free tools like KiCad are open-source and widely adopted, while EasyEDA offers cloud-based convenience with both free and premium plans. DesignSpark PCB provides solid functionality at no cost, Fritzing is ideal for Arduino and breadboard designs, and Osmond PCB is a Mac-only tool suited for small projects. On the paid side, Altium Designer stands as the industry gold standard with powerful features but comes at a high price. PADS offers flexible professional solutions at varying costs, and DipTrace is known for its user-friendliness, with free and paid tiers starting around \$75.

Even though ChatGPT offered us a lot of options to use, there were some key details that were not mentioned, specifically Altium. We wanted to use Altium from the beginning because it is a great PCB software, and we managed to obtain a free membership due to us being full time students. Overall, ChatGPT made some key points but there was a lot of important information that was not mentioned.

Chapter 6: Hardware Design

6.1 Hardware Overview

Going into this chapter gives details about the design and technical architecture of our PUMPED senior design project. This chapter presents the visual representations and technical descriptions of our hardware designs in schematic diagrams. Beginning with our subsystem block diagrams, we will outline the connections between the core components, and we will illustrate how modules such as the MCU, sensors, printer, power supply, and

other components interact within the system. The schematic diagram will follow right after, showing the connections and component placements in a deeper and more detailed level. We then will explore the system architecture, which will provide a high-level framework for understanding data flow, control logic, and the implementation of hardware and software elements. Additionally, structural illustrations will give a clear picture of the device's physical design and assembly, which will ensure that all parts are working together properly. Together, these designs will form the basis for our PUMPED project.

6.2 Hardware Block Diagram

The Hardware Block diagram shows us a high-level overview of all of our major components in the PUMPED system and how they all interact with each other.

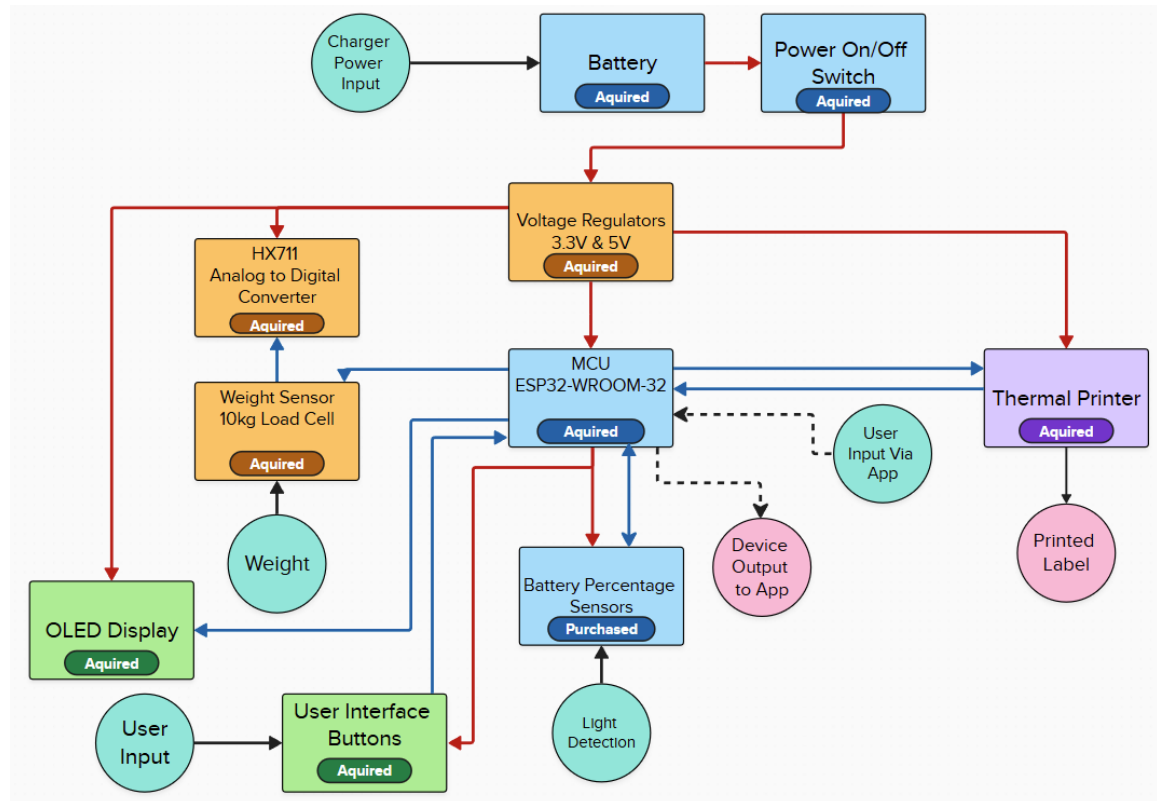


Figure 8 Hardware Block Diagram

It shows a visual representation of how the power flows through the system, starting with the battery, going to a power on / off switch, into the voltage regulators and then being dispersed into the peripheral components including the thermal printer, OLED screen, our ESP32-WROOM-32 microcontroller, the Load cell and analog to digital converter (HX711), a Bluetooth module, and the user interface buttons.

6.3 Power Supply

For our power supply we are using the ROMOSS Sense 8+. It weighs around 1.5 pounds and has the dimensions 6.6"x 3.1"x 1.3". This is the heaviest and largest component in our design. So, it will directly impact the size and weight of the PUMPED system. We wanted to design our system to be safe, efficient, and portable while still being able to efficiently

deliver power to all of our components. In terms of power delivery, we used two output ports. They are both USB-A connections, 1 of them will be powering stepped down to 3.3V to power the ESP32 microcontroller. The other will be used to power a 5V regulator which will power the thermal printer, OLED screen, and the load cell.

6.3.2 Voltage Regulators

To generate the stable voltage outputs that our system requires, we used one LM2576 voltage switching regulator, one that will step down our power bank's input of 5V to output a voltage of 3.3V. This switching regulator was chosen for its high efficiency and its ability to handle higher current outputs. The ability to handle higher current is essential to our design since the thermal printer can hit current draws of up to 2A while printing, and our system must be designed to handle it. Each regulator is supported by a diode, inductor, capacitor circuit to meet the design recommendations provided by Texas Instruments.

6.3.2.1- The 3.3V Voltage Regulator

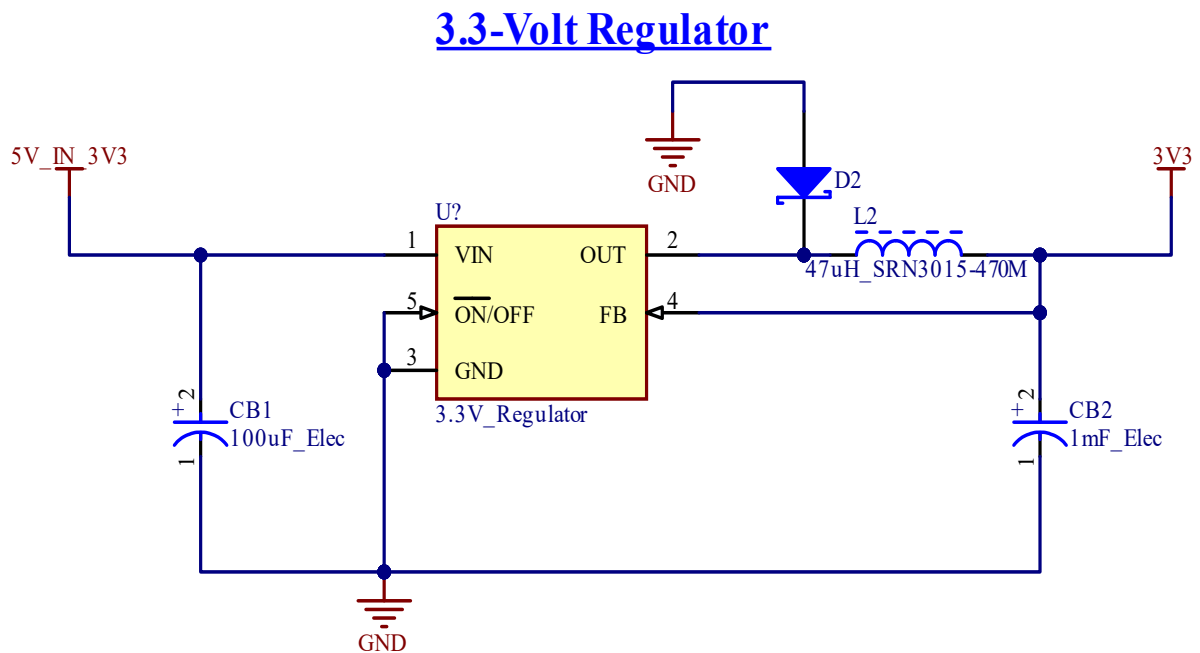


Figure 9 3.3 V Voltage Regulator

This schematic shows the layout of the LM2576HVSX 3.3-volt voltage regulator that we used. The voltage regulator will be the go between our power bank and our components that will require 3.3 volts, such as the MCU, the OLED display, and the push buttons. To properly function, this voltage regulator will require a 100uF input capacitor, a 1mF output capacitor, 47uH output inductor, and a diode that has a current rating above 3A and a reverse voltage rating 1.25 times the maximum input voltage. The port this voltage regulator will be connected to on the power bank will be rated to provide 5 volts at 2.1A so the regulator will be adequately capable of dropping down the voltage to 3.3 volts. To check that the voltage regulator is functioning properly, we have added an LED to the output that will illuminate when output is provided. The graph below was used to find the appropriate value for the output inductor to run this regulator efficiently at 3.3 volts.

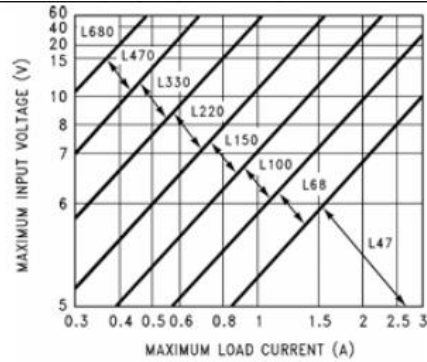


Figure 8-4. LM2576(HV)-3.3

Figure 10 Voltage Regulator Inductor Chart

6.3.2.2- The 5-V Voltage Regulator

5-Volt Regulator

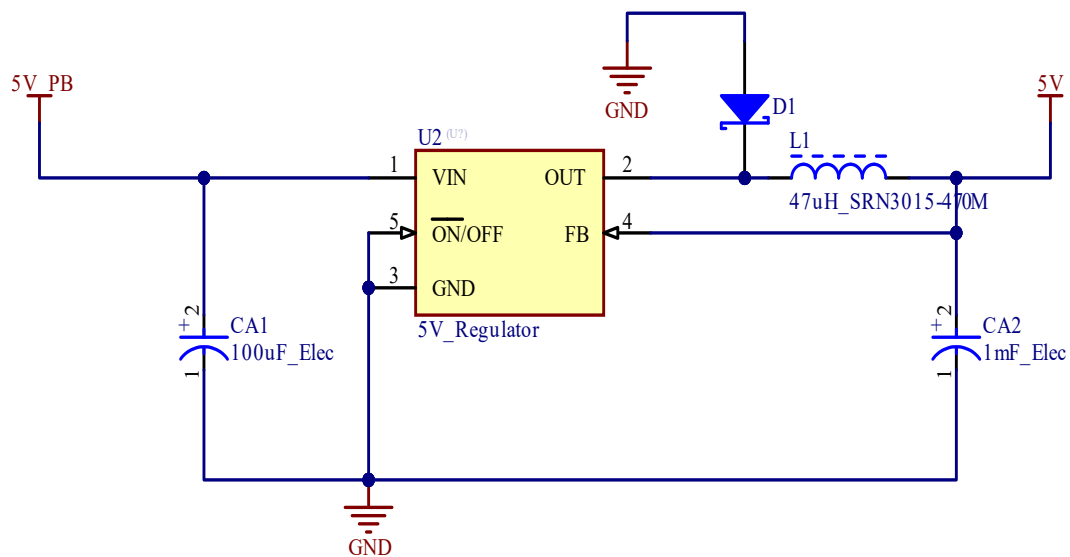


Figure 11 5V Voltage Regulator

The schematic above shows the layout of the LM2576HVSX 5-volt voltage regulator. This voltage regulator will serve as the go between the bank and the components that require 5-volts to function, such as the thermal printer and the Load cell and the ADC. For this regulator to function properly at its expected output it will need a 100uF input capacitor, a 1000uF output capacitor, a 47uH output inductor, and a diode that has a current rating above 3A and a reverse voltage rating 1.25 times the maximum input voltage. The port this voltage regulator will be connected to on the power bank will be rated to provide a maximum of 12 volts at 1.5 A and minimum of 5 volts at 3A, depending on what is drawing power, so this regulator will be adequately capable of dropping down the voltage to 5 volts. To check that the voltage regulator is functioning properly, we have added an LED to the

output that will illuminate when output is provided. The graph below was used to find the appropriate value for the output inductor to run this regulator efficiently at 5.0 volts.

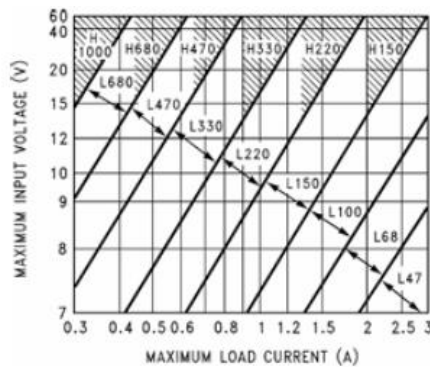


Figure 8-5. LM2576(HV)-5.0

Figure 12 5V Voltage Regulator Inductor Chart

6.4 Main PCB Schematics

The main PCB will serve as the central hub for all of the peripherals and the ESP32 microcontroller on the PUMPED device. The PCB will be handling the core processing and communication tasks while also distributing the regulated power to all of our subsystems. The schematic layout was designed with the idea of modular systems so that we could make it simpler to troubleshoot and test different systems in a much easier way. Each major group of subsystems such as the OLED display, the thermal printer, the user interface buttons, and the analog to digital converter (HX711) for the load cell, are routed through different labeled headers. This allows us to simply plug the systems into the PCB making it easy to replace or test components.

The schematics also include a USB-to-UART interface using the CP2102. This system closely follows the system implemented in the ESP32 development kits that allows us to program the ESP32 via a USB connection automatically. However, we also include the boot and reset buttons to allow a user to manually upload code to the ESP32 if necessary. This designs provides us with a little redundancy in the ways we can program our board. The schematics shown below show the main PCB divided into separate subsections that cover the different sections on our board and explain in detail what the purpose of schematic accomplishes.

6.4.1 ESP32-WROOM-32

ESP32-WROOM-32 SCHEMATIC

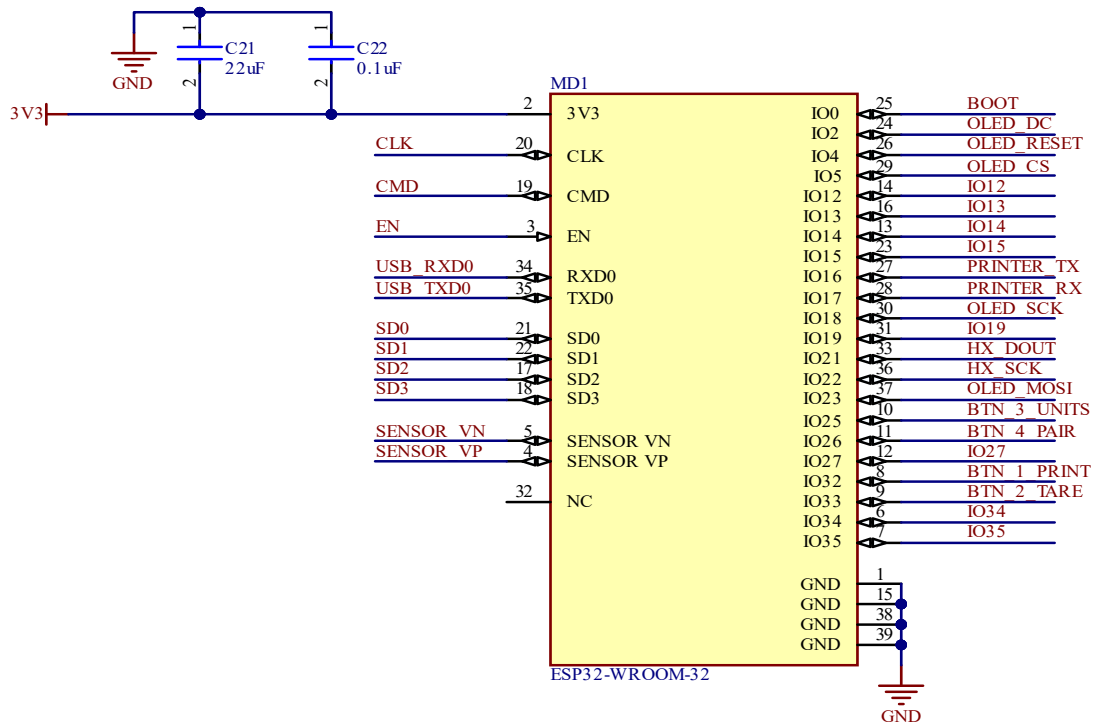


Figure 13 MCU Schematic

This schematic represents the brains of the PUMPED device. The MCU circuit is based on the ESP32-WROOM-32E module which we chose to control all of the device's peripherals. The ESP32 will be responsible for processing the weight data sent to it from the load cell/HX711, it will drive the OLED display, receive data from the user via buttons or via Wi-Fi, and it will send print commands to the thermal printer. This schematic is designed with the purpose of keeping the PCBs as modular as possible so that troubleshooting or replacement of components can be as quick as possible.

BOOT & RESET PULLUP-RESISTORS



Figure 14 Pullup Resistors

BOOT & RESET BUTTONS

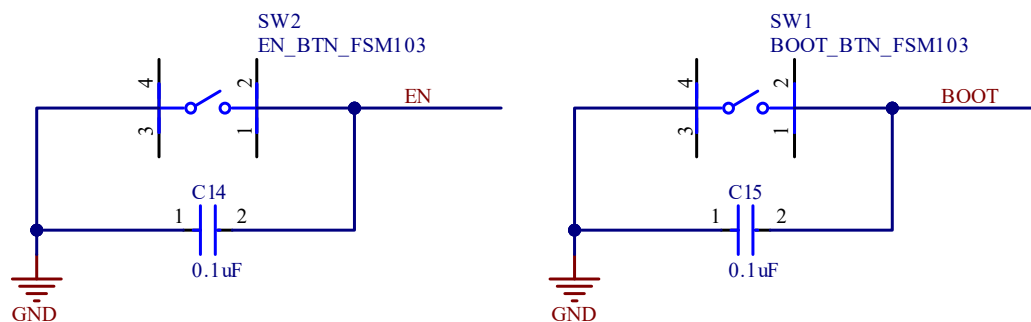


Figure 15 Boot and Reset Buttons

This section of the schematic shows the manual boot and reset buttons that will be located on the main PCB. The EN (Reset) and the BOOT (IO0) pins are both being pulled up to 3.3v using 10k Ohm resistors. Both pull up circuits also include a decoupling capacitor of 0.1uF to help with stability of the signal.

The buttons shown above in the schematic are used for manual control of the ESP32 MCU. If the EN button is pressed, it temporarily pulls the EN pin low, which will reset the system. If the BOOT button is held while the EN button is pressed, the ESP32 will go into its flash mode where we can upload code to it via its RXD0 and TXD0 pins. These buttons, like the pull up resistors, also have a 0.1uF capacitor in place to minimize noise and improve stability.

6.4.2 CP2102 USB-to-Serial and ESP32 Flash Interface

This section explains the design of our USB-to-Serial interface that allows us to program and communicate with the ESP32 microcontroller from our computers. The interface was built based on the CP2102, which is responsible for handling all of the USB communication protocols and converts it into a standard serial signal that our microcontrollers can actually

understand. This design also implements a similar system used on the ESP32 Development boards that facilitates an automatic reset circuit so that we can automatically upload code from our computer onto the ESP32.

6.4.2.1 USB-A Connector, Protection, and Auto Reset Circuitry

USB-A 5V & USB-UART W/AUTO-RESET

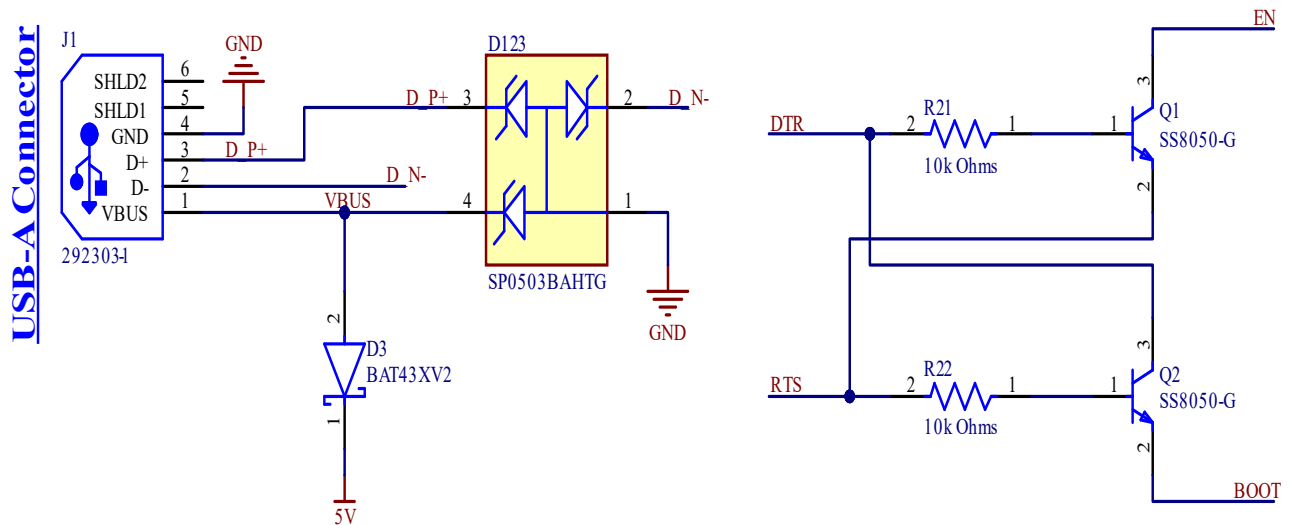


Figure 16 USB-A Connector Schematic

The above schematic section shows a part of the USB to UART interface and also shows the auto-reset circuitry that will be used for programming the ESP32. On the far-left side, we start the circuit with a USB-A connector, this will bring in 5V power to the VBUS, along with the data line signals D+ and D-. The data lines are protected using a diode array which guards the lines against any voltage spikes and electrostatic discharge. Along the VBUS line, we have a Schottky diode that protects the VBUS from the 5V power supply coming from the regulators.

On the right-hand side of the schematic, we have the DTR and RTS signals that come from the CP2102 IC and are routed to 2 NPN transistors. These transistors will be controlling the ESP32s EN and BOOT pins and they are what enable us to program the board automatically. When the CP2102 chip is triggered by an upload, the transistor will pull up their respective lines to low, which automatically resets the ESP32 and puts it into flash mode without having to manually press the buttons that we mentioned before.

6.4.2.2 CP2102 USB-to-Serial Converter

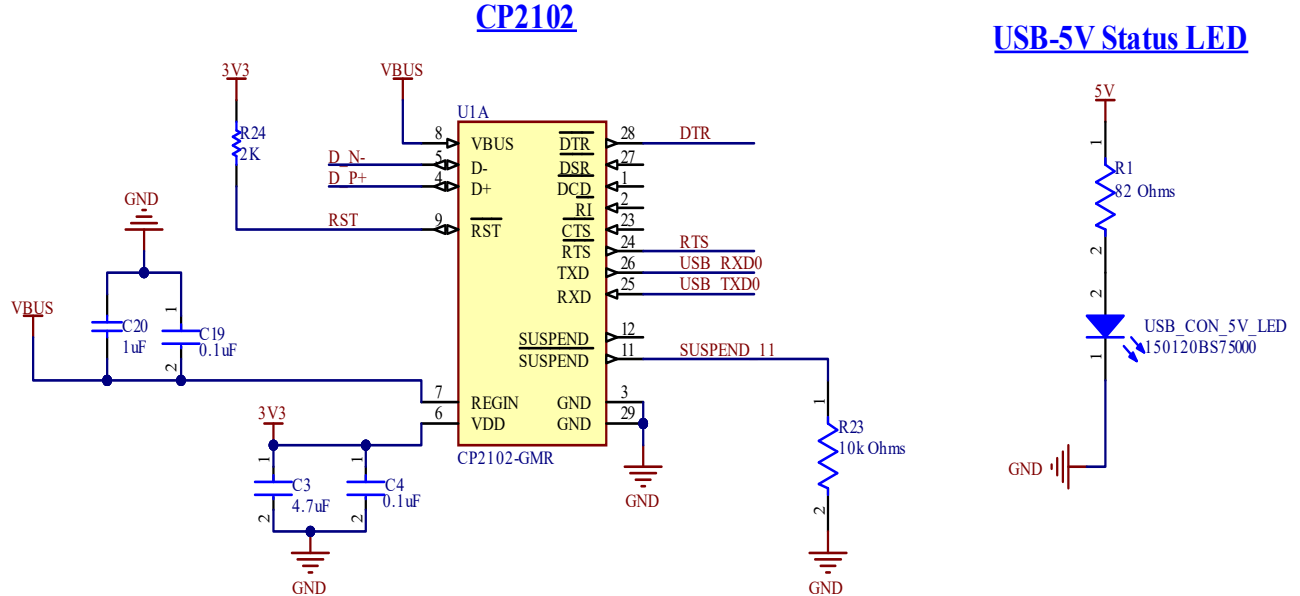


Figure 17 UART- USB to Serial Converter Schematic

The section shown above shows the CP2102 USB-to-Serial converter, this IC is what allows the computer to communicate with our ESP32 so that we can program it. It is powered by the 3.3V regulated output coming from the regulators and it receives the data from the USB-A connector we mentioned above. We have a pair of decoupling capacitors on the VBUS line. We also have another pair on the VDD pin that helps stabilize the chips internal regulator.

The CP2102 routes its TXD0 and RXD0 signals to the ESP32, and the DTR and RTS lines are connected to a reset circuit, which was shown above. On the right side of this schematic, we have a status LED that was included to serve as a visual signal for us to know that the VBUS is actually working and providing power. Connected there, we have a low-forward voltage LED (3.2V) and an 82 Ohm resistor to limit the current it receives to about 20mA.

This circuit together with the one shown above with the USB-A connector completes the USB-UART connection needed to program the ESP32. Replicating the way that the ESP32 Development boards upload code to the ESP32 without having to manually reset and boot the system.

6.4.3 Connectors for Peripherals & Testing

PERIPHERAL CONNECTORS / TESTING PINS

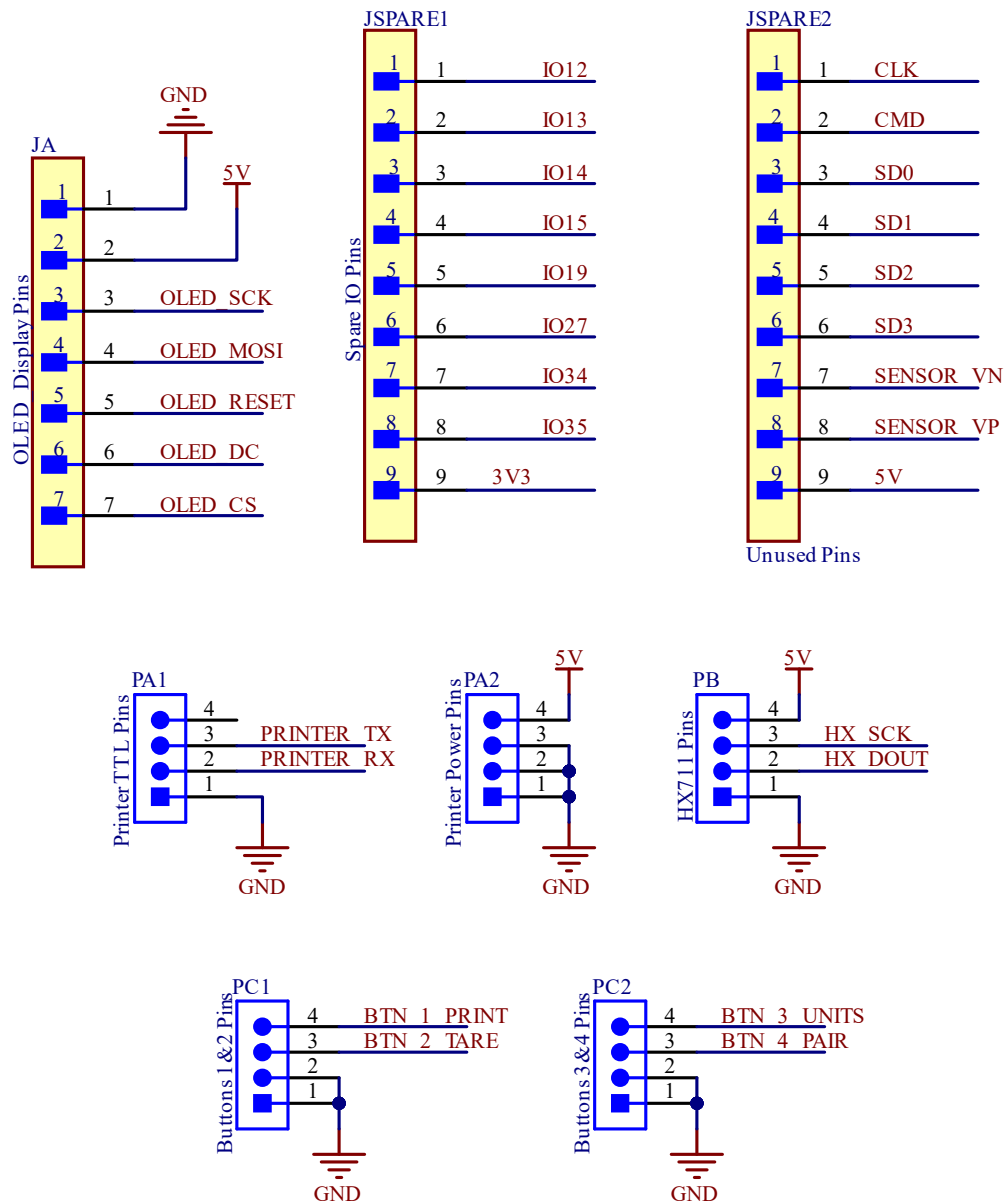


Figure 18 Peripheral Connectors/ Testing Pins

This section of the schematic shows all of the ESP32 IO pins organized in different headers so that our design can be easily testable. Each group of pins is labeled with what its purpose is, and each pin is labeled with where the pin is connected to on the ESP32. Here, we show different pin group for our peripherals including the OLED display, the thermal printer, the HX711, and the user buttons.

- OLED Display Header:

- This header routes the SPI signals needed for the ESP32 to communicate and show our weight values on the display. It includes the OLED_CLK, OLED_MOSI, OLED_RESET, OLED_DC, and the OLED_CS. Along with these data pins, the header also includes the 5V power and ground pins necessary for powering the display.
- Thermal Printer Headers:
 - For the printer, we have two 4-pin headers, one is dedicated to the printer's communication wiring and includes the connections for its TX/RX TTL lines along with a ground.
 - The other header is dedicated to powering the printer. This one includes the 5V power pin that will receive power from the 5V regulator. The power connectors have been separated from the communication pins to minimize any noise that the higher current draw from the thermal printer could inflict onto the data lines.
- Load Cell Header:
 - This header provides the pins needed for the HX711 ADC to interface and send the weight data from the load sensor to the ESP32. It includes pins for the HX711's SCK, and DOUT along with the power lines necessary for powering both the ADC and the load cell.
- Button Connectors:
 - The button headers will be used to route 4 buttons, the print button, tare button, reset button, and pair button. These are simply connected to the ESP32 and if the button is pressed, it will connect the pin low which can be read by the ESP32.
- We also included 2 general purpose headers which include pins that are not currently being used in our design, and pins that are not normally used from the ESP32. However, including these headers allows us easy access to pins for testing purposes, and it also opens up the possibility for us to add more functionality to the system. The extra IO pins could serve as connections for more buttons or any extra peripherals we may want to integrate. This is perfect for prototyping, testing, and troubleshooting.

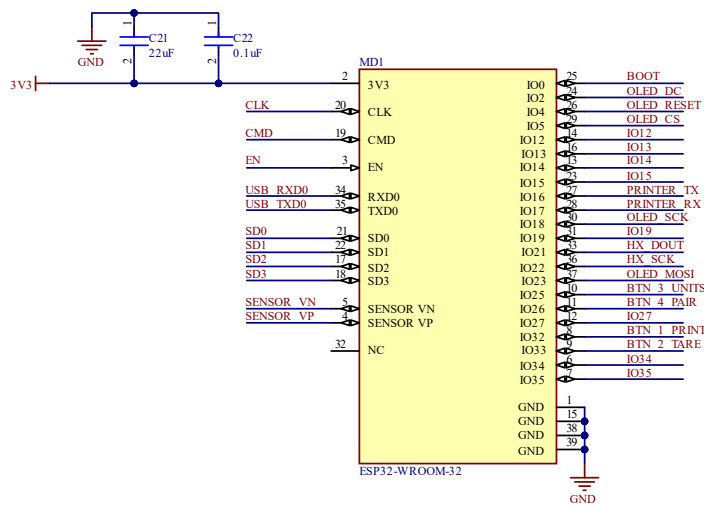
6.5 Peripherals

This section explains what our key peripherals are in our PUMPED device and how they will interface with the ESP32 microcontroller. Each peripheral plays a vital role in the function of our device and together they gather data, display outputs, and print the labels. The components covered in this section include the load cell with its analog to digital converter (HX711) module, it includes the thermal printer which will generate our physical labels to apply to the breast milk bags, and the OLED display so the user can see the weight of the bag before printing. Each device is powered by the regulators and controlled by the ESP32. The OLED display will be using SPI communication protocol, the printer will be using TTL serial communication, and the load sensor will be transmitting analog communication to our HX711 which will then amplify and convert the analog signal into a digital signal that our ESP32 can read.

6.5.1 Load Cell & HX711

6.6 Overall Schematic

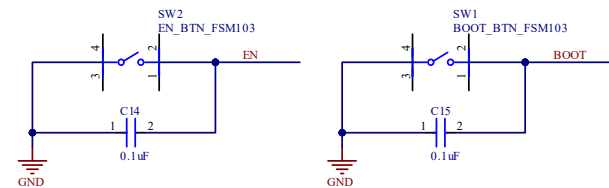
ESP32-WROOM-32 SCHEMATIC



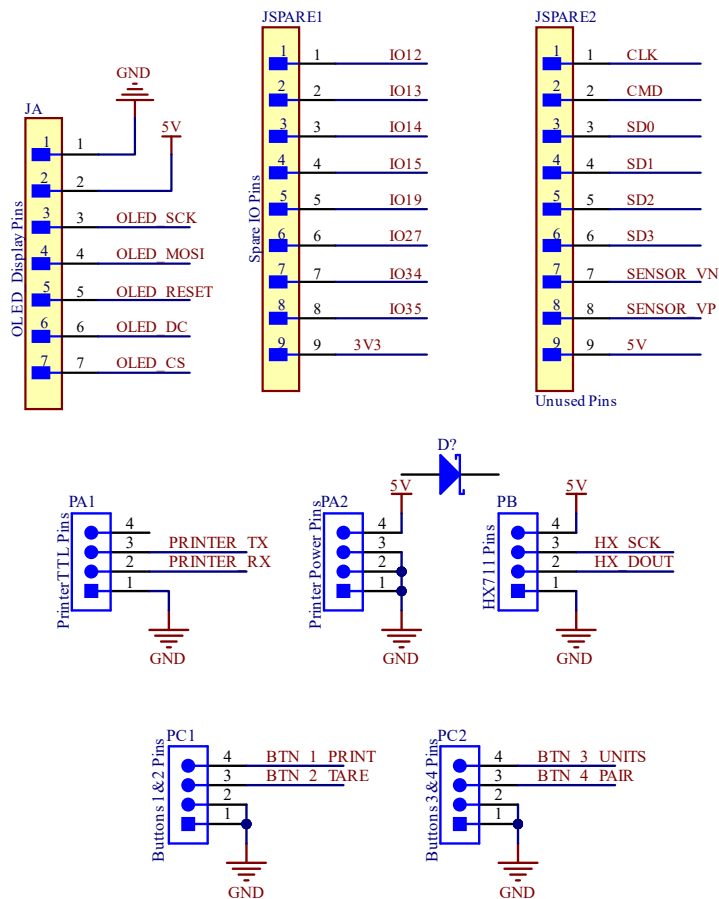
BOOT & RESET PULLUP-RESISTORS



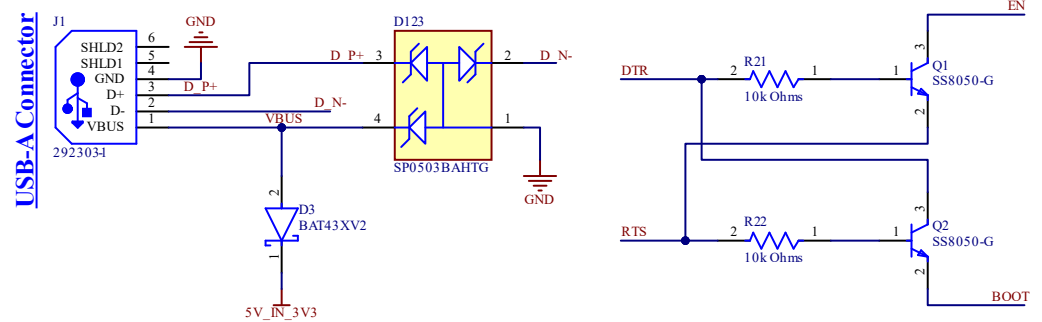
BOOT & RESET BUTTONS



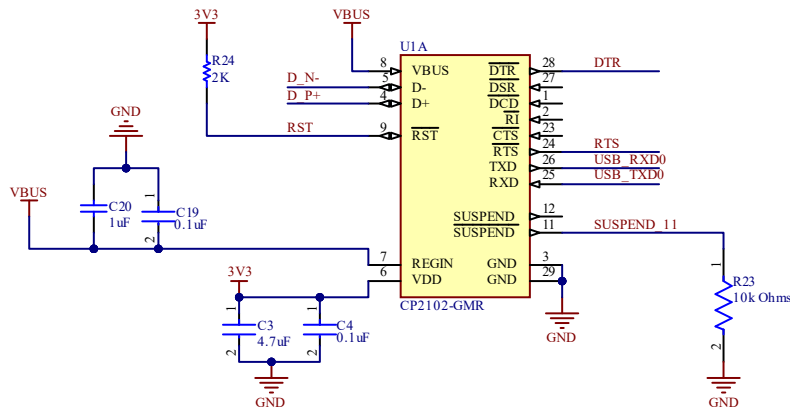
PERIPHERAL CONNECTORS / TESTING PINS



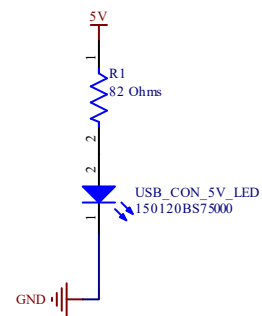
USB-A5V & USB-UART W/ AUTO-RESET



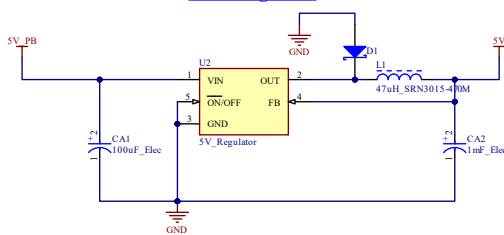
CP2102



USB-5V Status LED



5-Volt Regulator



3.3-Volt Regulator

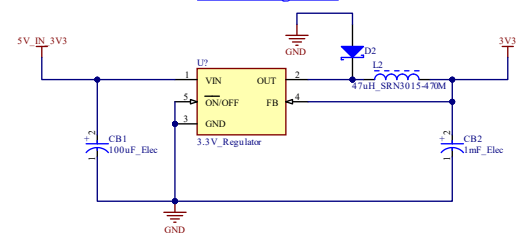


Figure 20 Overall Schematic

The overall schematic combines all of the subsystems and the main system of the PUMPED device into a single, comprehensive schematic of the entire design. It shows the microcontroller unit (ESP32-WROOM-32), the power regulator circuits, USB-to-Serial programming interface, the connections for the user input buttons, and all the key peripherals including the OLED display, load cell, and the thermal printer. This schematic will act as the foundation for our final PCB design layout and it offers a complete map of where each pin is connected. This schematic will be essential for troubleshooting the PCB in the future because we can reference each connection and what it is connected to.

At the top left, we have our ESP32 MCU, which serves as the brains of our systems. It communicates with all the peripherals and facilitates the user interface with the system via reading the button presses, showing the real-time weight on the OLED display, receiving the weight data provided by the load cell and amplified by the HX711, and controlling the printer by sending it data to print the weight. These connections to peripherals are all shown in the center of the schematic where we can see all the headers labeled with each peripheral.

Key feature of our design include:

- Dual Voltage Regulators:
 - At the bottom right of our schematic, we see our 3.3V LM2576 voltage regulator along with its supporting circuitry.
 - It is responsible for powering our ESP32 Microcontroller unit.
 - At the Bottom Left of our schematic, we see our 5V LM2576 voltage regulator with its supporting circuitry.
 - It is responsible for powering all of our peripherals including the weight sensor, the OLED display, and the thermal printer.
- USB-to-Serial Programming Interface:
 - Located just above the regulators, we see our CP2102 chip responsible for handling the USB communication to program the system.
 - To the right of it, we have an LED light indicating if we have power coming from the USB-A connection
 - On top of that, we have our auto reset circuitry, completed with transistors that will facilitate programming the board with an automatic function.
- Peripheral Headers:
 - The SPI lines to the OLED display include OLED_CLK, OLED_MOSI, OLED_DC, and OLED_RESET and are located on the OLED header.
 - The UART lines PRINTER_TX and PRINTER_RX which connects the thermal printer are located on their own dedicated header
 - Finally, the HX711 communicates via the HX_SCK and HX_DOUT pins located on its own header. This allows the load cell to send the weight to our ESP32 so that it can be projected on the screen and printed.
- User Input buttons:
 - Our current design incorporates the use of 4 buttons that do different things in the PUMPED system. They are the print button, tare button, pair button, and the reset unit's button. These are wired between two separate 4 pin headers that grant each button a connection to its respective pin on the ESP32 and a connection to ground.
- General Purpose I/O's and Testing Pins
 - We also have 2 headers with spare GPIO pins for testing and troubleshooting purposes. We have various spare I/O pins available to us in case we want to expand with more peripherals or add more buttons to improve the system's functionality.

In summary, the overall schematic serves as the complete electrical blueprint for the PUMPED system. It shows all the components together in an accessible format that we can reference at a later date to construct, evaluate, and troubleshoot our system.

Chapter 7: Software Design

Chapter 7 focuses on the software design process for our PUMPED project, giving a descriptive overview of the software that will drive our device and its functionality. In this chapter we are going to break down the software architecture using various presentational diagrams that will clarify the system's behavior and integrations. We will begin with flowcharts that will map out the logical sequence of operations, followed by a use case diagram that highlights the interactions between the user and the system. A state diagram will show the dynamic behavior of the system under different conditions, while a class diagram will define the structure of the software by detailing the relationships between objects and components. In addition, we will examine the user interface design, emphasizing an easy-to-use interface and intuitive navigation, outline data transfer protocols and structures and will ensure efficient and secure communication between the system modules and other devices. All these tools will show the development and implementation of the software that corresponds to the physical device and the mobile application that will be used as part of our PUMPED project.

7.1 Device Software

As previously stated in chapter 3, we have many different types of software applications that we can use to write out our software code, but there are few that are able to keep everything organized. For testing purposes, we have been using the Arduino IDE which has been capable of keeping our testing code neatly organized in one file. After using it for a while we decided to use VSCODE. The reason as to why we are deciding to make that change is because Arduino IDE is slow. It takes approximately 10 seconds for us to compile a small testing program that has a length of about 120 lines, and we can't imagine how long it would take to run all our code when implementing all our classes and functions.

There are many advantages to VSCODE, but the main advantage is that it is very fast, it has many integrated programs, it is cross platform, it is capable of running any programming language that we choose (we are choosing c++ for our embedded programming), and we can keep everything organized very neatly. Another piece of software that we are going to use is going to be GitHub. GitHub is going to be particularly important because it pairs well with VSCODE in the sense that it can have seamless integration, which is capable of cloning repositories, push/ commits, create and manage branches, view pull requests, resolve merge conflicts and more. It is also quite easy to collaborate, easy to have version control, integrated terminal, easy to clone repository, and the most important, automatic sync with GitHub. In an overall sense, using VSCODE and GitHub is a very powerful combo that is used by a lot of developers in the real-world industry, especially for version control, collaboration, and streamlined coding workflows.

7.1.1 State (Activity) Diagram

The activity diagram illustrates the process of managing a breastmilk bag's data from the moment our device is turned on to the point when the bag is used and removed from the inventory. The sequence begins with the activation of our PUMPED device. Once the device is on, a breastmilk bag is weighed using our Load cell. The weight measured by our load cell will then be displaced onto our OLED SSD1309 screen. At the same time, the system will send the weight data to our connected mobile application, where it will be stored and can be viewed by the user as part of their inventory. With a click of our main button, our device will then print out an adhesive thermal label that is capable of

withstanding freezing temperatures. Withing our mobile application, the system will continuously check whether each bag that is in the database is nearing expiration or not. If the bag is found to be closed (2 days) of expiration, the user will receive a notification prompting for further action. Regardless of whether the bag is expiring, once it is used, the user will be capable of removing that specific bag from its inventory, which then completes the process.

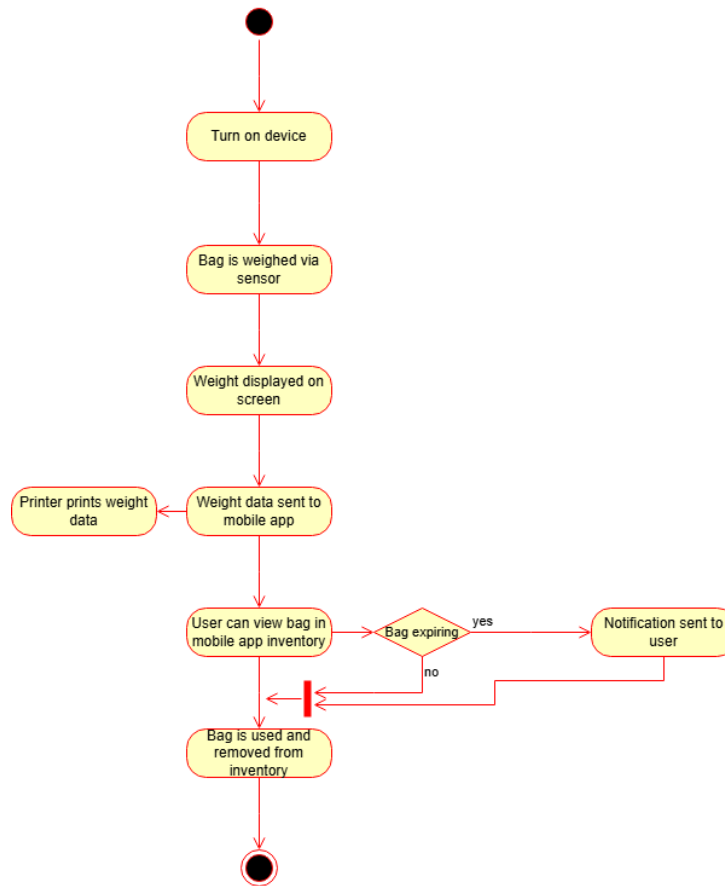


Figure 21 Software Activity Diagram

7.1.2 Use Case Diagram

The Use Case Diagram demonstrates the behavior of the ESP32, thermal printer, and our associated mobile app. The three actors in this diagram are the user, the ESP32, and the app. The user simply places the bag onto the scale and pushes the print button. This diagram illustrates that only this input is required from them and everything else is managed automatically.

The use case diagram also shows that the ESP32 is an actor in this software flow when it processes the bag's weight and sends it to the app. Its interaction with the other actor, the app itself, is also depicted here when it sends the data to the app. The application finishes

the software flow by updating its backend with the new bag information and sends auxiliary data such as the user's name back to be printed onto the label along with the weight data itself.

Overall, the software flow for this project when viewed from a high level is relatively straightforward. This use case diagram also demonstrates the usefulness of this device in general since the end user, most likely a parent, only has to take one action and everything else leading up to the generation of the printed label takes place automatically.

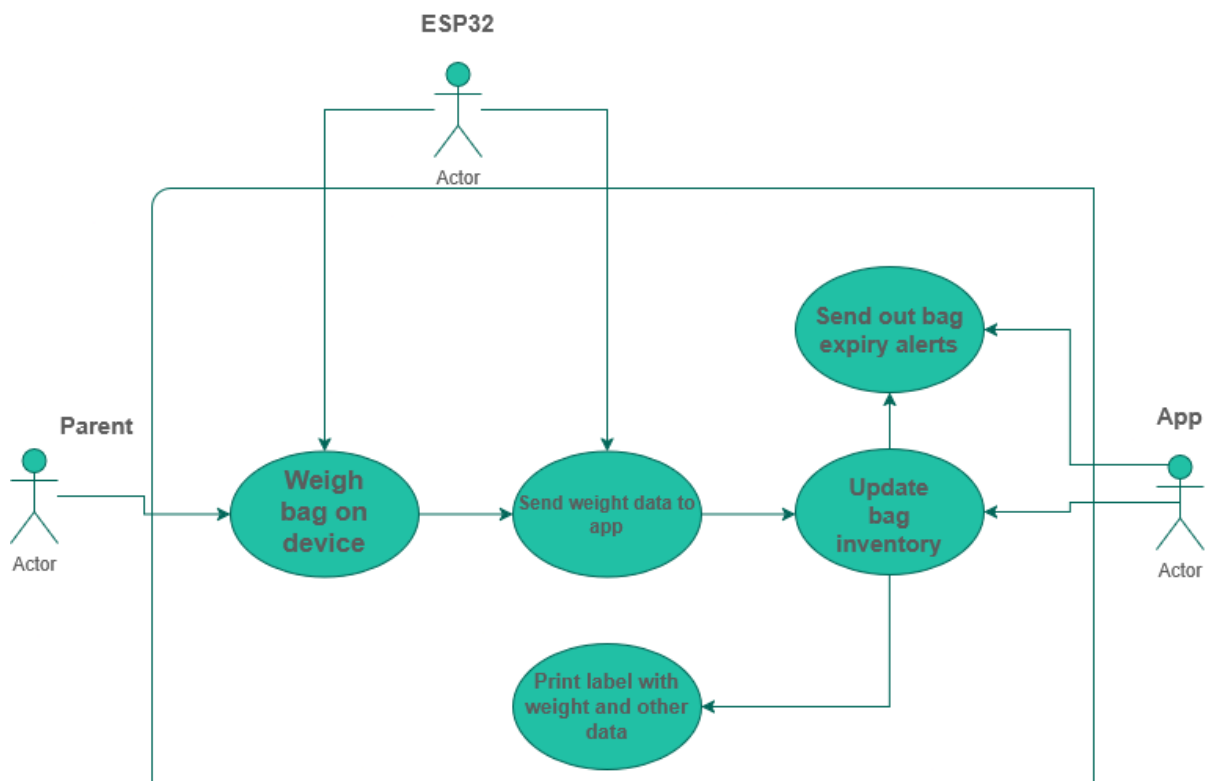


Figure 22 Use Case Diagram

7.1.3 Class Diagram

A class diagram is a type of structure diagram that uses Unified Modeling Language (UML) to represent the blueprint of a system by showing its classes, attributes, methods, and the relationships between them. In the case of our case diagram, before we wrote our actual code that we used for our project, we used a testing code that allowed us to connect all our individual parts for testing so that we could record our mini demonstration video. Our testing code is simple, we have one single main program that contains all the defined parts and their libraries that correspond to the scale, the SSD1309 display, but it uses a different library, a more universal library that is contained in Arduino IDE and contains all of the required components that we can use. We also have the printer library so that we can connect the printer to the scale successfully. A button function to use so that we can use and print when pressing and tare. All of the classes that are following are those of the functions for the scale, display, printer, and button function that each specifically will use.

There is an aggregation that consists of the scale along with the display with the main program and printer which means that they wouldn't work independently because they need each other's information in order to get the resulting information. But they can exist independently within themselves when coded separately. The button on the other hand is unable to work without connecting the other components and setting them up with a specific task. In the case of our button, it is used for taring, holding the value that is being weight and projected on the scale and then it is printed off of the printer.

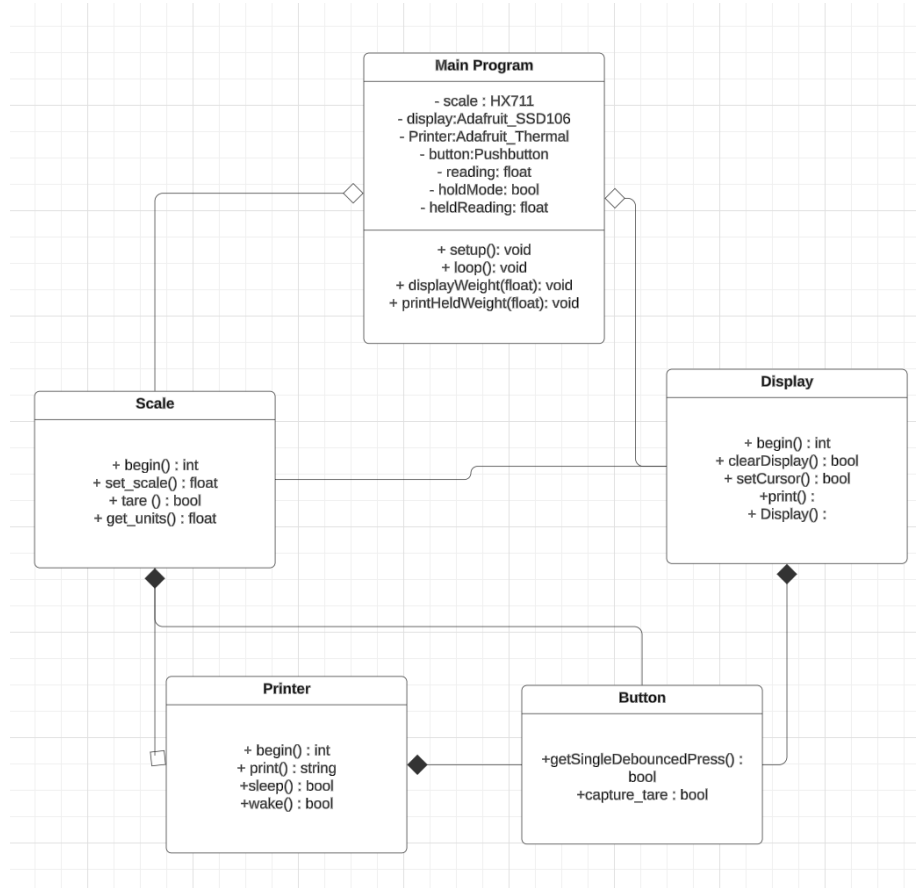


Figure 23 Class Diagram

7.2 Mobile Application Software

There are many different options that we could target when choosing how we are going to design our mobile application. One of the most important decisions is what operating system we are going to work with. There are mainly two options that we can choose between, Android and Apple. After doing a lot of in-depth and time-consuming research we decided that we would stick with Android. The reason as to why we decided to go with this option is because there is statistically a better and smoother experience when working with this operating system. Additionally, Android offers greater flexibility when it comes to wireless communication, which is critical for our application that needs to interact with external hardware and sensors. We also wanted to keep our app minimal and focused on

core functionality, ensuring every feature works properly without unnecessary complexity. This made Android an ideal platform, especially when paired with Android Studio, the official IDE for Android development. Android Studio provides access to a wide range of libraries and tools, like emulators for various Android devices and sensor simulation capabilities that make it easier to assess and iterate without needing physical hardware. We chose to develop the app natively using Kotlin, which is now the modern standard for Android development, offering concise syntax and advanced features like null safety. While alternatives like Flutter and other cross-platform frameworks were considered, we opted not to use them because they often introduce additional layers of abstraction that can complicate hardware-level communication and may lack native-level support for some Android-specific features. Focusing on Android exclusively allows us to take full advantage of the ecosystem and ensure seamless integration with all intended functionalities.

7.2.1 Mobile Application Software

The first-time app login gives an idea of what the software flow will be for the mobile app associated and integrated. It gives an indication of how the users could end up navigating the application to see the breastmilk bag data that is being transmitted to and from the ESP32 via the BLE connection. The users will have accounts associated with the device so that multiple people can use a single device and have their relevant personal info tied to the printing of the label so additional info can be put onto the printed label if needed. We can ensure verification of their accounts through authenticating their e-mail to further secure the mobile application and its use since it is so vital. All of this leads to a secure, useful mobile application that will hopefully reduce the burden on the users and make their lives easier.

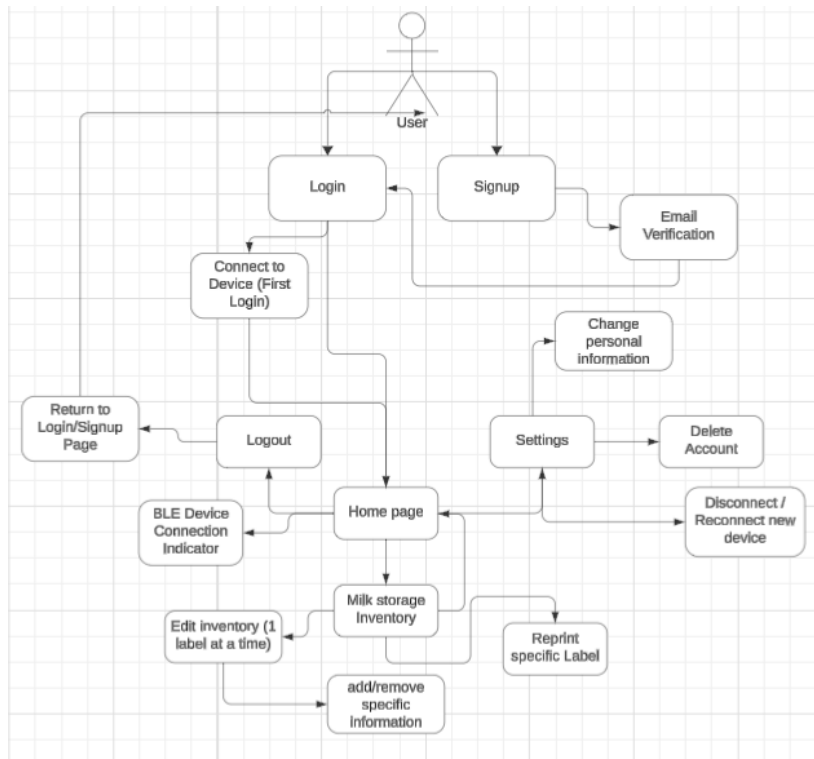


Figure 24 First Time App Login

The mobile app general functionality flowchart gives an approximate idea of how the software flow could look like for the users utilizing the mobile application associated with our device. The user having the ability to have a unique account tied to them on the device is important as it allows multiple users to utilize the device and still have their relevant data put onto the printer labels if it is required. Things such as child name and dietary information being printed onto the label can be important in households where there is more than one child and/or there are special dietary requirements that need to be tracked with regards to the breastmilk bags. The software flow displayed here helps facilitate an easy-to-use mobile application that will greatly help improve the ability of parents to properly track their children's nutritional needs.

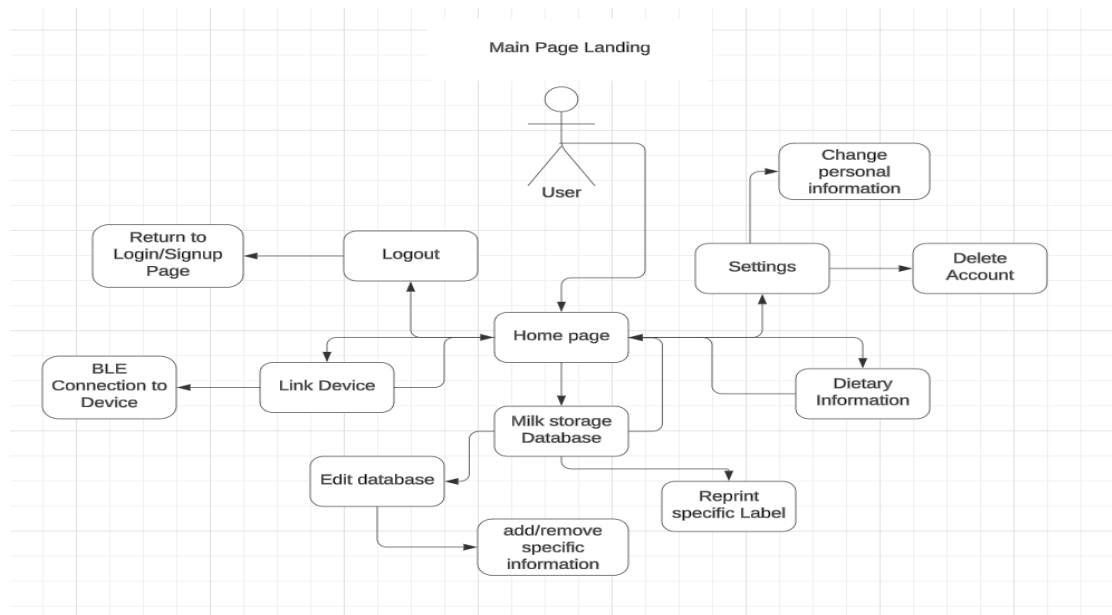


Figure 25 Mobile App General Functionality

All the following mockups have been created with Figma and UX Pilot incorporated.

The mobile app is an integral part of this overall project since the app will be associated with the automatic breastmilk bag label printer and contain the data from the user's actions. The mockups below indicate what the mobile app will likely look like so that the user can enjoy an easy-to-use mobile application. It also displays how we are utilizing the breastmilk bag data we are receiving and are displaying them in ways to inform the users. Additionally, the mockup on the right displays the BLE technology we used in this mobile application to facilitate communication between the app and the ESP32 microcontroller.

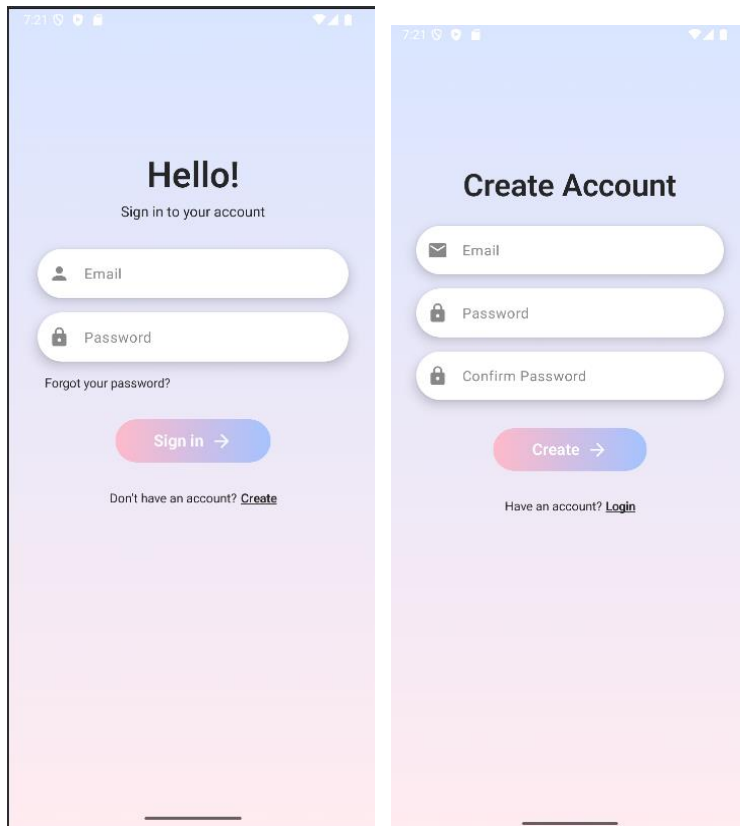


Figure 26 Mobile App – login + signup

These two mockups give further insight into how our mobile application will be structured for the user to easily navigate the statistics being tracked regarding their inventory of breastmilk bags. This is the key aspect and functionality of the app and as can be seen these are very simple to understand and deliver the important information quickly to the parents, so they do not have to spend too much time thinking. Additionally, it can be seen in the mockup on the right that we are encouraging further customization of the printed labels by giving the parents the ability to select dietary information concerns that will eventually appears on the automatically printed labels. Both things come together to encourage the use of this app among parents who are in need of a tool that makes their lives easier.

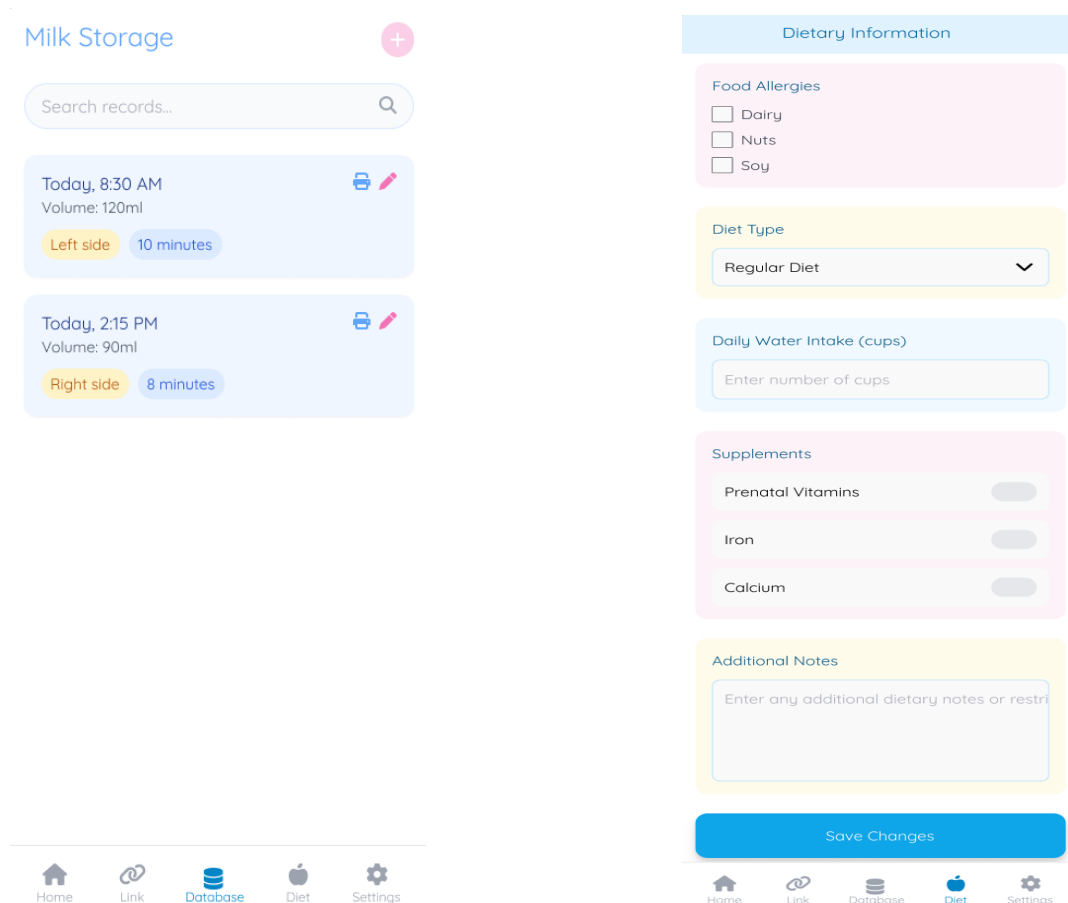


Figure 27 Mobile App - Data Base & Dietary Info

This last settings mockup further displays how user's accounts will be tied to the mobile app and by extension the device itself. This is very useful because it allows multiple users to use the same device (when it is not already currently in use) by associating their name and dietary information saved to their account so that the automatic label printer can print any kind of label needed. Users can update their info on the fly in case of any changes to the dietary or nutritional situations of themselves or their children making the entire device

very customizable. All of this comes together to make an application that serves the need of its users in a powerful and convenient fashion.

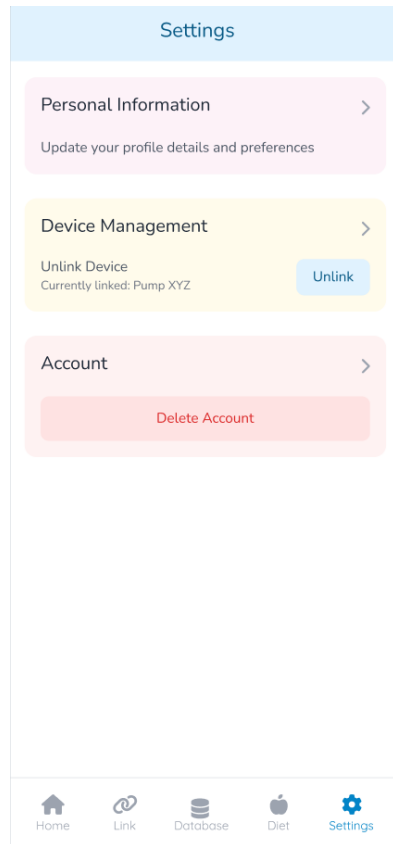


Figure 28 Mobile App - Settings

7.3 Efficient Power Consumption Software Design

Power consumption is a major concern of this project and not just for the hardware itself. The way in which we design our software and the type of software we use will have a significant impact on the power consumption of this project, so it is important that we optimize our designs to the most significant extent possible. The main aspects on the software side of this project where we can get the biggest energy savings are likely the printer behavior, the ESP32 itself, BLE communication, and load cell/HX711 use.

Efficient printer power consumption design:

- Utilize the printer sleep mode when no print jobs are incoming and the printer is idle.
- Structure user interaction flow to avoid unnecessary number of wakeups from the printer in sleep mode.

Efficient ESP32 power consumption design:

- Put the ESP32 into sleep mode when it is not taking or sending sensor measurements.
- Wake ESP32 upon print command button press or load cell sensor being utilized.
- Be mindful of how putting the ESP32 into sleep mode and waking it up can affect the BLE connectivity with the app.

Efficient BLE power consumption design:

- Dynamically lower the BLE range when users are very close to the ESP32 in order to reduce power consumption.
- Consider reducing the interval at which data is published while being mindful of the tradeoff of using less power for longer data transmission times.

Efficient Load Cell/HX711 power consumption design:

- Put the HX711 to sleep or power it down entirely when no weight readings are being measured.
- Consider powering down entirely when print job is being executed as prints and weighing are not occurring simultaneously

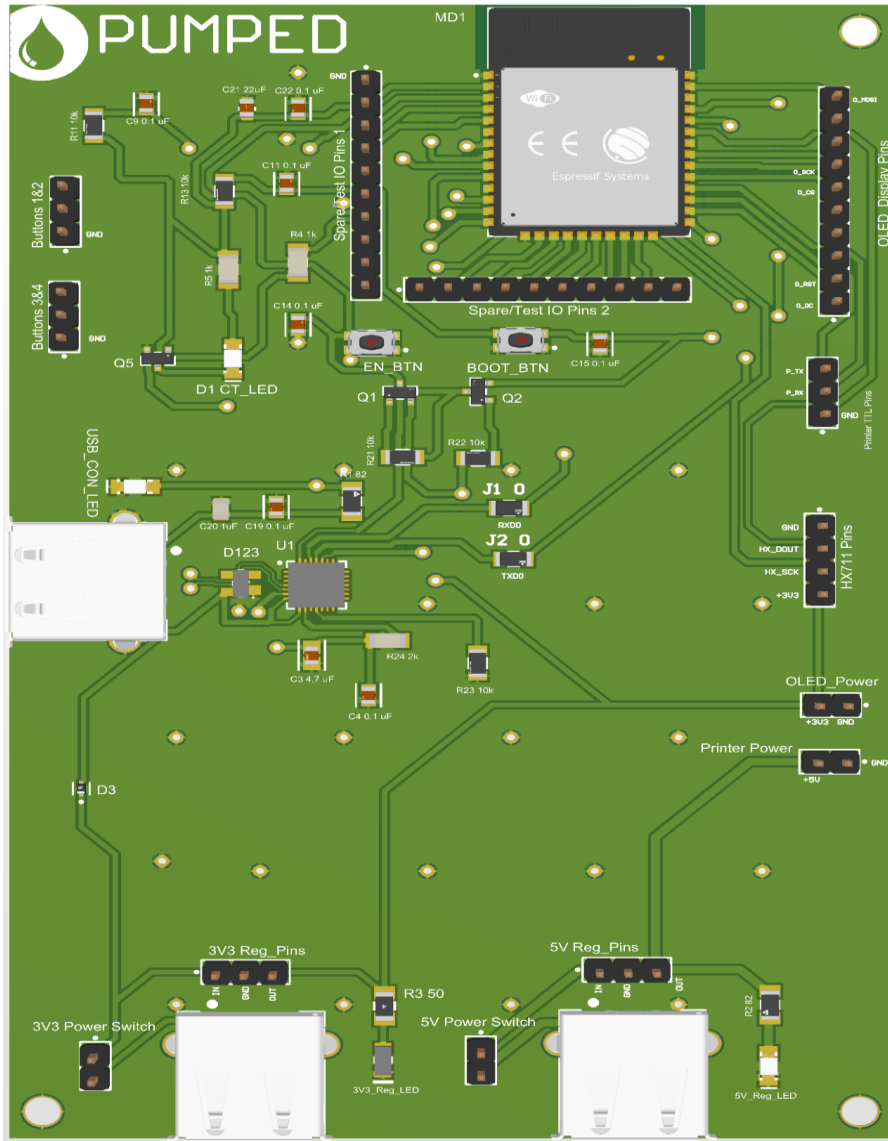
Overall, any optimizations to power consumption need to take into consideration when high power consuming components are not in use and can be put to sleep or powered down entirely.

Chapter 8: System Fabrication/Prototype Construction

Once we fine-tuned our schematics in Altium and got some more feedback at the beginning of SD2 on PCB design, our main focus was to order PCBs. The plan was to split the PCB into multiple parts where the main PCB will have the MCU, USB UART connections, Bluetooth module, and connectors. The smaller PCBs will hold the components tied to the voltage regulators. This system will ease the troubleshooting process. In case there is an issue with one PCB, the other functioning PCBs won't be affected. To increase reliability and save time we ordered multiple PCBs to bypass the long shipping times.

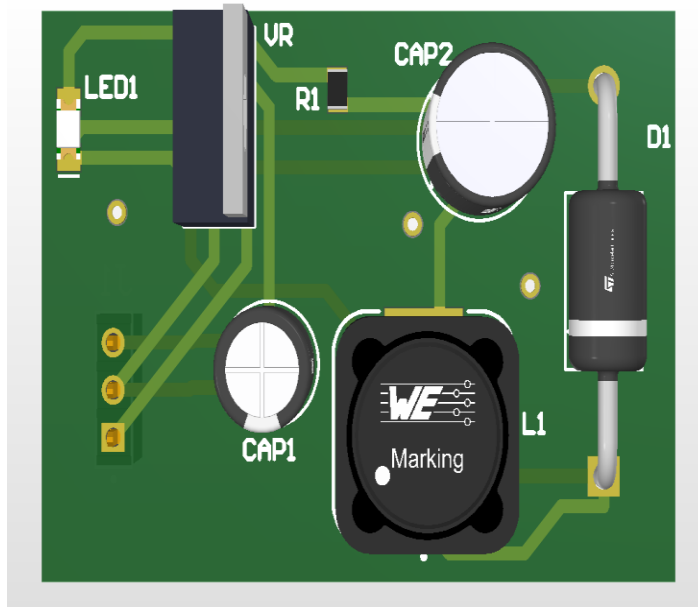
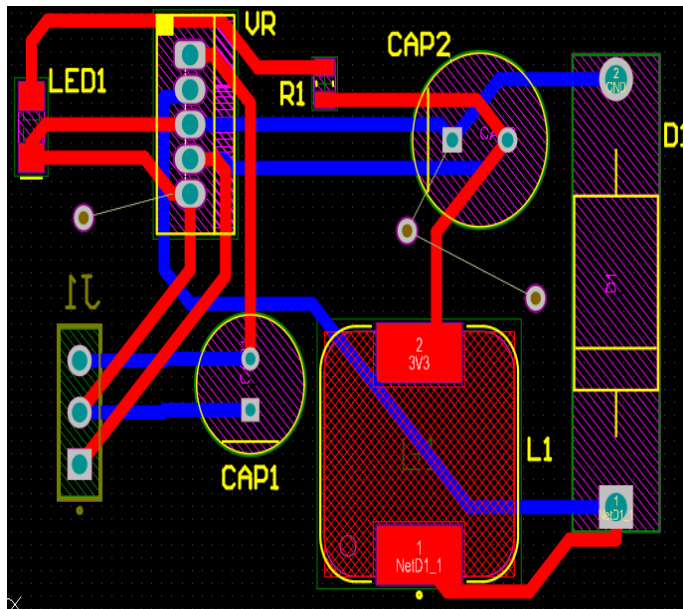
8.1 MCU PCB





The MCU PCB includes the ESP32 microcontroller, USB-to-UART bridge (CP2102), voltage regulator headers, and connectors for other subsystems. The board layout prioritizes short trace lengths and functional grouping of components. Both USB-A connectors are placed along the bottom edge of the board. The left-side USB-A port is used for programming the ESP32 and is placed close to the CP2102 to reduce the chance of signal degradation. The ground plane beneath the ESP32 antenna was removed on both the top and bottom layers to reduce RF interference and improve Bluetooth signal strength. High-current traces (e.g., those supplying power to the thermal printer) were widened to 32 mils to support the expected 2 A load. Mounting holes were added at several corners to secure the PCB inside the enclosure and prevent shifting during use.

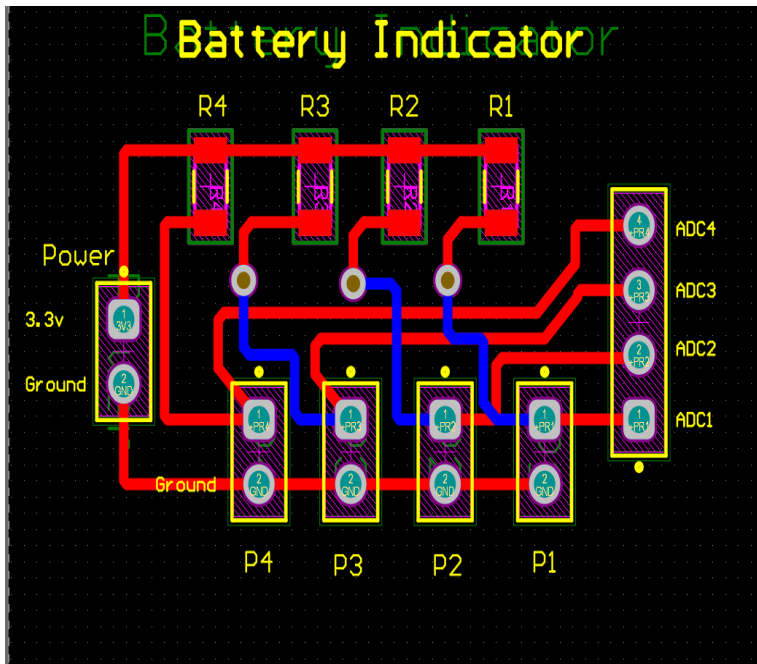
8.2 Voltage Regulator PCB

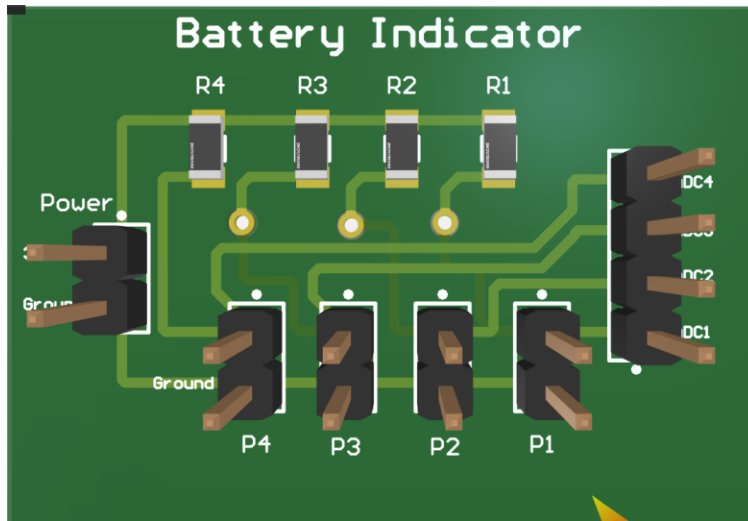


The voltage regulator PCB was designed as a separate module to isolate power circuitry from the MCU and minimize noise. This board contains a switching regulator to supply 3.3 V to different parts of the system. Component placement was done to keep high frequency switching traces short and separated from analog signal paths. Capacitors were placed near the regulator outputs to help with voltage stability. The board includes an indicator LED for verifying a closed circuit when testing. By separating the regulators from the main PCB, troubleshooting and replacement is simplified if a voltage regulation issue arises.

8.3 Battery Indicator PCB

We added the battery indicator in SD2, this was a great opportunity for us to create a well-rounded portable device that had the capability of being rechargeable without any concerns. We ran into an issue on how the consumer was going to determine what the battery percentage was going to be, but we knew that the battery bank was capable of illustrating the battery through 4 LED lights. Dr. Weeks and the group came up with an idea. Using Photoresistors to determine the battery percentage based on the LEDs on the battery bank. The way photoresistors work is that the material has high resistance when in the dark, and lower resistance when in light. We are going to sync this up with the ESP32, and we are going to be able to determine the battery amount based on the number of LEDs (out of 4) that are on and off.





The battery indicator PCB uses a voltage divider and LED driver to visually display battery level. It includes four LEDs that represent a rough percentage of the battery's charge. The board is connected to the main system via headers. Its compact design allows it to be positioned without interfering with the device's other mechanical or electrical components. This PCB does not handle significant current and is only used for low-voltage status indication

Chapter 9: System Testing and Evaluation

9.1 System Testing

The process to test all our major components involved many small steps until ultimately, we could integrate each system together to get the final testing product. Our method for testing was to take a system and get it to function with our proposed specifications and once that system was made to operate on its own a bigger step taken to create code that would allow each component to function as a simplified version of our final product.

9.1.1 Hardware Testing

9.1.1.1 Thermal Printer

The first component of the PUMPED design that we tested was the thermal printer. We first began by testing the thermal printer with a power supply. The thermal printer's voltage operating range according to documentation is 5V to 20V. We found that the optimum voltage for operation was 13 volts because the printer showed a solid green ready light. Next, we tested the printer's operation with sample code to produce a printed test sheet. It took a few iterations of testing to find the correct baud rate, code, and voltage setting. Once the printer was set up correctly the printed output could finally be produced.

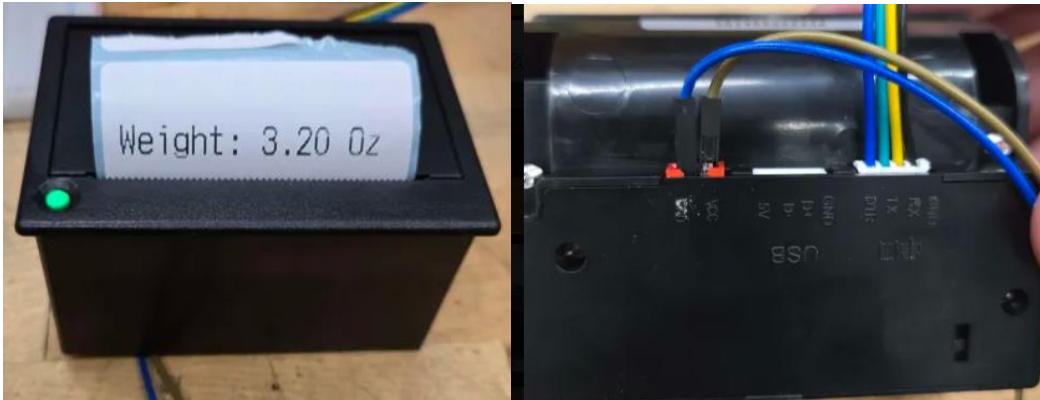


Figure 29 Thermal Printer

We found that to get a clear print of lettering from the thermal printer, a voltage of 20V was required, much greater than our expected voltage requirement. This setback led us to buying a second thermal printer that had a 5 Volt input requirement. The new thermal printer operates efficiently and produces a clear output at 5 volts and draws a maximum current of 2 amps. This printer required us to modify the baud rate to 9600 or else gibberish would be output.

9.1.1.2 Display

The OLED display we selected uses an SPI communication protocol, which required us to connect 7 pins to the ESP32 for it to function properly. The OLED display operates best at 3.3 volts and has a low current requirement. Once connected to the dev board little tuning needed to be done to get a functioning output on the display.



Figure 30 OLED display

9.1.1.3 Load Cell and ADC

The load cell we used for testing purposes is rated at 10kg, well above the max weight we believe to be imposed on the cell. Four wires are connected from the load cell to the HX711, which acts as an analog to digital converter. The load cell also is supplied with two platforms that serve as a base and a weighing platform. The HX711 is then connected to the ESP32 by four wires that function as power, ground, clock, and data, inputs. Once connected there was one major calibration procedure that we had to follow through software so that the load cell would work properly. Once calibrated the load cell would have function as precise scale constantly running a visual weight reading on our IDE's serial monitor.

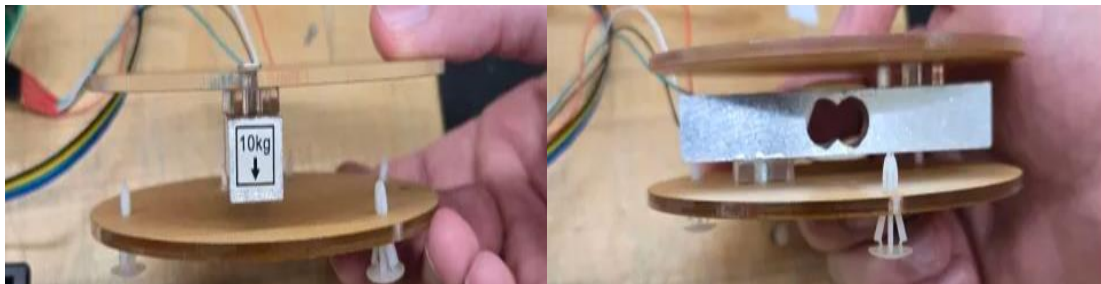


Figure 31 10kg load cell

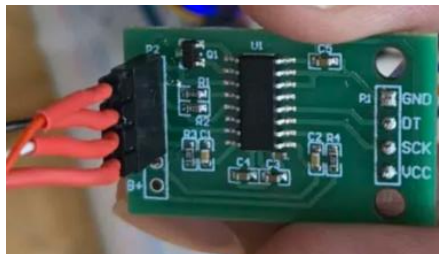


Figure 32 Analog to Digital Converter (HX711)

9.1.1.3 Power Source

Testing our power supply was for the most part straightforward. We started by testing the port that would be supplying power to the thermal printer and as expected the printer functioned perfectly with a clear printed thermal sheet. Next, we connected the ESP32 dev board with the Load cell and HX711 attached along with the OLED display, into the other USB A port, and found that the power supply is more than capable of powering all our major components.



Figure 33 ROMOSS Power Bank

Figure 9.5 Power Supply

9.1.2 Software Testing

9.1.2.1 ESP32 Microcontroller

First and foremost among the software testing steps was actually testing the likely microcontroller we were going to use. Through our research we discovered a plethora of options that met our criteria for this project. Ultimately, we have settled upon using the ESP32-WROOM. Testing the ESP32-WROOM included buying an ESP32 development board and connecting it to a laptop in order to program the microcontroller. We tested the ESP32 microcontroller by using the Arduino IDE to write C++ code to program it. First we wrote a basic program to ensure the ESP32 board was working as expected by blinking LEDs on it. From there we moved onto testing much more specific things with regards to how it would work in the project.

We further tested the ESP32 by seeing if we could successfully compile and upload code to the ESP32 that utilized its serial pins including TX and RX. Upon confirming that the ESP32 was programmable with this code we were ready to move onto the next phase of testing.

9.1.2.2 Load Cell/HX711

We tested the Load Cell/HX711 in conjunction with the ESP32 in order to calibrate the load cell that is acting as our digital scale. We did this by taking something that had a known weight and placing it onto the load cell to be weighed. We divided the weight reading by the known weight to get our constant calibration factor that we went on to use as a variable in our code. With our calibration factor known we could continue to write the code to actually weigh other objects and ensure the readings were accurate. We were able to verify that after setting the calibration factor that the load cell was weighing the items we placed onto it accurately.

9.1.2.2 OLED Display

The next step in the software testing process was to test the OLED display we had acquired which utilized an SPI connection with our ESP32 microcontroller. We wrote code to initialize the OLED display and ensure the hardware was in working condition. Once we verified that the OLED display was working in conjunction with our ESP32 we went on to write code to display the weight readings.

First, we did not take in the readings and simply wrote code to display the weight readings in our desired format. Once we verified that the OLED display could show the values in the way we wanted it to onto the display we went onto actually properly integrate it with the existing ESP32 and load cell connections. We wrote code that took in the weight readings from the ESP32 and updated the OLED display anytime there was a new update in the weight reading. This code was successfully tested, and we had a working OLED display for our project.

9.1.2.3 Thermal Printer

By far the longest and most intense software testing was on the embedded thermal printer working with the ESP32. This testing went through several stages with the printer being tested in multiple ways. We tested the serial connections, how to properly send the print command from the ESP32, the printer's compatibility with external libraries, and more.

The first thing that had to be done was ensuring our code was right for the configuration of the serial TTL connection. This proved to be problematic for a brief time in our testing.

We had to correctly identify and test opening the serial connection at the appropriate baud rates. We were initially confused on the correct baud rate for the printer, so it was printing out gibberish data onto the receipt when we attempted to print anything. After some time we found the correct baud rate of 9600. With this step taken care of we could move onto ensuring we were sending the print commands in the correct format from the ESP32 to the printer.

Next, we tested the ESP32 with the printer by trying out different ways of sending the print commands to the printer. We were able to successfully able to do this with generic serial print statements and then further enhanced it print in our desired fashion. This proved troublesome at first during our testing with us running into issues of the printer not stopping its print in a loop, but we were able to resolve this. We also wrote code to ensure that the printer would send print commands on a button press and were able to successfully do this.

With our printer now printing on command we were able to move onto enhancing the prints.

One important aspect that had to be tested with regard to the printer and its interaction with our software was if it was compatible with third party libraries we wanted to use. The main external library we wanted to use was the Arduino Thermal Printer Library. This library would allow us to format the actual printed text in terms of size, positioning, font, etc. onto our labels. We tested the compatibility by programing the ESP32 to use the Arduino Thermal Printer Library's print commands.

We were able to test this successfully on our first try and the print quality came out good. We moved onto testing the formatting options available in the library and positioned the text in the center of the label while also making it much larger. This also worked on the first try and the printer was working entirely as expected. Overall, the printer software testing went well despite the various hiccups along the way.

9.1.3 Overall Integration

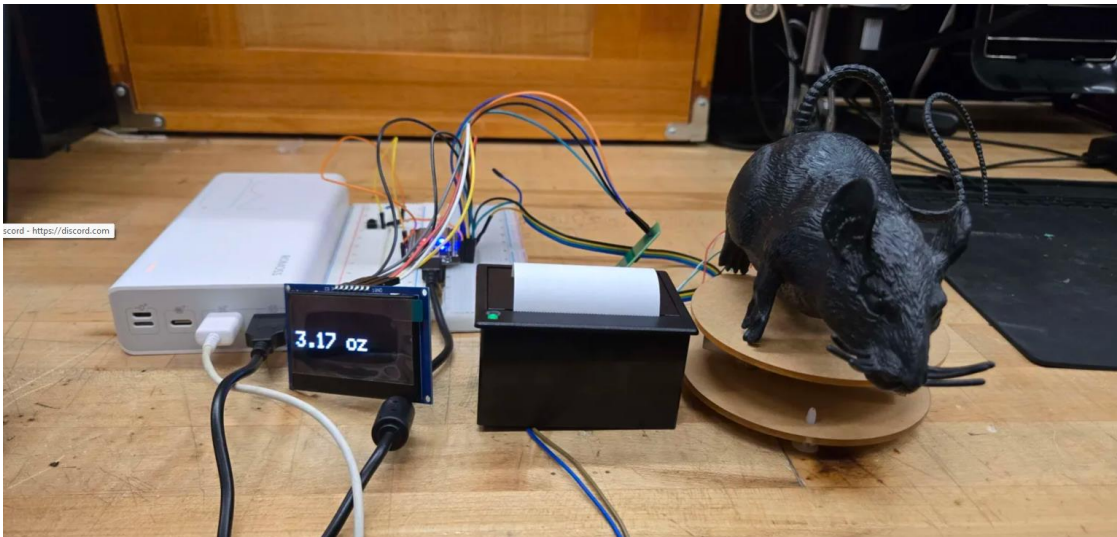


Figure 34 Overall System - Demo

Once each component was individually tested and able to function on its own, we developed software so that each component would communicate with each other in a meaningful way so that we could simulate what our final product should look like. We set up the system so that the load cell will continuously display the weight and once a weight is set onto the scale the user of the system can press a button, located on the bread board, that will send a signal to hold the value on the display and at the same time the printer will receive a command to print. The weight in ounces will be printed on the thermal printer.

9.1.4 Plan for SD2

Our plans for the continued system testing of our project going into senior design 2 are primarily centered around integrating the communication between our ESP32 microcontroller and the mobile application. On the hardware side, we were also focused on testing and debugging the printed circuit board we design and use for the project.

The system testing of our BLE connectivity that is going to bridge the data transmission between the ESP32 and the mobile application is of utmost importance. BLE, like any wireless protocol, can prove somewhat finicky so it is important that we rigorously and thoroughly test the use of the mobile application with its BLE connection to the ESP32.

We were on top of our printed circuit board design and eventual testing once we had our hands on it. There were minor shorts or other issues with the printed circuit board upon delivery that we had to find out if we could resolve ourselves or have to re-order the printed circuit board. This is an important distinction as the printed circuit board lead time can be up to 2 weeks if there is something wrong, which makes efficient and accurate testing very important to the success of our project.

Chapter 10: Administrative Content

For our Senior Design PUMPED project, we have carefully planned several key aspects to ensure smooth development and execution. Our budget and financing outline the estimated

costs and funding sources needed to complete the project efficiently. The bill of materials (BOM) will provide a detailed list of components required for construction of our project, making sure that we acquire all necessary parts on time. Additionally, we have established a timeline with project milestones to track progress during Senior Design 1 (SD1) and Senior Design 2 (SD2), helping us stay on schedule. To ensure a fair and efficient workload, we created a work distribution table, assigning each team member primary and secondary responsibilities based on our strengths. By organizing these aspects early, we aim to complete the project successfully while managing time resources effectively.

10.1 Project Budgeting / Cost

Pumped project is sponsored by Victoria and Matthew for a total of up to \$500. We are still brainstorming the parts for this project, but some of the items that we are going to be using are listed below. All prices here are general, but we are confident that the vast majority of the project will be covered by the \$500 in sponsor funds we are receiving from our two sponsors Victoria and Matthew. Table 10.1: Bill of Materials

Table 36 Bill of Materials

Component	Subsystem	Quantity	Unit Cost	Total Cost
ESP32 (Espressif)	All	1	\$12.99	\$14.78
LCD Screen	User Interface	1	\$16.99	\$18.09
Battery Bank	PSU	1	\$26.99	\$28.75
Thermal Label Printer	Printer	2	\$49.93	\$99.86
DFRobot Thermal Printer	Printer	1	\$53.00	\$53.00
Thermal Label Paper	Printer	4	\$9.99	\$10.95
Load Cell / Wheatstone Bridge + Hx711	Scale	1	\$9.99	\$10.95
Physical Buttons	User Interface	1-kit	\$7.99	\$8.95
More components are TBD	PCBs, Device Frame, etc.	TBD	TBD	TBD
Total	\$245.33			
Budget Provided by Sponsors	\$500			

10.2 Project Milestones

There are several meetings and events that will be occurring in the next few months. These meetings are required for a successful project so that we do not fall behind on our project and risk getting ourselves into a bad position. The list will be updated accordingly, but for now, the items listed have been planned for. A key part of our project milestones that may be different from some other groups is that we have a sponsor attached to our project. Since

we have a sponsor attached to our project, we will likely have to regularly check in with them on all of our project milestones. After meetings with them we may or may not have to engage in modifying our work depending upon their feedback. We anticipate making weekly progress on our project milestones and are committed to attempting to maintaining that pace as we believe it will help stave off issues such as procrastination which may be an issue on this kind of project. We also are planning on committing to the weekly milestones in order to take advantage of the mentors we have around us. We would like to ensure that by sticking to a preplanned timeline that we are able to arrange meetings or consultations with subject matter experts among our professors and faculty or our sponsors at the appropriate times. The project milestones are therefore important not only for us but also for the people around us who can plan their own schedules so that they have time to assist us. Table 10.2: Project Initialization Milestone

Table 37 Project Initialization Milestone

Project Initialization			
Task	Start Date	End Date	Description
Formation of group	1/06/2025	1/08/2025	Eric Hernandez - CPE Jimmie Rawls - EE Kevin Fernandez - EE Agustin Rodriguez - CPE
Brainstorming Ideas	1/06/2025	1/09/2025	Brainstorming for possible project ideas that we can pursue.
Sponsor Meeting	1/16/2025	1/16/2025	First meeting with the sponsor. Its purpose for introduction of us, also to talk a little bit about goals for the project
D & C Document	1/19/2025	1/24/2025	First 10 pages of D&C to declare the project's purpose and details.
D & C Group Meeting	1/29/2025	1/29/2025	Meeting with SD faculty that will determine whether the project is approved or not.
Mid-point, 60-page milestone	1/31/2025	3/24/2025	Halfway point of the Report
60-page Group meeting	3/14/2025	3/14/2025	Meeting about the 60-page milestone for checkup.
Sponsor Meeting	4/25/2025	4/25/2025	In person meeting with the sponsor. Its purpose is to demonstrate most of our physical components working together.
Final Document & Mini Demo	3/24/2025	4/22/2025	Completion of report and mini-Demonstration.

Table 38 Project Fabrication Milestone

Project Fabrication			
Task	Start Date	End Date	Description
Purchasing / Selecting Components	2/01/2025	3/01/2025	Gathering all of the required components for the project and creating a BOM.
Designing the System	2/20/2025	4/01/2025	Putting together and modifying the schematics for the PCB
Testing	3/20/2025	4/15/2025	Testing different components to make sure that they are compatible and work together.
PCB schematic Design	4/1/2025	4/21/2025	Designing the schematics for our PCB
PCB Design	4/29/2025	5/09/2025	Designing the PCB design, passing it through with our sponsor and placing the order for it.
Mobile Application Design	4/30/2025	5/07/2025	Designing our mobile application
PCB Developing	05/23/2025	6/01/2025	After getting our PCB, we will be soldering and putting it together with our other components
Mobile application Developing	5/12/2025	6/20/2025	Developing our mobile application and wireless connectivity.
Mobile Application Testing	5/12/2025	7/1/2025	Testing of our mobile application with the Physical components for wireless connectivity.
PCB Testing	6/01/2025	6/27/2025	Testing the design as early as possible to make sure it's working properly. We will have to order a new one if there's an issue.
Completion of our mobile App	4/30/2025	7/1/2025	Mobile app works correctly with physical device.

Completion of Prototype	05/05/2025	7/18/2025	Completion of the prototype and making sure it works without errors.
-------------------------	------------	-----------	--

10.3 Distribution of Work

The goal for our distribution of work is to be able to make it easy and fluent for all our group members. Some of the main key points that we want us to achieve when distributing the work is to ensure we have a balanced workload across all team members, we want to leverage each of our individual strengths so that we put all that energy in what we are good with already, we want to promote accountability and ownership of the roles, encourage collaboration, track progress, and support professional growth because this is the final step before graduation and going into the real world. We all have different classes that we are taking aside from senior design, so the distribution will allow us to have a successful project.

Table 39 Distribution Of Work

Distribution of Work	
Team Member:	Part of the project to be responsible for:

Computer Engineering	Responsibilities
Eric Hernandez	MCU Selection and Implementation
	Software Design and Implementation
	User Interface Selection and Implementation
	Mobile Application, Front End Implementation
	Mobile Application, Database Implementation
	Administrative Documentation

Electrical Engineering	Responsibilities
Jimmie Rawls	Voltage Regulation, Load Cell, ADC
	House of Quality
	Motivation and Background
	PCB design: Ordering, Soldering, Troubleshooting, Testing

Computer Engineering	Responsibilities
Agustin Rodriguez	Thermal Printer Selection
	Wireless Interface Protocol Selection and Implementation
	BLE Connectivity Testing and Debugging
	Mobile Application, Back End Implementation
	Third Party Library Selection and Integration
	MCU Embedded Programming

Electrical Engineering	Responsibilities
Kevin Fernandez	Power Supply - Battery
	Hardware Block Diagram
	Main PCB Schematics- MCU, USB-UART (CP2102), Peripheral Connectors.
	PCB design: Ordering, Soldering, Troubleshooting, Testing

Chapter 11: Conclusion

In Senior Design 1, we made significant progress on this sponsored project to automatically print labels for breastmilk bags. We met regularly as a team and communicated with our sponsors effectively which resulted in us getting a successful prototype on a breadboard with very few issues along the way. We believe that our regular meetings and mindful communication were key to our success this semester. This was not something we arrived at by coincidence but was an intentional strategy on our end. This is something we are being conscious about carrying into Senior Design 2 to ensure continued success as the project reaches its apex and conclusion

All our major hardware components were successfully integrated and tested together on the breadboard including our ESP32, OLED display, battery bank, Load Cell/HX711, and embedded thermal printer. With the exception of the thermal printer all of these components worked almost immediately. The ESP32-WROOM in our breadboard prototype worked seamlessly, powering the logic for the rest of the project and we anticipate no issues working with it going forward. We feel similar regarding the battery bank and OLED display with both of those components proving to work exactly as we anticipated with no obvious possibilities for issues further down the line. The load cell/HX711 acting as our digital scale worked very well in our testing this semester, and we have remained vigilant about whether the load cell is maintaining its accuracy. If we saw any issues with the weight readings, we recalibrated the scale by rewriting the calibration code.

The major hardware component that presented the most issues was the embedded thermal printer. The original embedded thermal printer presented unusual issues when we realized that the company we acquired the printer from had sent us the wrong printer. We had to rectify this by exchanging communication with the company based in China and were able to somewhat progress when they sent us the partial dataset in the project because the operating range for the voltage was far bigger than what we originally thought we were getting. We originally believed we would be getting a printer that operated in the range of 5 - 9 V but instead were sent an embedded printer that operated in the range of 9 – 24 V. This was not acceptable as the printer is the component in this project that has by far the highest power consumption, and we were seeking to limit that. To resolve all of this we acquired a new embedded thermal printer that operated in the 5 – 9 V range like we had originally anticipated. The testing with this new printer went very well and we had no issues on the hardware side nor any issues on the software side once we became familiar with how this specific printer worked. This embedded thermal printer is effectively the engine of our project so it was an important milestone in this project that we were able to resolve the issues we had with it and find an appropriate printer that can act as the heart of this project. In the interest of being abundantly cautious we opted to buy a backup of this specific embedded thermal printer so that if our current embedded thermal printer has some kind of serious firmware or hardware issue that may not be easily fixed, we can replace the engine of this project with ease.

One very important thing to note is that in the course of doing the breadboard prototype while we did integrate the hardware entirely, we have not integrated any of the android mobile applications. Our efforts were focused primarily on getting our hardware to be integrated fully and with that now done we can move full speed ahead on working on the mobile application. This is something that we are sure to fast track in order to begin testing the cross-functional use case of our software and hardware ends of the project interacting with each other. This is important because now we consider the area of this project that likely presents the most potential issues are the areas in which there is required communication between our ESP32 microcontroller and the mobile application. We want to do all we can to optimize the communication between these two things and reduce any latency or bad connections when they interact with one another. This is not something we can necessarily plan on as the possible issues that can occur are likely to be very specific

and some cases will not be able to be prevented ahead of time with hardware or software solutions until we know what they are. The very next step in this project aside from PCB development is getting a barebones app put together so we can begin testing the communication between the app and the microcontroller.

Our software testing went successfully throughout the course of Senior Design 1. With regards to the actual software testing all the code we wrote for our sensors and the printer itself ended up working almost exactly as we wanted them to. One thing about the code we wrote for the actual printing of the labels that we are still trying to figure how to possibly improve on is further increasing the size of the text. We believe that at this moment in time we have reached the max size we can print the text in with our currently known methods, so we are continuing to see if there are ways to further increase the size of this text by another 15-20%. However, this is not totally critical, and we are not concerned if the text size cannot be made any bigger as we believe it to currently be adequate. The code we wrote for all our other sensors and their interactions with the ESP32 are already very optimized and likely have little room to be iterated upon. They simply work as we want them to and barring any unforeseen issues that come up, we did not have to work on them too much more and the bulk of our efforts were focused on our mobile application and its integration into our existing software stack and codebase.

A major thing we are considering regarding our software setup is what tools we are going to continue using in Senior Design 2 as our codebase becomes increasingly complex and large in size. One specific tool we have some concerns about is the Arduino IDE that we are using to write embedded code for our ESP32. While the Arduino IDE is very useful in that sense that it is quite simple to use it also does lack a lot of quality-of-life features, depth, and overall has low performance with it taking quite a bit of time to both compile and upload code to the ESP32. We are concerned about this because at this point in time our embedded code for the ESP32 is only a few hundred lines long, but this project could end up easily having over a thousand lines of code depending on our exact implementation of the further integration from the mobile application with our existing hardware. We cannot be sure how much slower the Arduino IDE is going to become as our codebase continues to grow but it does present a point of concern if the upload times negatively affect our testing by making things proceed very slowly or even worse begin to have actual errors uploading. Furthermore, we were concerned about whether the Arduino IDE can handle some of the more complex BLE code we wrote to create the connectivity between our ESP32 and the client's mobile application, but this is ultimately only something we can discover by trial and error. If we do end up having issues with using Arduino IDE as our embedded integrated development environment, we anticipate switching to VS Code likely enhanced by PlatformIO. The PlatformIO environment is much more sophisticated and high performing but with an obvious drawback of requiring more setup from us which will slow down development time to an unnecessary extent.

As with any project ongoing in Senior Design 2 the area we perhaps have to be the most on top of and vigilant about is our printed circuit board (PCB) design and use. The printed circuit board is the most important aspect of this project to guarantee we have success. We double, and triple checked our designs with our sponsors and available mentors to ensure

we are setting ourselves up for success. We are focused on getting our first PCB designs ordered as soon as possible so that we can test them and re-order them if they do not work. More than any other component it is important we do this with the PCBs because the possible lead time on getting them delivered is up to two weeks. We were conscious of the fact that we could not afford to wait to work on the PCBs or else we would have been in a seriously adverse situation.

As we reach the end of Senior Design 1, we have begun the design process for our PCBs and hope to be able to order them right before or at the start of Senior Design 2. We feel confident that our printed circuit boards from a design standpoint will not require anything particularly complicated or difficult to achieve the performance we need. One thing we paid extra attention to with regards to the PCB though is ensuring our circuit facilitates ideal performance with our Bluetooth Low Energy (BLE) connectivity that we utilized in the project. We are heeding the advice of our mentors and ensuring that the components we are currently selecting for our PCB meet the criteria that has been set out for us. We are ensuring we are not picking components that are unnecessarily small, making them difficult to handle or solder onto the printed circuit board. We are also making sure that we select a microcontroller with the appropriate pin count and pin placement to meet the standards set for us by our mentors to prevent any issues.

Overall, we are very happy with the progress we made in Senior Design 1 and believe we are on track to have success with our project come the end of Senior Design 2. We believe we have made good progress in terms of where we are for the integration of our existing major hardware components. We are confident in the code we have written to serve as the foundation for the interaction between our sensors and the microcontroller that facilitates the physical label printing we are going to be doing so much of in this project. As a team we believe we have good systems in place in terms of ensuring regular progress via the regular meetings and great communication we are having with each other with regards to the state of the work in front of us. We made our plans for Senior Design 2 and are very confident with how we were able to execute them in a timely manner to ensure that we have a very successful Senior Design project to present to our sponsors at the end of next semester.

Appendix A: References and Disclaimer

Disclaimer: This idea is solely credited to our sponsors, Victoria Rodriguez and Matthew Aberman. They will retain full ownership of the final prototype, and any engineering design assets/materials created.

- [1] <https://walzeq.com/products/labeling/printers/specialty/breast-milk-labeling-system/#gsc.tab=0>
- [2] <https://a.co/d/7PHHq0T>
- [3] <https://www.themilkbank.org/donate-milk/>
- [4] <https://www.advocatechildrenshospital.com/health-resources/support-services/donor-milk-program>
- [5] <https://www.milkbanktn.org/guidelines>
- [6] <https://drtalbots.com/products/milkfresh-sensors>
- [7] <https://www.ti.com/product/MSP430FR6989>
- [8] <https://shop.pimoroni.com/products/raspberry-pi-pico?variant=32402092294227>

- [9] <https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf>
- [10] <https://docs.arduino.cc/hardware/nano/#tech-specs>
- [11] <https://docs.arduino.cc/hardware/uno-r4-wifi/>
- [12] <https://deepbluembedded.com/esp32-sleep-modes-power-consumption/>
- [13] <https://store-usa.arduino.cc/products/arduino-mkr-wifi-1010>
- [14] <https://lastminuteengineers.com/esp32-pinout-reference/>
- [15] https://en.wikipedia.org/wiki/Lithium-ion_battery
- [16] <https://poweringautos.com>
- [17] https://en.wikipedia.org/wiki/Lithium_polymer_battery
- [18] <https://tritekbattery.com>
- [19] <https://www.batterypkcell.com/news/advantages-and-limitations-of-lithium-polymer-batteries/>
- [20] <https://www.grepow.com/blog/li-ion-vs-lipo-battery-for-long-range-flights.html>
- [21] https://en.wikipedia.org/wiki/Nickel%E2%80%93metal_hydride_battery
- [21] <https://www.electricity-magnetism.org/electric-battery/nickel-metal-hydride-battery-how-it-works/>
- [22] <https://www.improvecn.com/articles/the-advantage-and-disadvantage-of-nimh-batteries>
- [23] <https://www.powerbankexpert.com/what-is-a-power-bank-power-banks-explained/>
- [24] <https://a.co/d/223JcOr>
- [25] <https://a.co/d/5vETS31>
- [26] <https://a.co/d/cVtkrib>
- [27] <https://a.co/d/8xn05uU>
- [28] <https://a.co/d/bmaObdq>
- [29] <https://a.co/d/igkT6El>
- [30] <https://bridgestudio.co/why-touch-screens-arent-always-the-best-choice-a-closer-look-at-the-ux-of-touch-screens-vs-real-buttons#:~:text=As%20a%20UX%20agency%2C%20we%20recommend%20considering%20all%20options%20before,intuitive%20and%20reliable%20user%20experience>
- [31] <https://www.autodesk.com/products/eagle/overview>
- [32] <https://www.kicad.org/>

- [33] <https://www.altium.com/altium-designer>
- [34] <https://www.hp.com/us-en/shop/tech-takes/laser-printer-vs-inkjet>
- [35] [SparkFun Load Cell Amplifier - HX711](#)
- [36] <https://resources.altium.com/p/cp2102-usb-uart-module-pcb-design-project>
- [37] [Electronics Manufacturing | IPC Standards](#)
- [38] [Complying with IPC Standards for PCB Design | Altium](#)
- [39] [IEC 62133-2:2017 | IEC](#)
- [40] [UL 2054 | UL Standards & Engagement | UL Standard](#)
- [41] [eCFR :: 47 CFR Part 15 Subpart C -- Intentional Radiators](#)
- [42] [EN 300 328-1 - V1.3.1 - Electromagnetic compatibility and Radio spectrum Matters \(ERM\); Wideband Transmission systems; Data transmission equipment operating in the 2,4 GHz ISM band and using spread spectrum modulation techniques; Part 1: Technical characteristics and test conditions](#)
- [43] [ISO 9241-210:2019 - Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems](#)
- [44] [ISO 14971:2019 - Medical devices — Application of risk management to medical devices](#)
- [45] [✓ RoHS Product Categories & Exemptions](#)

Appendix B: Copyright Permission

Everything that we have used so far has been open source. It has not been required for us to obtain copyright permission. If there is some information that requires us to obtain copyright permission, then we will update it in this appendix.

Appendix C: Data Sheet

There are a few datasheets that we have already used when identifying our components and using their information in correlation with our schematic development. Without the datasheets we will be developing our schematics wrong and that will affect our PCB designs.

- <https://docs.espressif.com/projects/esp-idf/en/release-v4.2/esp32/hw-reference/esp32/get-started-devkitc.html>
- https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
- <https://randomnerdtutorials.com/arduino-load-cell-hx711/>
- https://www.lcsc.com/datasheet/lcsc_datasheet_2410121720_SKYWORKS-SILICON-LABS-CP2102-GMR_C6568.pdf

Appendix D: Software Code

The official code for our project has not been written yet, but we did write a testing code that allowed us to sync all our major components together for our Pre-Demo video. The code has been written in Arduino C++, but it will change later in the future because we will migrate to VScode.

```

1  #include "HX711.h"
2  #include <SPI.h>
3  #include <Adafruit_GFX.h>
4  #include <Adafruit_SSD1306.h>
5  #include <Pushbutton.h>
6  #include "Adafruit_Thermal.h"
7
8  Adafruit_Thermal printer(&Serial1);
9
10 // HX711 circuit wiring
11 const int LOADCELL_DOUT_PIN = 21;
12 const int LOADCELL_SCK_PIN = 22;
13
14 HX711 scale;
15 float reading;
16 float lastReading;
17
18 #define GRAMS_TO_OUNCES 0.035274
19
20 // Calibration factor (adjust based on your setup)
21 #define CALIBRATION_FACTOR 220.79
22
23 // OLED SPI Display
24 #define SCREEN_WIDTH 128
25 #define SCREEN_HEIGHT 64
26
27 // SPI OLED pins (adjust if using different GPIOs)
28 #define OLED_MOSI 23 // D1 on display
29 #define OLED_CLK 18 // D0 on display
30 #define OLED_DC 2
31 #define OLED_CS 5
32 #define OLED_RESET 4
33
34 // printer pin stuff
35 #define printerRX 16
36 #define printerTX 17
37
38 // Create SPI display object
39 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
40                          &SPI, OLED_DC, OLED_RESET, OLED_CS);
41
42 // Button
43 #define BUTTON_PIN 32
44 Pushbutton button(BUTTON_PIN);
45
46 bool holdMode = false; // Starts in live mode
47 float heldReading = 0; // Stores the last frozen weight
48
49 void displayWeight(float grams){
50     float ounces = grams * GRAMS_TO_OUNCES;
51
52     display.clearDisplay();
53     display.setTextSize(1);
54     display.setTextColor(WHITE);
55     display.setCursor(0, 10);
56     display.println("Weight:");
57     display.display();
58     display.setCursor(0, 30);
59     display.setTextSize(2);
60     display.print(ounces, 2);
61     display.print(" oz");
62     display.display();
63 }
64

```

```

64
65 ✓ void printHeldWeight(float heldWeight)
66 {
67     float ounces = heldWeight * GRAMS_TO_OUNCES;
68     Serial1.begin(9600, SERIAL_8N1, printerRX, printerTX);
69     printer.begin();
70     printer.setTextSize('L');
71     printer.print("Weight: ");
72     printer.print(ounces);
73     printer.print(" Oz");
74     printer.println();
75     printer.println();
76     printer.println();
77
78     printer.sleep();
79     delay(3000L);
80     printer.wake();
81     printer.setDefault();
82 }
83
84 ✓ void setup() {
85     Serial.begin(115200); // match your Serial Monitor baud
86     // Initialize display
87     if (!display.begin(SSD1306_SWITCHCAPVCC)) {
88         Serial.println(F("SSD1306 allocation failed"));
89         while (true); // halt
90     }
91
92     delay(1000);
93     display.clearDisplay();
94     display.setTextColor(WHITE);
95
96     // Initialize scale
97     Serial.println("Initializing the scale...");
98     scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
99     scale.set_scale(CALIBRATION_FACTOR);
100    scale.tare(); // zero out
101 }
102 ✓ void loop() {
103     if (button.getSingleDebouncePress()) {
104         holdMode = !holdMode; // toggle between hold and live
105         if (holdMode) {
106             Serial.println("Hold mode ON");
107             heldReading = lastReading; // freeze current weight
108             displayWeight(heldReading);
109             // End Serial here?
110             // Do print stuff here
111             printHeldWeight(heldReading);
112         } else {
113             Serial.println("Hold mode OFF");
114         }
115     }
116
117     if (!holdMode && scale.wait_ready_timeout(200)) {
118         reading = (scale.get_units());
119         Serial.print("Weight: ");
120         Serial.println(reading);
121         if (reading != lastReading) {
122             displayWeight(reading);
123         }
124         lastReading = reading;
125     } else if (!scale.wait_ready_timeout(200)) {
126         Serial.println("HX711 not found.");
127     }
128 }

```

Appendix E: ChatGPT Prompts & Outcome

Prompt #1 regarding printers: “What is a good thermal printer for a project like this...”

That’s a great little embedded project — ESP32 + load cell + thermal printer is a super practical combo, especially for something like printing weight labels on the spot (like for breastmilk bags, food portions, or shipping). I can definitely help you narrow down a good printer for that setup!

Here are a few **tried-and-true thermal printers** commonly used in ESP32 and embedded microcontroller projects:

1. Adafruit Mini Thermal Receipt Printer (Product ID: 597)

- **Paper Width:** 58mm (standard receipt size)
- **Interface:** TTL Serial (works directly with ESP32 UART, 5V or 3.3V logic levels)
- **Voltage:** Runs off 5-9V (you can use a separate supply for the printer, ESP32 can’t supply enough current)
- **Print Speed:** ~50-80mm/sec
- **Driver/Library:** Arduino-friendly (Adafruit_Thermal library) or easy to control with basic serial commands.
- **Pros:** Super easy to integrate, rock-solid reliability, great support.
- **Cons:** A bit bulky for tight enclosures.

2. [Gainscha GP-5890 Series / ESC/POS Compatible Thermal Printers (58mm)

- **Paper Width:** 58mm
- **Interface:** Serial TTL, USB, or RS232 (pick the right model for your use).
- **Voltage:** Usually 5V–8.5V.
- **Print Speed:** Up to 90mm/sec.
- **Driver/Library:** Compatible with ESC/POS commands, so you can send plain text or use existing printer libraries (there are open-source ESP32 examples for ESC/POS).
- **Pros:** Widely available, affordable, and fairly simple serial interface.
- **Cons:** Often designed for point-of-sale systems, so form factor can vary.

3. MUNBYN Embedded Thermal Printer Module (58mm)

- **Paper Width:** 58mm
- **Interface:** Serial TTL, USB.
- **Voltage:** 5-8.5V.
- **Features:** Built specifically for embedded applications (compact size, mounting brackets).
- **Pros:** Clean integration into products and enclosures, good documentation, compatible with ESC/POS.
- **Cons:** Not as widely community-supported as Adafruit’s, but still reliable.

A few design notes:

- **Power Supply:** Thermal printers draw *a lot* of current when printing (usually spikes of 1.5–3A). Your ESP32 must be powered separately from the printer. A 5V 3A or 9V 2A wall adapter is common for the printer.
- **Interface Choice:** Serial is the easiest to implement on ESP32. Look for TTL-level (3.3V or 5V) UART printers. Avoid USB models unless you're using USB host chips.
- **Print Content:** For simple labels (e.g., "Date: 2025-04-21\nWeight: 122.4g\nUser: Alice"), you only need plain text, but many printers also support bitmap printing (logos, barcodes) via ESC/POS.

My Recommendation:

For most embedded ESP32 + label printing projects, I recommend starting with the **Adafruit Mini Thermal Printer (or a compatible clone)**. It's well-documented, easy to hook up, supports plain text or formatted output, and there are tons of examples.

If you're planning a production product (not a prototype) with a need for clean enclosure integration, I'd lean toward the **MUNBYN Embedded Module** — it's made for that kind of use case.

If you want, tell me:

- what kind of label info you plan to print,
- if you want battery operation or wall power,
- how much space you have for the printer,

... and I can help you choose one that's perfect!

Prompt #2 regarding mobile app technology: "Which technologies or frameworks would work best for an android mobile app like this..."

1. Native Android Development (Java or Kotlin)

- **Languages:** Kotlin (modern), Java (legacy support)
- **Tools:** Android Studio
- **Best for:** Full control, performance-intensive apps, deep system integration

Pros:

- Best performance
- Full access to Android APIs
- Backed by Google

Cons:

- Android only
- More code compared to cross-platform

2. Flutter (by Google)

- **Language:** Dart
- **Best for:** Beautiful UIs, fast development, Android + iOS from one codebase

Pros:

- Great performance (compiles to native)
- Hot reload = fast dev cycles
- Huge widget library for custom UIs

Cons:

- Dart is less common
- Larger app size compared to native

3. React Native (by Meta)

- **Language:** JavaScript / TypeScript
- **Best for:** Cross-platform apps, especially if you already know React

Pros:

- One codebase for Android & iOS
- Lots of libraries and community support
- Reusable components

Cons:

- Can hit performance issues with complex animations or large apps
- Native modules sometimes required

4. Kotlin Multiplatform Mobile (KMM)

- **Language:** Kotlin
- **Best for:** Sharing business logic across platforms while keeping native UIs

Pros:

- Native UIs with shared backend logic
- Kotlin all the way
- Good for teams with native dev experience

Cons:

- Not fully mature like Flutter/React Native
- Smaller community

5. Xamarin / .NET MAUI (by Microsoft)

- **Language:** C#
- **Best for:** Developers already in the .NET ecosystem

Pros:

- Cross-platform with shared code
- Access to native APIs via C#
- Supported by Microsoft

Cons:

- Slightly heavier apps
- Less trendy than Flutter or React Native

Prompt #3 regarding batteries:” What kind of battery would you recommend to power a thermal printer, an LCD screen, a load cell, and an MCU?”

- Lithium Polymer (LiPo) Battery
 - o Typically, 3.7V per cell
 - o Pros: High discharge rate, lightweight, rechargeable.
 - o Cons: Requires a proper battery management system (BMS) to prevent over-discharge.
- Lithium-Ion (Li-Ion) Battery (18650 Cells)
 - o 3.7V per cell
 - o Pros: More stable than LiPo, good capacity-to-weight ratio.
 - o Cons: Lower discharge rate than LiPo, needs a BMS.
- LiFePO4 (Lithium Iron Phosphate) Battery
 - o 3.2V per cell
 - o Pros: Safe, long cycle life, stable voltage.
 - o Cons: Lower energy density than LiPo/Li-Ion

Prompt #4 regarding charge controllers: “What kind of charge controllers could be used to prevent overcharging these batteries?”

- Recommended Charge Controllers:
 - TP4056 (for single-cell 3.7V LiPo)
 - Ideal for small-scale applications.
 - Includes overcharge, over-discharge, and short-circuit protection.
 - BMS (Battery Management System) for 2S/3S LiPo (e.g., 2S 7.4V or 3S 11.1V)
 - Ensures balanced charging & protection for multi-cell packs.
 - Example: 3S 12V 25A LiPo BMS Module.

Then it gave options for the Lithium-Ion Battery cells specifically 18650 cells,

- Recommended Charge Controllers:
 - TP4056 with Protection Circuit (for single 18650 cells)
 - BMS for multi-cell packs (e.g., 2S, 3S, 4S Li-Ion)
 - Example: BMS 3S 12.6V 10A Protection Board.

Finally, it mentioned possible choices for the Lithium Iron Phosphate battery cells,

- Recommended Charge Controllers:
 - LiFePO4-Specific BMS (e.g., Jikong 4S 12.8V BMS)
 - MPPT Solar Charge Controllers (for renewable energy applications)
 - Example: EPEVER MPPT 10A-20A for LiFePO4.

Prompt #5 regarding weight sensors: “The intention is to design a device that will assist parents with weighing and labeling breast milk inside bags. The device will be portable and will include a rechargeable battery, a weigh sensor, thermal printer, ESP32, and will incorporate an app that will receive information from the scale and other information. What types of weight sensors would you recommend for this project?”

Strain Gauge Load Cell (With HX711) - Recommended

- **How it Works:** Measures strain (deformation) in a metal bar when weight is applied.
- **Pros:**
 - High precision (grams-level accuracy)
 - Compact & low power consumption
 - Easily interfaces with ESP32 using the HX711 ADC
- **Cons:**
 - Requires calibration
 - Can be affected by temperature variations
- **Ideal for:** Your application due to its precision, affordability, and easy integration.
-

Capacitive Weight Sensor

- **How it Works:** Measures the change in capacitance due to applied force.
- **Pros:**
 - Highly sensitive & stable
 - Fast response time
- **Cons:**
 - More expensive than strain gauge load cells
 - Requires a capacitance measurement circuit or specialized IC

- **Ideal for:** Compact designs where high precision and speed are needed.

Piezoelectric Weight Sensor

- **How it Works:** Uses piezoelectric materials that generate voltage when compressed.
- **Pros:**
 - Very sensitive to small forces
 - Can be miniaturized for compact designs
- **Cons:**
 - Not as stable for continuous loads (best for force sensing, not static weight)
 - Requires signal conditioning (high-impedance amplifier)
- **Ideal for:** Applications where rapid force sensing is needed rather than static weight measurement.

Digital Pressure Sensors (MEMS-Based)

- **How it Works:** Uses a MEMS (Microelectromechanical System) pressure sensor to measure force via air pressure in a small chamber.
- **Pros:**
 - Highly compact (small chip form factor)
 - Easy integration with ESP32 via I2C or SPI
- **Cons:**
 - Requires special mechanical design to distribute pressure evenly
 - Limited weight capacity compared to load cells
- **Ideal for:** Ultra-small designs where traditional load cells are too bulky.

Resistive Force Sensors

- **How it Works:** Measures change in resistance when force is applied.
- **Pros:**
 - Ultra-thin, flexible
 - Cheap & easy to integrate
- **Cons:**
 - Lower precision than strain gauge load cells
 - Not suitable for continuous weight measurement (better for detecting presence)
- **Ideal for:** Devices where approximate weight measurement is sufficient (not ideal for your precise weighing needs).

Prompt #6 regarding voltage regulators: “The intention is to design a device that will assist parents with weighing and labeling breast milk inside bags. The device will be portable and will include a rechargeable battery, a weigh sensor, thermal printer, ESP32, and will incorporate an app that will receive information from the scale and other information. What types of voltage regulators would you recommend for this project?”

Recommended Linear Regulators

1. AMS1117-3.3V (LDO)
 - a. Input: 4.5V - 12V

- b. Output: 3.3V @ up to 1A
 - c. Use Case: ESP32, HX711
 - d. Pros: Cheap, easy to use
 - e. Cons: Not efficient for battery-powered applications
- 2. LP2985-3.3V (LDO)
 - a. Input: 3.7V - 16V
 - b. Output: 3.3V @ up to 150mA
 - c. Use Case: If the ESP32 is in deep sleep mode and you want ultra-low power draw
 - d. Pros: Low quiescent current (saves battery)
 - e. Cons: Limited current output

When to use LDOs?

- When powering ESP32, HX711, and small displays.
- When power efficiency isn't critical (e.g., when running from a USB charger).

B. Switching Regulators (Buck & Boost Converters) – For High Efficiency

Prompt #7 regarding Microcontrollers and Field Programmable Gate Arrays:

what are the differences between a microcontroller and a Field Programmable Gate Array?

1. Architecture

- Microcontrollers:
 - o These are self-contained computing systems on a single chip.
 - o They typically include a central processing unit (CPU), memory (both RAM and flash), and a range of peripherals (timers, ADCs, communication interfaces) all integrated into one package.
- FPGAs:
 - o FPGAs consist of an array of configurable logic blocks (CLBs), programmable interconnects, and often include dedicated memory blocks and even specialized hardware like DSP slices.
 - o They are essentially blank slates that can be programmed to perform a wide variety of digital functions by configuring their hardware structure.

2. Programming Model

- Microcontrollers:
 - o Programmers write sequential software in high-level languages such as C or C++ (or even assembly), which the microcontroller's fixed CPU executes.
 - o The functionality is defined by the software that runs on top of a predetermined hardware architecture.
- FPGAs:
 - o FPGAs are programmed using hardware description languages (HDLs) like VHDL or Verilog.

- o Instead of writing sequential code, you design and implement digital circuits. This approach lets you configure the hardware to perform specific tasks in parallel.

3. Performance and Parallelism

- Microcontrollers:
 - o They are well-suited for control tasks and applications where sequential processing is sufficient.
 - o The CPU processes instructions one after another, which can limit performance in highly parallel tasks.
- FPGAs:
 - o They excel at parallel processing because multiple hardware blocks can operate simultaneously.
 - o This makes them ideal for applications that require high-speed data processing, real-time signal processing, or custom data paths.

4. Flexibility and Customization

- Microcontrollers:
 - o They offer a fixed hardware platform that is optimized for general-purpose computing tasks.
 - o While you can change the software, the underlying architecture (and its performance limits) remains constant.
- FPGAs:
 - o Their reconfigurability allows you to implement custom hardware tailored to your application.
 - o You can modify the circuit architecture even after deployment (if the device supports reconfiguration), making them highly adaptable to evolving requirements.

5. Design Complexity and Development Time

- Microcontrollers:
 - o Generally easier to program and integrate into systems, especially for developers familiar with software programming.
 - o They typically have a shorter development cycle for standard control applications.
- FPGAs:
 - o The design process can be more complex and time-consuming because it involves hardware design, simulation, synthesis, and timing analysis.
 - o The learning curve is steeper, especially for those new to HDLs and digital circuit design.

6. Cost and Power Consumption

- Microcontrollers:
 - o Often more cost-effective and consume less power, which is advantageous for low-power and cost-sensitive applications.

- FPGAs:
 - They tend to be more expensive and power-hungry, though advances in FPGA technology continue to address these issues.
 - The benefits of parallelism and custom hardware often justify the additional cost and power usage for high-performance or specialized applications.

what are some great MCUs to use that are capable of wireless communication?

1. ESP32 Series

- Features: Dual-core processor, integrated Wi-Fi and Bluetooth (classic and BLE), plenty of GPIOs, and rich peripheral support.
- Pros: Versatile, cost-effective, and a strong community with plenty of libraries and examples.
- Ideal for: Projects requiring both Wi-Fi and Bluetooth connectivity, such as smart home devices or portable gadgets.

2. ESP8266

- Features: Affordable Wi-Fi-only MCU with enough processing power for many IoT applications.
- Pros: Great for simple Wi-Fi-based projects; extensive community support.
- Ideal for: Projects where Wi-Fi connectivity is the primary requirement and cost is a major factor.

3. Raspberry Pi Pico W

- Features: A recent addition to the Pico family, offering Wi-Fi connectivity along with the efficient RP2040 processor.
- Pros: Low cost, ease of use, and backed by the extensive Raspberry Pi community.
- Ideal for: Hobbyist projects and educational applications where quick prototyping is needed.

4. Nordic Semiconductor nRF52/nRF52840 Series

- Features: Focus on low-power Bluetooth Low Energy (BLE) with support for additional protocols like Thread or Zigbee (with proper firmware).
- Pros: Excellent power efficiency, strong for battery-operated devices, and robust wireless performance.
- Ideal for: Wearables, sensor nodes, and applications where power consumption is critical.

5. TI SimpleLink MCUs (e.g., CC2640, CC2650)

- Features: Designed for low-power wireless communication, particularly for BLE applications.
- Pros: Optimized for power efficiency and reliable wireless performance, with solid support from TI.
- Ideal for: IoT devices that require long battery life and robust BLE connectivity.

6. STM32WB Series

- Features: Dual-core STM32 microcontrollers with built-in BLE and 802.15.4 support.
- Pros: Combines the versatility of STM32 with advanced wireless features; strong in both performance and connectivity.
- Ideal for: Industrial IoT, smart sensors, and applications that may need both BLE and other wireless standards.

Considerations When Choosing an MCU:

- Wireless Protocol: Ensure the MCU supports the specific wireless technology (Wi-Fi, BLE, Zigbee, etc.) needed for your application.
- Power Consumption: For battery-operated devices, low-power consumption is crucial.
- Processing Requirements: Assess if the processing power and memory meet the needs of your application.
- Development Ecosystem: A strong community, available libraries, and development tools can significantly ease the project.
- Cost and Availability: Consider the overall budget and availability of the MCU in your region.

Prompt #8 regarding Displays:

What are the different types of displays that can be paired up with the ESP32 microcontroller?

1. OLED Displays

- Common Models: SSD1306, SH1106
- Interface: I2C, SPI
- Resolution: Usually 128x64 or 128x32 pixels
- Best For: Low-power, small graphical interfaces

2. LCD Displays (Character & Graphic LCDs)

- Character LCDs: 16x2, 20x4 (HD44780 controller, uses I2C or parallel interface)
- Graphical LCDs: ST7920 (128x64, SPI)
- Best For: Simple text displays, menus

3. TFT LCD Displays

- Common Controllers: ILI9341, ST7735, ST7789, HX8357
- Interface: SPI (some support parallel)
- Resolution: Varies (128x128, 240x320, 320x480, etc.)
- Best For: Colorful and detailed graphical interfaces

4. E-Paper (E-Ink) Displays

- Common Models: Waveshare EPD series (1.54", 2.9", 4.2", etc.)
- Interface: SPI

- Best For: Low-power applications, static images

5. RGB LED Matrix Displays

- Common Controllers: HUB75, WS2812 (Neopixels)
- Interface: GPIO, SPI
- Best For: Scrolling text, animations, LED dashboards

6. Touchscreen Displays

- Common Controllers: FT6206 (capacitive), XPT2046 (resistive)
- Best For: Interactive UI applications

Prompt #9 regarding User interface:

What are the advantages and disadvantages of using a touchscreen or physical buttons?

Advantages:

- Versatile & Dynamic: Can display different interfaces and adapt to multiple functions.
- Space-Saving & Minimalistic: Reduces the number of physical components, allowing for sleek designs.
- Easy Software Updates: Interface elements can be changed without modifying hardware.
- Multi-Touch Support: Enables gestures like pinch-to-zoom, swipe, and multi-finger input.
- More Intuitive for Certain Applications: Especially useful for smartphones, tablets, and kiosks.

Disadvantages:

- Lack of Tactile Feedback: Users must rely on visual confirmation rather than feeling buttons.
- Higher Power Consumption: Can drain batteries faster than physical buttons.
- More Expensive to Repair: A broken touchscreen often requires full replacement.
- Difficult to Use with Gloves or Wet Hands: Can be less practical in certain environments.
- Accidental Touches: Prone to unintended inputs.

Physical Buttons

Advantages:

- Tactile Feedback: Users can feel and operate them without looking.
- More Reliable in Harsh Conditions: Works better in cold, wet, or industrial settings.
- Lower Power Consumption: Doesn't require an active display.
- Easier to Operate Without Looking: Useful for tasks requiring muscle memory (e.g., volume and power buttons).

- More Durable for Specific Applications: Less prone to breaking compared to touchscreens in some use cases.

Disadvantages:

- Limited Functionality: Can't change their function dynamically like a touchscreen.
- More Mechanical Wear & Tear: Buttons can degrade or break over time.
- Can Take Up More Space: More buttons may clutter a device's design.
- More Complex Hardware Design: Each button requires dedicated circuits, increasing complexity.

Prompt #10 regarding PCB Software Programs:

What are some different PCB software design applications that are available to use? Are they free or do they cost money?

Free PCB Design Software

1. KiCad – Open-source, widely used for professional and hobbyist PCB design. Free.
2. EasyEDA – Cloud-based and easy to use, with a free version and premium plans.
3. DesignSpark PCB – Free software from RS Components with good functionality.
4. Fritzing – Great for beginners, especially for Arduino and breadboard-based designs. Free but accepts donations.
5. Osmond PCB – Free for small designs, Mac-only.

Paid PCB Design Software

6. Altium Designer – Industry-standard, powerful, but expensive (~\$8,000 per license).
7. OrCAD (Cadence) – Used in professional environments, requires a license.
8. EAGLE (Fusion 360 Electronics) – Formerly standalone, now part of Autodesk Fusion 360, available via subscription (~\$500/year).
9. PADS (Siemens) – Professional-grade software with multiple tiers, costs vary.
10. DipTrace – User-friendly with a free version for small projects, paid versions start at ~\$75.