An Open-Source, Modular Syringe Pump for Low Income Countries

To
Deborah Walter, Rose-Hulman Institute of Technology
Ashley Bernal, Rose-Hulman Institute of Technology
ECE462 - Team 11

From
Team 11
Jake Armstrong
Tyra Correia
Gabe Neise
Wei Huang

Created: 2/18/2024 8:57 PM
Last Edited: Thursday, February 22, 2024

Executive Summary

Syringe pumps are devices used for administering lifesaving medications and anesthesia to patients. While they are essential medical equipment, modern designs fail to satisfy the financial and physical limitations of use in low-income countries and resource-poor environments. Therefore, our team developed an open-source, low-cost, modular syringe pump design to remedy these barriers by creating our design from 3D-printed material and standard electronic components from maker communities that are inexpensive and easy to source and replace.

Our team collaborated with a mechanical engineering team to develop this design. While they created a physical apparatus to hold the syringe, our team programmed the stepper motor to depress it. Our goals consisted of developing an easy-to-navigate user interface for the user to operate and set specifications for the stepper motor and a control system to regulate the syringe's dosage rate, provide remote data logging, and implement fault alarms.

While our team succeeded in developing a user interface and programming the stepper motor, our control system could not, over time, deliver a consistent dosage. Data logging and fault alarms were also not completed, but safety features such as occlusion detection were installed in the system. The design should only be tested in laboratory cultures for research. We hope to see future teams solve the dosage tracking issues and add networking. Because the design is open source, all the project's CAD and software files are freely available on a public online repository, so educators, researchers, and doctors can download, print, and assemble their own syringe pumps and contribute to the design themselves. While we could not implement all the features we wanted, we successfully provided the software and electronic hardware to supply power and control to the stepper motor in our low-cost infusion pump design.

Table of Contents

1. INTRODUCTION AND PROJECT OVERVIEW

1.1. Background Information and Project Context

Developing a low-cost syringe pump improves medical care and decreases morbidity and mortality in developing lower-income countries. [1] Dr. Nate Goergen and Dr. Derek Sakata from the University of Nebraska Medical School and the University of Utah School of Medicine have requested the development of a low-cost, open-source modular syringe pump for low-income countries. A syringe pump is a device that uses a mechanized piston system to deliver infusions from the syringe reservoir at a controlled rate. As mainline equipment for supplying nutrients, anesthesia, and lifesaving medications to patients, they are a crucial asset in the medical industry.

Low-income countries struggle to access modern, working medical equipment due to their high cost, proprietary designs, and sensitive electronics. The lack of modularity of modern infusion pump designs makes it difficult to repair and service them in environments with fewer available resources. Additionally, the retail price of current designs is expensive, as a quick Google search will reveal that the cost of syringe pumps can range from $2000 to $3000. Furthermore, devices produced in countries with robust medical facilities cannot operate efficiently in countries with varying environmental conditions due to their inconsistent power grids and the substantial cost of simulating an environment suited for sensitive electronics. [2]

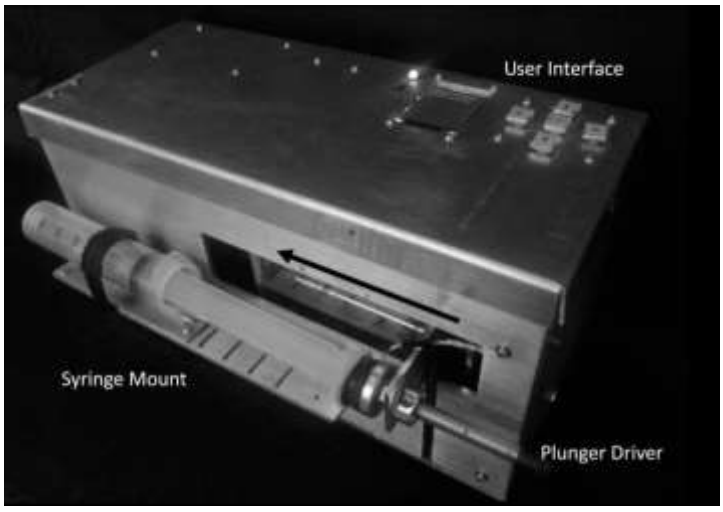

Figure 1. AutoSyp Syringe Pump

An example of a similar syringe pump designed to remedy the accessibility problem is the AutoSyp, a low-cost, low-power syringe pump created for use in hospitals in underdeveloped nations, as shown above in Figure 2. The AutoSyp uses a constant-force spring inside the device to deliver infusions at a steady rate, which allows it to run on rechargeable battery power for over 60 hours, accommodating environments with infrequent power grids. While AutoSyp is an excellent example of what this project is trying to accomplish, the cost of producing an AutoSyp

syringe pump at $500 per unit, we believe the price needs to be lowered for frequent use in resource-poor countries. [3]

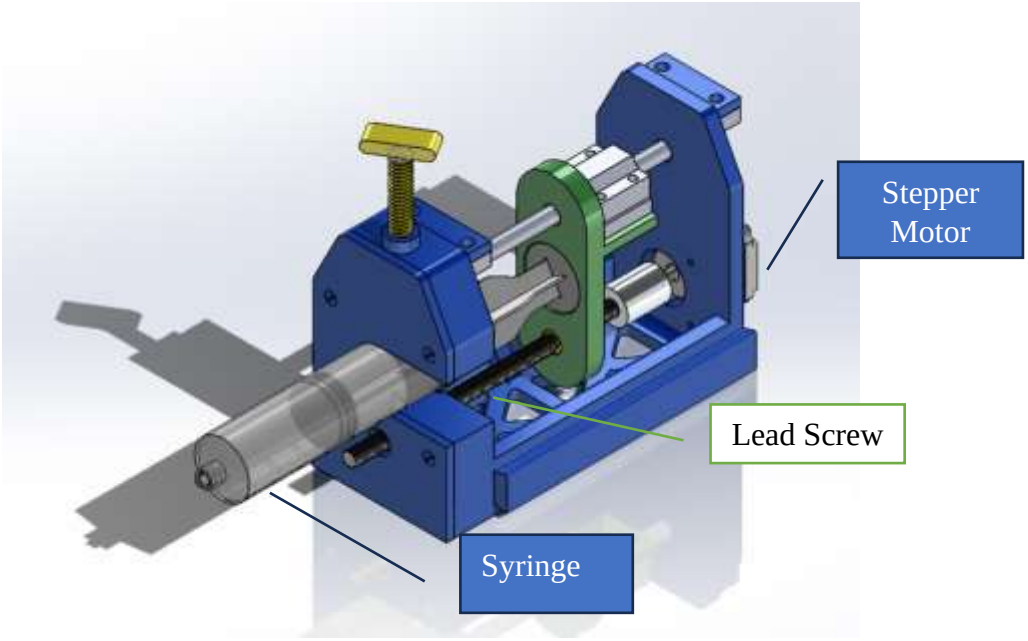

Figure 2. ME Team Syringe Apparatus CAD Design

To complete our goal of designing an inexpensive syringe pump tailored to the shortcomings of developing countries, our ECE team worked alongside a team of mechanical engineers to create our prototype. While the ECE team designed the motor controller and user interface, the mechanical engineering team developed the syringe channel system, a 3D-printed apparatus intended to hold the syringe shown in Figure 2. The device uses a NEMA stepper motor to turn a lead screw that shifts a module along a steel rod. As the module slides forward, it pushes the syringe, dispensing the liquid from its reservoir through the load cell. When the syringe reaches the end, it will trip a limit switch at the system's front end and terminate motor power.

This report focuses on implementing the software and control of the stepper motor. As shown in Figure 3, we implemented a user interface and stepper motor driver programmed by an ESP32 chip and integrated with an external power supply and a custom-designed PCB. When the user powers the system, they can utilize the touchscreen user interface to set the syringe's bolus and infusion rates and volumes. Bolus rates are usually a faster dosage rate to get a patient’s blood concentration to an effective level. They can also use it to start and stop and start the stepper motor to push the syringe. The system is programmed with an emergency stop switch that can override and shut down the system and occlusion detection from the stepper motor driver that will cease motor functions if the motor is stalled.

Our teams collaborated with Rose-Hulman Mechanical Engineering 2023-2024 Capstone Team 31 to reduce production costs, make our design low-cost, and quickly assemble in the field. The pump casing and structural components are 3D printed, which, in addition to limiting costs,

allows for parts to be printed in-country. Suppose a supporting module in the pump breaks or becomes non-functional in a country with limited resources. In that case, a replacement component can be printed and reassembled using a standard toolkit, saving time and shipping costs. The electronics are powered by a custom-designed printed circuit board that can function in various electrical environments. The NEMA stepper motor is a standard, off-shelf motor often used in 3D printers that only costs ~$15 and is easy to obtain. The microcontroller that programs the user interface and stepper drivers is an ESP32 chip that is easily sourced and only costs $2.80. We estimate the total cost of producing the electronics for our infusion pump to be about $97, at a production quantity of 2. An estimated cost to assemble the entire system, as pictured in Figure 3, appears to be about $150 to $200.

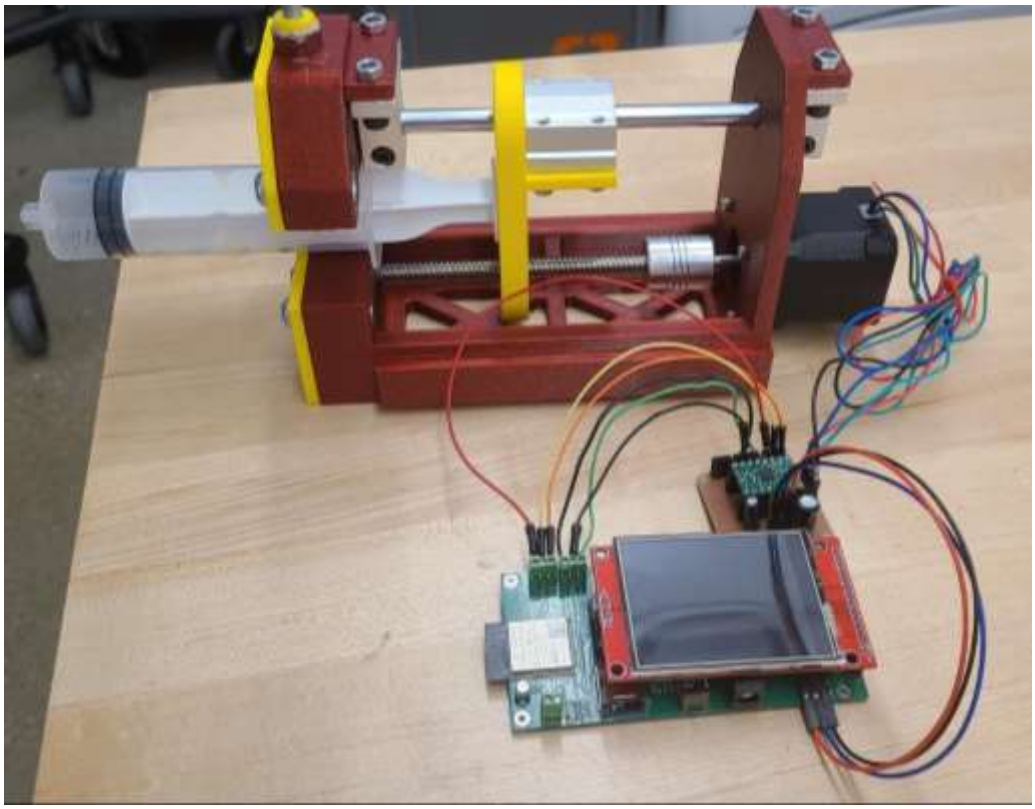

Figure 3. Syringe Pump Demonstration

1.2. Stakeholders, Features, and Metrics

Our system comprises six key stakeholders: the patient, distributor, hospitals, and medical practitioners. Table 1 provides insight into stakeholders' roles in the Syringe Pump Project. The features table explores our system's key characteristics, including Ease of User, Safety, Affordability, and Durability. Shortlisting features implemented by our design make it possible to gauge the degree to which the final product satisfies the stakeholders associated with it. Table 3, therefore, cross references the features with the corresponding stakeholders affected by its successful integration into the project.

Table 1. Stakeholders and descriptions

Stakeholder Name	Description
Patient (P)	The patient indicates the primary user of the OSMI syringe pump and will be the individual to whom the drug is administered.
Nurse/ Doctor (ND)	The end users are held accountable for administering anesthesia safely during medical procedures. The end user must possess medical expertise and insight into some of their anesthesia pump design requirements that align best with their clinical practices and the end user's specific needs.
Electronics Recyclers (ER)	Electronics recyclers are often concerned with minimizing the volume of hazardous waste. If the pump design is not environmentally friendly and incorporates non-sustainable or dangerous materials, the electronic recyclers would be burdened to clean and dispose of that waste.
Electronics Manufacturers (EM)	Electronics manufacturers can be integral to the selection of electronic components to be used in the anesthesia syringe pump.
Open-Source Distributors (OSD)	Open-source distributors decide which medical devices to build and supply Simulated Health Administrations in low-income countries. Distributors are often in direct contact with healthcare systems. They can provide valuable feedback to health organizations via customer support regarding the demand for specific features or improvements in anesthesia syringe pumps.
Hospitals (H)	Hospitals in low-income countries often operate in resource-constrained environments like conflict zones or low-income regions, requiring equipment that can cater to these challenging operational requirements. Such organizations cause a need for user-friendly interfaces and simplified training requirements.
Client (C)	The client’s role in the project sets up the expectations for the product’s essential features and requirements. Therefore, the client's needs will shape the final product's design and delivery. The client will also primarily source the funding for the development, and therefore, client budgetary constraints will limit the team's resources.

Table 2. Features and their definitions

Feature Name	Feature Definition
Ease of Use	The device should also be easy to employ by a registered medical practitioner with clear communication of essential elements via the user interface. End users should be able to customize the syringe pump to meet each simulated patient's needs, including factors such as standard dosage infusion rate and bolus infusion rate.
Safety	The project will rely on stable and functional anesthesia syringe pumps to deliver medications accurately and without errors. Therefore, safety features such as clear alerts, user-friendly controls, and safeguards are crucial in the case of accidental misjudgment by the medical practitioner or device manufacturer to prevent harm to the simulated patient.
Testability	Testable features in a syringe pump allow for greater quality assurance by enabling engineers to identify and rectify malfunctions or flaws in the medical device. The ability to conduct tests supports verifying and validating the syringe pump’s design requirements.
Durability	Durability contributes to the extended device lifespan, reducing the need for frequent replacement or repairs. This lowers the total cost of ownership and minimizes disruptions in clinical workflows. A durable design can withstand exposure to various chemical and physical stresses in the healthcare industry.
Affordability	The design should consider compatibility with low-cost, readily available materials and components, reducing production costs.
Hardware Maintainability	Maintainability contributes to sustainability by extending the lifespan of the syringe pump. A maintainable design minimizes downtime by allowing quick and efficient maintenance and repairs. A maintainable design allows for replacing individual components or modules rather than the entire device.
Portability	An emphasis on portability often leads to a compact and lightweight design, making it easier for the end user to carry and set up the device quickly.

The corresponding table below (Table 3) lists the set of features and a detailed description of their metrics. In addition, a list of stakeholder IDS has been used to track which features meet the needs of the stakeholders in Table 1, ensuring that all features selected to be implemented are relevant to the project's goals.

Table 3. Features & Corresponding Metrics

Feature	ID	Attribute/Metric	P	ND	ER	OSD	C	H
Ease of Use	EOU.1	A well-trained end-user should be able to be fully trained in how to use the device in under one hour.	x			x	x	x
Accessibility	EOU. 2	The system should be easy to learn and use within a limited time.						x
Customization	EOU.3	Control over how much material the syringe delivers, medication dosage, infusion rates, etc.	x					x
Safety	SF.1	The system should function in a manner that will not cause or pose a risk of harm to its users or beneficiaries.	x	x	x		x	x
Durability	DB.1	The device should be robust and not break easily when subject to physical stress.			x			x
Affordability	AF.1	The cost per singular unit of the device should be affordable to ensure industrial replicability. The total cost should aim to be under $100 per channel.				x	x	x
Hardware Maintainability	HD.1	Essential components of the system that fail or need to be interchanged should be able to be repaired or replaced quickly.			x	x	x	x
Portability	PT. 1	The device should be able to be transported easily and remain functional.			x			x

1.3. Project Conclusion

This project intends to develop an open-source modular, low-cost infusion pump that can be implemented in resource-poor environments. To satisfy this goal, our team proposed creating the following:

1. An easy-to-use user interface that will control the functionality of the syringe pump and display the current instrument status.
2. Control subsystem with fault alarms, medication data delivery logs, and reliability testing.
3. Device monitoring software that will monitor and record data such as power data, syringe pump data, error codes, and more.
4. Technician manuals include detailed documentation, data explaining how all schematics, code programs, and control systems function, and medical professional manuals with clear operating instructions.

While the team has developed a promising prototype for a syringe pump, several issues with the final dosage system need to be investigated before clinical trials. We believe we have met our goal of developing an easy-to-use interface and have a sound foundation for the control subsystem. While we could not complete the device monitoring software, we have completed our technician manuals, allowing future work to start efficiently.

2. ACCOMPLISHMENTS

2.1. – Description of the Solution

The primary goal for the system is to remain cost-effective and repairable for a medical device without compromising safety. Figure 4 shows a high-level interaction model of our final system design. The syringe pump's design uses an ESP32-powered LCD-touch screen user interface and a stepper motor to provide the force to depress a syringe accurately. Finally, as an extra precaution to protect the drive system, a limit switch to prevent the syringe pump from accidentally damaging the lead screw.

The motor has its own regulated supply partially separated from the rest of the logic power. For extra modularity, the stepper motor driver is socketed on its chip carrier board for easy replacement since it is the part that will be most likely to fail first during regular operation. For the rest of the system, all parts are soldered onto a singular, 4-layer printed circuit board.

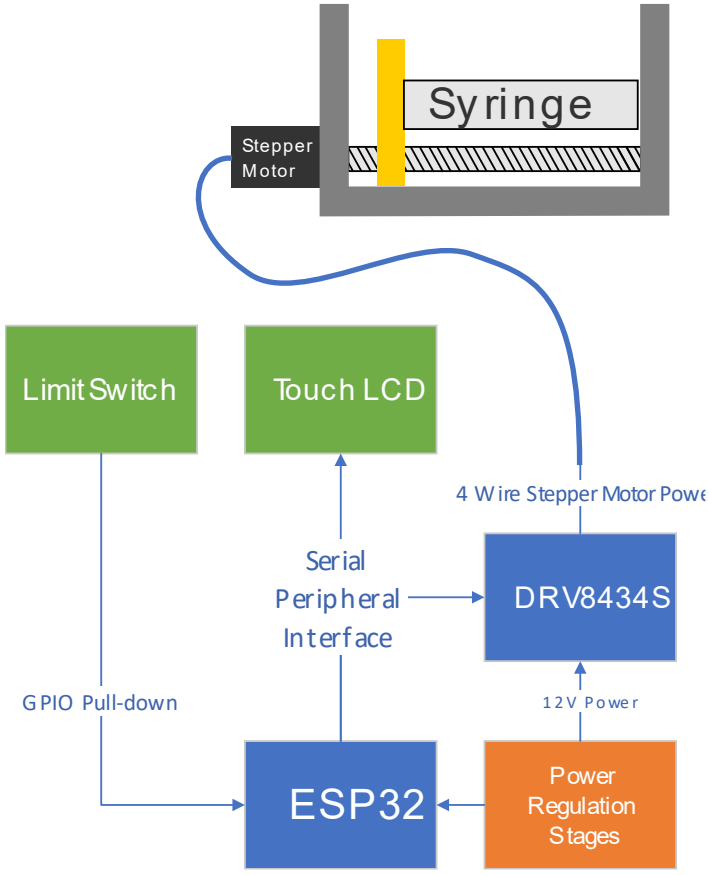

Figure 4. High-Level System Functional Architecture

In Figure 5, we have a much more detailed interaction model for our system. Starting from the bottom, we have two ways to power the system: the USB-to-UART converter and a barrel jack. The USB converter has indicator LEDs for transmitting and receiving data from the

microcontroller. That power is then sent to the two switching regulators, one for regulating five volts for digital logic and one for regulating twelve volts for the stepper motor. The five-volt supply also has a power indicator. The five-volt supply feeds the Class-D amplifier and the 3.3V regulator for the rest of the digital logic on the board. All connections to the ESP32 operate at a 3.3V logic level, so the touch-LCD, the ESP32, and the DRV8434S need the 3.3V reference to function correctly.

Inside the ESP32 block, we can see the software system elements. The green blocks signify real-time operating system (RTOS) tasks. These programs are periodically performed by the microcontroller, allowing more efficient use of the processor's time. The display task is responsible for rendering the user interface and reacting to user input, and the control tasks perform the calculations for the control algorithm as needed. The control task is the first layer responsible for ensuring the system’s health. The control task interacts with our Motor Control Driver subsystem, which is designed to be drop-replaced to maximize the flexibility of the control system. Inside the driver, we have a function to calculate the desired syringe velocity, which is then converted to step frequency and sent to the PWM peripheral on the ESP32. In tandem with the step frequency, we also configure the DRV8434S to have the correct step divider to maximize the precision of the stepper driver motor. Finally, we can control the motor with the PWM, SPI configuration, and 12V from the boost converter.

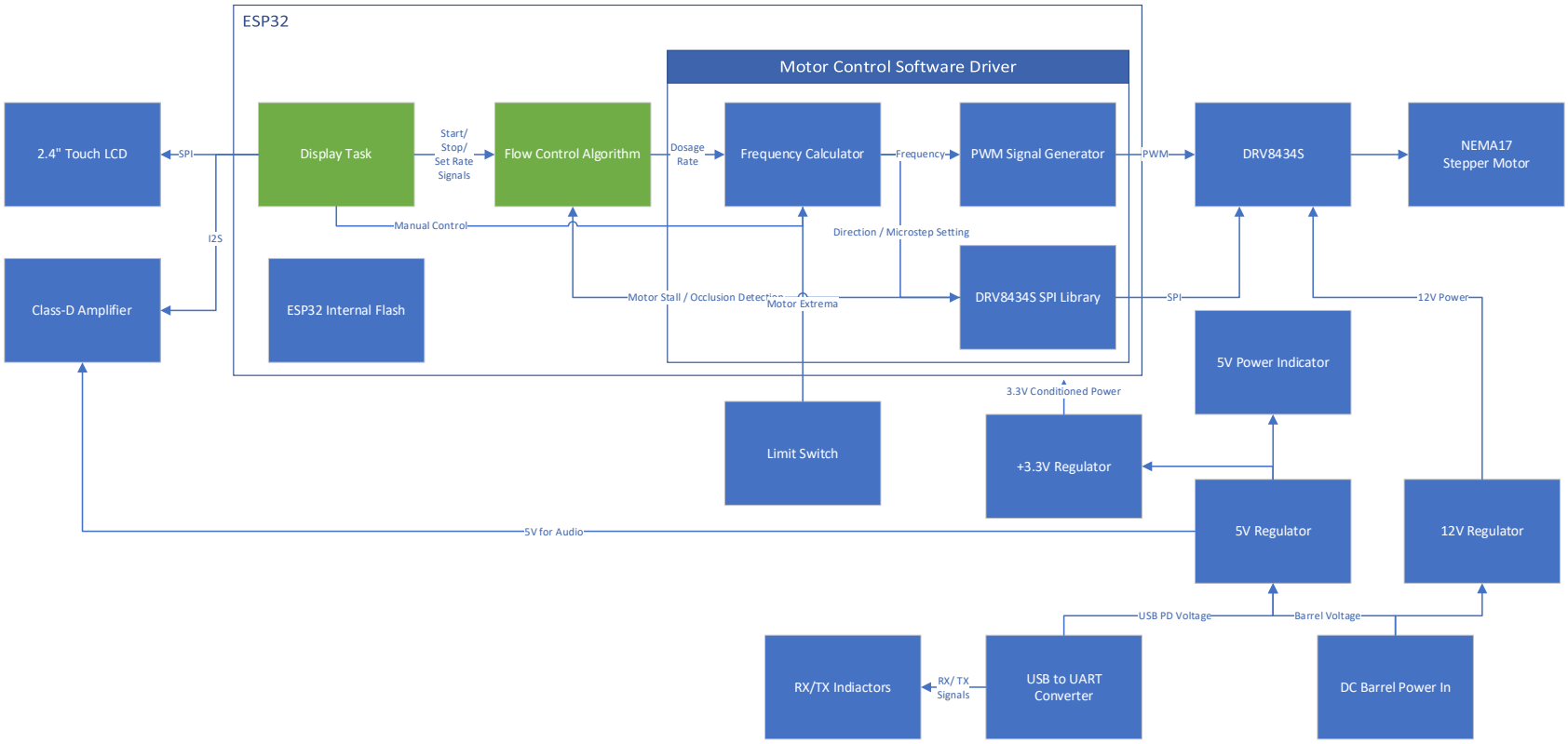

Figure 5. In-Depth Interaction Model of the Final System

2.1.1. Microcontroller & Software

The ESP32-WROOM-32E is an optimal choice for low-cost IoT devices. It features a dual-core CPU, Wi-Fi, PWM signal generation, and pulse counters. The ESP32 is also highly flexible because of its full GPIO matrix, allowing for almost every pin to be configured to have any feature required for this system. Lastly, we use the built-in watchdog timers to ensure the system remains in a valid state. Figure 5 shows an in-depth interaction model of the system with specific part names for essential functions of the system.

Our user interface uses a modern, flexible, and extensible framework called the Light and Versatile Graphics Library (LVGL) [4], allowing future UI extensions to be streamlined and familiar to anyone with previous graphics programming experience. Our software also features an easily extensible application programmer interface (API) that allows for custom driver implementation if users need to add another stepper motor driver to the set of available driver implementations. Along with drivers, more advanced control algorithms could also be implemented following the Control System API, as shown in Figure 6. One of our primary goals with the system was to design an accessible, extensible system, and we believe that our software provides both.

Figure 6. Software Architecture of the Pump Control System

2.1.2. 12V Regulator

We used a positive twelve-volt boost converter to provide consistent power output for the DRV8434S motor driver. We boosted up to 12V for the output motor voltage to balance torque output and current put requirements. The TI LM51551DSSR was used for the regulator because of the rapid design cycle and limited project budget. To attempt to decrease the design time for the regulator, we used Texas Instruments WEBench to quickly find the optimal power supply for the cost and board area. A schematic for the regulator can be found in Figure 7.

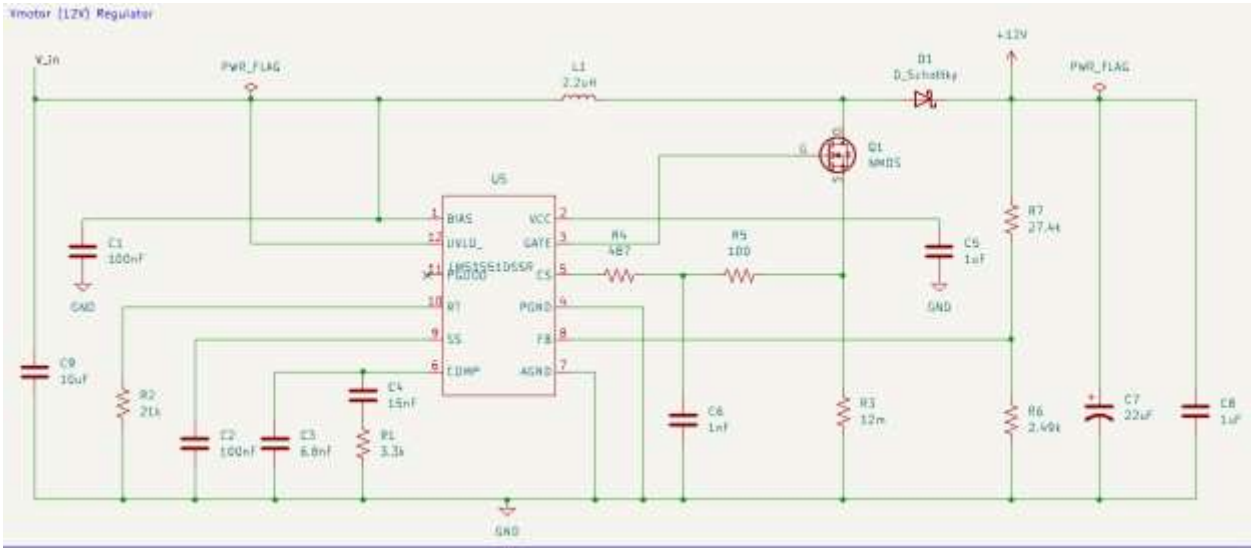

Figure 7. +12V Boost Regulator for Stepper Driver

Unfortunately, during testing, the 12V boost regulator was not functional, and we could not diagnose the cause of this failure due to time constraints.

2.1.3. 5V Regulator

By also choosing to step down and regulate our system voltage into 5V with a switching voltage regulator, it is possible to drive the sound chip with the necessary 5V while additionally acting as an intermediary voltage to be stepped down further to 3.3V for the rest of the system. The circuit for the 5V regulator utilized the TPS530804NM, and the circuit for the power schematic may be found in Figure 8. While the sound chip lacks the software configuration to work actively, the power needed to drive the chip has been set up on the board for future use. A red LED indicator also ensures the 5V regulator is operating in standard. Test points may be added at the 5V flag and input.

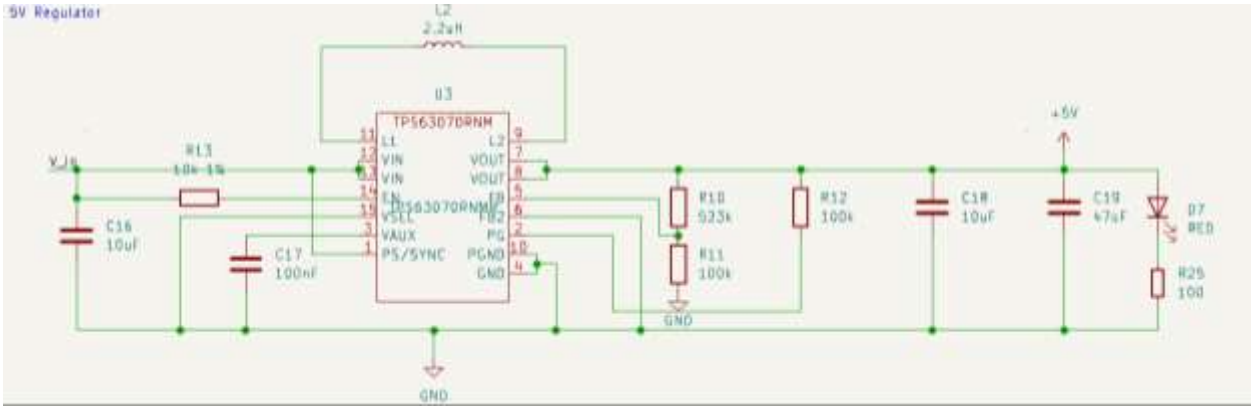

Figure 8. 5V Regulator with Power Indicator LED.

The regulator has been proven to work for our system. D7 is the power indication LED, which is helpful for issue diagnosis.

2.1.4. 3.3V Regulator

The 3.3V regulator plays a crucial role in stepping down the 5V power input using the AMS117[1] chip to power the ESP32, LCD screen, and SD card. The circuitry for the regulator is relatively simple and can be found in Figure 9. The yielded output can be verified by observing the standard operation of all the components utilizing 3.3V.

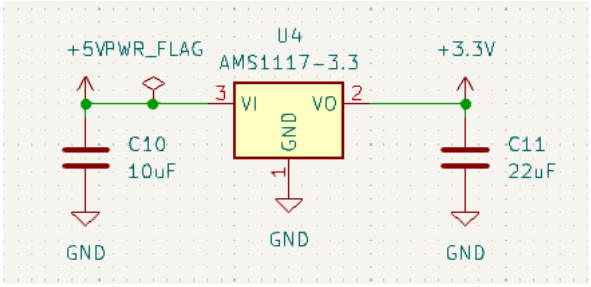

Figure 9. 3.3V Regulator

2.1.5. Class-D Audio Amplifier

By implementing a Class D Audio Amplifier using the MAX98357A, it is possible to ensure that the desired safety mechanisms are met to warn users of potential mishaps in the system. While the hardware has been implemented on the PCB, the software for the sound system has not yet been configured. Should future teams desire to implement this feature, the hardware framework accounts for this potential need. To ensure the success of the implementation, we followed a reference design from Adafruit under the CC BY-SA 3.0 license.

[1] Due to parts availability for AMS117, we currently have the LDI1117-3.3 instead on the PCB.

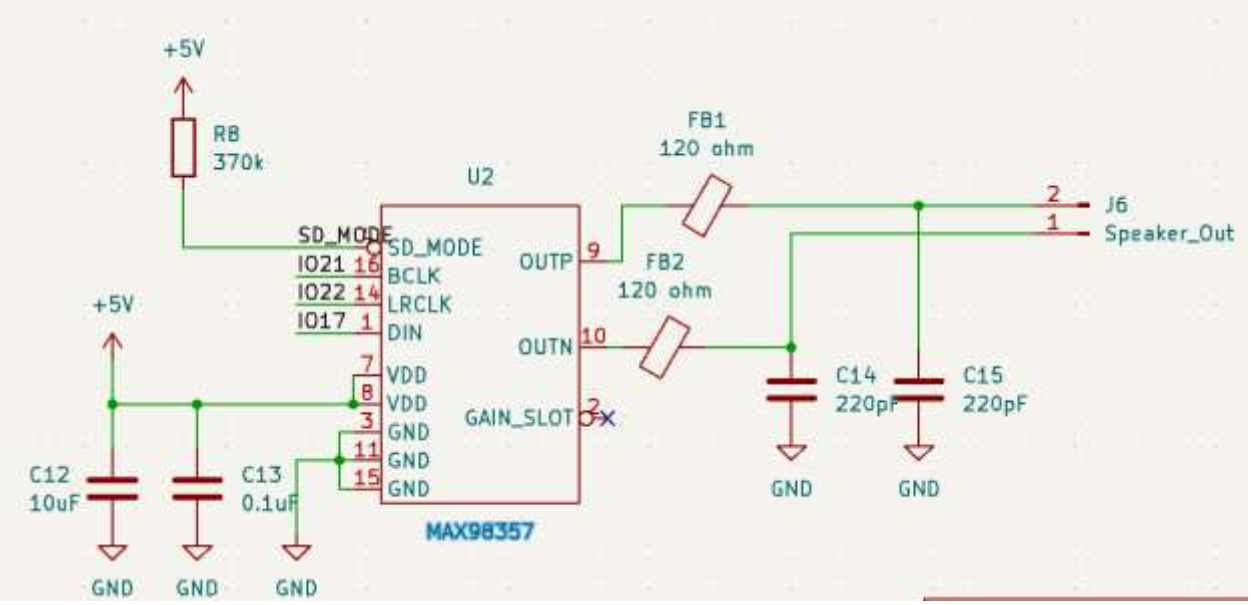

Figure 10. Sound System IC

2.1.6. USB Programming / Serial Out

In most development kits for both ESP32s and Arduinos, there is a chip dedicated to translating UART into USB and vice versa. We decided to add a USB to UART chip to the board for accessible debug output and easy programming for users and technicians. As part of the design, we also wanted to allow USB Power Delivery, a modern standard for power delivery to USB devices. Figure 11 shows our implementation for getting serial communications with a host computer. We opted to use the FT232HP, a USB-PD capable version of the FT232R, which is used in the ESP32 development kit we used. As part of the design, we followed the recommended design from Espressif for programming over USB/UART [5], which involves two NPN transistors to latch the strapping pins on the ESP32. However, since this feature was new to the final PCB revision, we were focused on troubleshooting the control system to be able to diagnose why our computers did not detect the converter.

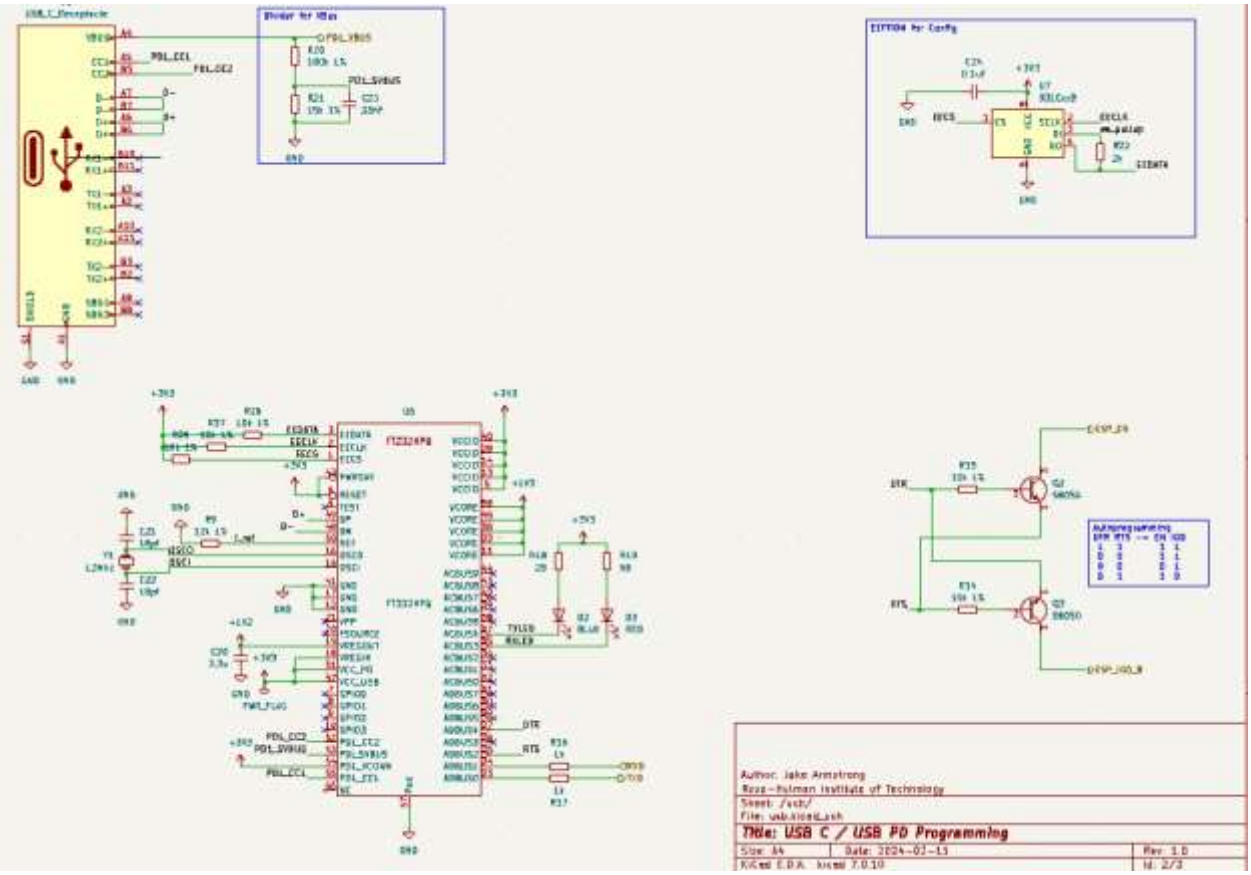

Figure 11. USB-C Power Delivery Schematic

2.1.7. LCD Touch Display

Due to its resistive touch functionality, we chose the 2.4” TFT IL9341 LCD touchscreen for our display. In medical settings, resistive touch screens may minimize the impact of conductive fluids and can be easier to use with gloves to maintain sterility. We decided to use this display for its low cost, high availability, and robust driver support for embedded systems. The display uses the SPI bus to communicate with the microcontroller, thus making the overall development cycle much quicker. This also allows the same IO pins on the ESP32 to be shared between the stepper motor driver and display to minimize the GPIO used. The display can be socketed directly onto the motherboard, allowing easy installation and maintenance. The display has shown usability in simple user testing, but the viewing angles are limited, impacting usability.

2.1.8. SD Card

The SD card is a simple addition to the board with a significant value added by allowing extra non-volatile storage to the system with an easily removable SD card. A removable SD card has been integrated into the PCB to provide system logs and status information. SD cards can be configured to run on the SPI bus, making it a logical solution for long-term logging and data storage. Currently, testing of the SD card has not been performed, but the hardware has been

installed on the board. Future work will include using the dedicated SD port on the ESP32 for improved performance.

2.1.9. Stepper Driver Module & Daughter PCB

The stepper driver board to mount the driver chip was separated from the primary PCB to implement a modular design. The stepper driver chip selected was the DRV8434S by Texas Instruments for its stall detection capabilities, PWM controllability, potential for occlusion detection, and power requirements. While we initially had our stepper motor input power supply 12V via a buck-boost converter, the stepper motor will have to use unregulated 9V input voltage from the barrel jack connector. Ideally, this regulated 12V would allow for a consistent torque from our motor under most conditions. The limit switch was connected to the stepper driver board, with distinct header pins of signal inputs and motor outputs. At a dimension size of 2x4 inches, the stepper driver board meets the feature requirements for portability and hardware maintainability. A copy of our initial circuit schematic can be found below in Figure 12.

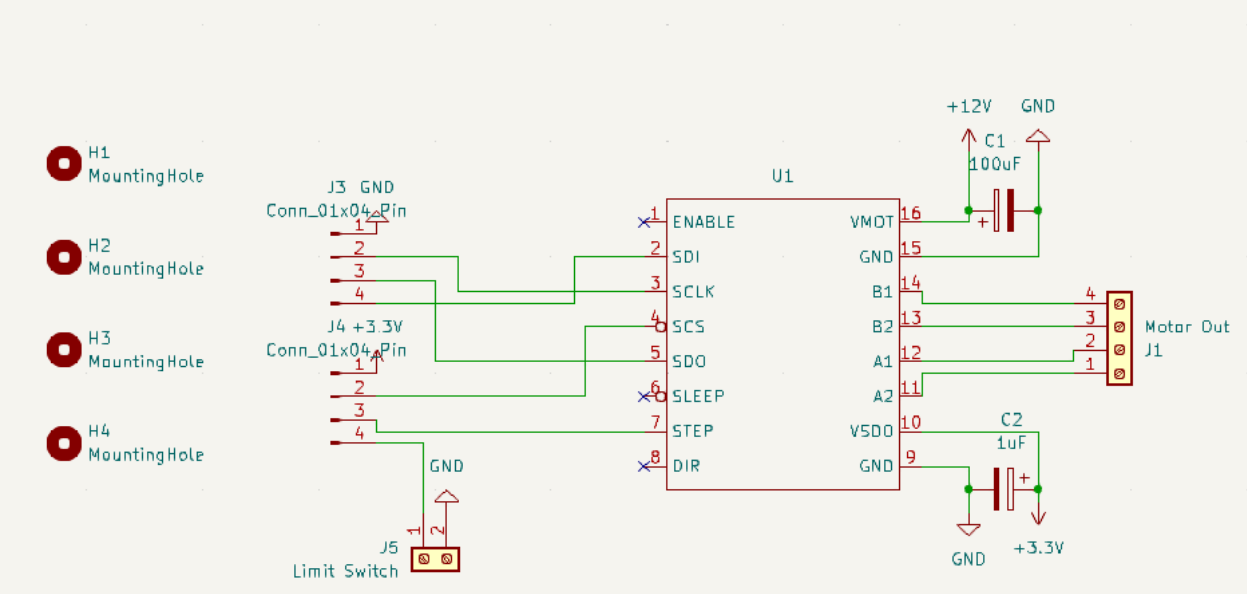

Figure 12. Stepper Driver Circuit

2.2. Verification of Requirements

The Test Plan verifies the system's top three features: ease of use, safety, and UI. Ensuring patient safety by verifying accuracy in dosage rates, bolus rates, occlusion detection, and end-of-delivery detection is crucial in determining whether the electrical subsystem can address the user's needs. Testing for customization demonstrates the ability for users to change graphical settings and syringe rates easily. The electrical subsystem will be integrated into a mechanical housing designed by the ME team assigned to the project. It must be thoroughly tested to ensure the system meets the user's needs.

The following features in the system have been verified through experiments and demonstrations. The feature ID helps identify the system-level features implemented by the OSMI project and their corresponding test requirements in the table below.

Table 4. Testing Requirements

Feature ID	Requirement	Validation	Achievement
EOU.3a	The control system will generate a PWM signal with a frequency between 80 and 9.6k Hz with an accuracy of +/- 5mHz.	Experiment	Passed
EOU.3b	The control system should switch between the bolus and infusion rates at the 1-minute mark of the 2-minute infusion duration with an accuracy of 1s.	Experiment	Failed
SF.1a	The control system must detect a torque load of 0.05 N*m applied to the motor.	Experiment	Passed
SF.1b	The control system should receive a signal from a limit switch when the syringe movement has ended.	Experiment	Passed
UI.1a	The user interface shall display each channel's dosage with a three-digit display between 0.1ml/min and 99.9ml/min.	Demonstration	Passed
UI.1b	The user interface shall display the channel’s remaining volume as a percentage accurate to within 1% percent.	Demonstration	Passed
UI.1c	The user interface shall display the channel’s remaining bolus dosage (the higher delivery rate dosage at the beginning of delivery) within 0.1 mL.	Demonstration	Passed

Table 5. Feature Identification, Attributes, and Verification Status

Feature	ID	Attribute/Metric	Verification
Ease of Use	EOU.1	A well-trained end-user should be able to be fully trained in how to use the device in under one hour.	Incomplete
Accessibility	EOU. 2	The system should be easy to learn and use within a limited time.	Incomplete
Customization	EOU.3	Control over how much material the syringe delivers, medication dosage, infusion rates, etc.	Experiment
Safety	SF.1	The system should function in a manner that will not cause or pose a risk of harm to its users or beneficiaries.	Experiment
Durability	DB.1	The device should be robust and not break easily when subject to physical stress.	Incomplete
Affordability	AF.1	The cost per singular unit of the device should be affordable to ensure industrial replicability. The total cost should aim to be under $100 per channel.	Demonstration
Hardware Maintainability	HD.1	Essential components of the system that fail or need to be interchanged should be able to be repaired or replaced quickly.	Demonstration
Portability	PT. 1	The device should be able to be transported easily and remain functional.	Demonstration

2.2.1. Experiment 1 – Control System Pulse Frequency Consistency (EOU.3a)

For this experiment, we are testing the reliability and functionality (EOU.3A) of the control system to assess how well it can maintain a target pulse frequency as a stand-in for our target dosage rate. This test aims to verify the accuracy of the dosage rate across different frequencies and at various time intervals during the medicine delivery process. A range of frequencies from 80Hz to 9.6kHz was used as a testing interval. If all frequency data points from motor feedback fall within +/-0.05Hz of the dosage rate setpoint for the frequency range of 80Hz, 101.3Hz, 122.667Hz, 160Hz, 896Hz, and 9.6kHz, the system has passed the test.

2.2.2. Experiment 2 – Bolus to infusion rate consistency test (EOU.3b)

This experiment aims to confirm the control system will consistently change from its bolus rate to its standard infusion rate. The bolus rate is commonly used to get a patient's dosage concentration to a target rate, while the standard infusion rate is used to maintain the concentration. This experiment was performed by setting a desired initial bolus rate and a typical dosage rate for the system to switch to. Calculating the expected time for which the pump will switch between rates is possible. By measuring the actual time at which the pump can switch between rates, this value can then be compared to its desired result. It fails if the system does not change within +- 100ms of the expected time.

2.2.3. Experiment 3 – Torque Limit Test (SF1.a)

The torque limit test is meant to emulate what the system will see if the IV tube is occluded. This essential safety feature allows the pump to detect faults in the motor's operation or any external unintended forces acting on the system. We can verify this by configuring the control system to respond to a known torque load. The test was performed using a pulley system and a lab weight set, which could secure onto the motor using a 3D-printed attachment. The experiment was performed by setting a manual torque threshold and moving the motor at 1ml/hr. A series of weights were used with torques varying from 0.009Nm to 0.09Nm. For the test to have passed, the motor should have automatically ceased function when trying to turn when a force applied by the weight acts on it.

2.2.4. Experiment 4– Limit Switch Stop (SF.2)

This experiment confirms that the control system will stop when the limit switch is activated. A limit switch will act as a way for the pump to detect when it has reached the end of delivery and, therefore, stop turning. This reduces errors with control system data and saves power when the control system is unused. The test was performed by measuring the PWM signal sent to the motor and the signal produced by the limit switch. Measurements were taken directly from the stepper driver board. By measuring the distance between the limit switch signal toggling high and the PWM signal flattening, it is possible to test whether or not the PWM signal meets its desired specifications. For the system to pass the test, the desired specification ensures that the PWM signal ceases within 500ms of the pressed limit switch.

2.2.5. Demonstration 1 – UI Demonstration (UI1)

The UI demonstration will validate that an assigned dosage rate can be set and displayed on the LCD screen. Infusion rate, remaining volume, and bolus settings shall be displayed on the LCD screen as part of the UI verification specifications. This demonstration can, therefore, confirm that the LCD screen can be configured without user bugs to ensure that the product is accessible and easy to learn by a new user.

2.2.6. Test Results and Conclusions

Control System Pulse Frequency Consistency (EOU.3a)

A sampled average of the PWM signal was obtained at six time samples to demonstrate that the control system can produce a repeatable dosage rate with an error of +/-2%. The results were then reproduced at different sample rates, as indicated by Table 6. The requirement ensures the stepper motor successfully implements the frequency translated by the control algorithm to the PWM peripheral. The deviation from the expected value produces a differential error from the expected linear dosage rate. According to Figure 13, the measured stepper frequency remained consistent across the delivery duration, ensuring that the control system yields a stable and consistent output.

Table 6. Experiment 1 Data

Experiment 1		Ambient Temp = 22 deg C									
Dosage Rate	30ml/hr (80Hz)	Dosage Rate	38ml/hr (101.3Hz)	Dosage Rate	46ml/hr (122.667)	Dosage Rate	60 ml/hr (160Hz)	Dosage Rate	336ml/hr (896Hz)	Dosage Rate	3600 ml/hr (9.6kHz)
Time (s)	**Rate (Hz)**	**Time (s)**	**Rate (Hz)**	**Time (s)**	**Rate (Hz)**	**Time (s)**	**Rate (Hz)**	**Time (s)**	**Rate (Hz)**	**Time (s)**	**Rate (Hz)**
30	80	30	101	30	122.97	30	159.95	30	896.057	30	9602
60	80	60	101	60	122.97	60	159.95	60	896.057	60	9601
90	80	90	101	90	122.97	90	159.95	90	896.057	90	9602
120	80	120	101	120	122.97	120	159.95	120	896.057	120	9601
150	80	150	101	150	122.97	150	159.95	150	896.051	150	9602
180	80	180	101	180	122.97	180	159.95	180	896.051	180	9601
Dev	**0**		**-0.3**		**0.3**		**-0.95**		**0.055**		**1.5**

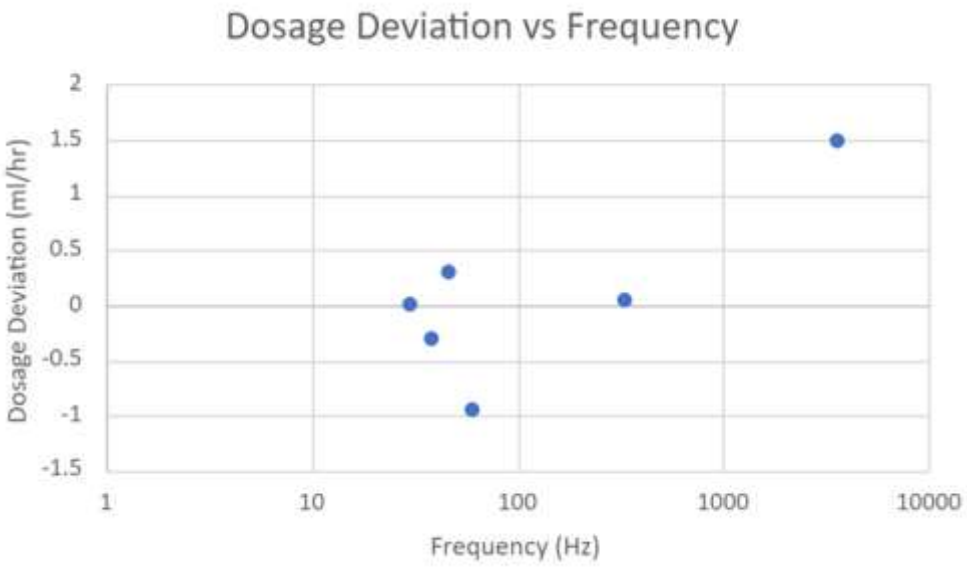

Figure 13. Dosage Deviation vs Frequency

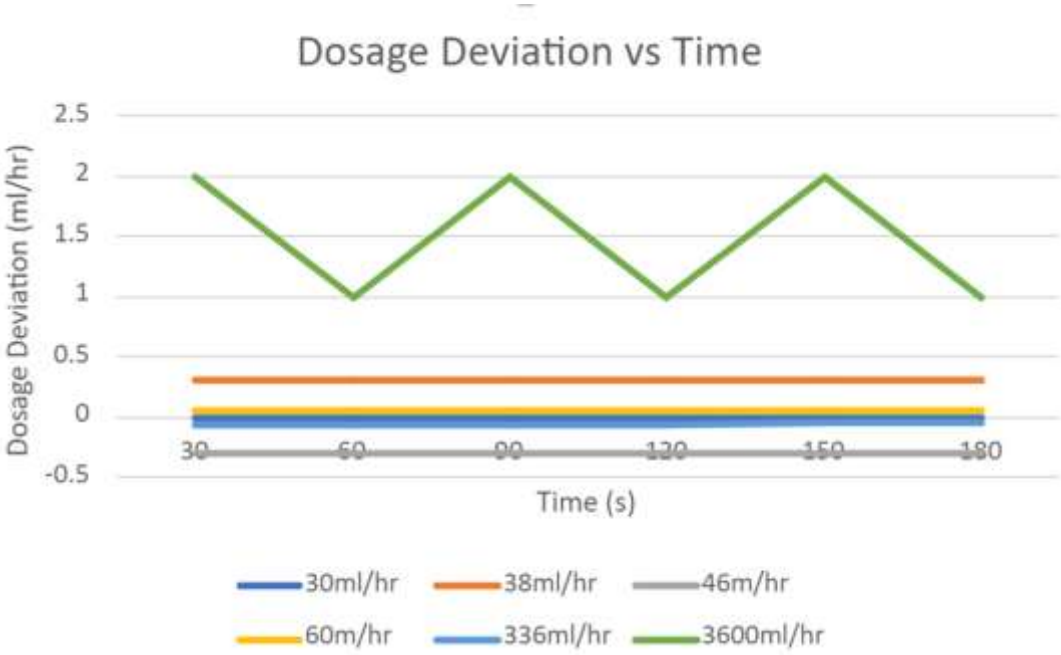

Figure 14. Dosage Deviation vs Time

Bolus to infusion rate consistency test (EOU.3b)

The bolus rate allows for a customizable initial infusion rate separate from the standard dosage rate, which the system defaults to upon delivering the bolus dosage. Five tests were performed on pairs of standard dosage and bolus rates to test for the bolus rate. The measured time indicates when the bolus dosage rate switches to the standard dosage rate. Five tests were performed on

pairs of standard dosage and bolus rates to test for the bolus rate. The average time taken for the system to switch from the bolus rate to standard infusion was roughly 16s, which fails the requirement. This may be insignificant for blood transfusions but significant for complex surgical procedures.

Table 7. Experiment 1 Data

Experiment 2					
Dosage Rate (ml/hr)	Dosage Volume (ml)	Bolus Rate (ml/hr)	Bolus Volume (ml)	Measured Time (s)	Expected Time (s)
18	4.4	82	4.1	261.04	240.00
30	4	70	3.5	250.84	240.00
48	3.4	52	2.6	256.98	240.00
60	3	40	2	255.22	240.00
72	2.6	28	1.4	258.67	240.00
84	2.2	16	0.8	254.84	240.00

Torque Limit Test (SF1.a)

To verify that the system can detect occlusions in the syringe, we used the pulley to simulate blockages in the pump. The motor did not stall until the 70 g weight was applied (0.0311 N*m). At 70 g, 100 g, and 200 g, which provided 0.0311 N*m, 0.0444 N*m, and 0.0888 N*m torques, respectively, we observed that the motor recognized that it was stalling and that the occlusion detection message programmed into the code was sent to the serial monitor. Therefore, the tests passed.

Table 8. Torque Limit Data.

Mass	**Torque (N*m)**	**RESULT**
20 g	0.00887	FAIL
50 g	0.0222	Fail
70 g	0.0311	PASS
100 g	0.0444	PASS
200 g	0.0888	PASS

Limit Switch Stop (SF.2)

The limit switch test verifies that the pump will stop delivering medication once the syringe contacts it. The time for the PWM to halt after the limit switch was pressed is compared to the expected time for it to cease delivery. The system passed the limit switch test as it could stop the PWM signal in 5.3ms.

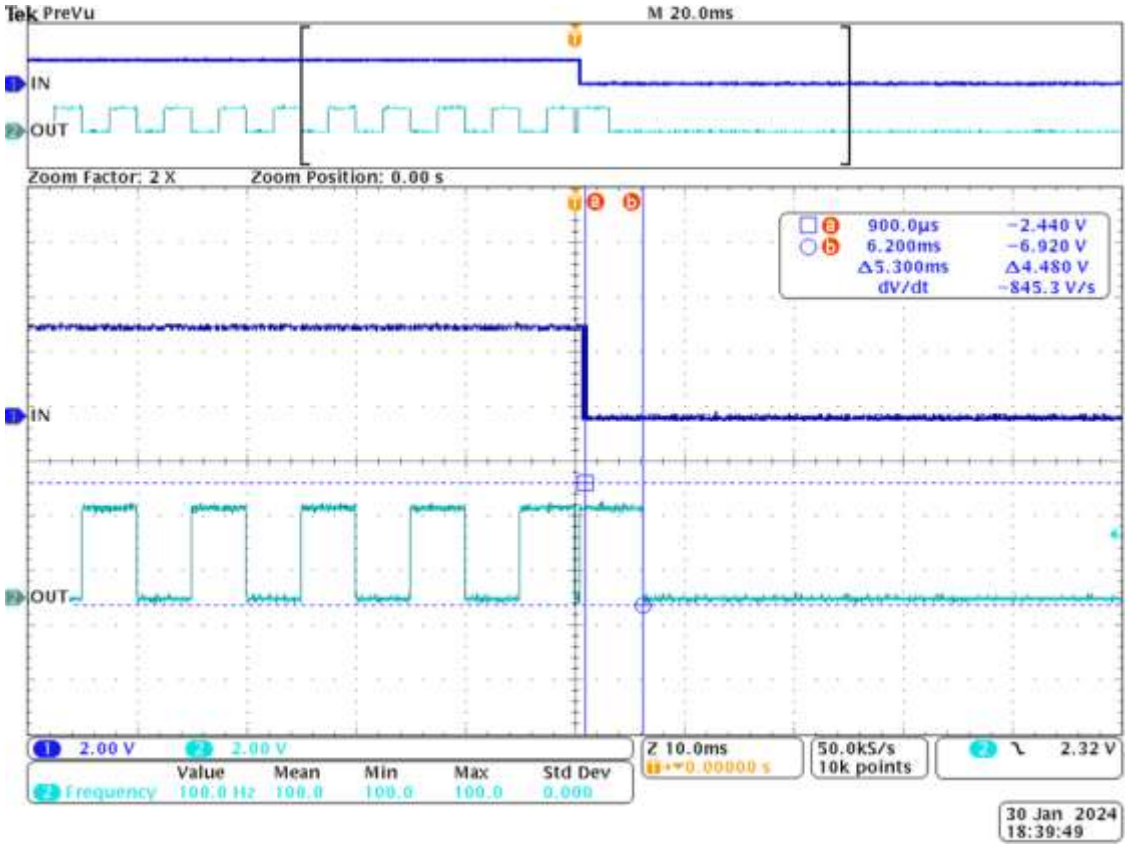

Figure 15. Button Input and PWM Signal

Figure 16.

Table 9. Demonstration 2 Results

Demonstration 2	
Time for the PWM signal to stop	5.3ms
Expected Time	500ms
Pass/Fail	PASS

UI Demonstration (UI1)

According to Figure 17, the standard dosage rate and volume can be found in its associated widgets and adjusted accordingly. The bolus dosage rate and volume can also be found in the secondary widgets and adjusted as necessary. The figures validate specifications UI1.a AND UI1.c, which allow the user to manually change the syringe pump dosage rates, fulfilling an essential system requirement.

The screen capture in Figure 17 will be displayed to the user during medication delivery. The duration of medication, dosage settings, remaining volume, and the start and stop buttons will be displayed to the user. The figure, therefore, validates specification UI1.b. All figures have been obtained using an LVGL UI simulator for demonstration purposes.

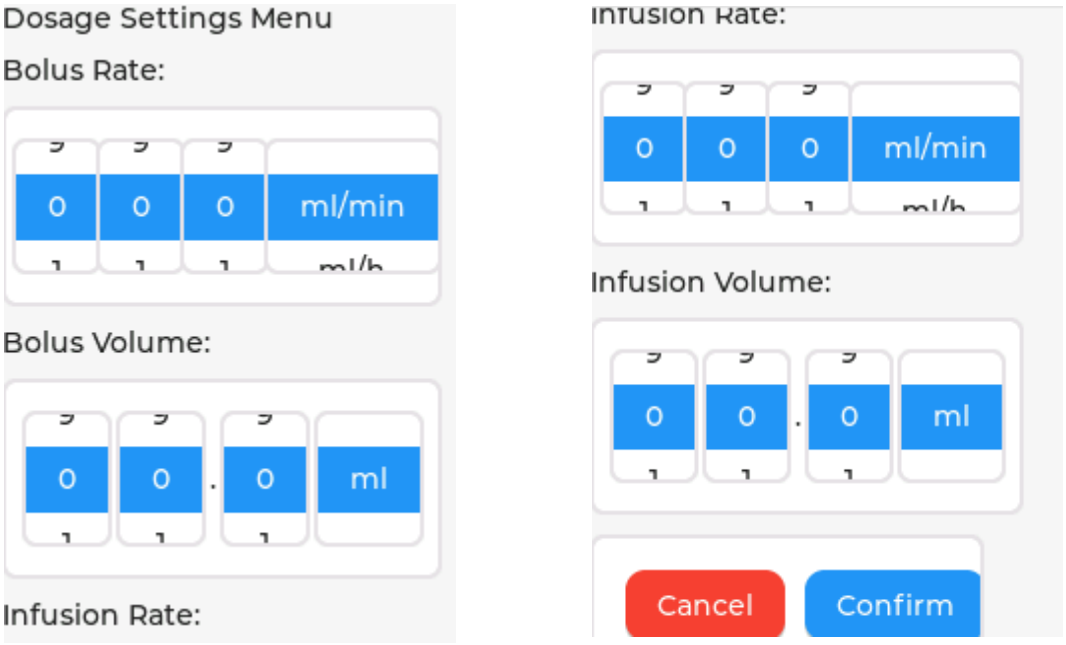

Figure 17. Dosage Settings Menu

3. Project Plan and Timeline

The project plan and timeline aim to provide the reader with an overview of how the project was projected to progress compared to the outcome of the work delivered. Determining realistic goals by the team has been a consistent point of struggle with the team due to the large scope of the project. An initial prototype developed in the fall quarter acted as proof of concept and proved to be a setback due to the need for design revisions during the Winter Quarter. Therefore, the setbacks and adjustments to the project plan will be covered in detail in this section to provide perspective into the design process.

3.1. Original Plan

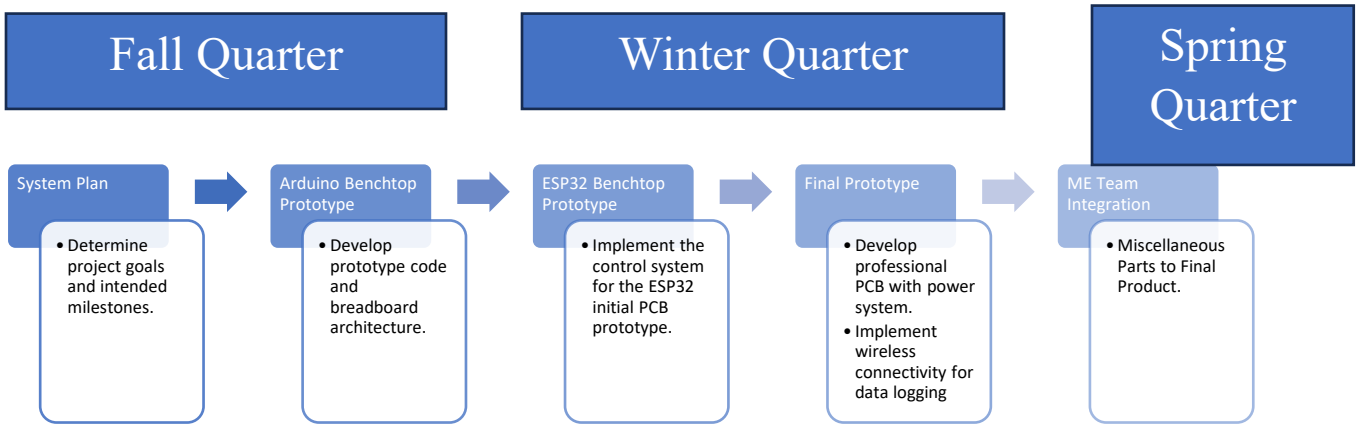

Figure 18. Original Timeline

The original milestone plan for the fall quarter involved developing a system plan and Arduino benchtop prototype. This prototype utilized Arduino code to drive a stepper motor and a breadboard circuit for the LCD and ESP32 dev kit. The UI initially used the TFT display drivers instead of the latter LVGL implementation.

During the winter quarter, we anticipated designing our final prototype in addition to the wireless interface. Our final prototype will be yielded for ME integration for the spring quarter. While we achieved a functional, variable dosage rate prototype, our 12V regulator and USB PD failed to meet the desired functionality. Most software features needed to be dropped because of delays with implementing a functional control system during week 5. These delays resulted in a team decision to forfeit design features such as SD card logging, Sound and Wireless connectivity.

3.2. Assigned Tasks

The initial spread of tasks was allocated such that Jake was primarily in charge of the first version of the PCB design, Tyra oversaw the TFT driver display for the UI and research on the power delivery, Wei implemented the stepper driver configuration via Arduino, and Gabe designed the control system for the syringe pump. This spread of work did not effectively utilize the individual skillsets within the team and was later readjusted as per section 4.5.

Table 10. Initial Member Roles & Responsibilities

Team Member	**General Roles and Responsibilities**
Jake	Primary Responsibilities: PCB Design.
Tyra	Primary Responsibilities: UI and Power delivery.
Wei	Primary Responsibilities: Stepper Driver Configuration
Gabe	Primary Responsibilities: Control System Design (Electronics & Software) and Documentation

3.3. Milestones and Timeline

Table 11. Milestones

Milestone Name	**Description**	**Verified by**
ESP32 Bench Top Prototype Parts Order	The ESP32 Development Kit, LCD screen, Stepper motor, Lab kit including breadboard, wires, power supply, and any other components required for the project should be provided or ordered.	Receipts from the websites and companies the parts are ordered from.
Arduino Breadboard Prototype with Buttons, Graphical Text Display, and Single Stepper Control	A prototype for the system must be developed that consists of: • A control system that regulates a singular stepper motor to rotate it one revolution/hour. • A standard power supply that powers the system. • A user interface programmed to an LCD screen that displays data from the syringe and can be used to manage user preference settings such as changing menu screens and the rate of the syringe pump.	Confirm that the syringe rotates at one revolution/hour by measuring its speed over time. Observe that the power supply can deliver the power required to each component in the system to function as intended. Operate the user interface to ensure

	The device will be implemented using Arduino software and hardware for the microcontroller functions.	the functions work and manipulate the system as intended. Ensure that each software component is programmed using Arduino software.
ESP32 Bench Top Prototype with Milled Prototype PCB and Simple Graphical Widgets for Control and Simple Alarm sound	A prototype for the system must be developed that consists of the system functions outlined for the Arduino prototype but is now programmed and developed using ESP32 hardware and software. In addition to the previous tasks that will be transferred to the ESP32 prototype: • A prototype PCB that will supply power to the system must be developed and milled. • Graphical widget animations that will enhance the visual aesthetic of the system will be included in the user interface. • A pause/start feature that will stop and start system functions at the user's discretion. • An alarm will be programmed into the system to be sounded under austere conditions.	Confirm that the syringe rotates at one revolution/hr by measuring its speed over time. Observe that the PCB can deliver the power required for each component in the system to function as intended. Operate the user interface to ensure the functions work and manipulate the system as intended. Activate the pause/resume feature and observe that the system functions can start and stop as intended. Ensure that each software component is programmed using Arduino software.

Final Etched Prototype PCB with external RTC, Watchdog Timer, and wireless connectivity.	The final PCB prototype for the system must be completed. Additionally, the following features should also be programmed into the existing system: • Enable a Real-Time Clock that displays the time on the graphical interface. • Write logs to an SD card with the date & time of the system. • Enable watchdog timers that monitor the program to ensure all functions are proceeding correctly and will sound the alarm if otherwise. • Improve the GUI to allow for different drug delivery settings & set the current time of the procedure.	Compare the clock's time with the time recorded on an iPhone to determine that the time is accurate and is proceeding at the same pace. Analyze the SD cards to confirm that the data has been written to and stored within the card's memory. Set the program to deliberately trip the errors on the watchdog timers and assess that they sound the alarms as intended. Use a timer from an external mobile device to confirm the system's timer is accurate. Confirm that the system functions can be controlled as intended using a wireless controller.
Prototype Testing, Integration, and Documentation	The final prototype will be complete and ready to be integrated with the ME team's design. • All system functions will be implemented and consistent. • All system functions will be combined into one system and can work together.	End-user, unit, and electrical characteristic testing will be completed and verified.

	• Additional safety programs and quality-of-life enhancements will be incorporated to reduce errors during system use. • Thorough documentation of the system's functionality and design with instructions on using it will be completed.	Errors will be intentionally simulated to confirm that the safety programs will function as intended. Necessary system tests will be repeated after quality-of-life enhancements have been included to ensure that the system will still work as intended. The system will be tested on people unfamiliar with the device to assess the accessibility and simplicity of its user interface.

3.4. Adjustments to the Project Plan

Multiple adjustments were made to the project milestone plan during the project due to a failure in estimating the workload per milestone. The development of the control system took longer than anticipated to debug and therefore delayed multiple software features such as fault alarms, Wi-Fi connectivity, and SD card logging to future work by other teams.

However, the infrastructure for the fault alarms, Wi-Fi connectivity, and SD card have been accounted for in the hardware for the board. These features may be easily implemented and further expanded upon. Please refer to the Recommendations section for further information.

The first of our initial adjustments to the plan would be on what tasks people would initially be designed. When we first started assigning roles to the project, we had asked electrical engineers to work on software while a computer engineer was working solely on PCB design. While we eventually resolved this issue, having our tasks sorted out earlier would have been beneficial. In the future, we will be more steadfast in our abilities as teammates to help reduce the amount of time wasted due to inexperience.

Another adjustment we should have made to our initial project plan was to axe any mention of a salvaged power supply. At the start of this project, we sought to demonstrate that a wide range of power supplies would support our device, which is not actually critical to the project itself. We could have had more development time had we focused entirely on the system. In the future, we will organize what is critical to the project more so that distracting portions of our goals do not interrupt our project development.

3.5. Adjustments to Tasks

During the quarter, tasks were rotated to efficiently utilize our team members' skill sets. The reassigned roles and responsibilities have been formatted in Table 12 below.

Table 12. Team Member and Responsibilities

Team Member	General Roles and Responsibilities
Jake	Primary Responsibilities: Software UI & Control System Electronics
Tyra	Primary Responsibilities: PCB design and assembly, Documentation
Wei	Primary Responsibilities: Software UI & Control System Electronics
Gabe	Primary Responsibilities: Control System Design, ECE representative to ME team, Documentation

4. DEVELOPMENT BUDGET AND SPENDING

One of the many goals of this project is to design a low-cost syringe pump, defined in our proposal to have a total cost of one hundred dollars per syringe pump in electronics. We projected a hardware development budget of three hundred dollars for overall development costs, thus making the materials used ultra-low cost for developing this device.

4.1. Free and Open-Source Projects Used

Many tools used throughout this project's development were free and open-source. Thus, no license fees apply to the following: Visual Studio Code, PlatformIO, ESP-IDF, FreeRTOS, TFT_eSPI by Bodmer, the Light-weight Versatile Graphics Library (LVGL), DRV8434S-arduino by Polulu, KiCad EDA, and PrusaSlicer.

Other essential free tools used were Zadig, a generic USB driver tool critical for using debugging tools such as Open On-Chip Debugger (OpenOCD) and GDB.

4.2. Overall Project Materials & Equipment Costs

The overall project costs include all costs associated with the construction, engineering, testing, and prototyping of the OSMI electrical hardware and software.

Table 2 includes a high-level cost and part cost used throughout the project. This includes our Digikey orders that will be broken down in Section 0 below10, PCB Bill of Materials. All equipment used has been accounted for to provide a detailed evaluation of the costs associated with the OSMI project.

For paid software, we used SolidWorks to assist in manufacturing mechanical components needed for occlusion testing. This program was obtained through a license from the university and would cost the team $2820 for a 12-month license. Additionally, we use Cadence PSpice, also provided by the university, which would have cost the team an additional $2,400 for a 12-month license.

Table 13. Overall Project Materials & Equipment Costs

Row Labels	Quantity	The sum of Cost per each (USD)	The sum of the Total Cost (USD)
Tools	**4**	**$3,820.94**	**$3,820.94**
Aoyue Soldering Iron	1	$355.95	$355.95
E3631A 80W Triple Output Power Supply	1	$965.00	$965.00
Rigol MSO5102	1	$1,500.00	$1,500.00
Saleae Logic Analyzer	1	$999.99	$999.99
PCB	**11**	**$166.84**	**$247.66**
Final Prototype Board	5	$6.32	$31.60
Final Prototype Stencil	1	$10.72	$10.72

Shipping JLCPCB	1	$51.31	$51.31
Digikey Shipping	1	$28.00	$28.00
Chip Quick Solder Paste	1	$14.95	$14.95
Final Materialsice Parts	2	$55.54	$111.08
Software	**3**	**$5,679.00**	**$5,679.00**
Solidworks	1	$2,820.00	$2,820.00
Mini Prusa 3D Printer	1	$459.00	$459.00
Pspice	1	$2,400.00	$2,400.00
Materials	**16**	**$141.93**	**$188.72**
Breadboard	2	$5.00	$10.00
DRV8434S	4	$12.94	$51.76
ESP-Prog	1	$15.00	$15.00
Extra Final PCB Parts	1	$60.00	$60.00
Miscellaneous Parts	1	$30.00	$30.00
V1	1	$6.00	$6.00
V2.1	1	$6.00	$6.00
V2.2	1	$6.00	$6.00
DoIT ESP32-Materialskit-V1	4	$0.99	$3.96
Grand Total	**34**	**$9,808.71**	**$9,936.32**

Figure 19. Pie Chart of Material Distribution Cost

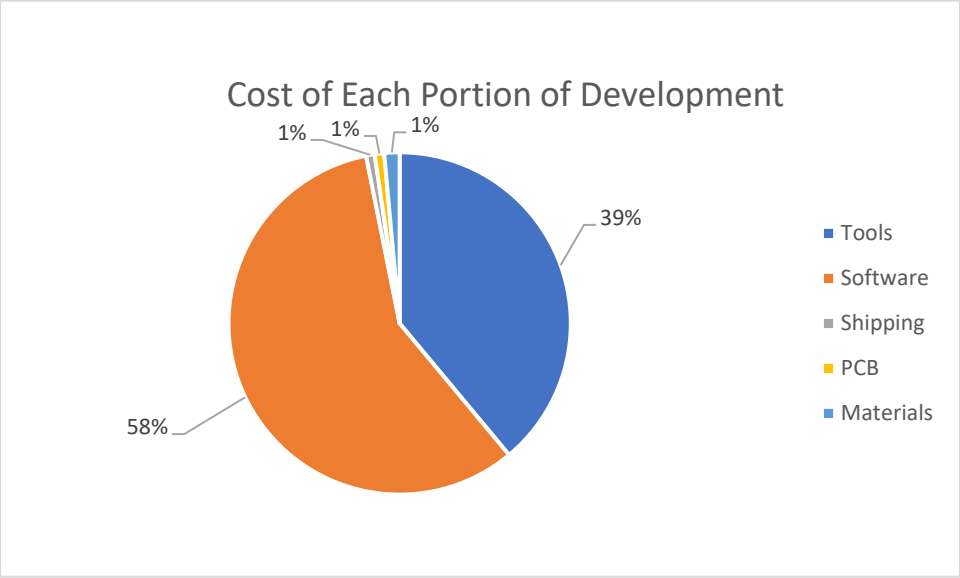

Figure 20. Actual Cost Breakdown

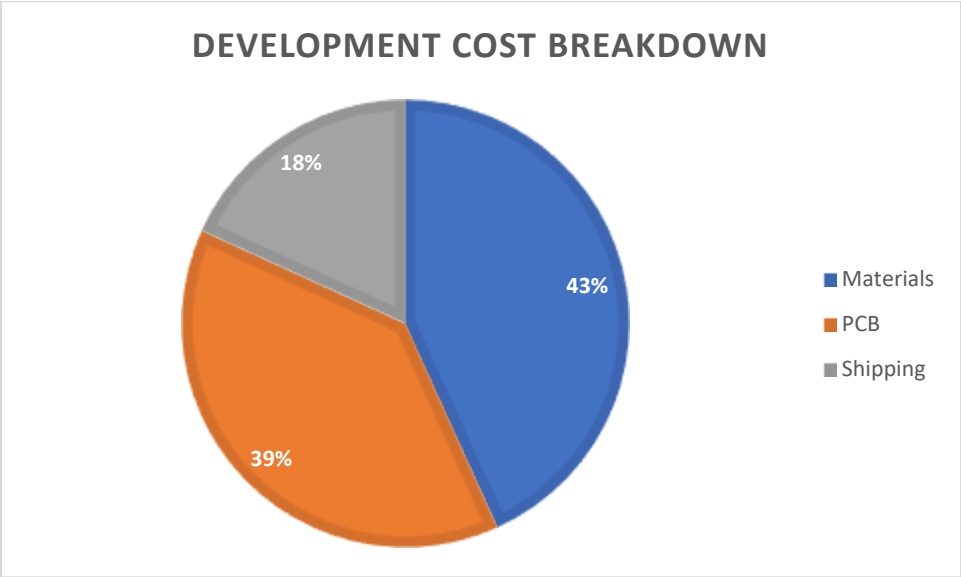

Table 14. Material Cost Distribution

Cost Type	**Total Cost**
Final Prototype	$195.99
Early Prototyping	$95.07
RHIT Equipment	$6199.94
Other	$96.31
Total	**$7487.31**

4.2.1. Labor Costs

The labor costs account for the work hours put into by the team, as shown in Table 15, in addition to faculty or technician hours used, as shown in Table 16. A cumulative breakdown of labor costs can be found in Table 17, along with a pie chart in Figure 21 for assistance in visual representation.

Table 15. Weekly Breakdown of Student Labor Costs

Week #	**Jake's Hours**	**Tyra's Hours**	**Gabe's Hours**	**Wei's Hours**	**Total Hours**	**Total Cost**
1	4	4	4	4	16	$320
2	13.75	10.75	6.25	12.75	43.5	$870
3	7.5	8	6	10	31.5	$630
4	15.5	10.25	9.5	3.75	39	$780
5	11	7	13.75	9	40.75	$815
6	7	6	6.5	7	26.5	$530
7	12	10	9.25	11.5	42.75	$855

8	9.5	7	7	6.75	30.75	$615
9	18.75	7	11.75	16	53.5	$1070
10	11.5	9	9	9	38.5	$770
11	13.25	13	11.25	6.75	44.25	$885
12	23.5	21	16	10	70.5	$1410
13	21.5	15	16	10	62.5	$1250
14	18.25	12	10.5	10	50.75	$1015
15	18	12	10	12	52	$1040
16	15.5	10	10	14	49.5	$990
17	28.5	26.5	25.75	6	86.75	$1735
18	46.75	35	27	50	158.75	$3175
19	29	27	32	30	118	$2360
Total	**324.25**	**251.5**	**241.5**	**238**	**1055.25**	**$21105**

For the team to execute the project, faculty time was consulted to gain advice on control system development, grant proposals, and additional guidance on software debugging. Additionally, technician time was used during board assembly and hardware debugging.

Table 16. Faculty, Technician, and Engineer Costs

Week #	Faculty Hours	Faculty Cost	Technician Hours	Technician Cost	Engineer Hours	Engineer Cost
1	0	$0	0	$0	0	$0
2	0	$0	0	$0	0	$0
3	0	$0	0	$0	0	$0
4	0	$0	0	$0	0	$0
5	0	$0	0	$0	0	$0
6	0	$0	0	$0	0	$0
7	0	$0	0	$0	0	$0
8	0	$0	0	$0	0	$0
9	0	$0	1	$50	0	$0
10	0	$0	1	$50	0	$0
11	0	$0	0	$0	0	$0
12	0.25	$37.50	0.5	$25	0	$0
13	0	$0	0	$0	0	$0
14	0	$0	0.5	$25	0	$0
15	0	$0	0	$0	0	$0
16	0	$0	0	$0	0	$0
17	0.5	$75	0.5	$25	0.25	$22.50
18	0.5	$75	1.5	$75	0	$0

19	0	$0	0	$0	0	$0
Total	**0.75**	**$187.50**	**5**	**$250**	**0.25**	**$22.50**

Table 17. Cumulative Labor costs

People	**Total Hours**	**Total Cost**
Students ($20/hr)	1055.25	$21105.00
Faculty ($150/hr)	1.25	$187.50
Technician ($90/hr)	5	$250.00
Engineer ($50/hr)	0.25	$22.50
Total	**1061.75**	**$21565.00**

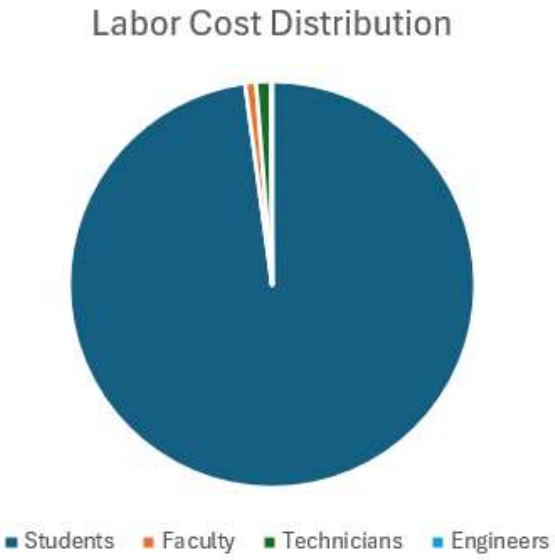

Figure 21. Pie Chart of Labor Distribution Cost

4.3. Major Costs Incurred and Future Mitigations

During our proposal phase, we estimated the development cost to be about $300. However, we have totaled around $436.38 in material costs for the project. However, most development costs, Figure 20, are because of shipping. We believe these costs are justified because we needed at least two weeks to assemble, debug, and verify the board before technical design work had to end. However, a longer development cycle would have allowed us not to require faster shipping.

4.4. Final Budget Reflection

For most of the project design cycle, we have kept additional costs to the university to a minimum. There were several tools that we did not anticipate needing at the start of our project and did not factor into our initial budget estimate. We did not account for the institution

providing licenses, but we included them in our budget. Lastly, we fortunately did anticipate some parts to fail during prototyping and adjusted our parts cost estimate, which was close to the original estimate.

5. RECOMMENDATIONS

The following recommendations may be used to outline potential avenues for enhancing the functionality of the syringe pump project. The project provides a well-designed hardware solution to the problem, paving the groundwork for software features such as data logging, Wi-Fi connectivity, and fault alarms. With additional effort, such software features may greatly enhance overall system performance and meet end-user needs to a greater degree. The current project can meet the feature set of hardware maintainability, affordability, and portability. The project, however, could be better suited to meet customization and ease of use due to the failure to implement a pulse counter for closed-loop control, which significantly improves motor accuracy.

The team did a great job of allocating tasks initially but underestimated the software roadblocks. As a result, involving all members in software from the beginning of the project would have improved the chances of debugging some of the roadblocks faced during the project, such as driver implementation and LVGL UI bugs.

5.1. Hardware Improvements

5.1.1. USB Power Delivery

During the development of the final PCB board, multiple roadblocks in the design were faced, especially regarding successfully implementing USB power delivery as a standard for the device. The current design allows for a 5V output via USB power but fails to detect the board when plugged in, making writing a software program impossible through USB. A lack of experience implementing USB-Power Delivery and poor online technical resources made the process challenging to accomplish within the timeframe. However, the board utilizes barrel jack power, which, in the context of a low-income setting, maybe more accessible, therefore meeting the minimum power requirements. Future teams should consider troubleshooting this feature as USB-PD is an up-and-coming standard and will continue to see increasing support.

5.1.2. Motor Input Power

The initial design of the final PCB utilized a buck-boost 5V-12V DC converter; however, failed to be successfully implemented as the regulator began to smoke when the device was powered on. While the stepper motor works with the 9V barrel jack DC power input, it is recommended to use a higher current capacity and verified buck-boost design for future implementations of the OSMI syringe pump to boost output power utilization. The team could not narrow down the specific point of failure for the 12V regulator, so more research must be done to analyze potential causes of failure and possible solutions. Due to constraints on shipping time, the team could not perform this.

5.1.3. Indicators and Test Points

The PCB produced for the final prototype lacked indicators and test points to help debug issues during assembly. To ensure that future replications in the hardware design are easy to debug, more indicators and test points should be implemented before manufacturing the PCB.

Furthermore, the current design requires that all components be placed on the board during one session. Reflowing components after baking the board makes the assembly process more difficult for the technician.

5.2. Software Improvements

Investing in software enhancements can present a unique opportunity to enhance system performance and facilitate ease of user interaction. The current software can use a timing-based open-loop control system to adjust the dosage and bolus rate of the syringe. Nevertheless, implementing a functional pulse counter for closed-loop control proved challenging. This reduced the software development to Wi-Fi connectivity, SD logging, and sound alerts. Regardless, the hardware has been established to support any software dedicated to these features.

5.2.1. Pulse Counter in Stepper Driver

The first improvement can be finishing adding the pulse counter. It can be used to trace the real-time amount of dosage being delivered. The code is already half-implemented but failed to pass the test. It would be nice to have the pulse counter because it is the essence of controlling the bolus rate. With bolus settings, the device will bump the patients’ blood concentration of the pharmaceutical to reach the effective level first. The current software can only use the time to track the dosage being delivered, which is not as accurate as the pulse counter because it does not provide direct feedback to the control system.

We found the pulse counter peripheral compatible with the PWM signal-generating LEDC peripheral during preliminary testing on the same output pin. However, this functionality would cease when fully integrated with the driver class. We have spent much development time attempting to diagnose the cause of the peripherals' failure to interact when combined but could not find a solution in time for winter completion.

5.2.2. More Drivers to Drive the Implemented Parts

Several hardware parts soldered on the device are not used, like the sound chip and the SD card. The drivers to make alarm sounds and log data while using our device can be implemented. The alarm sound can be used in the system self-on test, and the SD card logging is very helpful for tracing the device's usage history. The processor in the device, ESP32, also supports Wi-Fi communications. It can be used to design a cloud data logging function or a remote controller like a phone application or a website.

5.2.3. User Interface Improvements.

The user interface is currently in a minimum-viable-product state. However, the display's current refresh rate is below par for most user-facing applications and negatively impacts the system's usability. As the system stands, the current refresh rate is below 5 Hz, leading to a clunky and non-responsive experience for the user.

One way we believe we can improve the performance of the system is to take advantage of the ESP32's dual-core architecture. We initially planned to use the dual-core architecture at the start of the project but could not see it through due to time constraints.

Another way we believe we can improve the user's experience will be to limit the opportunities where the system will lag. One such situation is when the user is attempting to scroll. A solution for the scrolling issue is to break larger "screens" into several smaller menus where the user does not have to scroll.

Finally, the design of the user interface is as minimal as possible to attempt to keep design time fast. We have also not been able to finalize the system's design language, so while the simplicity of the interface is a benefit, we believe plenty of details could be added to make the system a complete product.

While we believe we have a viable product that meets our usability features, there is much to improve.

5.2.4. Wi-Fi / Dual-Core Monitoring

One of our earliest design decisions was the choice of the ESP32 as our microcontroller for its integrated 802.11b/g/n radio and dual-core architecture. However, we could not exploit these features due to the shortened development time of a Winter completion team.

We intended to develop a simple HTTP server to allow clinicians to monitor the pump's status securely remotely. HTTP would be preferred as it can be used easily across platforms like Android, iOS, or desktop.

This server should run on a core other than the leading system control core, as it is meant to act as a redundancy check for the rest of the system.

6. FURTHER MATERIAL

6.1. Standards Used

6.1.1. USB PD

The USB PD is a communication protocol that provides specifications for delivering higher power through USB connections [6]. We considered this when selecting our USB receptacle by ensuring it met the specified power and voltage ratings. Our voltage regulators also needed to be adjusted to support the input voltages from the USB PD. They referred to the power specifications in the USB PD so we could prevent damaging our board.

6.2. Impacts of Project on the Broader World

6.2.1. Public Health, Welfare, and Safety

In 2020, maternal mortality accounted for 287,000 deaths, almost 95% of which occurred in low and lower-middle-income countries due to various health complications [7]. The lack of access to reliable and affordable anesthesia equipment in low-income countries is a prominent factor in this high mortality rate. Therefore, because our syringe pump is designed to be cheaper and easy to repair on-site if needed, it makes this potentially life-preserving product more accessible to countries that need it. Suppose a component breaks while administering an emergency medication to a patient. In that case, restoring the device will not take as long because replacement parts can be manufactured quickly and locally. Making this equipment more available to developing nations can significantly reduce maternal mortality.

6.2.2. Global Issues

If a syringe pump component breaks, acquiring a new one while working in a lower-income region can be difficult and time-consuming as it will most likely need to be delivered internationally. International shipping is an arduous ordeal due to compliance issues, customs regulations, shipping costs, and delays. Lower-income countries, in particular, experience higher shipping costs because of longer transportation times and inefficient routes that make them challenging to access. By developing a syringe pump design that only requires off-the-shelf components to produce, the design is much easier and more efficient to repair when a component breaks down. Replacement parts for the syringe apparatus can be 3D-printed using standard 3D-printing supplies, and the electronic components that we used are common in maker communities and easy to source. This design decision reduces the need for international shipping for replacement parts. It allows for reparation to be confined locally wherever the operation takes place if a replacement component is needed.

6.2.3. Societal Issues

Making our syringe pump design open source enables the potential for developing and improving syringe pump designs in the future. By making the hardware and software for our

design freely available via an Open-Source license, we anticipate that medical research centers around the world will have ready access to the design, allowing educators, practitioners, and researchers in the medical field to freely download, print, and assemble their syringe pumps as needed. Additionally, educators can use the design for teaching purposes in a classroom setting when studying low-cost medical equipment design. This open collaboration can allow for continuous development of low-cost syringe pump designs as anyone can expand on and improve our current design.

6.2.4. Economic Issues

The prodigious price of infusion pumps is one of the most significant drawbacks of using them in resource-poor communities, with single-channel pumps costing up to $900 while multi-channel pumps range from $2000 to $4000. Our system is designed to utilize cheaper, accessible materials to reduce the cost of production, such as 3D-printed material, an ESP32 chip, and a PCB. Additionally, the ability to 3D-print a new replacement component for a broken part on-site will preserve shipping costs and the cost of ordering a new part altogether. However, developing low-cost syringe pumps made from cheaper, more accessible materials could become a more common practice for designing low-cost medical equipment in the industry. If a more affordable alternative to manufacturing becomes available, this could decrease business for major medical device manufacturers significantly if this design practice is expanded to all regions for its preservation of costs.

6.2.5. Environmental Issues

Using open-source software benefits the environment by conserving energy consumption and code storage. Because open source already provides users with available code to work from, it produces better code optimization and reduces the amount of system resources, such as RAM and CPU, required to run the software by suppressing the need for additional code development. Additionally, it reduces the carbon footprint because it minimizes the need for travel and transportation by enabling remote work and online collaboration.

6.3. System Failure Mode Analysis

Failure mode analysis is an essential facet of medical device engineering. Identifying where a system will fail can lead to a more resilient system design required for devices responsible for the safety of human lives, especially in a medical context.

This analysis will aim to find where the system is likely to jeopardize the safety of clinicians and prospective patients.

6.3.1. Analysis Methodology

For this project, we have conducted a failure modes effect analysis (FMEA) following Siemens PLM guidelines [8]. FMEA has two objectives: identify failure modes and describe the effects of these failures, aiming to mitigate them as profoundly as possible. This allowed us to determine where most of the team's work would be needed to make a safe design. Along with safety, we

can also decide what parts of the design will likely fail first and how to make it as easy to repair as possible.

6.3.2. Severity Ratings

The following table explains the relative severity scores for each system property. Table 18 correlates a score with a given meaning for the severity. Table 19 presents the likelihood scores, with the projected expectation of the event occurring. Finally, Table 20 illustrates the observability of a failure and how likely it is that testing will be able to detect the issue before delivering to the customers.

For each failure mode, we will assign a score from each of these three categories, multiplying them to determine the failure's overall risk factor. A lower risk factor is favorable to a higher risk factor.

Table 18. FMEA Severity Scores & Meaning

Score	Meaning
1	Very minor and can only be noticed by technicians.
2	Moderate, some usability may be degraded.
3	The device is non-functional but does not pose a threat to users.
4	The device is actively a hazard and can cause injury or death.

Table 19. FMEA Likelihood Scores & Meaning

Score	Meaning
1	Very unlikely to occur (< .01% chance)
2	Unlikely to occur (0.01% < Likelihood < 1%)
3	Possible to occur (1% < Likelihood < 10%)
4	Very likely to occur (10% to 40%)
5	Expected to occur, more than 40% likely

Table 20. FMEA Observability Scores and Meaning

Score	Meaning
1	Failure will certainly be caught during testing.
2	Failure is unlikely to be missed during testing
3	Failure is likely to be missed during testing.
4	Failure will pass, undetected, to the customer.

6.3.3. Failure Modes and Scores

In Table 21, we have a list of our expected failures with their respective scores. Overall, the most consequential failures would be running out of RAM or the LCD receiving an invalid frame

draw. In embedded systems, ensuring the system does not run out of memory is a key challenge because it is very possible in testing.

Table 21. Failure Modes with Respective Scores

ID	Failure Mode	Severity	Likelihood	Observability	Total
1	Microcontroller overloaded	2	3	2	12
2	Out of RAM	3	3	2	18
3	Out of program flash	1	4	1	4
4	Runaway program	4	1	3	12
5	Pulse counter peripheral glitch failure	4	1	1	4
6	LEDC peripheral setting failure	4	1	1	4
7	Stepper Motor Driver Failure	3	1	1	3
8	LCD receives invalid draw frame	2	2	4	16
9	RTOS Task Race Condition	3	2	2	12
10	The desired step frequency is out of range	3	4	1	12
11	Motor voltage does not meet 12V +- 10%	2	4	1	8

6.3.4. Failure Mode Mitigations

Table 22 explains the potential mitigations that could be used to prevent the failures in the above. These mitigations are recommended, but not all are implemented into the system. Mitigations in bold are included in the final system design.

Table 22. Failure Mode Mitigations

ID	Failure Mode Mitigation
1	A microcontroller overload is where the processor attempts to do too many blocking tasks at once, reducing the total functionality of the system. - **Ensure unused tasks are deleted when no longer used.** - Measure and record how long the idle task is used on average.
2	- **Ensure all dynamic memory allocation is called at the start of any given RTOS task.** - **Ensure the number of dynamic memory allocation calls is kept at a minimum, each with a 'free memory' call.**
3	- **Measure the program flash used after compilation.** - Add external memory to the system if compilation reports too much memory usage. - **Minimize image/font size in program memory**.
4	- Use watchdog timers that are only satisfied by a valid system state. - **Use general watchdog timers if state cannot be confirmed.**
5	- Enable the PCNT glitch filter, limited to the maximum possible frequency for moving the stepper motor times 10.
6	- Compare the expected pulse count to the actual pulse count. - **Add a physical limit switch to prevent carriage from attempting to overload.** - Enable stall detection on motor driver and stop when a stall is detected.
7	- **Check fault register on driver, and alert as necessary.** - **Confirm DRV8434S is on through SPI bus.**
8	- Connect the LCD to its own SPI bus for validating the frame buffer on the LCD.
9	- **Use FreeRTOS provided data types such as Mutexes, Semaphores, or Message Queues to safely send data between tasks and interrupt handlers.** - Use a memory safe programming language.
10	- **Limit the stepping frequency to a maximum frequency.** - Gradually change the step frequency instead of an impulse-like frequency change.
11	- Use an off-the-shelf power supply. - Use a confirmed power supply design for the V_{motor} supply. - Add fusing to prevent over-current events to protect the supply. - **Measure motor voltage on the output.**

7. APPENDIX A: USER MANUAL

User Manual for OSMI Syring Pump Project

Power

The device must be powered through the barrel jack connector. Do not utilize 5V USB PD for the device. Additionally, the device must not be left powered on for more than 7 days.

Graphics

The user UI has been divided into three primary screens OSMI home, Dosage configuration and System Status. A stylus is recommended when using the LCD display to switch between screens and interact with widgets.

The OSMI Home Screen has been configured to display the option to edit Dosage Settings. Furthermore, the option ResetSyringe at a slow or fast setting manually moves the syringe back to the original configuration. The option ForwardSyringe at a slow or fast setting manually moves the syringe forward at a chosen pace. The OSMI home screen can be found in the screen below:

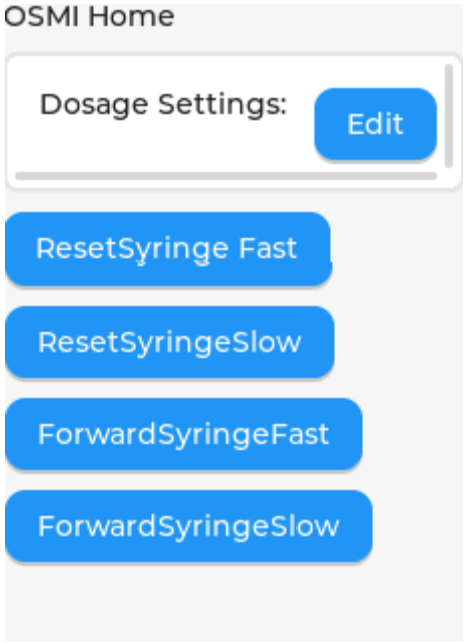

Figure 22. OSMI Home Screen.

The Dosage Settings Menu allows the user to custom configurate the Bolus rate and Dosage Rate at XX.X ml/min or ml/hr as well as the Bolus Volume and Dosage Volume. The Bolus rate allows for custom configuration of an initial delivery rate that can then switch to standard dosage rate. If a Bolus dosage is not to be selected its volume and rate can be left to its default settings of 0ml at 0ml/min. The corresponding Dosage Settings screen can be found below:

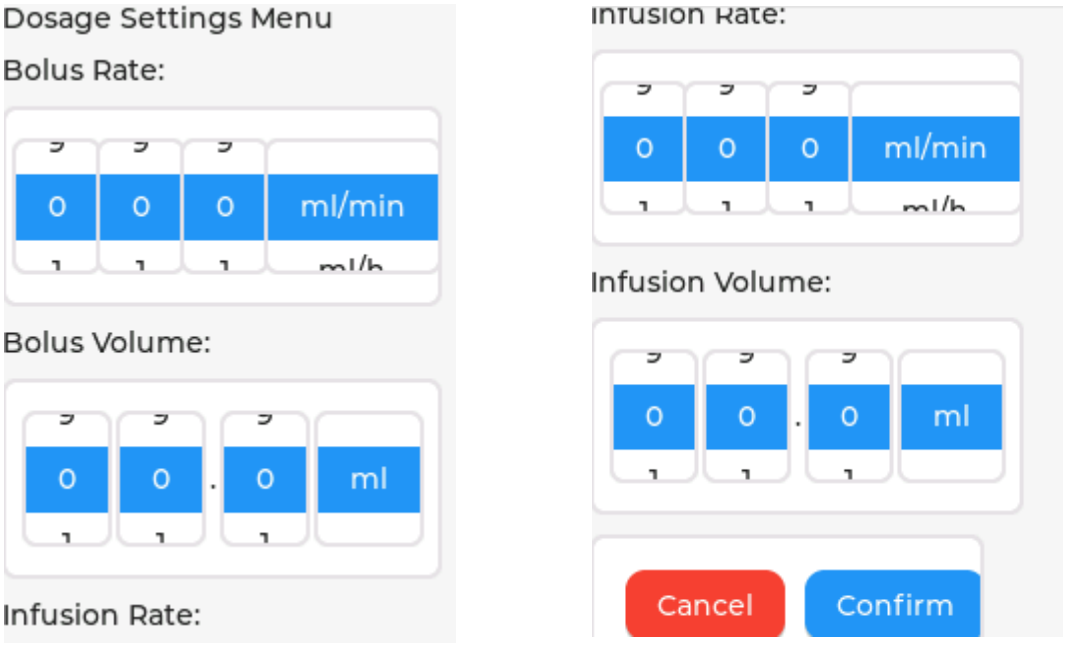

Figure 23. Dosage Settings Screen

The user may pause or start the delivery at this rate to return to the OSMI home menu or proceed with the delivery by pressing the cancel or confirm button. Should a standard dosage rate of 0 be set and delivery of the medication started an alert will be sent to the user to warn of "Error in configuration for delivery".

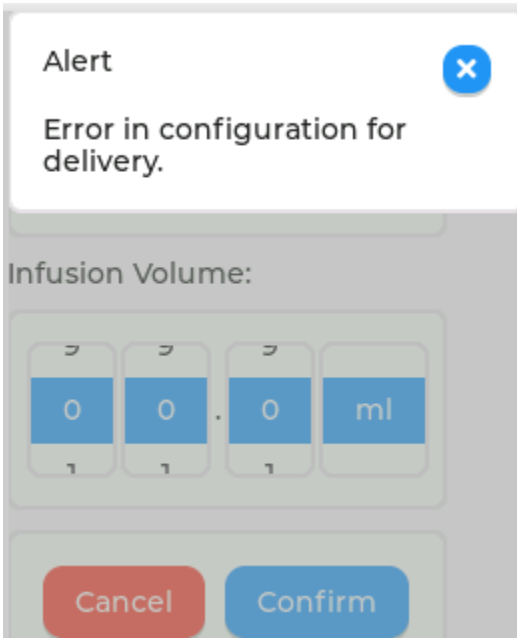

Figure 24. Configuration Error

Once delivery begins, the user will be moved to the System Status Screen, where the user may observe the current rate of infusion, time left as well as a visual indicator on the bolus and infusion volumes. A button at the end of the screen shall allow the user to pause delivery if sought fit.

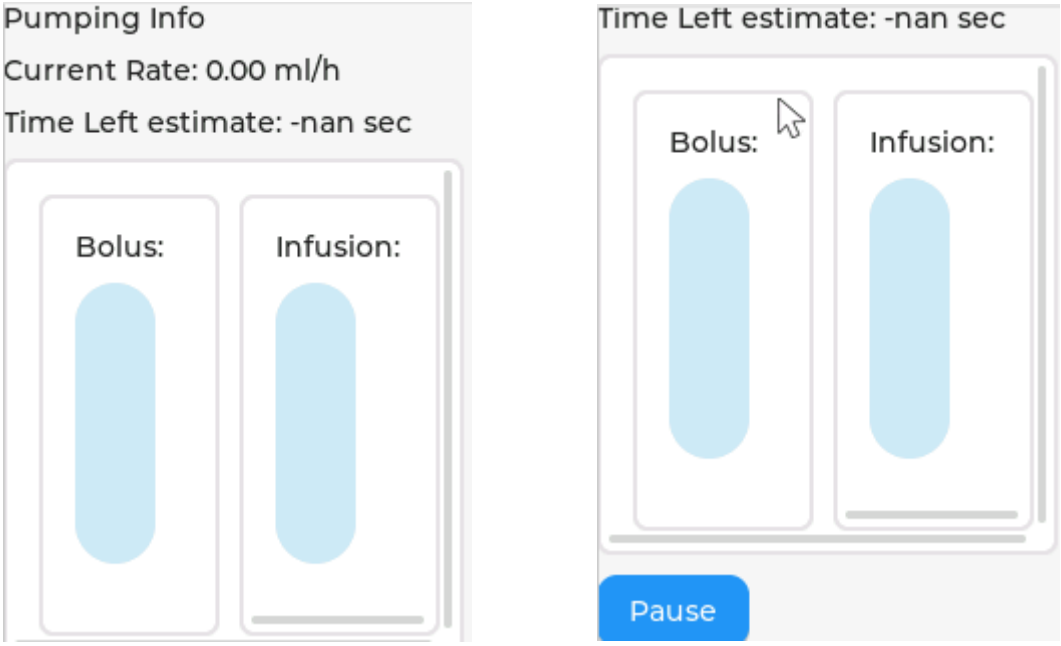

Figure 25. Dosage Status Screen

8. APPENDIX B: TECHNICIAN MANUAL

8.1. Introduction

This manual has two main goals. The first describes the system delivered to the Rose-Hulman Mechanical Engineering Capstone Team 31, and the second describes how to recreate the system for future work or peer review.

Part Name	Aliases
DRV8434S	Stepper motor driver, stepper driver.
ESP32	Microcontroller, MCU, ESP.
HiLetGo 2.4” TFT IO9341 LCD	LCD, Display, touchscreen.

8.2. General System Information

Two printed circuit boards are required to reproduce the system: the motherboard and the motor output board.

The motherboard is a four-layer printed circuit board that manages the power and control circuitry.

The daughterboard is a single-layer board with a mount for the stepper motor driver. Two capacitors regulate the Vmotor and Vcc for the stepper driver, ensuring the chip's regular operation.

Nine (9) connections need to be made between these two boards—three for power, three for channel control, and three for the serial peripheral interface bus. Please refer to section SCHEM for where each connection needs to be made. If you need to refer to which connections are which, you can look on the bottom side of the board, where each connection is named.

8.3. System-as-Delivered

All nine control connections have already been made as of delivery. However, the wires that were used are short, and we recommended cutting fresh, stranded wire to the desired length.

On the motherboard, four unpopulated components were not critical to the board's operation. First are two PMOS-FETs used for backflow voltage protection on the device; these were removed and shorted for debugging and were never reapplied. You do not need to reapply them.

Second, is the voltage regulator IC, the LM51551, which will immediately burn out if populated. As a result, do not attempt to populate this part.

For the daughter card, remember that the bottom of the daughter card is exposed copper, with most of it being a ground plane. Please ensure the daughter card is securely mounted somewhere the ground plane will not be short against.

Lastly, we recommend changing the outer pin sockets on the daughter card to terminal blocks. All pin spacing is 2.54mm or 0.1in apart. This is not required but would make the risk of wires popping out far less likely.

8.3.1. Program Flashing

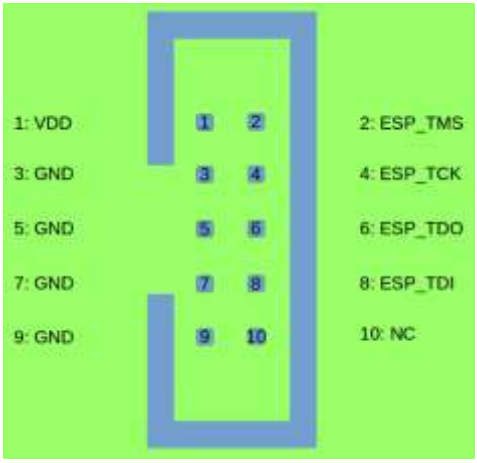

Figure 26. JTAG Pinout on Board

We did not have the correct JTAG pin header at the time of delivery, so you must solder jumper wires to the JTAG header on the motherboard. Figure 26 describes the pinout on the motherboard. The top of the board is the top of the image, and only one ground needs to be connected.

8.4. Recreating the System

8.4.1. Recommended Tools and Materials

Table 23 contains the tools that we used to build the board, along with some alternatives.

Table 23. Recommended Tools

Tool	**Purpose**
Soldering Iron	For soldering through hole components.
Reflow Oven, Hot Plate, or Hot Air Gun	For soldering surface mount (SMT/SMD) components.
Antistatic Tweezers	For placing surface mount components efficiently.
Solder Paste Stencil	For precisely applying the correct amount of solder paste for each SMT part. Ordered with the PCB blanks.
Solder Paste Spreader	Spreads solder paste evenly on the stencil to the board.

Wire Stripping Tool	Strips insulation from wires.

Table 24 contains all consumables, apart from the final bill of materials.

Table 24. Required Extra Materials

Material	**Purpose**
Solder	Feed solder for welding surface mount components.
18 Gauge Wire	For bridging unpopulated components.

8.4.2. Hardware Assembly

When assembling a PCB, you must start with the surface mount components. It is preferred to use anti-static tweezers to protect against damage from static electricity for sensitive components.

1. Prepare the PCB for surface mount assembly by aligning the PCB blank with the solder mask stencil on a flat surface.
2. Dispense an even line of solder paste on the solder mask, avoiding the cut-outs.
3. Using the solder paste spreader, spread the paste evenly. Repeat step 2 in different places until every spot in the stencil has an even layer of solder paste.
4. Delicately lift the solder mask as vertically as possible to avoid spreading the paste from where it's placed.
5. Begin populating the PCB. Start with the smallest components, then populate the larger components. **Do not populate components L1 and U5**. Use wire to bridge the pads across L1. Do not bridge anything else. Figure 28 provides a reference for where components can be placed.
    a. For extra assistance, we recommend visiting the "OSMI-FinalPCB" repository (URL can be found in section 10.1) and follow the link visible in the about section of the repository (Figure 27).

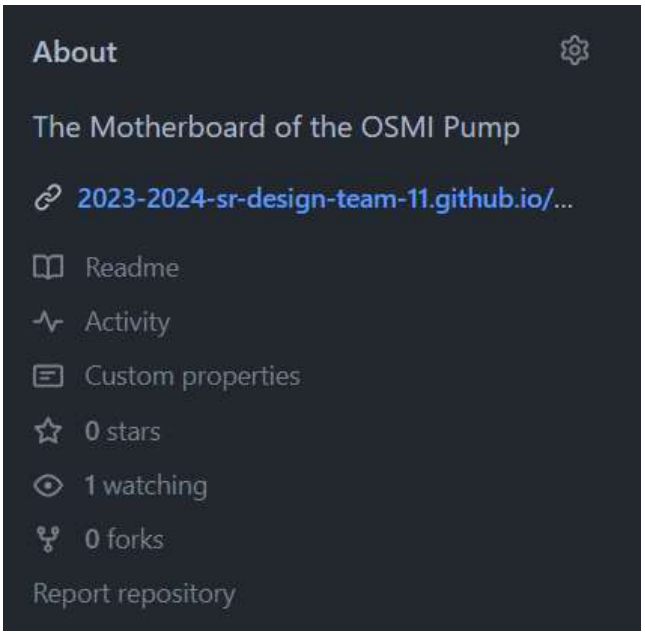

Figure 27. About section in the “OSMI-FinalPCB” Respository.

This is an interactive web page allows you to check off components as you can source, and place the components.

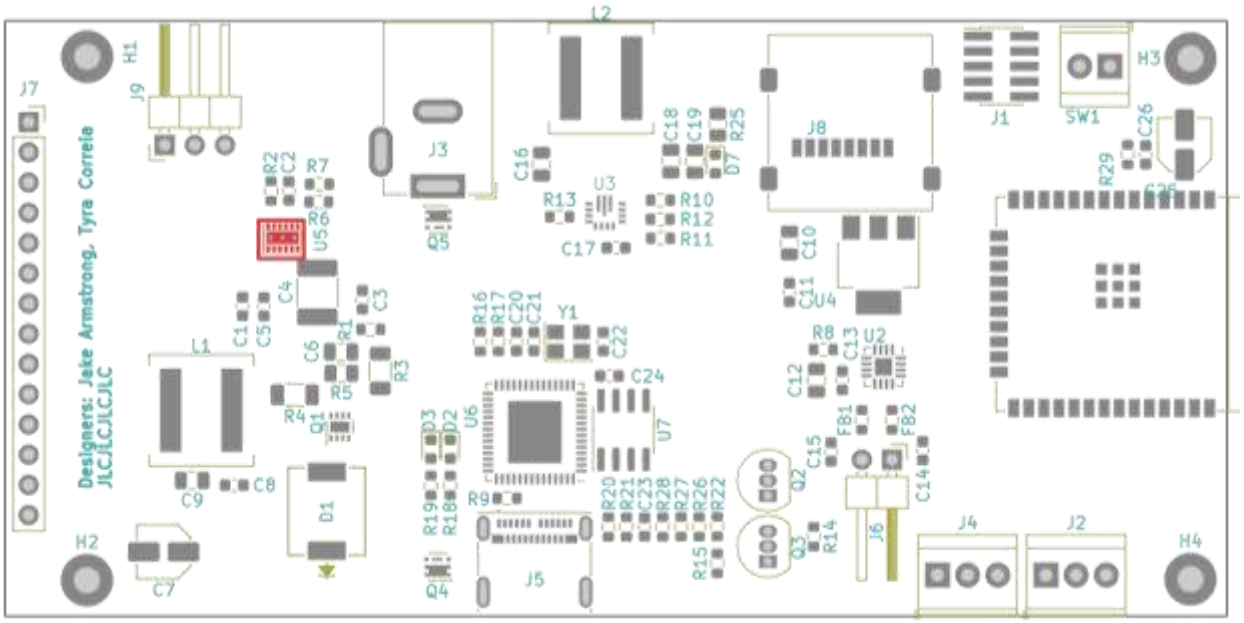

Figure 28. Top Layer Reference.

6. Once every sufrace mount component is populated, reflow the board.
    a. If using a reflow oven, do your best to follow the temperature curve used on the solder paste.

b. If using a hotplate, set the hot plate to the peak melting temperature on the curve.

c. If using a hand-reflow tool, use the peak melting temperature on the curve. Try to heat the area around the board as evenly as possible to avoid damage.

7. Solder the through-hole components, which includes pin sockets, pin headers, terminal blocks, and the barrel jack.
8. Before moving on, perform a continuity check on all connections, ensuring that all intended connections are made and there are no unintended connections.

    a. If there is an unintended short, de-solder a component and test again. Repeat until the connection is removed.

    b. Q4 and Q5 can be removed and shorted if there power in is not functioning.

8.4.3. Connecting the boards together.

Figure 29 shows, in blue text, what is printed on the silkscreen of the PCB and Figure 30 has pin number references. To connect to the Stepper Carrier, follow the instructions in TABLE in order.

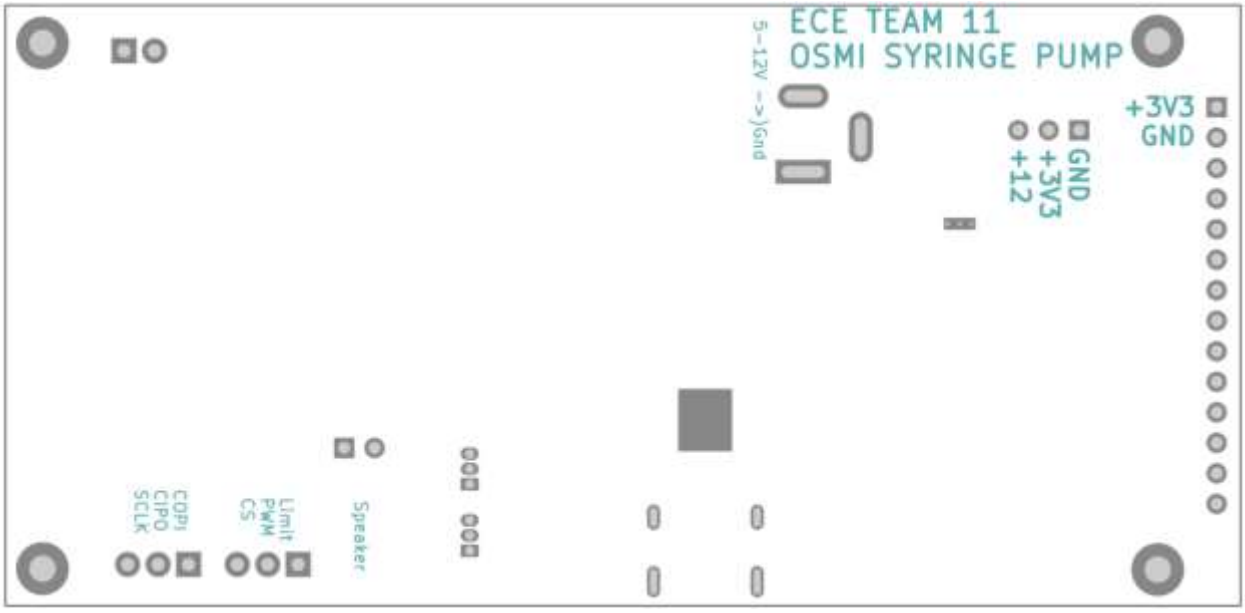

Figure 29. Back of the Syringe Pump Motherboard.

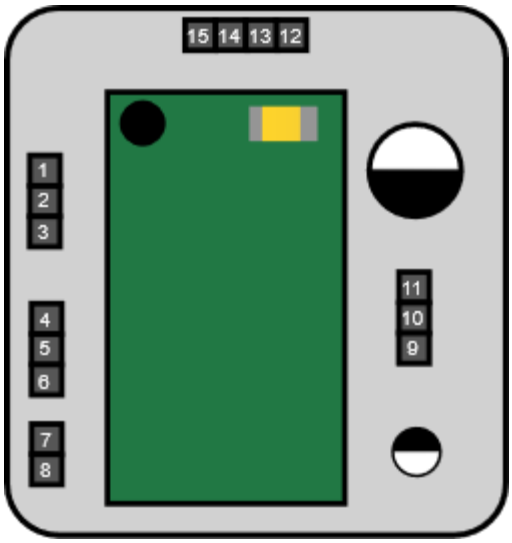

Figure 30. Stepper Motor Carrier Board

Motherboard	**Stepper Board**
SCLK, CIPO, COPI	Pin 1, Pin 2, Pin 3
CS, PWM, Limit	Pin 4, Pin 5, Pin 6
N/A	Pin 7, Pin 8
GND, +3V3, +12V	Pin 9, Pin 10, Pin 11
N/A	Pin 12, Pin 13, Pin 14, Pin 15

Pins 7 and 8 are for the end stop limit switch. Place that switch where the syringe is fully depressed.

8.4.4. Program Flashing

The confirmed best way to flash the firmware to the device is through the JTAG adapter. We recommend using an ESP-Prog, which can be bought from any electronics distributor.

1. Install WinUSB drivers by using Zadig.
    a. The installer can be found at https://zadig.akeo.ie/
2. Connect the provided JTAG adapter with the provided 10 pin adapter cable.
3. Using the delivered project, use the “Upload” button to flash the board.

9. Appendix C: Licenses Used

9.1. CC BY-SA 3.0

Under requirement of using material licensed under the CC BY-SA 3.0 license.

https://creativecommons.org/licenses/by-sa/3.0/

9.2. MIT License

Required for using LVGL, an open source and free to use software library.

9.2.1. MIT License Text

MIT licence

Copyright (c) 2021 LVGL

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

10. APPENDIX D: PROJECT RESOURCES

10.1. Project GitHub Links

Repository Name	Github URL
OSMI-Software	https://github.com/2023-2024-Sr-Design-Team-11/OSMI-Software
OSMI-FinalPCB	https://github.com/2023-2024-Sr-Design-Team-11/OSMI-FinalPcb
OSMI-StepperPCB	https://github.com/2023-2024-Sr-Design-Team-11/OSMI-stepperPCB

10.2. PCB Bill of Materials

Table 25 covers all the parts for the completely assembled PCB. However, please refer to Appendix B: Technician Manual for which parts are not required.

Table 25. Fully Assembled PCB Bill of Materials.

Reference	Value	digikey_pn	Qty
C1, C2, C17	100nF	1276-CL10E104KC8VPNCDKR-ND	3
C3	6.8nF	399-C0603C682K5TACAUTODKR-ND	1
C4	15nF	13-CC1812KKX7RBBB153DKR-ND	1
C5, C8	1uF	1276-1102-6-ND	2
C6	1nF	399-16643-6-ND	1
C7, C25	22uF	493-9831-6-ND	2
C9, C10, C12, C16, C18	10uF	1276-1096-6-ND	5
C11	22uF	1276-7076-1-ND	1
C13, C24, C26	0.1uF	1276-1935-1-ND	3
C14, C15	220pF	1276-1048-1-ND	2
C19	47uF	1276-1852-1-ND	1
C20	3.3u	445-7476-1-ND	1
C21, C22	18pf	1292-1494-1-ND	2
C23	33nF	1276-2041-1-ND	1
D1	D_Schottky	SL42-E3/57TGITR-ND	1
D2	BLUE	160-1827-6-ND	1
D3, D7	RED	160-1447-6-ND	2
FB1, FB2	120 ohm	445-181518-6-ND	2
H1-H4	MountingHole		4
J1	JTAG	1175-1735-ND	1
J2, J4	3Term	A98334-ND	2

J3	Barrel_Jack	2073-DCJ200-10-A-K1-K-ND	1
J5	USB_C_Receptacle	WM12856CT-ND	1
J6	Speaker_Out	2057-PH1RB-05-UA-ND	1
J7	Display Socket	S7012-ND	1
J8	Micro_SD_Card	WM6698CT-ND	1
J9	Power Out		1
L1, L2	2.2uH	2457-XAL8080-222MED-ND	2
Q1	NMOS	296-37193-6-ND	1
Q2, Q3	S8050	4878-BC547BBK-ND	2
Q4, Q5	PMOS	SIA461DJ-T1-GE3DKR-ND	2
R1	3.3k	13-RC0603JR-133K3LCT-ND	1
R2	21k	RMCF0603FT21K0CT-ND	1
R3	12m	P0.012BVCT-ND	1
R4	487	2019-RK73H2BTTD4870FCT-ND	1
R5, R25	100	311-100CRCT-ND	2
R6	2.49k	P2.49KDBCT-ND	1
R7	27.4k	RMCF0603FT27K4CT-ND	1
R8	370k	2019-RN73H1JTTD3703F100CT-ND	1
R9	12k 1%	RMCF0603FT12K0CT-ND	1
R10	523k	RMCF0603FT523KCT-ND	1
R11, R12	100k	RMCF0603FT100KCT-ND	2
R13-R15, R26-R29	10k 1%	RMCF0603FT10K0CT-ND	7
R16, R17	1k	RMCF0603FT1K00CT-ND	2
R18	25	13-RP0603BRD0725RLCT-ND	1
R19	68	1292-WR06X680JTLCT-ND	1
R20	100k 1%	RMCF0603FT100KCT-ND	1
R21	15k 1%	P15KDBCT-ND	1
R22	2k	P2.00KHCT-ND	1
SW1	SW_Push	A98333-ND	1
U1	ESP32-WROOM-E	1965-ESP32-WROOM-32E-N8CT-ND	1
U2	MAX98357	MAX98357AETE+TCT-ND	1

U3	TPS63070RNMR	296-44874-1-ND	1
U4	LDI1117-3.3H	4878-LDI1117-3.3HCT-ND	1
U5	LM51551DSSR	296-LM51551DSSRCT-ND	1
U6	FT232HPQ	768-FT232HPQ-TRAY-ND	1
U7	93LCxxB	93LC66B-I/SN-ND	1
Y1	12Mhz	3155-OSC12M-3.3I/S3TCT-ND	1

11. REFERENCES

[1] D. E. Wamble, M. Ciarametaro and R. Dubois, "The Effect of Medical Technology Innovations on Patient Outcomes, 1990-2015: Results of a Physician Survey," *Journal of Managed Care & Specialty Pharmacy,* vol. 25, no. 1, 2018.

[2] M. Magome, "'What can we do?': Millions in African countries need power," AP News, 25 March 2023. [Online]. Available: https://apnews.com/article/electricity-africa-just-energy-transition-d20d1ba86e90c3b9c81f0fc76979acfc. [Accessed September 2023].

[3] M. K. S. E. M. E. R.-K. R. D. Q. O. Z. Juarez A, "AutoSyP: A Low-Cost, Low-Power Syringe Pump for Use in Low-Resource Settings.," *The American Journal of Tropical Medicine and Hygiene,* vol. 95, no. 4, pp. 964-969, 2016.

[4] LVGL, "Light and Versaitle Graphics Library," [Online]. Available: https://lvgl.io/. [Accessed 1 September 2023].

[5] Espressif Systems, "Introduction to the ESP-Prog Board," [Online]. Available: https://docs.espressif.com/projects/espressif-esp-iot-solution/en/latest/hw-reference/ESP-Prog_guide.html. [Accessed 8 January 2024].

[6] USB Implementers Forum, 31 10 2023. [Online]. Available: https://www.usb.org/document-library/usb-power-delivery.

[7] World Health Organization, "Maternal mortality," 23 February 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/maternal-mortality.

[8] Siemens PLM Software, "How to conduct a failure modes effect analysis (FMEA)," Plano, 2016.