Team Contract
University of Utah, Bioengineering
Project 19
Spring/Fall 2024, BME 3801/4801, Thursday
Revision 03 12/03/2024

I. Communication

a. Contact Information

Contact information is provided as a means of coordinating and facilitating the design process facilitated by this course. Information provided should only be used for purposes associated with participation in the University of Utah's bioDesign course. Use outside this scope is prohibited unless authorized by the owner of the information.

Name	Phone Number	Gmail
Jude	208-830-2645	U1373453@utah.edu
Wyatt	907-854-0464	young.wyatt@utah.edu
Kyle	385-449-8570	Kharshny@gmail.com
Daniel	210-842-1927	U1326261@utah.edu

b. Communication

1. Email
    a. Emails are to be checked daily at a minimum by 10 pm.
    b. Response by email should take place within 24 hours
    c. Weekends and holidays are exempt from
2. Texts
    a. Text messages are to be checked regularly
    b. Response by phone or text is required within 12 hrs.
    c. Texts received after 11 pm are exempt, but should be responded to ASAP
3. Phone
    a. Phone calls should be limited to the hours of 8 am – 10 pm
    b. Response by phone or text is required within 12 hrs.

II. Responsibilities a. Scheduled Meetings

Lab Section

a) Required meetings will be held during lab on Thursdays from 2 to 5pm

b) Assignments from prior meetings and new assignments will be given

c) Assignment questions and concerns should be addressed during lab

d) Allowed to miss meetings with 48 hours' notice unless it is an emergency

Additional Meetings

a) Upon request and arrangement, additional meetings may be held

b) Such meetings will be limited to necessary team members

b. Roles and responsibilities

All responsibilities listed are intended to illustrate the general category of work for each individual on the team. Specific assignments dictated in the most current revision of the DDP supersede any conflict with general responsibilities listed. Team roles can be changed at any time with both parties agreeing.

1. Team Leader/App Design: Jude Werth
    a. Weekly Progress Reports
    b. Lead Meeting Discussions
    c. Head DDP Development and Execution
2. Mechanical/Electrical Design: Daniel Candland
    a. Manage Digital Documents (Ubox)
    b. Head up Design History File (DHF) and Document Control
3. Mechanical Design: Kyle Harshany
    a. Lead in Prototyping and Generation of Design Specifications
    b. Assist in Market Research
4. Electrical Design/Integration: Wyatt Young
    a. Lead in compilation of Marketing Requirements Documentation
    b. In charge of maintaining meeting minutes
5. All Team Members
    a. Responsible for providing prompt and meaningful feedback
    b. Participating actively in team meetings
    c. Proactively completing assigned tasks
    d. Being familiar with topics related to device being designed
    e. Maintaining compliance to SOPs and relevant FDA laws and regulations
    f. Communicating promptly and professionally with teammates
    g. Communicating if help is needed with adequate time for team members to provide assistance.
    h. Hold your position for one month then rotate. Quality to Leader, Design to Quality, etc.

c. Paperwork

1. Weekly Progress Reports
    a. Prior to lab meetings, task progress will be reported to the team leader

b. Team leader will compile a progress report 48 hours prior to submission
c. Team will be allowed 24 hours to provide feedback on the report
d. Meetings will begin with a review of the prior week's minutes from which an accounting of assignments and progress will be made

2. Documents and Assignments
   a. Assigned documents will be completed prior to meetings for team review
   b. All team due dates will be at least one day prior to class due dates
   c. All team input will be required during the subsequent 24 hour period
3. Digital File Management
   a. All files will be stored, shared, and accessed using Ubox
   b. The Quality Engineer will manage this database

III. Decision Making Process

1) Methods
   a. Decisions will be made based upon majority rule
   b. In a split decision, the following may be used to attain majority decision
      i. Opinions from third party source (Instructors)
2) Conflict
   a. Yelling, rudeness, and physical violence will not be tolerated
   b. It is the responsibility of the team to resolve any conflict that arises
   c. Two representatives outside the team may be selected to resolve full
   d. Uninvolved team members will serve as mediators in other conflicts, but
   e. only in the case in which neutrality on the issue can be verified by all parties f. involved
   g. Personal conflicts should be addressed in a one-on-one setting
   h. Help in addressing unresolved conflict will be sought from the following
   i. Course Instructors
      ii. Micah Frerck
      iii. Other students not affiliated with the conflict in any form

IV. Consequences

1) Tardiness
   a. There will be no consequence for arriving to team meetings up to 10 mins late
   b. Excessive tardiness (3x 15min+) will result in point loss decided by team
   c. Anticipated tardiness should always be promptly communicated
2) Absences
   a. Excessive absences (2x +) will result in point loss decided by team
      i. One unexcused absence is permitted but must be communicated
      ii. Additional excused absences may be granted by the team
      iii. Any assigned tasks must be completed ahead of time or assigned to another teammate prior to absence
   b. Anticipated absences should always be promptly communicated

3) Point Deduction
    a. Points will be initially distributed equally among all team members
    b. Points will be deducted from individual at fault in the following instances
        i. Absence or Tardiness
            1. According to conditions stated above ii. Poor Quality of Work
            1. Repeated submission of poor work, characterized as work which results in point loss on a team assignment or production of work requiring intensive editing from other members of the team prior to submission
            2. Claims must be substantiated by course instructor iii. Incomplete Assignments
            1. Failure to complete assigned task which results in a point reduction on a team assignment or assessment
        iv. Late Assignments
            1. Late submission of assignments which results in a point reduction on a team assignment or assessment
        v. Other
            1. Unanticipated action significantly impacting team grade 2. Only upon consultation and agreement of course instructors
    c. Value of points deducted from team member at fault will depend on grade impact to the rest of the team
    d. The degree of deduction will be determined by the rest of the team Signatures

-

Bioengineering 3801/4801 Biomedical Engineering Design
Executive Summary
Project: Pharmacokinetic control of opioid dosing with feedback
Thursday PM: November 7, 2024 (Update)

Poor regulation and guidance of opioid intake allows the opportunity to abuse prescription pain relievers. Universal dosage recommendations are insufficient to support the healthy use of prescription pain relievers. Tolerance can vary between individuals, and pain can't be accurately quantified. This leads to overuse or abuse of prescription pain relievers. Opioids can have side effects like drowsiness, mental fog, nausea, and constipation. Abuse of this medication can change breathing patterns and heart rate and, in the worst cases, cause death. These risks have caused a negative stigma and fear in prescribed medication. This gap in understanding of prescribed opioids between patients and medical professionals has negatively impacted health care.

This project aims to provide personalized dosage recommendations to reduce opioid misuse and increase patient confidence. Considerations include competitive products, environment, user needs, intended uses, and functional requirements. The target users are patients with chronic pain who have opioid prescriptions.

The team comprises four members: two design engineers, Daniel Candland and Kyle Harshany, responsible for developing and testing the physical device, and two coding engineers, Jude Werth and Wyatt Young, focused on the software interface. Each member's responsibilities and expectations are detailed in a signed team contract to ensure effective collaboration. The timeline is set from BIOEN 3801 to BIOEN 4801 course.

After establishing roles, a design and development plan (DDP) was developed according to the FDA and course guidelines, covering scope, resources, design requirements, goals, and timelines. Additionally, project goals and desired completion dates were outlined. Certain sections requiring a specific design concept were left blank.

After outlining an agreed-upon project plan, background research was conducted. During this period, relevant papers were analyzed to understand better the problem and user needs, potential solutions and design ideas, and current products available. During this research period, helpful background information was shared. Dr. Kai Kuck, the clinician associated with this project, was contacted to schedule a meeting with the team. In preparation for this meeting, questions were prepared to clarify the parameters of the project.

On February 12, 2024, Dr. Kuck provided valuable input on the project's scope, problem clarification, and product comparisons, informing the completion of the DDP. Subsequently, a team meeting with Dr. Kelly Broadhead led to adjustments to the DDP, finalizing a concept for a pill container to monitor and regulate pill intake.

The design requirements were then refined, focusing on preventing opioid misuse and tailoring dosage. Marketing requirements were developed in alignment with FDA and ISO regulations. Public health and global impacts were considered, and competing products and patents were analyzed. Three design concepts were created, and design review slides were prepared for a presentation on February 29th, after which feedback from Robert Hitchcock and Kelly Broadhead was incorporated into revised documents.

The team proceeded with design requirements based on the DDP, splitting tasks according to the DDP. Daniel and Kyle managed mechanical and material sections, while Jude and Wyatt handled software and electrical sections. Under Micah Frerck's guidance, compatible electrical components were selected, and a prototype was designed in Fusion 360. Risk analysis was conducted using FTA trees and an FMEA table, and a finalized design prototype integrated all mechanical, material, electrical, and software considerations.

Once risk analysis and design specifications were complete, the prototype build began. Kyle and Daniel ordered materials and 3D-printed the initial bottle design, addressing dish design issues through trial and error. Jude and Wyatt ordered circuit components, including a Wi-Fi microcontroller, solenoid, and load cell. The project was then split into four areas: screwing mechanism, bottle/circuit container, electrical circuit, and app interface. Each team member took on a section: Daniel fine-tuned the screwing mechanism, Kyle optimized the container design, Wyatt assembled and coded the circuit, and Jude developed the app interface to connect with AdafruitIO cloud storage.

Bioengineering 3801/4801 Biomedical Engineering Design
Executive Summary
Project: Pharmacokinetic control of opioid dosing with feedback
Thursday PM: November 7, 2024 (Update)

Following the completion of risk analysis and design specifications, the group determined a plan for verification testing. Our verification strategy was formulated around the idea of making a product that can serve its functionality in various situations. Each component underwent basic pass/fail tests to confirm its ability to meet functional requirements. Following this, we tested each component's ability to interact effectively with others to create an integrated, functional prototype. We established specific statistical parameters for these integration tests with a narrow, predefined range of acceptable outcomes.

The next phase of our project will be design verification. Each part of the device—mechanical, electrical, and software—will be tested individually to ensure it meets functional requirements, followed by integrating these components to verify they work together as a cohesive system. Any issues identified during testing will lead to refinement, where adjustments will be made to optimize performance and ensure the system meets the project's goals. This phase will ensure that all components function properly both individually and together.

Following design verification, we will build a final prototype of the device as proof of concept to present our results. Daniel is designing the physical components and mechanisms, specifically the bottle, load cell platform, and electrical compartment. Wyatt is constructing the circuit board and calibrating the electrical components. Jude is finshing the app UI design, and Kyle is conducting verification testing. The final phase is to combine each of these components and create a cohesive and functional product.

Once we have a final prototype, we will look forward to deploying it and find a way to improve the opioid dosage situation. This begins with university sponsored events such as bench-to-bedside. We will continue a cycle of presenting our work, receiving feedback, and updating the device until we have a product and audience to push into the marketplace.

Biomedical Engineering 3801/4801 Biomedical Engineering Design
Design and Development Plan

Pharmacokinetic Control of Opioid Dosing with Feedback
Revision: 7
12/03/24

Name	Position	Signature	Date
Jude Werth	Team leader, Code and Software	Jude	01/18/24
Kyle Harshany	Design Engineer, Mechanical and CAD Modeling	Kyle	01/18/24
Wyatt Young	Code and Software, Market Researcher	Wyatt	01/18/24
Daniel Candland	Quality engineer, Electronics and PCB	Daniel	01/18/24
Grace Ojewia	Teaching assistant	Grace Ojewia	01/18/24
Tomasz Petelenz	Instructor		01/18/24
Parker Mason	Teaching Assistant		08/22/24

Revision Log			
Revision	Description	Initials	Date
1	Initial release	WY	01/18/24
2	Assigned design specs.	WY	02/23/24
3	Added initials, completed dates to DR table	WY	03/21/24
4	Updated for design specs	WY	04/13/24
5	Updated to add prototype elements, and assignments for verification plan	WY	08/22/24
6	Updated for test plan submission	WY	10/05/24
7	Updated Initials for verification testing completion	WY	12/3/24

1 Introduction

The purpose of this document is to outline the Design and Development Plan for development of a device for the pharmacokinetic control of opioid dosage with patient feedback in compliance with requirements set forth in the bioDesign course in the University of Utah Department of Biomedical Engineering.

2 Scope

This document applies to the creation of a device for the pharmacokinetic control of opioid dosage with patient feedback by a group in the Thursday lab section of the Spring semester and Fall semester of 2024 of the bioDesign 3801/4801 course in the University of Utah Department of Biomedical Engineering.

3 References

3.1 External References

- 21 CFR 820.30. FDA Quality System Regulation Subpart C – Design Controls. Effective Date – 6/6/23
- Design Control Guidance for Medical Device Manufacturers. FDA – CDRH Effective Date - 3/11/97

3.2 Internal References

- Design Control – DC 04 01 – Design Control Policy SOP

Effective Date – 1/18/24

- Design Control – DC 04 02 – Design and Development Planning
  Effective Date – 1/18/24
- Design Control – DC 04 04 – Design Requirements
  Effective Date – 1/18/2024

4 Team Members and Resources

Jude Werth: Team Leader, Coding Specialist

- Awarded UROP (Spring 2024)
- UCUR Speaker (Spring 2024)
- Developed Neural Data Analysis Pipeline in MATLAB
- Contributed on Atrial Segmentation project to quantify Fibrosis
- Coding Engineer for Datajoint

Kyle Harshany: Design Engineer

- Developed Monte Carlo simulations, neuroimaging and geodesic shooting algorithms through MATLAB
- Teaching assistant for 2 semesters, prioritized one-on-one interaction.
- Awarded UROP Award 2x

Wyatt Young: Code Specialist

- Developed a Data Structuring Pipeline in Python for patient gait data sets obtained from the Orthopedic Rehabilitation Training System (ORToS).
- Served as point of contact for clinicians during ORToS clinical study.
- Presented research on novel gait signal batch-processing Python code at NCUR 2024 and at UROP symposium.

Daniel Candland: Design Engineer

- Chemistry Minor
- Lab Associate for Dr Yu in department of BME/Pharmaceutics, Synthesis of and applications for Collagen Hybridizing Peptides in detection of lung cancer metastases. (Solid-phase peptide synthesis, HPLC, Immunohistochemistry)
- Soldering experience
- Medical Research Representative at ICON PLC Early Development Services, training in GCP, FDA and ICH requirements for investigative drugs and medical devices in phase I clinical trials.

Other Resources:

- BioDesign 3801 Instructors
  - Kelly Broadhead
  - Thomasz Petelenz
  - Robert Hitchcock
  - Alphonsus Ng
- Google Search
- Clinicians:

    - Kai Kuck- Dept. of Anesthesiology
      kai.kuck@hsc.utah.edu
- BioDesign 3801 TAs
    - Grace Ojewia
- Lab Engineer
    - Micah Frerck
- BioDesign 4801 TAs
    - Parker Mason

5 Overall project organization

The project will be organized into 2 segments, design input, which includes tasks in this document, as well as planning the requirements and specifications of the device, and design output, which involves the creation of a prototype, and testing to verify the design requirements and user needs are met.

	Project Milestones	Design Review Date
Design Input	**DDP**	1/25
	Design Requirements	2/1
	Design Specifications	4/18
Design Output	**Prototyping**	10/22
	Design Verification Testing	12/3

6 Project details

6.1 Risk (Hazard) Analysis

Risks will be explored and documented before the first design review, after which risk analysis will be reassessed weekly throughout the design process, with updates being tracked. Communicating on how to eliminate risks to team members will also take place throughout the design process.

6.2 Design Input

6.2.1 Consulting Sources

- End Users
    - Clinicians
    - Nurses
    - Pharmacists
    - Patients
- Literature
    - Reviews, Articles
- Clinician Consultant

6.2.2 Design Requirements

The following design requirements need to be fully explored and recorded by one week before the Design Review. For tasks requiring input or assistance from more than one member, a lead for that task will be assigned (first name in 'assigned to' column is lead)

Task	Assigned To	Date Due	Completed: '(y/n)-Initial'

Problem Statement - Clear description of problem to be solved	Jude	1/26/24	Y-JW
User Needs - Provide supporting user need statements with appropriate metrics	Wyatt	1/26/24	Y-WY
Product Description - General description of product and its use	Kyle	1/29/24	Y-KH
Indications for Use - Conditions the device is designed to treat	Daniel	1/29/24	Y-DC
Intended Uses - List of uses the device is designed for	Jude	1/29/24	Y-JW
Intended Users - General description of stake holders (clinicians, patients, investors)	Wyatt	1/29/24	Y-WY
Intended Environment of Use - Environment(s) in which device will be used	Kyle	1/29/24	Y-KH
Address Considerations of Public Health	Jude	2/2/24	Y-JW
- Public Health		2/2/24	Y-JW
- Safety		2/2/24	Y-JW
- Welfare		2/2/24	Y-JW
Address Global, Cultural, Social, and Environmental Factors	Daniel	2/2/24	Y-DC
- Global		2/2/24	Y-DC
- Cultural		2/2/24	Y-DC
- Social implications		2/2/24	Y-DC
- Environmental		2/2/24	Y-DC
Economic Factors	Wyatt, All	2/9/24	Y-WY
- Market Analysis o Market size o Market segments o Estimate of market capture		2/5/24	Y-WY
- Competitive Product Analysis o Descriptions and images of competitive products o FDA Medical Device Database		2/7/24	Y-WY
- Intellectual Property Review o Include patent classes and key patents related to competitive products o Identify other relevant patents and patent applications		2/8/24	Y-WY
- Sales Price Estimates o Estimates based on competitive products		2/9/24	Y-WY
Regulatory Requirements	Daniel, All	2/12/24	Y-DC
- Regulatory Requirements and Approach o Regulatory classification o Statutory requirements that must be followed in the development of the device o Pathway to legally market (FDA) and include device classification		2/9/24	Y-DC
- Applicable Standards o Applicable standards that facilitate use, compatibility, and compliance. List any standards that are appropriate for the product. A typical starting point may include ISO 13485		2/12/24	Y-DC

(Quality Management System), 14971 (Risk), 60601 (Electrical Safety), and 10993 (Biological Evaluation), 11607 (Packaging).			
Functional Requirements	Kyle, All	2/12/24	Y-KH
- Requirements from the user's perspective		2/9/24	Y-KH
- Broad terms that address and describe how the device will meet the user's needs and expectation		2/9/24	Y-KH
- Clinical Functional Requirements (what it does) to meet user needs		2/12/24	Y-KH
- Operational Capabilities and Expectations (how it does it) to meet the user needs		2/12/24	Y-KH
Performance and Physical Requirements	Jude, All	2/14/24	Y-JW
- Performance Requirements (speed, strength, response times, accuracy, limits, etc.)		2/12/24	Y-JW
- Physical Embodiment (size, weight, materials)		2/13/24	Y-JW
- Describe the environmental conditions in engineering terms that the device will be used in (temp, humidity, shock, etc)		2/12/24	Y-JW
- Device compatibility requirements to meet environmental conditions		2/14/24	Y-JW
- Safety requirements o Patient and user		2/13/24	Y-JW
- Environmental and Disposal requirements		2/13/24	Y-JW
- Reliability - Stability - Shelf life - Sterilization		2/14/24	Y-JW
Human System Interfacing Requirements	Wyatt, All	2/16/24	Y-WY
- Human Factor Considerations (Ergonomics) List ergonomic considerations for this product. Consider how the device can be used, misused and confused by the user.		2/14/24	Y-WY
- User and/or Patient Interface What is/are the interface(s) (e.g. controls, displays, handles, grips) that interface the user and / or patient to this product.		2/15/24	Y-WY
- Interfacing Devices and Accessories with Accompanying Interface Requirements - List devices and accessories that will be used with this product. List all requirements that pertain to interfacing with these devices and accessories		2/15/24	Y-WY
- Packaging and Labeling Requirements, Provide a description of the detailed requirements for labeling and packaging. Include unit, dispenser and shipping package requirements as well as special instructions for use. Include all relevant standards that apply to packaging and labeling (including instructions for use)		2/16/24	Y-KH
Conceptual Designs - Include at least three examples of preliminary product concepts.	Jude	2/19/24	Y-JW

6.2.3 Specifications

For tasks requiring input or assistance from more than one member, a lead for that task will be assigned (first name in 'assigned to' column is lead)

Task	Assigned To	Date Due	Completed

Description of Design Solution and how it meets Design Requirements			
Description of Design Solution and how it meets Design Requirements: - Describe what it does and how it works to solve the design problem - Addresses user needs - Indications for use - Intended uses - Intended users - Use written, drawings, or renderings to explain	Jude, All	3/7//24	Y-JW
Summarize functional and performance specifications - Use a table to summarize metrics - Describe how the design is expected to meet performance requirements	Kyle, All	3/7/24	Y-KH
Provide a description of how the design solution addresses considerations of public health, safety, and welfare factors(consider entire life cycle)	Daniel, All	3/7/24	Y-DC
Provide a description of how the design solution addresses global, cultural, social, and environmental factors that impact the design requirements	Wyatt, All	3/7/24	Y-WY
Provide a description of how the design solution addresses economic factors - Include market size implications - Competitive products and IP concerns - Sale Price Estimates	Wyatt, All	3/7/24	Y-WY
Provide a description of how the design addresses regulatory requirements and standards	Jude, All	3/7/24	Y-JW
Provide a description of how the design addresses reliability and stability requirements	Daniel, All	3/7/24	Y-DC
Provide a description of how the design addresses regulatory requirements and standards	Kyle, All	3/7/24	Y-KH
Provide a description of how the design addresses reliability and stability requirements	Jude, All	3/7/24	Y-JW
Human Factors and Usability Engineering			
User and/or patient interface - List the interfaces (e.g. controls, displays, handles, grips) that exist between the user and / or patient and this product. - List design solutions that address these user/patient interfaces. - Identify safety considerations related to these user/patient interfaces.	Wyatt, All	3/14/24	Y-WY
External interfacing devices, accessories, and materials	Daniel, All	3/14/24	Y-DC

- List external interfacing devices, accessories, and materials that will be used with this product. - List design solutions that address these external interfaces. - Identify safety considerations related to these external interfaces.			
Human factor considerations (Ergonomics) - List how the device can be used, misused and confused by the user. - List ergonomic solutions for this product. - Identify safety considerations related to device ergonomics.	Kyle, All	3/14/24	Y-KH
Device use in the intended environment - Reference environmental requirements that need to be addressed. - List design solutions that address these requirements - Identify safety considerations related to operation in the intended environment	Jude, All	3/14/24	Y-JW
Risk (Hazard) Analysis Summary (Include key risks from FTA and FMEA)			
Analyze and evaluate findings from FTA and FMEA and provide summary - Include four FTA charts, and an FMEA Table in Appendix of thisdocument	Wyatt, All	3/14/24	Y-WY
Provide mitigation strategies to identified risks - Focus on risks with high-risk priority numbers	Daniel, All	3/14/24	Y-DC
Summarize Residual Risk - Describe how risk analysis addresses user and patient safety requirements. - Provide benefit-risk analysis (see ISO 14971:2019 section 7.4) - Evaluate overall residual risk (see ISO 14971:2019 section 8)	Jude, All	3/14/24	Y-JW
Mechanical Design Details			

Engineering drawings shall be provided for the proposed design from which a machinist could create the design.
- Use standard ANSI size A (8.5" x 11") or ISO size A4 (210 mm x 297 mm) in portrait orientation (same orientation as spec. doc.).
- Provide a set of orthographic projections of each part
    o Include appropriate dimensions with tolerances on the drawing
    o Fill in drawing block with necessary information
	Kyle, Daniel	3/28/24	Y-DC

- Include a rendered isometric projection
- Provide an assembly drawing
    o Reference each element to its appropriate drawing number as appropriate
    o Include an appropriate Bill of Materials
    o Fill in drawing block with necessary information
- Provide engineering drawings of the device in the body of the document (do not attach to the end of the document).
- Make sure all drawings are readable when zoomed in (readable text).
- Use one of the following programs to generate drawings: Autocad, Solidworks, Creo, or other engineering drawing package that can generate 3d solid models			

Material Selection Details			
Provide List of Materials and Sources that were used in mechanical drawings' BOMs and include: - Mechanical suitability (mw, modulus of elasticity, yield strength, etc.) - Material interface concerns - Material choices with regard to biocompatibility - Toxicity/ biocompatibility profile based on ISO 10993 - Other relevant material properties (e.g. dielectric strength, biodegradation, stability, chemical compatibility, etc.) - How environmental and disposal requirements will be met	Kyle, Daniel	3/28/24	Y-DC
Electrical Design Details			
- Electrical block diagrams - Electrical wiring diagrams - Electrical circuit schematic - PCB layouts and mechanical drawings (as appropriate) - Provide Component List - Electrical power analysis - Electrical Safety Considerations - Electrical Leakage Current Protections - Electromagnetic compatibility/interference (emc/emi)	Wyatt, All	3/28/24	Y-WY
Software Design Details			
- High-level software architecture (use flowcharts, algorithms, or storyboards to show structure)	Jude, Wyatt	3/28/24	Y-WY

- Address software development methods (e.g. Programming languages, hardware environment, operating system, etc.) - Other considerations including key variables (and type), memory usage, cycle times, sampling rates, etc.			
Cleaning, Disinfection, and/or Sterilization Methods			
- Pyrogenicity requirements and how to achieve requirement - Sterilization (SAL, method/standards, validation, packaging and material compatibility) - Chemical compatibility (cleaning agents, sterilization, lubricants, disinfectants, etc.)	Kyle	4/4/24	Y-KH
Shipping/Storage Conditions			
- Environmental and physical conditions during shipping - Environmental and physical conditions for storage - Shelf life (time from manufacture to use) - How will the design meet these conditions?	Wyatt	4/4/24	Y-WY
Packaging and Labeling (P&L)			
- Provide packaging design (include engineering drawings) - Product label, package label, carton label (include artwork) - Instruction for Use (IFU) (include draft document) - Disposal / recycling instructions - Other P&L considerations (e.g. web site, advertising, etc.)	Jude	4/4/24	Y-JW
Using a Traceability Matrix, demonstrate how the device meets Design Requirements			
- Include all design requirements and enter as rows of the traceability matrix (include outline number) - Include all design specifications and enter as columns of the traceability matrix (include outline number) - For each design requirement, mark with an X every corresponding design specification used to meet that requirement - Evaluate how complete the matrix is (especially comment on requirements that are not being fulfilled)	Jude, All	4/4/24	Y-JW

Conceptual Prototypes			
Documentation of product concepts must be included in the specifications document. Although they may not represent the final design, these concepts provide a starting point and can promote discussion and understanding of design elements. Examples must include (as appropriate to your design): - Physical, hand-on models demonstrating tha human factors and usability engineering requirements will be met (prototyping tools like clay, wax, and 3D printing). - Functional representation (demonstration of feasibility) showing core function behind technology (electronic circuits, flow circuits, etc.) - Set of computer interface screens demonstrating user interface (use Matlab or Labview or other) - Design Analysis (mechanical, electrical, materials, etc.) that demonstrates the proposed solution is capable of achieving intended performance requirements ○ Demonstrate that design solution is feasible (use conceptual prototypes and the design analysis) ○ Especially examine the conceptual prototypes for ability to meet requirements (i.e. include conclusions of what you learned from the prototypes) ○ Possible analysis tools include Comsol, Solidworks, Ansys, LTSpice (free download), Matlab, Labview, Excel, etc. (see Engmann lab for additional resources)	All	4/11/24	Y-WY

6.3 Design Output

6.3.1 Prototypes

Task	Assigned To	Date Due	Completed
Scope, Introduction, References	All	Oct 18	10/18/24
Prototype App	Jude, Wyatt	Oct 18	10/18/24-DC/WY
- Element Description ○ Prototype Design ○ Human Interface function ○ Functional performance	Wyatt	Oct 18	10/18/24-WY
- Step-by-Step Building instructions ○ Sketches ○ Flow Charts for GUI	Jude	Oct 18	10/18/24-JW

Lid and Locking Mechanism	Daniel, Kyle	Oct 18	10/18/24-DC
- Element Description o Prototype Design o Human Interface function o Functional performance	Daniel	Oct 18	10/18/24-DC
- Step-by-step building instructions o Bill of Materials o Engineering Drawings	Kyle	Oct 18	10/18/24-KH
Electrical Control Circuit	Jude, Wyatt	Oct 18	10/18/24
- Element Description o Prototype Design o Functional performance	Jude	Oct 18	10/18/24-JW
- Step-by-step building instructions o Bill of Materials o Circuit diagram	Wyatt	Oct 18	10/18/24-WY
Pill Container and Housing	Daniel, Kyle	Oct 18	10/18/24
- Element Description o Prototype Design o Functional performance	Kyle	Oct 18	10/18/24-KH
- Step-by-step building instructions o Bill of Materials o Engineering Diagram o Packaging and Labeling	Daniel	Oct 18	10/18/24-DC

6.3.2 Verification testing

Task	**Assigned To**	**Date Due**	**Completed**
Introduction, Scope, References	Jude	Oct 4	**9/26/24-DC**
Safety Considerations	Daniel	Oct 4	**9/26/24-DC**
Verification Testing Activities:	All	Oct 4	
Updated FMEA Table	Wyatt	Oct 4	**9/19/24-WY**
References to Specifications and Requirements	All	Oct 4	10/4/24-JW
Testing Criteria:	All	Oct 4	
- Pill Container can hold 32 pills for extended period of time without issue	Kyle	Oct 4	10/4/24-KH
- Pill Container can withstand everyday stress - Drop test from 1 meter	Daniel	Oct 4	No-WY
- Upside-down shake test to ensure lid stays attached	Kyle	Oct 4	10/4/24-KH
- Test load cell output with 1, 5, and 32 pills	Jude	Oct 4	10/4/24-WY
- Test locking mechanism by attempting to open the bottle when closed	Daniel	Oct 4	10/4/24-DC
- Test proper unlocking, able to open easily when unlocked	Kyle	Oct 4	10/4/24-KH
- While the container is locked, emergency button unlocks lid, and the bottle is able to open	Wyatt	Oct 4	10/4/24-WY
- Initialize app with a group member's information, check functionality of dosing timing and feedback	Wyatt	Oct 4	10/4/24-WY
- Ensure proper display of medication safety information	Daniel	Oct 4	10/4/24-DC

- General bug testing for app	Jude	Oct 4	10/4/24-JW
- Ensure bottle is able to connect to and interface properly with the app	Wyatt	Oct 4	10/4/24-WY
- While testing app-device interaction, ensure LED displays proper dosing state	Wyatt	Oct 4	10/4/24-WY

All tests outlined in Design Verification Plan will be completed by Dec 5th. If for some unforeseen reason the tests are not able to all be completed, they will be completed in order of risk likelihood and severity of the component failing. I.e. components will be tested in the order of their importance.

6.4 Design Validation

Not Considered in the scope of this class, will be done at a later date

6.5 Process Validation

Not Considered in the scope of this class, will be done at a later date

6.6 Employee Training

Training will occur in BME 3801/4801 lecture and lab at the University of Utah.

6.7 Design Transfer

Not considered in the scope of this class, will be done at a later date

6.8 Design Reviews

Preparation plan for design review presentation-T designates the day of the design review which is found Section 5: Project Milestones

Task	Assigned To	Date Due
Create Slides (Individually)	All	T-5 Days
Collaborate Presentation (group)	All	T-4 Days
Edit Slides	All	T-4 Days
Rehearse (individual)	All	T-2 Days
Rehearse (group)	All	T-1 Day
Meeting Minutes	All	T+1 Day

6.9 Generation of the Design History File

The design history file will be updated throughout the design process by Wyatt and Jude. The current DHF, along with all previous versions will be stored on the UBox where all members can access.

6.10 Generation of the Device Master Record (Design Dossier / Technical File)

Not considered in the scope of this class, will be done at a later date

Pharmacokinetic Control of Opioid Dosing With Feedback
Revision: 3
11/28/2024

Name	Position	Signature	Date
Jude Werth	Team leader, Code and Software	JW	2/22/2024
Kyle Harshany	Design Engineer, Mechanical and CAD Modeling	KH	2/22/2024
Wyatt Young	Code and Software, Market Researcher	WY	2/22/2024
Daniel Candland	Quality Engineer, Electronics and PCB	DC	2/22/2024
Grace Ojewia	Teaching assistant		
Tomasz Petelenz	Instructor		

Revision Log			
Revision	Description	Initials	Date
1	Initial release	DC	2/22/2024
2	Updated problem statement, scope, and applicable standards.	WY	10/21/24
3	Updated 5.12 performance requirements to reflect Vicodin concentration/ weight accounted for in prototyping	WY	11/28/24

1 Introduction
This document details the requirements, user needs, and considerations for the design of a medical device to aid in pharmacokinetic control and record feedback for patients taking opioid medications for pain relief.

2 Scope
The scope of this document is to outline the basic starting points for device development, how patient factors will affect the device, provide economic factors and market analysis, define regulatory requirements, and begin to define the device structure.

3 References
3.1 External References
21 CFR 820.30. FDA Quality System Regulation Subpart C – Design Controls. Effective Date – 6/6/23

3.2 Internal References
Design Control – DC 04 01 – Design Control Policy SOP

Effective Date – 1/18/24
Design Control – DC 04 04 – Design Requirements
Effective Date – 1/18/2024

4 Sources Consulted For Information

Food and Drug Administration
United States Patent Office
Controlled Substances Act
Dr. Kai Kuck – University of Utah

5 Content that must be addressed

5.1 Problem Statement

Prescription Opioid abuse has become more prevalent. Patients are given general dosage instructions which are insufficient for guiding healthy opioid use.

User Needs

- The device will provide opiate-prescribed patients with proper dosage recommendations (number of pills).
- Based on patient pain input, pharmacokinetic factors, and patient weight, the device will determine the number of pills subjective to specific patient.
- The device will accept input from patients regarding their pain level on a given day (pain scale: very bad, bad, neutral, good, very good).
- The device should unlock and lock allowing users to access pill storage.
- The device should be portable.
- The device needs to be rechargeable.

5.2 Product Description

The device calculates and dispenses the optimal dosage of medication based on patient weight, pharmacokinetics, and current pain level. The device reduces the risk of both overdose and underdose by accurately calculating individual dosages. User friendly interface allows patients to actively participate in their pain management.

5.3 Indications for Use

The device is designed to aid in the treatment of pain for patients receiving opioid pain medications.

5.4 Intended Uses

This device is intended to determine timing and size of opiate dosage for a given patient and lock/unlock accordingly.

5.5 Intended Users

The intended users of the opiate dispensing device are patients with chronic pain who have been prescribed an opiate prescription and pharmacies.

5.6 Intended Environments of Use (hospital, clinic, home, etc.)

The device is intended to be used at home or in a personal space.

5.7 Address considerations of public health, safety, and welfare factors (consider entire life cycle)

5.7.1 Public health

This device will have a positive impact on public health by addressing the unmet needs in opioid abuse. This device will reduce cases of opioid abuse or avoidance by providing personalized dosage to patients. This device will increase public knowledge by providing more information about the opioid to the patients.

5.7.2 Safety

The manufacturing of this device will not impose unnecessary risks on those involved in the manufacturing process. The disposal of this device will not release substances which will pose risk to the surrounding environment or the public well-being. This device will lock the bottle during certain time windows. A mechanical override will be implemented in case of emergencies.

5.7.3 Welfare

Patients will have more confidence in the personalized instructions based on pharmacokinetic properties of the medication. This device will give healthcare professionals more confidence in patients properly using opioids within the expected parameters.

5.8 Address global, cultural, social, and environmental factors that impact the design requirements

5.8.1 Global

Opioid Use Disorder is a global issue, with opiates and synthetic opioids ranking second in number of worldwide users after cannabis. The weight of this issue must be balanced with their widespread use and efficacy as analgesics. Thus, the device should be usable worldwide, with multiple language options.

5.8.2 Cultural

Similar to the global factors, the device should be usable across many different cultures, without need for adaptation. Different cultures have different views about prescribing opioids, which will affect where the device will be useful.

5.8.3 Social implications

Due to how addictive they are, opioids carry a social stigma that can lead to feelings of trepidation for users. The device can address this by simplifying dosing, and ensuring that patients can feel safer when taking medication that they might be unsure about.

5.8.4 Environmental

The design of the device should reduce environmental pollutants such as disposable plastics, single-use batteries, or electrical components containing heavy metals.

5.9 Economic Factors

5.9.1 Market Analysis: Market Size (foreign, domestic) including the Market Segments

The primary and immediate market is expected to be pharmacies approved for filling opiate prescriptions and opiate-prescribed patients. We do not expect any other market segments besides hospitals approved to fill opiate prescriptions. Thus, estimating the number of pharmacies, assuming 95% of US pharmacies are DEA-approved and for every single US sale we'll capture 3 sales outside the US, we expect the following market sizes:

- Domestic Market: ~42,465 (assuming 95% of US pharmacies are DEA-approved) plus 3,250 at-home users (.05% of 5-8 million opiate prescribed patients).
- Foreign Market: ~ 127,395 (assuming ~3 foreign sales per US)

We assume each approved pharmacy will require ~50 units to replace current bottle stock, aiming for a 3% market capture (1st year, 8% after 5 years) based on the number of pharmacies and opiate-prescribed patients:

Year 1: 3% of market

- Domestic Initial Stock Sales: ~66,948 units
- Foreign Initial Stock Sales: ~191,100 units

Year 5: 8% of market

- Domestic Initial Stock Sales: ~173,110 units
- Foreign Initial Stock Sales: ~509,580 units

5.9.2 Competitive Product Analysis

Philips Spencer SmartHub™

At-home medication management platform that includes a smart hub that dispenses medications at scheduled times, integrates with telehealth services, and monitors health data. Uses 'adherence insights' that consider schedules, beliefs, habits, and individual conditions. Utilizes feedback from patients and caregivers.

https://www.usa.philips.com/healthcare/services/population-health-management/patient-engagement/medication-adherence/medication-dispenser -accessed 2/17/24

Price: Through provider

Hero Smart Dispenser

Automated medication dispenser capable of holding up to 90 days' worth of up to ten different medications. It features a sorting mechanism for easy loading, a companion app (IOS and android) for monitoring and alerts, and audible dose reminders (as well as blinking light). Connects via WiFi, and includes safety measures such as password protection, dispensing limits, and low-pill alerts. Supports any pill size or shape.

https://herohealth.com/ -accessed 2/17/24

Price: Device costs $99 upfront, with an added $29.99/month for access to the caregiver app. Includes many price packages.

MedControl Systems DoseControl Smart Bluetooth Pill Dispenser Model 2021

DoseControl uses a rotating pill container with 28 compartments. It pairs with an app (Android only, BT connectivity) for remote programming and monitoring, featuring customizable alarm settings (email notifications) and a locking mechanism. LCD display with 4 mechanical buttons. Charging via USB-C.

https://www.medcontrol.eu/p/386/smart-automatic-pill-dispenser-with-alarm-dosecontrol-new-model-2021-english-transparent-lid-connected - accessed 2/17/24

Price: 129 €, ~$139.01 USD

MedReady Inc. PLUS CELLULAR w/ Flashing Red Light MR-357FL

Mechanically locked dispenser with a flashing light for dose alerts. Wireless connectivity through internal cellular module for remote monitoring without phone lines or internet. Capacity for multiple daily doses. Many safety measures are tied in with dosage timings, i.e. sends an alarm to Monitoring center to call, email, and text emergency contacts of patient if medications are not removed be the end of alarm duration.

https://www.medreadyinc.net/ -accessed 2/17/24

Price: $359.00

5.9.3 Intellectual Property Review

- A61J:
  - A61J 3/00 - "Containers specially adapted for medical or pharmaceutical purposes"
  - A61J 1/03 - "for pills of tablets with special dispensing means therefor B65D 83/04"
  - A61J 7/00 - "Devices for administering medicines orally"
  - A61J 7/02 - "Pill counting devices [2006.1]"
  - A61J 7/04 - "Arrangements for time indication or reminder for taking medicine, e.g. programmed dispensers"
  - A61J 2200/74 - "Device provided with specific sensor or indicating means for weight"
- G16H:
  - G16H 20/13 - "ICT specially adapted for therapies or health-improving plans, e.g. for handling prescriptions, for steering therapy or for monitoring patient compliance delivered from dispensers"

Patent Applications:

"Apparatus And Method For Dispensing Medication" Filed by Karpman et al.

- **Application Number:** 17/306919
- **Current US Class:**
  - A61J 7/0084, B25J 15/0683, G16H 20/13, A61J 2200/74, A61J 2205/30
- **Relevant Claims:**
  - **Claim 1:** A method for periodically dispensing multiple medications to at least one individual comprising the steps of: providing a plurality of storage bins to store dosages of specific medications, each storage bin of said plurality of storage bins being designated to hold only dosages of one specified medication; providing a probe with a suction cup at its end wherein said probe with cup at its end can be moved among said storage bins and said probe can grasp a dosage of medication located in any of said storage bins on activation of a pneumatic vacuum system connected to said suction cup and move said probe with said suction cup grasping the dosage to a delivery bin and deposit it in said delivery bin, wherein said suction cup has an articulated body surrounding a hollow interior which is in communication with a vacuum system at an opening at a first

end of said suction cup articulated body, and a large lipped surface at a second end, said large lipped surface having an aperture at its center which opens into said hollow interior, said suction cup being made of a soft pliable rubber like material; providing a computer system that can be programmed to control and sequence the operation of said probe with said suction cup attached thereto: a. move at a preset times with said probe and suction cup predetermined dosages of medication form said storage bins to said delivery bin; b. signaling that said delivery bin is filled with the predetermined dosages for consumption by a patient at a preset time; c. continuing to send signals that the dosages of medication are ready for consumption until a sensor indicates that the dosages have been removed from said delivery bin; d. recording in a memory of said computer the fact the dosages of medication have been removed from said delivery bin; e. providing information on dosages of medication dispensed from the delivery bin previously recorded to various care givers and providers; and f. periodically conducting a purge cycle to remove particulate matter from said probe and said suction cup; programming said computer for loading dosages of medication into predesignated storage bins of said plurality of said storage bins comprising the steps of: a. initiating a start of a computer program medication dosage loading routine; b. indicating whether the dosages of medication to be loaded are for a new prescription, or an existing prescription already stored in a specific predesignated storage bin; c. running the following subprogram if the prescription has been indicated to be an existing prescription already stored in a specific predesignated storage bin: i. reviewing information on said existing prescription in said computer system; ii. making any corrections to the information in the computer system on the selected existing prescription; and iii. verifying the correctness of the information after step c. ii above is completed; iv. depositing the new dosages of mediation of the existing prescription in said specific predesignated storage bin; d. running the following subprogram if said prescription is new: i. designating an available empty storage bin from among said multiple of said storage bins as the storage bin into which the dosages of the new prescription will be stored in; entering the information regarding the new prescription into said computer system; iii. confirming the information regarding the new prescription is correct; and iv. depositing the dosages of mediation of the new prescription into said predesignated storage bin for the new prescription.

- **Claim 3:** The method of claim 1 wherein the step of entering the prescription information is the step of entering the following information regarding the prescription: 1) name of the medication, 2) prescribing physician, 3) number of dosages being inserted into said specific storage bin, 4) amount of medication in each dosage, and 5) time at which the medication is to be taken and the amount of dosage to be taken.

FIG. 17

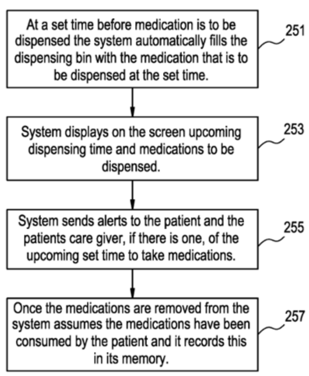
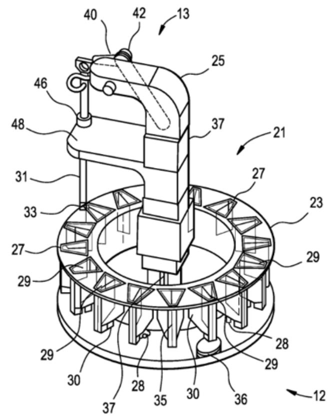


FIG. 2



"Cap Assembly For A Medication Container" Filed by Hoffman et al.

- **Application Number:** 17/122656
- **Current US Class:**
  - B65D 83/049, B65D 83/0427, A61J 1/03, A61J 7/049, A61J 1/1418, B65D 83/0083, A61J 7/0454, A61J 7/02, A61J 7/0418, A61J 2200/70, A61J 2205/10, B65D 2583/0472, A61J 2200/30, A61J 2200/72
- **Relevant Claims:**
  - **Claim 1:** A medication container, comprising a receptacle having an inner space for holding medications; a cap assembly coupled with the receptacle for retaining the medications in the inner space, the cap assembly including an outer piece and a middle piece and an inner piece, the outer and inner pieces being fixedly attached, and the middle piece being fixedly attached with the receptacle; the outer, middle, and inner pieces have at least one pill opening, the pill openings of the outer and inner pieces are circumferentially spaced apart from one another; and the middle piece being rotatable with the receptacle relative to the outer piece and the inner piece to transport a pill through a curved path from the pill opening of the outer piece to the pill opening of the inner piece or from the pill opening of the inner piece to the pill opening of the outer piece to either dispense the pill from the receptacle or to insert the pill into the receptacle.
  - **Claim 5:** The medication container as set forth in claim 4 wherein the cap assembly further including at least one medication sensor configured to detect the passage of pills through the cap assembly either into or out of the receptacle.
  - **Claim 6:** The medication container as set forth in claim 5 wherein the cap assembly further includes a memory and a microprocessor that is configured to

record data related to the passage of pills into or out of the receptacle to the memory.

- **Claim 8:** The medication container as set forth in claim 6 wherein the cap assembly further includes a light that is attached with the probe for providing an alert to a user.
- **Claim 9:** The medication container as set forth in claim 8 wherein a contact extends through a through opening in the probe from the light to a circuit board that is attached with the disk.

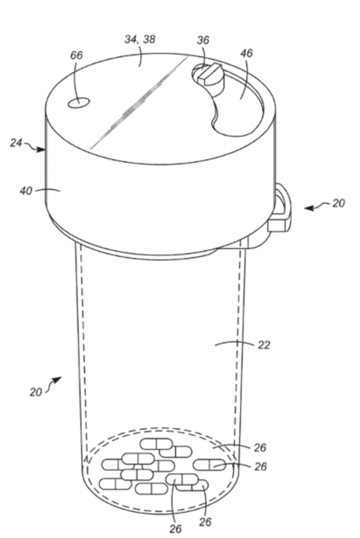


FIG. 4

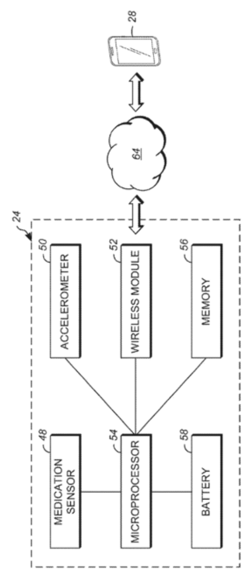


FIG. 7

"Narcotics And Opioids Secure Storage And Dispensing Apparatus And Method Of Use" Filed by Daniel Gershoni.

- **Application Number:** 17/726455
- **Current US Class:**
  - A61J 7/0427, A61J 7/0454, A61J 7/0069, A61J 7/0084, A61J 7/0418, G16H 20/13, A61J 1/035, A61J 7/0481, A61J 2200/70, A61J 2200/30, A61J 2205/60
- **Relevant Claims:**
  - **Claim 1:** A secure medication dispensing system comprising: a multi-tray medication dispensing apparatus and software for operating the multi-tray medication dispensing apparatus, the multi-tray medication dispensing apparatus comprising: a. a microprocessor; b. memory in signal communication with the microprocessor; c. software comprising a set of instructions that are executed by the microprocessor, the software being stored in the memory; d. a wireless

transceiver in signal communication with the microprocessor; e. a first tray, the first tray comprising a plurality of first tray medication compartments containing primary medication, the first tray being arranged to dispense the primary medication from each first tray medication compartment into a medication dispensing location; f. a second tray, the second tray comprising a plurality of second tray medication compartments containing secondary supplemental medication, the second tray being arranged to dispense the secondary medication from each second tray medication compartment into the medication dispensing location; g. at least one closure, wherein the at least one closure provides access to and a secure seal to each of the first tray and the second tray; h. a user identification system in signal communication with the microprocessor, i. a first tray operating control mechanism, wherein the first tray operating control mechanism positions the first tray respective to the dispensing location, the first tray operating control mechanism being controlled by the software; j. a second tray operating control mechanism, wherein the second tray operating control mechanism positions the second tray respective to the dispensing location, the second tray operating control mechanism being controlled by the software; k. a sensor in signal communication with the microprocessor, the sensor being arranged to detect when medication has been dispensed; and l. A housing carrying the microprocessor, the memory, the sensor mechanism, the first tray, the second tray, and at least one medication dispensing mechanism, the software comprising steps of: requesting dispensing of the primary medication from the first tray; initiating a request for information from the user as a result of the request for dispensing of the primary medication from the first tray, wherein the information requested is to determine an appropriateness of the primary medication for the user; entry of information for authorizing dispensing of the primary medication from the first tray, wherein the entry of information is provided by the user, the entered information is considered to determine: (a) if dispensing the primary medication from the first tray is appropriate based upon the information entered by the user, (b) if dispensing the supplemental medication from the second tray is appropriate based upon the information entered by the user, or (c) if providing an alternative action to dispensing any medication is appropriate based upon the information entered by the user; wherein, in a condition where the entered information determines the condition to be appropriate to dispense primary medication from the first tray, the multi-tray medication dispensing apparatus dispenses primary medication from the first tray, wherein, in a condition where the entered information determines the condition to be appropriate to dispense supplemental medication from the second tray, the multi-tray medication dispensing apparatus dispenses supplemental medication from the second tray, and wherein, in a condition where the entered information determines the condition to be appropriate to provide an alternative action to dispensing any medication, the system initiates an alternative action.

- **Claim 5:** The secure medication dispensing system as recited in claim 4, further comprising a punch lever arranged to excise medication from the at least one blister pack via a combination of a radial motion of the punch lever in combination with a rotational motion of the respective one of the first tray and the second tray.

- **Claim 6:** The secure medication dispensing system as recited in claim 1, wherein the primary medication stored in the first tray medication compartments of the first tray is classified as a class II drug.

- **Claim 19:** The secure medication dispensing system as recited in claim 1, the multi-tray medication dispensing apparatus further comprising: m. a display in signal communication with the microprocessor; n. a microphone in signal communication with the microprocessor; and o. a speaker in signal communication with the microprocessor, wherein the housing additionally carries the display, the microphone, and the speaker, the software further comprising a step of: interacting with a user via an interactive artificial intelligence virtual assistant, wherein the interactive artificial intelligence virtual assistant communicates with the user via the display, the speaker, and the microphone.

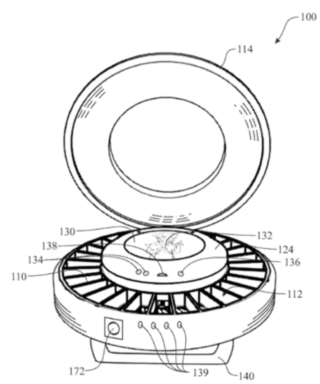


FIG. 2

"Dispensing Cartridge" Filed by Akdogan et al. Assignee: Hero Health, Inc.

- **Application Number:** 15/174187
- **Current US Class:**
  - G16H 20/13, H04N 7/188, A61J 7/02, B01D 53/261, B65D 83/0409, A61J 1/03, G16H 40/67, G07F 9/026, A61J 7/0481, B65D 81/268, G07C 9/32, G07F 11/44, A61J 7/0076, G07F 17/0092, B01D 53/0454
- **Relevant Claims:**
  - **Claim 5:** The device of claim 1 further comprising an attachment coupled to the housing by an attachment mechanism, wherein one the items is dispensed through the dispensing hole and into the attachment.

- **Claim 6:** The device of claim 1 further comprising a memory configured to store data regarding the items.
- **Claim 8:** The device of claim 1 wherein each of the items has a specific item size and item shape, and wherein the dispenser tube includes a size and shape configured to cooperate with the item size and item shape and to dispense one of the items.

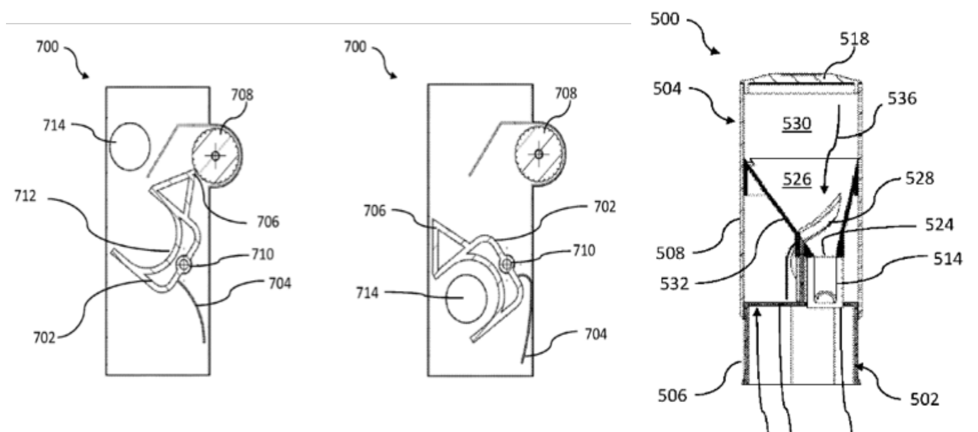


FIG. 7A FIG. 7B

"Pill Dispensers, Systems, And/Or Methods" Filed by Duda et al.

- **Application Number:** 17/839428
- **Current US Class:**
  - A61J 7/0454, B65D 83/04, A61J 7/0084, A61J 1/00, A61J 7/0481, A61J 7/0418, A61J 1/03, A61J 7/0427, A61J 2200/30, A61J 2205/10
- **Relevant Claims:**
  - **Claim 1:** A pill dispenser device, system or method as shown and/or described herein.
  - **Claim 2:** A system for dispensing one or more pills comprising: a receptacle bin sub-assembly including a receptacle bin, the sub-assembly configured for collecting one or more pills in the receptacle bin; a rotational control sub-assembly including a rotational motor connected to the receptacle bin sub-assembly for rotating the receptacle bin sub-assembly; a nozzle sub-assembly including a nozzle configured for connecting to the receptacle bin for receiving and passing the one or more pills from the receptacle bin through the nozzle; a chute sub-assembly including a chute and a chute known location connected to the nozzle sub-assembly for receiving one or more pills from the nozzle sub-assembly and moving the pill to the known location; a delivery sub-assembly for

receiving one or more pills from the chute known location to then deliver the one or more pills to a final delivery position.

- **Claim 8:** A system according to any of the preceding claims, the rotational sub-assembly comprising one or more of: a turntable bearing; or a gear plate; or one or more contacts; or one or more sensors; and the gear plate having one or more of: one or more alignment posts; one or more inlets.

- **Claim 12**: A system according to any of the preceding claims, the scale sub-assembly further comprising one or more of: a scale platform; a swiper motor; and a pill swiper cooperatively connected to the swiper motor.

- **Claim 17:** A method according to any of the preceding claims further comprising: weighing the one or more pills on the scale.

- **Claim 24:** A method according to any of the preceding claims further comprising: providing instructions or signal from a main controller or CPU board to one or more of: one or more pill alert LEDs; low voltage LED; pill refill LED; and speakers.

- **Claim 29:** A pill dispenser device, system, or method according to claim 25, the chute known location includes one or more cams for lifting one or more pills form the know location and moving the one or more pills to the delivery area.

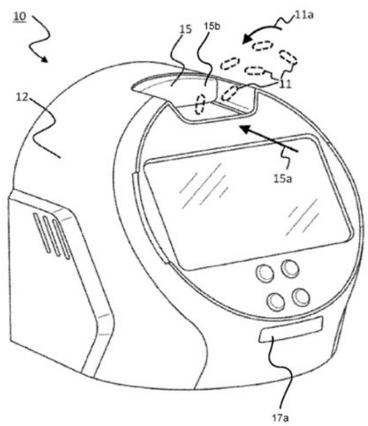
- **Claim 35:** An apparatus according to claim 30 further comprising: a swiper sub-assembly cooperatively connected to a motor configured to move one or more pills to an output delivery cup, or the collection area of the return tube.

- **Claim 41:** A method for dispensing one or more pills from a pill dispensing device comprising one or more of: entering prescription drug name and/or NDC, entering the dosage and/or manufacturer; entering a regime or schedule; placing pills in the pill receptacle bin area or a bottle located in the dispenser; running a timer according to the regime or schedule; dispensing pills; and/or alerting user to take pills.

- **Claim 42:** A method for dispensing one or more pills according to claim 41, further comprising: counting the pills remaining or counting the pills consumed; determining if a sufficient quantity of pills are present; notifying a pharmacy or third-parties if the count is running low, or if in need of a refill.

- **Claim 51:** A pill dispenser device, system or method according to claim 50 wherein the chute is configured to move the pill in any one or more of a lateral, upward or downward direction.

- **Claim 73:** A system comprising: one or more microprocessors; a communication interface device; and one or more internal data storage devices operatively coupled to the one or more microprocessors: the communication interface used to input one or more of pill or regimen or user data; the internal storage storing one or more of pill or user data; the one or more processors using the input at/from the interface and/or the data from the storage to determine appropriate delivery times and amounts; and, the one or more processors controlling the delivery process, including at least initiating the pick-up members movement to move a pill from the known location.



5.9.4 Sale Price Estimates

Similar pill dispensing devices have a range of prices based on their connectivity with patient and clinician, storage size, number of types of pills in storage, and electronic features (LED interface, speakers/alarms, LED alarm lights, etc.). The proposed product will feature limited connectivity with a smartphone app for the patient and just a couple of the specified features seen in competitive products. The automated pill bottle will also be much smaller in storage than competitor products. Due to these differences, the product must be priced lower than competitive products while also being manufacturable for profit.

Proposed Sale Price: 50$

5.10 Regulatory Requirements and Standards

5.10.1 Regulatory Requirements and Approach

Classification: Class I – Minimal potential for harm
The device must follow all general controls for medical devices. Because the device is class I and does not fall under any specific exemptions, it will be 510k exempt.
Must keep following records:

- Design History File (DHR) 820.3e
- Quality System Record 820.186
- Complaint files 820.198
- Device Master Record (DMR) 820.181
- Device History Record (DHR) 820.184

Because it will be used in the administration of controlled substances, the device must not interfere with or prevent compliance with regulations for filling prescriptions, including schedule II drugs, which includes most opioids. Regulations for prescription medications are outlined in the Food, Drug, and Cosmetics Act, 21 CFR 1306. Regulations for controlled substances are found in the Controlled Substances Act, 21 U.S.C § 801. The regulatory pathway to market is first, determine device classification (I, General Controls). If there are issues with classification, it will be necessary to select, prepare, and submit premarket submission (510k), and await response from FDA. Then, establish GMP, device listings, and set up post-market surveillance

5.10.2 Applicable Standards

The following standards will be adhered to:

- ISO 13485 – Quality Management System
- ISO 14971 – Risk Management
- IEC 60601 – Electrical Standards
- IEC 62366 – Usability
- IEC 62304 – Software Standards

5.11 Functional Requirements (user's terms of what the device must do)

5.11.1 Clinical Functional Requirements

- Patients need a device that accurately calculates opiate dosage based on individual factors like pain level, weight, and pharmacokinetics.
- It will lock and unlock based on calculated dosage.
- The device will weigh its contents and calculate the number of pills.

5.12 Performance and Physical Requirements (engineering terms of how the device must perform and look)

5.12.1 Performance Requirements

- This device will be able to withstand basic stress levels without deforming
- This device will have a lid which tightly seals to the bottle (CT specification: 28-400)
- This device will have an internal locking mechanism that disallows cap from being removed.
- This device will be able to store a full prescription of 5 mg Hydrocodone/ 305 mg Acetaminophen (15 mm, oval shaped) capsules.
- This device will be able to connect to a device via cloud data transmission
- This device will be able to weigh its contents with less than 200 mg of error
- This device will be able to transmit it’s detected weight to cloud storage
- This device will provide dosage recommendations based on software
- The device’s interface will accept patient feedback (no pain, little pain, heavy pain)
- The device’s interface will adjust dosage recommendations based on feedback
- The device’s interface will have maximum dosage rates based on the pharmacokinetic properties of the medication

5.12.2 Physical Embodiment

- This device should be a cylinder that is 2.85” tall with a diameter of 1.28” (industry standard pill bottle)
- The device should fit in a small purse or handbag (portable)
- This device should weigh no more than 40g

- This device should be capable of locking its cap given conditions transmitted from the patient interface
- This device should have a mechanical override to open the bottle

5.12.3 Describe the Environmental Conditions (in Engineering Terms) that the device will be used in

Device will be used in normal environmental conditions, around room temperature, normal humidity, and usually indoors.

5.12.4 Device Compatibility Requirements to meet Environmental Conditions

The device should function well under normal use circumstances, and in the event of user error, (getting it excessively wet, leaving in extreme heat) the mechanical override will allow the patient to access medication in emergencies.

5.12.5 Safety Requirements (User, Patient)

The device should be designed to avoid sharp edges or pinch points, as well as eliminating risk of electric shock.

5.12.6 Environmental and Disposal Requirements

The device will be reuseable, and if broken, should be safe to dispose of in the trash, or with deconstruction, recyclable.

5.12.7 Reliability

Device should have a battery life of 36-48 hours, fully charge in 1 hour or less, and should be made of durable materials. An ideal minimum lifespan of the device would be 3-5 years.

5.12.8 Stability

The device should be made using stable materials, meaning stability over time should not be an issue. The device should protect medication from excess exposure to air, light, and humidity, with comparable or improved efficacy to a prescription pill bottle

5.12.9 Shelf Life

On its own and stored away from high humidity and heat, the device should last as long as the electrical components inside of it, most semiconductors are rated for shelf life of around 20 years (Texas Instruments, *Long Term Storage Evaluation of Semiconductor Devices*), which would likely be the shelf life of the device.

5.12.10 Sterilization / Sterility

The device should be cleanable for reuse, but does not need to be sterilized at any point.

5.13 Human and System Interfacing Requirements

5.13.1 Human Factor Considerations (Ergonomics)

- The device must be light and small enough to be carried in a small purse or large pant pocket (portable).
- Any information provided to instruct the user shall be provided in the instructions for use (IFU).
- Labels/markings on device surface must be legible.
- Edges of device and instrumentation case should be chamfered to avoid sharp edges.
- Device must be useable by patients experiencing pain or loss of mobility in the hands.
- The device must have simple locking and unlocking mechanism similar to common pill bottle.
- Specifically, include normal threading that can only be initiated once mechanism unlocks.

5.13.2 User and/or Patient Interface

- The device's accompanying app software should prompt patient, upon startup, to enter weight as well as any other dosage timing information needed for their prescription.
- The device will be operated via the app, as well as simple buttons on the device

5.13.3 Interfacing Devices and Accessories with Accompanying Interface Requirements

- The device will come with a charger, to allow the patient to recharge when necessary.
- The device will interface with a smartphone app, which will provide the patient with drug, dosing, and pharmacokinetics info, as well as query the user to collect data that will be used to provide personalized support, as well as helping doctors understand the patient's dosing behavior.
- The device will interface with the app via local Bluetooth.

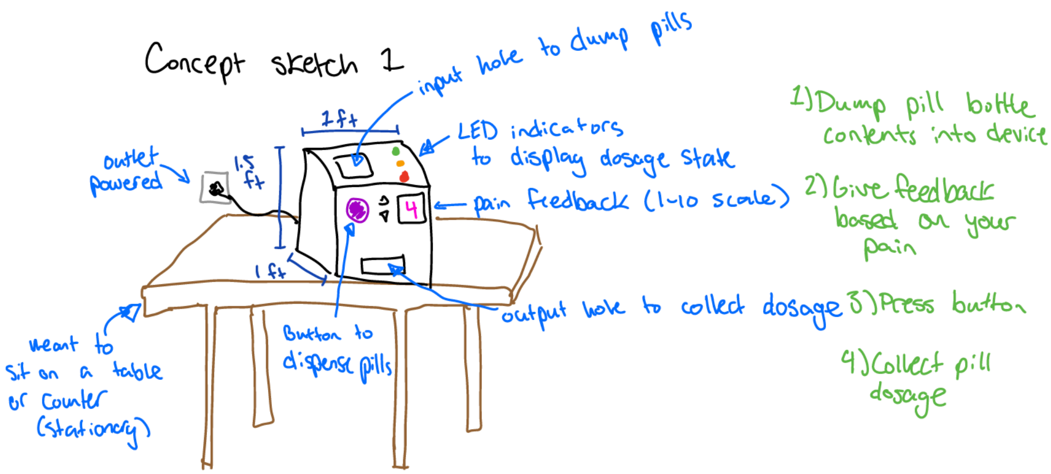
5.13.4 Packaging and Labeling Requirements

- Packaging and Labeling will follow 21CFR820.130 (packaging) and 820.120 (labeling).
- Each device will have a unique device number printed or embossed on it All labels, markings, or symbols on the device will be printed or embossed in a manner that is clear and resistant to wear.
- The device must have space available for proper prescription labeling as defined in 21 CFR 1306.14.
- Packaging will be a simple box, containing the device, a charging cord, and an information pamphlet with instructions for use of the device and accompanying app.
- Labeling on the box and information pamphlet will include basic information on the device's purpose and use, and specifications such as supply voltage, battery life, weight, dimensions, and storage capacity.

- Any standards certifications will be printed on the box and information pamphlet.
- Packaging should protect the device from drops or scratches during shipping and storage in normal environments.

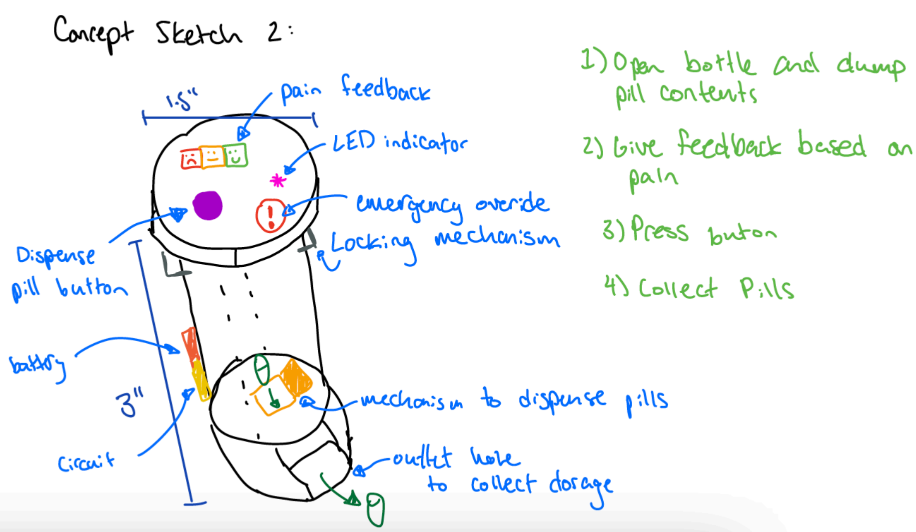
5.14 Conceptual Designs (see SOP for examples)

5.14.1 Concept 1



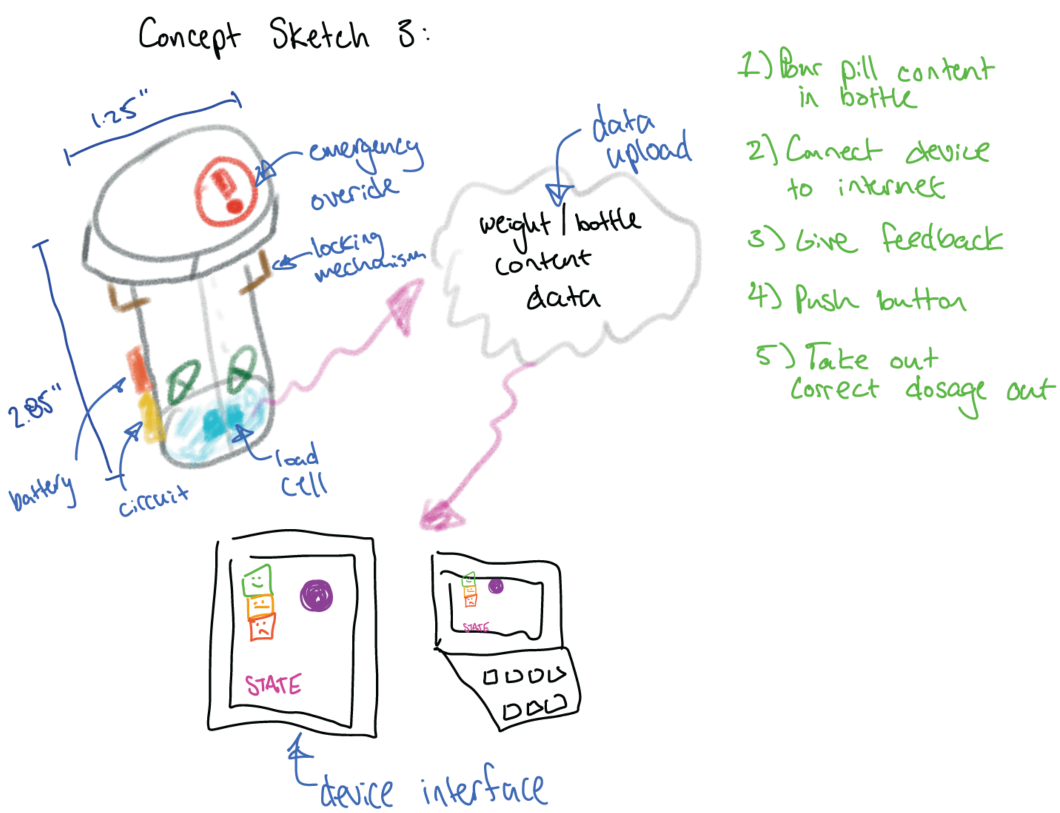
Concept 1: “Full Hardware” Device has a very simple accompaning app, dispenses pills one at a time, and contains almost all interfacing on the device itself.

5.14.2 Concept 2



Concept 2: “Light Hardware” Device is essentially a locking pill bottle with simple imput controls, which communicates with an app that provides information, collects data, and can be used (instead of the buttons) to report feedback and dispense pills. This version would likely use a load cell to monitor dosing.

5.14.3 Concept 3



Concept 3: “Full Software” Device is a very simple locking pill bottle, with no input controls. All input is through the accompaning app.

5.14.4 Storyboard or other mechanism to describe how your preferred design concept will meet user needs and solve the design problem

The “Light Hardware” concept provides simple operation through the on-device buttons, while also providing plenty of information and intractability through the app, meaning the patient has multiple options of how to use the device.

Name	Position	Signature	Date
Jude Werth	Team leader, Code and Software	JW	3/28/2024
Kyle Harshany	Design Engineer, Mechanical and CAD Modeling	KH	3/28/2024
Wyatt Young	Code and Software, Market Researcher	WY	3/28/2024
Daniel Candland	Quality Engineer, Electronics and PCB	DC	3/28/2024
Grace Ojewia	Teaching assistant		
Tomasz Petelenz	Instructor		

Revision Log			
Revision	Description	Initials	Date
1	Initial release	WY	04/12/24
2	Updated functional specs table	WY	10/6/24
3	Updated 5.3.2 to current status(no longer relying on RTC)	WY	11/25/24
4	Added SOP reference for sources generated, updated app specifications as well as locking mechanism specifications. Added updated prototype images.	WY	12/05/24

1 Introduction
This document will cover the design specifications for the doseX device for pharmacokinetic control of opiate dosing. The following design specifications are based on the design requirements document. This document will be formatted according to Design Control – DC 04 06 – Design Specifications rev 07 03-08-23 provided for BME 3801/4801 2024 created in accordance with FDA 21CFR 820.30(c).

2 Scope
The following design specifications document's scope is to develop specifications for the design of a device for pharmacokinetic control of opiate dosing during the Fall 2024 semester of BME 4801 bioDesign course in the Department of Biomedical Engineering at the University of Utah.

3 References

3.1 External References
- FDA 21 CFR 820.30 of the Quality System Regulation
- Design Control Guidance for Medical Device Manufacturers, available

from CDRH
- Do It by Design: an Introduction to Human Factors in Medical Devices, available from the FDA

3.2 Internal References

4 Sources Used to Generate Specification (identify)

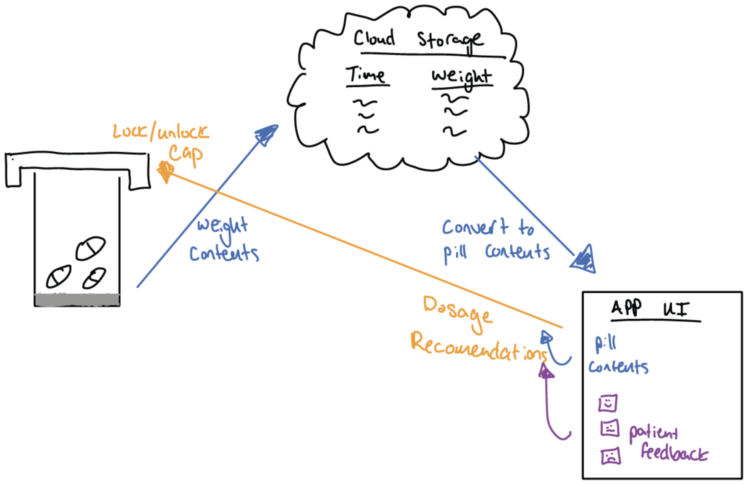
- Design Control – DC 04 06 – Design Specifications rev 05 07-18-19

5 Content (that must be addressed)

5.1 Description of Design Solution and how it meets Design Requirements

5.1.1 Provide a description of the design solution

- This device is a pill container which controls opioid access based on pharmacokinetic factors and patient feedback. This solves the design problem by deterring prescription opioid abuse and keeping an intake record. The device will use Wi-Fi connection to upload its weight contents to a cloud database, Adafruit IO. With this information the pill contents can be tracked to ensure proper dosing occurs. This information will be available to the patient via an app interface. The app will also give dosage recommendations based on patient physiology and a tolerance profile. The tolerance profile will be generated by using patient feedback also available on the app interface. Patient feedback will be quantified using three options relating to the pain level of the previous dosing interval ("high pain", "low pain", "no pain"). Dosage intervals and maximum dosage parameters will be based on the drug profile. This profile will display its contents, pharmacokinetic properties, side effects, and important properties. For a prototype, we will profile Vicodin, an oxycodone medication.
- This device addresses the user needs by giving personalized dosage recommendations on the app interface. This solution addresses convenience needs by being rechargeable and portable (defined below).
- The device is indicated for the improved self-dosing of opioid/pain relieving medications. It does this by giving personalized dosage recommendations based on patient physiology, patient feedback, and pharmacokinetic properties of the medication.
- This device is only intended for a single opioid/pain relieving medication prescription. The device is only designed to be used for medication defined by the drug profile (Vicodin).
- This device is intended to be used by patients experiencing chronic pain. Patients will be expected to interact with the device, give feedback, and take dosages independent of clinicians.



5.1.2 Summarize functional and performance specifications

Feature	**Description**	**Associated Metric**
Bottle	• Hold an entire prescription (32 pills) • Be able to withstand everyday heat and force stress. • The bottle will not be reactive with the pills	• It will be a cylinder with a diameter of 1.03 in and a height of 2 in. • The bottle will be made from Polyetherimide (PEI).
Electric Container	• Hold electric components on the device. • Be able to withstand everyday heat and force stress. • The compartment will not be reactive with the pills	• It will be a rectangle with the dimensions 2.08in x 1.04in x 1.21in. • The container will be made from PEI.
Lid	• Keep bottle closed and pills inside • It will have a threading system to secure the lid and bottle together.	• Made from PEI material. • ~~Thread Dimensions:~~ • ~~Height: 0.213 in~~ • ~~Diameter: 1.2 in, on center~~ • ~~Rotation: 0.5~~ • ~~Thickness: 0.05 in~~ • Instead of threads, a 90-degree push and twist lock system will be used

Load Cell	• Low-Capacity Load Cell capable of accurately measuring bottle contents.	• 35 mm single point load cell • Max capacity: 100 g • Min supply voltage: 3 V DC • 4 mounting holes
Solenoid Lock	• Capable of locking/unlocking pill bottle and lid.	• Solenoid with 20 mm long body • Activated with 6 V DC
External Button	• Manual override for lock, located on the bottle.	• Dimensions: 7.8 mm x 7.8 mm x 4.9 mm • At least 2 pins: one to ground, one to GPIO • Operating force: 120 ± 40 gf
Battery	• Provides energy to the device, needs to be rechargeable.	• LiPo battery • Size: 35mm x 24mm x 5.2mm • Output: 3.7V-4.2V • 1200 mAh capacity
Microcontroller	• Interpret electrical device information. • Transmit information via wifi. • Control circuited hardware components such as LED, button, load cell, and solenoid circuit.	• Onboard RBG LED • 2.4 GHz wifi radio • 5V to 3.3V LDO • Micro-USB charging regulator circuit • 8 MB PSRAM • 8 MB Flash • > 6 GPIO pins • 1 I2C connection header • On-board LED
LED light	• Capable of displaying dosage status i.e red, yellow, and green.	• NeoPixel • Addressable via GPIO pin • 3.3 V

		• 3.5 mm x 3.5 mm • Max brightness of 24 lumens per led • Linearly adjustable via PWM
Analog Digital Converter (ADC)/Signal conditioning chip	• Convert load cell bridge to value with resolution • Capable of differentiating between specified Vicodin pills.	• 24-bit resolution • 128 gain • Calibration circuitry • I2C terminals

5.1.3 Provide a description of how the design solution addresses considerations of public health, safety, and welfare factors (consider entire life cycle)

- The PEI material used for the bottle and lid is nontoxic and nonreactive. Carrying it in your hand and exposing the surface to everyday materials will not cause a reaction.
- The device will be rechargeable and reusable, therefore, less waste will be produced compared to the standard pharmacy pill bottle.
- If broken or contaminated, the device will be disposed of in standard trash bins. This disposal method will not produce any harmful substances or destroy wildlife or resources necessary for good public health.
- The electrical components will be safely and securely shielded from the exterior. This will preserve the device's function and prevent electrical related injuries.
- The device will be thoroughly sterilized and cleaned before being shipped. This will ensure there's no contamination of the stored medicine.
- This device will provide patients with background information about the drugs they're taking on the drug profile.
- This device will improve safety of self-dosing (lower overdoses) by setting dosage parameters.
- This device will have an external button as a manual override to the lock. This will ensure patients can take their medication in the event of an emergency.

5.1.4 Provide a description of how the design solution addresses global, cultural, social, and environmental factors that impact the design requirements

- The app will feature buttons with symbols such as emoticons representing pain levels, a symbol depicting a lock to represent unlocking the bottle, gear for settings, and will contain text instructions for use in a digital format, allowing for easier translation.
- The physical device will utilize symbols for buttons.

- Charging will be done via a micro-USB cable, which can easily be adapted to 110 or 220 V standards.
- Avoid language stigmatizing the use of opiates for pain management, by using proper medical language, according to the AMA style guide.
- The device will limit access to medication when dosing is not safe, and let the patient know when it is safe to take another dose to help the patient regulate and be conscious of their intake, to avoid overdosing.
- The device is reusable, being refilled rather than thrown away.
- The device will be made of recyclable PEI plastic

5.1.5 Provide a description of how the design solution addresses economic factors

Include market size implications

- Market segments include ~43,000 US pharmacies that are DEA-approved to distribute Hydrocodone, ~128,000 international Hydrocodone distributors, and an additional ~3,250 individual purchasers.
- Pricing the pill bottle device as a cheaper alternative to expensive stationary automated pill dispensers, such as those seen in our competitors, will still capture an untapped share of the market; portable automated pill dispensing bottles that still interact with the patients pain level and dosage history. The device will obviously be more expensive than the common UV-protected simple threaded pill bottle, which goes for $0.06-0.10. However, by selling to pharmacies and the reusability of the automated pill device we expect to capture a notable portion of that market as well.
- Due to this device being prototyped to handle Hydrocodone prescriptions, the market capture will be limited to a single opiate. However, simple inputting of pharmacokinetic factors (such as drug release timing) for other drugs allows for the device to be utilized across all opiate prescriptions.

Competitive products and IP concerns

- Phillips Spencer SmartHub: Priced at around ~$350 based on other products with similar features. Distributed internationally. Due to basic functionality of supplying patients with pills, IP overlap could occur in claims pertaining to CPC A61J. Overlap also occurs in CPC G16H 20/13, referring to the SmartHub accepting input from physicians and patients to steer compliance or drug therapy.
- Hero Smart Dispenser: Priced at $99.00 upfront, with an added $29.99/month for access to accompanying app. Distributed internationally. Due to basic functionality of supplying patients with pills, IP overlap could occur in CPC A61J. Overlap also occurs in CPC G16H 20/13, referring to the Smart Dispenser dosage limits and safety measures. Many of the mechanical elements in Hero Health Inc. main

patent uses broad language mostly pertaining to the dispensing action. However this patent also includes many claims pertaining to the data storage which will cause IP overlap with the portable automated pill bottle device.

Sale Price Estimates

- Manufacturing methods include 3D printing, printed parts assembly, microelectronic assembly, and software flashing. All of these manufacturing processes are inexpensive. Thus most manufacturing costs will come from filaments, electronic components, and labor of course. This strategic manufacturing scheme makes the device accessible to most international regions, as the most expensive equipment required is a 3D printer. By minimizing manufacturing costs and materials the device will have a lower price making domestic and international market capture goals attainable.
- **The price of the device will be $70 per unit.** Includes profit margin while still being inexpensive enough to capture market between high-end pill dispensers and the commonly used orange pill bottles.

5.1.6 Provide a description of how the design addresses regulatory requirements and standards

- The bottle will contain a smooth space where a 3”x2.5” pharmacy sticker labels can be applied, in order to comply with 21 CFR 1306.14 for Schedule I and II narcotics, or 21 CFR 1306.24 for Schedule III-V

5.1.7 Provide a description of how the design addresses reliability and stability requirements

- In order to provide 36 hours of battery life, a 5 volt, 2200 mAh rechargeable battery will be used.
- The electrical components will be protected from tampering and damage by encasing them within a PEI shell
- The PEI bottle will be opaque to protect medication from light, and the screwcap will have a tolerance of < 0.5 mm in order to keep out dust and water

5.2 Human Factors and Usability Engineering

5.2.1 User and/or patient interface

List the interfaces (e.g. controls, displays, handles, grips) that exist between the user and / or patient and this product.

1. Opiate prescribed patients will need to be able to grip the cap and access the pill compartment when unlocked, even if patient is losing mobility in hands due to pain.
2. Patients must be able to access pill compartment without software intervention in case of emergency (phone loses power/connection).
3. An LED indicator light must be visible by users that tells patients their current dosage status.

4. Patients must be able to access companion app software.
5. Patients must be able to enter their current pain level into app software via a pain scale from 0-2.
6. Device should include a tamper-proof lock to prevent access to compartment when not authorized.
7. Device must be rechargeable by patients.
8. Design must include a compartment for patients to load their entire prescription of pills into.

List design solutions that address these user/patient interfaces.

1. Well-beveled threads (Dimensions: Height: 0.213 in, 1.2 in on center, Rotation: 0.5, Thickness: 0.05 in, Triangle external shape with 0.01 in chamfer on the outer edge) to easily catch when screwing the cap on, 90° ± 10° rotation to open or close. Knurled cap and bottle cap diameter of 40 mm ± 2 mm to allow easy grip.

2. A simple tactile button (7.8mm x 7.8mm x 4.9mm) will be wired to just below the surface aligned with a hole in the bottle surface such that the button is level with the outside material to avoid accidental activation when bottle is dropped, knocked, etc. The button will have a light pressure threshold of 120 ± 40 grams-force. When activated, button will activate unlocking mechanism just as the software would under normal circumstances. The button will be low profile in its color and extrusion above the bottle surface (~0 mm)

3. An LED(3.5 mm x 3.5 mm) that is onboard the microcontroller will be embedded in the surface of the bottle. A hole in the bottle surface(3 mm x 3 mm) will be aligned with the onboard LED to allow light to be seen by the patient from the outside of the bottle. When addressed in software, the LED will illuminate at a brightness of 100 millicandelas (mcd) ± 20 mcd for clear visibility in dark environments. To denote prescription status 3 light settings will be written into software; Green: adequate time from prior dose(>6 hours depending on patient for Vicodin). Yellow: less than adequate time from prior dose but still greater than threshold time for dose(4<hours since dose<6) . Red: inadequate time from prior dose(<4 hours).

4. The companion app software will be compatible with Android and iPhone operating systems. Included in the IFU will be direction for downloading the accompanying app from the associated app store. The app will feature a “first time start-up” page that will clearly ask the patient to enter their name, and number of pills in prescription. The apps landing page will include a simple UI; minimal amount of buttons/words, large buttons greater than or equal to 200x200 pixels), visual feedback cues that tell patients a response has been

recorded, button shrinks when tapped or prompt box leaves, or message that response has been recorded. There will be 3 page tabs accessible to the patient at the bottom of the app GUI: "Dosage History" (live queue of most recent 5 doses with date and size), "Take Dose" (or home page), and "Information" (includes brief information on Hydrocodone and its risks and benefits if there are any as well as a reset button to restart the prescription cycle).

5. On accompanying app "Take Dose" page, the launch screen when users access the app, will contain a large central button (450 x 450 pixels for 300 PPI screen) that says, "Take Dose". If software concludes patient can take dose based on previous dosage time and the release time of Hydrocodone, 3 large buttons (300x300 pixels) will appear prompting the user to input their current pain level. The buttons will be illustrated with high contrast colors from left to right: frowning face indicating high pain, neutral face indicating medium pain, and a smiling face indicating low-no pain. Once patient inputs pain level, software algorithm that considers patients' current pain, and Hydrocodone pharmacokinetic markers will determine and then display the number of pills the patient should retrieve from the compartment in the device. Meanwhile the microcontroller will be pinged to activate the solenoid to unlock the compartment allowing the patient to grab specified pills.

6. A 20 mm long push-pull solenoid will be mounted to the inside of the main bottle body (below where the cap and bottle meet) and wired to the microcontroller. When addressed in software, the solenoid will be receiving LOW voltage, and the solenoid slug will extrude through a 8 mm x 6 mm rivet in the side of the cap Once activated via software or the override button, the solenoid will receive HIGH voltage (5-6 V DC) causing the pin to retract back through the rivet on the side of the cap allowing it to be unthreaded and accessed.

7. The microcontroller housing will feature a micro-USB port hole of dimensions ~8.4±0.5 mm x 3±0.5 mm ensuring the connected micro-USB port connected on the microcontroller can slide into hole allowing patients to access the charging port. The port hole should lay exactly flush to the bottle surface to ensure cord clearance of the micro-USB cord included with the device does not interfere with access to the charging port. The microcontroller will have an attached 3.7 V Lithium polymer (LiPo) battery that will be recharged via a built-in charging circuit in the microcontroller that is in series with the micro-USB port.

8. The device includes a 13-dram (equal to 1.6 fluid ounces thus full size is ~47 mL = ~47±8 cm^3) sized pill compartment. Size of single Vicodin is 14 mm long oval shape, assuming a width of 7 mm and

a height of 5 mm, $V = \frac{4}{3}\pi(0.7)(0.35)(0.25) = 0.19\ cm^3$. Taking into accound unexact pill dimensions and packing into bottle, we estimate being able to hold a max of 80±10 Vicodin pills.

Identify safety considerations related to these user/patient interfaces.

1. Cap ridges must avoid being too sharp to pierce a patients' skin when the force required to unscrew the cap is applied. Tops of each ridge will be flush and avoid sharp angles. The cap's design should be such that it does not require fine motor skills, considering the potential lack of dexterity in patients. It must be easy to grip and manipulate. Non-slip ridges such as those outlined in above design solution.
2. The tactile button for emergency access must be designed to prevent accidental activation while still being accessible in emergencies. The force required to press it should consider the potential weakness or lack of dexterity in patients due to pain or other conditions.
3. Choose high-contrast LEDs in the proper luminous intensity range ensuring the LED isn't too bright to cause injury to patients' eyes.
4. The app should be designed following accessibility guidelines such as WCAG and ADA compliancy to accommodate users with various disabilities.
5. Above compliancy pertains to patient pain input interface as well.
6. The locking mechanism must not have any exposed electrical components that could cause electrical shock to a patient. Solenoid should operate at voltage less than 50 V AC or 120 V DC under normal conditions to be safe from shock on skin contact. Solenoid casing should be vented to prevent overheating in extreme circumstances. Device should be tested according to 16 CFR 1700, in order to ensure device is child resistant.
7. The micro-USB hole in the casing should be precise enough to prevent movement and potential exposure of wiring that could shock a patient. The microcontroller should be properly ventilated for heat dissipation in case of extended time device is plugged in. Voltage protections will be integrated on the microcontroller, but this should be revisited at each step in design process.
8. Device must be able to store entire Vicodin prescription ~32 pills for 2-week refill period. This is lower than compartments max pill storage.

5.2.2 External interfacing devices, accessories, and materials

External interfacing devices, accessories, and materials that will be used with this product.

This device will interface with:

1. Cloud storage to store weight data

2. An online program to serve as the patient user interface.
3. The device will be directly touching the drug capsule.
4. An information pamphlet
5. An instructions document (IFU)

Design solutions that address these external interfaces.

1. The chosen microcontroller has Wifi capabilities to connect to the Adafruit IO server. Specifically, a 2.4 GHz wifi radio antenna to ensure connection with wide array of networks.
2. The programed software will be capable of interacting with both the physical device and the online system (computer, phone, etc)
3. The PEI material of the bottle isn't reactive with the gelatin capsules used in Vicodin.
4. Words on the information pamphlet and instructions sheet will be easy to read and understand.

Safety considerations related to these external interfaces.

1. An online interface will not work when the patient is not within WiFi range. This has been accounted for by the external button for manual override.
2. Different pills have different capsule materials, those materials in relation to our PEI material will need to be considered in the future.

5.2.3 Human factor considerations (Ergonomics)

- Proper use of the product involves the patient opening the bottle via the app, removing the recommended dose, then resealing the bottle, which will automatically lock.
- The patient is expected to give basic data such as dosage and pain level
- Issues might include:
  - Patient not closing the bottle completely, leading to the bottle not locking
  - Patient unable to open the bottle due to pain or loss of movement
  - Patient not providing enough data
  - The patient may not have access to their phone, and will need medication
- Solutions:
  - If bottle is not fully closed, the patient will receive a push notification from the app, along with a flashing LED on the bottle
  - The cap will be knurled and have 0.5mm bevels on the threads
  - The patient will be prompted to provide data before each dose, in the form of a quick 3-button survey on pain.
  - The bottle will feature a unlock button that is hardwired to release the lock, this will be reported on the app when used.

- Edges will be filleted or chamfered no less than .5 mm, and the cap edge will overhang the outer surface of the bottle such that pinching the patient is not possible.
- Electrical components are enclosed in a box on the side of the bottle, making them inaccessible to prevent shocking.

5.2.4 Device use in the intended environment

Reference environmental requirements that need to be addressed.

Device will be unpackaged and used in homes or personal spaces of patients:

1. Environment must have a power outlet capable of outputting at least 5 V and 500 mA.
2. Device must be able to withstand common environmental factors in homes such as dust, humidity, and changing temperatures.
3. Patients smartphone must be able to connect to a Wifi router where they plan to use the device to connect with pill bottle module.

List design solutions that address these requirements

1. Micro-USB adapter must be included that can convert power output to 5 V DC and various amperage outputs to match the devices charging requirement of 5 V DC.
2. The exterior of the pill bottle device will be 3D printed in polyetherimide (PEI). PEI has a transition temperature of ~390°C, thus it will not deform in the intended environment of use. Regarding humidity, PEI can absorb a level of humidity thus the device will include instruction to not keep in direct sunlight or near heat vents.
3. Our device will connect via WiFi module to the downloaded software on a patient’s phone in order to transmit data and store data.

Identify safety considerations related to operation in the intended environment

1. Voltage protections should be integrated in the circuitry of the device to ensure it does not blow a fuse or cause a surge when connected to power.
2. As stated, PEI absorbs moisture and thus should be kept away from heat sources to avoid degradation of the material.
3. Cloud storage of patient data must be secured via encryption in order to avoid a security breach. Adafruit IO uses unique keys to data feeds to allow for safe transfer and storage of patient data.

5.3 Risk (Hazard) Analysis Summary (Include key risks from FTA and FMEA)

5.3.1 Analyze and evaluate findings from FTA and FMEA and provide summary

- See Appendix A for fault tree Analysis (FTA) and Appendix B for Failure Modes and Effect Analysis (FEMA).
- Summary: Risk analysis and mitigation will be targeted and evaluated throughout the development process of this device, utilizing the ISO 14971 for safety standard applications for medical devices through risk management. Risk mitigations are to target user safety and preventative risk management to reduce the events of harm. The FTA for this device was developed through logic events and Boolean gates to examine the potential routes of fault. Highest RPN risks were the main focus when determining risk mitigation strategies to put patient safety above all else. The device can cause most harm to the patient through miscalculated dosages or by locking the patient out of the pill compartment. Thus special limits will be placed on the dosage size as well as any dose intervals to ensure the device will never tell the patient to take a dangerous dosage of hydrocodone (<30 mg Hydrocodone in a day, 6 Vicodin pills) during a day. Software logic and hardware timing will be intensely tested to ensure under any conditions the patient will be stuck in a “red light” stage. Regarding the locking mechanism, an emergency override button will allow patients to access the pill compartment regardless of the bottle light status.

5.3.2 Provide mitigation strategies to identified risks

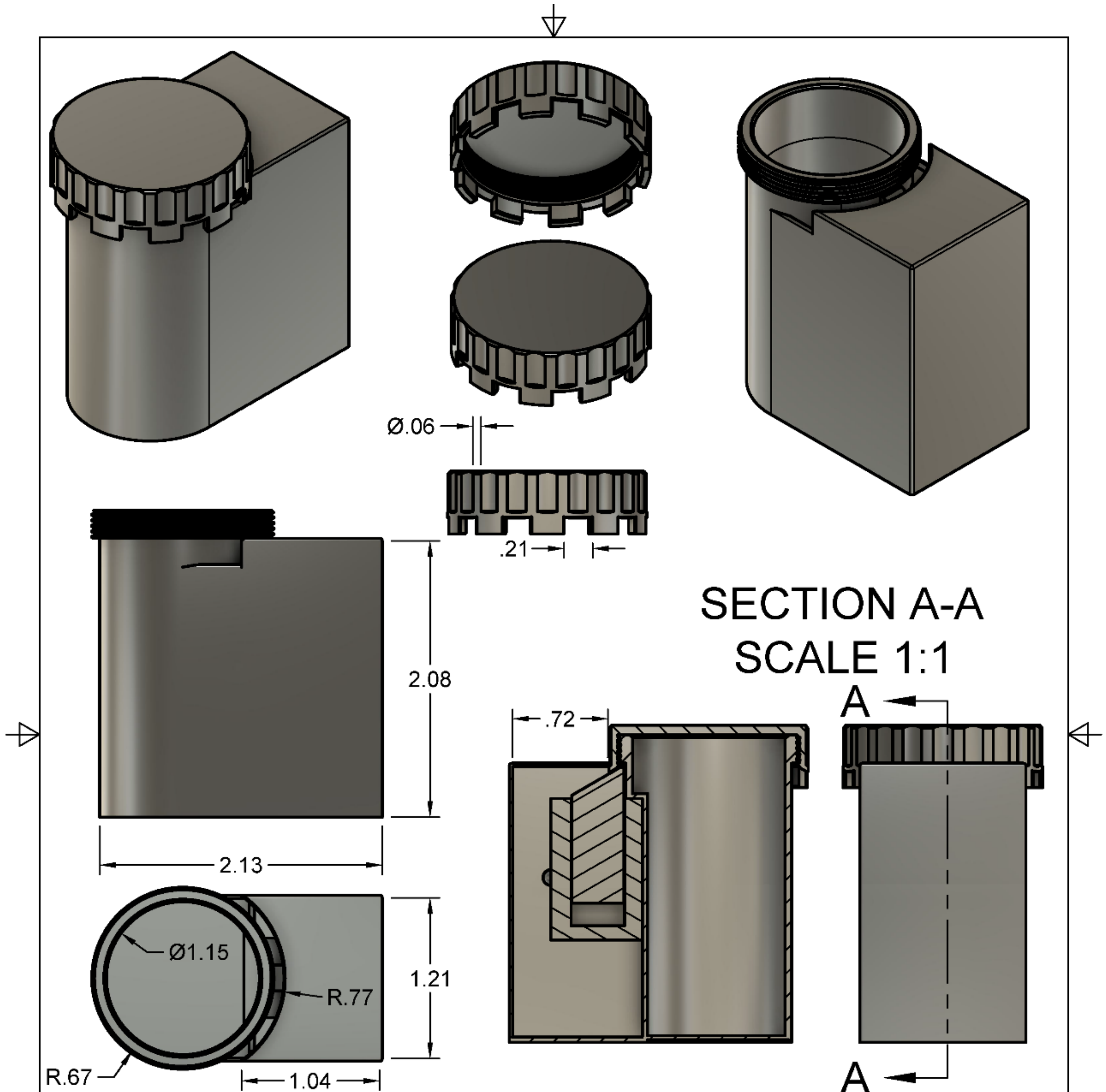
- Risk: App Software doesn’t properly calculate dose size yielding inaccurate dosages.
    - Implement software testing procedures, including unit testing for each function and integration testing for the complete system.
    - Include validation checks in software that confirms dose size against an expected range of dose. (0-3 pills)
- Risk: Electrical shock from solenoid pin when patient puts fingers in pill compartment
    - Due to battery powering of device ensure device follows standard IEC 60601 for Class 2 device.
    - Ensure metal conductive surfaces of solenoid and other wiring components are properly insulated and concealed from pill compartment.
    - Use low-voltage power systems that fall under Safety Extra Low Voltage (SELV) regulations to minimize the risk of electrical shock.
- Risk: Battery doesn’t provide enough current to each process (fails)
    - Implement a battery management system that monitors the battery's charge level and current delivery, alerting the user when the battery is low.
    - Design the electrical system to operate on minimal current and include a power-saving mode to extend battery life.

    - Include USB charging cable in packaging.
- Solenoid locking pin becomes jammed
    - Incorporate a mechanical override or manual release mechanism that allows the device to function or be safely opened in case the solenoid becomes jammed.

5.3.3 Summarize Residual Risk

ISO 14971:2019 7.4 (accessed 4/6/24) requires that all individual unacceptable residual risks be checked in a benefit-risk analysis, and because none of the residual risks of this device are unacceptable, this is unnecessary. Clause 8 requires an overall evaluation of residual risk. The main benefits of the device – better understanding of opioid dosing, far outweigh the above risks.

5.4 Mechanical Design Details



	PROJECT Pharmacokinetic Control of Opioid Dosing				
	TITLE DoseX				
APPROVED	SIZE	CODE	DWG NO		REV
CHECKED	A				1
DRAWN Daniel Candland 4/12/2024	SCALE 1:1		WEIGHT	SHEET 1/1	

5.5 Material Selection Details

The pill bottle and cap are made from Polyetherimide(PEI).

- PEI's tensile and yield strength are 114 MPA and 135 MPA, making it quite strong. Its dielectric strength is 25.6 kV/mm, making it an excellent insulator in the event of electrical shock.

- PEI has a softening point of 392°F, resistant to UV, extremely resistant to hydrolysis and non-hydroscopic, allowing for usage in multiple environments

Regarding the material interface:

- There are no known interactions between hydrocodone and PEI allowing for safe storage of the Vicodin.

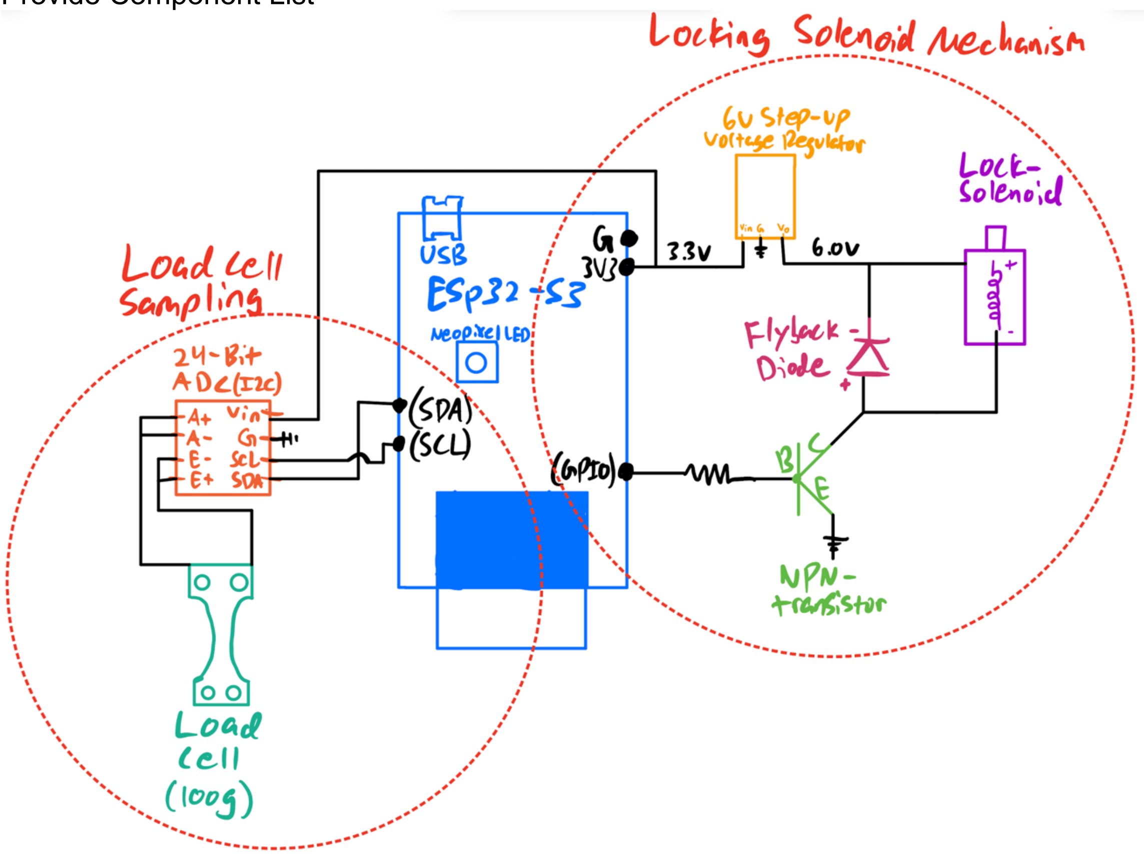
- PEI meets ISO 10993 certification criteria for biocompatibility and has a shelf life of ~1 year if kept at 15-25C. The solenoid and load cell will have their casings made of stainless steel(yield strength roughly 205 MPa and tensile strength is at least 515 MPa).

- There are also no known interactions between stainless steel and PEI, preventing self-degradation of the device

5.6 Electrical Design Details

- Electrical block diagrams
- Electrical wiring diagrams
- Electrical circuit schematic
- PCB layouts and mechanical drawings (as appropriate)

- Provide Component List



Component	Product	Important Specifications
Microcontroller	Espressif ESP32-S3-DevKitC-1-N (Adafruit ID: 5336)	• Dual-core Xtensa LX7 (240 MHz, 32-bit architecture) • Integrated Floating-Point Unit (FPU) • 8 MB Flash • 8 MB PSRAM • WiFi 4 (802.11 b/g/n) • Supports 2.4 GHz WiFi frequency • Operating voltage: 3.0 V - 3.6 V • Built-in LDO regulator • 500 mA peak current output from onboard voltage regulator

		• 45 GPIO pins (multipurpose for ADC, DAC, PWM, UART, SPI, I2C, etc.) • Dedicated USB OTG controller with native USB 1.1 support • RTC (Real-Time Clock) • I2C: Up to 2 buses • 2 mounting holes • -40°C to 85°C • Onboard addressable RGB NeoPixel LED • JST connector for LiPo battery (charging circuit included)
Push-Pull Solen	Adafruit Small Lock-Style Solenoi (Adafruit ID:5135)	• 20 mm long body • Activated with 5 V DC • Current(at 5 VDC): 1.1 A • Weight: 12.6g • Throw(at DC 5 V): 3 mm
Load Cell	Single Point Load Cell-100g (Phidge ID:3139_0)	• Maximum weight capacity: 100g • Operating Range: 3 V – 12 V • Aluminum Alloy • 4 mounting holes • 52 mm x 12.7 mm x 12.7 mm • 4-wire configuration •
LED	Adafruit NeoPixel RGBW LED w/ Integrated Driver Chip (**Adafruit ID:** 2759)	• Single LED: 5mm x 5mm • 4 pins: VDD, DOUT, GND, DIN • Power supply voltage: 3.5-5.5 V Working temperature: -40°C

		85° C • Red: 320-580 mcd, Green: 8 1275 mcd, Blue: 160-321 mc
Tactile Button	Adafruit Soft Tactile Button (8mm) (Adafruit ID: 3101)	• Dimensions: 7.8 mm x 7.8 mm x 4.9 mm • Weight: 0.1 g • DC 12 V 50 mA • Operating force: 120 ± 40 gf • Life: 50000 cycles Min.
Lithium Battery	Lithium Ion Polymer Battery – 3.7 V 1200 mAh (Adafruit ID: 258)	• Weight: 8.8g • Size: 34mm x 62 mm x 5 mm • Output: 1200 mAh at 3.7V nominal
ADC	Adafruit NAU7802 24-bit ADC (Adafruit ID: 4538)	• Dual input channel • 24-bit resolution • Various gain options 1-128 • Operating Range: 2.7 V – 5.5 V • I2C communication protocol • Built in offset and gain calibration
6V step-up volta regulator	6V Step-Up Voltage Regulator U3V16F6 (Polulu ID: 4942)	• Output Voltage: 6 V • Max input current: 1.6 A • Min input voltage: 2.7 V startup
Flyback Diode	1N4001 Diode (Adafruit ID: 755)	• 0.7 V drop • Pass up to 1 A • Protect against up to 50 V reverse voltage
NPN Transistor	ZTX851STZ (DigiKey: ZTX851SCT-	• Ic max: 5 A

	ND - Cut Tape (CT))	• Vce(max): 150 mV @ 200 mA(Ib), 5 A(Ic)

- Electrical power analysis
  - Espressif ESP32-S3-DevKitC-1-N8R8:
    - **Operating voltage:** 3.3V
    - **Current Consumption:** Active mode 40 mA, during data transmission through WiFi module ~250 mA (once per hour at max)
    - **Estimated Power Draw:** (0.04 A x 3.3V x 59/60) + (0.25 A x 3.3 V x 1/60) = 0.132 W + 0.0275 W = 0.1595 W
  - Solenoid:
    - **Operating voltage:** 5V
    - **Current Consumption:** When activated, 1 A
    - **Estimated Power Draw:** 5 V x 1 A = 5 W when active (60 seconds every 4 hours). 5 W x (60 s/(4 hours×3600 s/hour)) = 0.0207 W
  - NeoPixel RBGW LED:
    - **Operating voltage:** 5V
    - **Current Consumption:** At max brightness, 60 mA
    - **Estimated Power Draw:** 5V x 0.06 A = 0.3 W for continuous operation
  - Load Cell:
    - **Operating voltage:** 3.3 V – 12 V
    - **Current Consumption:** ~10 mA
    - **Estimated Power Draw:** 3.7 V x 0.01 A=0.037 W for continuous operation.
  - ADC
    - **Operating Voltage:** 3.3 V
    - **Current Consumption:** ~400 uA during operation
    - **Estimated Power Draw:** 3.3 V x 0.0004 A = 0.00132 W
  - **Total Power Consumption:**
    - **~0.5 W**

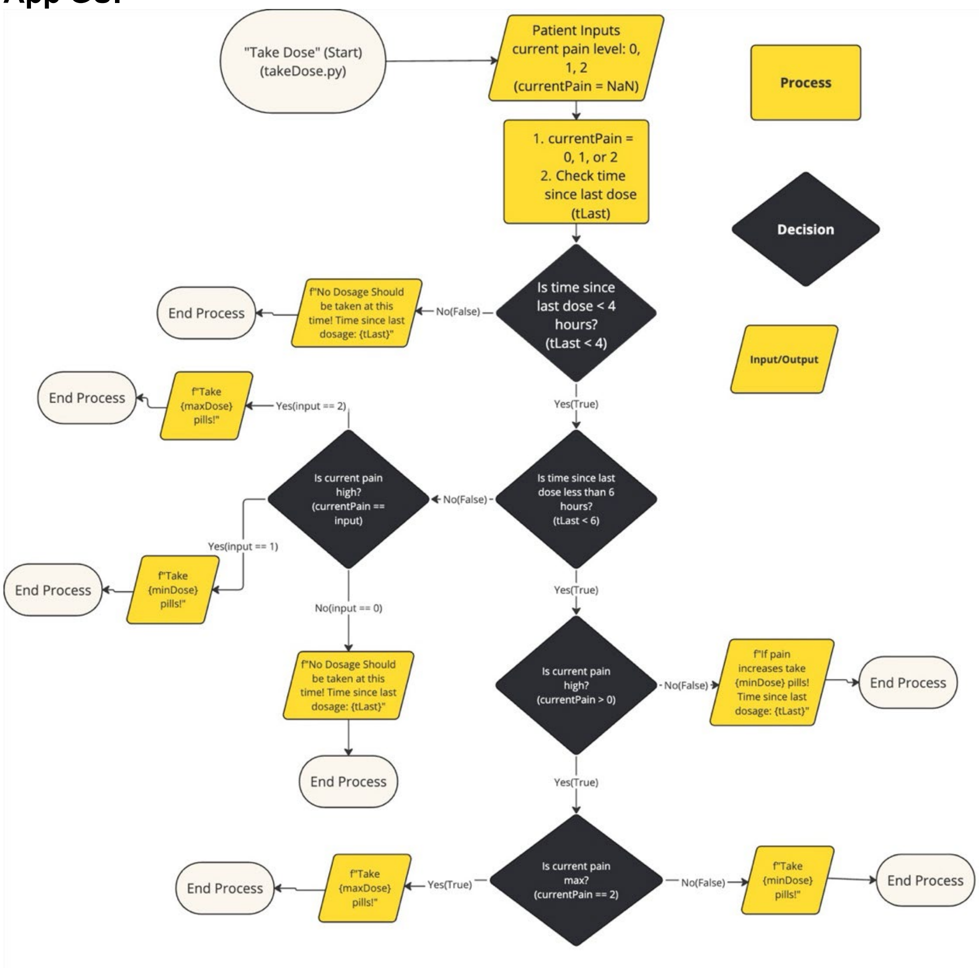
- Battery:
  - **Energy Capacity:** 1200(3.7 V/ 1000) = 4.44 Wh
- Battey Life:
  - 4.44 Wh / 0.5 W = ~ 9 hours
  - (in order to achieve a higher battery life closer to our specifications a 2200 mAh LiPo battery should be used)

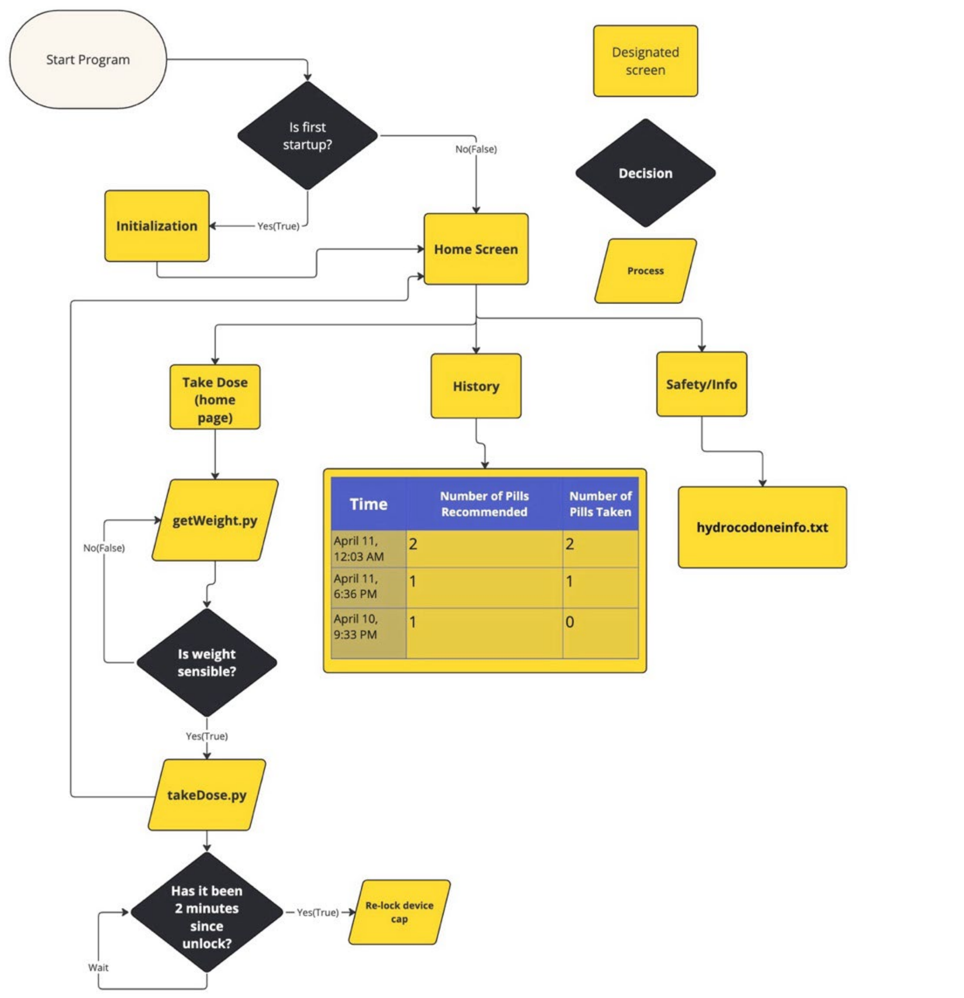
- Electrical Safety Considerations
  - Device is battery-powered and inherently has 2 layers of insulation between user and circuitry; bottle casing and battery insulation. Thus this device is classified as a **Class 2** device. (IEC 60601-1)
  - Relevant Standards(all accessed 4/10/24):
    - **IEC 60601-1**: The general standard for the safety and essential performance of medical electrical equipment, which includes guidelines on leakage currents.
    - **IEC 60601-1-2**: A collateral standard that covers electromagnetic disturbances—requirements and tests for medical electrical equipment and medical electrical systems regarding EMC.
    - **IEC 60950-1**: Information technology equipment; general requirements, which can be relevant for non-medical components like the WiFi module.
  - Electrical Leakage Current Protections
    - Ensure all conductive parts a user may interact with are properly insulated, namely the wiring for the solenoid locking mechanism.
    - Design circuitry to operate at Safety Extra Low Voltages (SELV), which is typically considered safe to touch and greatly reduce the risk of current leakage.
    - Perform current testing per standards laid out in IEC 60601.1
  - Electromagnetic compatibility/interference (emc/emi)
    - Use components that comply with EMC standards, which include shields for high frequency components such as the WiFi module and the solenoid to minimize emissions that could affect other components.
    - Incorporate filters and surge protectors to mitigate EMI on both power and signal transmission lines.
    - Ensure device circuitry adheres to EMC standards outlined in IEC 60601.1.2.

5.7 Software Design Details

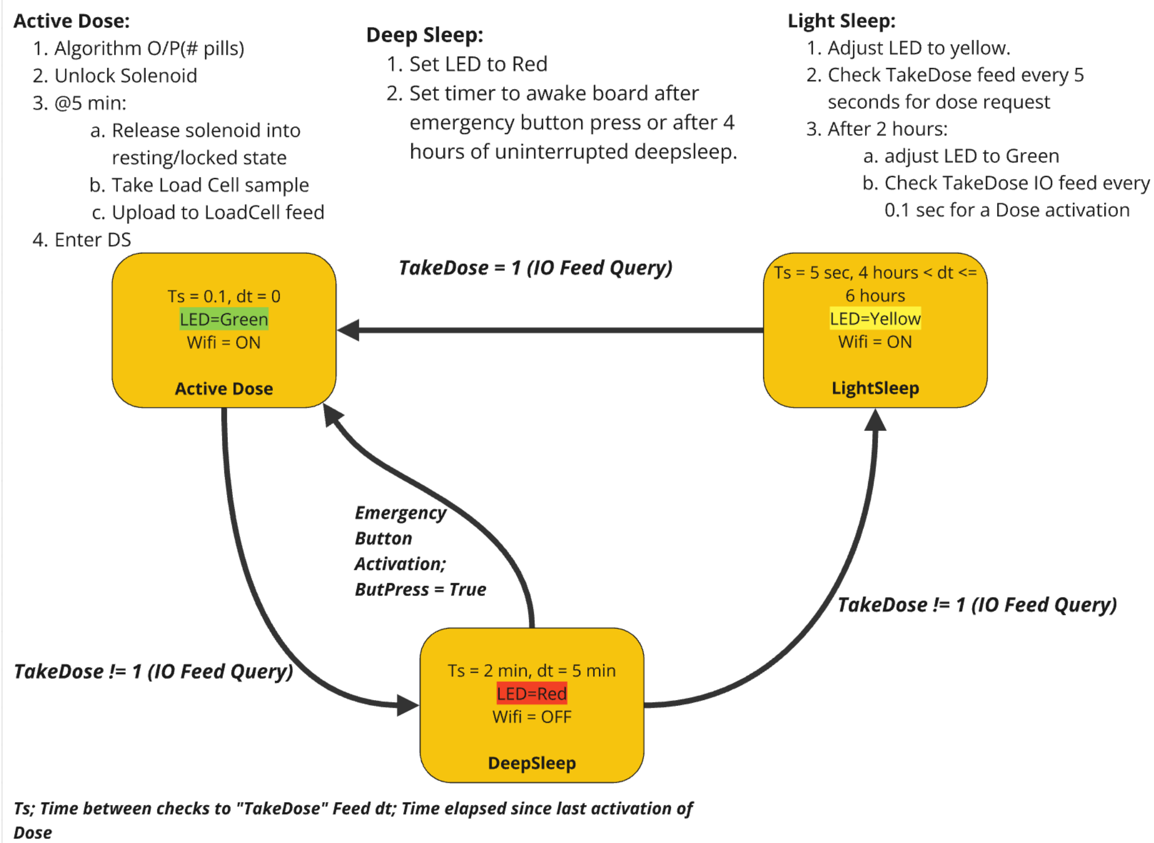
- High-level software architecture (use flowcharts, algorithms, or storyboards to show structure)

App GUI




Start Program
Is first startup?
No(False)
Yes(True)
Initialization
Home Screen
Designated screen
Decision
Process
Take Dose (home page)
History
Safety/Info
getWeight.py
No(False)
Is weight sensible?
Yes(True)
takeDose.py
Has it been 2 minutes since unlock?
Yes(True)
Re-lock device cap
Wait
Time
Number of Pills Recommended
Number of Pills Taken
April 11, 12:03 AM
2
2
April 11, 6:36 PM
1
1
April 10, 9:33 PM
1
0
hydrocodoneinfo.txt

Hardware Code State Diagram



- Address software development methods (e.g. Programming languages, hardware environment, operating system, etc.)
    - Hardware will be written in circuitPy (circuit Python libraries), utilizing the Thonny IDE.
    - Hardware will be flashed onto an ESP32-S3 microcontroller to address and control each of the connected devices.
    - Software will be written in React Native for easy cross-compatibility between IOS and Android operating systems.
    - Data storage of time and patients pain inputs will consist of Adafruit's IO cloud data service for Wifi data transfer between device and cloud, and app and cloud.
- Other considerations including key variables (and type), memory usage, cycle times, sampling rates, etc.

Object Address (type)	Description
tLast (float – 32 bits)	Counter since patients last activation TakeDose state, resets on each occurrence.

currentPain (int – 32 bits)	Holds patient inputted pain level. Can take on integers 0, 1, or 2
fourToSix (Boolean – 1 bit)	Boolean flag, initialized to False, to ch for whether patient is in-between pharmacokinetic time range of 4 to 6 hours, set to True if 4<tLast<6
minDose (int – 32 bits)	For a given patient the higher accepta dose i.e 2 pills over 1. (may be adjuste future)
maxDose (int – 32 bits)	For a given patient the lower acceptab dose i.e 1 pills over 2. (may be adjuste future)

- The state diagram contains sampling rates to the IO server in each of the designated states as well as information pertaining to the wifi radio state.
- Microcontroller has limited flash memory and RAM, thus hardware code must be optimized for memory space and use proper garbage collection. Flash: 8 MB, PSRAM: 8 MB
- Cycle Time**:** Microcontroller clocked at 48 MHz, based on sampling rate.

5.8 Cleaning, Disinfection, and/or Sterilization Implementation

5.8.1 Pyrogenicity Requirements and how to achieve requirement

There are no known cases of polyetherimide causing pyrogenicity. However, the load cell and the solenoid will contain stainless steel which is known to be pyrogenicity. The design will avoid pyrogenicity by ingestion by meeting these conditions by:

- Keeping the load cell and solenoid on the exterior surface of the bottle
- Keep the Vicodin capsules on the interior of the bottle to minimize interactions with metal

5.8.2 Sterilization (SAL, method, validation, packaging restrictions, material restrictions)

Our device will not need sterilizing, however if in specific circumstances sterilization would be necessary, PEI can be sterilized while retaining its physical and chemical properties through various methods, each accompanied by a verification process:

- Ethylene oxide- residual gas testing is performed to ensure that no hazardous levels of the sterilant gas remain on the surface.
- Steam- Monitor temperature and pressure inside the autoclave to verify sterilization conditions.
- Autoclave- typically around 121°C at 15 psi for 15-20 minutes

- High-energy radiation(gamma and e-beam)- Measure radiation dosage with dosimeters to ensure effective sterilization.

The packaging layout is as follows:

- The packing is a 4.5cm x 4.5cm x 8.5cm rectangular box
- There is a polyurethane mold inside to securely hold and cushion the device.
- Consists of 3 layers. The middle layer will be made of cardboard, providing support and protection during shipping. The exterior is covered with polyethylene plastic film to prevent scratches. The interior will be made of polyurethane which also acts as a cushion for the device

5.8.3 Chemical compatibility (cleaning agents, sterilization, lubricants, disinfectants, etc.)

In order to clean and disinfect the device regularly the device should:

- Not be disinfected with strong acids and bases(both weak and strong) as they can degrade PEI.
- Be treated with safe disinfectants including isopropyl alcohol or ethanol solutions
- Not be treated with strong oxidizing agents such as bleach as they degrade PEI.

5.9 Shipping/storage conditions

5.9.1 Environmental and physical conditions during shipping

The Product will be shipped by cargo ship, train, plane, and truck leading to a wide variety of conditions. These can be specified as follows:

- Temperature ranges from –20 degrees to 50 degrees Celsius.
- Humidity ranging from 0 to 100%.
- Vibrations up to 100 Hz. Usually in the form of bumps rather than constant jostling.
- Non-sterile environment during transport.
- Broad range of pressures: 200-800 mmHg

5.9.2 Environmental and physical conditions for storage

The pill bottle device will be stored in packaging at a pharmacy prior to use, then stored in an outpatient space. When packaged and unpackaged, device may be stored in non-sterile areas. Pharmacies, according to ANSI 170-2017 “Medication Rooms” must be held to a max humidity of 60%, and a temperature range of 21-24°C. However, patient personal spaces will not be temperature controlled. Pill bottle device should be stored out of direct sunlight, and at 40-60% battery charge as battery is the main limit on shelf life.

5.9.3 Shelf life (time from manufacture to use)

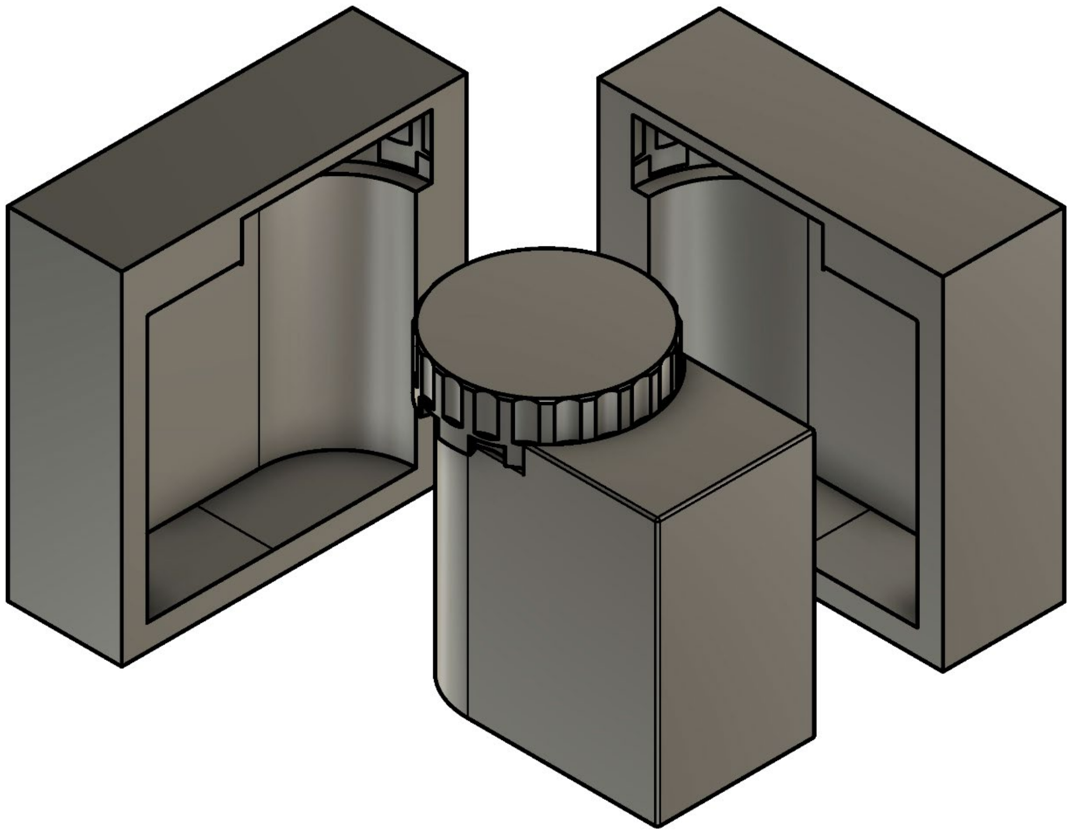
The shelf life of the pill bottle device will be primarily influenced by the battery's lifespan and the durability of the electronic components when not in use. Given the device is made from PEI and uses a 3.7 LiPo battery, the shelf life of our device will be approximately 2 years.

5.9.4 How will the design meet these conditions?

- The device should be made from a 3D printable material that can withstand temperature extremes and humidity, protecting the internal components from environmental conditions, Polyetherimide (PEI) filament.
- Design internal mountings for the electronic components to reduce the impact of vibrations and shocks during transport.
- Include a battery management system that can maintain the charge level within the optimal range during storage and incorporate a power-down or sleep mode to conserve battery life.
- Provide instructions for the end-user on how to store the device properly when not in use, emphasizing the need to avoid extreme temperatures, direct sunlight, and to maintain battery charge levels.
- Design packaging with protective cushioning to shield the device from physical shocks and with barriers to moisture and sunlight. Packaging should also be tamper-evident to ensure the integrity of the device up to the point of use.
- Packaging design will abide by packaging regulations as stated by the FDA CFR title 21 section 820.130 (accessed 4/3/24)

5.10 Packaging and Labeling (P&L)

5.10.1 Provide packaging design



5.10.2 Product label, package label, carton label (include art work)

5.10.3 Instruction for Use (IFU) (include draft document)

Instructions for use:

1. Download the DoseX app on your smartphone
2. Upon opening the app, you will be prompted to enter the device ID, which is found on the bottom of the bottle.
3. The app may take a moment to connect. Once connected, ensure the bottle is upright on a flat surface, allowing it to calibrate.

To open the bottle:

1. First, unlock via the app, by first entering pain level, and any adverse events. Then, a dosage will be suggested, and if accepted, the bottle will unlock.
2. Open bottle, remove suggested dosage, and reseal, ensuring the lock clicks into place.
3. After resealing, ensure the bottle is placed upright on a flat surface.
4. The app will then display the amount of time until your next recommended dose.

To view dose and pain history, open the history log in the app, which records this information for you and your provider to view. You can also report and view any adverse events such as side effects, changes in symptoms, or unrelated injuries or events that may affect your treatment.

If the app is not working or unavailable, the bottle can be unlocked via the emergency unlock button, hold down the button for 3 seconds, and if it has been enough time since your previous dose, the bottle will open, however note that use of the emergency unlock will be recorded in the history log.

The bottle has a battery life of around 36 hours of active use, and because the bottle will only be actively used for a few minutes at a time, it is unlikely you will need to recharge it often. If the DoseX bottle needs to be charged however, a red indicator light will begin to flash at intervals, indicating it needs to be charged. In this case, use the provided USB charging cord to plug the bottle in until the indicator light turns solid red, indicating a full charge.

5.10.4 Disposal / recycling instructions

- Disconnect any power sources and remove batteries and place the device in a sturdy trash bag to prevent any leakage or damage. The packaging can simply be recycled

5.10.5 Other P&L considerations (e.g. web site, advertising, etc.)

- Logos will be prominently displayed on the packaging to enhance brand recognition.

- QR codes will be displayed on the packaging to provide users a link to instructional material on how to use the device.

5.11 Using a Traceability Matrix, demonstrate how the device meets Design Requirements

Table: Specifications to Requirements Traceability Matrix

Outline #	#	Design Solution	Humanity factors and Usibility Eng.	Mech. Design	Material Selection	Elec. Design	Software Design	Cleaning/ Sterilization	Shipping / Storage	Packaging / Labeling
#	Spec. / Req.	4.1	4.2	4.4	4.5	4.6	4.7	4.8	4.9	4.10
5.1	Device will provide proper dosage recommendations	X					X			
5.1	Device will accept patient input	X	X				X			
5.1	Device will have a secure lid which can unlock and lock	X		X						
5.1	Device will be portable and rechargeable	X		X		X			X	X
5.8	Device will not produce any unnecessary risk to the surrounding environment				X			X	X	X
5.9	Device will be an affordable price compared to competitors.				X	X				
5.11	Device will accurately calculate its weight and translate into the number of pills.			X		X	X			
5.12	Device will be able to withstand basic stress levels without deforming.				X					
5.12	Device will be able to hold entire prescription.		X	X						
5.12	Device will transmit weight and time data to cloud storage.					X	X			
5.12	Device will have mechanical override to unlock without software.			X		X				
5.12	Device will avoid sharp edges or exposed electrical components.		X	X						
5.12	Device will have a battery capacity		X			X			X	

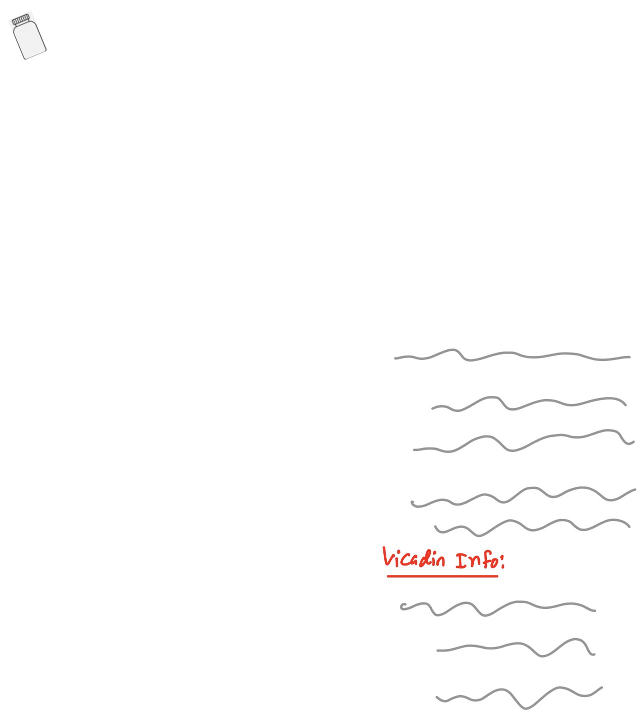
	of at least 36 hours.									
5.13	Instructions (IFU) will be provided.		X							X
5.13	Labels will be legible.									X
5.13	Device will be useable by those with mobility issues in their hands.		X	X						
5.13	Accompanying app will have start-up process for first time users.						X			
5.13	Each device will have a unique number printed or embossed on it.									X
5.13	Information pamphlet will include basic information on the devices purpose and use.		X						X	X

5.11.1 Evaluate how complete the matrix is (especially comment on requirements that are not being fulfilled)

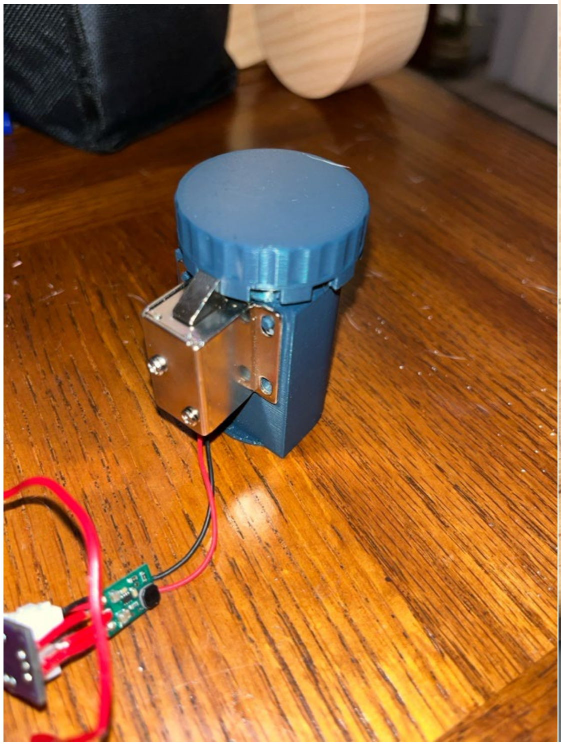
The matrix is mostly complete. The previously stated design requirements are properly considered most of which in multiple sections. However, the requirement “Device will be useable by those with mobility issues in their hands” was not completely met. We were unable to design a lid system which was easily accessible by those with damaged hands and followed the other design requirements.

5.12 Conceptual Prototypes

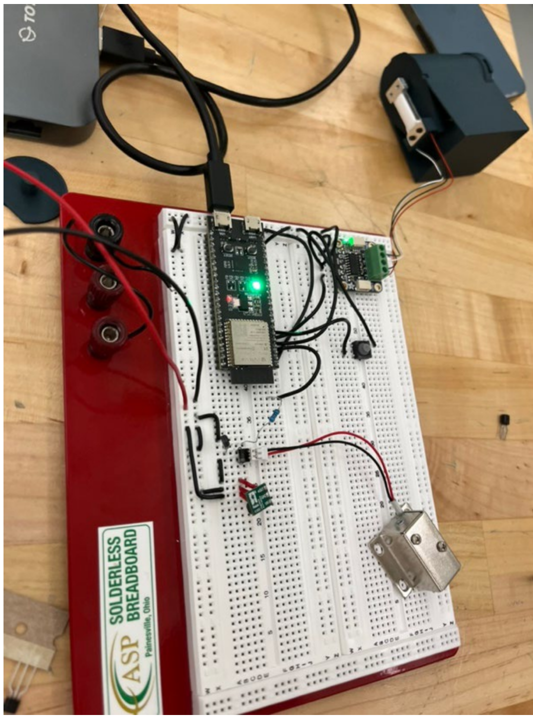
App Concept Art:

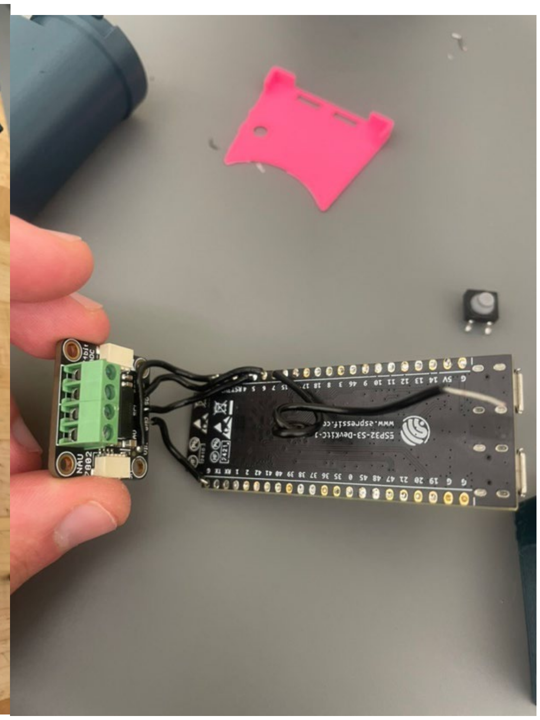


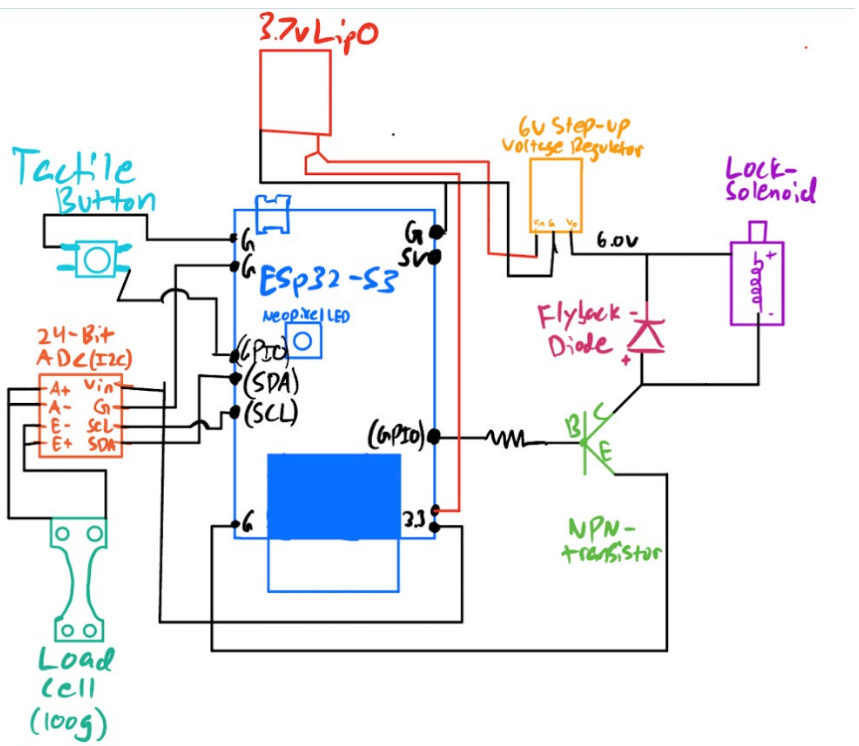
Pill Bottle Mechanical Prototype:



Circuit Prototype:







Design Analysis:

- Mechanical prototype was developed using Fusion 360 for mechanical CAD drawings then printed in PLA on a Prusa MK4 3D printer. We learned from prototype:
    - 13-dram pill compartment size must be revisited to ensure it can hold Vicodin prescription with added load cell plate.
    - 20-30% infill is an appropriate infill for device intended uses.
    - Wiring of components will be complex, ensuring wire routing is sensible will be important in design phase.
    - Cap threading should be revisited; prototype threading is too small to properly thread as needed.
    - Cap design allows ample grip.
- Electrical Prototype consisted of wiring/block diagrams in **5.6 Electrical Design Details.**
    - Proper circuitry prototype should be a wired breadboard with components.
    - Microcontroller can handle all wired accessories in device
    - A 3.7 2200 mAh LiPo battery should be able to power all processes for intended power duration
- Software prototype consisted of App software mock drawings that outline general app features.

6 Content that may be addressed (optional)

6.1 Cost Constraints- preliminary manufacturing cost estimates

6.2 Preliminary Freedom to Operate, Patentability, & Infringement Review

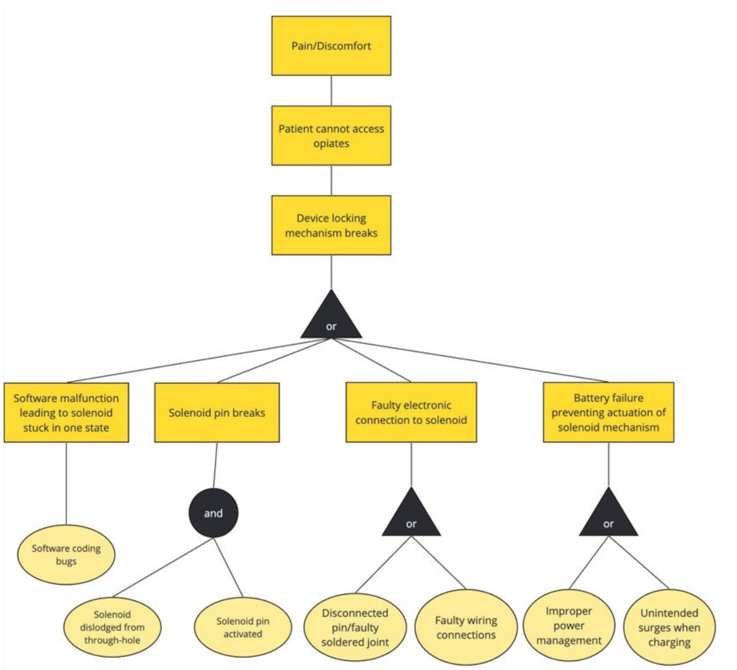
6.3 Service and Repair

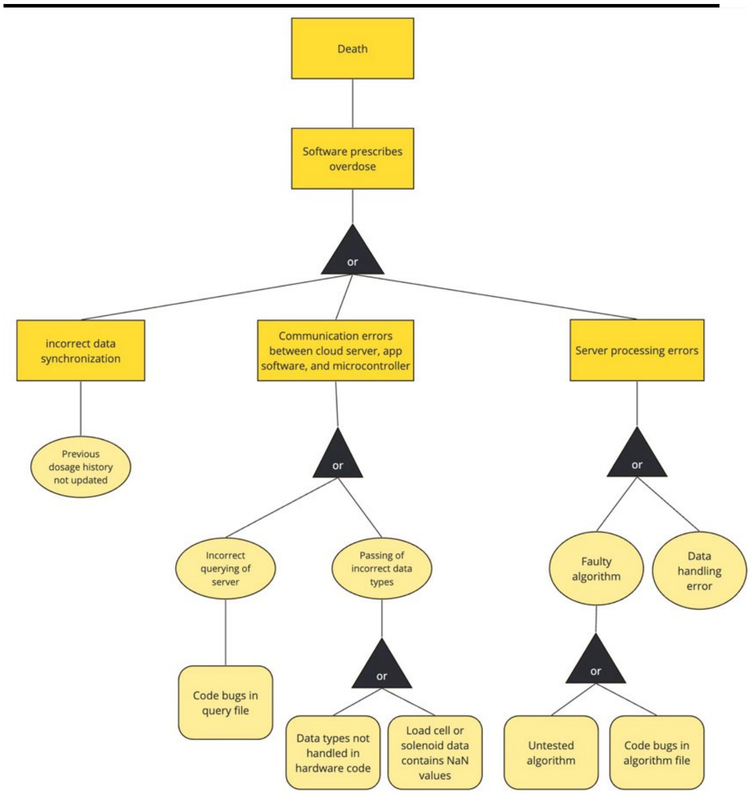
6.4 Preliminary Build Plan

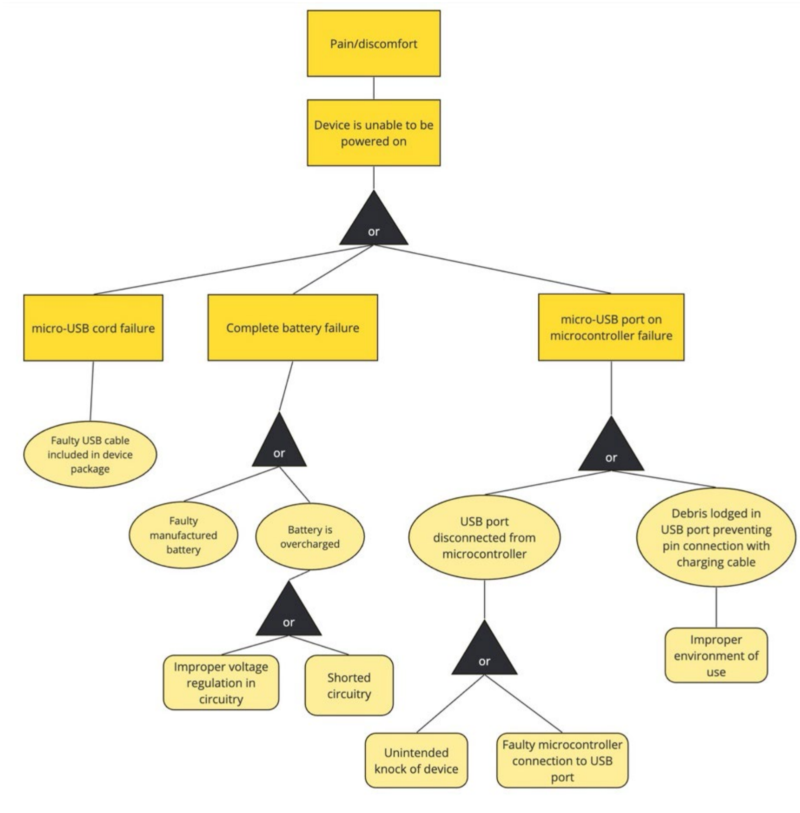
6.5 Preliminary Manufacturing Operations Sequence (to evolve into manufacturing SOP’s)

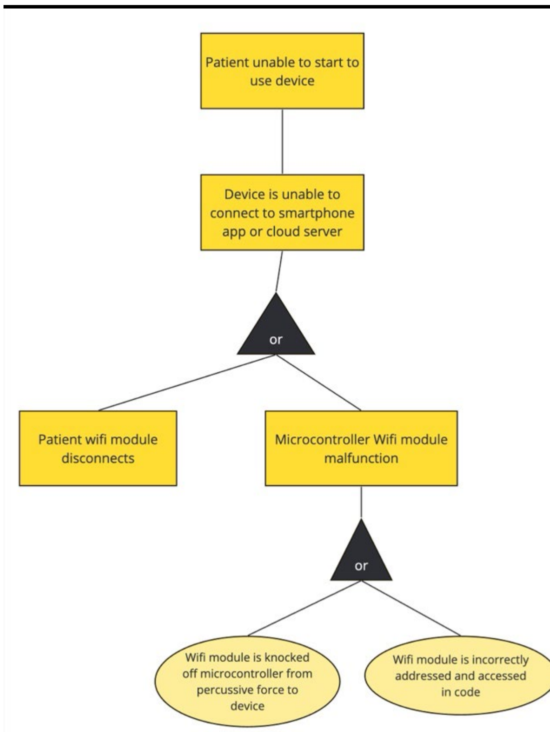
6.6 Process Capability

Appendix A: FTA Trees


Pain/Discomfort
Patient cannot access opiates
Device locking mechanism breaks
or
Software malfunction leading to solenoid stuck in one state
Solenoid pin breaks
Faulty electronic connection to solenoid
Battery failure preventing actuation of solenoid mechanism
Software coding bugs
and
Solenoid dislodged from through-hole
Solenoid pin activated
or
Disconnected pin/faulty soldered joint
Faulty wiring connections
or
Improper power management
Unintended surges when charging


Death
Software prescribes overdose
or
incorrect data synchronization
Communication errors between cloud server, app software, and microcontroller
Server processing errors
Previous dosage history not updated
or
Incorrect querying of server
Passing of incorrect data types
Code bugs in query file
or
Data types not handled in hardware code
Load cell or solenoid data contains NaN values
or
Faulty algorithm
Data handling error
or
Untested algorithm
Code bugs in algorithm file





Appendix B: FMEA Table

Process Step/Input	Potential Failure Mode	Potential Failure Effects	SEVERITY (1 - 10)	Potential Causes	OCCURRENCE (1 - 10)	Current Controls		RPN	Risk Mitigation Action	SEVERITY (1 - 10)	OCCURRENCE (1 - 10)	RPN
What is the process step, change or feature under investigation?	In what ways could the step, change or feature go wrong?	What is the impact on the customer if this failure is not prevented or corrected?		What causes the step, change or feature to go wrong? (how could it occur?)		What controls exist that either prevent or detect the failure?			What are the recommended actions for reducing the occurrence of the cause or improving detection?			
App Software	Fails to connect to patients pill bottle device	Patient cannot use accompanying app	***1***	Bad patient Wifi router, bad Wifi radio on microprocessor.	***6***	Ensuring app software properly connects with Wifi module. Ensuring patient is aware they need a Wifi connection		***6***	Include in IFU that patient must have Wifi and how to connect device to app. Test microcontroller Wifi module function.	***1***	***1***	***1***
App Software	Incorrectly queries data from server	Innacurate load cell and dosage data leading to innacurate doses for patient.	4	Code bugs in query code in app software.	6	Ensuring app software code is bug free and samples sensors from IO server correctly.		***24***	Test app software connection with IO server and accuracy of data queries	4	1	***4***
App Software	Doesn't properly calculate dose size.	Patient given inaccurate Doses	3	Dosage algorithm fails; inaccurate weights on each factor, or breaks when running resulting in 0 or NaN type.	4	Testing algorithm and adjusting weights to get validated dose size outputs in variety of scenarios.		***12***	Test algorithm through simulated patient chronic opiate prescription.	3	3	***9***
App Software	Allows “Take Dose” action to be performed within 4 hours of previous dose.	Patient given unsafe amounts of drug.	8	App software not properly timing periods between doses.	3	App software unit testing. App software/microcontroller code black box testing.		***24***	Proper unit/bug testing of software and validation of input/output testing.	8	3	***24***
Microcontroller: Wifi Module	Fails connection with IO server.	Patient cannot use app or device.	3	Wifi Module disconnected from microcontroller.	1	Testing device connection prior to packaging.		***3***	Test microcontroller Wifi module function. Add reconnect protocols for accidental disconnects.	3	1	***3***
Microcontroller: Charging Circuit	Fails to properly manage battery charging.	Undercharged device, or overcharged device.	2	Charging circuit fails to charge and recharge connected battery.	6	Proper power analysis and testing of battery life over time.		***12***	Conduct proper power analysis of device over functionality and time.	3	2	***6***
Microcontroller: Power Management	Fails to properly distribute battery power across sensors and processes.	Undercharged but still recharges.	2	Incorrect power management in circuitry of connected electronics in device.	3	Proper power analysis and testing of battery life over time.		***6***	Conduct proper power analysis of device over functionality and time.	2	3	***6***

Microcontroller: micro-USB Port	Fails to accept charge from USB cable	Device never recharges. Patient cannot use device.	5	USB port dislodges from microcontroller connection or becomes faulty from port lodged debris	4	Ensuring device connects with USB cable prior to packaging.		***20***	Test device to ensure stable and repeatable connection to micro-USB port.	4	2	***8***
Battery	Doesn't provide enough current to each process (fails)	Patient cannot use device.	5	Incorrect power management in circuitry of connected electronics in device.	7	Proper power analysis and testing of battery life over time.		***35***	Conduct proper power analysis of device over functionality and time.	5	3	***15***
Hardware Code	Doesn't sample load cell at correct time (synchronization error)	Patient given inaccurate count of remaining pills	2	Untested hardware code leading to bugs and loss of function.	8	Testing load cell output with different levels of pills		***16***	Testing load cell output when pill compartment is filled then slowly decreased by 1-4 pills at a time.	2	4	***8***
Hardware Code	Fails to unlock solenoid gate upon request. (logic error)	Patient cannot access pill compartment	6	Untested hardware code leading to bugs and loss of function.	8	Testing solenoid locking mechanism.		***48***	Test lock/unlock mechanism activated from app software for many more cycles than expected from patient use.	6	2	***12***
Load Cell	Accuracy of load cell is effected over time: Drift	Patient given inaccurate count of remaining pills	2	Load cell accuracy not gauged over long periods of time.	8	Testing load cell accuracy over long time intervals.		***16***	Test load cell over time period greater than a full chronic opiate prescription (>2 weeks- 1 month)	2	5	***10***
Load Cell	Inaccurate data output.	Patient given inaccurate count of remaining pills. Or no count output whatsoever.	2	Inaccurate load cell calibration. 24-bit ADC inaccurately divides voltage. Load cell loaded to greater than 100 g of pills causing break in resistances.	5	Ensure generated calibration curve used is accurate. Ensure capacity of pill storage cannot fit >100g of pills.(~307 pills)		***10***	Test load cell at a range of values both within expected range and outside expected range after calibration.	2	1	***2***
Lid and Locking Mechanism	Lid can be forced off when locked	Patient has access to pills when locked	5	Flexing or damage to the lid or bottle	5	Ensure components involved in locking the bottle are protected		***25***	Test lid and locking mechanism and redesign if necessary	5	1	***5***

Push-Pull Solenoid	Solenoid pin becomes jammed.	Patient either unable to lock or unlock pill storage	5	Inaccurate alignment of solenoid pin with cap hole.	5	Determine how solenoid pin could be dislodged or voltage step-up connection fails.		***25***	Drop test device to ensure that once assembled, solenoid pin cannot be dislodged and that electrical connections remain in place.	5	2	***10***
Emergency Override Button	Failure to register presses	Patient could lose access to pill storage if connection to server or app fails.	7	Faulty electrical connection with microcontroller.	3	Determine where button wiring could rub against device causing a disconnect.		***21***	Drop test device to ensure that once assembled, electrical connections from button remain in place. Ensure wiring has appropriate path to microcontroller.	7	1	***7***
LED Indicator Light	Doesn't update with new dose	Patient unable to see what "state" their prescription is in from device.	1	code logic used in hardware fails to switch status at given time intervals	6	Review microcontroller code logic in every possible iteration.		***6***	test microcontroller code logic in every possible state of time relative to a dose taken.	1	2	***2***
LED Indicator Light	Fails to blink red when battery is less than 10%	Patient would not realize device is low on battery.	1	Code logic fails to blink red light or fails to detect current battery level.	6	Review microcontroller code logic when battery power is around 10%.		***6***	Test microcontroller code logic at varying battery levels, especially right around 10%.	1	2	***2***
LED Indicator Light	Too dim to be seen	Patient would be unable to track dosage timing using LED	1	LED not properly lightpiped to surface. LED extrusion not large enough for adequate light to escape.	2	Ensure LED casing is viewed in a dark room by multiple people.		***2***	Test LED light emittance in a dark room by multiple people to ensure visibility.	1	2	***2***
LED Indicator Light	Fails to turn on	Patient would not know if device is operating.	2	Faulty LED/LED wiring or hardware fails to address and activate it when powered up.	3	Test LED function and ensure microcontroller can properly activate LED.		***6***	Test LED function prior to packaging as well as hardware code to ensure it can properly activate LED.	2	1	***2***

6 V step up converter	Step up converter overheats or wiring disconnects causing inactivation of solenoid.	Patient unable to unlock pill storage.	5	Inadequate venting or connection of step up.	5	Determine how solenoid pin could be dislodged or voltage step-up connection fails.		***25***	Run > 10 repetitions of unlocking solenoid via the voltage step up to ensure overheating doesn't occur. Drop testing as outlined above.	5	2	***10***

Bioengineering 3801/4801 Biomedical Engineering Design
Prototype Development

Pharmacokinetic Control of Opioid Dosing with Feedback
Revision: 4
12/05/24

Name	Position	Signature	Date
Jude Werth	Team leader, Code and Software	Jude	11/21/24
Kyle Harshany	Design Engineer, Mechanical and CAD Modeling	Kyle	11/21/24
Wyatt Young	Code and Software, Market Researcher	Wyatt	11/21/24
Daniel Candland	Quality engineer, Electronics and PCB	Daniel	11/21/24
Parker Mason	Teaching assistant		
Tomasz Petelenz	Instructor		

Revision Log			
Revision	Description	Initials	Date
1	Initial release	KH	11/21/24
2	Added engineering design and functional performance in circuitry element section.	WY	12/03/23
3	Added step by step to complete circuitry design element	WY	12/05/24
4	Added app element section as well as source code in appendix	WY/JW	12/05/24

1 Introduction

The purpose of this document is to indicate the general process and follow the documentation
required to establish a Prototype Document that outlines the steps to build and feature highlights for the final prototype of the doseX pharmacokinetic control device for opioid dosing for BME 4801 of Fall 2024.

2 Scope

The scope of this document pertains to any activities related to prototyping of the doseX pharmacokinetic control device for opioid dosing control device during Fall 2024.

3 References

- Design and Development Plan v5
- Design Specifications Document v3
- Design Control - DC 04 24 - Design Verification rev 07 3/27/2023
- Design Control - DC 04 01 - Design Control Policy - rev 03 01/22/2021
- FDA 21 CFR 820 Subparts C, F, and G – Design Control, Identification and Traceability, and Production and Process Controls

- Design Control Guidance for Medical Device Manufacturers, available from CDRH

4 Sources Consulted For Information (identify)

- BME4801 Lab Instructors
- Micah Frerck
- Adafruit (various component datasheets and address directions)
- Espressif
- DigiKey
- OpenAI

5 Content that must be addressed (required)

5.1 Prototype App

5.1.1 Description of prototype element

(in the following subsections provide description of the finished element using appropriate photographs and stills captured from videos. All images need to include appropriate captions that fully describe the content. Fully describe how the element meets the intended human factors and ergonomics)

5.1.1.1 Prototype design

The app UI design is based in React Native and typescript. The decision to make the UI design in the app was made with creativity and engineering considerations.

Outside of ingenuity, using an app UI has engineering benefits. The decision to use a mobile app is to reduce the resources and costs needed in electrical hardware. Additionally, operating in software allows for more flexibility in adjusting the specifications.

The first problem that needed to be solved was being able to communicate with the microcontroller. To do this we decided to use adafruitIO feeds. In this method, the link to the adafruitIO website was inputted into typescript along with account access information. With this, we are able to both export data and fetch data to cloud storage an effectively communicate with the microcontroller.

Figure 1: Screenshot of feeds from Adafruit IO showing most previous value received by the feed.

The next issue was data storage and the method to store patient dosing data. In this regard we made two decisions. The first was to have the app and backend system based on specific users and be able to distinguish different accounts. In the scope of the design review this functionality is not particularly relevant, however, it has future implications when expanding this project. An important factor in our device is bridging the communication between, patients, pharmacies, and clinicians. For this to happen, a large scale app system will need to be able to distinguish different patient accounts. To physically store the data a local json file was used.

Figure 2: Screenshot of data structure used to store JSON formatted patient dosing information for history tab in software app.

The next decision was the specific way the app and microcontroller interact. To do this we used two Boolean adafruitIO feeds. The first is TakeDose which initiates the pill unlock and load cell recording in the microcontroller. The next is DoseSent which the microcontroller send to communicate that a new load cell value was taken. The app listens for this code as an indicator the dose recording session is over and the user can return to the home page.
For the algorithm which determines the number of pills. We decided to base this off of two factors, patient pain feedback and time since the last dose. To quantify patient feedback, a three input system was used based on high pain, medium pain, no pain. To quantify the time since the last dose we also deployed three states, green, yellow, red. After taking a dose, you are in the red state for 4 hours, then between 4-6 hours you are in the yellow state and afterward in the green.

The final choice was how to display the dosage history to the user. Initially, the plan was to have graphics visually showing the history data. Unfortunately typescript is not well equip to handle math related functions. To compromise we designed a table to store the recent dosage entries.

5.1.1.2 Human interface function

The app is a tab based navigation system with two tabs, a home tab, and a history tab.

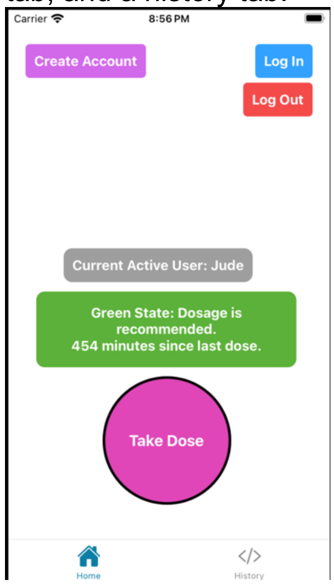


Figure 3: Home page of app software. Shows large Take Dose button that triggers dose actions to microcontroller. At top the user tools to create an account and log in is shown along with time since previous dose.

The home tab has three central components, the account access, account display, and take dose. Starting with account access, this has three buttons, "Log In", "Log Out", "Create Account". Log out will simply log the user out. Log In brings out a window which prompts the user to enter a UserId, if the id is in the json file it will elevate that account to the active user.

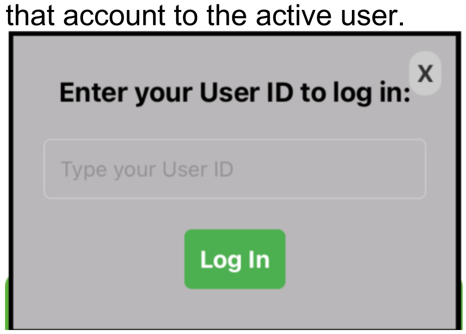


Figure 4: Log in prompt displayed when log in button is pressed.

The create account button has similar functionality as the log in button but instead checks if the userID already exists.

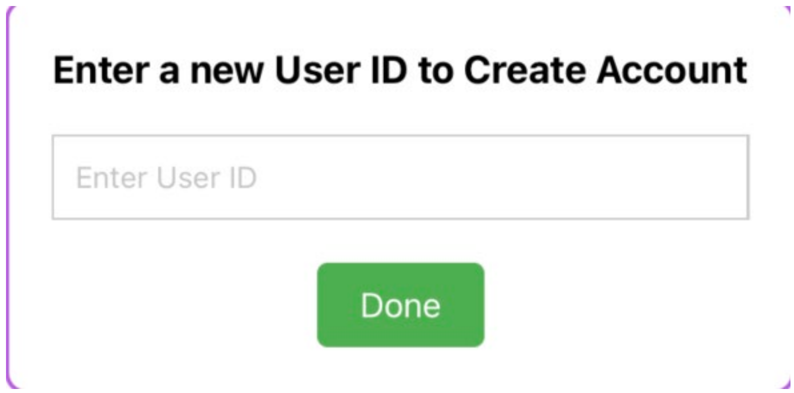


Figure 5: Input for user id to create an account in the doseX app software.

The create account button has additional functionality to initialize the account. The user is prompted to keep the bottle on a flat surface and put in their prescription.

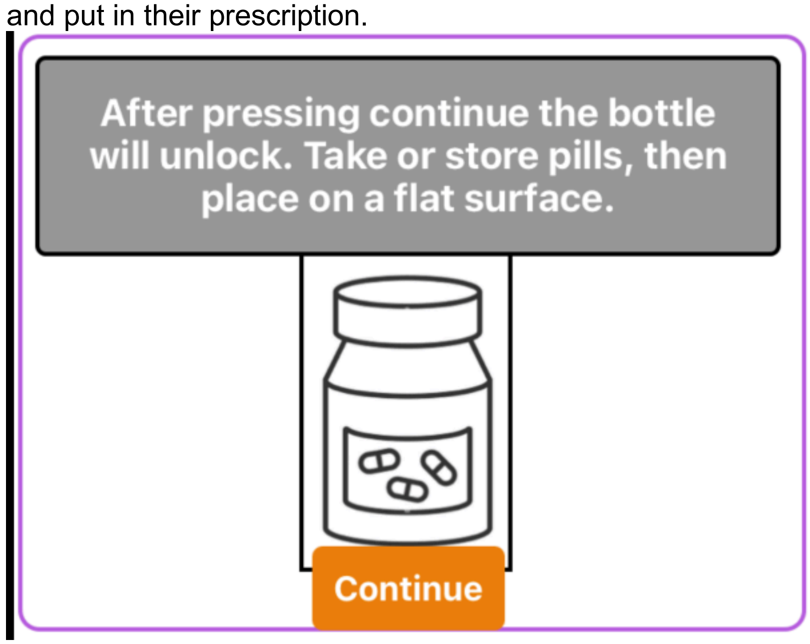


Figure 6: Prompt shown to patient after TakeDose button is activated and a dosage recommendation is given. Allows for an accurate load cell sample from microcontroller after dose is retrieved by the user.

Once the Continue button is hit the app will communicate to the microcontroller (via TakeDose) to unlock the bottle and later take a load cell measurement. This process allows specific users to have different pill contents, once again this isn't relevant for a single bottle, but establishing this protocol will help upscaling in the future. The next section is the account display section which has two components, display user and display state. The display user section simply displays the current active account and also serves as a refresh button. When this is pressed, the app UI refreshing updating the account state to all of the child components. The display state is a conditional rectangle which displays either green, yellow, or red depending on the time since the last dose.

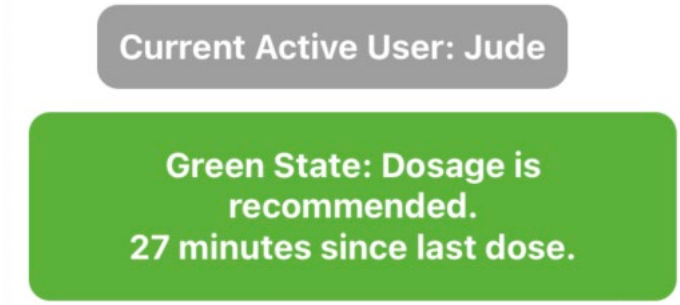


Figure 7: Close up of time since previous dose as well as dose state which aligns with the LED on the bottle.

The final section of the take dose. Which is a button that triggers the dosing procedure. When pressed the pain level input prompt appears which will contribute to the dosage recommendation.

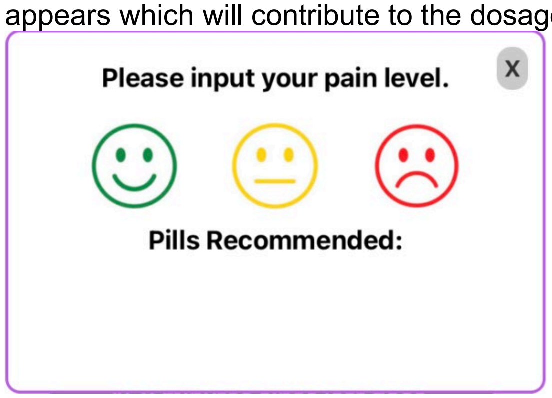


Figure 8: Prompt to take patient pain feedback before algorithm determines number of pills to recommend to patient.

Afterward the same "flat surface prompt" from the create account procedure will be used to unlock the bottle, take out pills, submit a new load cell value then relock the bottle.
The next tab is for dosage history, which contains a title and a table with the dosing history.

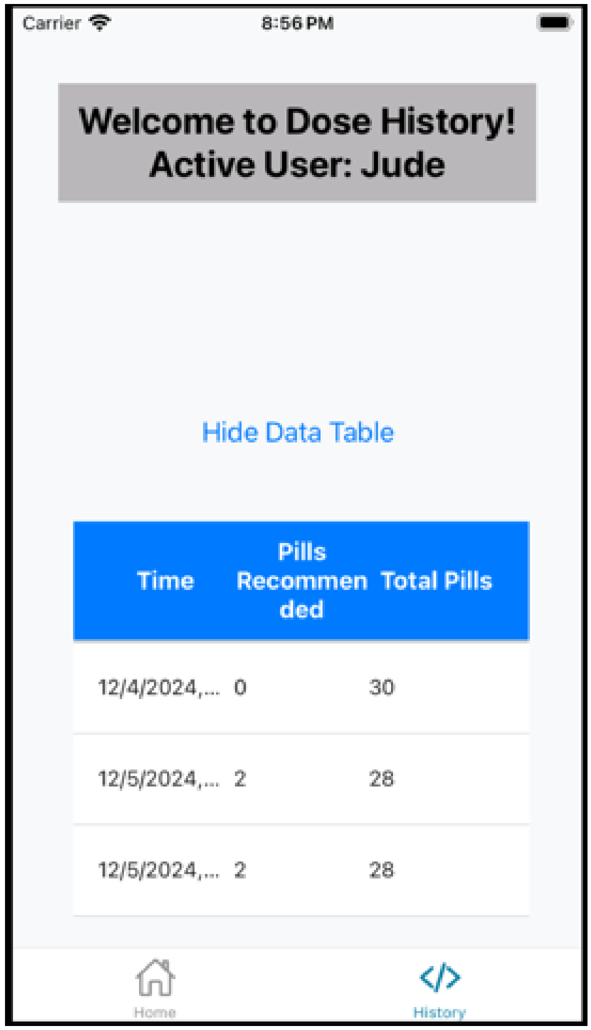


Figure 9: This table displays the time of the dose, the number of pills recommended based on our algorithm, and the total number of pills measured by the load cell. There is also functionality built in to either show or hide the data table.

5.1.2 Provide step-by-step instructions to build this prototype element

1) Download React Native and Expo components in the terminal
2) Create Expo Project
3) Format data storage (data-storage.json)
4) Create necessary functions:
   a. SaveData: input(userID, dateTime, pillsRecommended, numPills) → save these variables in data-storage.json
   b. set_active_account: input(userID) → sets this userID as the active account
   c. get_active_account: → output(the active account user ID)
   d. fetchAdafruitIO: input(feedKey) → output(adafruitIO value)

e. exportAdafruitIO: input(value, feedKey) → exports that value to the labeled feedkey
f. getTimeSinceLastDose: input(userID) → finds the most recent dateTime in the json file then finds the current time. Using these two factors this function output(timeDiff)

5) Create pop-up child components

a. bottleUpright: Establishes a pop-up which prompts the user to keep a bottle of a flat surface. Then communicates the necessary information to the microcontroller. Afterward this function exports the two variables taken from the microcontroller.

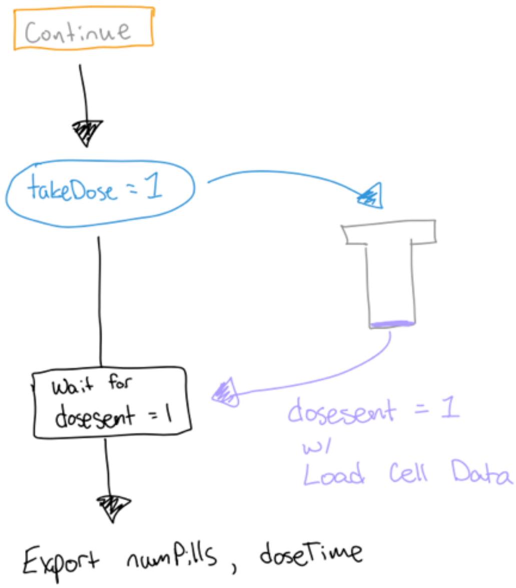
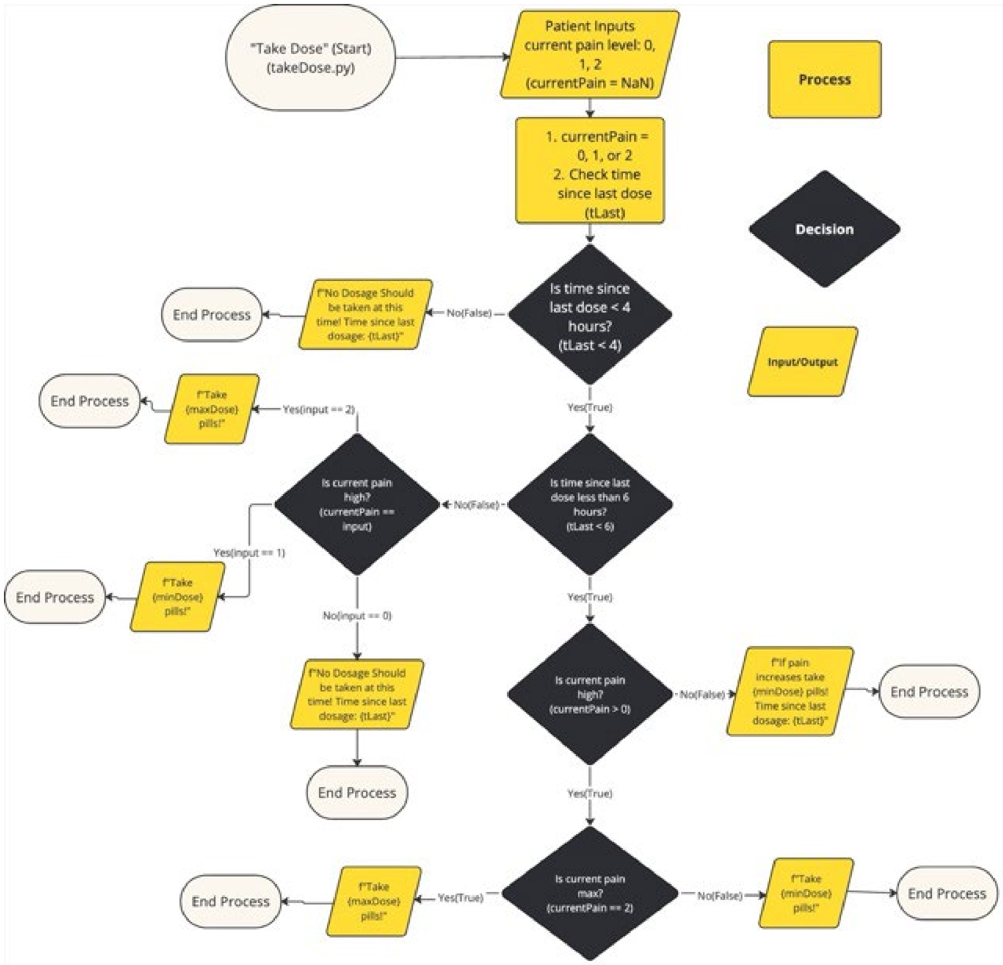


Figure 10: Simple diagram which details the order of operations for activating a TakeDose and receiving the corresponding DoseSent feed to then export dose information to the history table.

b. createAccount: Creates a pop-up window which prompts the user to enter a new UserID, it then initiates the bottle upright function. Using the variables (userID, dateTime, numPulls, and pillsRecommended=0) to SaveData in the json file.
c. dataTable: Creates a table display from information in data-storage.json
d. conditionalRectangle: Uses getTimeSinceLastDose to determine the account time state. It then creates a rectangle based on the state.
e. LogIn: Prompt that has a space available to enter a userID and then exports that value.
f. painLevelInput: Establishes a pop-up with three buttons (happy, neural, sad face). Afterward it uses an algorithm to recommend a number of pills.



Flowchart 1: Flowchart outlining the algorithm used to determine the number of pills to recommend to the patient based on their pain feedback and the current state of the bottle.

6) Create tab files: Deploy the child components to create the tab screens which will make up the UI. This is the file which contains the styling information and manages whether or not the pop-ups should be visible.

5.2 Lid and Locking Mechanism

5.2.1 Description of prototype element

The lid and locking mechanism element includes the physical mechanisms that interact with each other to lock the bottle, the lid and the solenoid, as well as the circuitry required to drive the solenoid, including a transistor, flyback diode, resistor, and input line from the GPIO on the microcontroller.

5.2.1.1 Prototype design

The majority of the design decisions relied on how to lock a rotating screw cap, the final design ended up using a small notch in the side of the lid that the solenoid bolt would slot into when the lid is screwed on. From there, the electronic components were chosen based on functional requirements and circuit analysis.

5.2.1.2 Functional performance

The solenoid-lid interaction is quite simple, and works well. As the solenoid is spring-loaded, when the lid is pushed down onto the bottle, it pushes the solenoid bolt out of the way until it is twisted, exposing the notch into which the bolt fits (see images below). The electronics side took some work to get right, as the resistor value had to be tampered with, due to the fact that the resistance calculated was too low, and overwhelmed the microcontroller with a high current draw. The transistor also had to be switched out for one that had a higher bridge-emitter impedance, for the same reason.

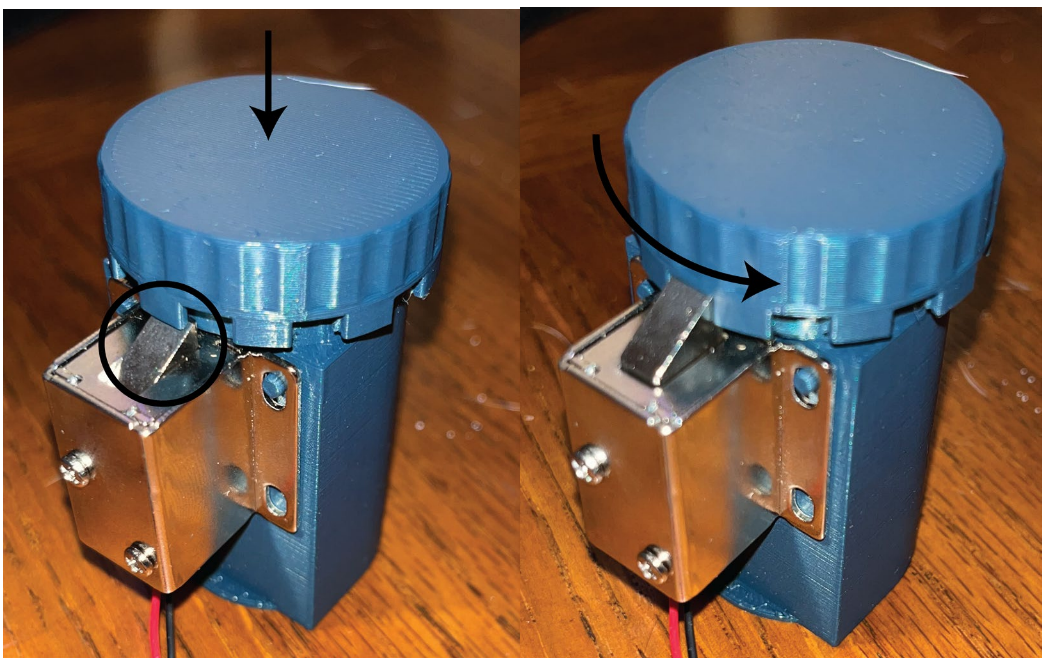


Figure 11: Up close look at solenoid locking mechanism and associated rivet used to catch the solenoid slug when in a locked state.

5.2.2 Provide step-by-step instructions to build this prototype element

Bill of Materials:

Item	Supplier	Part #
Small Lock-style solenoid	Adafruit	5135
6V Step-Up Voltage Regulator U3V16F6	Pololu	4942
NPN Bipolar Transistor (PN2222)	Adafruit	756
Prusamelt PLA Gentleman's Grey	Prusa3D	3644
1N4001 Diode	Adafruit	755
150 Ohm Resistor	Design Lab	N/A
Custom PCB (Schematic Below)	OSHPark	N/A

26 ga solid core Wire	Design Lab	N/A

The lid was 3D printed according to the drawing below. The electronics were tested on a breadboard, and the solenoid was mounted on the bottle using small pins (see below, breadboard and circuit diagram).

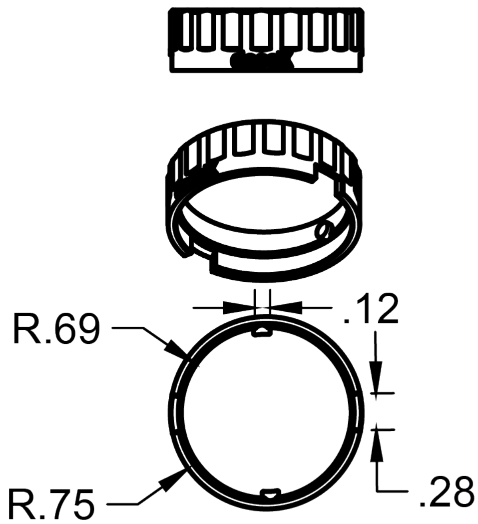


Figure 12: Diagram of the Lid

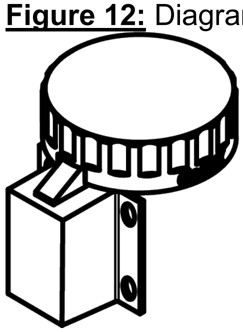


Figure 13: Interaction of solenoid and lid, bottle (not shown) supported the solenoid via pins through the mounting holes.

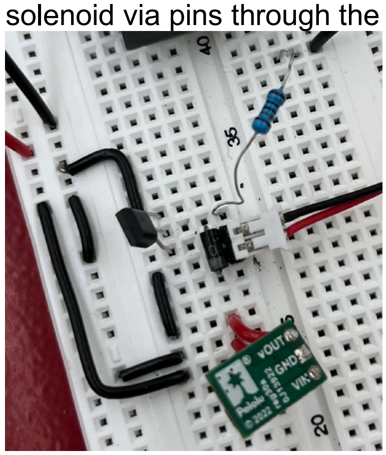


Figure 14: Breadboard setup, resistor from GPIO (left), transistor (bottom left), diode (center), wires to solenoid (top), and 6V regulator (right)

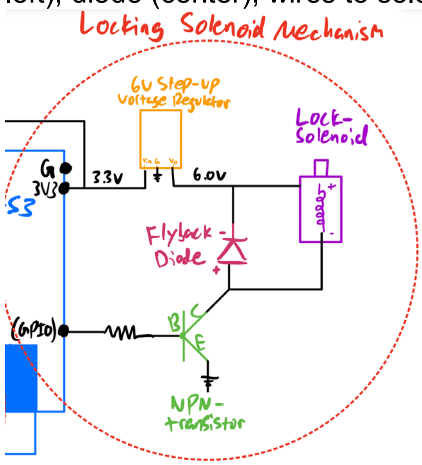


Figure 15: Circuit diagram for solenoid driving circuit

5.3 Electrical Control Circuit

5.3.1 Description of prototype element

(in the following subsections provide description of the finished element using appropriate photographs and stills captured from videos. All images need to include appropriate captions that fully describe the content. Fully describe how the finished prototype element meets the intended form and function requirements)

5.3.1.1 Prototype design

(describe the creativity and engineering in your prototype element)

The electrical control circuit features 2 main branches of circuitry from the ESP32-S3 microcontroller; one to power and control the solenoid, and one to sample the load cell. The solenoid locking mechanism is also described in the Lid and Locking Mechanism section (5.2) but will be briefly discussed here.

The solenoid is driven with 500 mA of current and 5 V DC voltage. To achieve this we used a 3V to 5V step up converter to power the solenoid from the microcontrollers native 3.3 V pin output. Since the solenoid is ultimately a large inductor, a flyback diode was placed in parallel with the solenoid to prevent voltage spikes back into the microcontrollers GPIO pin. An NPN transistor was used to control the solenoid.

The load cell was a 100 g single point load cell capable of weighing pill contents. The decision to use a load cell instead of a proximity sensor or a single loaded spring to hold and also keep track of the pills was a creative and robust engineering decision. With our setup, the patient doesn't have to load each opiate pill into the bottle one at a time and can instead load their entire prescription exactly as they would a normal pill bottle. Using a proximity sensor to count the incoming pills would have also most likely been very inconsistent and hard to integrate. The load cell obviously required a Wheatstone bridge configuration to convert the change in bridge resistances from load to a voltage. We used a 24-bit ADC to accomplish this. The microcontroller powers as well as reads from the load cell when appropriate. The 24-bit resolution form the ADC allowed our device to have the resolution needed to weight individual pill contents and detect changes in the pill compartment.

A simple and small tactile button was connected to the microcontroller to act as the emergency button in case a patients phone lost power or their device couldn't access wifi. When discussing with our clinician, Dr. Kuck, this was a highlight feature of our device and represented the amount of forward thinking the device took into account when considering a patient using the device on a day-to-day basis. Once the device is in a red(i.e no dose allowed state) it will listen for a button press which will awaken the microcontroller and trigger the unlocking of the compartment.

The ESP32-S3 features a NeoPixel LED onboard. This allowed the microcontroller code to easily address and control the LED between the green, yellow, and red states when appropriate. This feature allows the patient to easily glance at the pill bottle and get an idea of when their last dose was taken and when it could be appropriate to take their next dose.

The meat and potatoes of the device circuitry is the ESP32-S3 microcontroller which is the center of all of these processes. The code(written in CircuitPy version 9.1.3) flashed onto the microcontroller is at its most simplest form a steady state machine. It includes States.py which contains a constructor which addresses and initializes the solenoid, LED, button, and load cell. It then defines three main methods; "ActiveDose", "DeepSleep", and "LightSleep". The way that these states cycle between each other is shown in the state diagram below. The flashed code also includes a LoadCell.py which creates an object representing the NAU7802 24-bit ADC and contains driver methods to sample, calibrate, and zero the load cell. It also contains helper methods to convert the raw counts from the ADC to number of pills.

In code.py, a wifi network is connected too using the onboard 2.4 GHz wifi radio included with the ESP32-S3. If a network isn't found or connected too, a retry connection loop will begin which exponentially increases the time between connection retrys starting at 0.1 second. Once connected to a network, an MQTT client is setup to communicate with the Adafruit IO server setup for the project. 3 main feeds are listened for and uploaded to by the MQTT client; "TakeDose"(listens for dose requests made in the app), "LoadCell"(where load cell samples are uploaded for retrieval by the app), and "DoseSent"(a boolean feed for ensuring the app can tell once a load cell sample has been uploaded). All 3 feeds are subscribed to and a States.py object is initialized to loop states. In the main loop, the MQTT client ensure its connected and then awaits a dose request. Once a dose request is received the States.py object enters the "ActiveDose" state. This is where the solenoid lock is unlocked and once the patient has retrieved their pills, a load cell sample is taken and the LED is updated from green to red indicating it's entering deep sleep. The last call from the ActiveDose method is to enter DeepSleep.

Once the bottle is in the DeepSleep state, it enters a loop to check for emergency button presses but does not use the MQTT client to listen for dose requests from the app. The bottle is in this state for 4 hours indicating an unsafe period of dosage from the last dose. We figured this would be the best place to save battery life. Thus, code was written to access the microcontrollers deep sleep mode which would use an alarm for either when the button is pressed or when 4 hours is up. However, this code was unable to be implemented in the time given. With a bit more time to troubleshoot this feature could have easily been implemented.

After the DeepSleep state, the controller enters LightSleep state which first adjusts the LED color to yellow indicating an in-between range for 4-6 hours when a dose is safe but may not be necessary. It uses the MQTT client to listen for dose activations as a lower sampling period than when in the active state. It listens for a dose request every 1 second. Once 2 hours of LightSleep has passed the LED is adjusted to green and in an active state where it checks for a dose request every 0.1 seconds as it expects a dose past 6 hours from the previous dose (it should be clarified that this transition all occurs in the LightSleep method in States.py and once a dose request is detected it calls the ActiveDose method which then loops back to DeepSleep).

The battery used to power the circuitry was a 3.7 LiPo battery with 1200 mAh. The positive terminal was soldered to the 5 V pin and a GND pin on the ESP32. This ensures that the 3.7 V which can fluctuate was regulated to the required 3.3V needed for onboard processes such as the button, ADC, etc. LiPo batteries tend to be slimmer and that allowed us to still meet our portability requirement. The ESP32 also features a power management system that allows the device to be recharged when connected to USB power.

5.3.1.2 Functional performance

After determining our transistor had too high of a voltage between the collector and emitter at saturation (Vce(SAT)) to consistently drive the solenoid, a new transistor was ordered and used with an updated resistor between the GPIO pin and the base pin of the transistor. This adjustment allowed the solenoid to be driven by our circuitry with no misses. We ran a solenoid test to confirm this which can be viewed in the Verification Testing document. The solenoids success is also shown in the demo videos below.

The load cell's accuracy after calibration was our devices main shortcoming. We were able to achieve an accuracy of plus/minus 3 pills from the true number of pills. We believe if we had more time to fine tune the calibration we could have achieved fantastic accuracy to the load cell. However, we did set out to ensure our load cell could differentiate between 1g, 2g, and 5g weights to get a baseline functionality for the application. This test was passed with flying colors and can be seen in the Verification Testing document.

Regarding the LED and emergency button, tests were done to ensure these features were addressed and adjusted properly according to the state of the bottle. Both of these features worked flawlessly once implemented correctly throughout prototyping. For example, the button was often used for testing the solenoid activation and was never an issue. Similarly the LED light was implemented early and thus went through hundreds of cycles through states when testing the solenoid or load cell. Both these features are shown in the demo video.

Overall, the hardware code that was flashed onto the microcontroller proved to be robust in controlling the states and power the solenoid and load cell processes. It was validated using OpenAI's ChatGPT 4o model as well as during the testing phase as the code was the only way to test other electronic components. The code is documented below in the appendix and shown to be effective in the demo video.

The 3.7 V 1200 mAh LiPo battery was also a shortcoming of the device. In order to meet the battery requirements for the design specifications a 2200 mAh specified battery should be used. Although the specification of the battery used was inadequate(granted given more time, a higher capacity could have been ordered and proper power analysis could have been completed) the LiPo battery was successful in allowing us to meet both our portability as well as rechargeable design requirements.

5.3.2 Provide step-by-step instructions to build this prototype element

1. **Bill of Materials:**

Component	Product	Important Specifications
Microcontroller	Espressif ESP32-S3-DevKitC-1-N8R8 (Adafruit ID: 5336)	• Dual-core Xtensa LX7 (240 MHz, 32-bit architecture)

		• Integrated Floating-Point Unit (FPU) • 8 MB Flash • 8 MB PSRAM • WiFi 4 (802.11 b/g/n) • Supports 2.4 GHz WiFi frequency • Operating voltage: 3.0 V - 3.6 V • Built-in LDO regulator • 500 mA peak current output from onboard voltage regulator • 45 GPIO pins (multipurpose for ADC, DAC, PWM, UART, SPI, I2C, etc.) • Dedicated USB OTG controller with native USB 1.1 support • RTC (Real-Time Clock) • I2C: Up to 2 buses • 2 mounting holes • -40°C to 85°C • Onboard addressable RGB NeoPixel LED • JST connector for LiPo battery (charging circuit included)
Push-Pull Solenoid	Adafruit Small Lock-Style Solen (Adafruit ID:5135)	• 20 mm long body • Activated with 5 V DC • Current(at 5 VDC): 1.1 A • Weight: 12.6g

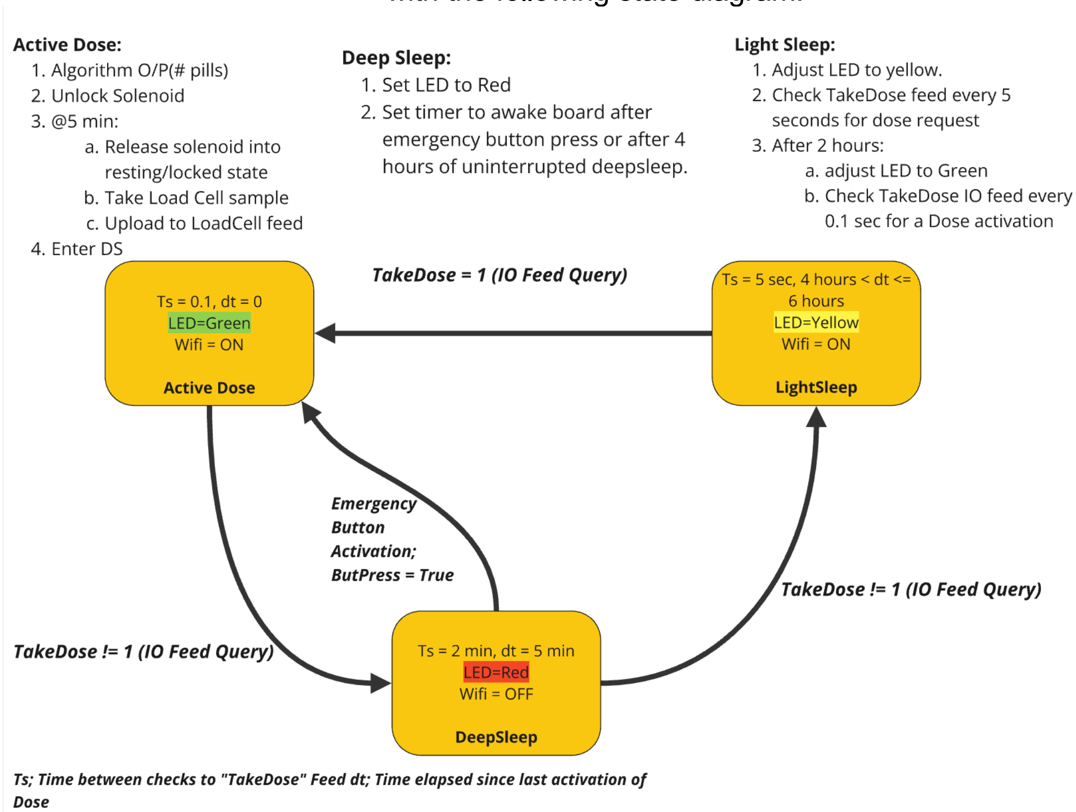
		• Throw(at DC 5 V): 3 mm
Load Cell	Single Point Load Cell-100g (Phidgets ID:3139_0)	• Maximum weight capacity: 100g • Operating Range: 3 V – 12 V • Aluminum Alloy • 4 mounting holes • 52 mm x 12.7 mm x 12.7 mm • 4-wire configuration •
LED	Adafruit NeoPixel RGBW LED w Integrated Driver Chip (**Adafruit** 2759)	• Single LED: 5mm x 5mm • 4 pins: VDD, DOUT, GND, DIN • Power supply voltage: 3.5-5.5 V Working temperature: -40°C 85° C • Red: 320-580 mcd, Green: 815-1275 mcd, Blue: 160-321 mcd
Tactile Button	Adafruit Soft Tactile Button (8mm) (Adafruit ID: 3101)	• Dimensions: 7.8 mm x 7.8 mm x 4.9 mm • Weight: 0.1 g • DC 12 V 50 mA • Operating force: 120 ± 40 gf • Life: 50000 cycles Min.
Lithium Battery	Lithium Ion Polymer Battery – 3.7 V 1200 mAh (Adafruit ID: 258)	• Weight: 8.8g • Size: 34mm x 62 mm x 5 mm • Output: 1200 mAh at 3.7V nominal

ADC	Adafruit NAU7802 24-bit ADC (Adafruit ID: 4538)	• Dual input channel • 24-bit resolution • Various gain options 1-128 • Operating Range: 2.7 V – 5.5 V • I2C communication protocol • Built in offset and gain calibration
6V step-up volta regulator	6V Step-Up Voltage Regulator U3V16F6 (Polulu ID: 4942)	• Output Voltage: 6 V • Max input current: 1.6 A • Min input voltage: 2.7 V startup
Flyback Diode	1N4001 Diode (Adafruit ID: 755)	• 0.7 V drop • Pass up to 1 A • Protect against up to 50 V reverse voltage
NPN Transistor	ZTX851STZ (DigiKey: ZTX851SCT-ND - Cut Tape (CT))	• Ic max: 5 A • Vce(max): 150 mV @ 200 mA(Ib), 5 A(Ic)

*Other needed components include a soldering iron, wire, computer for flashing hardware, M3 screws (specified in mechanical prototype section), 820 ohm resistor, and other tools needed for ease of assembly.

2. **Flashing microcontroller with appropriate hardware CircuitPy driver code:**
    a. Install Thonny IDE
    b. Plug ESP32-S3 board into computer via micro-USB
    c. Use the RESET and BOOT button to ensure ESP32 is running a 9.x.x version of CircuitPy and select/configure interpreter in Thonny for CircuitPy
        i. If board is not flashed with any version of CircuitPy or a non-9.x.x version; download the .UF2 file for 9.x.x CircuitPy, hold the RESET button for 3 seconds and then press the BOOT button.
        ii. Drag .UF2 file into the BOOT file in the drive and press RESET to flash the board with the CircuitPy bootloader.
    d. Copy paste CircuitPy source code from appendix into 3 different

classes: code.py, States.py, and LoadCell.py. Code should align with the following state diagram:



Flowchart 2: Finite State Diagram describing the cycle of states used by the doseX device. Details the 3 main states; Active Dose, Deep Sleep, and Light Sleep and their corresponding MQTT client sampling times, wifi activation, load cell sampling and solenoid activation, as well as query values from the IO feeds. Also details emergency button activation procedure.

e. Ensure the required libraries are added to the "lib" folder of the ESP32-S3 device by cross referencing the import statements at the top of each code file with the version 9.x.x CircuitPy libraries that can be downloaded from CircuitPy's website.
f. Once flashed, connect microcontroller to computer used for flashing and run code.py, output should look similar to the below figure:

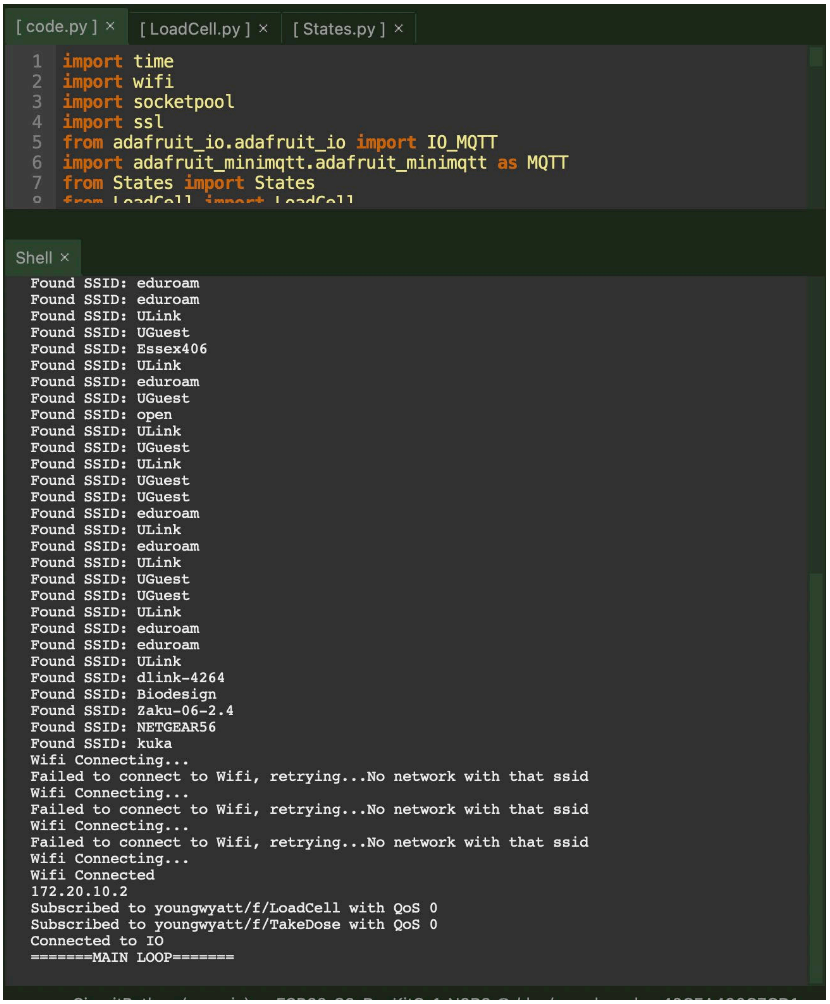


Figure 16: Thonny output once microcontroller is flashed. Shows wifi connections loop, IO MQTT client connection to feeds and main loop initialization.

3. Setup Adafruit IO networking service and appropriate feeds for MQTT client
    a. Create an Adafruit IO account and add the specified feeds from above; LoadCell, TakeDose, and DoseSent to communicate with the app software.
4. Solder components to the microcontroller
    a. Solder components to the microcontroller according to the following wiring diagrams and images:
    b. (Ensure to be extremely careful with bridge wires from load cell as they can be easily damaged during soldering and calibration)

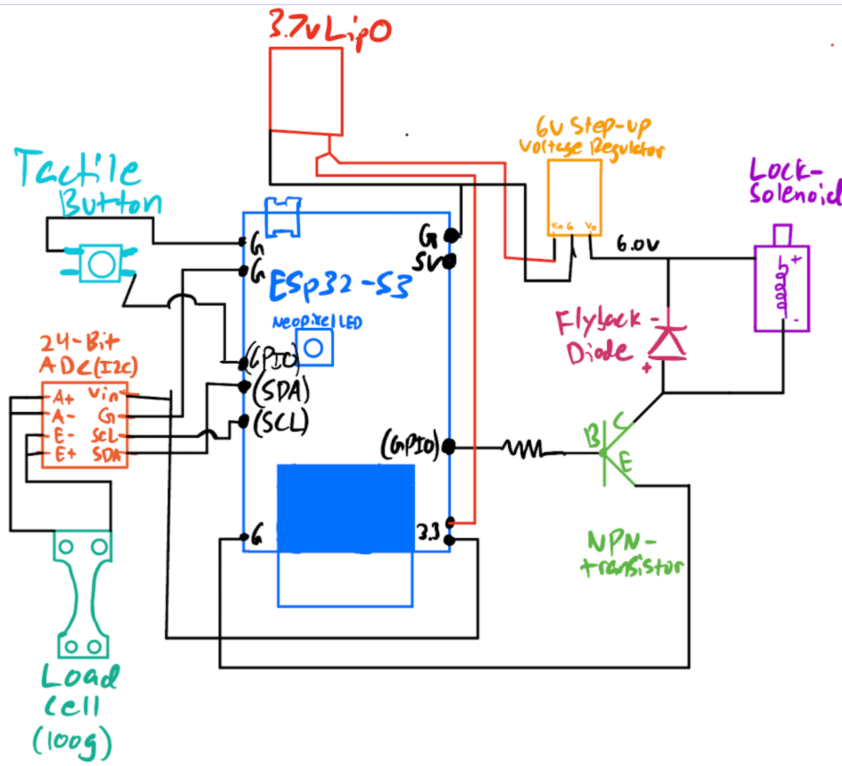


Figure 17: Circuitry diagram showing connections to be made to the microcontroller and the supported peripherals. Includes the solenoid driving circuit, battery splicing, emergency button pinout, and ADC/load cell circuitry.

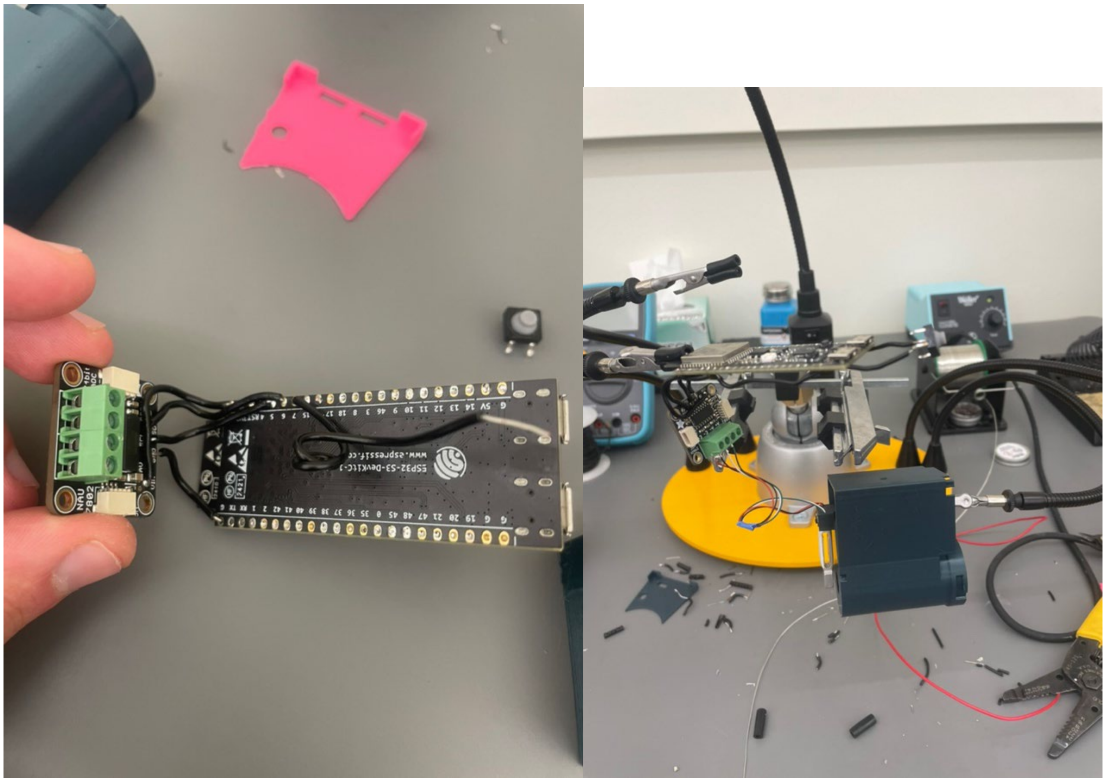


Figure 18/19: Soldered ADC circuit. Shows wires connected to appropriate pins which correlate to the wiring diagram shown in Figure 2. As well as bridge wires from load cell connected to ADC.

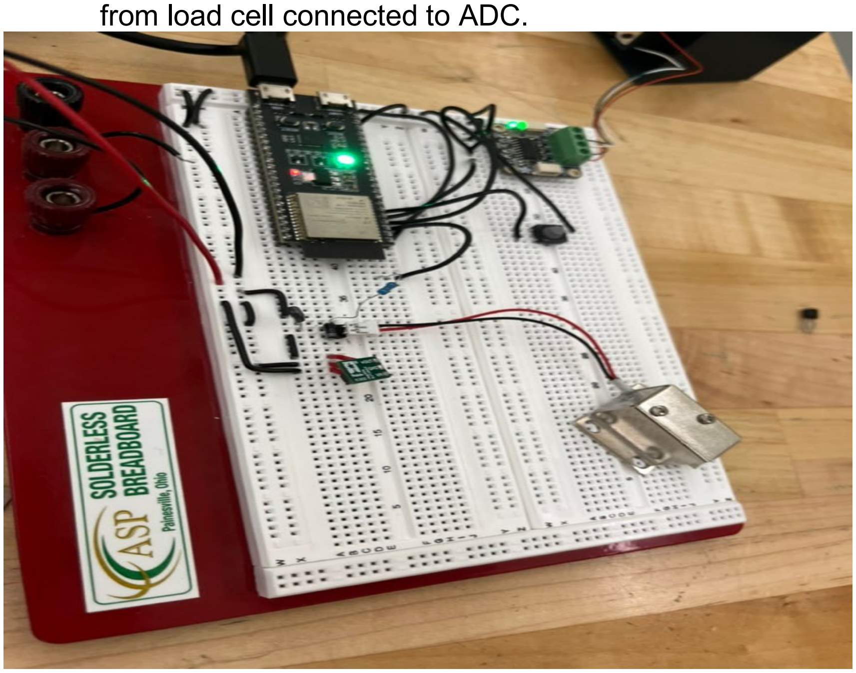


Figure 20: Breadboard representation of circuitry with emphasis on solenoid driving circuit.

Resistor value is 820 ohms and a simple PCB should be used to hold transistor, voltage step-up regulator, resistor, and diode.

5. Calibrate Load Cell to specified opiate being used
    a. Once load cell has been connected to ADC and properly addressed by ESP32 hardware code, attach the load cell to the bottom of the pill compartment as described in the mechanical prototype section.
    b. Uncomment the calibration loop at the bottom of code.py to display the raw counts value from the load cell and the number of pills(or paper clips, screws, or other objects similar to pill weight used for calibration).
    c. Load the pill compartment tray with one pill/object at a time and wait for loop to display updated counts from ADC.
    d. Repeat this till the range + 20% needed for the application is met.
    e. Copy the raw counts into a Python(**or MATLAB file if you're a simpleton**) list along with a range of integers representing the number of pills in the compartment that correlated with the ADC outputs.
    f. Use linregression() to get the slope and intercept of the calibration curve and plot to determine linear response
    g. In LoadCell.py replace the slope and intercept arguments in the EstimatePillCount() method to reflect the obtained calibration.
    h. An average of multiple calibration curves can be taken to increase robustness of linear regression output and increase r^2 value.
    i. A curve for the prototype is shown below with a max capacity of 32 pills

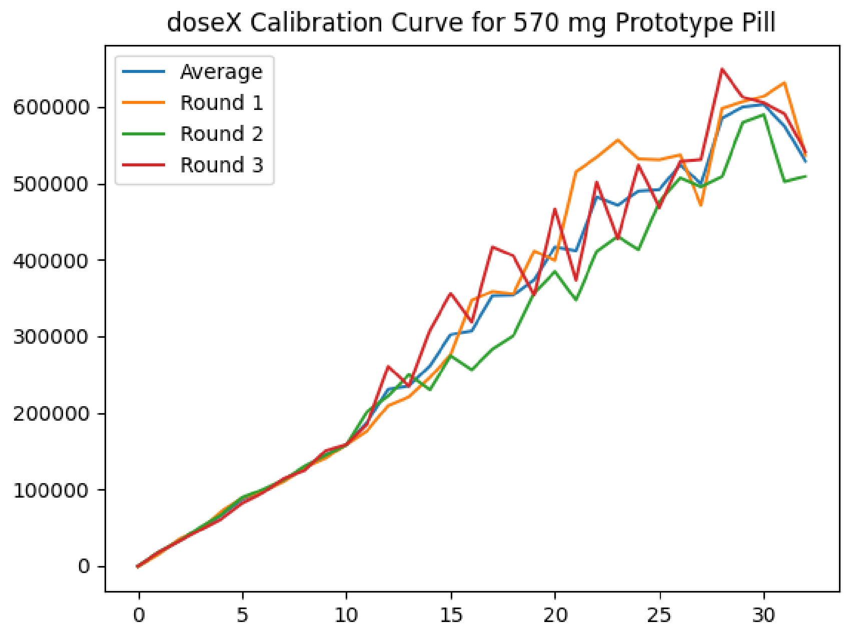


Figure 21: Calibration curve used to obtain slope and intercept values to convert raw ADC counts form the NAU7802 to number of pills. Max of 32 pills using 3 rounds of calibration and an average between all 3 to reduce fluctuations achieving an r^2 value of 0.9892.

5.4 Pill Container and Housing

5.4.1 Description of prototype element

This element consists of the bottle that holds the pills, and the enclosure for housing all non-user components, such as electronics, the locking solenoid, and load cell.

5.4.1.1 Prototype design

The bottle section was designed to be exactly the same dimensions as a 13-dram bottle, 0.57 in radius, and 1.45 in height, and the housing section consists of a rectangular prism attached to the side of the bottle. Part of the bottle has a flattened section to accommodate the solenoid. The housing section also extends below the bottle by approx 0.7 in, and was based off the approximate volume of the electronic components when oriented as needed. Despite the design specifications stating PEI would be used, the prototype was made using PLA for ease of printing and low cost.

5.4.1.2 Functional performance

Both the bottle and housing sections function as necessary, the bottle can hold 32 of the test pills (generic 500mg acetaminophen pills, larger than the design-oriented vicodin pills), and the housing section holds all electronics and is able to fully seal when the device is assembled.

5.4.2 Provide step-by-step instructions to build this prototype element

All components for this element where 3D printed using Prusament PLA on a MK4 Prusa printer.

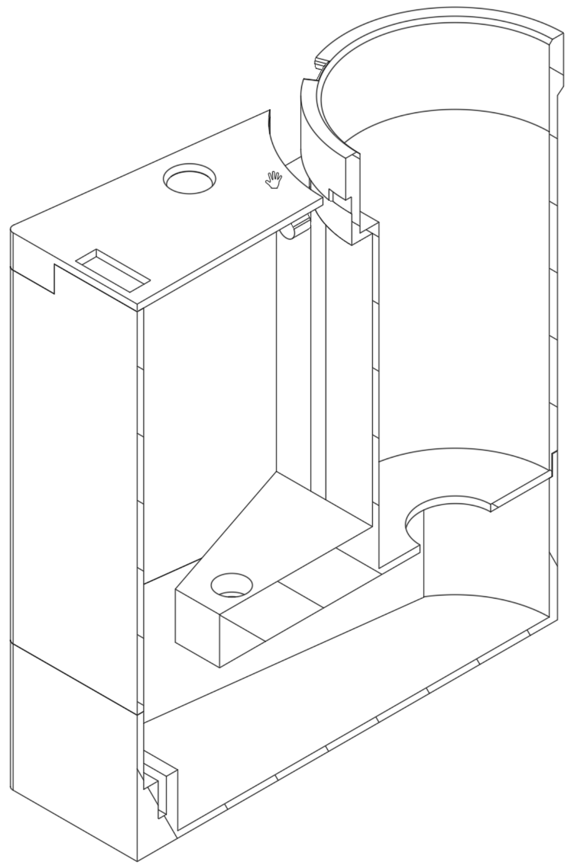


Figure 22: General form of bottle and housing

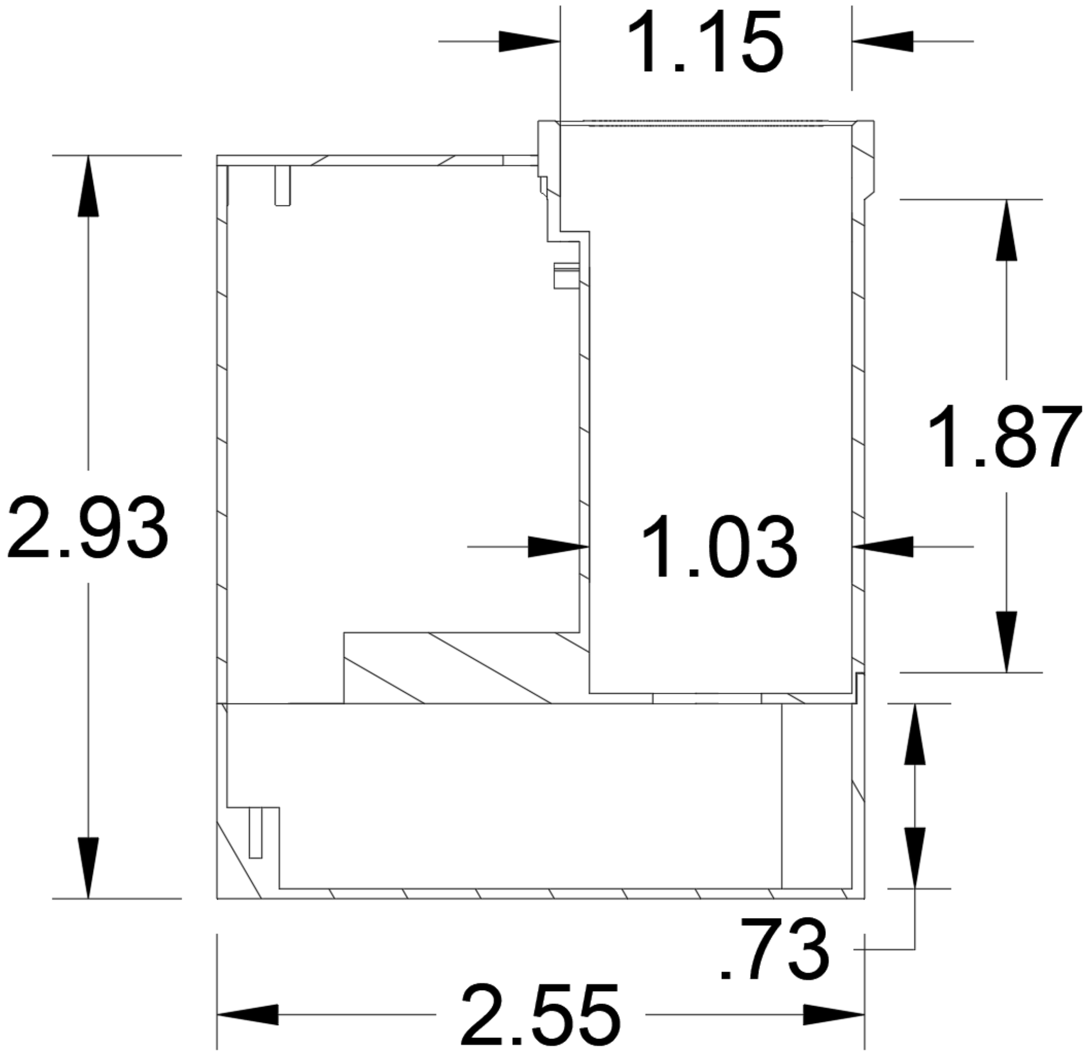


Figure 23: Dimensions of bottle and housing prototype

Appendix:

```
import time
import wifi
import socketpool
import ssl
from adafruit_io.adafruit_io import IO_MQTT
import adafruit_minimqtt.adafruit_minimqtt as MQTT
from States import States
from LoadCell import LoadCell
import board
import neopixel
import digitalio
import alarm
import busio
from cedargrove_nau7802 import NAU7802

# Test code to ensure code.py is ran
print("Code.py is running")
#pixels = neopixel.NeoPixel(board.IO38, 1, brightness=0.5, auto_write=True)

# while True:
#     # Set LED to red
#     pixels[0] = (255, 0, 0)
#     time.sleep(0.5)
#
#     # Turn off the LED
#     pixels[0] = (0, 0, 0)
#     time.sleep(0.5)

# Use hotspot for DR
```

```
ssid = "YOUR WIFI SSID”
password = "YOUR WIFI PASSWORD"

ADAFRUIT_AIO_USERNAME = "YOUR IO USERNAME"
ADAFRUIT_AIO_KEY = "YOUR IO KEY"

# add feed and board to IO
LoadCellFeed = "LoadCell"
TakeDoseFeed = "TakeDose"
DoseSentFeed = "DoseSent"

networks = wifi.radio.start_scanning_networks()
for network in networks:
    print(f"Found SSID: {network.ssid}")

# attempt to connect to Wifi and attempt reconnection
interval = 0.1 # start interval at tenth of a second
maxAttempts = 100
attempt = 0
while attempt < maxAttempts:
    try:
        print("Wifi Connecting...")
        wifi.radio.connect(ssid,password)
        print("Wifi Connected")
        print(wifi.radio.ipv4_address)
        break
    except Exception as e:
        attempt += 1
        print(f"Failed to connect to Wifi, retrying...{e}")
        time.sleep(interval)
        # log scale the interval
```

```
        interval = min(interval * 2, 60)


pool = socketpool.SocketPool(wifi.radio)


# MQTT setup for IO
def connected(client):
    io.subscribe(LoadCellFeed)
    io.subscribe(TakeDoseFeed)
    print("Connected to IO")

def disconnected(client):
    print("Disconnected from IO")

def subscribe(client, userdata, topic, grantedQos):
    print(f"Subscribed to {topic} with QoS {grantedQos}")

# ** This function will handle all incoming messages **
def message(client, feed_id, payload):
    print(f"Received data from {feed_id}: {payload}")

    if feed_id == TakeDoseFeed:
        if payload == "1":  # 1 indicates a new dose request
            print("Starting Active Dose Processes")
            BottleState.ActiveDose()

# Initialize the MQTT client for Adafruit IO
mqttClient = MQTT.MQTT(
    broker="io.adafruit.com",
    username=ADAFRUIT_AIO_USERNAME,
```

```
    password=ADAFRUIT_AIO_KEY,
    socket_pool=pool,
    ssl_context=ssl.create_default_context()
)

# Initialize MQTT Adafruit IO client
io = IO_MQTT(mqttClient)

# Assign event callback functions and connect client to io
io.on_connect = connected
io.on_disconnect = disconnected
io.on_subscribe = subscribe
io.on_message = message
io.connect()

# initialize state manager
BottleState = States(io)
print("=======MAIN LOOP=======")
# main loop to handle MQTT
while True:
    try:
        # Call io.loop() to check for incoming messages and handle them
        io.loop()
        # Check if the MQTT client is still connected
        if not mqttClient.is_connected():
            print("MQTT client not connected. Reconnecting...")
            io.reconnect()
    except Exception as e:
        print(f"Failed to maintain MQTT connection: {e}")
        # Attempt to reconnect if disconnected
        try:
```

```
            io.reconnect()
        except Exception as e:
            print(f"Reconnection failed: {e}")
            time.sleep(10)  # Wait and retry
    finally:
        import gc
        gc.collect()
    #delay to avoid hammering loop
    time.sleep(0.5)

# Testing for load cell calibration
# loadCell = LoadCell()
# loadCell.printLdoGain()
#
# print("Load cell zeroed for calibration.")
#
# # 305 mg/vidacin pill 32*305 mg = 9760 mg = 9.760 g
# ### Main loop: Read load cells and display raw values
# numNuts = 32
# while True:
#     if numNuts == 32:
#         loadCell.ZeroCalibrate()
#         time.sleep(8)
#     print("======================")
##     print(f"No. of Pills: {numNuts}")
#     time.sleep(5)
#     value = loadCell.ReadRawVal()
#     print("channel 1 raw value:", value)
#     #time.sleep(2)
#     pills = loadCell.EstimatePillCount(value)
#     print(f"Pills: {pills}")
```

```
#     time.sleep(2)
#     numNuts -= 1
```

Source Code 1: Source code to be pasted into code.py file on microcontroller. Includes import statements, test code to ensure code.py is ran when powered via battery, wifi connection loop, IO MQTT client initialization to appropriate feeds, main loop to listen for IO dose requests, and code to calibrate the load cell.

```
import time
import board
import neopixel
import wifi
import digitalio
import busio
from cedargrove_nau7802 import NAU7802
import alarm
from LoadCell import LoadCell
class States:
    def __init__(self, ioClient):
        # IO client from code.py
        self.io = ioClient
        # Create a Neopixel object for controlling the RGB LED
        self.pixels = neopixel.NeoPixel(board.IO38, 1, brightness=0.6, auto_write=True)
        # set pixels to green for presentation
        self.setRGBColor("blue")
        # Solenoid setup; D4
        self.SolenoidPin = digitalio.DigitalInOut(board.IO4)
        self.SolenoidPin.direction = digitalio.Direction.OUTPUT
        self.SolenoidPin.value = False
        # Setup for load cell across I2C; pins 5(SCL) and 6(SDA)
        #self.LoadCell = LoadCell()
        # button setup
        self.button = digitalio.DigitalInOut(board.IO7)
        self.button.direction = digitalio.Direction.INPUT
```

```
    self.button.pull = digitalio.Pull.UP

def ActiveDose(self):
    self.setRGBColor("green")
    # Unlock solenoid right away
    self.UnlockSolenoid()
    # Wait 5 minutes( 1 for testing)
    counter = 0
    while counter < 2:# * 60: CHANGE FOR REVIEW
        print("ActiveDose: Processing Dose uno momento")
        counter += 1
        time.sleep(6)
    # Relock solenoid
    print("ActiveDose: Locking Solenoid")
    self.LockSolenoid()
    # Zero the channel and take the reading
    #self.LoadCell.ZeroCalibrate()
    #loadCellValue = self.LoadCell.ReadRawVal()
    #numPills = self.LoadCell.EstimatePillCount(loadCellValue)
    #print(f"Load cell value: {loadCellValue}")
    #print(f"Number of Pills: {numPills}")
    self.PublishLoadCellReading(28)
    self.PublishDoseSent()
    print("ActiveDose: Complete garbage collecting and entering DS")
    import gc
    gc.collect()
    self.DeepSleep()

def DeepSleep(self):
    try:
        print("DeepSleep: starting DS")
```

```
            # Update LED to red
            self.setRGBColor("red")
            # Set button to wake up device from deep sleep
            counter = 0
            while(counter < 600): # 60 seconds * 60 * 4 = 4 hours (60 seconds for DR)
                if not self.button.value:
                    print("Emergency! Entering Dose")
                    self.ActiveDose()
                    return
                counter += 1
                time.sleep(0.1) # check button press every 0.1 seconds?
            print("DeepSleep: 4 hours of DS, entering LS")
            self.LightSleep()
        except Exception as e:
            print(f"Exception occurred: {e}")

    def LightSleep(self):
        print("LightSleep: Entering LS")
        self.setRGBColor("yellow")

        counter = 0
        while (counter < 60):  # 60*60*2 = 2 hours(60 seconds for DR)
            print(f"LightSleep: Time elapsed: {counter * 1} seconds")
            # Call io.loop() to process any incoming messages from Adafruit IO
            try:
                self.io.loop()  # Keep processing messages to check for feed updates
            except Exception as e:
                print(f"Failed to maintain MQTT connection in LS: {e}")
            counter += 1
            time.sleep(1)  # Check every 1 second
        import gc
```

```
        gc.collect()
        print("LightSleep: 6 hours has passed since last dose")
        # LED to green
        self.setRGBColor("green")
        # Continue checking till TakeDose is true
        while True:
            # Call io.loop() to process any incoming messages
            try:
                self.io.loop()  # Keep processing messages
            except Exception as e:
                print(f"Failed to maintain MQTT connection: {e}")
            time.sleep(0.1)  # Check every 0.1 second

    def UnlockSolenoid(self):
        print("Unlocking solenoid")
        self.SolenoidPin.value = True

    def LockSolenoid(self):
        print("Locking solenoid")
        self.SolenoidPin.value = False

    def PublishLoadCellReading(self, val):
        try:
            self.io.publish("loadcell", val)
            print("Successfully published load cell sample")
        except Exception as e:
            print(f"Failed to publish load cell data: {e}")

    def PublishDoseSent(self):
        try:
            self.io.publish("dosesent", 1)
```

```
            print("Successfully published dose sent")
        except Exception as e:
            print(f"Failed to publish dose sent: {e}")

    # Function to set RGB color
    def setRGBColor(self, color):
        if color == "red":
            self.pixels[0] = (255,0,0)
        elif color == "green":
            self.pixels[0] = (0,255,0)
        else:
            self.pixels[0] = (255,255,0)
        self.pixels.show()

## Deep Sleep Code ##
# # Create an alarm for 60 seconds from now, and also a pin alarm.
# time_alarm = alarm.time.TimeAlarm(monotonic_time=time.monotonic() + 5)
# pin_alarm = alarm.pin.PinAlarm(pin=board.IO7, value=False, pull=True)
# print("alarnms")
# # Deep sleep until one of the alarm goes off. Then restart the program.
# alarm.exit_and_deep_sleep_until_alarms(time_alarm, pin_alarm)
# for cases when deepsleep is awoken by alarm, store which alarm woke it up
# WakeAlarmType = alarm.wake_alarm
# print(WakeAlarmType)
# # check if alarm was used and what woke it up from DS
# if isinstance(WakeAlarmType, alarm.pin.PinAlarm):
#     print("Woke from button press, entering active dose")
#     BottleState.ActiveDose()
# elif isinstance(WakeAlarmType, alarm.time.TimeAlarm):
#     print("Woke from timer, entering LS")
#     BottleState.LightSleep()
```

```
# else:
#     print("Standard Startup: --- Main Loop ---")
```

Source Code 2: Source code to be pasted into States.py file on microcontroller. Includes import statements, States object constructor that initializes the on-board NeoPixel, Solenoid GPIO pinouts, and emergency button GPIO pins. Includes 3 states methods; ActiveDose, DeepSleep, and LightSleep as well as other helper methods for publishing data and locking/unlocking the solenoid as well as setting the RGB color. Also includes code to set an alarm for either a button press or 4 hours to be implemented in DeepSleep mode using the ESP32-S3 low power mode.

```
import time
import board
import neopixel
import wifi
import digitalio
import busio
from cedargrove_nau7802 import NAU7802
import alarm

class LoadCell:
    def __init__(self):
        I2C = busio.I2C(board.IO5, board.IO6)
        self.nau7802 = NAU7802(I2C, address=0x2A, active_channels=1)
        self.ldoVoltage = self.nau7802.ldo_voltage
        self.gain = self.nau7802.gain
        # Initialize the NAU7802 and enable power
        enabled = self.nau7802.enable(True)
        print("LoadCell(): Digital and analog power enabled.")
        self.mgPerPill = 568.5 # 570 mg test pill, adjust for which pill


    def ZeroCalibrate(self):
        self.nau7802.channel = 1
        """Initiate internal calibration for current channel.Use when scale is started,
```

```
        a new channel is selected, or to adjust for measurement drift. Remove weight
        and tare from load cell before executing."""
        print(
        "channel %1d calibrate.INTERNAL: %5s"
        % (self.nau7802.channel, self.nau7802.calibrate("INTERNAL")))
        print(
        "channel %1d calibrate.OFFSET:   %5s"
        % (self.nau7802.channel, self.nau7802.calibrate("OFFSET")))
        print("...channel %1d zeroed" % self.nau7802.channel)

    def ReadRawVal(self, samples=3):
        sampleSum = 0
        count = 0
        while count < samples:
            while not self.nau7802.available():
                pass
            self.nau7802.channel = 1
            sampleSum += self.nau7802.read()
            count += 1
        return int(sampleSum / samples)

    def EstimatePillCount(self, counts, slope = 19739.66, intercept = -13067.16):
        # Convert the current weight to the estimated number of pills
        rawPillCount = ((counts - intercept) / slope) - 0.373272
        print(rawPillCount)
        # threshold
        fractional = rawPillCount - int(rawPillCount)
        if fractional >= 0.5:
            estPills = int(rawPillCount) + 1
        else:
            estPills = int(rawPillCount)
```

```
        # ensure no outliers
        if estPills > 32:
            estPills = 32
        elif estPills <= 0:
            estPills = 0
        return estPills

    def printLdoGain(self):
        print(self.ldoVoltage)
        print(self.gain)
```

Source Code 3: Source code to be pasted into LoadCell.py file on microcontroller. Includes import statements, LoadCell object constructor that initializes the I2C bus on GPIO pins 5 and 6. Includes methods to zero calibrate the load cell as well as read raw values and estimate pill counts from count values from the bridge resistances and ADC. Also includes helper methods to access the LDO voltage and gain of the NAU7802 ADC. Can be easily adjusted for different calibration curves; slope and intercepts depending on the individual pill weight.

```
Account_state:
import * as FileSystem from 'expo-file-system';


// Path to the JSON data file
const dataFilePath = `${FileSystem.documentDirectory}data-storage.json`;


export const getTimeSinceLastDose = async (activeUser: string): Promise<number | boolean> =>
{
  try {
    // Check if the JSON file exists
    const fileInfo = await FileSystem.getInfoAsync(dataFilePath);
    if (!fileInfo.exists) {
      throw new Error("Data storage file not found.");
    }


    // Read and parse the JSON file
    const fileContent = await FileSystem.readAsStringAsync(dataFilePath);
    const data = JSON.parse(fileContent);
```

```
    // Check if the user exists in the JSON
    if (!data[activeUser]) {
      throw new Error(`User '${activeUser}' not found in the data.`);
    }

    // Retrieve user data
    const userData = data[activeUser];

    // Check if userData array is empty
    if (!userData || userData.length === 0) {
      console.log("No dose data available for this user.");
      return false; // Return boolean if no data
    }

    // Retrieve the last dose time
    const lastDose = userData[userData.length - 1].dateTime;
    const lastDoseTime = new Date(lastDose);

    // Get the current time
    const currentTime = new Date();

    // Calculate the time difference in milliseconds
    const timeDiffMs = currentTime.getTime() - lastDoseTime.getTime();

    // Convert milliseconds to rounded minutes
    const timeDiffMinutes = Math.round(timeDiffMs / (1000 * 60));

    return timeDiffMinutes;
  } catch (error) {
    console.error("Error calculating time since last dose:", error);
    throw error; // Rethrow the error for upstream handling
  }
};
```

adafruitIO:

```
import axios from "axios";

// Define the types for the response data
interface AdafruitData {
  value: string;
  created_at: string;
  // updated_at: string;
}

// constant values (credentials)
const ADAFRUIT_IO_KEY = 'aio_rSMn16oXVn6HTUrOxx2Pp3jv1Gic';
const ADAFRUIT_IO_USERNAME = 'youngwyatt';

// Function to fetch a value from a feed key (input)
export const fetchAdafruitIOData = async (feedKey: string): Promise<AdafruitData | null> => {
  try {
    const response = await axios.get<AdafruitData>(
      `https://io.adafruit.com/api/v2/${ADAFRUIT_IO_USERNAME}/feeds/${feedKey}/data/last`,
      {
        headers: {
          'X-AIO-Key': ADAFRUIT_IO_KEY,
        }
      }
    );

    console.log('Fetched data from Adafruit IO:', response.data);
    return response.data;

  } catch (error) {
    console.error('Error fetching data from Adafruit IO:', error);
    return null;
  }
```

```
};


// Define the types for the response data
interface AdafruitResponse {
  id: string;
  value: string;
  created_at: string;
}


export const exportAdafruitIOData = async (value: number, feedKey: string): Promise<AdafruitResponse | null> => {
  try {
    const response = await axios.post<AdafruitResponse>(
      `https://io.adafruit.com/api/v2/${ADAFRUIT_IO_USERNAME}/feeds/${feedKey}/data`, // URL for sending data to Adafruit IO
      { value }, // Send the value as part of the request body
      {
        headers: {
          'X-AIO-Key': ADAFRUIT_IO_KEY, // Include the API key for authentication
        },
      }
    );


    console.log('Successfully posted data to Adafruit IO:', response.data);
    return response.data;
  } catch (error) {
    console.error('Error posting data to Adafruit IO:', error);
    return null;
  }
};


save_to_json:
import * as FileSystem from 'expo-file-system';
```

```
interface UserData {
  numPills: number;
  dateTime: string;
  pillsRecommended: number;
}

interface DataStorage {
  current_active_account: string; // Explicitly define this as always a string
  [userId: string]: UserData[] | string; // Allow userId keys or current_active_account
}

const dataFilePath = `${FileSystem.documentDirectory}data-storage.json`;

export const SaveData = async (
  userID: string,
  numPills: number,
  dateTime: string,
  pillsRecommended: number
) => {
  try {
    // Check if the file exists
    const fileInfo = await FileSystem.getInfoAsync(dataFilePath);
    let data: DataStorage = { current_active_account: userID };

    if (fileInfo.exists) {
      const fileContent = await FileSystem.readAsStringAsync(dataFilePath);
      data = JSON.parse(fileContent);
    }

    // If user already exists, append the new data to their list
    if (data[userID] && Array.isArray(data[userID])) {
      const newEntry: UserData = { numPills, dateTime, pillsRecommended };
      (data[userID] as UserData[]).push(newEntry);
    } else {
```

```
      // If user does not exist, create a new list with the first entry
      data[userID] = [{ numPills, dateTime, pillsRecommended }];
    }

    // Update the current active account
    data.current_active_account = userID;

    // Write updated data back to the file
    const formattedData = JSON.stringify(data, null, 2);
    await FileSystem.writeAsStringAsync(dataFilePath, formattedData, {
      encoding: FileSystem.EncodingType.UTF8,
    });

    console.log("Success", `Data for user ${userID} saved successfully.`);
  } catch (error) {
    console.error("Error saving data:", error);
  }
};

export const set_active_account = async (user_id: string): Promise<boolean> => {
  try {
    const fileInfo = await FileSystem.getInfoAsync(dataFilePath);
    let data: any = {};

    // Check if the file exists
    if (fileInfo.exists) {
      const fileContent = await FileSystem.readAsStringAsync(dataFilePath);
      data = JSON.parse(fileContent);
    }

    // Check if the user_id exists in the JSON data
    if (user_id != "") {
      if (!data[user_id]) {
        console.error(`User with ID ${user_id} does not exist.`);
```

```
      return false; // Indicate failure
    }
  }

  // Update or set the current_active_account field
  data.current_active_account = user_id;

  // Write the updated data back to the file
  const formattedData = JSON.stringify(data, null, 2);
  await FileSystem.writeAsStringAsync(dataFilePath, formattedData, {
    encoding: FileSystem.EncodingType.UTF8,
  });

  console.log(`Active account successfully updated to ${user_id}.`);
  return true; // Indicate success
 } catch (error) {
  console.error("Error setting active account:", error);
  throw error; // Propagate any unexpected errors
 }
};

export const logOutUser = async () => {
 try {
  // Read the current data from the file
  const fileContents = await FileSystem.readAsStringAsync(dataFilePath);
  const data = JSON.parse(fileContents);

  // Reset the active user
  data.current_active_user = ""; // Set the active user to an empty string

  // Write the updated data back to the file
  await FileSystem.writeAsStringAsync(dataFilePath, JSON.stringify(data, null, 2));
  console.log('Logged out successfully, current_active_user is reset.');
 } catch (error) {
```

```
    console.error('Error logging out:', error);
  }
  return null
};

export const get_active_account = async (): Promise<string | null> => {
  try {
    const fileInfo = await FileSystem.getInfoAsync(dataFilePath);

    // Check if the file exists
    if (!fileInfo.exists) {
      console.log('Error', 'Data storage file not found.');
      return null;
    }

    // Read and parse the file
    const fileContent = await FileSystem.readAsStringAsync(dataFilePath);
    const data = JSON.parse(fileContent);

    // Check if current_active_account exists
    if (!data.current_active_account) {
      console.log('No Active Account', 'There is currently no active account.');
      return null;
    }

    // Return the active account ID
    console.log(`Current active account: ${data.current_active_account}`);
    return data.current_active_account;
  } catch (error) {
    console.error('Error accessing active account:', error);
    return null;
  }
};
```

bottle_upright:

```
import React, { useState } from 'react';
import { View, Text, Image, TouchableOpacity, StyleSheet } from 'react-native';
import { fetchAdafruitIOData } from '@/constants/adafruitIO';
import { exportAdafruitIOData } from '@/constants/adafruitIO';


const BottleUpright = ({ onClose, onDataExtracted }: { onClose: () => void, onDataExtracted: (numPills: number, doseTime: string) => void }) => {
  const [isWaiting, setIsWaiting] = useState(false);
  const [numPills, setNumPills] = useState<number | null>(null);
  const [doseTime, setDoseTime] = useState<string | null>(null);
  const [isComplete, setIsComplete] = useState(false);


  const initiateProcess = async () => {
    try {
      await exportAdafruitIOData(1, 'takedose');
      setIsWaiting(true);


      const checkDoseSent = async () => {
        const result = await fetchAdafruitIOData('dosesent');
        return result && Number(result.value) === 1;
      };


      const interval = setInterval(async () => {
        const doseSent = await checkDoseSent();
        if (doseSent) {
          clearInterval(interval);


          const loadCellData = await fetchAdafruitIOData('loadcell');
          if (loadCellData && loadCellData.value !== undefined) {
            const numPillsValue = Number(loadCellData.value);
            setNumPills(numPillsValue);
            setDoseTime(String(loadCellData.created_at));
            onDataExtracted(numPillsValue, String(loadCellData.created_at)); // Pass data back to
```

```
CreateAccount
     }

     await exportAdafruitIOData(0, 'dosesent');
     setIsWaiting(false);
     setIsComplete(true);
    }
   }, 300);
  } catch (error) {
   console.error('Error during process:', error);
   setIsWaiting(false);
  }
 };

 return (
  <View style={styles.container}>
   <View style={styles.textBackground}>
    <Text style={styles.title}>
     After pressing continue the bottle will unlock. Take or store pills, then place on a flat surface.
    </Text>
   </View>
   <View style={styles.Background}>
    <Image source={require('@/assets/images/bottle.png')} style={styles.image} />
   </View>
   {!isWaiting && !isComplete && (
    <TouchableOpacity style={styles.continueButton} onPress={initiateProcess}>
     <Text style={styles.continueButtonText}>Continue</Text>
    </TouchableOpacity>
   )}
   {isWaiting && (
    <View style={styles.waitContainer}>
     <Text style={styles.waitText}>Please Wait...</Text>
    </View>
   )}
```

```
    {isComplete && (
      <TouchableOpacity style={styles.closeButton} onPress={onClose}>
        <Text style={styles.closeButtonText}>Close</Text>
      </TouchableOpacity>
    )}
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  position: 'absolute', // Added to prevent potential issues with zIndex stacking
  backgroundColor: '#f2f2f2',
 },
 textBackground: {
  position: 'absolute',
  top: 150, // Adjust as needed to place it within the container
  left: 10,
  backgroundColor: 'rgba(128, 128, 128, 0.8)', // Grey background for text
  padding: 15, // Increased padding for more space around the text
  borderRadius: 5,
  marginBottom: 20,
  width: 350,
  borderWidth: 2,
 },
 title: {
  fontSize: 18,
  fontWeight: 'bold',
  textAlign: 'center',
  color: 'white',
  lineHeight: 20, // Optional: Adjust line height for better readability
```

```
},
Background: {
  padding: 20,
  backgroundColor: '#fff',
  borderRadius: 10,
  borderWidth: 2,
  borderColor: '#b95fe0',
  position: 'absolute',
  left: 2,
  top: 140,
  height: 280,
  width: 370,
  zIndex: -1, // Ensure this is below the BottleUpright pop-up
},
image: {
  position: 'absolute',
  top: 100, // Position the image at the top within the container
  left: 130,
  width: 100,
  height: 150,
  resizeMode: 'contain',
  marginBottom: 20,
  zIndex: 1,
  borderWidth: 2,
},
continueButton: {
  position: 'absolute',
  top: 380, // Position the continue button at the bottom of the container
  left: 140,
  backgroundColor: '#ed810e',
  padding: 10,
  borderRadius: 5,
  zIndex: 1,
```

```
},
continueButtonText: {
  color: 'white',
  fontSize: 16,
  fontWeight: 'bold',
  zIndex: 1,
},
waitContainer: {
  position: 'absolute',
  top: 380, // Position the wait container just above the continue button
  left: 120,
  backgroundColor: '#D3D3D3', // Light grey for waiting
  padding: 10,
  borderRadius: 5,
  zIndex: 1,
},
waitText: {
  fontSize: 16,
  fontWeight: 'bold',
  color: '#000',
  zIndex: 1,
},
closeButton: {
  position: 'absolute',
  top: 380, // Position the close button at the top
  left: 150, // Align it to the right
  backgroundColor: '#2196F3',
  padding: 10,
  borderRadius: 5,
  zIndex: 1,
},
closeButtonText: {
  color: 'white',
  fontSize: 16,
```

```
    fontWeight: 'bold',
    zIndex: 1,
  },
});

export default BottleUpright;
create_account:
import React, { useState } from 'react';
import { View, Text, TouchableOpacity, StyleSheet, TextInput } from 'react-native';
import BottleUpright from '@/assets/pop_ups/bottle_upright'; // Import the BottleUpright component
import { SaveData } from '@/constants/save_to_json';

const CreateAccount = ({onClose}: {onClose: () => void;}) => {

  const [userID, setUserID] = useState(''); // State to store the userID
  const [numPills, setNumPills] = useState<number | null>(null); // State to store numPills
  const [doseTime, setDoseTime] = useState<string | null>(null); // State to store doseTime

  const [showUserIDPrompt, setShowUserIDPrompt] = useState(true); // Control for showing userID prompt
  const [showBottleUpright, setShowBottleUpright] = useState(false);

  // Function to handle the creation of the user account
  const handleCreateAccount = () => {
    setShowUserIDPrompt(true); // Show the userID input prompt
  };
  const handleCloseCreateAccount = () => {
    setShowUserIDPrompt(false); // Show the userID input prompt
  };

  // Callback function to handle the extracted data from BottleUpright
  const handleDataExtracted = (extractedNumPills: number, extractedDoseTime: string) => {
    setNumPills(extractedNumPills); // Store numPills
    setDoseTime(extractedDoseTime); // Store doseTime
```

```
  const pillsRecommended = 0;

  // Call SaveData to store the account data
  SaveData(userID, extractedNumPills, extractedDoseTime, pillsRecommended);

  // Log the extracted values to the console
  console.log(`Saved data userID: ${userID}, numPills: ${extractedNumPills}, doseTime: ${extractedDoseTime}, pillsRecommended: ${pillsRecommended} to data-storage.json`);
 };

 // Function to close the BottleUpright pop-up
 const handleBottleUpright = () => {
  setShowBottleUpright(true); // Close the BottleUpright pop-up
 };
 const handleCloseBottleUpright = () => {
  setShowBottleUpright(false); // Close the BottleUpright pop-up
 };

 return (
  <View style={styles.container}>
   {/* Show the "Enter your User ID" prompt when triggered */}
   {showUserIDPrompt && (
    <View style={styles.userIDPrompt}>
     <Text style={styles.promptText}>Enter a new User ID to Create Account</Text>
     {/* Input field for the User ID */}
     <TextInput
      style={styles.userIDInput}
      value={userID}
      onChangeText={setUserID}
      placeholder="Enter User ID"
     />
     <TouchableOpacity
      style={styles.doneButton}
```

```
          onPress={() => {
            handleCloseCreateAccount()
            handleBottleUpright()
          }}
        >
          <Text style={styles.doneButtonText}>Done</Text>
        </TouchableOpacity>
      </View>
    )}

    {/* Conditional rendering of the BottleUpright pop-up */}
    {showBottleUpright && (
      <BottleUpright
        onClose={handleCloseBottleUpright}
        onDataExtracted={handleDataExtracted} // Pass callback to extract data
      />
    )}

   </View>
  );

};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    position: 'absolute', // Added to prevent potential issues with zIndex stacking
  },
  createAccountButton: {
    position: 'absolute',
    top: 40, // Keeps it at 40px from the top
    right: 20, // Keeps it 20px from the left
```

```
  backgroundColor: '#b95fe0',
  borderRadius: 5,
  zIndex: 2, // Ensure it's placed above other elements
},
buttonText: {
  color: 'white',
  fontSize: 16,
  fontWeight: 'bold',
},
userIDPrompt: {
  padding: 20,
  backgroundColor: '#fff',
  borderRadius: 10,
  borderWidth: 2,
  borderColor: '#b95fe0',
  alignItems: 'center',
  justifyContent: 'center',
  position: 'absolute',
  left: 2,
  top: 150,
  zIndex: 1, // Ensure this is below the BottleUpright pop-up
},
promptText: {
  fontSize: 18,
  fontWeight: 'bold',
  marginBottom: 20,
},
userIDInput: {
  height: 40,
  borderColor: '#ccc',
  borderWidth: 1,
  width: '100%',
  paddingLeft: 10,
  marginBottom: 20,
```

```
  },
  doneButton: {
    backgroundColor: '#4CAF50',
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 5,
  },
  doneButtonText: {
    color: 'white',
    fontSize: 16,
  },
});

export default CreateAccount;

data_table:
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet } from 'react-native';
import * as FileSystem from 'expo-file-system';
import { DataTable } from 'react-native-paper';

// Path for the data file
const dataFilePath = `${FileSystem.documentDirectory}data-storage.json`;

const DataTableComponent = ({ userID }: { userID: string | null }) => {
  const [accountData, setAccountData] = useState<any[]>([]);

  // Get the current active account data from data-storage.json
  useEffect(() => {
    const fetchData = async (userID: string | null) => {
      let data: any = {};

      try {
        // Read data from data-storage.json
```

```
    const fileInfo = await FileSystem.getInfoAsync(dataFilePath);
    if (fileInfo.exists) {
      const fileContent = await FileSystem.readAsStringAsync(dataFilePath);
      data = JSON.parse(fileContent);
    }

    if (userID && data[userID]) {
      setAccountData(data[userID]); // Set the account data for the given user ID
    } else {
      console.warn('No data found for userID:', userID);
      setAccountData([]); // Reset if no data is found
    }
  } catch (error) {
    console.error('Error fetching data:', error);
  }
 };

 fetchData(userID);
}, [userID]);

// Format dateTime to a readable format
const formatDateTime = (dateTime: string) => {
 const date = new Date(dateTime);
 return date.toLocaleString(); // Format as "MM/DD/YYYY, HH:MM:SS AM/PM"
};

return (
 <View style={styles.container}>
  {/* Table Header using react-native-paper DataTable */}
  <DataTable style={styles.dataTable}>
    <DataTable.Header style={styles.tableHeader}>
      <View style={styles.headerTextWrapper}>
        <Text style={styles.headerText}>Time</Text>
      </View>
```

```
        <View style={styles.headerTextWrapper}>
          <Text style={styles.headerText}>Pills Recommended</Text>
        </View>
        <View style={styles.headerTextWrapper}>
          <Text style={styles.headerText}>Total Pills</Text>
        </View>
      </DataTable.Header>

      {/* Data Table Rows */}
      {accountData.length > 0 ? (
        accountData.map((item, index) => (
          <DataTable.Row key={index} style={styles.tableRow}>
            <DataTable.Cell>{formatDateTime(item.dateTime)}</DataTable.Cell>
            <DataTable.Cell>{item.pillsRecommended}</DataTable.Cell>
            <DataTable.Cell>{item.numPills}</DataTable.Cell>
          </DataTable.Row>
        ))
      ) : (
        <View style={styles.emptyView}>
          <Text style={styles.emptyText}>No data available for this User ID.</Text>
        </View>
      )}
    </DataTable>
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
  padding: 20,
  backgroundColor: '#f8f9fa',
 },
 dataTable: {
```

```
  borderRadius: 10,
  width: 300,
  //overflow: 'hidden',
},
tableHeader: {
  height: 80,
  backgroundColor: '#007bff', // Blue background for header
  borderBottomWidth: 2,
  borderBottomColor: '#ccc',
  flexDirection: 'row', // Ensure the header titles are placed horizontally
},
headerTextWrapper: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  paddingVertical: 8,
},
headerText: {
  color: '#fff',
  fontWeight: 'bold',
  textAlign: 'center',
  fontSize: 16,
},
tableRow: {
  borderBottomWidth: 1,
  borderBottomColor: '#ddd',
  backgroundColor: '#fff',
  height: 60,
},
emptyView: {
  alignItems: 'center',
  marginTop: 20,
},
emptyText: {
```

```
    color: '#6c757d',
    textAlign: 'center',
  },
});

export default DataTableComponent;

display_rectangle:
import React, { useEffect, useState } from 'react';
import { View, Text, StyleSheet } from 'react-native';
import { getTimeSinceLastDose } from '@/constants/account_state'; // Import the function

interface ConditionalDisplayRectangleProps {
    activeUser: string | null; // Allow activeUser to be null
    onAccountStateChange?: (state: 'red' | 'yellow' | 'green' | 'NoUser') => void; // Include NoUser state
  }

  const ConditionalDisplayRectangle: React.FC<ConditionalDisplayRectangleProps> = ({ activeUser, onAccountStateChange }) => {
    const [timeDiff, setTimeDiff] = useState<number | boolean>(false); // Time since last dose
    const [accountState, setAccountState] = useState<'red' | 'yellow' | 'green' | 'NoUser'>('NoUser'); // Default state is NoUser

    const t1 = 5; // Threshold 1 (4 hours)
    const t2 = 10; // Threshold 2 (6 hours)

    useEffect(() => {
      const fetchTimeDiff = async () => {
        if (!activeUser) {
          // Handle case when no user is logged in
          setAccountState('NoUser');
          if (onAccountStateChange) {
            onAccountStateChange('NoUser');
          }
```

```
      return;
    }

    try {
      const diff = await getTimeSinceLastDose(activeUser);
      setTimeDiff(diff);

      let newState: 'red' | 'yellow' | 'green' | 'NoUser' = 'green'; // Default state
      if (typeof diff === 'number') {
        if (diff < t1) {
          newState = 'red';
        } else if (diff < t2) {
          newState = 'yellow';
        }
      } else {
        newState = 'green'; // Fail-safe state
      }

      setAccountState(newState);

      if (onAccountStateChange) {
        onAccountStateChange(newState);
      }
    } catch (error) {
      console.error('Error fetching time difference:', error);
      setAccountState('green'); // Fail-safe: set state to green
      if (onAccountStateChange) {
        onAccountStateChange('green');
      }
    }
  };

  fetchTimeDiff();
}, [activeUser]);
```

```
const getDisplayText = () => {
  switch (accountState) {
    case 'red':
      return 'Red State: No Dosage is available at this time.';
    case 'yellow':
      return 'Yellow State: Dosage is available.';
    case 'green':
      return 'Green State: Dosage is recommended.';
    default:
      return '';
  }
};

if (accountState === 'NoUser') {
  // Do not render anything if no user is logged in
  return null;
}

return (
  <View
    style={[
      styles.rectangle,
      accountState === 'red'
        ? styles.red
        : accountState === 'yellow'
        ? styles.yellow
        : styles.green,
    ]}
  >
    <Text style={styles.text}>{getDisplayText()}</Text>
    {typeof timeDiff === 'number' && (
      <Text style={styles.text}>{`${timeDiff} minutes since last dose.`}</Text>
    )}
```

```
    </View>
  );
};

const styles = StyleSheet.create({
  rectangle: {
    width: '90%',
    padding: 20,
    borderRadius: 10,
    marginVertical: 10,
    justifyContent: 'center',
    alignItems: 'center',
    top: -80,
  },
  red: {
    backgroundColor: '#f44a4a', // Red color
  },
  yellow: {
    backgroundColor: '#ecd31d', // Yellow color
  },
  green: {
    backgroundColor: '#5cb139', // Green color
  },
  text: {
    fontSize: 16,
    fontWeight: 'bold',
    color: '#FFF', // White text
    textAlign: 'center',
  },
});

export default ConditionalDisplayRectangle;
```

log_in:

```
import React, { useState } from 'react';
```

```
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert } from 'react-native';
import * as FileSystem from 'expo-file-system';
import { set_active_account } from '@/constants/save_to_json'; // Import set_active_account

const LogIn = ({
  onLogIn,
  onClose,
}: {
  onLogIn: (userId: string) => void;
  onClose: () => void;
}) => {
  const [userId, setUserId] = useState('');
  const [errorMessage, setErrorMessage] = useState(''); // State to hold the error message

  const dataFilePath = `${FileSystem.documentDirectory}data-storage.json`;

  const handleLogIn = async () => {
    try {
      // Try setting the active account
      const success = await set_active_account(userId);

      if (!success) {
        // If the user ID does not exist, set an error message
        setErrorMessage("That User ID doesn't exist. Please Create an Account and try again.");
        return; // Exit early
      }

      // If login is successful, pass the userId to the parent component
      onLogIn(userId);

      // Close the pop-up
      onClose();
    } catch (error) {
      console.error('Error during login:', error);
```

```
    }
  };

  return (
    <View style={styles.container}>
      {/* Close Button */}
      <TouchableOpacity style={styles.closeButton} onPress={onClose}>
        <Text style={styles.closeButtonText}>X</Text>
      </TouchableOpacity>

      {/* Title */}
      <Text style={styles.title}>Enter your User ID to log in:</Text>

      {/* Input Field */}
      <TextInput
        style={styles.input}
        placeholder="Type your User ID"
        value={userId}
        onChangeText={setUserId}
      />

      {/* Error Message */}
      {errorMessage ? <Text style={styles.errorText}>{errorMessage}</Text> : null}

      {/* Log In Button */}
      <TouchableOpacity style={styles.logInButton} onPress={handleLogIn}>
        <Text style={styles.logInButtonText}>Log In</Text>
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
```

```
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 20,
    position: 'absolute',
    top: 170,
    left: 35,
    width: 300,
    height: 200,
    backgroundColor: '#bab7b9',
    borderWidth: 3
  },
  closeButton: {
    position: 'absolute',
    top: 10,
    right: 10,
    padding: 5,
    backgroundColor: '#ccc',
    borderRadius: 10,
  },
  closeButtonText: {
    fontSize: 16,
    fontWeight: 'bold',
    color: '#333',
  },
  title: {
    fontSize: 18,
    fontWeight: 'bold',
    textAlign: 'center',
    marginBottom: 20,
  },
  input: {
    width: '100%',
    height: 40,
```

```
    borderColor: '#ccc',
    borderWidth: 1,
    borderRadius: 5,
    paddingHorizontal: 10,
    marginBottom: 20,
  },
  errorText: {
    color: 'red',
    fontSize: 14,
    marginBottom: 10,
  },
  logInButton: {
    backgroundColor: '#4CAF50',
    padding: 10,
    borderRadius: 5,
  },
  logInButtonText: {
    color: 'white',
    fontSize: 16,
    fontWeight: 'bold',
  },
});

export default LogIn;

pain_level_input:
import React, { useState, useEffect } from 'react';
import { View, Text, Image, TouchableOpacity, StyleSheet } from 'react-native';
import { SaveData } from '@/constants/save_to_json';
import BottleUpright from '@/assets/pop_ups/bottle_upright'; // Import the BottleUpright component
import { get_active_account } from '@/constants/save_to_json';

const PainLevelInput = ({
  onClose,
```

```
  onSelect,
  accountState,
  userID,
}: {
  onClose: () => void;
  onSelect: (level: string) => void;
  accountState: 'red' | 'yellow' | 'green' | 'NoUser';
  userID: string | null;
}) => {
  const [selectedLevel, setSelectedLevel] = useState<string>(''); // Initially empty
  const [pillsRecommended, setPillsRecommended] = useState<number | null>(null); // Initially null
  const [showBottleUpright, setShowBottleUpright] = useState(false);
  const [numPills, setNumPills] = useState<number | null>(null);
  const [doseTime, setDoseTime] = useState<string | null>(null);
  const [showPainInputPrompt, setPainInputPrompt] = useState(true);
  const [showContinueButton, setShowContinueButon] = useState(false);

  // Function to handle the creation of the user account
  const handlePainInput = () => {
    setPainInputPrompt(true);
  };

  useEffect(() => {
    if (selectedLevel) {
      const recommendedPills = getRecommendedPills(selectedLevel);
      setPillsRecommended(recommendedPills);
    }
  }, [selectedLevel, accountState]);

  const getRecommendedPills = (level: string): number => {
    if (accountState === 'yellow') {
      return level === 'happy' || level === 'neutral' ? 1 : 2;
    } else if (accountState === 'green') {
      return level === 'happy' ? 1 : level === 'neutral' ? 2 : 3;
```

```
  }
  return 0;
 };

 const handleSelectPainLevel = (level: string) => {
  setSelectedLevel(level); // Update the selected pain level
  onSelect(level); // Pass the selected level to the parent
  setShowContinueButon(true)
 };

 const handleDataExtracted = (extractedNumPills: number, extractedDoseTime: string) => {
  setNumPills(extractedNumPills);
  setDoseTime(extractedDoseTime);
  SaveData(String(userID), extractedNumPills, extractedDoseTime,
Number(pillsRecommended));
  console.log(
   `Saved data userID: ${userID}, numPills: ${extractedNumPills}, doseTime:
${extractedDoseTime}, pillsRecommended: ${pillsRecommended} to data-storage.json`
  );
 };

 const handleCloseBottleUpright = () => {
  setShowBottleUpright(false);
  setPainInputPrompt(false)
 };

 return (
  <View style={styles.container}>
   {showPainInputPrompt && (
    <View style={styles.Background}>
     <TouchableOpacity style={styles.closeButton} onPress={onClose}>
      <Text style={styles.closeButtonText}>X</Text>
     </TouchableOpacity>
     <Text style={styles.promptText}>Please input your pain level.</Text>
```

```
      <View style={styles.imageRow}>
        <TouchableOpacity onPress={() => handleSelectPainLevel('happy')}>
          <Image source={require('@/assets/images/happy_face.png')} style={styles.image} />
        </TouchableOpacity>
        <TouchableOpacity onPress={() => handleSelectPainLevel('neutral')}>
          <Image source={require('@/assets/images/neutral_face.png')} style={styles.image} />
        </TouchableOpacity>
        <TouchableOpacity onPress={() => handleSelectPainLevel('sad')}>
          <Image source={require('@/assets/images/sad_face.png')} style={styles.image} />
        </TouchableOpacity>
      </View>
      <Text style={styles.pillsText}>
        Pills Recommended: {pillsRecommended !== null ? pillsRecommended : ''}
      </Text>

      {showContinueButton && (
        <TouchableOpacity
          style={styles.continueButton}
          onPress={() => {
            setShowBottleUpright(true);
            setPainInputPrompt(false);
          }}
        >
          <Text style={styles.continueButtonText}>Continue</Text>
        </TouchableOpacity>
      )}
    </View>
  )}
  {showBottleUpright && (
    <BottleUpright
      onClose={handleCloseBottleUpright}
      onDataExtracted={handleDataExtracted}
    />
  )}
```

```
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    position: 'absolute', // Added to prevent potential issues with zIndex stacking
  },
  Background: {
    padding: 20,
    backgroundColor: '#fff',
    borderRadius: 10,
    borderWidth: 2,
    borderColor: '#b95fe0',
    position: 'absolute',
    left: 2,
    top: 140,
    height: 250,
    width: 370,
    zIndex: -1, // Ensure this is below the BottleUpright pop-up
  },
  closeButton: {
    position: 'absolute',
    top: 10,
    right: 10,
    padding: 5,
    backgroundColor: '#ccc',
    borderRadius: 10,
  },
  closeButtonText: {
    fontSize: 16,
```

```
  fontWeight: 'bold',
  color: '#333',
},
promptText: {
  fontSize: 18,
  fontWeight: 'bold',
  textAlign: 'center',
  marginBottom: 20,
},
imageRow: {
  flexDirection: 'row',
  justifyContent: 'space-evenly',
  width: '100%',
},
image: {
  width: 60,
  height: 60,
  resizeMode: 'contain',
},
pillsText: {
  fontSize: 18,
  fontWeight: 'bold',
  textAlign: 'center',
  marginVertical: 10,
},
continueButton: {
  marginTop: 20,
  backgroundColor: '#4CAF50',
  padding: 10,
  borderRadius: 5,
},
continueButtonText: {
  fontSize: 16,
  fontWeight: 'bold',
```

```
    color: '#fff',
    textAlign: 'center',
  },
});

export default PainLevelInput;

History:
import React, { useState } from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';
import DataTable from '@/assets/pop_ups/data_table'; // Adjust the path as needed
import { get_active_account } from '@/constants/save_to_json';

const App = () => {
  const [showTable, setShowTable] = useState(false); // Toggle for showing the data table
  const [activeUser, setActiveUser] = useState<string | null>(null);

  const handleToggleTable = () => {
    setShowTable((prev) => !prev);
  };

  const fetchActiveUser = async () => {
    const user = await get_active_account();
    setActiveUser(user);
  };
  fetchActiveUser()

  return (
    <View style={styles.container}>

      <Text style={styles.title}>Welcome to Dose History! Active User: {activeUser}</Text>

      {/* Button to show/hide the DataTable */}
      <Button
```

```
        title={showTable ? "Hide Data Table" : "Show Data Table"}
        onPress={handleToggleTable}
      />

      {/* Conditionally render DataTable */}
      {showTable && (
        <View style={styles.dataTableContainer}>
          <DataTable
          userID={activeUser}
          />
        </View>
      )}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 20,
    backgroundColor: '#f8f9fa',
  },
  title: {
    position: 'absolute',
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
    textAlign: 'center',
    top: 50,
    backgroundColor: '#bab7b9',
    padding: 10,
  },
```

```
  subTitle: {
    fontSize: 18,
    marginBottom: 10,
    textAlign: 'center',
  },
  dataTableContainer: {
    marginTop: 20,
    width: '100%', // Ensures the DataTable takes up the full width
  },
});

export default App;

index:
import React, { useState, useEffect } from 'react';
import { View, Text, TouchableOpacity, StyleSheet, Alert } from 'react-native';

import { set_active_account, get_active_account } from '@/constants/save_to_json';

import CreateAccount from '@/assets/pop_ups/create_account'; // Ensure this import is correct
import LogIn from '@/assets/pop_ups/log_in'; // Import the LogIn component
import ConditionalDisplayRectangle from '@/assets/pop_ups/display_rectange';
import PainLevelInput from '@/assets/pop_ups/pain_level_input';

const BlankTab = () => {
  // Define Constants
  // Create Account

  const [accountInfo, setAccountInfo] = useState<{ userId: string, numPills: number, doseTime: string } | null>(null);
  const [showCreateAccount, setShowCreateAccount] = useState(false); // Control PainLevelInput visibility

  // Log In
```

```
const [showLogIn, setShowLogIn] = useState(false); // State to control LogIn visibility

// Display Active User
const [activeUser, setActiveUser] = useState<string | null>(null);
const [refreshKey, setRefreshKey] = useState(0); // State to track refreshes
const fetchActiveUser = async () => {
  const user = await get_active_account();
  setActiveUser(user);
};
// Display State
const [accountState, setAccountState] = useState<'red' | 'yellow' | 'green' | 'NoUser'>('NoUser');

// Take Dose Button
const [showPainLevelInput, setShowPainLevelInput] = useState(false); // Control PainLevelInput visibility
const [selectedPainLevel, setSelectedPainLevel] = useState<string | null>(null); // Store the selected pain level
const [showTakeDose, setShowTakeDose] = useState(false);

// Define Handles
// Create Account
const handleCreateAccountPress = () => {
  setShowCreateAccount(true); // Show the PainLevelInput pop-up
};
const handleCreateAccountClose = () => {
  setShowCreateAccount(false); // Hide the PainLevelInput pop-up
};
const handleAccountCreated = (userId: string, numPills: number, doseTime: string) => {
  setAccountInfo({ userId, numPills, doseTime});
  console.log(`Account created for ${userId} with ${numPills} pills.`);
};

// Log In
const handleCloseLogIn = () => {
```

```
  setShowLogIn(false); // Close LogIn pop-up
};
const handleLogIn = (userId: string) => {
  console.log('Logged in as:', userId);  // Handle successful login actions here (e.g., set active user, navigate to main screen, etc.)
};

// Log Out
const handleLogOut = async () => {
  try {
    // Set the active account to an empty string to log out
    await set_active_account("");
    setActiveUser(null); // Clear the active user in state
    console.log("Logged Out", "You have been successfully logged out.");
  } catch (error) {
    console.error("Error logging out:", error);
  }
};

// Display Active User
useEffect(() => {
  const fetchActiveUser = async () => {
    const user = await get_active_account(); // Get the active account using the function
    setActiveUser(user);
  };
  fetchActiveUser();
}, []);
useEffect(() => { // Fetch the active user whenever refreshKey changes
  fetchActiveUser();
  handleShowTakeDose();
}, [refreshKey]);
const handleRefresh = () => {
  setRefreshKey((prevKey) => prevKey + 1); // Increment refreshKey to trigger re-render
};
```

```
// Take Dose Button
const handleTakeDosePress = () => {
  setShowPainLevelInput(true); // Show the PainLevelInput pop-up
};
// Callback to handle when PainLevelInput is closed
const handlePainLevelInputClose = () => {
  setShowPainLevelInput(false); // Hide the PainLevelInput pop-up
};
const handlePainLevelSelect = (level: string) => {
  setSelectedPainLevel(level); // Store the selected pain level
  console.log(`Selected pain level: ${level}`); // Log for debugging
};
const handleShowTakeDose = () => {
  if (accountState === 'yellow' || accountState === 'green') {
    setShowTakeDose(true)}
  else {
    setShowTakeDose(false)
  }
}

return (
  // Create Buttons
  <View style={styles.container}>
    {/* Create Account Button */}
    <TouchableOpacity
      style={styles.createAccountButton}
      onPress={handleCreateAccountPress} // Show the modal when the button is pressed
    >
      <Text style={styles.createAccountButtonText}>Create Account</Text>
    </TouchableOpacity>

    {showCreateAccount && <CreateAccount
    onClose={handleCreateAccountClose}
```

```
/>}

{/* Log In Button */}
<TouchableOpacity
  style={styles.logInButton}
  onPress={() => setShowLogIn(true)} // Show LogIn pop-up
>
  <Text style={styles.logInButtonText}>Log In</Text>
</TouchableOpacity>
{/* Conditionally render LogIn pop-up */}
{showLogIn && <LogIn onLogIn={handleLogIn} onClose={handleCloseLogIn} />}

{/*Display Active User*/}
<TouchableOpacity style={styles.button} onPress={handleRefresh}>
  <Text style={styles.activeUserText}>
    {activeUser ? `Current Active User: ${activeUser}` : 'No Active User. Please Log in.'}
  </Text>
</TouchableOpacity>

{/* Log Out Button */}
<TouchableOpacity style={styles.logOutButton} onPress={handleLogOut}>
  <Text style={styles.logOutButtonText}>Log Out</Text>
</TouchableOpacity>

{/* Display State */}
<View style={styles.stateRectangle}>
  <ConditionalDisplayRectangle
    activeUser={activeUser}
    onAccountStateChange={(state) => setAccountState(state)}
  />
</View>

{/* Take Dose*/}
{showTakeDose && (<TouchableOpacity
```

```
      style={styles.takeDoseButton}
      onPress={handleTakeDosePress} // Show the modal when the button is pressed
     >
      <Text style={styles.takeDoseButtonText}>Take Dose</Text>
     </TouchableOpacity>
     )}

     {showPainLevelInput && <PainLevelInput
     onClose={handlePainLevelInputClose}
     onSelect={handlePainLevelSelect}
     accountState={accountState}
     userID={activeUser}
     />}

    </View>



  );
};

const styles = StyleSheet.create({
 // Design Button
 container: {
  backgroundColor: '#fff',
  position: 'absolute',
  top: 0,
  bottom: 0,
  left: 0,
  right: 0,

 },
 // Create Account
 createAccountButton: {
  position: 'absolute',
```

```
  top: 40, // Keeps it at 40px from the top
  left: 20, // Keeps it 20px from the left
  backgroundColor: '#d269ea',
  padding: 10,
  borderRadius: 5,
  zIndex: 1, // Ensure the button is on top of other elements
},
createAccountButtonText: {
  color: '#fff',
  fontSize: 16,
  fontWeight: 'bold',
},

// Log In
logInButton: {
  position: 'absolute',
  top: 40,
  right: 20,
  padding: 10,
  backgroundColor: '#33a2ff',
  borderRadius: 5,
},
logInButtonText: {
  color: 'white',
  fontSize: 16,
  fontWeight: 'bold',
},

// Log Out
logOutButton: {
  position: 'absolute',
  top: 85,
  right: 20,
  padding: 10,
```

```
  backgroundColor: '#f44a4a',
  borderRadius: 5,
  justifyContent: 'center',
  alignItems: 'center',
},
logOutButtonText: {
  color: 'white',
  fontSize: 16,
  fontWeight: 'bold',
},

// Display Active User
button: {
  padding: 10,
  backgroundColor: '#9e9e9e',
  borderRadius: 10,
  justifyContent: 'center',
  alignItems: 'center',
  position: 'absolute',
  bottom: 300,
  right: 90,
  zIndex: -1, // Moves this to the background

},
activeUserText: {
  color: 'white',
  fontSize: 16,
  fontWeight: 'bold',
  zIndex: -1, // Moves this to the background

},

// State Rectange
stateRectangle: {
```

```
    position: 'absolute',
    zIndex: -1,
    bottom: 120,
    right: 5,
  },

  // Take Dose
  takeDoseButton: {
    position: 'absolute',
    width: 150,
    height: 150,
    borderRadius: 75, // Circular button
    backgroundColor: '#e046b8', // Green color for the button
    justifyContent: 'center',
    alignItems: 'center',
    bottom: 40,
    right: 115,
    borderWidth: 3,
  },
  takeDoseButtonText: {
    fontSize: 18,
    fontWeight: 'bold',
    color: '#fff',
  },
});

export default BlankTab;
```

Source Code 4: All code required for React Native Expo app software using typescript for the doseX device.

Biomedical Engineering 3801/4801 Biomedical Engineering Design
Design Verification Plan

Pharmacokinetic Control of Opioid Dosing
Revision: #7
12/2/24

Name	Position	Signature	Date
Jude Werth	Team leader, Code and Software	JW	08/22/24
Kyle Harshany	Design Engineer, Mechanical and CAD Modeling	KH	08/22/24
Wyatt Young	Code and Software, Market Researcher	WY	08/22/24
Daniel Candland	Quality Engineer, Electronics and PCB	DC	08/22/24
Parker Mason	Teaching assistant		
Tomasz Petelenz	Instructor		

Revision Log			
Revision	Description	Initials	Date
1	Initial release	WY	08/22/24
2	Updated Design Verification Test Methods	WY	9/24/24
3	Updated Risk Analysis, Design Requirements and Specification, Acceptance Criteria	DC	9/26/24
4	Updated Statistical Criteria	WY	9/28/24
4	Updated Design Verification Test methods	WY/KH	10/3/24
5	Justification of the test conditions and updated traceability matrix	JW/DC	10/6/24
6	Updated Statistical Criteria	KH	10/31/24
7	Added verification results as well as updated FMEA table and discussion of design results.	WY	12/2/24

1 Introduction
The purpose of this document is to indicate the general process and follow the documentation required to establish a Design Verification Plan and perform Design Verifications of new and updated designs for the doseX pharmacokinetic control device for opioid dosing for BME 4801 of Fall 2024.

2 Scope
The scope of this document pertains to any activities related to the verification testing of the doseX pharmacokinetic dosing control device during fall 2024.

3 References
3.1 External References
- FDA 21 CFR 820 Subparts C, F, and G – Design Control, Identification and Traceability, and Production and Process Controls
- Design Control Guidance for Medical Device Manufacturers, available from CDRH

3.2 Internal References

- Design and Development Plan v5
- Design Specifications Document v3
- Design Control - DC 04 24 - Design Verification rev 07 3/27/2023
- Design Control - DC 04 01 - Design Control Policy - rev 03 01/22/2021
- Design Control - DC 04 22 - Prototype Development and Documentation rev 01 11/29/2016

4 Responsibility-Identification of team responsibilities

Entire Team: All members will be expected to understand the verification testing process and are expected to document all activities pertaining to verification testing on this document.
Wyatt and Jude: Wyatt and Jude are responsible for activities related to software and select electrical hardware.
Daniel and Kyle: Daniel and Kyle are responsible for activities related to the physical construction and mechanics of the device.

5 Safety considerations when conducting verification testing

There is no foreseeable safety risks associated with the verification testing planned, however safety considerations will be continuously monitored and updated. Any new risks that may arise will be evaluated and planned for, with safety plans updated in this document prior to testing.

6 Design Verification Activities (Individuals responsible are listed in parentheses.) Use Tables provided in the DC 04 24 – Design Verification rev.05 8-10-20

6.1 Provide Updated Risk Analysis to reflect current design (provide updated FMEA table)

Process Step/Input	Potential Failure Mode	Potential Failure Effects	SEVERITY (1 - 10)	Potential Causes	OCCURRENCE (1 - 10)	Current Controls		RPN	Risk Mitigation Action	SEVERITY (1 - 10)	OCCURRENCE (1 - 10)	RPN
What is the process step, change or feature under investigation?	In what ways could the step, change or feature go wrong?	What is the impact on the customer if this failure is not prevented or corrected?		What causes the step, change or feature to go wrong? (how could it occur?)		What controls exist that either prevent or detect the failure?			What are the recommended actions for reducing the occurrence of the cause or improving detection?			
App Software	Fails to connect to patients pill bottle device	Patient cannot use accompanying app	***1***	Bad patient Wifi router, bad Wifi radio on microprocessor.	***6***	Ensuring app software properly connects with Wifi module. Ensuring patient is aware they need a Wifi connection		***6***	Include in IFU that patient must have Wifi and how to connect device to app. Test microcontroller Wifi module function.	***1***	***1***	***1***
App Software	Incorrectly queries data from server	Inaccurate load cell and dosage data leading to innacurate doses for patient.	4	Code bugs in query code in app software.	6	Ensuring app software code is bug free and samples sensors from IO server correctly.		***24***	Test app software connection with IO server and accuracy of data queries	4	1	***4***

Pharmacokinetic Control of Opioid Dosing

Revision: #7

12/2/24

App Software	Doesn't properly calculate dose size.	Patient given inaccurate Doses	3	Dosage algorithm fails; inaccurate weights on each factor, or breaks when running resulting in 0 or NaN type.	4	Testing algorithm and adjusting weights to get validated dose size outputs in variety of scenarios.		***12***	Test algorithm through simulated patient chronic opiate prescription.	3	3	***9***
App Software	Allows “Take Dose” action to be performed within 4 hours of previous dose.	Patient given unsafe amounts of drug.	8	App software not properly timing periods between doses.	3	App software unit testing. App software/microcontroller code black box testing.		***24***	Proper unit/bug testing of software and validation of input/output testing.	8	3	***24***
Microcontroller: Wifi Module	Fails connection with IO server.	Patient cannot use app or device.	3	Wifi Module disconnected from microcontroller.	1	Testing device connection prior to packaging.		***3***	Test microcontroller Wifi module function. Add reconnect protocols for accidental disconnects.	3	1	***3***
Microcontroller: Charging Circuit	Fails to properly manage battery charging.	Undercharged device, or overcharged device.	2	Charging circuit fails to charge and recharge connected battery.	6	Proper power analysis and testing of battery life over time.		***12***	Conduct proper power analysis of device over functionality and time.	3	2	***6***
Microcontroller: Power Management	Fails to properly distribute battery power across sensors and processes.	Undercharged but still recharges.	2	Incorrect power management in circuitry of connected electronics in device.	3	Proper power analysis and testing of battery life over time.		***6***	Conduct proper power analysis of device over functionality and time.	2	3	***6***
Microcontroller: micro-USB Port	Fails to accept charge from USB cable	Device never recharges. Patient cannot use device.	5	USB port dislodges from microcontroller connection or becomes faulty from port lodged debris	4	Ensuring device connects with USB cable prior to packaging.		***20***	Test device to ensure stable and repeatable connection to micro-USB port.	4	2	***8***
Battery	Doesn't provide enough current to each process (fails)	Patient cannot use device.	5	Incorrect power management in circuitry of connected electronics in device.	7	Proper power analysis and testing of battery life over time.		***35***	Conduct proper power analysis of device over functionality and time.	5	3	***15***
Hardware Code	Doesn't sample load cell at correct time (synchronization error)	Patient given inaccurate count of remaining pills	2	Untested hardware code leading to bugs and loss of function.	8	Testing load cell output with different levels of pills		***16***	Testing load cell output when pill compartment is filled then slowly decreased by 1-4 pills at a time.	2	4	***8***

Pharmacokinetic Control of Opioid Dosing
Revision: #7
12/2/24

Hardware Code	Fails to unlock solenoid gate upon request. (logic error)	Patient cannot access pill compartment	6	Untested hardware code leading to bugs and loss of function.	8	Testing solenoid locking mechanism.		***48***	Test lock/unlock mechanism activated from app software for many more cycles than expected from patient use.	6	2	***12***
Load Cell	Accuracy of load cell is effected over time: Drift	Patient given inaccurate count of remaining pills	2	Load cell accuracy not gauged over long periods of time.	8	Testing load cell accuracy over long time intervals.		***16***	Test load cell over time period greater than a full chronic opiate prescription (>2 weeks- 1 month)	2	5	***10***
Load Cell	Inaccurate data output.	Patient given inaccurate count of remaining pills. Or no count output whatsoever.	2	Inaccurate load cell calibration. 24-bit ADC inaccurately divides voltage. Load cell loaded to greater than 100 g of pills causing break in resistances.	5	Ensure generated calibration curve used is accurate. Ensure capacity of pill storage cannot fit >100g of pills.(~307 pills)		***10***	Test load cell at a range of values both within expected range and outside expected range after calibration.	2	1	***2***
Lid and Locking Mechanism	Lid can be forced off when locked	Patient has access to pills when locked	5	Flexing or damage to the lid or bottle	5	Ensure components involved in locking the bottle are protected		***25***	Test lid and locking mechanism and redesign if necessary	5	1	***5***
Push-Pull Solenoid	Solenoid pin becomes jammed.	Patient either unable to lock or unlock pill storage	5	Inaccurate alignment of solenoid pin with cap hole.	5	Determine how solenoid pin could be dislodged or voltage step-up connection fails.		***25***	Drop test device to ensure that once assembled, solenoid pin cannot be dislodged and that electrical connections remain in place.	5	2	***10***
Emergency Override Button	Failure to register presses	Patient could lose access to pill storage if connection to server or app fails.	7	Faulty electrical connection with microcontroller.	3	Determine where button wiring could rub against device causing a disconnect.		***21***	Drop test device to ensure that once assembled, electrical connections from button remain in place. Ensure wiring has appropriate path to microcontroller.	7	1	***7***
LED Indicator Light	Doesn't update with new dose	Patient unable to see what "state" their prescription is in from device.	1	code logic used in hardware fails to switch status at given time intervals	6	Review microcontroller code logic in every possible iteration.		***6***	test microcontroller code logic in every possible state of time relative to a dose taken.	1	2	***2***

LED Indicator Light	Fails to blink red when battery is less than 10%	Patient would not realize device is low on battery.	1	Code logic fails to blink red light or fails to detect current battery level.	6	Review microcontroller code logic when battery power is around 10%.		***6***	Test microcontroller code logic at varying battery levels, especially right around 10%.	1	2	***2***
LED Indicator Light	Too dim to be seen	Patient would be unable to track dosage timing using LED	1	LED not properly lightpiped to surface. LED extrusion not large enough for adequate light to escape.	2	Ensure LED casing is viewed in a dark room by multiple people.		***2***	Test LED light emittance in a dark room by multiple people to ensure visibility.	1	2	***2***
LED Indicator Light	Fails to turn on	Patient would not know if device is operating.	2	Faulty LED/LED wiring or hardware fails to address and activate it when powered up.	3	Test LED function and ensure microcontroller can properly activate LED.		***6***	Test LED function prior to packaging as well as hardware code to ensure it can properly activate LED.	2	1	***2***
6 V step up converter	Step up converter overheats or wiring disconnects causing inactivation of solenoid.	Patient unable to unlock pill storage.	5	Inadequate venting or connection of step up.	5	Determine how solenoid pin could be dislodged or voltage step-up connection fails.		***25***	Run > 10 repetitions of unlocking solenoid via the voltage step up to ensure overheating doesn't occur. Drop testing as outlined above.	5	2	***10***

6.2 Design Requirements and Specification (provide references to each)

Smartphone App

- Successfully connects to bottle
- Able to lock, unlock, and probe weight
- Stores dosage history
- Utilize dosing algorithm and make recommendations.

Lid and Locking Mechanism

- Keep bottle closed and pills inside
- Capable of locking/unlocking pill bottle and lid.

Electrical Hardware

- Low-Capacity load cell capable of accurately measuring bottle contents
- Battery provides energy to the device, needs to be rechargeable.
- Power LED and control states
- Listen for emergency button presses and trigger a series of actions in response to an emergency button press (or a trigger from the app software)
- Capable of driving solenoid locking mechanism

Pill Container and Housing

- Hold an entire prescription (~32 pills)
- Be able to withstand everyday heat and force stress.
- The bottle will not be reactive with the pills
- Hold electric components on the device.

6.3 Provide pre-determined acceptance criteria

6.3.1 Need to address all items with their failure modes and effects identified in the risk analysis plus prototype element specifications

Table 1. Verification Test Summary Table

Test #	Prototype Element		Specification	Acceptance Criteria
1	Smartphone App	Patient Feedback	Patients must be able to enter their current pain level into app software via a pain scale from 0-2.	App software passes all algorithm input/output unit testing. Pass if all test inputs result in expected outputs. (Pass/Fail)
2		Dosage Calculations	Pharmacokinetic release algorithm based on patient feedback.	App software passes all algorithm black box tests, ensuring correct dosage recommendation based on pain level input and pharmacokinetic algorithm. Pass if correct dosage is recommended for N=20 test cases and no unexpected outputs. (Pass/Fail)
3		Electrical Control Circuit /Prototype App Interface	Patients must be able to access companion app software.	Wifi radio connects to a valid ssid network then attempts reconnection every 5 seconds. N=5 networks/hotspots, no failures upon reconnection. (Pass/Fail)
4				Microcontroller code successfully performs all ‘Take Dose’ actions, including solenoid unlock, LED state update, and data transmission to Adafruit IO. Pass if N=20 unit tests result in the expected outputs. (Pass/Fail)
5		History Data Collection	App must be able to store and display dosage history	Microcontroller uploads valid/accurate load cell data to Adafruit IO within 2 minutes of a dose.

6				App software passes all dosage history unit tests with proper timestamping in UTC. N=4 tests, no unexpected outputs. (Pass/Fail)
7		Dosage Control	Device will not allow dosing while unsafe	"Take Dose" app button becomes inactive within 5 seconds of dose confirmation. (Pass/Fail)
8	Lid and Locking Mechanism	Lid-Bottle Tolerance	Tolerance ≤ 0.0313 inches ± 0.0157in	ΔDiameter = 0.0469-0.0157 in
9		Locked lid cannot be forced open	Able to withstand 3N*m for 5 seconds	No breaking after 5 seconds under 3N*m of force.
10		Locks successfully	Solenoid releases and locks the lid	Locks properly 95% of the time
11	Electronic Control Circuitry	Load Cell	Can differentiate between three weights, 1g, 2g, and 5g, recording 10 measurements each.	Difference between measurement sets(5 at each weight) for each weight is statistically significant p ≥ 0.001
12		Battery	Outputs 3.7V ± 0.2V DC	3.5-3.9V DC Output over 6 hours with 200 mA current draw(for 1200 mAh LiPo).
13		Emergency Button	Patients must be able to access pill compartment without software intervention in case of emergency (phone loses power/connection).	Button properly unlocks pill compartment with no app connection and with < 10% battery. N = 20 iterations no unintended locking.
14		Emergency Button	The button will be low profile in its color and extrusion above the bottle surface (~2 mm).	Drop test from 1 meter height. Accidental activations are defined as any unintentional solenoid activation during drop or immediately after drop. No accidental activations in 90% of trials after N=15 drops from a height of 1 meter.
15		LED Indicator	An LED indicator light must be visible by users that tells patients their current dosage status.	Microcontroller successfully cycles LED through all states (green, yellow, red). N=5 unit tests, no unexpected states. (Pass/Fail)
16		LED Indicator	luminous intensity range of 100 millicandelas (mcd) ± 20 mcd for clear visibility in dark environments.	LED outputs an intensity range of 100 mcd ± 20 mcd. N=10 sensor readings within accepted range.
17	Pill	Pill Capacity of	Internal dimensions of radius =	r = 0.57-0.58in, h ≥ 1.75

	Container and Hardware Housing	30 pills	0.575±0.005 in, height ≥ 1.75 in	
18		Electrical Component Housing	Housing must be ≥ 2.73 in by 1.4 in by 1.5 in	≥ 2.73 in by 1.4 in by 1.5 in

6.3.2 Provide statistical criteria for each test. The criteria need to be precise, quantifiable, and unambiguous including appropriate statistical approach. Please provide a description of how you plan to handle variable and attribute data sets.

Table 2. Summary of Statistical Methods

Test #	Test method	[n]	[m]	DOF	Risk Likelihood	Conf level (alpha)	Single or Double Sided	Stat Test	Test Critical Parameter(s))
1	Software Unit Testing: Dosage Algorithm	1	10	9	4	N/A	N/A	Attribute Test	Correct outputs based on inputs.
2	Software Black Box Testing	1	10	9	4	N/A	N/A	Attribute Test	Correct outputs based on inputs.
3	Unit test: Wifi Connection	1	5	4	6	N/A	N/A	Attribute Test	Wifi connects and properly attempts reconnection.
4	Unit Tests: “Take Dose”	1	20	19	3	N/A	N/A	Attribute Test	Correct actions triggered by app
5	Load Cell Sampling	1	10	9	8	0.0025	Double	T-Test	Load Cell Feed Updates every 60 seconds ± 10 seconds
6	Unit Tests: App Dosage History	1	4	3	6	N/A	N/A	Attribute Test	App front end displays timestamped and validated dosages.
7	Unit Testing: Tracking time between doses	1	5	4	3	0.00005	Single	T-Test	App doesn’t allow for activation of “Take Dose” button within 4 hours ± 10

									minutes of previous dose.
8	Lid-Bottle Tolerance Measurement	1	15	14	5	0.0025	Double	T-test	ΔDiameter ≤ 0.0313 inches ±0.0156 inches
9	Locked Lid Torque Test	1	20	19	3	N/A	N/A	Attribute Test	No breaking after 5 seconds under 3N*m of force(pass/fail)
10	Solenoid Lock Functionality Test	1	30	29	5	0.05	Single	Proportion Test	Locks successfully 95% of the time
11	Load Cell Weight Differentiation Test	1	15	14	3	0.05	Double	ANOVA	Significant differences between 1g,2g ang 5g measurements. 5 sensor readings at each weight.
12	Battery Voltage Output Test	1	20	19	5	0.0025	Double	T-test	Battery output voltage between 3.5V and 3.9V at 10 minute splits over 6 hours.
13	Unit Testing: Tactile Button Activation	1	20	19	3	N/A	N/A	Attribute Test	Button activates Dose processes with no connection.
14	Drop Tests 1 m height	1	15	14	7	0.05	Single	Proportion Test	<10% accidental activations from series of drops.
15	Microcontroller Unit Tests: LED state cycling/sampling	1	5	4	6	N/A	N/A	Attribute Test	LED properly cycles and updates through states.
16	LED viewing testing	1	15	14	2	0.00001	Double	T-test	LED output 100 mcd ± 20 mcd
17	Pill Container Dimension Measurement	1	15	14	3	0.00005	Single	T-test	Internal Radius= 0.575 ±0.005in,

									height ≥1.75 in
18	Electrical components housing dimension test	1	10	9	3	0.00005	Single	T-test	Housing dimensions ≥ 2.73 in x 1.4 in x 1.5 in

6.4 Provide a description of Design Verification test methods and conditions (must be approved before testing commences)

6.4.1 Verification test methods include specific, step-by-step instructions for performing tests to determine if a design meets its predetermined acceptance criteria.

Tests 1,2,6,7

Software Unit testing will also be performed in the same manner as the microcontroller code but will be written in JavaScript. This will cover tests 1, 6, and 7. We will also be using Black Box testing in order to test the algorithm and appropriate outputs of the algorithm based on varying inputs. This covers test 2(and 1 partly).

Tests 3, 13, 15

All Unit Testing performed on the microcontroller will be done in Python. A suite of tests will be developed based on the importance of the specific function being tested(designated by N, in the above tables). Unit tests will be ran to determine bugs/unintended behavior in the code by using various inputs and checking for expected outputs. This unit testing will cover tests 3 and 15. We will also be performing stress unit tests (Test no. 13) to ensure button activation. This will repeatedly trigger the button to ensure the solenoid properly activates. There will be the software component of these tests as well as a physical component ensuring physical activation of the button has the same expected outputs.

Test 4

Microcontroller code validation will be done by passing code.py and states.py into OpenAI's ChatGPT model 4o to determine errors with the code. If errors are found by the model the code will be updated accordingly and tested after each change to ensure the suggestions validity. Furthermore, to ensure real use conditions the "Take Dose" actions in code will be ran repeatedly ensuring consistent solenoid locking as well as load cell sampling and upload to IO.

Test 5

In order to test the speed at which the microcontroller uploads its load cell sample to the IO network, a repeatable test involving taking a load cell sample and uploading it via the MQTT client to IO will be performed. The code used in the test will be identical to the code used under actual use conditions besides a timing loop. Once a load cell sample is taken, the timing loop will start to measure the time till the LoadCell feed in IO is updated by printing the 'time since upload' to the console. The point at which IO refreshes with the data will be confirmed visually and the time for that iteration will be recorded.

Tests 8,17,18

The Lid-Bottle Tolerance Measurement (8) will measure the diameter of the bottle's opening and the lid using a micrometer. Following data collection, assessment of whether the measurements fall within the specified tolerance limits is evaluated. The same process will be repeated for the pill container dimension measurements (17) and the electrical components housing dimension test (18).

Test 9

The Locked Lid Torque Test (9) will involve a torque wrench applying a force of 3 N·m to the lid for 5 seconds. The lid will be monitored for any signs of breaking or failure. After the 5-second period, the outcome is documented as a pass if there is no breaking, or a fail if the lid exhibits any failure during the test.

Test 10

The Solenoid Lock Functionality Test (10) involves initiating locking cycles by sending a signal to engage the lock, counting each successful operation, defined as the solenoid fully engaging within a specified timeframe. After the locking cycles, unlocking cycles are performed by sending a signal to disengage the lock, with success rates recorded similarly. Any failures during these processes are documented, including the cycle number, and the nature of the failure (e.g., failure to engage/disengage).

Test 11

The Load Cell Differentiation Test requires the load cell to be screwed in via its through holes on the end connected to the 4 wires. Once properly stabilized, zero out the ADC/Load Cell using the ADC driver code and repeat this for each weight standard. Hang 1 g, 2 g, and 5 g weights (use the same weight for each iteration) and record the outputted counts value from the ADC on the IDE console. Repeat this 5 times for each weight standard. If a count value outputted is greater than a magnitude from either a previous sample of the same weight or is completely out of a feasible range, re-zero the ADC and retry.

Once data is collected, an online ANOVA one-sided calculator is used with an alpha of 0.05 in order to determine the statistical difference between the outputted counts and the associated weight standards.

Test 12

The Battery Output Voltage Test involves simply hooking up each terminal of the battery to an oscilloscope and then measuring the voltage output over time. In order to consistently draw current, a 20 ohm resistor is placed between the terminals. Every 10 minutes, record the voltage output of the battery for up to 6 hours(or until voltage drops below 3.5 V to ensure no damage to battery).

Test 14

The Drop Test (14) will assess the product's durability by dropping it from a height of 1 meter to determine its ability to withstand impacts without accidental activation. The testing surface will consist of a hard, flat material, such as concrete or a hardwood floor, to simulate real-world environments. Monitoring for any accidental activations, physical damage, or malfunction will occur.

Test 16

The LED Light Emittance Test will be performed using the smartphone app StellarRAD. In a dark room environment, ensure the light sensor is accurate by checking it against a known light source. On the app, select the intended color to test and ensure LED is set to that color and that the brightness parameter used is correct for actual use case (brightness=0.6 for NeoPixel). Position the sensor directly to the LED from a 1 meter distance and record the luminous intensity reading from the app.

6.4.2 Written justification of the test conditions and how and why they differ (when these differ) from those recommended in standards.

Our device will be used directly by patients which can differ significantly based on the individual. Therefore, we need to design a product which can perform its function in a variety of situations. We designed our components to each perform simple functions which can interact with each other intendent of the patient. Our test conditions were created with this in mind. Individual functions were often left to a pass fail test while the tests which involve the components interacting are given statistical parameters with lower alpha levels.

6.5 Provide Verification to Spec traceability matrix (note: this is a new matrix, not the DR to Spec matrix)

Change Specification description to Specification Number

Specification:	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18
5.2.1 Section 1 #5 Patients must be able to enter their current pain level into app software via a pain scale from 0-2.	X																	
Pharmacokinetic release algorithm based on patient feedback.		X																
5.2.1 #Section 1 #4 Patients must be able to access companion app software.			X	X														
App must be able to store and display dosage history					X	X												
Device will not allow dosing while unsafe							X											
5.1.7 Tolerance ≤ 0.0313 inches ±0.0156 inches								X										
Able to withstand									X									

3N*m for 5 seconds																		
5.2.1 Section 2 #5 Solenoid releases and locks the lid										X								
Can differentiate between three weights, 1g, 2g, and 5g, recording 10 measurements each.											X							
5.2.1 Section 2 #7 Outputs 3.7V ± 0.2V DC												X						
5.2.1 Section 2 #2 Patients must be able to access pill compartment without software intervention in case of emergency (phone loses power/connection).													X					
5.2.1 Section 2 #2 The button will be low profile in its color and extrusion above the bottle surface (~2 mm).														X				
5.2.1 Section 2 #4 An LED indicator light must be visible by users that tells patients their current dosage status as well as indicate low battery.															X			
5.2.1 Section 2 #4 Luminous intensity range of 100 millicandelas (mcd) ± 20 mcd for clear visibility in dark environments.																X		
5.2.1 Section 2 #8 Internal dimensions of radius = 0.575±0.005 in,																	X	

height ≥ 1.75 in																			
.2.1 Section 2 #8 Housing must be ≥ 2.73 in by 1.4 in by 1.5 in																			X

7 Design Verification Reports must be generated for each testing activity

Table 3: Results Summary

Test #	Prototype Element	Acceptance Criteria	Test Result	Tolerance Interval for Sample	Test Evaluation Pass/Fail
1	Smartphone App	Pass/Fail	The pain level input is accurately stored in each of the 10 trials.	N/A	Pass
2	Smartphone App	Pass/Fail	Given a pain-input and account state the algorithm recommended the correct number of pills 10 out of 10 times.	N/A	Pass
3	Smartphone App	Pass/Fail	Wifi radio was able to connect to 5 valid networks/hotspots. If not connected on first attempt, each network was connected to within 3 attempts.	N/A	Pass
4	Smartphone App	Pass/Fail	Microcontroller code successfully triggered Dose and uploaded load cell reading to IO greater than 20 times. For the designated test samples proper actions occurred all 20 times. Code was validated via ChatGPT to ensure no bugs.	N/A	Pass
5	Smartphone App	Uploads < 1min	Data was accurate and received by IO server well within 60 seconds for all 10 samples.	(-1.236 sec, 4.214 sec)	Pass

6	Smartphone App	Pass/Fail	In each of our 4 trials, the time stamp of data extracted from adafruitIO matched the timestamp stored within the app.	N/A	Pass
7	Smartphone App	Pass/Fail	In each of the 5 trials, the Take Dose button became visible exactly at 2 minutes in every trial (t1 variable shortened for convenience)	[2,2] minutes	Pass
8	Lid and Locking Mechanism	Diameter= 0.0313±0.016 in	The distance between the inner diameter of the pill container the outer diameter of the lid nearly matched the acceptance criteria.	[0.021, 0.055]	Fail
9	Lid and Locking Mechanism	Time under stress > 2s	The locking mechanism did not fail at all across 15 trials.	100%	Pass
10	Lid and Locking Mechanism	Mean solenoid activation > 95%	Locked/relocked 97% of time over 30 signals sent by microcontroller	Lower Bound: 91.28%	Pass
11	Electronic Control Circuitry	Load Cell differentiates between 1g, 2g, and 5g weights	Load Cell outputted statistically different counts for each of the weight standards.	P<0.0001	Pass (p<0.0001)
12	Electronic Control Circuitry	3.5-3.9V DC Output			
13	Electronic Control Circuitry	Pass/Fail	Tactile button triggered take dose bottle state with no connection to IO 100% of the time over 20 samples.	?	Pass
14	Electronic Control Circuitry	Pass/Fail			
15	Electronic Control Circuitry	Pass/Fail	LED properly cycled from green(dose) to red(DS) to yellow(LS) every time without fail. It		Pass

			also properly updated from red to green when button was pressed. Greater than 20 samples.		
16	Electronic Control Circuitry	Pass/Fail			
17	Pill Container and Hardware Housing	r = 0.57-0.58in, h ≥ 1.75	Tolerance dimensions were met according to specifications	[0.562, 0.579]	Pass
18	Pill Container and Hardware Housing	≥ 2.73 in by 1.4 in by 1.5 in	Tolerance dimensions were met according to criteria	[2.76 2.92] [1.51 1.61] [1.42 1.49]	Pass

7.1 Discuss each acceptance criteria test result

- Describe whether the acceptance criteria were passed or failed for each test (include your statistical justification as part of this discussion)
- Include all test results with means and standard deviations as appropriate (include all test data in appendix)
- Describe how the measurement technique may have impacted the result
- Provide your planned remedies for each failed test

Test no. 1:

- The acceptance criterium is based on 10 trials of inputting subjective pain level. In our case the criterium was passed because in each of our trials the user inputted pain level was correctly recorded in the app.
- In this case we ran an attribute test which the mean and standard deviation isn't relevant. The criterium for acceptance is the correct recording of every trial.
- The measurement technique was adding code which prints the input into the terminal. In this case the measurement technique couldn't have had an effect.

Test no. 2:

- The acceptance criterium is based on 10 trials of dosage recommendations. In our case the criterium was passed because in each of our trials the dosage recommendation level was correctly recorded in the app.
- In this case we ran an attribute test which the mean and standard deviation isn't relevant. The criterium for acceptance is the correct recording of every trial.
- The measurement technique was adding code which prints the input into the terminal. In this case the measurement technique couldn't have had an effect.

Test no. 3:

- The acceptance criteria included two main parts; connecting to 5 valid networks/hotspots and then reconnecting every 5 seconds and no failures upon reconnection. The network would always connect in less than 3 tries for all 5 network/hotspot samples. However, this is less than our no failures on 2nd attempt. We view the test as passed but the above details should be noted.
- The measurement technique used was consistent for all networks(i.e used the same microcontroller and code to connect).
- Due to networks that did not connect within 2 tries, the microcontroller code was updated to attempt its first reconnection in just a second but then logarithmically

increase the time between reconnection attempts to avoid draining the battery or exhausting the stack.

Test no. 4:

- The acceptance criteria of ensuring correct actions are triggered by the apps “Take Dose” button was met in each of the test samples denoted as pass fails for each sample.
- Samples 1-20 propagated through the appropriate actions leading to the load cell sample being uploaded to IO.
- The main difference in our test setup and actual use conditions is the app being simulated through a desktop and the bottle circuitry being layed out on a breadboard instead of in the bottle. This test was meant to ensure proper connection between the app, pill bottle, and the IO server. Thus, I don’t see these differences severely impacting the efficacy of the results.
- Code was also validated via ChatGPT for errors. None were found.

Test no. 5:

- The acceptance criteria of ensuring data upload to IO within 60 $\pm$ 10 seconds was easily met through our test results. Our mean was 1.49 seconds till visual confirmation the data was uploaded to IO. This is well below our lower acceptance of 50 seconds. Our confidence interval at an alpha of 0.0025 was (-1.236 sec, 4.214 sec) allowing us to confidently pass this test with flying colors.
- Mean: 1.49 seconds, STD: 1.88 seconds, test data in appendix.
- The measurement technique could have been improved to not depend on visual confirmation of the data showing up in IO. This is prevalent in an outlier data point of around 6 seconds till upload which dramatically increased our standard deviation and thus margin of error. In retrospect a better approach would have been to keep the MQTT client listening for new data from the load cell feed and then stopping the timing loop internally. Although this method would have counted for two-way communication, not included in the acceptance criteria.
- Even though we had an outlier data point, the outlier was still a factor of nearly 10 less than our acceptance criteria leading to no changes in design based on this test.

Test no. 6:

- The acceptance criteria of ensuring the time stamp listed in adafruitIO translates to the app storage. In this case we have 4 trials and need all 4 to pass this test. In this case all 4 trials passed so we accept the criteria.
- No statistics are appropriate for this test.
- The measurement was taken from logging the imported value directly in the console. There is no way it could have affected the result.

Test no. 7:

- Our acceptance criterium is taken through a T-test with an a=.00005. This test accepted the criterium.
- Across the 5 trials the mean was 2 minutes with a standard deviation of 0.
- This was measured by adding a console log statement that outputs the time as soon as the Take Dose button appears, by comparing the last dose with this time we can get an idea. This method doesn’t affect the result and was the most accurate way possible.

Test no. 8:

- This test failed to meet the acceptance criteria for the lid diameter of 0.0313 ± 0.016 inches, as the measured range [0.021, 0.055] exceeded the upper tolerance limit of 0.0473 inches. The failure likely resulted from variability in the 3D printing manufacturing process. This indicates the inherent limitations of the 3D printing method in achieving precise tolerances. To address this, we will explore alternative manufacturing techniques, such as CNC machining to meet tighter tolerances. We used a doubled sides T-test with a= 0.0025 and N=15.

Test no. 9:

-

Test no. 10:

- The acceptance criteria were at least 95% activation of solenoid driving circuit when given signal from microcontroller. Since the solenoid failed to be activated in one of the 30 trials, we get a mean activation of 96.67% which allows us to pass the test based on the criteria.
- Mean: 96.67%, STD: 3.28%, test data in appendix.
- Under actual use conditions many other things can go wrong besides the solenoid functioning. Our test conditions used the same code that would be used in actual use conditions thus I don't see any effect on our data and thus results.
- The failed sample revealed an issue with our solenoid driving circuit. Specifically, our transistor was specified to just barely provide enough current and voltage to drive the solenoid. A new NPN transistor was ordered and integrated into the circuit to increase consistency.

Test no. 11:

- Our acceptance criteria for the Load Cell Differentiation Test was that 1g, 2g, and 5g weights were differentiable to a degree of variation determined by an ANOVA test using an alpha of 0.05. The ANOVA resulted in a p value of < 0.0001 suggesting our mean count values within each weight standard are significantly different. Thus, we pass this test.
- The main statistical criteria to report that backs our pass of this test is the p value < 0.0001. Other ANOVA statistics (i.e means in each groups, variations between groups, etc.) is listed in the appendix under *Test 11 Data*.
- Not re-zeroing the ADC via software as well as slight adjustment of the load cell between samples could be a cause of error in our samples and thus was payed great attention too when testing.
- No samples were failed, only one sample came out to be an outrageous count value greater than 8,000,000 during the 1g weight testing. We believed this was caused by a connection error between the Wheatstone resistances leading to a large count value. This ended up being the case after further testing which resulted in us ordering a new load cell that was not damaged. The new load cell yielded no bizarre count outputs at any time during subsequent testing.

Test no. 12:

-

Test no. 13:

- The acceptance criteria involved no failed take dose activations over 20 emergency button presses. Every time the button was pressed after being properly addressed in the microcontroller code, the Take Dose state was triggered in States.py. With no failures we pass this test.

- 100% success rate. Test data in appendix.
- The testing of all 20 button presses occurred when the circuitry was layed out on breadboard and the button was not yet nested in the bottle. Thus, if wiring becomes faulty under actual use conditions this could cause an effect on the robustness of the button activation.
- Due to the potential shortcomings of our testing conditions(even though there were no failures) this test will be revisited with a smaller n once the circuitry is assembled in the bottle.
    - Update: Once assembled button activation was successful 100% of samples(5)

Test no. 14:

- Drop Test was not tested

Test no. 15:

- Acceptance criteria required the microcontroller LED to properly cycle between red, green, and yellow. The LED successfully cycled through states in 5 unit tests representing a pass, this also includes tests that involved the emergency button activation.
- 5 unit tests passed. “Test Data” is unit test in Test.py.
- Because all the testing was done in software, the only errors that could occur was in software. This makes sense considering adjusting the LED is purely done in software. However, there could be potential iterations or inputs missed by the unit tests that could affect the results.
- No failed samples.

Test no. 16:

Test no. 17:

- The pill container met the required dimensions of radius r=0.57−0.58 and height h ≥1.75 inches. The measured radius range of [0.562, 0.579] inches fell within the specified tolerance. The height exceeded the minimum requirement in all samples(not shown). No changes to the design or process are necessary at this time. We used a single sided T-test with a= 0.00005 with an N=15

Test no. 18:

- The hardware housing met the dimensional requirements of at least 2.73 inches by 1.4 inches by 1.5 inches. All measurements exceeded these minimum criteria and all tolerance criteria was met([2.76 2.92], [1.51 1.61] and [1.42 1.49]. No deviations or outliers were measured. No further adjustments to the design or process are required. We used a single sided T-test with a= 0.00005 with an N=15

7.2 Describe how the results will impact the risk assessment and provide an updated FMEA that includes your test results

Many of our tests resulted in positive verification of our design choices enforcing our strategy moving forward. The main factors in the risk assessment that were affected is the occurrence factor of the final RPN value. For example, we had an occurrence value of 6 for the case where a patient is unable to connect the bottle to a valid wifi network. After testing the microcontrollers wifi radio and the associated code on multiple hotspots and wifi networks we bumped the occurrence value down to 2. We were able to verify that the wifi antenna when at 2.4 GHz can effectively (and quickly) connect to a network or hotspot with a very low rate of complete failure leading to the patient being unable to use the bottle to access their opiate prescription.

Successful tests such as the one described were reflected mainly in a decrease in occurrence value in the FMEA table, as stated. However, other aspects of the FMEA table such as severity, causes, and controls were unchanged. Severity should never change because we found the occurrence of a certain error to be lower or higher than expected during verification testing. Our causes and controls were accurate and thoughtful when written prior to prototyping and thus remain unchanged. Updated FME table is shown below.

Unrelated to verification testing, certain rows in the FMEA table were omitted that concerned components/features of our device that were no longer relevant to the prototype.

Appendix

Test no.1 Data

Trial	Inputted pain level	Pain level output	Pass/ Fail

1	'happy'	'happy'	Pass
2	'happy'	'happy'	Pass
3	'sad'	'sad'	Pass
4	'happy'	'happy'	Pass
5	'sad'	'sad'	Pass
6	'neutral'	'neutral'	Pass
7	'sad'	'sad'	Pass
8	'neutral'	'neutral'	Pass
9	'happy'	'happy'	Pass
10	'neutral'	'neutral'	Pass

Test no.2 Data

Trial	Pain Input	State	Recommendation	Pass/Fail
1	'sad'	'yellow'	2	Pass
2	'neutral'	'red'	0	Pass
3	'neutral'	'red'	0	Pass
4	'happy'	'green'	1	Pass
5	'sad'	'yellow'	2	Pass
6	'happy'	'green'	1	Pass
7	'happy'	'green'	1	Pass
8	'sad'	'red'	0	Pass
9	'neutral'	'yellow'	1	Pass
10	'neutral'	'red'	0	Pass

Test no.3 Data:

Network:	Immediate Connection?	Number of reattempts:	Connect at all?
Biodesign room	Yes	1	Yes
Wyatt hotspot	yes	1	Yes
Wyatt home	No	3	Yes
Brian hotspot	No	2	Yes
Daniel home	yes	1	yes

Test no.4 Data:

Sample no.	**TakeDose Payload Received?**	**Solenoid Unlocked?**	**LoadCell IO Feed Updated?**
1	Yes	Yes	Yes
2	Yes	Yes	Yes
3	Yes	Yes	Yes
4	Yes	Yes	Yes
5	Yes	Yes	Yes
6	Yes	Yes	Yes
7	Yes	Yes	Yes
8	Yes	Yes	Yes
9	Yes	Yes	Yes

10	Yes	Yes	Yes
11	Yes	Yes	Yes
12	Yes	Yes	Yes
13	Yes	Yes	Yes
14	Yes	Yes	Yes
15	Yes	Yes	Yes
16	Yes	Yes	Yes
17	Yes	Yes	Yes
18	Yes	Yes	Yes
19	Yes	Yes	Yes
20	Yes	Yes	Yes

Test no.5 Data:

Sample no.	**Time till uploaded(seconds):**	**Data received by IO and accurate?**
1	1.23	Yes
2	0.33	Yes
3	0.59	Yes
4	0.54	Yes
5	1.22	Yes
6	1.03	Yes
7	6.48	Yes
8	0.58	Yes
9	1.18	Yes
10	0.51	Yes

Test no.6 Data:

Trial	Adafruit Value	App Value	Pass/Fail
1	"2024-12-04T23:13:14Z"	"2024-12-04T23:13:14Z"	Pass
2	"2024-12-04T23:14:48Z"	"2024-12-04T23:14:48Z"	Pass
3	"2024-12-04T23:15:12Z"	"2024-12-04T23:15:12Z"	Pass
4	"2024-12-04T23:15:56Z"	"2024-12-04T23:15:56Z"	Pass

Test no. 7 Data

Trial	Time until Take Dose appears	Pass/Fail
1	2:00	Pass
2	2:00	Pass
3	2:00	Pass
4	2:00	Pass
5	2:00	Pass

Test no.8 Data:

inner measurement	outer measurement	difference (in)

(mm)	(mm)	
33.57	34.67	0.0433
33.7	34.84	0.0449
33.48	34.26	0.0307
33.36	34.43	0.0421
33.43	34.6	0.0461
33.5	34.67	0.0461
33.47	34.23	0.0299
33.58	34.55	0.0382
33.45	34.23	0.0307
33.45	34.16	0.0260
33.39	34.57	0.0441
33.42	34.55	0.0433
33.33	34.32	0.0366
33.28	34.21	0.0311
33.47	34.36	0.0406

Mean difference= 0.0379
STD= 0.0067
N=15

Test no.10 Data:

	Button Activation:	**IO TakeDose Payload Activation:**
Successes:	14	15
Failures:	1	0

Test no. 11 Data:

Weight(g)	Sample 1 (counts):	Sample 2 (counts):	Sample 3 (counts):	Sample 4 (counts):	Sample 5 (counts):
1	20108	19985	20011	20099	19892
2	40201	39943	40114	39897	40003
5	119745	119886	121004	119846	120120

ANOVA output:
F-statistic value = 146920.4475
P-value = 0.0000
By conventional criteria, this value is considered to be **extremely statistically significant**.

Data Summary:

Groups	**N**	**Mean**	**Std. Dev.**	**Std. Error**
Group 1	5	20019.00	88.9803	39.7932
Group 2	5	40031.60	124.7710	55.7993
Group 3	5	120120.20	512.8140	229.3374

Anova Summary:

Source	**Degrees of Freedom**	**Sum of Squares**	**Mean Square**	**F-Stat**

Between Groups	2	28058230416.93	14029115208.47	146920.4475	
Within Groups	12	1145854.00	95487.83		
Total	14	28059376270.93			

Test no. 13 Data:

Sample no.	**Solenoid Unlocked?**	**LoadCell IO Feed Updated?**
1	Yes	Yes
2	Yes	Yes
3	Yes	Yes
4	Yes	Yes
5	Yes	Yes
6	Yes	Yes
7	Yes	Yes
8	Yes	Yes
9	Yes	Yes
10	Yes	Yes
11	Yes	Yes
12	Yes	Yes
13	Yes	Yes
14	Yes	Yes
15	Yes	Yes
16	Yes	Yes
17	Yes	Yes
18	Yes	Yes
19	Yes	Yes
20	Yes	Yes

Test no. 17 Data

Sample no.	**Radius(in)**
1	0.571
2	0.573
3	0.573
4	0.565
5	0.567
6	0.571
7	0.571
8	0.574
9	0.569
10	0.573
11	0.566
12	0.573
13	0.569

14	0.574
15	.569

Mean= 0.532
STD= 0.147
N=15

Test no. 18 Data

Sample no.	Height(in)	Length(in)	Width(in)
1	2.862	1.558	1.471
2	2.873	1.575	1.464
3	2.860	1.567	1.475
4	2.845	1.549	1.460
5	2.820	1.563	1.470
6	2.812	1.543	1.465
7	2.863	1.537	1.466
8	2.798	1.564	1.466
9	2.813	1.547	1.474
10	2.804	1.561	1.471
11	2.865	1.595	1.440
12	2.848	1.559	1.464
13	2.808	1.588	1.454
14	2.871	1.568	1.476
15	2.822	1.548	1.482
mean	2.839	1.562	1.465
std	0.028	0.016	0.009

Revision 1 Wyatt Young (WY) 12/05/2024
Updates Made to Verification Testing Document:

- Updated testing statistics for non-pass/fail tests
- Added references to original specification document
- Added AI validation methods instead of unit testing/other code testing methods(per Tomasz advice)
- Adjusted light test to not be in dark room with multiple people to actual light meter measurements using phone app
- Added revision numbers and updated revision log to reflect current state
- Added class and year to summary and such
- Updated all tolerance tests to be t-tests instead of binomial
- Updated test descriptions to be more thorough and to cover tests that were missing
- Added thorough results sections with details on what went wrong and what was updated
- Added test data in appendix

Bioengineering 3801/4801 Biomedical Engineering Design
Research Summary

Pharmacokinetic Control of Opioid Dosing with Patient Feedback
Revision: 03 (Update)

1. Introduction

The purpose of the research summary is to organize research papers accessed and used as resources for the development of a device for Pharmacokinetic Control of Opioid Dosing in an accessible, organized form.

Marketing Requirements Research		
Date Accessed	Citations	Notes and Implications
1/28/24	Michael Von Korff, Joseph O. Merrill, Carolyn M. Rutter, Mark Sullivan, Cynthia I. Campbell, Constance Weisner, "Time-scheduled vs. pain-contingent opioid dosing in chronic opioid therapy",PAIN, Volume 152, Issue 6, 2011,Pages 1256-1262,	• Patients receiving COT who exclusively used time-scheduled opioid dosing received substantially higher average daily doses than patients who used only pain-contingent opioid dosing. • Patients using time-scheduled dosing reported being more preoccupied with opioid use and less able to control their opioid use, and they were more worried about opioid dependence. • Patients using time-scheduled dosing received higher opioid doses than did patients with pain-contingent dosing. These patients took opioids on more days per week and more times per day, and were more likely to use higher-potency schedule II opioids.
1/28/24	Christopher JG. Green, Somnath Bagchi, "Techniques of opioid administration", Anaesthesia & Intensive Care Medicine, Volume 20, Issue 8, 2019,Pages 430-435,	• Administration techniques for Class 2 Opiates. • Basic information on clinical protocol for administering opioid prescriptions.
2/2/24	"FACTSHEET: Utah's Oversight of Opioid Prescribing and Monitoring of Opioid Use" USGOV https://oig.hhs.gov/oas/reports/region7/71805115.asp [Factsheet Reviewed: 2/8/19 Audit(A-07-18-05115), accessed: 2/2/24]	• Utah's policies and procedures regarding schedule 2 drugs (most chronic opiates) • Outlines filling/refilling procedures

2/2/24	Wenzel, J. T., Schwenk, E. S., Baratta, J. L., & Viscusi, E. R. (2016). Managing Opioid-Tolerant Patients in the Perioperative Surgical Home. *Anesthesiology clinics*, *34*(2), 287–301.	• Basic information on tolerance for opiate prescribed patients. • Includes time periods, specific drugs, etc. • Factors for weighing factors in software algorithm.
2/2/24	Eduardo Bruera et al.,Cancer Pain Management: Safe and Effective Use of Opioids. *Am Soc Clin Oncol Educ Book* **35**, e593-e599(2015).	• Opioid Metabolism • Active metabolite table • Extended release opioids • Pharmacokinetics table of common drugs • Cancer treatment context
2/18/24	"CPC Definition - A61J Containers specially adapted for medical of pharmaceutical purposes;…" *United States Patent and Trademark Office*. [Online]. Available: https://www.uspto.gov/web/patents/classification/cpc/html/cpc-A61J.html [Accessed: 2/18/24].	• A61J 3/00 - "Containers specially adapted for medical or pharmaceutical purposes" • A61J 1/03 - "for pills of tablets with special dispensing means therefor B65D 83/04" • A61J 7/00 - "Devices for administering medicines orally" • A61J 7/02 - "Pill counting devices [2006.1]" • A61J 7/04 - "Arrangements for time indication or reminder for taking medicine, e.g. programmed dispensers" • A61J 2200/74 - "Device provided with specific sensor or indicating means for weight"
2/18/24	"CPC Definition – G16H Healthcare Informatics, i.e. Information and Communication Technology,…" *United States Patent and Trademark Office*. [Online]. Available: https://www.uspto.gov/web/patents/classification/cpc/html/cpc-G16H.html#G16H [Accessed: 2/18/24].	• G16H 20/13 - "ICT specially adapted for therapies or health-improving plans, e.g. for handling prescriptions, for steering therapy or for monitoring patient compliance delivered from dispensers" • G16H 70/40 – "ICT specially adapted for the handling or processing of medical references relating to drugs,…"

2/18/24	R. Karpman, P. R. Aronson, N. Bal, and D. Lipson, "Apparatus and method for dispensing medication," U.S. Patent 11,896,556 B1, May 3, 2021.	US11896556B1 Apparatus and Method for Dispensing Medication • For multiple medications • Rotating containers specified medications • Probes with suction method for grabbing pills • Probe programmed via computer software
2/18/24	"US10160588B2 - Dispensing cartridge - Google Patents," *Google.com*, Jun. 06, 2016. https://patents.google.com/patent/US10160588B2/en?oq=15%2f174187 (accessed Feb. 18, 2024).	US10160588B2 Dispensing Cartridge • Actuator within cartridge to align positions for dispensing • Filed by Hero Health Inc. • For use in larger pill organizers/dispensers
2/18/24	"US11833113B2 - Cap assembly for a medication container - Google Patents," *Google.com*, Apr. 25, 2022. https://patents.google.com/patent/US11833113 (accessed Feb. 18, 2024).	US11833113B2 Cap assembly for a Medication Container • Cap includes gate that is selectively opened. • Gate locking mechanism; only unlocks to allow the gate to open in response to application of downward force on a portion of the gate. • Cap includes electronic components; stores memory of number of pills that pass through cap. • Uses sensor to detect pill passage.
2/18/24	*"US11854680B1 - Narcotics and opioids secure storage and dispensing apparatus and method of use - Google Patents," Google.com, Apr. 21, 2022. https://patents.google.com/patent/US11854680B1/en?oq=17%2f726455 (accessed Feb. 02, 2024).*	US11854680B1 Narcotics and Opioids Secure Storage and Dispensing Apparatus and Method of Use • Multi-tray (2 motorized trays) • Tamper-resistant housing design specs • Sensors, reclamation safe • Wireless communication module to notify remote operators of unauthorized access. • Dispensing schedule or patient input
Specifications Research		
Date	Citations	Notes and Implications
3/21/24	Habibi M, Kim PY. Hydrocodone and Acetaminophen. [Updated 2022 Dec 19]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2024 Jan-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK538530/	• Hydrocodone is one of the most common pain medications prescribed by clinicians and one of the most abused by patients • Hydrocodone reaches maximum serum

		concentrations within 1 hour with an elimination half-life of 4 to 6 hours. • For initial oral tablet therapy, the usual adult dose of hydrocodone and acetaminophen (5 mg / 325 mg) is one or two tablets every four to six hours as needed for pain. The total daily dosage should not be more than eight tablets. • The addition of acetaminophen can enhance the analgesic effect of hydrocodone.
3/28/24	Ng, Phan, Nguyen, DES 40A Research project W2023: *The Life Cycle of Orange Pill Bottles*. 16, March 2023. https://www.designlife-cycle.com/orange-pill-bottles	• Manufacturing process of plastic • Wastes/emissions • Human factors out the ass on this resource! • Recycling practices/shortcomings
3/28/24	"Amber Pill Bottles and Empty Pharmacy Vials, Push Down & Turn Lid," *www.thorntonplastics.com*. https://www.thorntonplastics.com/pill-bottles-pharmacy-vials/amber.html	• 6-60 dram • UPS 671 Moisture Permeation testing • 16 CFR 1700 test • UPS 661 Plastic Packaging Systems
4/3/24	Singla A, Sloan P. Pharmacokinetic evaluation of hydrocodone/acetaminophen for pain management. J Opioid Manag. 2013 Jan-Feb;9(1):71-80. doi: 10.5055/jom.2013.0149. PMID: 23709306. https://pubmed.ncbi.nlm.nih.gov/23709306/	General release times, half life, etc.
4/13/24	Cofano S, Yellon R. Hydrocodone. [Updated 2023 Nov 17]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2024 Jan-.	Most effective hydrocodone formulation based on pain, duration, age, etc.
4/13/24	"Hydrocodone And Acetaminophen (Oral Route) Proper Use - Mayo Clinic," *www.mayoclinic.org*. https://www.mayoclinic.org/drugs-supplements/hydrocodone-and-acetaminophen-oral-route/proper-use/drg-20074089	Dosage info for hydrocodone/acetaminophen, dosing limits, etc.

4/13/24	"Vicodin - FDA prescribing information, side effects and uses," *Drugs.com*. https://www.drugs.com/pro/vicodin.html	Vicodin info: 14 mm oval
4/13/24	"What is Power Analysis and Power Measurements," *Data Acquisition | Test and Measurement Solutions*. https://dewesoft.com/blog/what-is-power-analysis	Power analysis info for wiring components.
Prototype 1 and Verification Testing Research		
Date	Citations	Notes and Implications
9/24/24	"ESP32-S3-DevKitC-1 v1.1 - ESP32-S3 - — ESP-IDF Programming Guide v5.2.1 documentation," *docs.espressif.com*. https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html	Contains pin layout for ESP32-S3 board. Also designates pins for I2C communication with ADC for load cell.
9/24/24	Adafruit Industries, "Small Lock-style Solenoid - 12VDC @ 350mAh with 2-pin JST," *Adafruit.com*, 2024. https://www.adafruit.com/product/5065 (accessed Nov. 07, 2024).	Contains info on driving solenoid as well as appropriate transistor and flyback diode configs.
9/26/24	"CircuitPython - Libraries," *circuitpython.org*. https://circuitpython.org/libraries	Contains libraries for Circuit Python. Also includes UF2 bootloaders used for our version (v9.x) on our board.
9/26/24	OpenAI, "ChatGPT," *ChatGPT*, 2024. https://chatgpt.com/	Used for code verification as well as questions regarding MQTT client for IO server.
10/9/24	"Hydrocodone And Acetaminophen (Oral Route) Proper Use - Mayo Clinic," *www.mayoclinic.org*. https://www.mayoclinic.org/drugs-supplements/hydrocodone-and-acetaminophen-oral-route/proper-use/drg-20074089	Dosage info for hydrocodone/acetaminophen, dosing limits, etc. (Heavily referred back to)
11/26/24	Adafruit Industries**.** (n.d.). *Adafruit CircuitPython NAU7802 Library*. 2024, https://github.com/adafruit/CircuitPython_NAU7802	Contains driver code for ADC for sampling load cell. Used for zero and internal calibration.

12/2/24	Appinio. (n.d.). *ANOVA calculator: Perform a one-way analysis of variance online.* Retrieved November 19, 2024, from https://www.appinio.com/en/tools/anova-calculator	Used for calculating ANOVA statistics for the Load Cell Sampling Test.