PUMPED

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

Sponsors: Victoria Rodriguez, Matthew Aberman

We Are Group 15 and this is our PUMPED Team!

RIP

Agustin Rodriguez
Computer Engineer

Jimmie Rawls
Electrical Engineer

Eric Hernandez
Computer Engineer

Kevin Fernandez
Electrical Engineer

Project Motivation

- There is no consumer product on the market that does this.
  - Opportunity to serve important groups.
  - Opportunity to improve quality of life.
- Properly labeling breast milk is critical for mothers and prone to errors
  - Tired mothers labeling bags at odd hours may make mistakes.
  - Diet/time of day affects milk composition so its important these are both properly recorded.
- Associated mobile app provides extra tracking and information for breast milk bags
  - Mobile app provides ability to check milk inventory remotely.
  - Fixes problem of limited label space.
  - Gives statistical view of milk production to review later.

Goals

Basic Goals

- The device accurately weighs the breastmilk bag placed on the scale portion of the device.
- Device prints a label for containing volume, date, and allergen/dietary details.
- An integrated app that displays current breastmilk stock for the 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.
- OLED screen displaying various bag information

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/milkbanks.
- Automatic shutdown of device depending on certain conditions

Microcontroller Comparison and Selection

When we were deciding between what MCU to choose from all the existing ones, we had a couple of key features that we needed to have that would allow us to successfully and easily implement all of our features. The key features that were a requirement for our MCU to have were that it needed to have Wifi and Bluetooth capabilities, good power efficiency, be cost effective, have good amounts of GPIO pins, and be able to run SPI, I2C, and UART communication interfaces. We needed all those communication interfaces to be able to connect multiple peripherals to the MCU.

MCU options that we had in mind:

- MSP430
  - Familiarity, low power, no wireless connection
- Raspberry Pi Zero 2 W
  - Wifi, bluetooth, costly, voltage concerns (4.75V - 5.25V)
- Arduino Uno R4 Wifi
  - Wifi, costly, voltage concerns (6V - 24V)
- Arduino MKR Wifi 1010
  - Wifi, bluetooth, costly, good voltage (3.3V)
- ESP32S

MCU of Choice:

- ESP32S-WROOM
  - Wifi
  - Bluetooth 4.2
  - 3.3V
  - Cheap
  - Powerful (up to 240 MHz)
  - 4 SPI, 2 I2C, 3 UART, CAN
  - Able to work with Arduino IDE
  - Excellent flexibility and scalability for interfacing with many different sensors and peripherals

TTL Thermal Printer Comparison and Selection

- Finding a printer that operated at a lower voltage range was important.
- Lower baud rate mitigates potential data integrity issues due to electrical noise
- Utilizes UART Serial
- Compact enough to fit within our 3D printed case.

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

Battery Comparison and Selection

During development, we looked at different ways to power our device. Our goal was to find an option that would be safe, easy to use, and able to power everything in the system. Including some of our peripherals that need a lot of energy, such as the thermal printer that draws a peak of 2 Amps. We also wanted something that would keep the device portable and convenient for everyday use. Below are the main options we considered:

Battery options that we had in mind:

- AA Rechargeable Batteries
  - Easy to find and reuse
  - Limited current output
  - Would require a BMS to charge.

- 18650 Lithium-Ion Cells
  - High energy density
  - High output current
  - Requires custom BMS and charging circuit

- USB Power Bank
  - Rechargeable and regulated
  - Built-in protection circuitry (BMS)
  - Bulky compared to internal battery packs

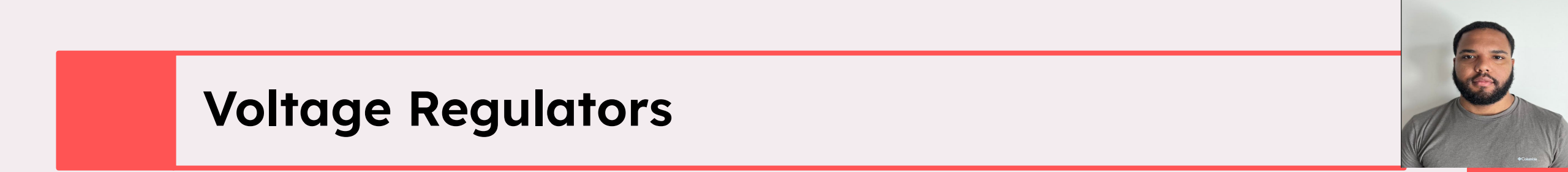
Battery of Choice:

- ROMOSS Sense 8+ (30,000mAh)

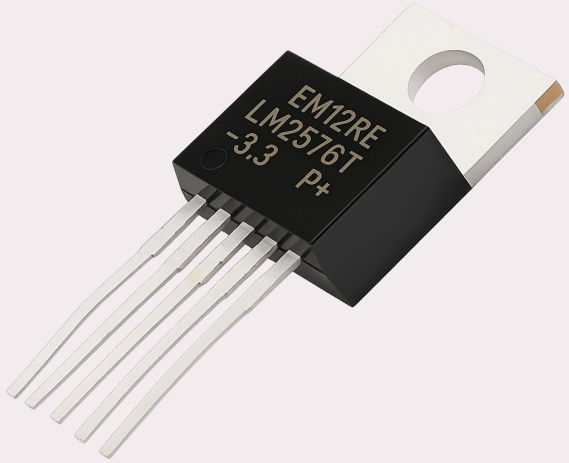
  - High capacity – 111Wh
  - Two USB-A outputs – 5V/2A and 5V/3A
  - Built-in protection (overcurrent, overcharge, short-circuit)

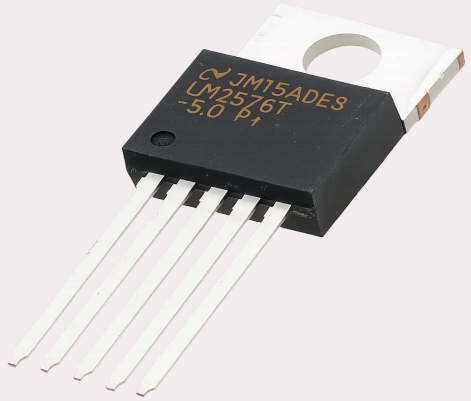
Our system uses about 9–10 Watts during operation. To meet our 10-hour usage goal, we needed a battery with around 100Wh capacity. The ROMOSS Sense 8+ provides 111Wh (30,000mAh at 3.7V), giving us enough power to meet our runtime needs.

Voltage Regulators

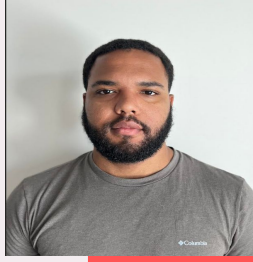


Our design will require two different voltages to operate all the components. The thermal printer will run at 5 volts while the MCU, Load cell, OLED display will operate at 3.3 volts. To bring the voltage from the battery down we will be using two LM2576 step down voltage regulators that will supply voltage to the peripherals at their respective operating ranges.





Weighing Technology Comparison and Selection



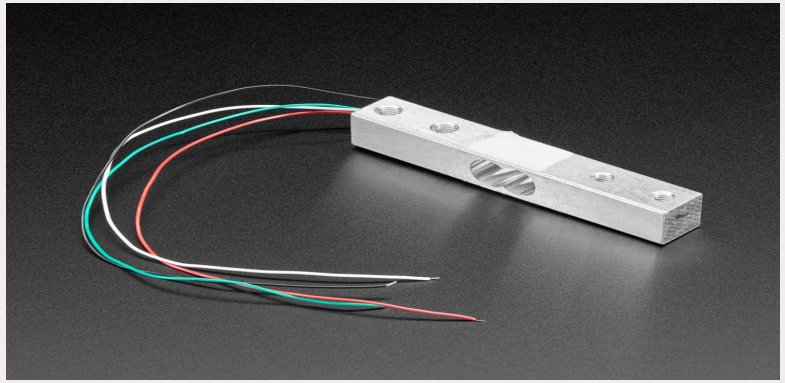
When selecting a scale to implement in our device we needed it to have a fast response time when calculating the weight while also providing sufficient accuracy so that the user can have confidence in knowing the exact amount of milk that is in their inventory.

Scale choices

- Strain gauge load cell
  - Accurate
  - Low cost
  - Simple to implement: Libraries in Arduino IDE
  - Works with the HX711 ADC
  - Static measurement
- Capacitive load cell
  - Accurate
  - Costly
  - Complex implementation
  - Static measurement
- Piezoelectric sensor
  - Accurate
  - Costly
  - Complex implementation
  - Dynamic measurement

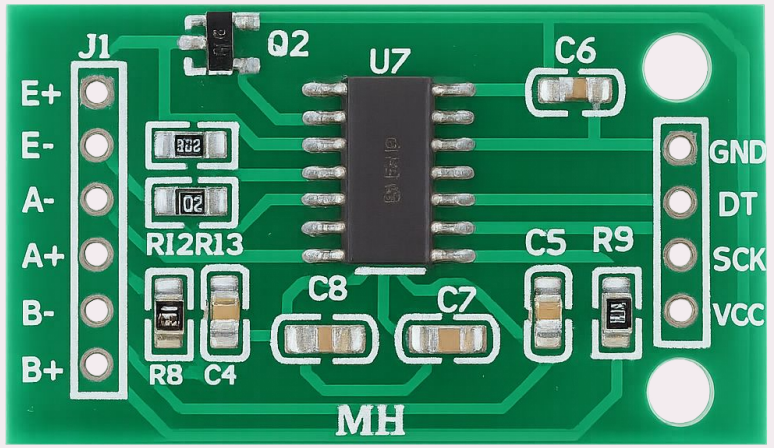
Selection ,

10 Kg (22 lbs) Strain gauge load cell



HX711

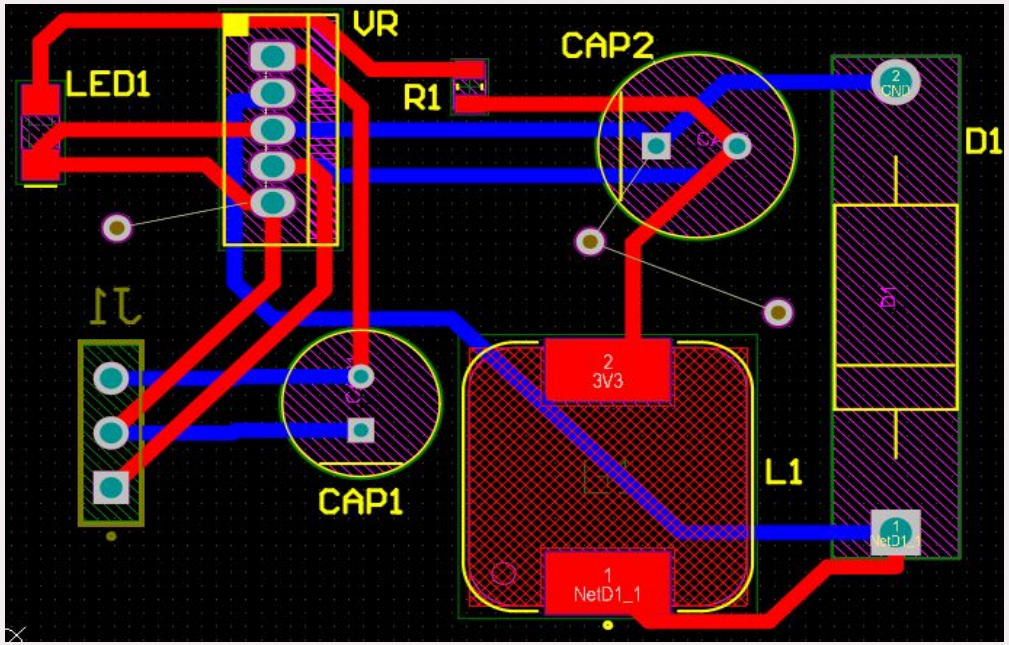
- 24- bit analog-to-digital converter (ADC)
- Designed specifically for load cells
- No extra programming required

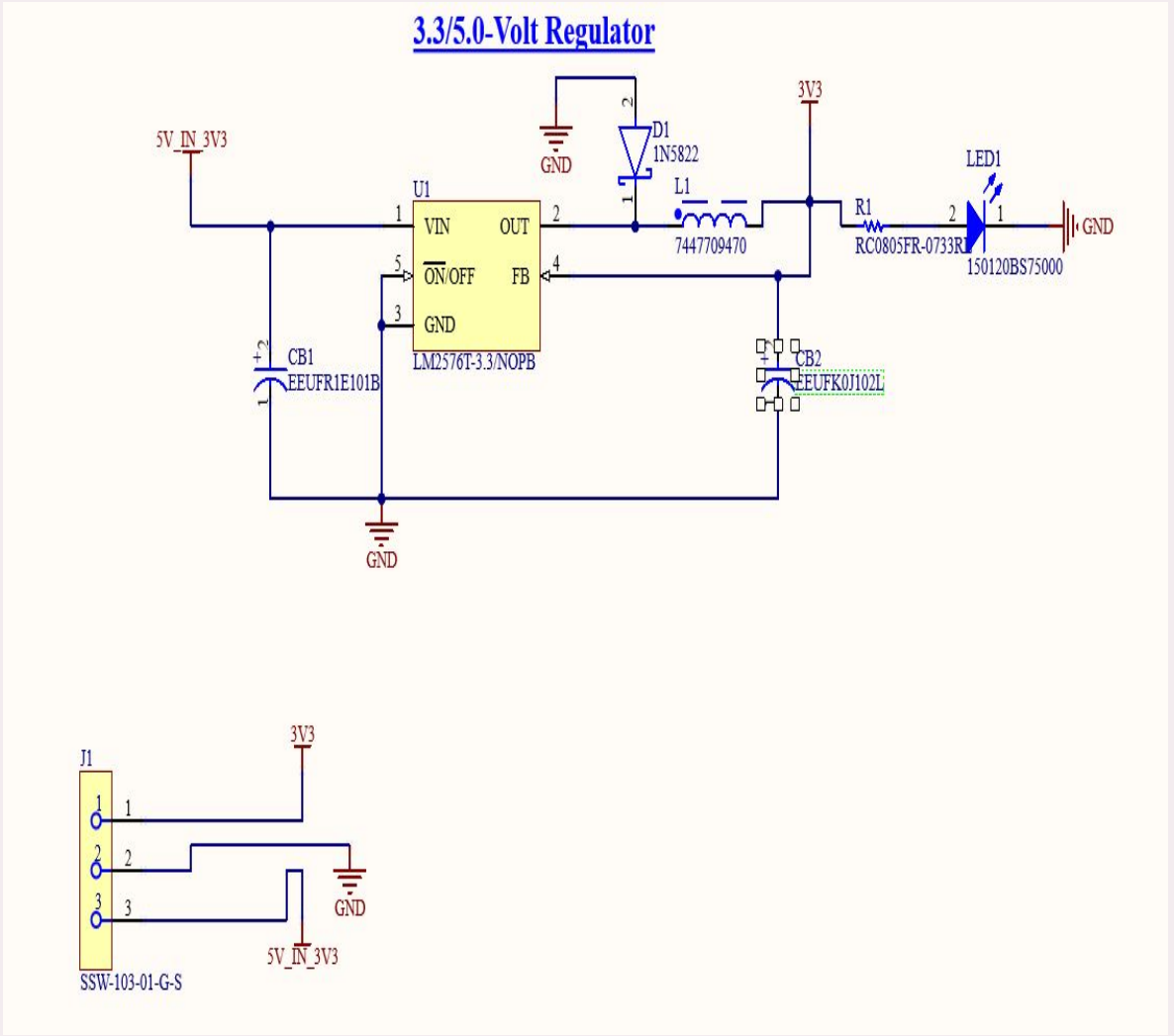


Engineering Specifications

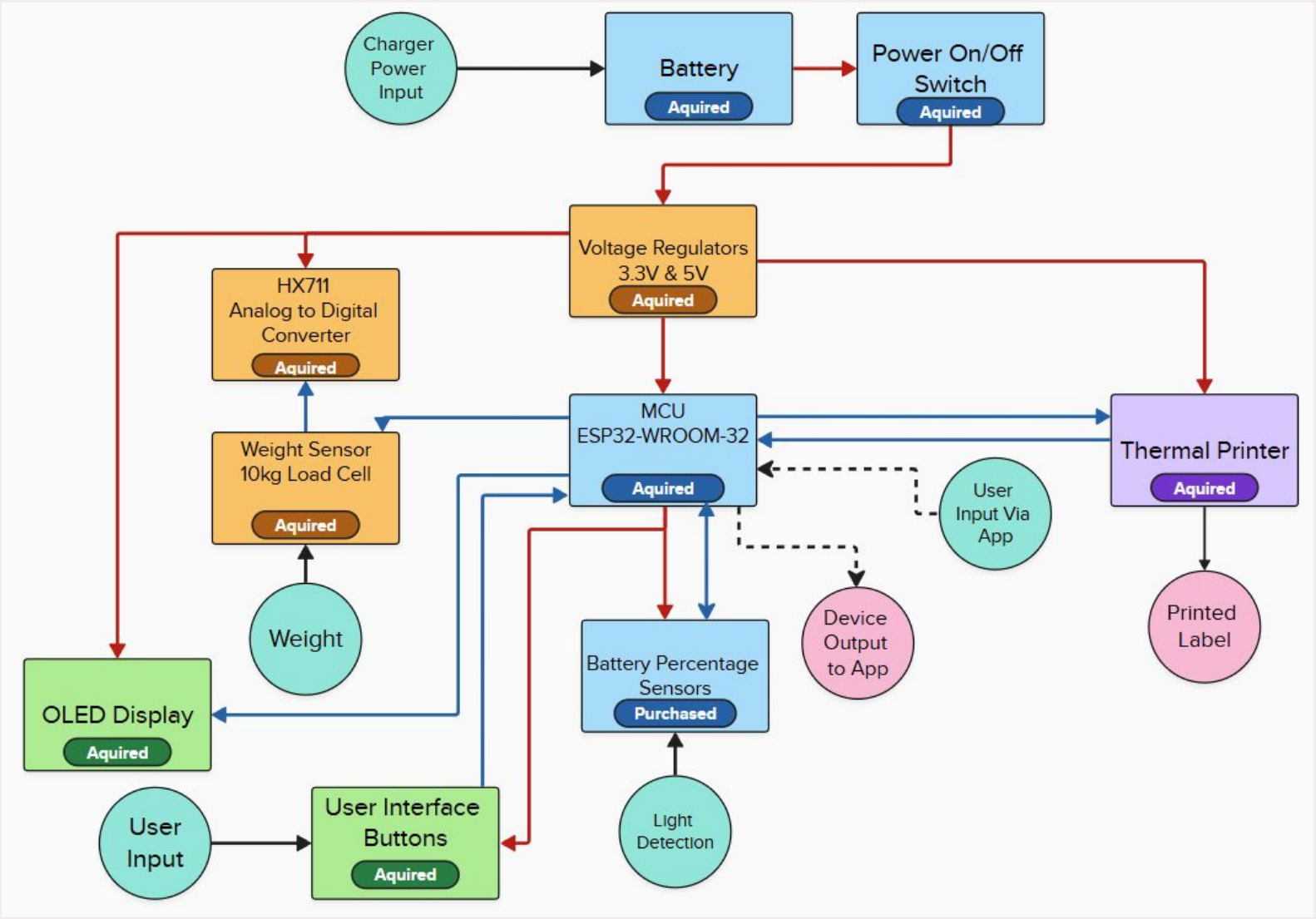
Specification	Engineering Requirement
• Weight sensor should be accurate	Accurate within +/- 3 grams
• Time to print after print button is pressed should be immediate	5 seconds or less
• Time for recorded breastmilk bag data to be sent to the mobile app should be short	5 seconds or less
• Time for breastmilk 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

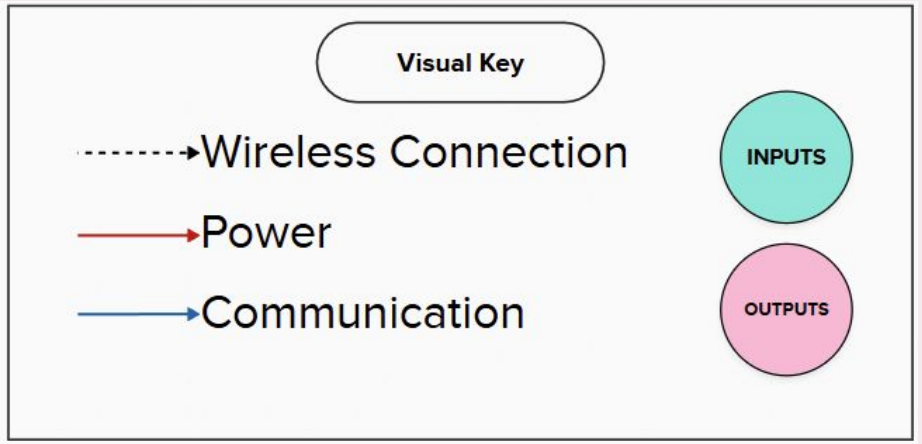
Voltage Regulator PCB Schematic

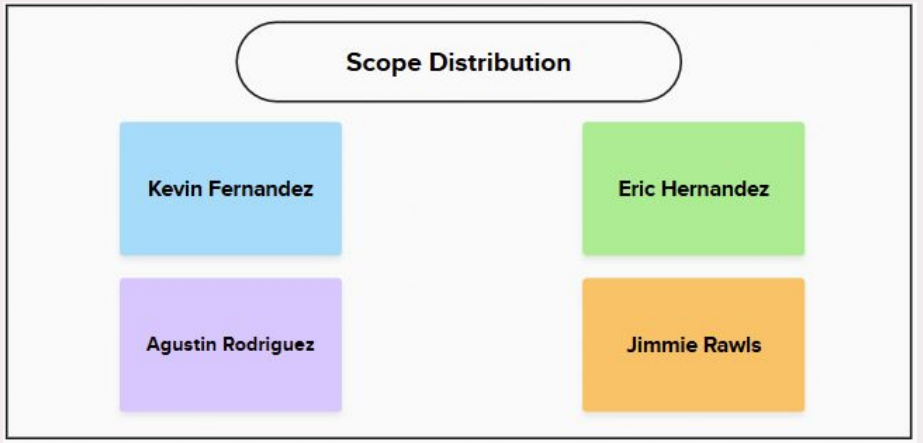




Hardware Block Diagram

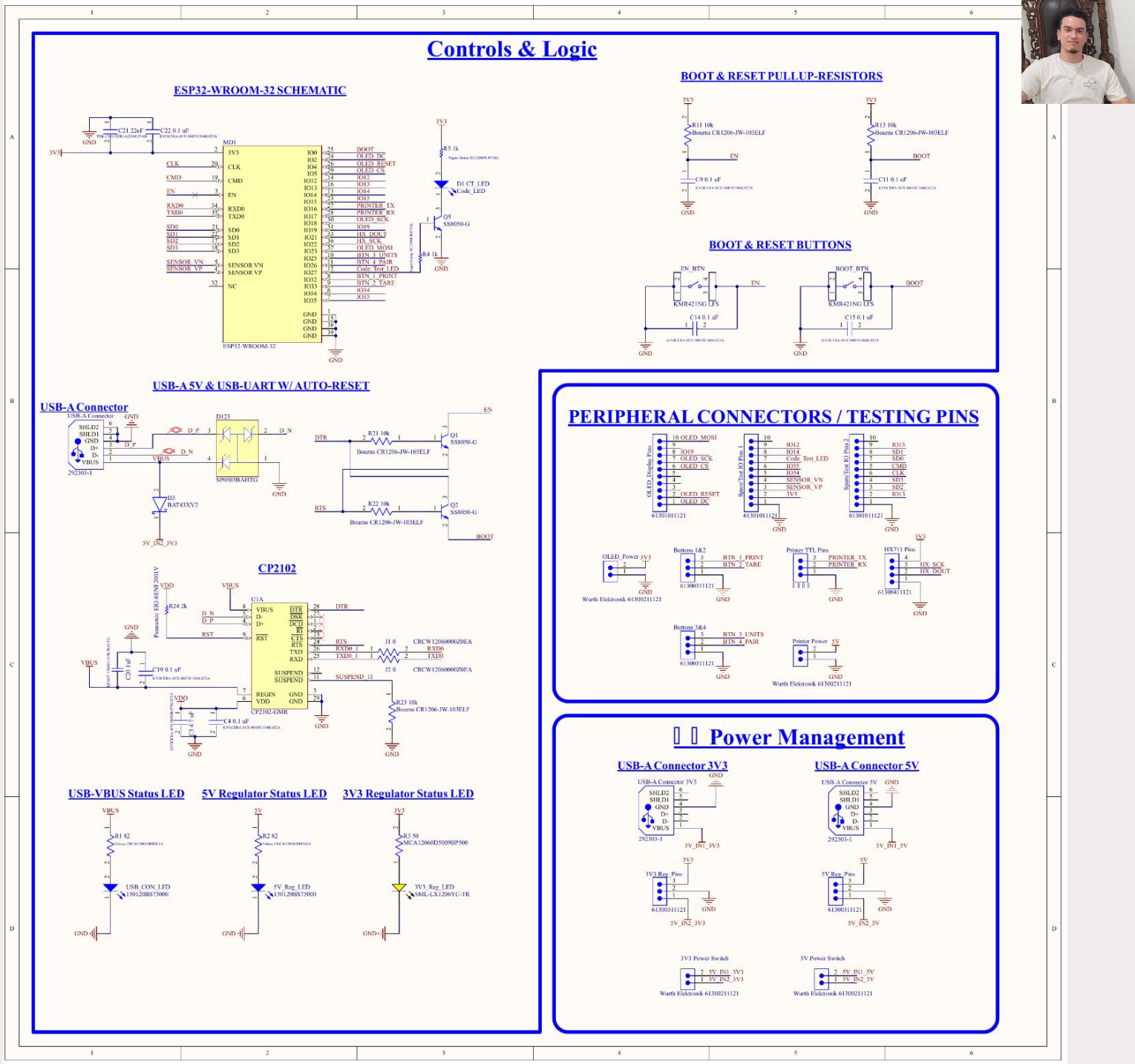






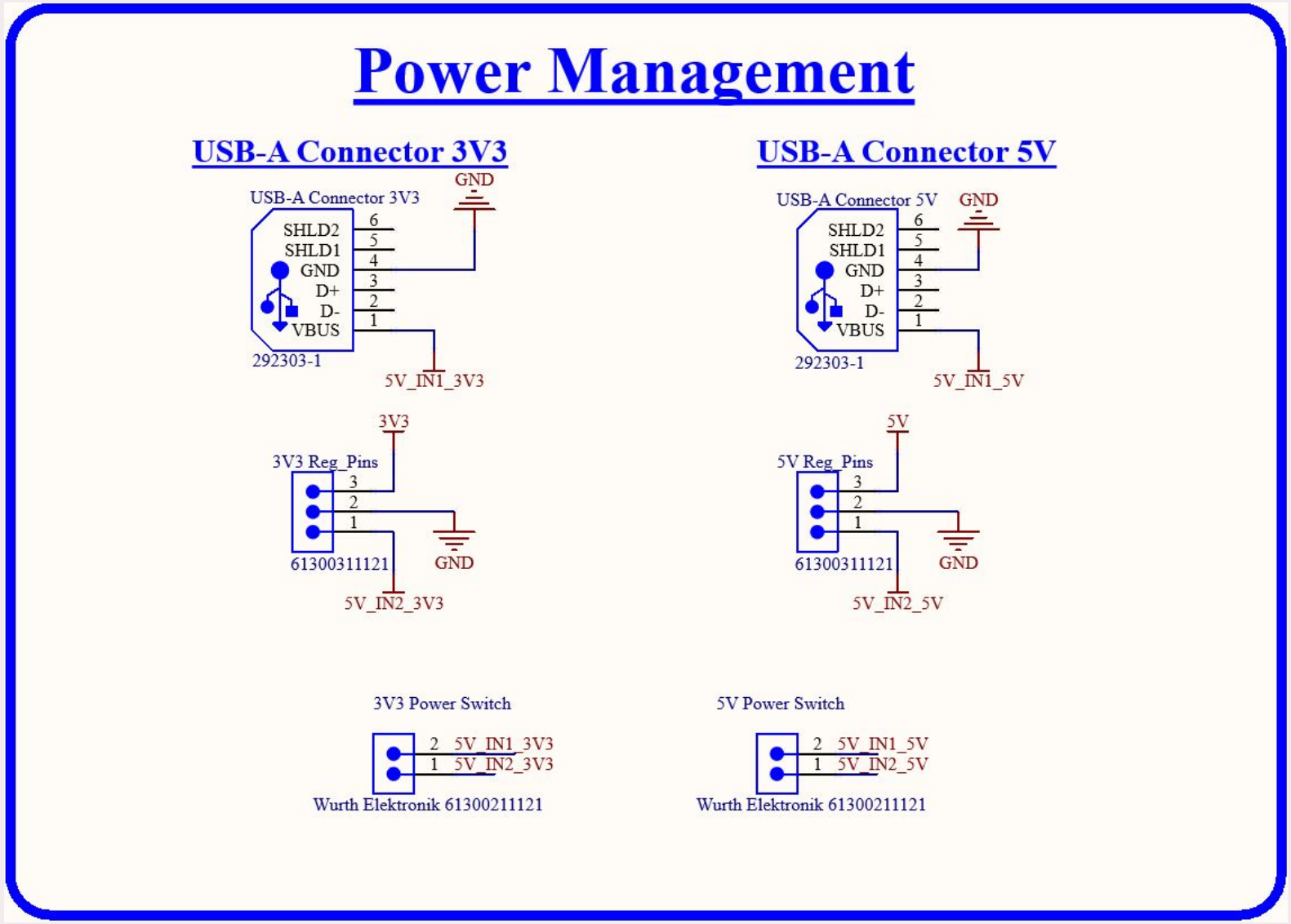
PUMPED MCU Schematic Overview

- **Power Management**
  - Headers for Off/On Switch
  - Modular 3.3V and 5V voltage regulators
- **Controls & Logic**
  - ESP32-WROOM-32 microcontroller
  - CP2102 USB-to-UART interface
  - Power Status LEDs
  - IO controlled LED
- **Peripherals:**
  - Load cell with HX711 amplifier
  - OLED display (SPI)
  - Embedded thermal printer
  - Four user interface buttons
  - Battery Percentage Indicators
- **Testing & Expansion**
  - Spare/test IO headers



Schematic: Power Management

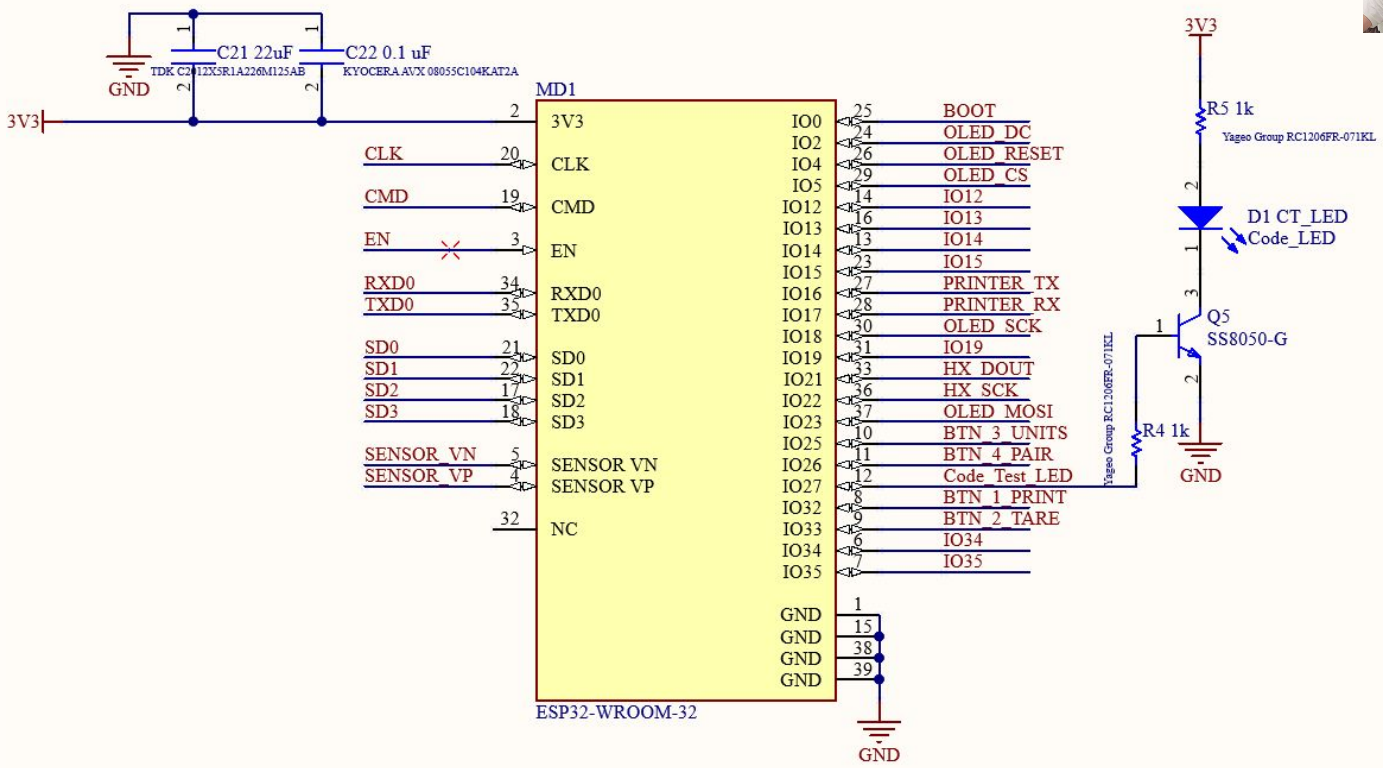
- **Power Input**
  - System powered by two USB-A inputs using USB-A to USB-A cables from the ROMOSS Battery Bank.
  - Avoids possible noise induced by high current draw from the Printer.
- **ON/OFF Switch**
  - Power rails are cut off at individual headers which will be connected to a DPDT Switch.
- **Modular Voltage Regulators**
  - Includes 3 pin header connectors that the 3.3V and 5V regulators will be connected to.

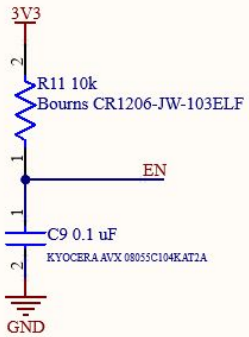


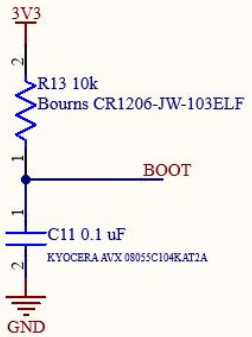
Schematic: Controls & Logic 1/3

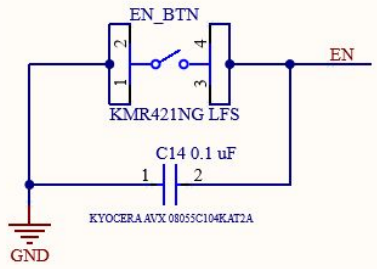
- **MCU (ESP32-WROOM-32)**
  - Central processor for all system functions/Peripherals
  - Powered by 3.3V with 22µF and 0.1µF capacitors for voltage stability and noise filtering
  - Interfaces with CP2102 for programming
  - Includes a Code Status LED on IO27, toggled by an NPN transistor.

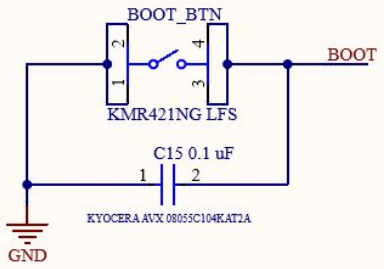
- **Boot & Reset Circuitry**
  - Pull-up resistors on BOOT and EN pins ensure proper startup
  - When the BOOT button is pressed it enters the MCU into FLASH mode
  - When the EN button is pressed the MCU is manually reset.

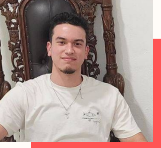










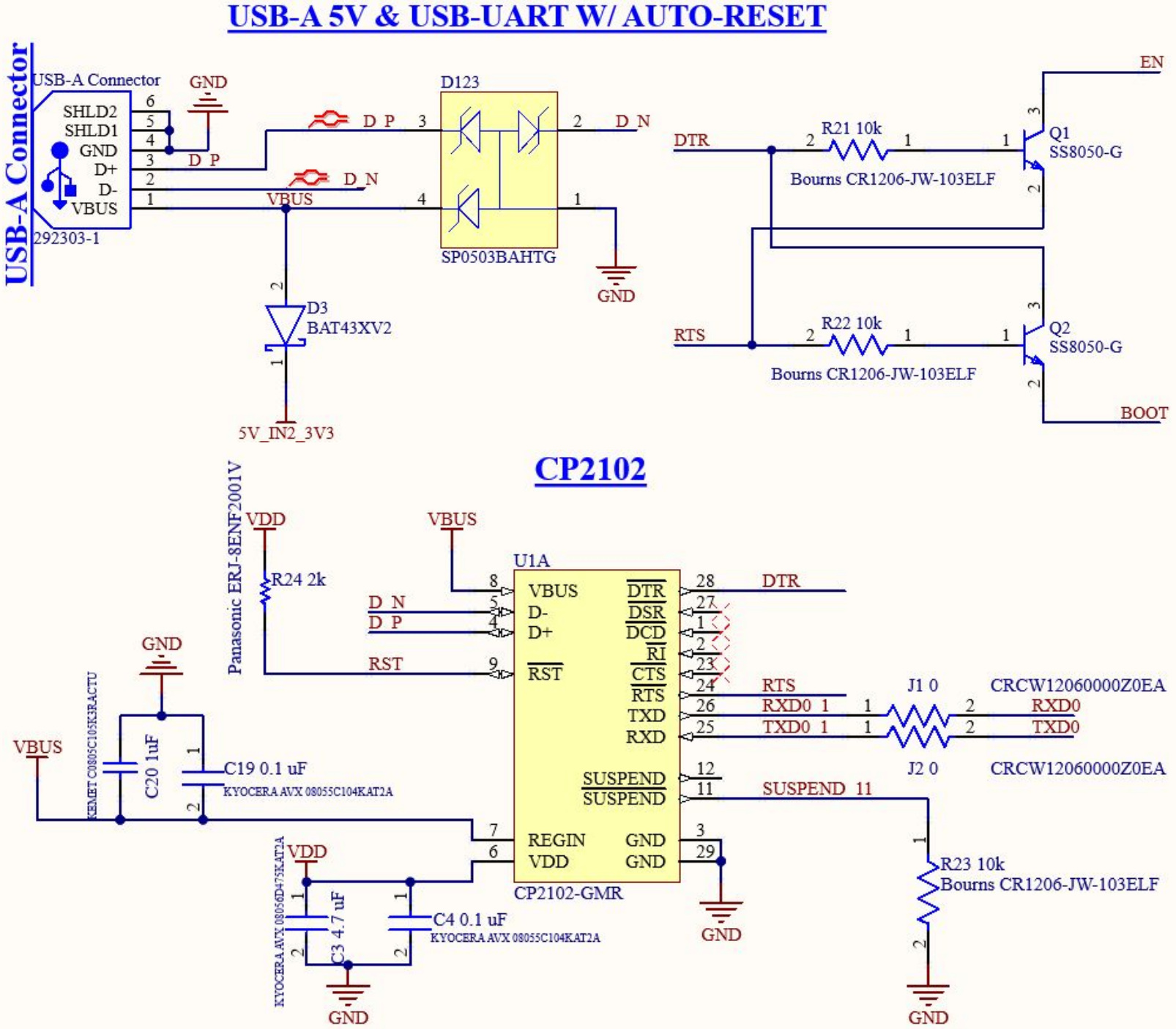


Schematic: Controls & Logic 2/3

- **USB-A Input**
  - Used to power the CP2102.
  - D- and D+ lines provide the USB data connection.
  - Data lines connected to a TVS diode (SP0503BAHTG) for ESD protection.

- **CP2102 USB-to-UART**
  - Handles serial communication and programming for the ESP32.
  - REGIN input and VDD output are stabilized/smoothed by capacitors.
  - Reset pin is pulled up with a 2k Ohm resistor.
  - Suspend pin is pulled down with a 10K Ohm resistor.
  - 0 Ohm resistors on the TX and RX lines are used for debugging or possible isolation.

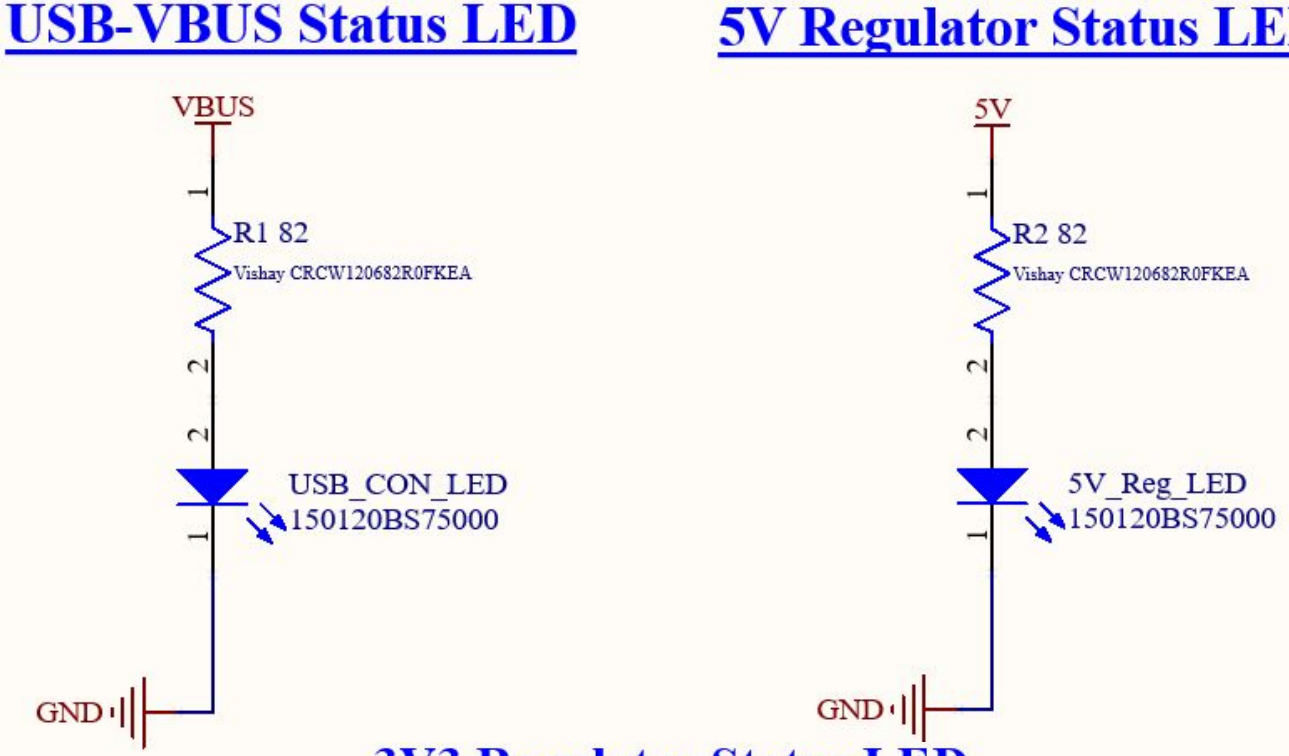
- **Auto Reset Circuitry**
  - Allows the ESP32 to automatically enter reset and boot modes during code uploads.
  - Uses two NPN transistors controlled by DTR and RTS from the CP2102.

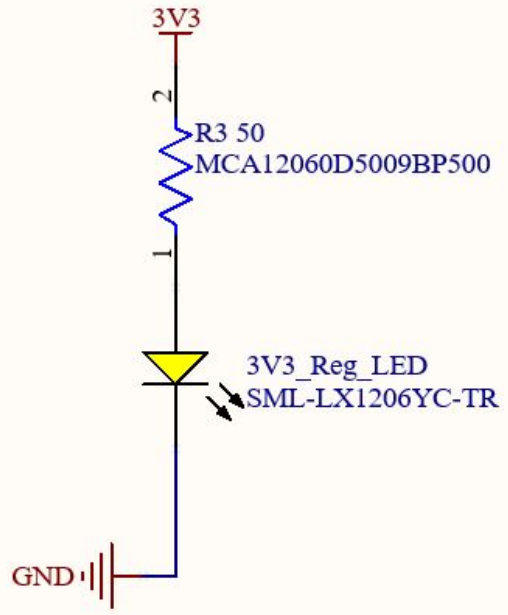


Schematic: Controls & Logic 3/3

Power Status LEDs

- **USB-VBUS LED**
  - Uses an 82 Ohm resistor to bias a blue LED with a forward voltage of 3.2V and a forward current of 30mA
  - Used to indicate that our VBUS when programming the MCU is active.
- **5V Regulator LED**
  - Same circuitry as the USB-VBUS LED
  - Used to indicate that our power output from the 5V regulator is working.
- **3.3V Regulator LED**
  - Uses a 50 Ohm resistor to bias a yellow LED with a forward voltage of 2.0V and a forward current of 30mA.
  - Used to indicate that our power output from the 3.3V regulator is working.





Schematic: Peripherals & Expansion

10-Pin Female Headers (x3)

- OLED display Header
  - SPI Interface for connecting the OLED Screen
- Spare IO Pins Header 1
  - Contains additional IOs for future use (will now be used for the battery level indicator)
  - Has an extra connection to 3.3V and a ground.
- Spare IO Pins Header 2
  - Also Contains additional IOs for future use.
  - Contains an additional connection to 5V and a ground.

4-Pin Female Header (x1)

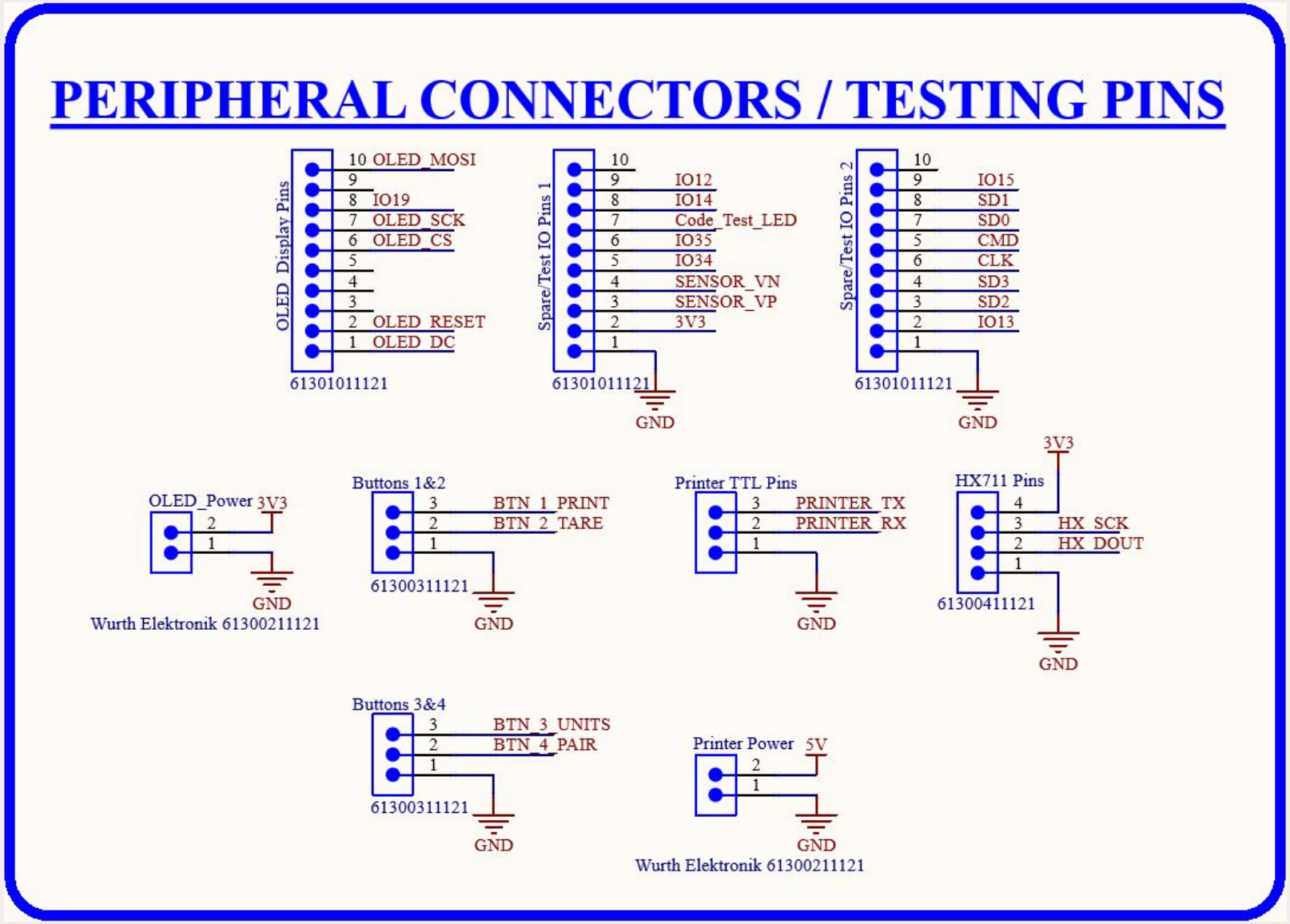
- HX711 Header
  - Direct plug in for the HX711 that includes 3.3V, GND, and its communication connections.

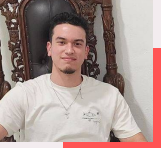
3-Pin Female Headers (x3)

- Buttons 1-2 Header and Button 3-4 Header
  - Both contain 2 connections to IO pins, and a ground (pulls IOs low when pressed)
- Printer TTL Header
  - Includes the RX and TX lines to communicate to/from the ESP32 and a GND pin.

2-Pin Female Headers (x2)

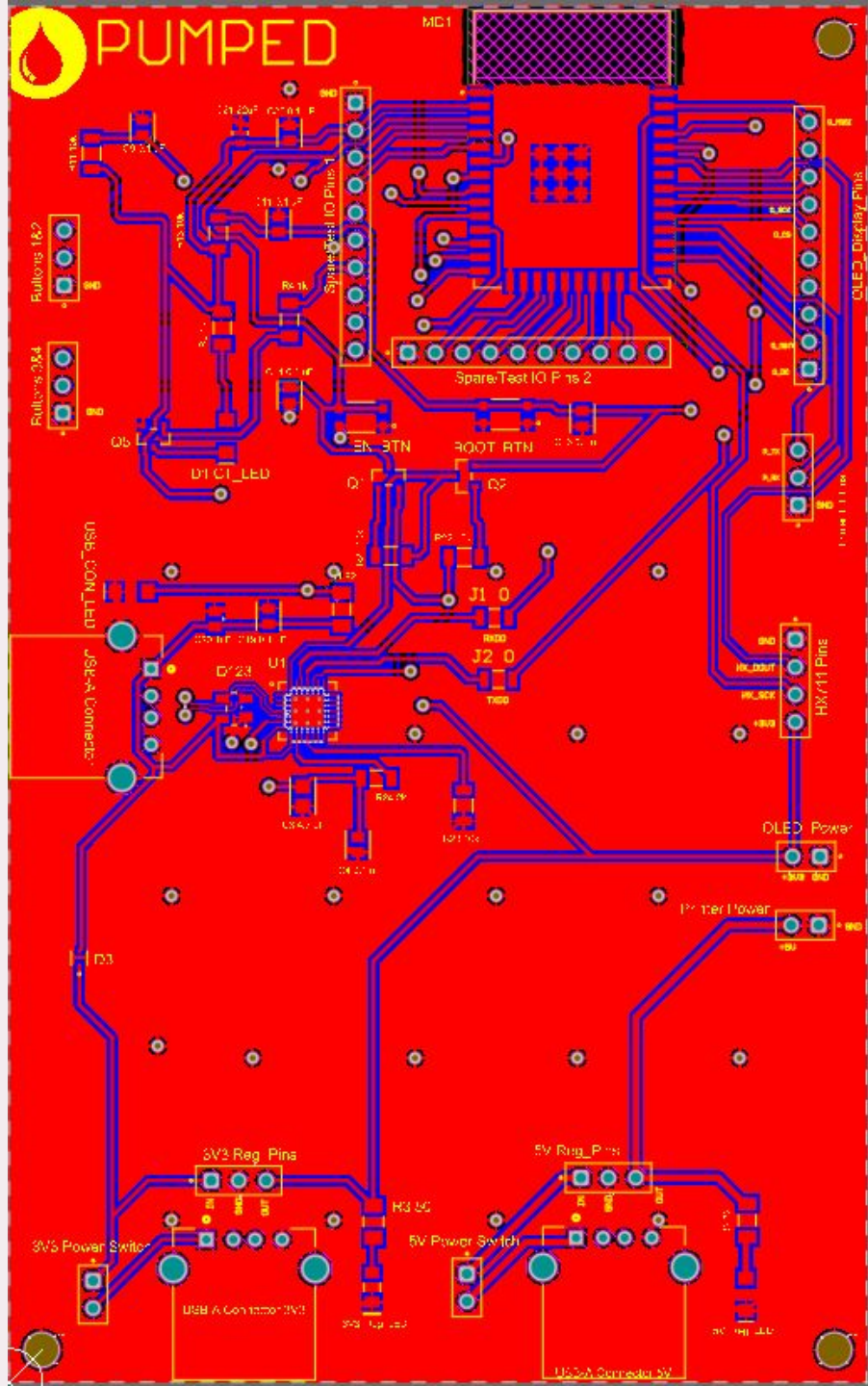
- OLED Power Header
  - Supplies 3.3V and a ground pin
- Printer Power Header
  - Supplies 5V and a ground pin





PUMPED MCU PCB Overview

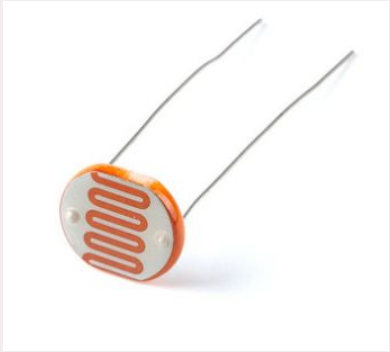
- **USB-A Connector Placement**
  - Both USB-A power inputs are located at the bottom edge of the board and are near the status LEDs along the voltage regulator headers.
  - The USB-A for programming the ESP32 is placed on the left side of the board and the CP2102 circuitry is placed closely to it to reduce signal noise and data errors.
- **ESP32 Antenna Cutouts**
  - The ground plane underneath the ESP32's antenna has been removed in both the top and bottom layers.
  - This helps mitigate interference and improves wireless signal performance.
- **Trace Sizing**
  - 32 mil trace was used for the printers trace lines so that they could withstand the 2 amps of current the printer peaks at.
- **Mounting holes**
  - Added mounting holes to some corners to secure the PCB to the inside of the devices enclosure.



Battery Percentage Indicator (1/2)

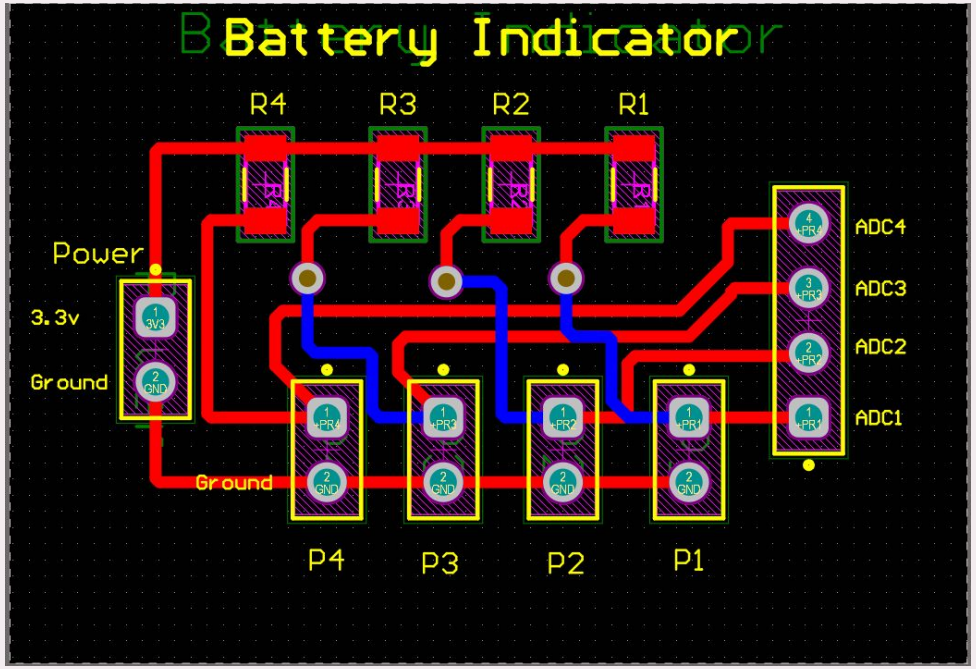
We decided to go with the ROMOSS Sense 8+ (30,000mAh) over regular batteries. This was a great opportunity for us to create a well rounded portable device that had the capability of being rechargeable without any concerns. We ran into an issue on how the consumer was going to determine what the battery 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.

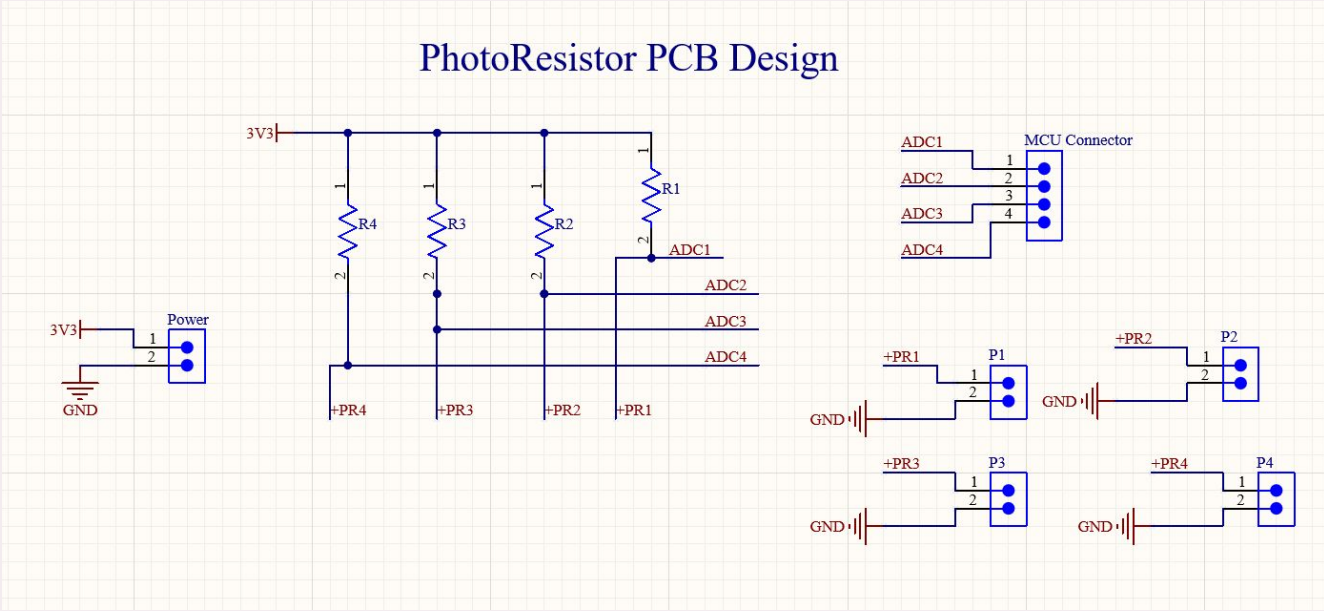
Using Photoresistors to determine the battery percentage based on the LEDs on the battery bank. The way photoresistors work is that the material has high resistance when in the dark, and lower resistance when in light. We are going to sync this up with the ESP32, and we are going to be able to determine the battery amount based on the amount of LEDs (out of 4) that are on and off.



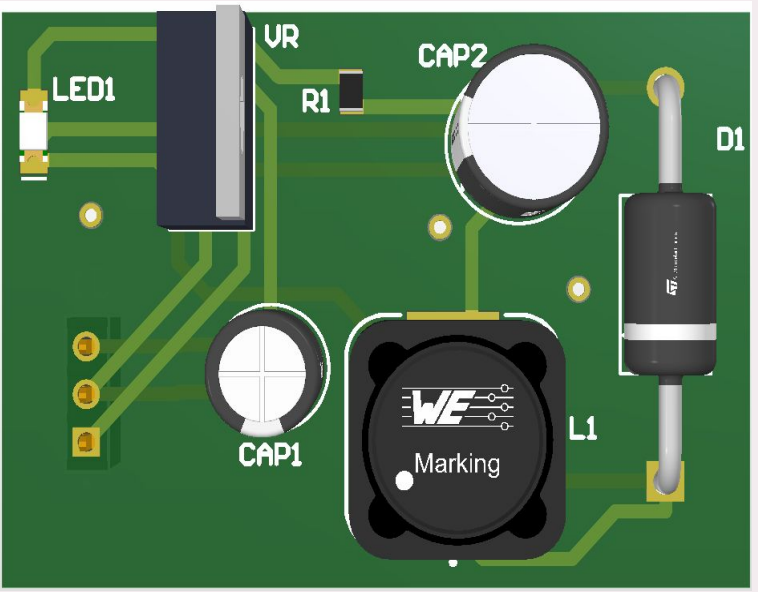
Battery Percentage Indicator (2/2)

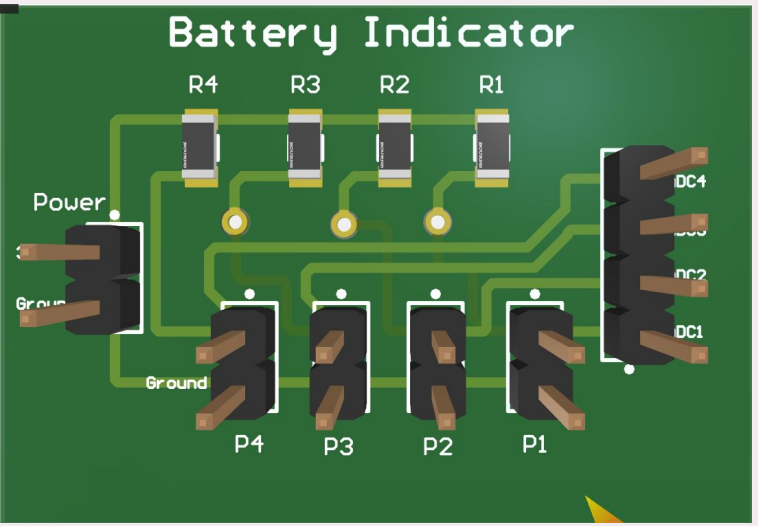
We came up with an idea to use 4 photoresistors (1 for each LED). they would be directly attached on top of the battery bank via a compartment, it will then be connected to the MCU via an ADC connection, and then the MCU will transmit it onto the display.

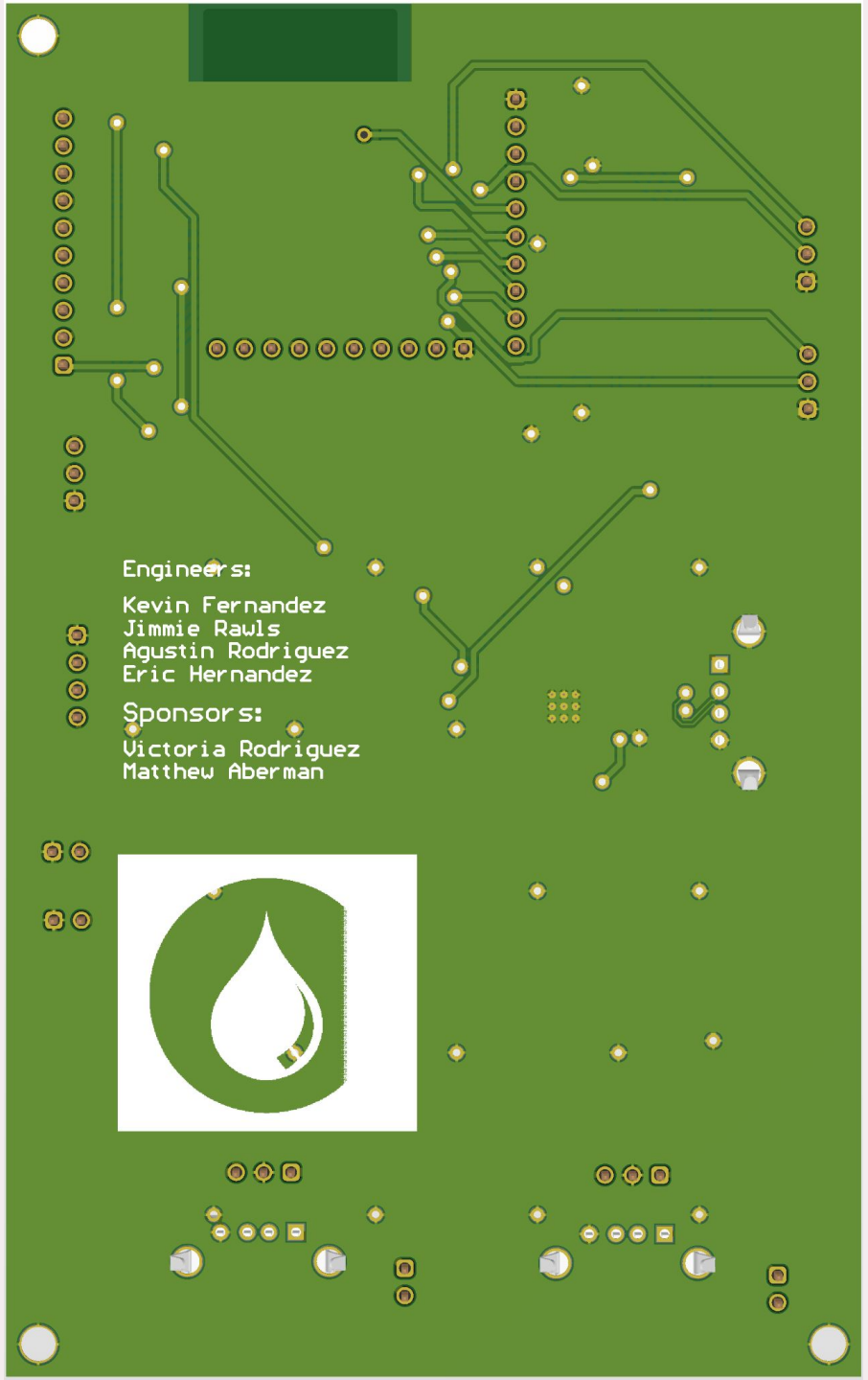


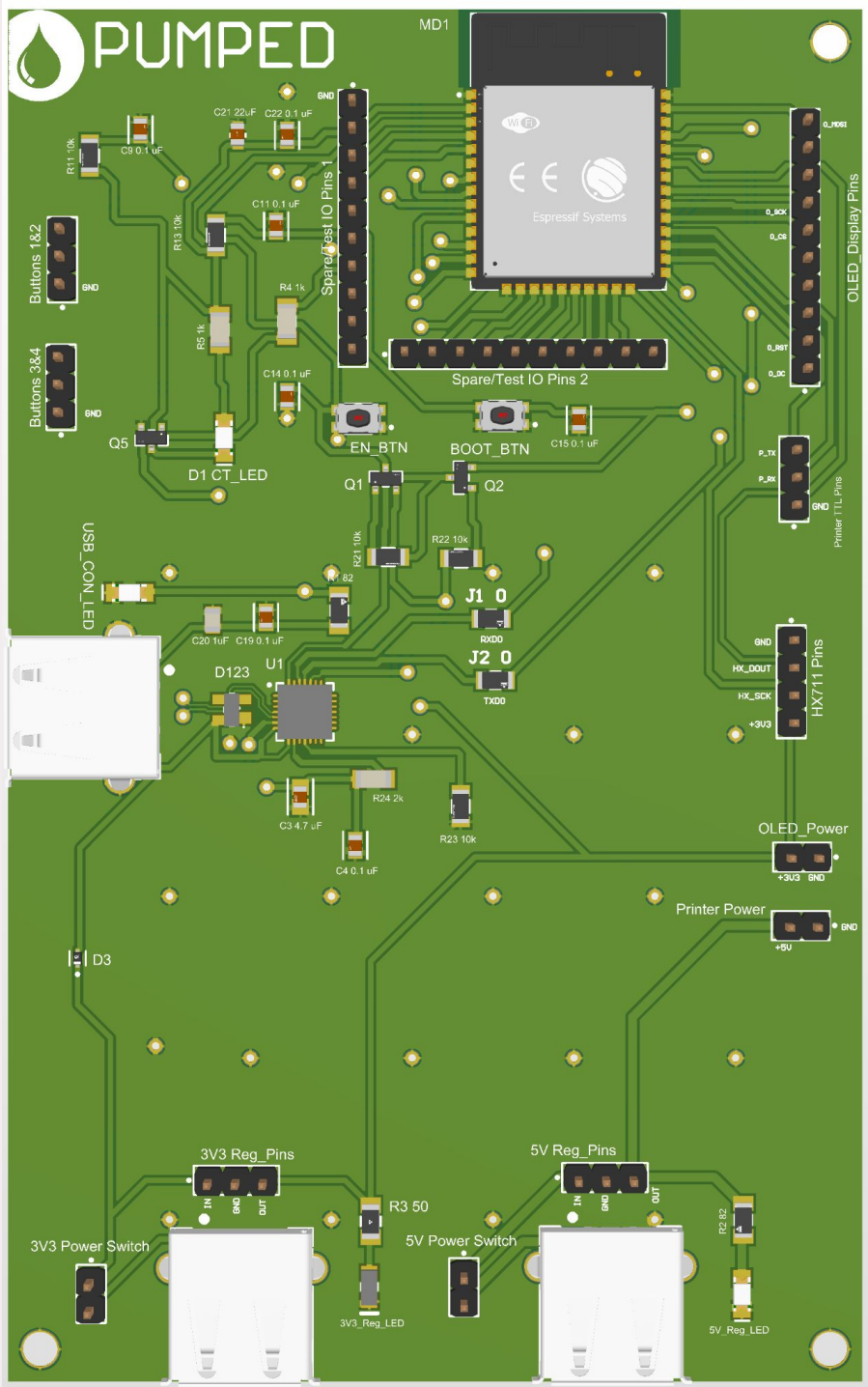


Overall PCB 3D View

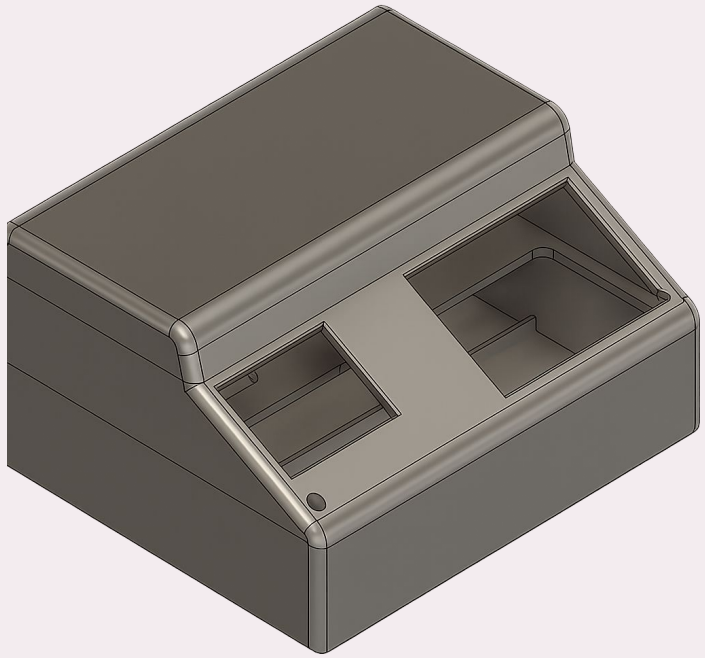


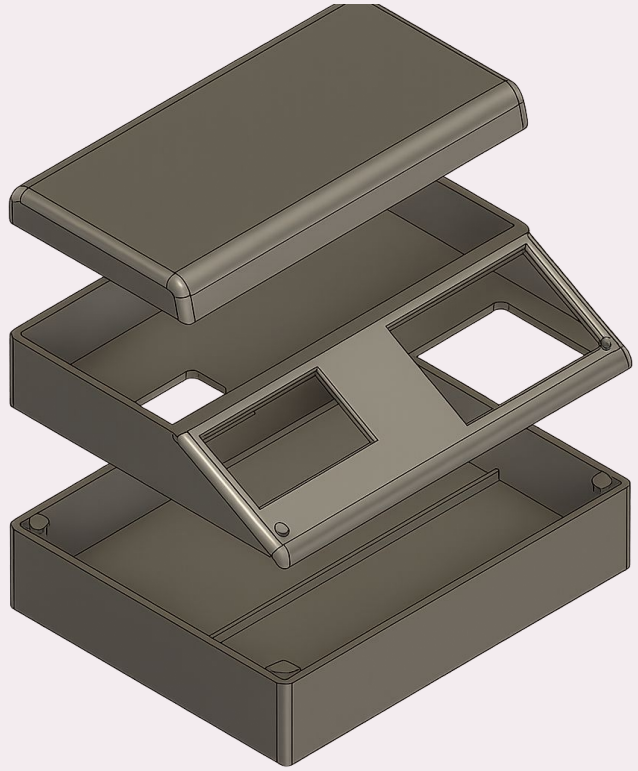


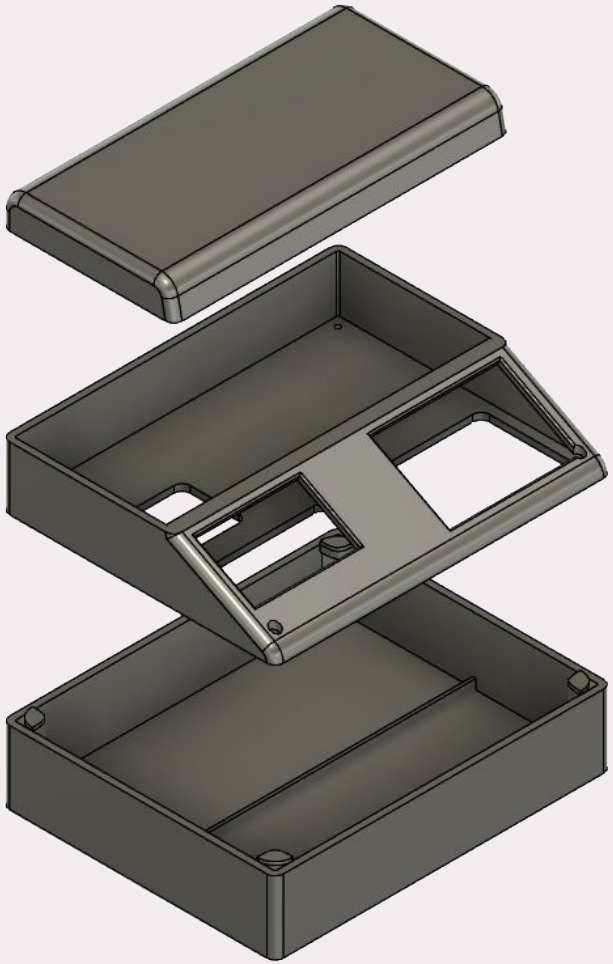


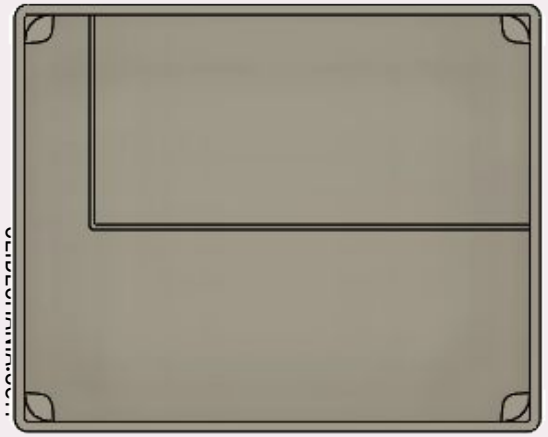


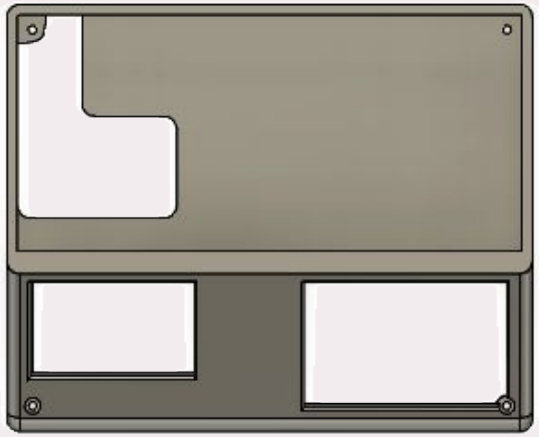
PUMPED Enclosure Overview (Unfinished)



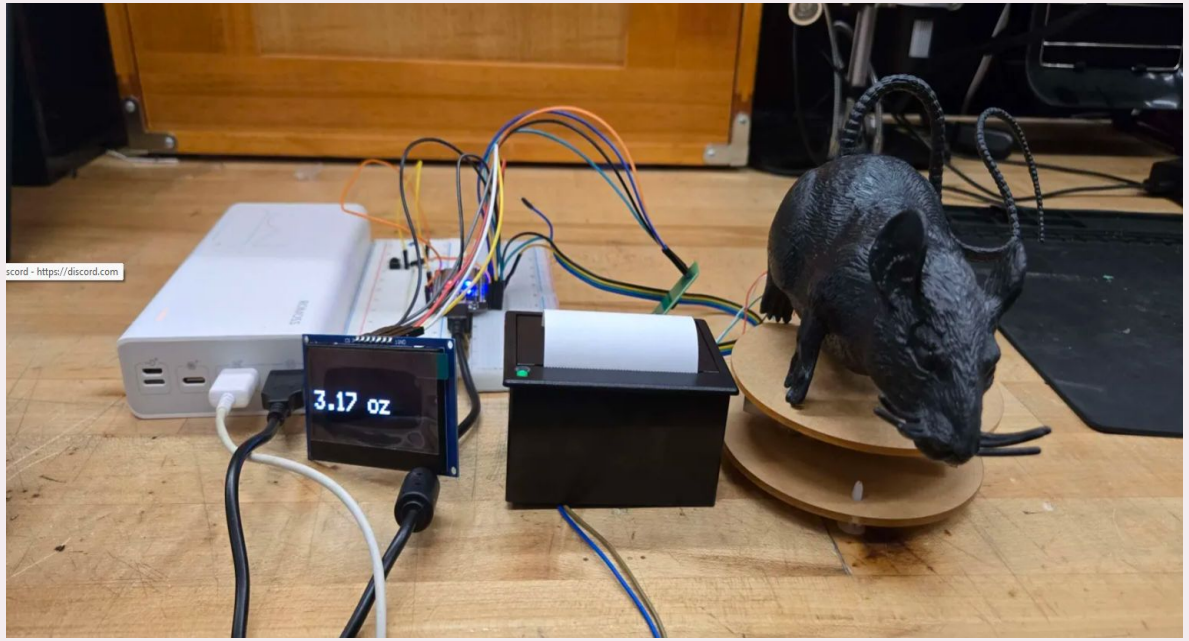






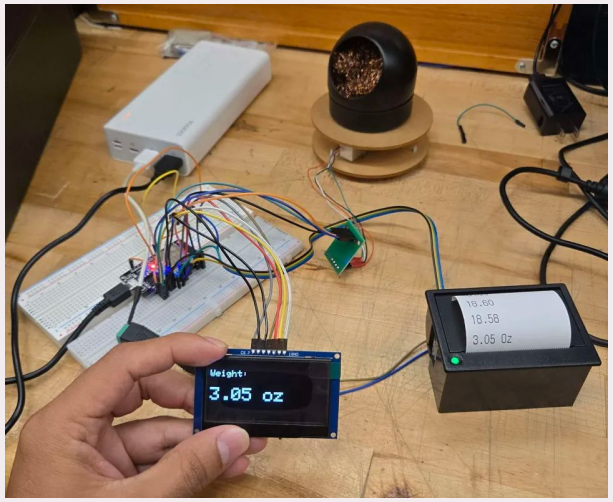


Prototype



When working with our prototype we were able to get all of the subsystems to work together to produce a label with a printed weight on it.

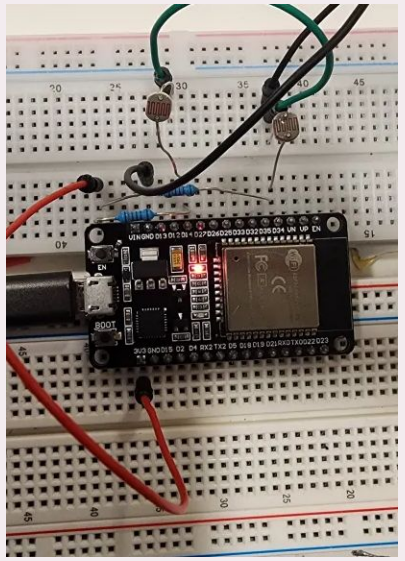

```
#include "HX711.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Pushbutton.h>

// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 2;
const int LOADCELL_SCK_PIN = 3;
```



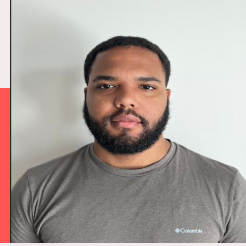
Output Serial Monitor ×

Message (Enter to send message to 'ESP32-WROOM-DA Module' on

```
GPIO34: 1907    ||    GPIO14: 1456
GPIO34: 2702 --> Battery 50%    ||    GPIO14: 2032
GPIO34: 3120 --> Battery 50%    ||    GPIO14: 2288
GPIO34: 3186 --> Battery 50%    ||    GPIO14: 2335
GPIO34: 3212 --> Battery 50%    ||    GPIO14: 2323
GPIO34: 3194 --> Battery 50%    ||    GPIO14: 2341
GPIO34: 3188 --> Battery 50%    ||    GPIO14: 2347
GPIO34: 3206 --> Battery 50%    ||    GPIO14: 2336
GPIO34: 3195 --> Battery 50%    ||    GPIO14: 2343
GPIO34: 3175 --> Battery 50%    ||    GPIO14: 2354
```



Prototype Troubleshooting



Problem:
One issue we came across when building the prototype was brought on by the first thermal printer we wanted to use. In the datasheet the printer was rated for 5 volts to 20 volts at 1 amp but when we tested the printer we found that it would only print legibly when its input voltage was at 17 volts or above. This was an issue because in our proposed design we were not anticipating needing a power supply that could produce more than 12 volts.

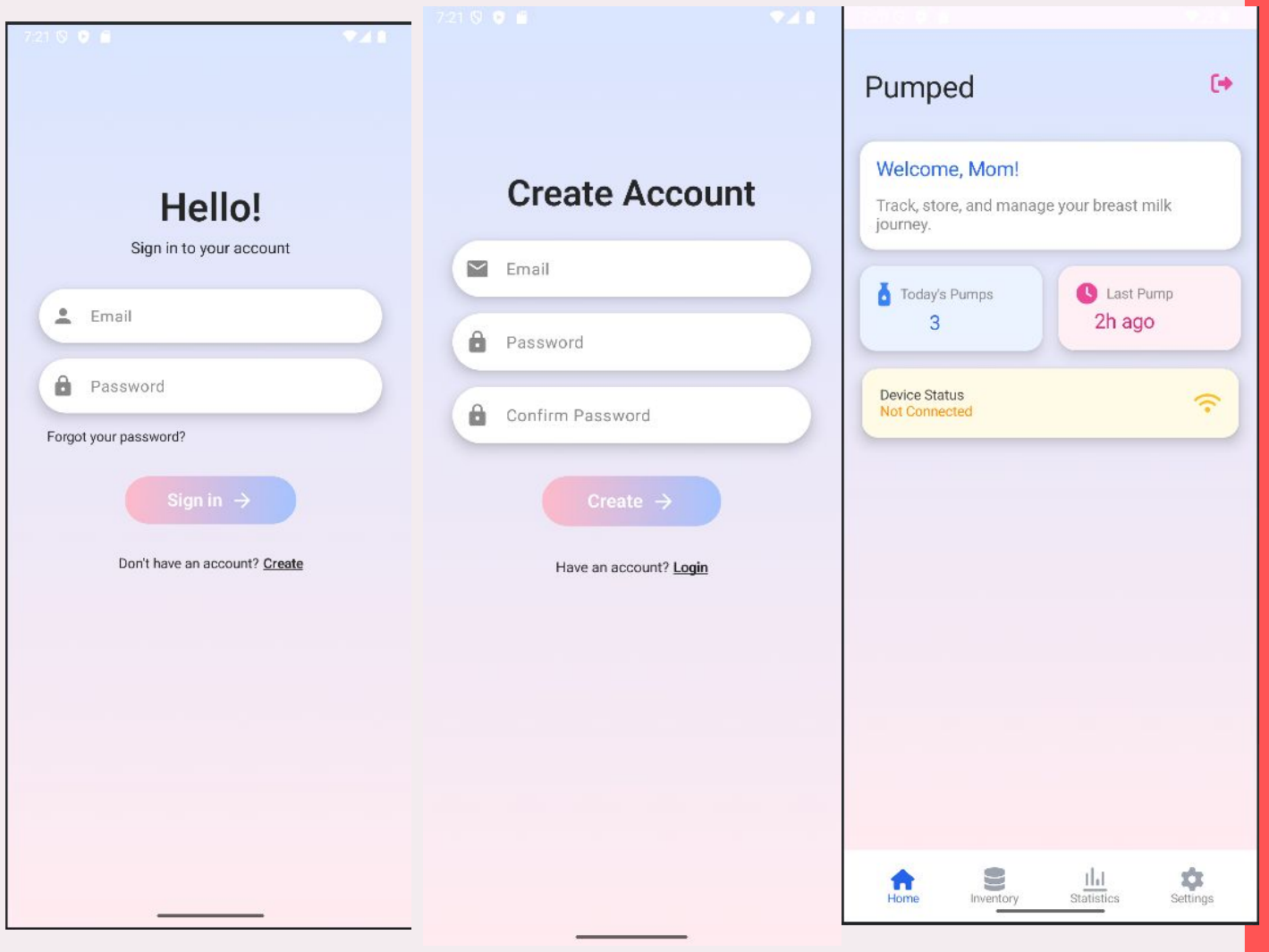
Considered Solution:
Replace the battery with one that could meet the printers voltage requirement.

Solution:
We decided to switch out the thermal printer with a different brand that is rated for 5 to 9 volts at 2 amps. This printer worked as intended with our power supply even with the increased current requirement.

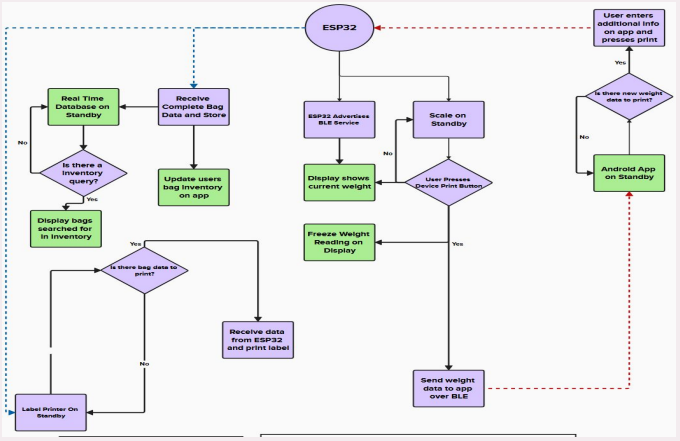
Mobile Application

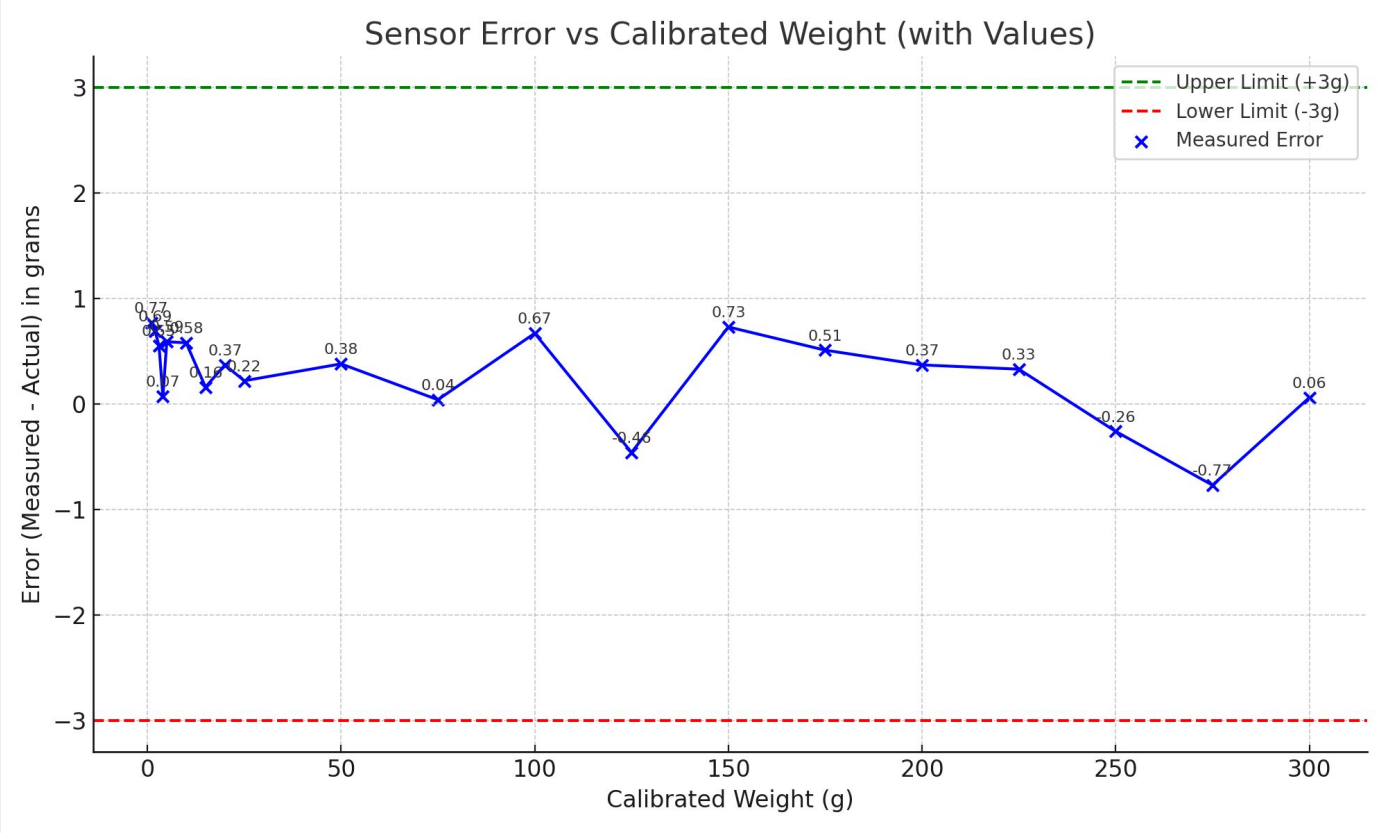
The automatic label printer device will have an associated Android mobile app.

- Built in Android Studio using Kotlin + Jetpack Compose for the UI.
- App is connected to Firebase Realtime Database.
- The app sends and receives data from the ESP32 via a BLE connection.

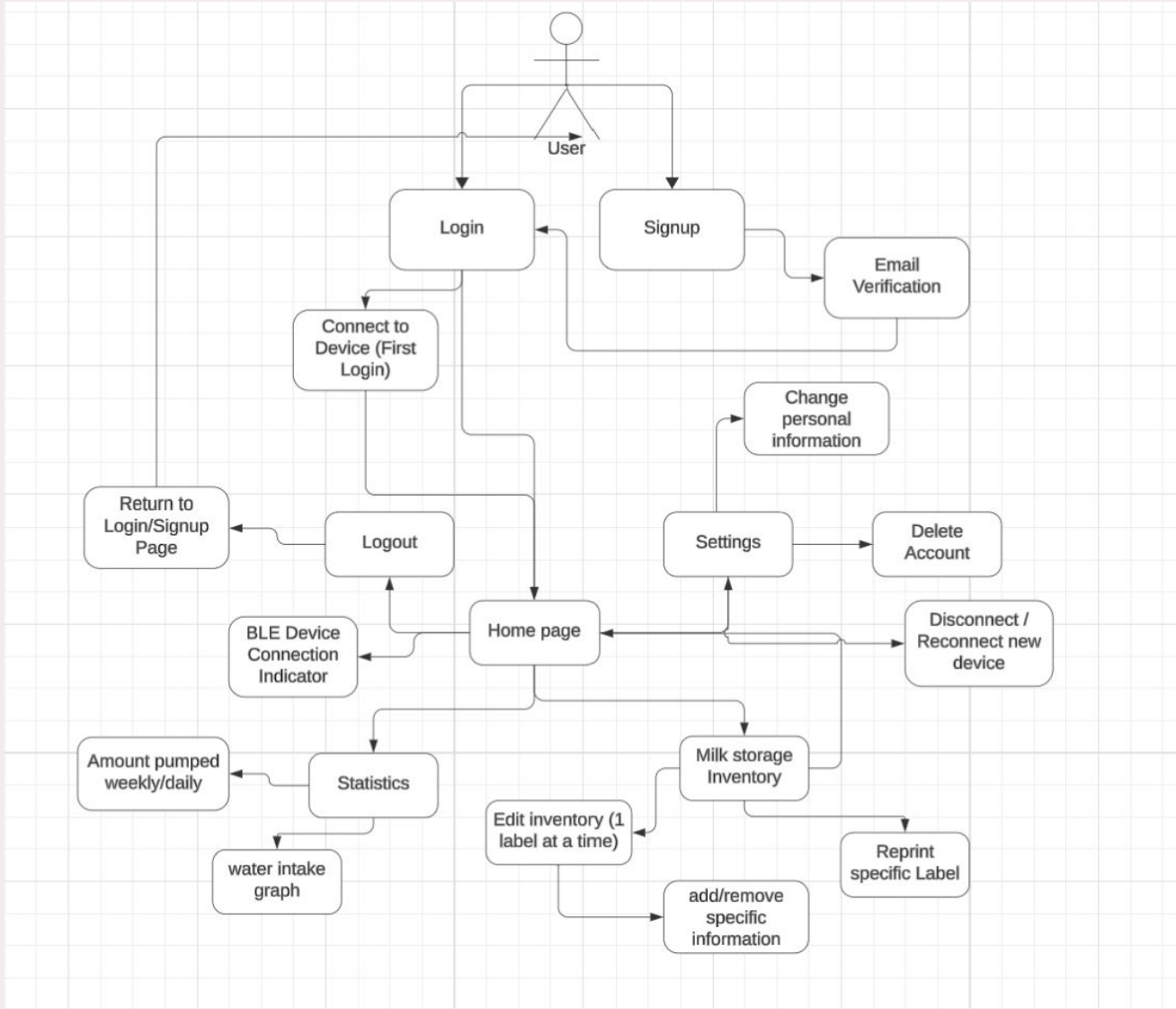


Software Block Diagram





Mobile App Block Diagram



Frontend Comparison and Selection

- Flutter
  - Cross platform (Android, IOS, Web)
  - Dart Language
  - Widget Based UI
  - Growing rapidly
  - Android Studio, VSCode
- React Native
  - Cross Platform (Android, IOS)
  - JavaScript/TypeScript, CSS
  - Component Based UI
  - Mature, huge communities and libraries
  - VSCode Android Studio, WebStorm
- XML
  - Not Cross Platform (Android only UI)
  - XML + Java/Kotlin for logic
  - Layout files, Separate code UI
  - Legacy layout, not recommended to be used in new projects, only used to maintain older projects.
  - Mature Community
  - Android Studio

Front End Framework of choice

- Jetpack Compose
  - Not Cross Platform (Android only)
  - Kotlin (Similar to Java, classes and functions)
  - Function Based UI
  - Built in Live Preview & Live updates
  - Growing
  - Modern features
  - Android Studio

  - Tight integration with modern Android APIs.
  - Seamless integration with jetpack libraries.
  - Positioned by google as the future of Android UI Development.

  - We wanted to solely focus on Android and not IOS because we have previously ran into issues with connections and development using a cross platform framework. We also wanted to get the best features that Android Studios and Jetpack Compose has to offer.

Backend Comparison and Selection

Among the options available for software's backend we decided on using Firebase. Firebase is a backend as a service (BaaS) owned and maintained by Google. We choose it for various reasons including:

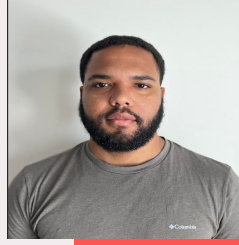
- Provides real time database to ensure minimum latency for device.
- Ease of use thanks to many backend features being prebuilt.
- Generous free tier that we are unlikely to exceed.
- Provides our sponsors the ability to scale this device if they choose.

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)

Budget Provided by Sponsors: $500

Part	Description	Quantity + Total Price
ESP32	MCU	1 + $14.78
OLED Display	Display	1 + $18.09
Battery Bank	Battery	1 + $28.75
Thermal Printer	Printer	2 + $99.86
DFRobot Thermal Printer	Test Printer	1 + $53.00
Thermal Label Paper	Paper	4 + $10.95
Load Cell + HX711	Scale components	1 + $10.95
PCB	PCB design	10 + $113.64
PCB Components	Resistors, Capacitors, Inductors, Buttons, etc.	1 + 88.16
Total		$438.18

Work Distribution



Name	Work Assigned
Kevin Fernandez	• Main MCU PCB design, soldering, and troubleshooting • Photoresistor prototype programming and testing • Photoresistor Enclosure 3D modeling and Printing • PUMPED Enclosure 3D modeling and Printing
Jimmie Rawls	• Regulator PCB design, soldering, and troubleshooting • Load cell/ADC implementation • General troubleshooting and programming
Eric Hernandez	• Team Leader • Battery Percentage Indicator Design, soldering, and troubleshooting • Front End development for the mobile application. • OLED Display UI
Agustin Rodriguez	• ESP32 Printer Driver • ESP32 BLE Connectivity • Android Development • Backend Development

Progress Plan

Task	Percentage
Research + Report	90%
Component Testing	100%
PCB Design + Testing (All)	70%
Front End development	65%
Backend Development	20%
3D Model Design	60%
Total Completion	67.5%