P.U.M.P.E.D(Precision Weighing, User-Friendly Heating, Mobile App Integration, Printed Labels, Expiration Tracking, and Data Management)

Zachary Celentano, Jason Devito, Jamieson Foley, and Isaiah Robinson

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* – Safely preparing breast milk demands precise portioning, controlled warming, and verifiable record keeping. P.U.M.P.E.D unifies these tasks in a palm sized appliance that weighs, heats, and labels milk bags while synchronizing data to a Swiftbased iOS application through an Expresspowered REST API. An ESP32 controls a 10 kg load cell scale, a 12V resistive heater, A Type K thermocouple monitored by a MAX6675 , and a TTL thermal printer. Tentrial validation demonstrated ≤ 1 % weight error, water reaching 115 °F in ten minutes, and 0.1 s LCD latency. By integrating hardware accuracy with mobile inventory tracking, P.U.M.P.E.D streamlines neonatal feeding workflows and bolsters caregiver confidence.**

***Index Terms* — Database Systems, Health Information, Mobile Application, Pediatrics, Temperature Measurement, Water Heating, and Weight Measurement**

I. Introduction

Expressed breast milk enables flexible feeding schedules yet imposes strict handling requirements for volume, temperature, and storage duration. Caregivers often eyeball measurements, guess serving temperature, and hand write labels; these practices violate Centers for Disease Control and Prevention guidelines and increase the risk of nutrient loss or bacterial growth. Existing devices address individual steps such as scales, bottle warmers, or mobile trackers, but none provide an integrated, affordable solution. P.U.M.P.E.D fills this gap by combining precision weighing, closed loop heating, and automatic label printing in a single Internet connected unit. Data from every action syncs to an iOS app, which provides real time inventory counts and expiration alerts. This paper describes the system architecture, hardware implementation, mobile software, and experimental results that confirm compliance with the design goals.

II. System Components

The P.U.M.P.E.D platform is organized into six tightly coupled modules, each chosen to minimize cost while meeting neonatal feeding safety margins. Figure 2 later in the paper traces the signal flow among these modules; the following subsections describe their hardware and firmware roles in depth.

A. Microcontroller

An ESP32 WROOM 32 serves as the system brain. Its dual Tensilica LX6 cores run at 2.4 GHz with 512 kB SRAM ample headroom for concurrent FreeRTOS tasks. One core handles hard real time duties (load cell sampling, PID heater control, SPI thermocouple reads), while the second core manages user interface updates, HTTP requests, and power saving transitions. Integrated 802.11 b/g/n WiFi eliminates the need for an external radio, and on-chip touchpad GPIOs provide low debounce user buttons. The module is powered from a regulated 3.3 V rail and draws 24 mA in active mode and 0.15 mA in deep sleep.

B. Load Cell Scale

P.U.M.P.E.D uses a 10 kg singlepoint aluminum strain gauge load cell mounted beneath a stainless steel plate. The bridge is excited at 5 V and digitized by an HX711 24-bit delta sigma ADC running at 80 samples s^{-1}. Two Point calibration with NISTtraceable 100 g and 1000 g masses yields a linear transfer function with ±0.05 % full scale nonlinearity. Firmware averages ten consecutive readings to achieve a noise floor of 0.1 g, after which a moving average filter compensates for residual vibration. Temperature drift measured across 20–40 °C is 3.2 g, well below the ±1 % volume target for typical 4–16 fl oz milk bags.

C. Heating Module

A 3D Printer heated plate paired with a Universal Regulated Switching power supply. At 220 Watts and 24 Volts the hotbed's function is to heat the water, placed in our metal container, to a max temperature of 115 degrees.

D. Thermocouple Sensor

A 1/8in diameter typeK probe immersed in the bath feeds a MAX6675 cold junction compensated converter. Samples arrive every 220 ms over SPI (CS = GPIO5, SCK = GPIO18, SO = GPIO19). Factory calibration limits error to ±1.5 °F between 32 °F and 176 °F;

laboratory comparison against a Fluke 51 II reference showed ±1.2 °F across the operating range. Firmware averages four samples and converts to Fahrenheit before passing the value to the PID controller and LCD.

E. TTL Thermal Printer

A 58 mm thermal receipt printer (Adafruit PN 597) accepts 9600 baud rate TTL serial on UART2 (TX = GPIO17, RX = GPIO16). It prints 60 mm long, self-adhesive labels containing a logo header, volume in fl oz & mL, ISO8601 timestamp, and expiration. The printer operates from the 12 V regulator and draws up to 2.5 A when firing.

F. Relay

The relay in our design essentially functions as both a switch and a PID controller. The heating element in our design will require a separate larger power supply than the rest of the system that will need to be controlled by our ESP32. The relay will receive at one end a 24 Volts signal at 10 Amps, and the other end will be the control signal from the ESP32.

G. Power Supply

The power supply is not just one component but rather a group of components that include a signal transformer, rectification PCB, a 3.3 Volt regulator, a 5 Volt Regulator, and a 12 Volt Regulator. The power supply will also include another section as this device needs to power supplies, one for the main components and peripherals and the other for the heating element. The next section will talk about the secondary power supply.

H. Switching Power Supply

To power the hot bed we will implement a 400 Watt 24 Volt Switching Power Supply. This power supply is 8.5 x 4.4 x 2 in. Its main purpose is to power the heating module through converting the initial AC input from a receptacle into a DC input for the heating module. A relay will be implemented before our main power supply to branch off of. The secondary power supply (the switching power supply) will connect from the relay to then power the hot plate.

I. Mobile Application

The companion iOS app is written in Swift with Xcode for broad version support. At launch it runs on a lightweight JavaScript express powered backend with API points exposing measurement functionality(POST, DELETE, PATCH, GET). The app shows inventory of pumped milk alongside an OpenAI powered assistant. . Payloads arrive via HTTP POST from the ESP32 and are cached in PostgreSQL which can be accessed by the server. The UI provides real-time inventory counts, and low supply push notifications via email when inventory falls below a threshold.

III. System Concept

In order to have a complete understanding of our system, a hardware block diagram, software flowchart and state diagram are useful.

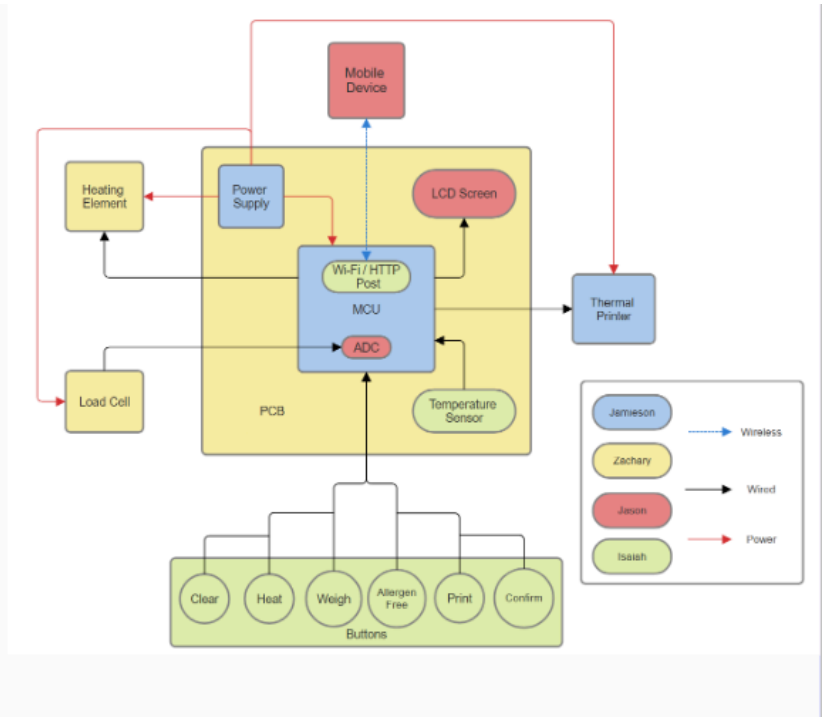


Fig. 1. Complete hardware block diagram that shows all hardware components and how they interact with each other.

As shown in the figure above, a visual map of the P.U.M.P.E.D architecture, the narrative below walks through the signal flow that links a caregiver's button press to a live database entry on the mobile device. The six push buttons on the front panel generate rising-edge interrupts on dedicated ESP32 GPIO pins. An interrupt service routine identifies which button fired, stamps the event, and posts it to an in-memory queue where the real-time tasks can collect it without risking missed edges. This scheme keeps latency low while avoiding the timing jitter that often plagues delay-based polling loops.

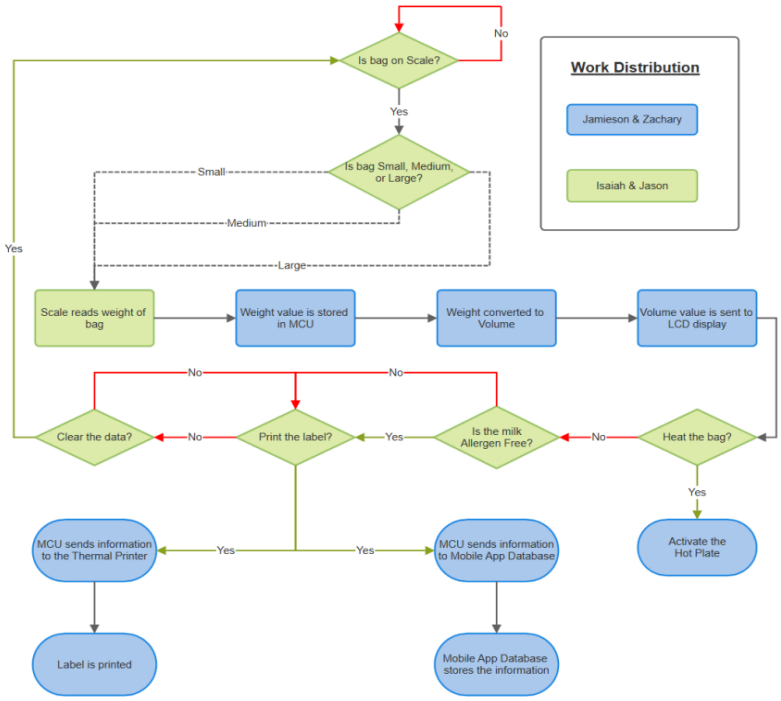


Fig. 2. Complete software flowchart that shows how the user's interactions affect the system and the processes involved.

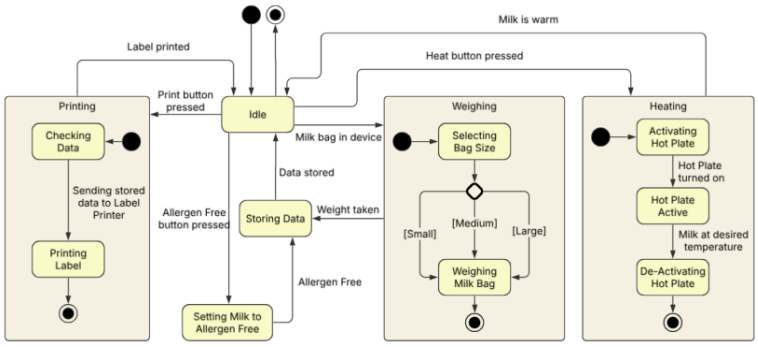


Fig. 3. Complete state diagram that shows the transitions of states that occur within the system.

Once a **Weigh** event reaches the WeightTask, the firmware enables the HX711 amplifier, samples ten consecutive readings at eighty samples per second, and computes an average mass. The result is converted from grams to both fluid ounces and millilitres using a density constant of one point zero three grams per millilitre. The LCD then reveals the three values in a large font so that the caregiver can confirm the measurement. If the user presses **Confirm**, the value is frozen in non-volatile memory; otherwise, a second tap on **Clear** tares the scale and restarts the weighing sequence.

A short press on **Heat** activates the heating loop while a long press of more than one point two seconds simply reports the current bath temperature, allowing a quick safety check. During active heating, the MAX6675 converter supplies a new sample every two hundred twenty milliseconds. The PID algorithm updates the duty cycle of a logic-level MOSFET that modulates the twelve-volt rail feeding the sixty-watt silicone pad. Temperature and set-point values are streamed to the LCD so the caregiver knows exactly when the milk has reached one hundred fifteen degrees Fahrenheit.

When both weight and temperature are satisfactory, the caregiver triggers **Print**. The PrintTask constructs a fixed-width string that holds volume, timestamp, expiration date, and an allergen flag if the **Allergen-Free** key was toggled earlier in the session. The string passes to the TTL printer over the secondary UART at ninety-six hundred baud, producing a sixty-millimetre self-adhesive label in about four seconds. Immediately after the printer acknowledges the final line feed, the NetTask serialises the same data into JSON and sends a Hypertext Transfer Protocol post request to the Express server that runs inside the iOS application.

On the mobile side, Express parses the request, writes a new row to the PostgreSQL database, and pushes an update through a post request to the frontend where the live inventory is updated. From there users can choose to, delete, or even manually add an entry to the inventory if needed. When supply is low a push notification will be sent via email notifying the user that milk is running out. Additionally, as the user utilizes the milk bags the amount of fluid ounces can be edited within the app. To ensure a friendly and easy user experience, OpenAI embedding can be used within the mobile app to answer any related breast milk or pumping questions. It is important to note that this OpenAI API will not retrieve any user information or data from the inventory, it is strictly for questions and answers. With full integration, the result is a seamless loop: one set of buttons drives weighing, heating, and printing in the physical world, while the same data appear instantly in the digital realm without any manual transcription by the caregiver.

A. Equations

When building our design we had to figure out the power draw of each component. After finding out the total power draw the decision on a power source that

could sufficiently power our device could be chosen. To calculate the power consumption of each individual component we used the basic power formula provided below.

$$Power = Voltage * Current \quad (1)$$

Weighing the Milk bags is a crucial part of our design. With that in mind, we needed to implement an equation to take the raw input data from the load cell and convert it to a legible number that could be displayed on the LCD screen. The formula for the Load Cell weight conversion is displayed below

$$Volume = Mass / Density. \quad (2)$$

IV. Hardware Detail

A. Load Cell and Amplifier

For the purpose of this design only minimal knowledge of a Load cell was needed:

(1)A load cell is a force sensor that measures weight and converts the applied mechanical force into an electric signal. This force creates mechanical deformation in the load cell's internal structure.

(2)The strain gauge of the load cell is part of a Wheatstone Bridge Circuit. When the strain gauge's resistance changes the voltage balance in the bridge shifts.

(3)The load cell is powered through the HX711 Amplifier that also doubles as a way to convert the mechanical force into a readable electrical signal.

Full-bridge strain gauge circuit

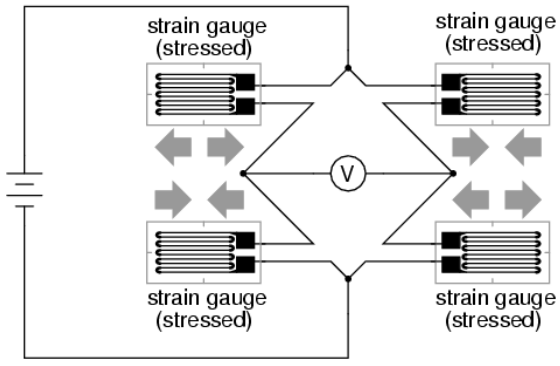


Fig. 4. A general diagram of the Wheatstone Bridge Circuit that is a Load Cell.

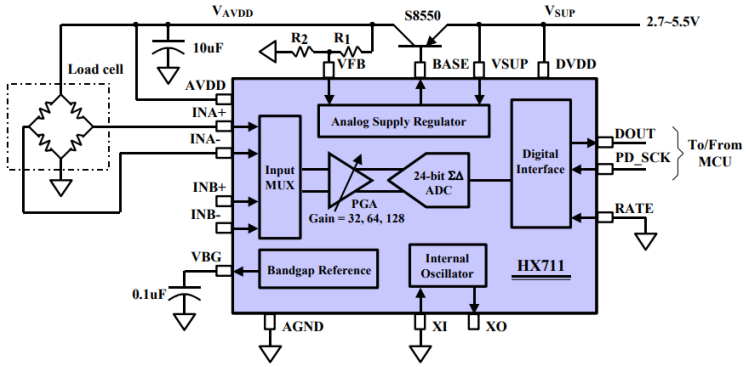


Fig. 5. A diagram of the HX711 Amplifier used to convert the mechanical deformation of the load cell into a readable electrical signal.

The load cell is a high accuracy scale that allows for a wide range of measurable weights. With simple integration with the ESP32 and the HX711 amplifier, our design will accurately weigh the intended milk bags at a ±1g difference.

B. Temperature Monitoring

The temperature monitoring system consists of a TypeK thermocouple, A MAX6675 amplifier, and an LCD screen. When measuring the temperature of the water the Type K thermocouple will be inserted into the container of water. The tip of the sensor is all that is needed for the readings to occur.

Once the sensor gathers the data it is sent down to the MAX6675 and the raw input data is converted to Celsius and Fahrenheit. After the conversion, the readings are sent to the MCU. The MCU then processes the data received from the amplifier and then sends the readable data to the LCD screen to be displayed.

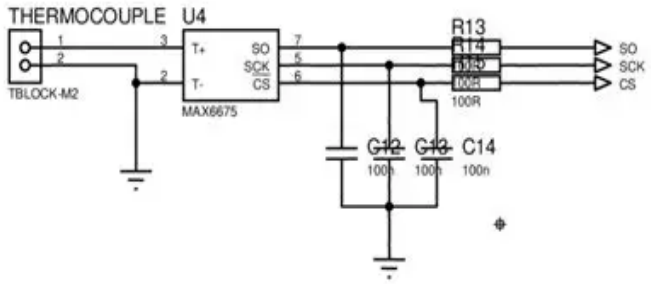


Fig. 6. A diagram of the MAX6675 Amplifier used to amplify the small voltage signal from the thermocouple. This conversion is from analog to a 12-bit digital signal.

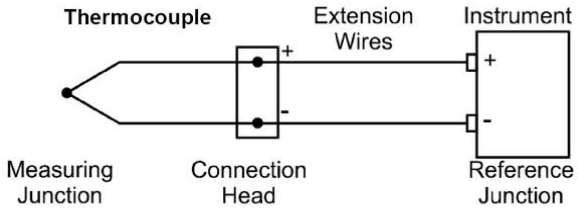


Fig. 7. A general schematic of the TypeK Thermocouple.

As the measuring junction experiences heat or cold, it will generate a small voltage due to the Seebeck effect. This effect is due to the two different conductive materials joined at the measuring junction. At this junction the sensor is changing from hot to cold depending on the element it is touching. The reference junction will not be the same as the measuring junction and therefore will cause a thermoelectric voltage between the two junctions. This temperature difference is what causes the difference in voltage. The voltage produced is then measured and converted into a temperature reading.

C. Heating Module with a Relay

To manage the heating within our system, an intermediate level of knowledge was required in the context of control systems:

(1) How to power a 3D Printer Hot Bed with a Switching Power supply

(2) Setting up a switching power supply to receive power from a receptacle using a stripped power cord

(3) Implementation of a relay, which is a form of a switch that allows comparatively smaller electrical sources to control the output of a larger electrical load.

Fig. 8. Above is the inside of the 3D Printer Hot Bed..

The 3D Printer bed heater is a resistive heating element that heats up as more current runs through the wires. With the relay and power supply connected this current will be controlled to allow the heating element to be turned up and shut off at the user's expense.

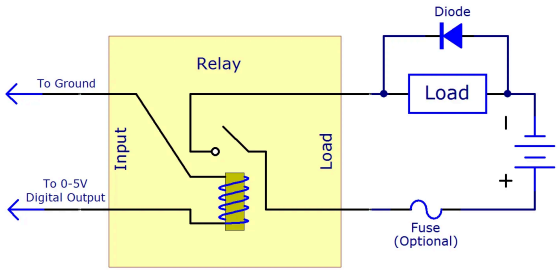


Fig. 9. A schematic of a basic relay

The relay functions as a switch, which upon closing completes the circuit and allows the current to flow to our heating element from our switching power supply, through control of the ESP32. In our case, this relay is not a mechanical switch, but rather a Solid State Relay, which utilizes semiconductor technology to turn the heating element on.

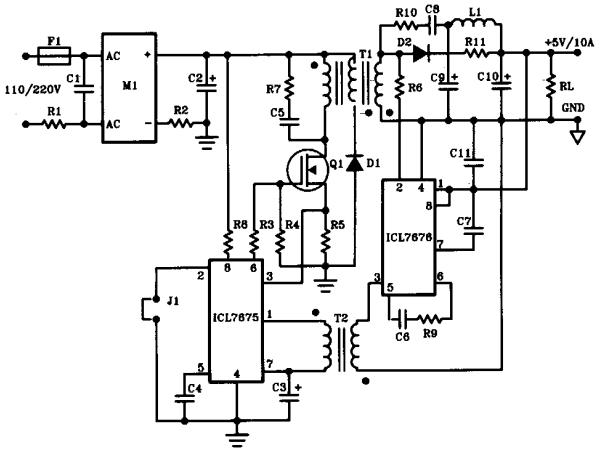


Fig. 10. A schematic of a basic Switching Power Supply

The Switching Power supply will convert the AC input into a usable DC output, which will be used by the 3D Printer Hot Bed. This Power supply can handle 110 or 220 Volts and can be switched between both through a lever embedded into the circuit. The output will be a constant DC signal at 24 Volts , 16.5 Amps, and 400 Watts. The output is sufficient to heat the heating pad to 115 degrees in 15 minutes.

D. Thermal Printer

The thermal printer is a wonderful addition to the system already in place. It allows for the user to keep track of when the milk was bagged and how long the bags can be stored for. The thermal paper used will have a sticky side that will be applied onto the bag to keep the information connected to the bag. This storage system is not only attached to the bags itself but also implemented into a mobile application. The MCU will communicate through Bluetooth and give data to be stored in the cloud storage inside the Mobile Application. There will be no actual storage when it comes to our device, but rather a virtual storage system through the mobile application.

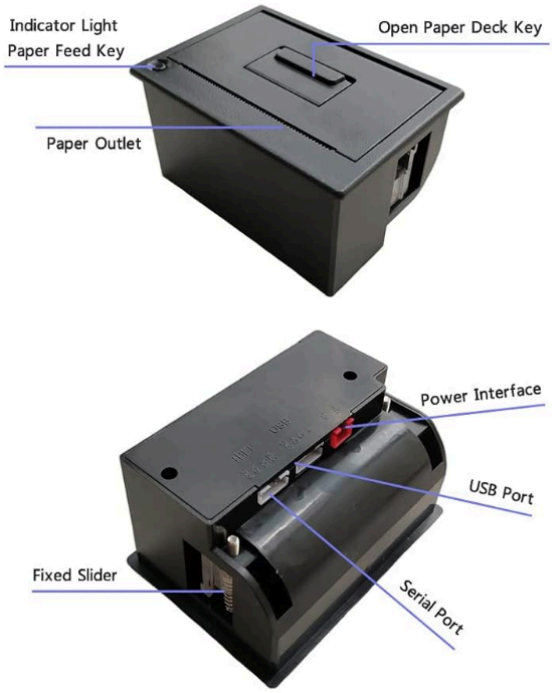


Fig. 11. A picture describing all of the thermal printer parts.

V. Software Architecture

The objective of this project is to design a device that allows for a user to quickly and easily determine the volume of a bag of milk, then print a label with all the relevant information. To add to that, there should also be an option for a user to heat up a bag of milk to a suitable temperature for consumption.

A. System Firmware

To accomplish the objectives of our project above, we wrote the software controlling our system using C++. We referenced many in-built libraries, as well as other additional libraries for our code. The loop() C++ function contains most of the other functions that are used to activate the processes of our device, with each having a corresponding button. The system continuously loops through that function, acting as an idle state, until the user interacts with one of the buttons connected to a separate process of the system. The loop() function contains the following function calls:

```
if(digitalRead(weightButton) == HIGH) {
    weighMilk(bool weigh); }
if(digitalRead(heatButton) == HIGH) {
    heatMilk(bool heat); }
if(digitalRead(labelButton) == HIGH) {
    printLabel(bool label); }
```

Fig. 12. Simple representation of the loop() function.

(1)The weighMilk() function acts as the method to activate the load cell and store the value of the mass of the milk bag. Once the mass is found, it is then converted into volume.
(2)The heatMilk() function acts as the method to activate both the heating element and the temperature sensor.
(3)The printLabel() function acts as the method to activate the thermal printer, which will create a label with all the relevant information for the bag of milk.
(4)The digitalRead(button) function acts as the method to determine whether or not the user has interacted with the corresponding button.

When the user interacts with the button corresponding to weight, that triggers the weighMilk() function. Going further into detail of this function, the purpose is to activate the load cell and run a continuous while() loop, which will repeatedly take the mass value being read from the load cell. Whenever a mass value is taken, that will trigger the convert2Volume(weight) function, which converts the mass value (grams) to the volume values (fluid ounces & milliliters). At any time, the user can interact with the "Confirm" button to exit the while() loop and store the current volume values in the microcontroller.

```
float weighMilk(bool weigh) {
   float weight;
   while(weigh) {
      if(digitalRead(confirmButton) == HIGH) {
          return fluid, milli; }
       weight = scale.get_units();
       fluid, milli = convert2Volume(weight);
   }
```

Fig. 13. Simple representation of the weighMilk() function.

When the user interacts with the button corresponding to printing the label, that triggers the printLabel() function. Going further into detail of this function, the

purpose is to activate the thermal printer, sending all the relevant data stored in the microcontroller to be printed as a label. The information that should be on the label is the volume of the milk bag, the current time & date the breast milk was pumped, the expiration date of the milk and whether or not the milk is allergen free. The volume of the milk bag is determined by the weighMilk() function, explained above. However, the rest of the information is determined in the printLabel() function.

To determine the current time & date, we made use of the ESP32's in-built Wi-Fi functionality to connect to the "pool.ntp.org" NTP time server. Once the current time and date are found, we determine the expiration date for the milk by adding 4 days to the date, as that's how long breast milk stays safe for consumption when properly refrigerated.

Finally, to determine whether the milk is allergen free, we run a while() loop that continuously loops giving the user time to interact with the "Allergen Free" button. If this button is triggered, then the "allergens" value is toggled between true/false. At any time, the user can interact with the "Confirm" button to exit the while() loop and send all the relevant information to the thermal printer, as well as to the mobile app database to be stored.

```
void printLabel(bool label, float fluid, float milli) {
    getTime();
    while(label) {
        if(digitalRead(confirmButton) == HIGH) {
            label = false;
            printer.print((volume));
            printer.print((time));
            printer.print((date));
            printer.print((expire));
            printer.print((allergens));
        }
        if(digitalRead(allergenButton) == HIGH) {
            allergens = !allergens; }
    }
}
```

Fig. 14. Simple representation of the printLabel() function.

VI. Enclosure Design

The enclosure of our design will encapsulate our hardware components into a smaller compact design. This design has to be able to withstand the stress of regular use as well as the stress of heat from the internal components. For our enclosure we chose to go with an Acrylonitrile Butadiene Styrene filament (ABS). We chose to go with this filament because:

(1)Strength. ABS has a high tensile strength and is impact resistant.

(2)Flexibility. ABS is more flexible than its counterpart PLA, while also being less brittle.

(3)Durability. ABS can withstand stress and wear overtime, which is crucial for a product that will be used daily.

(4)Heat resistant. ABS has a higher heat tolerance than PLA and melts at ~220-250 degrees Celsius.

These strengths not only outline why ABS was the better filament choice, but will allow us to create an enclosure that is durable and heat tolerant.

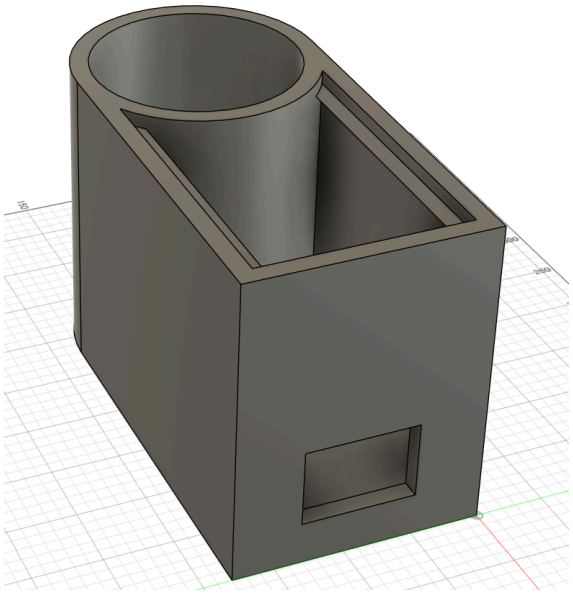


Fig. 15. A general schematic of the enclosure design.

In the figure above we can see two different compartments. The cylindrical compartment will hold the container that will be used to weigh the milk bags. Below that container, the load cell will be configured in a way that the container will rest upon the load cell to ensure accurate measurements.

The closer compartment will house the PCB components, power supplies, thermal printer, and the necessary space to allow for cooling. The shape of this compartment is built to allow for stacking of the PCBs. They will be stacked on top of each other to create a compact design. As we stack the PCBs we will ensure there is enough space between to allow for cooling.

The frontward facing wall has a hole in it to account for the thermal printer. This hole allows the thermal printer to be connected to the MCU, while having an opening for the thermal paper to dispense to.

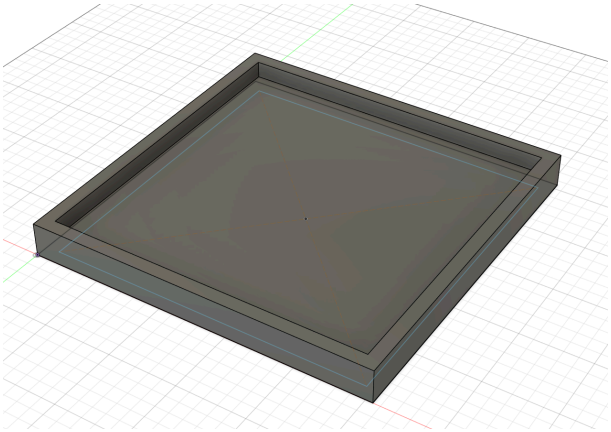


Fig. 16. A general schematic of the Heating Bed Enclosure.

In the figure above we can see the schematic of the Heated Bed enclosure. This enclosure is not enclosing the heating pad, but rather allowing the heating pad to rest upon it to allow for safe use. The Heating Bed Enclosure will be attached to the main enclosure seen in Figure 8. While not attached initially, the Heating Bed Enclosure will be attached through screws drilled into the right side of the main enclosure, near the base. The Heating Bed Enclosure will also be padded with insulation to prevent damage on the Enclosure.

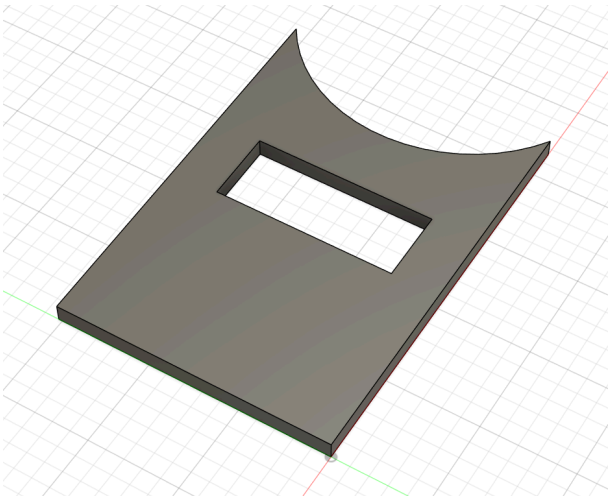
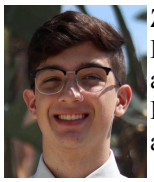


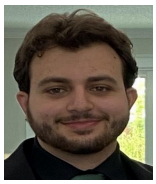
Fig. 17. A general schematic of the TypeK Thermocouple.

Above is the “Lid” that will be applied to the top of the original enclosure. This lid will seal the PCB components, Power supplies, and the Thermal Printer. The rectangle in the middle of the lid is where the LED will be mounted. The LED screen will be mounted from the bottom of the lid so that the screen will be flush with the lid. With the user interaction being a crucial part in this design, we will implement the buttons onto the lid as well. While not on the figure above, holes will be drilled to mount the buttons on the lid for the user to interact with the system.

The Engineers



Zachary Celentano is a 22 year old Electrical Engineering student who has accepted a position as an Electrical Engineer at MAS HVAC in Minnesota after graduating.



Jason Devito is a 22 year old Computer Engineering student who is interning at a security company called Serverli. Jason hopes to land a job as a Machine Learning Engineer or Embedded Software Engineer.

Isaiah Robinson is a 25 year old Computer Engineering student who aspires to work in a field related to Aerospace or Robotics after graduating.

Jamieson Foley is a 22 year old graduating Electrical Engineering student who is interning at Electrical Design Associates and hopes to one day become a PE Engineer and open his own firm.

Acknowledgement

The authors wish to acknowledge the assistance and support of Dr Arthur Weeks, Dr Chung Yong Chan, as well as sponsors Victoria Rodriguez and Matthew Aberman; University of Central Florida.

References

[1] Avia Semiconductor, HX711: 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales, Datasheet, Rev. 2.0, China, [Online]. Available: https://www.aviaic.com/

[2] Random Nerd Tutorials, ESP32 with Load Cell and HX711 Amplifier (Digital Scale), Tutorial, 2022, [Online]. Available:https://randomnerdtutorials.com/esp32-load-cell-hx711/