Smart Vitamin Drink Mix

(Mana)

ECE 445 Final Report

Team # 68

Date: May 1st 2024

Team Members

Andrew Chang (andrew51)

Dhruv Panchmia (dhruvp4)

Horace Yu (horacey2)

TA

Nikhil Arora

Professor

Viktor Gruev

Abstract

The purpose of this report is to outline our design choices, implementation, and reason for developing devices that can dispense the amount of micronutrients the user is lacking based on their daily diet. The system consists of two parts, the actual machine which holds and dispenses the micronutrients, as well as an application that can be run on an external device like a smartphone in which the user can input their meals for the day. This extremely personalized device can allow for the user's quality of life to be significantly improved by ensuring that there are no large deficiencies in any of the important micronutrients that one would intake.

Table of Contents

1. Introduction

1.1 Problem

There are many people who, on a day to day basis, simply aren't getting the correct amount of nutrients such as vitamins as they should due to trivial matters such as diet and not going out in the sun, amongst many others. Deficiencies in these important nutrients can lead to decrease in performance both physically and mentally and if too extreme can lead to further complications.

1.2 Solution

Our solution is a device that can make a powder mix of different supplements based off of the user's nutrients consumed during the day. The way the amount that would need to be supplemented would be calculated would be by inputting information regarding the user (such as how long they were outside, what types of food they had, etc.) into the app on the phone / laptop. There would be calculations made based on the information inputted and then depending on the calculations, a certain mix "recipe" will be sent over to the mix maker to make. After receiving the recipe, the mix maker would turn to align the whichever ingredient's compartment directly above the cup the user has placed. It will then proceed to measure and dispense the required amount of powder for that respective supplement. The mixer will repeat this step until all the required powders with their correct measure have been deposited into the cup. With this, the user can intake their custom mix of supplements for the day, thus solving our problem caused by deficiencies of these nutrients.

1.3 High-Level Requirements

- The supplement mix maker should be able to make the correct mix with a 15 mg margin for each ingredient.
- The mix maker should be able to wirelessly interface with applications on other devices and should receive the correct mix recipe 80% of the time.
- The user will be notified when any storage compartment is below 15% of capacity.

1.4 Subsystem Overview

We divided the project into four distinct subsystems: Mobile Integration, Storage, Dispenser, and Power. Below are the main purposes that we hope each subsystem to serve:

- Mobile Integration: To act as a way to track a user's daily diet and calculate any nutrient deficiencies as well as help the user interface with the mix making device.
- Storage: To contain and track all the nutrients in powder form in each individual compartment as well as the capacity in said compartment. Components in this subsystem will be directly controlled by our microcontroller.
- Dispenser: To dispense the desired amount of nutrient powder from each compartment based on the recipe the device receives from Mobile Integration subsystem. Components in this subsystem will be directly controlled by our microcontroller.
- Power: To provide the desired amount of power to all of our electronic components which operate at either 5V or 3.3V. For easy use, the device can be plugged into a wall socket to draw said power.

Below is our high-level block diagram which shows how the different components of each subsystem operate and interface with each other.

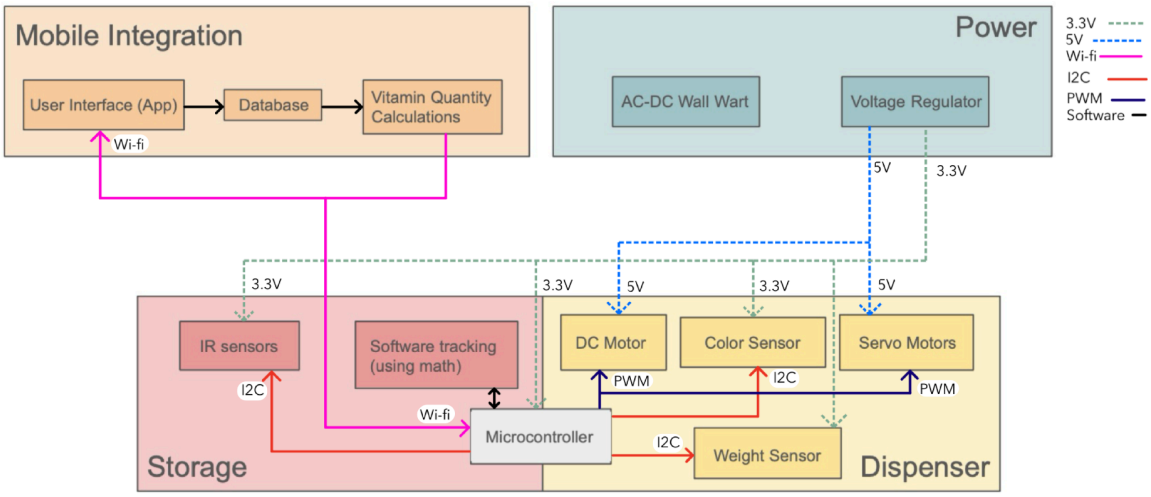

Figure 1. High-level block diagram

2. Design

2.1 Mobile Integration

2.1.1 Design Details

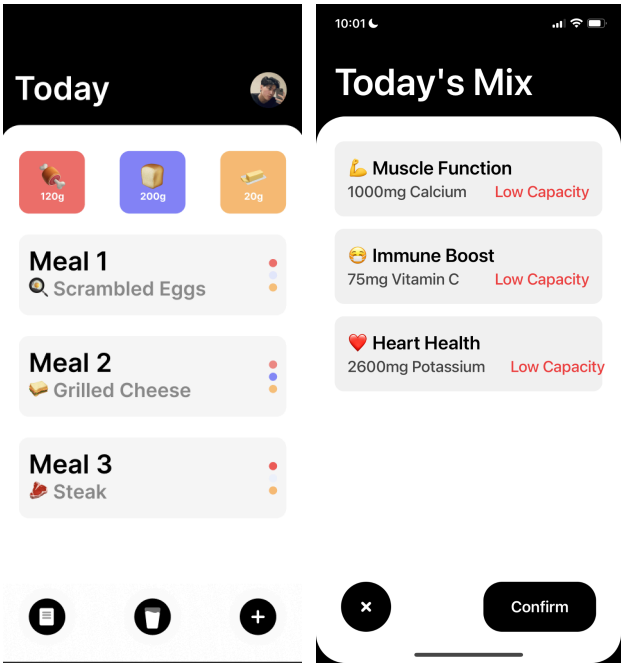

Figure 2. Meal input screen on phone application (Left)

Figure 3. Drink Request with containers at low capacity (Right)

The goal of our mobile application is to provide the user with a platform to communicate with the hardware (requesting drinks), and to gather information on their daily dietary deficiencies and any medical restrictions they have. With this being said, we had to design our Mobile Subsystem around these core values, and we did so by creating both a user friendly front end, and a server to host communication between the app and the hardware through fetch requests. The app can allow the user to send requests to the server and the hardware will always be listening through fetch requests. Since we needed to allow both the mobile app and the ESP32 to communicate with the server, we hosted the server locally and had to forward the port to a public address to enable server access to both clients.

2.1.2 Requirements

Our first requirement for the Mobile Subsystem was that our database must return the macronutrients and micronutrients of a food item when a query is made with a 90% success rate. The goal of this requirement was to create a good recipe that would cover any nutritional deficiencies our user had. In order to do that, we need to know what nutrients they are in fact getting so we can create a recipe to supplement what's missing. In hindsight, we ended up using the server to host this information instead of using a database just to simplify our software architecture, but the concept remained the same.

Our second requirement was that given what the user had eaten on a given day, we need to generate a recipe that covers any nutritional deficiencies from their meals with an 80% success rate. As for design choices, we again took the approach of hosting the calculations and recipe generation on the server. Since we store all the information for the food items that the user can select as a meal on the backend, the calculations and recipe generation were very simple. All we needed to do was send what the user ate on a given day when they requested a drink from the mobile app, and this was done with just a POST request

Our last requirement was that our hardware must be able to receive and process a drink request with 80% success rate. The design choice for our hardware was very similar to our frontend, but focused on listening for any state changes on the server. This was done by having a constant loop that would send a GET request on timed intervals in order to check if a drink request was being sent. When the mobile app sends a recipe, it would be stored on the server and the hardware would retrieve it and effectively create a communication loop between the frontend and hardware over wifi. In order to communicate with the server though, we had to forward our local port to a public http address (http and https). Specifically, the ESP32 couldn't process requests to https due to security reasons, so we also had to host a regular http port in addition to the https port for our mobile application, which required https.

2.1.3 Verification

Given that we changed our first requirement from having our database return the macronutrients and micronutrients of a food item when a query to having our server return the data, we set up POST and GET endpoints for the state variables of each meal and our recipe. This way, we could add a meal on the frontend and check in the server logs to confirm the macro and micronutrients are correct. This worked 100% percent of the time out of 10 trials. For every meal that we added on the frontend, we would receive a log from the server listing its macro and micronutrients.

As for creating the recipe using the meals our user inputted, we would use the state variables that were posted to the server by the frontend and run the calculations to create the recipe and log it. To verify that an order is being sent to the server and a recipe is being returned, we would add 2 meals on the frontend and request a drink. This would then log the recipe from the server and also worked 100% percent of the time out of 10 trials.

Lastly, to verify that our hardware could process the drink request, we had to create a loop that would check the state variables located on the server using GET requests. The hardware would log the recipe that it received from the server so that we could verify that the drink could be processed. To verify this, we added 2 meals and submitted a drink request from the mobile app. From here, we could check both the server and hardware logs to verify that a drink was being processed and the correct recipe was being sent. This also worked 100% of the time out of 10 trials.

2.2 Storage

2.2.1 Design Details

For the storage subsystem, since one of our high-level requirements is to alert the user in the case that a storage compartment's powder volume is under 15%, we needed to try and identify a way of tracking the powder level. This is so that the device actually knows when to send said signal over to the user's end. The design choice we decided to go with in the end was to use three pairs of IR sensors, one pair per storage compartment which each pair consisting of the receiver and the emitter.

Whenever there is any powder in between the receiver and the emitter, the beam produced by the emitter is no longer picked up by the receiver and is thus broken resulting in the sensors being able to identify that there is something in between. In the case that the beam isn't broken, we know that the powder level is lower than that of the IR pair's level. With this, we calculated at what height from the bottom of each compartment the 15% mark would be for volume and put in 3mm holes on opposite sides so that we can mount our IR pairs on them.

2.2.2 Requirements

Our first requirement for the storage subsystem is that the system must be able to recognize whenever a compartment's powder level reaches less than 15% of the compartment's volume. The reason why this is important is to ensure that we have enough powder in that compartment to carry out every single mix request that is made from the user. If we don't have sufficient powder in all of the compartments, then we will not be able to produce the desired mix each time which is a requirement.

Our second requirement is that our system must be able to correctly identify when a container is below the 15% threshold at least 80% of the time. This is just to ensure that there is consistency for our first requirement for the Storage subsystem.

2.2.3 Verification

The way we went about verifying that our storage subsystem requirements were fulfilled was by firstly just checking the signal we were receiving on the serial monitor on the arduino IDE. While the powder level was under the line, meaning the beam between the IR pair wasn't broken, the signal we were receiving was true. We next added enough of a test powder (in this case salt) to ensure that our powder line was above the IR pair line. Once we filled the storage compartment with a sufficient amount of salt, we checked the signal on the serial monitor and the signal read false because the beam was broken due to the salt getting in between the receiver and emitter. We repeated this for the remaining two compartments, both of which worked as well.

For the second requirement we simply did the same as stated in the paragraph above except we repeated that for each compartment 10 times, out of which we were receiving the correct signal 100% of the time for all compartments. The way we were accessing the signal that the IR sensor was sending to us was by using the digitalRead() function after assigning the respective GPIO pin on our chip as an input with the following command pinMode([pinNumber], INPUT) since we use the IRs as an input for to our chip

2.3 Dispenser

2.3.1 Design Details

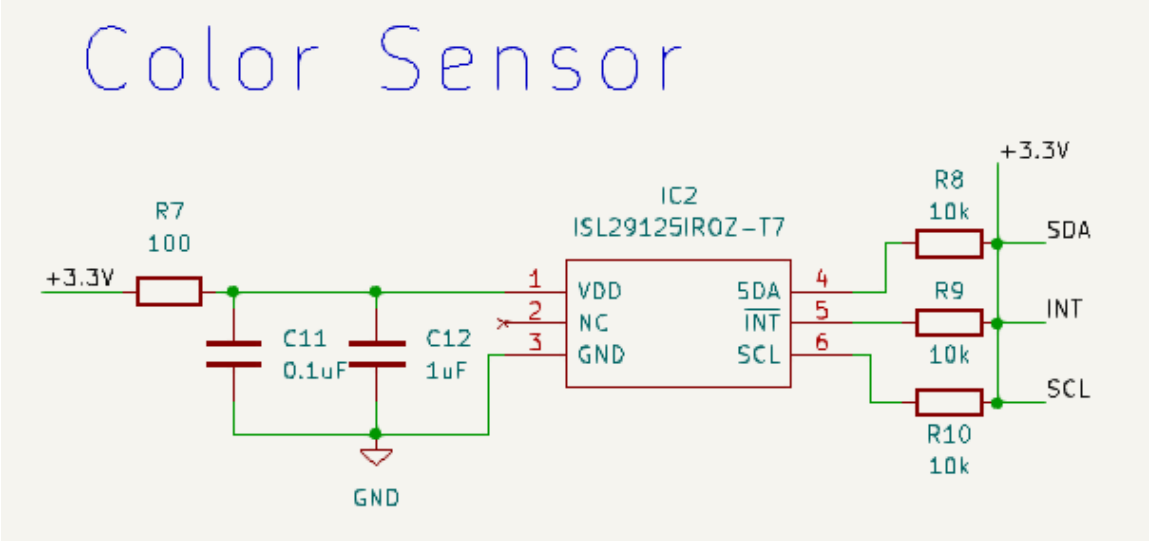

Figure 4. Color Sensor Schematic

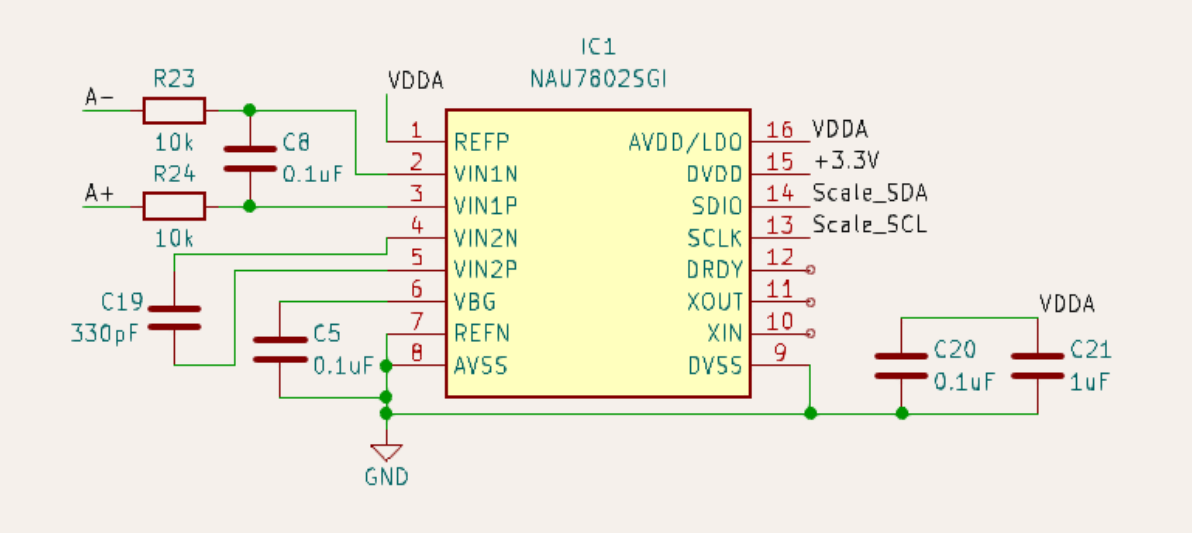

Figure 5. Load Cell ADC Converter Schematic

Our dispensing subsystem originally consisted of five different types of electronic components, the components in particular being three 180° servos, one continuous servo, one load cell, one ADC converter, and one color sensor.

We used one 180° servo per compartment to toggle back and forth between allowing powder to flow and just sealing the storage compartment. We used the continuous servo for our turntable / lazy susan mechanism which allows the storage compartments to rotate along the center axis so that we can change which storage compartment is hovering the cup which the powder would get dispensed into.

The load cell and ADC converter go together since we used the ADC converter to be able to process the data we would be getting from the load cell and make it more quantifiable.The load cell is for being able to track the weight of the cup that would be placed on top and how it would change as more and more powder would be dispensed. By just taking the change in weight between each step we can identify how much of each powder was actually dispensed.

Lastly we have the color sensor which we originally intended to use to track which compartment is aligned with the cup (in order to make sure that the correct compartment will start dispensing) but due to certain circumstances which have been explained in the uncertainties section, as well as it just not really being all that useful or efficient we decided to cut it out from our final design.

For the color sensor and ADC converter schematics shown above we referenced the ISL 29125 Color Sensor datasheet [1] and the NAU7802 ADC converter datasheet [2].

2.3.2 Requirements

Our first requirement for the dispensing subsystem was that the system must be able to dispense each powder within a 15mg tolerance. The reason why this is important is because we want to limit our margin of error as much as possible to prevent any severe case of underdosing and especially overdosing. The last thing we want to do is harm the end user by giving them too much of one supplement.

Our second requirement for the dispensing subsystem is that the desired storage compartment should always be correctly aligned with the cup to ensure we dispensing the correct powder, directly into the cup. This is important because we wouldn't allow the storage compartment to start dispensing when it isn’t directly above the cup as all that would do would spill powder everywhere and make a mess.

2.3.3 Verification

While trying to verify the first requirement, something that we learned really quickly was that the variance in the values of the load sensor would be anywhere from 20mg to 50mg. This was a huge issue because even the floor value of the variance was above the threshold value we were originally aiming for, and thus because of that we were unable to fulfill our first requirement for the dispenser subsystem.

We were more successful in verifying that our second requirement was fulfilled but the method of verification had to be altered as the original one (which can be viewed in the appendix) relied on the color sensor which is a component we had scrapped out of our project. Instead we simply tracked the position of the continuous servo within the hardware program. To verify, we put a cup under and would see after each compartment change whether the powder’s fall trajectory would be into the cup or not (i.e. making sure that the dispense hold is directly above the top hole of the cup).

2.4 Power

2.4.1 Design Details

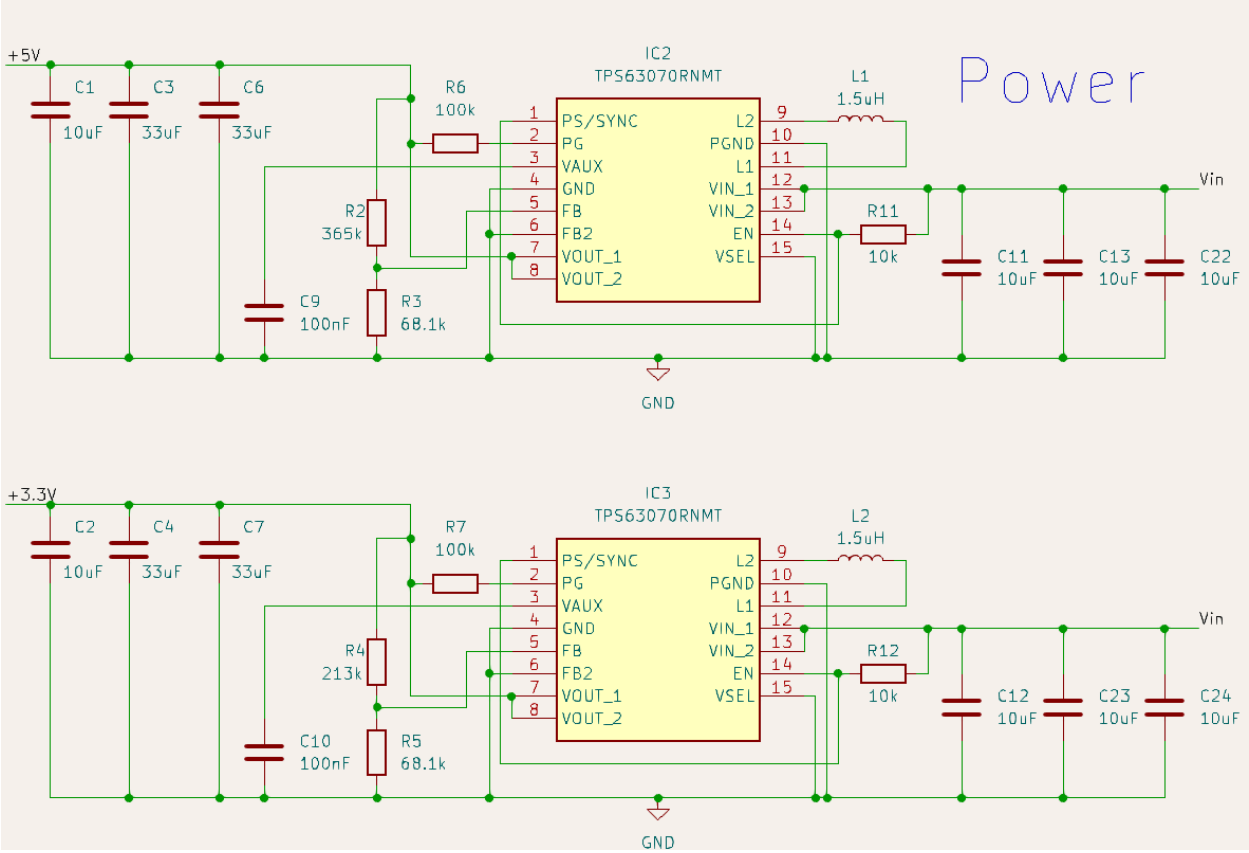

Figure 6. Buck Converter Schematics, 5V (Top) 3V3 (Bottom)

The power subsystem is the backbone of our design. It utilizes a few components, mainly a 12V AC-DC wall wart, as well as two buck controllers that are used to buck down from 12V to the respective voltages required for our design. We initially chose to use LDOs in our design, however once considering power dissipation and overheating, we decided to pivot into using buck converters, as they would be more power efficient and increase the robustness in our design. Below are the calculations of the worst case current draws of each rail.

Part	Quantity	Worst Case Current Draw @ 3.3V	Comments:
IR Sensor	3	10mA	Constant draw
NAU7802	1	2mA	Constant draw
ESP32 Microcontroller	1	355mA	Max
Total		387mA	

Part	Quantity	Worst Case Current Draw @ 5V	Comments:
FeeTech FS5103R	1	850mA/180mA	Stall/Idle
TowerPro SG92R	3	650mA/10mA	Stall/Idle
Total		880mA	Only one will be running at a time, other 3 will be idle

The total current of both add to under 1.5A, and considering overhead as well as unforeseen scenarios where current could possibly spike, we chose to have an input current of 1.5A. The buck converters were chosen off of a few requirements: input voltage range, output voltage range, and maximum output current. The TPS63070 seemed to be a suitable option for our needs, boasting a input and output voltage range that was adequate for our specifications, as well as up to a 2A output current. Thus, this was the design that we decided to move forward with.

2.4.2 Requirements

Our first requirement for this subsystem was that it would be able to provide 3.3V and 5V of voltage with a total ripple of ±5%. For the tolerance range, we chose this as it was a range that was obtainable, and the upper and lower bounds were within the operating voltages that our components required.

Our second requirement for this subsystem was that our 12V input would be able to supply 1.5A of constant current. As shown above, the total amount of current required at a time was less than this figure, so we decided that being able to go above this worst case current draw in case there were unforeseen situations would be the safe route to go.

2.4.3 Verification

As shown below in Figure 7 and 8, we were able to achieve our first requirement. The ripple in each case is less than 1mV, and the value that it hovers around is also well within our ±5% range.

Figure 7. 3V3 Scope Capture

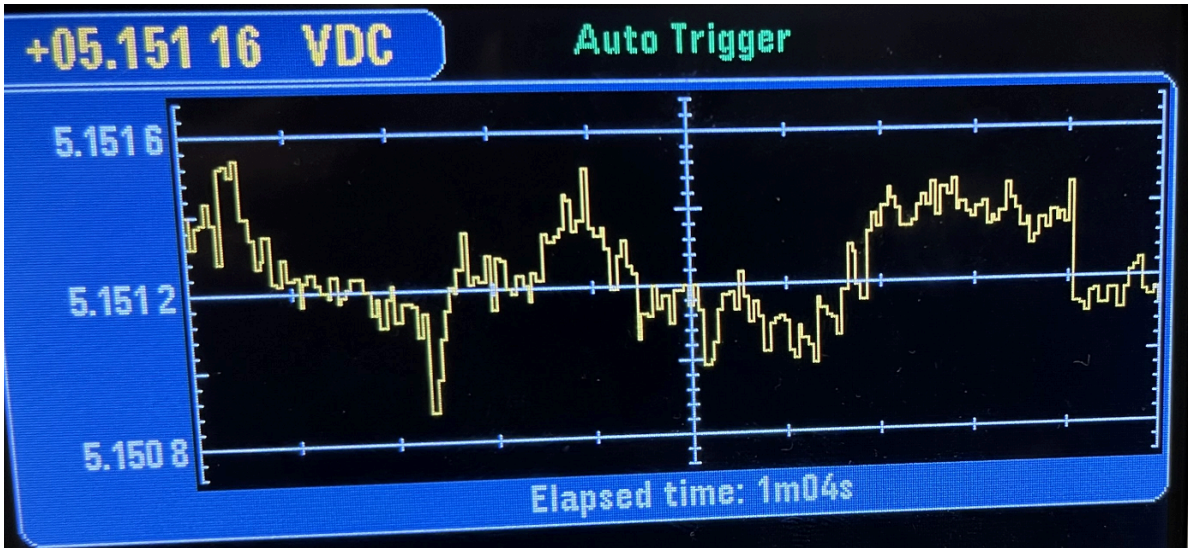

Figure 8. 5V Scope Capture

Our second requirement was also fulfilled. We hooked up a DC load to the output of the 12V Wall Wart and drew 1.5A of constant current from it. We were able to keep this sustained for as long as we were testing it each time, verifying its functionality.

2.5 Physical Design

The biggest design challenge we faced when designing was preventing clogs in the powder, while having it still flow at a slow rate. Our first prototypes focussed on trying to find designs that would prevent clogs. Once that we established that using an hourglass curve on the inside, formed from two separate curves, we moved onto reducing the flow rate. To do this, we adjusted the ratio of the curves of the pseudo-hourglass shape of our dispenser. We were able to optimize this part and later, integrate into the rest of the design. The rest of the design was referenced off a lazy susan design, where continuous servo is used as the axis of rotation, and the three dispensers are attached to the lazy susan table. The simplicity of this design allows us to minimize mechanical error while still being able to have the design fit the specifications. In figure 9, we can see the base of this machine. At the top of the base, the continuous rotation servo is mounted. The rotation part of the servo is then mounted into the middle of figure 10. Each dispenser section has 2 holes for the pair of IR sensors, as well as a platform for the 180 degree rotating servos. In figure 11, a bottom view is shown, and in figure 12, we have the whole model printed out and assembled.

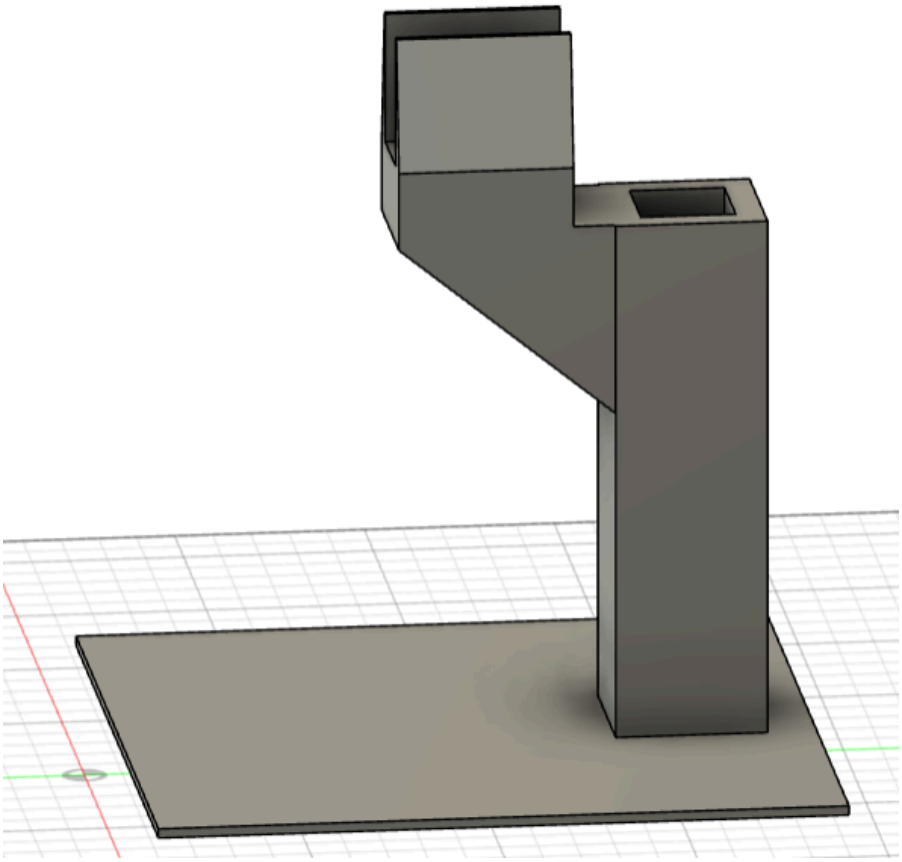

Figure 9. Base of machine

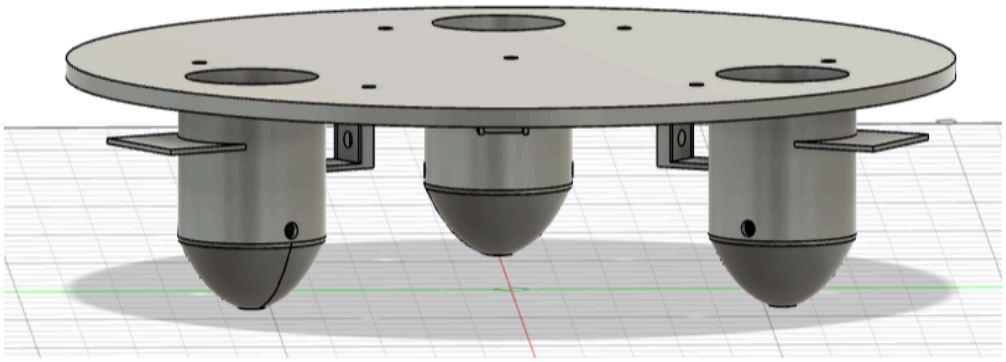

Figure 10. Side view of dispenser

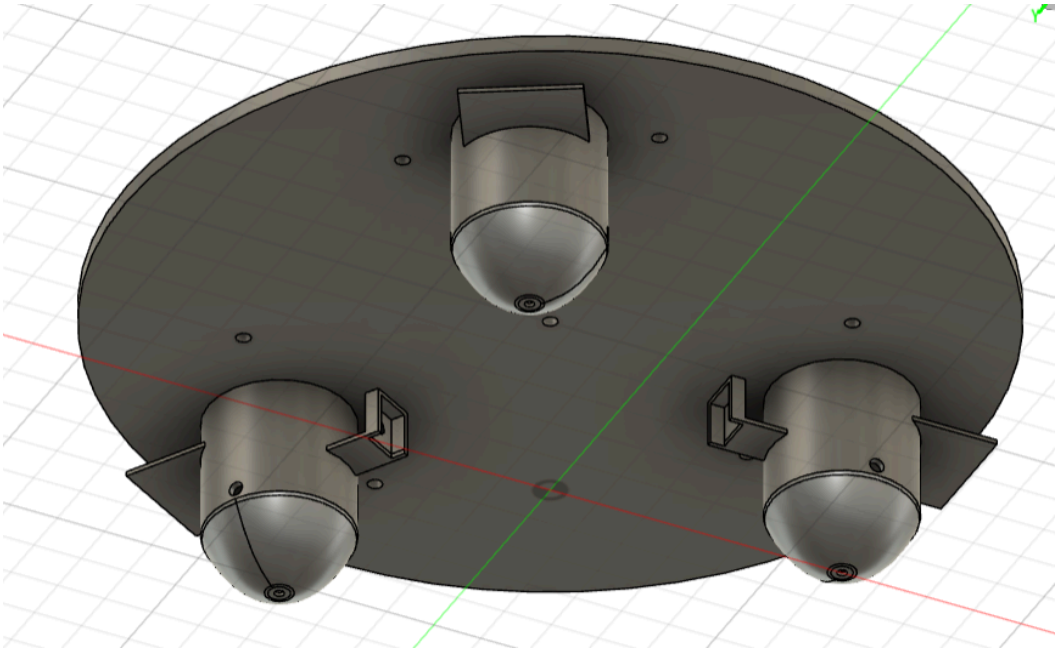

Figure 11. Bottom view of dispenser

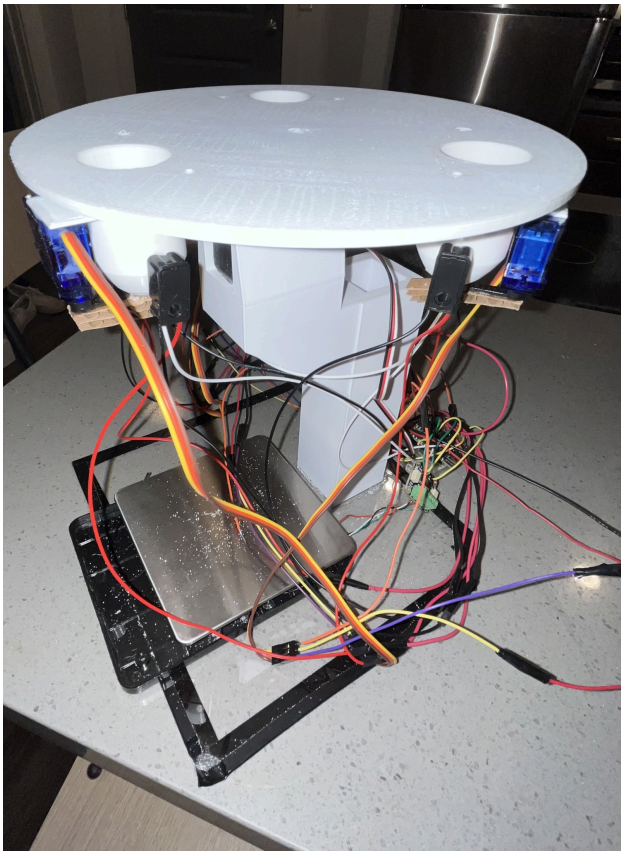

Figure 12. Mix maker completely assembled

3. Cost and Schedule

3.1 Cost

Manufacturer	Part #	Quantity	Cost	Total	Description
ESPRESSIF	ESP32-S3-WROOM	1	$2.95	$2.95	Microcontroller with Wifi
FeeTech	FS5103R	1	$11.95	$11.95	Continuous Rotation Servo
TowerPro	SG92R	3	$5.95	$17.85	Micro Servo
Texas Instruments	TPS63070	1	$2.70	$2.70	Buck-Boost IC
STMicroelectronics	LM-317 TO-220	1	$0.76	$0.76	Linear Regulator
Adafruit	2167	3	$2.95	$8.85	IR Break Beam Sensors
N/A	B0852HX9HV	1	$11.99	$11.99	12V 24W AC-DC
N/A	N/A	1	$15.99	$15.99	Kitchen Scale
Nuvoton	NAU7802	1	$1.73	$1.73	ADC Converter
N/A	N/A	1	$40.00	$40.00	3D Printing
Yageo	RC0201FR-07240RL	1	$0.10	$0.10	240 Ohm
Vishay	D55342E07B720ARWS	1	$0.10	$0.10	720 Ohm
Yageo	RC0603FR-07100RL	1	$0.10	$0.10	100 Ohm
Yageo	RC0402JR-0710KL	4	$0.10	$0.40	10k Ohm
Yageo	RC0402FR-07100KL	1	$0.10	$0.10	100k Ohm
Yageo	RC1206FR-0768K1L	1	$0.10	$0.10	68.1k Ohm
Yageo	RT0603BRD07213KL	1	$0.14	$0.14	213k Ohm
Panasonic	EEE-1EA100SR	5	$0.32	$1.60	10uF, 25V
Panasonic	EEE-HB0J2220AR	3	$0.36	$1.08	22uF, 6.3V
Nichion	UUP1H0R1MCL1GS	2	$0.64	$1.28	100nF, 50V
Meritek	SRB50V1R0MD054	1	$0.04	$0.04	1uF, 50V
Murata Electronics	LQM21NN1R5K10D	1	$0.26	$0.26	1.5uH
N/A	N/A	3	$9,000	$27,000.00	Labor

Parts Cost: $111.85
Total Project Cost: $27,111.85
Calculation for Labor: 2.5 * $50/hr * 72 hours = 9000 per person

3.2 Schedule

Timeframe	Dhruv	Andrew	Horace
Week 1 (1/15)	Brainstorm Ideas	Brainstorm Ideas	Brainstorm Ideas
Week 2 (1/22)	Brainstorm Ideas Revise Idea Proposals	Brainstorm Ideas Revise Idea Proposals	Brainstorm Ideas Revise Idea Proposals
Week 3 (1/29)	Finalized on Project Idea Project Approved	Finalized on Project Idea Project Approved	Finalized on Project Idea Project Approved
Week 4 (2/5)	Start Project Proposal Find parts		Start Project Proposal Find parts
Week 5 (2/12)	Revise Project Proposal Finalize Design Document Brainstorm design Sketch out schematics	Write specs for mobile app Revise Project Proposal Finalize Design Document	Brainstorm design Sketch out schematics Revise Project Proposal Finalize Design Document
Week 6 (2/19)	Order Parts	Wireframe for UI	First draft of schematics
Week 7 (2/26)	Prototype on Breadboard Test Microcontroller Test Sensors	Create Firebase db Compile list of food data Populate db with data	First draft of PCB
Week 8 (3/4)	Start Mixing Device code	Set up IOS app	Revise PCB
Week 9 (3/11)	Test Mixing Device code on Breadboard Debug Mixing Device code	Set up Firebase user auth Set up onboarding ui	Start working on CAD files
Week 10 (3/18)	Continue working on arduino code	Set up all pages and navigation Make recommendation algorithm	Soldering PCBs 3D print CAD files
Week 11 (3/25)	Test wireless interface	UI testing Set up wireless communication with hardware	Test PCB Debugging
Week 12 (4/1)	Debug wireless interface and storage code	Test if IOS app can consistently send accurate recipes to device	PCB version 2 revision
Week 13 (4/8)	Optimize dispenser code and continue debugging	Debug app and integration	Solder PCB Debug PCB
Week 14 (4/15)	Continue Debugging code	Help with integration	Fit electrical components into 3D printed components
Week 15 (4/22)	Project Demos	Project Demos	Project Demos
Week 16 (4/29)	Project Presentation	Project Presentation	Project Presentation

4. Conclusion

4.1 Accomplishments

Out of the three high-level requirements that we had set at the beginning of the semester we were able to successfully accomplish two of them, those two being:

- The mix maker should be able to wirelessly interface with applications on other devices and should receive the correct mix recipe 80% of the time.
- The user will be notified when any storage compartment is below 15% of capacity.

We are able to fulfill both of these high-level requirements 100% of the time based on our testing.

Additionally we were also able to fulfill the majority of our subsystem requirements, with the ones we were not able to accomplish being explained in the uncertainties section.

4.2 Uncertainties

The major uncertainty with our project came to be our first high level. We originally proposed to have a ±15mg tolerance, but once we started testing our entire system as a whole, it became apparent that this was an extremely ambitious number that was not achievable by our system at the budget we were given. The load cell that we used, which already was as accurate as we could obtain within a reasonable budget, varied 30 or more milligrams even when there was no weight on the scale. These inaccuracies contributed to the uncertainty that we came across with this design.

Another uncertainty we came across during implementation was the color sensor. During our testing trials, we would shine one of red, green, or blue light at the color sensor and verify whether or not it was returning a value that was clearly skewed towards the respective one. However, this was not the case, even in a dark environment where there were no sources of light except for the one projected towards the color sensor. Therefore, we deemed the color sensor to be unreliable and found another solution to align our dispenser instead of using the color sensor.

Our PCB implementation was also another one of our uncertainties. The first thing checked on the board was whether or not the buck converters were able to provide adequate voltage. Since it passed this test, we had skipped on and not verified whether or not the board was able to supply enough current for our components. This led to many hours of debugging our components on the board, since they would not turn on. This may have been the root cause of the rest of the issues that we had with the board, as if there was not adequate power, it would make sense why we were not able to detect the microcontroller, or the on-board NAU7802 IC. We eventually figured out that the current was the issue, and used the DC electronic load to find that the buck was only able to supply ~178mA of current.

4.3 Next Steps

For future work regarding this project, we have a few issues to tackle. Delving into the issues with the PCB implementation, the first issue to iron out is in regards to redesigning the power topology from ground up. It was clear from our testing that our buck controllers were not able to supply sufficient current that was required of them from the initial design.

Our dispensing subsystem requires more optimization and debugging, as our servos were not working properly, and we had failed to hit a tolerance that we had initially set out to achieve. We should write our own custom functions to control the servos, as the libraries we used were not allowing the servos to consistently work the way we wanted them to. Additionally, we need to optimize the dispensing design, whether that means going back to the physical design or continuing to iron out bugs in the dispensing code in order to achieve a more realistic 10% tolerance.

If this design were to be upscaled, integrating a color sensor would allow us to make the design more modular. It would be easier that way, if the user wanted to slot a random compartment into a hole in the lazy susan design. This way, we would not need a predetermined location to switch to, instead using a feedback loop with the color sensor.

4.4 Ethics and Safety

Since this project involves administering supplements that when consumed in high dosage would be fatal, we need to ensure that the amount which we are administering on each use will never cause any harm to our end user. Firstly, we simply don't want anyone to get hurt because of what we created, but also as electrical engineers we have the IEEE code of ethics which we have to abide by as well. Causing someone to get hurt by potentially overdosing them would be in direct violation of 7.8.I.1 which states, "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment," [3].

To ensure that we didn't violate the aforementioned code of conduct, we researched various micronutrients' and their daily recommended values as well as their upper limit for consumption. We then made a list of potential micronutrients which our device could potentially administer due to their recommended values being fairly high in comparison to other micronutrients. The list / table which said micronutrients can be seen below, all information and values within the table were taken from Harvard School of Public Health [4].

Vitamin/ Mineral	Womens Recommended Intake	Mens Recommended Intake	Upper Limit
Vitamin C	75 milligrams	90 milligrams	2,000 milligrams
Choline	425 milligrams	550 milligrams	3,500 milligrams
Calcium	1,000 milligrams	1,000 milligrams	2,500 milligrams
Potassium	2,600 milligrams	3,400 milligrams	Unknown
Phosphorus	700 milligrams	700 milligrams	4,000 milligrams

Condensed Version of Table from [4]

As evident, all upper limit values are significantly higher than the recommended intake values thus acting as a secondary safety measure in the case that there is a bit extra amount of powder of a specific nutrient, that it would not affect the user in a negative way.

In the final product, we have accounted for the concern discussed earlier within the ethics section and as mentioned before have come up with countermeasures to ensure that those concerns are resolved. Due to this we feel that our device meets the ethics and safety levels that we would have hoped for it to.

References

[1] Renesas, “ISL2915 Datasheet,” renesas.com, Jan. 2017. https://www.renesas.com/us/en/document/dst/isl29125-datasheet

[2] Nuvoton, “NAU7802 24-Bit Dual-Channel ADC For Bridge Sensors,” nuvoton.com, January. 2012. https://www.nuvoton.com/resource-files/NAU7802%20Data%20Sheet%20V1.7.pdf

[3] IEEE, “IEEE Code of Ethics,” ieee.org, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html

[4] Harvard School of Public Health, “Vitamins and minerals,” hsph.harvard.edu, March. 2023. https://www.hsph.harvard.edu/nutritionsource/vitamins/

Appendix A - Requirement & Verification Tables

Requirement	Verification	Verification Status
Mobile Integration		
When prompted with an input food item, the database must return: 1 JSON File *Micronutrients Present (units in IU and mg) - Vitamin C - Vitamin B3 - Vitamin B6 Macronutrients Present - Protein (grams) - Carbohydrates (grams) - Fat (grams) - Calories (kCals) With a success rate of 90% *Specific micronutrients chosen for MVP for higher daily upper limits	1. Using the mobile app, submit a meal (i.e. pasta, chicken, etc.) 2. Verify that the app returns the macro and micronutrients of the meal 3. Repeat for all foods stored in the database 4. Verify that 90% of the foods requested returned a valid JSON file	Yes
When prompted to make a drink, the system must be able to accurately recommend a list of supplements based on what the user is lacking in their diet with an 80% success rate.	1. Using the mobile app, submit 2 sample meals 2. Request today's drink mix 3. Verify that the app analyzes today's meals and returns the vitamins and minerals that weren't present in today's meals 4. Verify that the app returns a blend that supplement what the user was lacking 5. Repeat 10 times and verify that step 4 held true 8 out of the 10 times.	Yes
When a drink order has been confirmed in the mobile app, the software must be able to communicate with the hardware by sending control signals over wifi with an 80% success rate.	1. Verify that the app is paired with the device 2. Request today's drink mix 3. Verify the doses are accurate and within the safe range 4. Confirm and request the drink	Yes

	5. Verify that the device processes the request and dispenses a drink 6. Repeat 10 times and verify that step 5 held true 8 out of the 10 times.	
Storage		
System must be able to recognize when a compartment has reached 15% of its capacity.	1. Put excess amount of powder (well over 15% of compartment capacity) into the storage compartment 2. Check log for verification that the sensor registers powder as above threshold. 3. Remove powder until it is below 15% of compartment volume. 4. Check log for verification that the sensor doesn't register any powder as above threshold. 5. Repeat for any remaining compartments.	Yes
System must be able to correctly identify when a container is running low 80% of the time.	1. Repeat Requirement 1's verification procedure steps 1-4 10 times. 2. Verify that 8 of the 10 tests were successful. 3. Repeat steps 1-2 for any remaining compartments.	Yes
Dispenser		
System must be able to dispense each powder within a 15 mg tolerance.	1. Send a signal to dispense 30mg of powder from the desired storage compartment. 2. Let the dispensing process reach completion. 3. Check log for weight sensor and note down change in weight from before and after dispensing process. 4. Check the difference from 30mg and the value noted in step 3 and ensure said difference is within our desired 15mg tolerance. 5. Repeat steps 1-4 for remaining storage compartments.	No
System must be able to rotate and align onto	1. Send a signal to dispense powder from	Yes

the correct storage compartment.	the desired storage compartment. 2. Let the machine rotate until reaching full stop. 3. Check log for color sensor and make sure it is within 10% of RGB value for the LED's intended RGB value. 4. Repeat steps 1-3 for all containers with different start positions for turn mechanism and different external lighting factors.	(Difference verification method after scrapping color sensor)
Power		
System must be able to supply 5VDC +/- 5% and 3.3VDC +/- 5%	1. Hook up power output of buck and LDO to digital multimeter and test voltage ratings. 2. Verify that from the buck converter output the voltage is between the range of 3.465~3.135 volts using an oscilloscope. 3. Verify that the linear regulator output voltage is between the range of 5.25~4.75 volts using an oscilloscope.	Yes
System must be able to provide up to 1.5A of current while maintaining 12V	1. Place a 8 ohm resistance across the 12V rail and ground 2. Measure the voltage across the resistor	Yes (Different verification, used DC load instead)