Group 9
Yoga Timer
Project Document

Craig Brod, Bryan Gaona Herrera, Rory Patterson

Table of Contents

Group 9 Yoga Timer Project Document
Craig Brod, Bryan Gaona Herrera, Rory Patterson

1 Overview

1.1 Executive Summary

The purpose of our project is to create a wearable device that can either be clipped to a user's clothing or placed on the ground and helps guide a user's yoga session with limited distractions. This will be accomplished by using a capacitive touch screen and software that will allow the user to set timers based either on the amount of sets they want to do vs the amount of time they have or on the amount of sets vs the amount of time the user wants for each set. The device will use a clip-like stand in order to accomplish the two use scenarios. The user will be able to interact with a simple UI consisting of button prompts on the screen that allows them to customize their session. A user will also be able to use the device for a few sessions before needing to recharge. Many of the components will be decided to limit both current draw and size.

1.2 Team Contacts and Protocols

1.2.1 Team Contacts

Table 1.1-Group Contact Information

Group Member Name	**E-Mail**
Craig Brod	brodc@oregonstate.edu
Bryan Goana Herrera	gaonaheb@oregonstate.edu
Rory Patterson	patterro@oregonstate.edu

1.2.2 Team Protocols

Table 1.2-Team Protocols

Protocol	**Expectations**
Protocol name here	How the protocol is expected to be carried out
Weekly meeting	Meetings will be held in Dearborn 211 at 2pm for the fall term. This will be subject to change as our schedules change in the coming terms. Meetings can be rescheduled or canceled if reasonable.
Task Delegation	All individual tasks will be discussed and

	agreed upon as a team before being delighted to one person.
Documentation	All documentation must be present in google drive before discussion.
Discord Communication	Discord channel will be used for general communication regarding meetings, assignment questions, and updates. Files should not be sent here and instead uploaded to the Google drive.
Missed Meetings	If someone misses a meeting they are responsible for reading the meeting notes and asking questions in the Discord as needed.

1.3 Gap Analysis

Although there are many yoga and mindfulness timers available currently, the market lacks a timer that is able to fit the life of a busy person who seeks zero distractions during their session. Many yoga timers offer limited customization with building a yoga session, and often end users are required to break their immersion by resetting the timer. This project will seek to offer end users a yoga experience with limited distractions, allowing the user to focus solely on their session. With a complementing slim design, this will also allow users to take a break wherever they are in their daily routine. The project will also allow users to develop a complex yoga session without the need to set multiple timers. Many market products may offer some of these features, but they often sacrifice something else like size or complexity.[1] This project will provide the full experience instead.

1.4 Timeline

Table 1.3-Proposed Timeline

Task	Fall					Winter					Spring				
Documentation															
System Block Diagram															
Individual Block Design															
Part Research															
PCB Design															
Enclosure Design															
Design Impact Assessment															
Block Testing															
System Integration															

1.5 Reference and File Links

1.5.1 References

[1] A. Dehnke, “The best training timers,” *Yoga Journal*, 10-Nov-2021. [Online]. Available: https://www.yogajournal.com/osp/the-best-training-timers/. [Accessed: 18-Nov-2022].

1.5.2 File Links

1.6 Revision Table

Table 1.4-Revision Table

10/14/22 Craig, Bryan, Rory	Created section 1 draft
10/31/22 Craig	Moved revision table to its own section and updated headings
11/17/22 Craig	Combined sections 1 and 2
11/18/22 Craig, Bryan, Rory	Updated team contact layout, references and file links page layout, gap analysis, executive summary, timeline, team protocols, and IEEE formatting

2 Impacts and Risks

2.1 Design Impact Statement

Our project, a small wearable device designed to benefit those who need a way to fit yoga and mindfulness into their busy days, will have some public health impacts, cultural and social impacts, environmental impacts, and economic impacts. The purpose of this state is to inform possible producers of this product of possible liabilities of the product and its impact.

Our product could contribute to the negative impacts of practicing yoga on the mental health of young adults. With easier accessibility as a result of our project, more young adults could fit time in their schedules to practice yoga. This could result in “comparative and inner critique during yoga [which] may be harmful to body image.” [3] This is a struggle that is rooted in yoga and many other factors, which makes it difficult to account for in our project design.

Yoga is a practice with deep roots in the culture of India. With the introduction of yoga to the West, it has evolved into something completely different from its cultural context. According to yoga teacher Pranidhi Varshney, yoga practice today looks more like “a commodity that must be consumed and perfected through books, training, retreats, and endless online discussion,” as opposed to “the way Indians do—as a lighthearted part of everyday life.” [2] The cultural appropriation of yoga is something that must be acknowledged with our project, since our project aims to make practicing yoga more efficient and accessible. This issue is very complex and systematic, which makes it difficult for our team to tackle our project. As such, a good place to start is to acknowledge the impact of our project on the cultural appropriation of yoga and how it feeds into commoditization of the practice if our project were to ever go into a larger scale of production.

Lithium is a difficult metal to mine. The most cost effective way to do it is to pump brine into certain salt flats and wait for various metals to seep out of the ground in the slurry and dry in concentrations worth processing. It takes 500,000 gallons of water to process 1 metric ton of lithium according to the Institute for Energy research [4]. The same places that have these metal rich salt flats are also usually arid, making lithium mining consume the majority of water in certain areas. It would be well worth the effort to look into finding a replacement for the lithium polymer battery that we use in our design.

Yoga has evolved into a large market in the present day. With yoga studios, brands, and influencers pushing yoga more into a capitalist experience, it becomes quite contrasted to its root ideas of harmony and self-actualization. Bringing our product to the ;arger market could only further create “inherent paradoxes of yoga as a conscious luxury experience and yoga as a traditional practice.” [1] A possible solution to this however, could be to ensure that our product is as low-cost as possible. This could mitigate the expansion of capitalism ideas in yoga and help our consumers not worry about the price of doing yoga.

2.2 Risks

Table 2.1-Risk Table

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Performance Indicator	Action Plan
1	Team member needs to take time for personal matters unexpectedly	Organizational	Med	Med	Team member sends short notice of absence	-Gather time-sensitive materials team member had -List work team member had to complete and plan to delay/adjust deadlines -Redistribute work for expected time of absence. If it is unreasonable, contact course instructor
2	LiPo battery catches fire	Safety	Low	High	Fire	-If outside, safely kick fire to area away from burnable materials -Locate fire extinguisher and call 911 -Use extinguisher to

						prevent surround materials from igniting
3	Fry components with power supply	Technical / Safety	Low	Med	Magic smoke	-Order new parts -Rework PSU design
4	Team member dropping class	Organizational	Low	High	Team member no longer in class	-Contact course instructor -Reassess current workload
5	Parts unavailable	Organizational / Technical	Med	Med	Parts out of stock	-Source new parts -Change design if needed
6	Exceeds budget	Organizational	Low	High	Spent over $300	-Create BOM with item importance -Contact Ingrid
7	Project is stolen	Technical / Organizational / safety	Low	High	Project is missing	-Contact Instructors -Reassess budget -Reorder parts if needed

8	Design does not work	Technical	High	Low/Med	Project not working as intended	-Analyze what's gone wrong -Re-design faulty component -Implement new design

2.3 References and File Links

2.3.1 Reference

[1] D. Neumark-Sztainer, A. W. Watts, and S. Rydell, “Yoga and body image: How do young adults practicing yoga describe its impact on their body image?,” *ScienceDirect*, 2018. [Online]. Available: https://www-sciencedirect-com.oregonstate.idm.oclc.org/science/article/pii/S1740144518301177. [Accessed: 04-Nov-2022].

[2] P. Varshney, “My take on cultural appropriation and yoga,” *Yoga Journal*, 12-Aug-2022. [Online]. Available: https://www.yogajournal.com/yoga-101/cultural-appropriation-yoga/. [Accessed: 04-Nov-2022].

[3] Institute for Energy Research. (2020, ⁹November 12). *The environmental impact of lithium batteries*. IER. [Online]. Available: https://www.instituteforenergyresearch.org/renewable/the-environmental-impact-of-lithium-batteries/ [Accessed:12-Nov-2022]

[4] J. L. Mora, J. Berry, and P. Salen, “The Yoga Industry: A conscious luxury experience in the transformation economy,” Luxury, vol. 5, no. 2, pp. 173–196, 2018.

2.3.2 File Links

2.4 Revision Table

Table 2.2-Revision Table

10/31/22 Craig, Bryan, Rory	Created section 2 draft
11/17/22 Craig	Combined section 2 with section 1
11/18/22 Craig, Rory	Made minor changes to match with rubric and added to risks table
4/28/2023 Bryan	Added Design Impact State and references.
5/11/2023 Bryan	Indented references

Group 9 Yoga Timer Project Document
Craig Brod, Bryan Gaona Herrera, Rory Patterson

3 Top-Level Architecture

3.1 Block Diagram

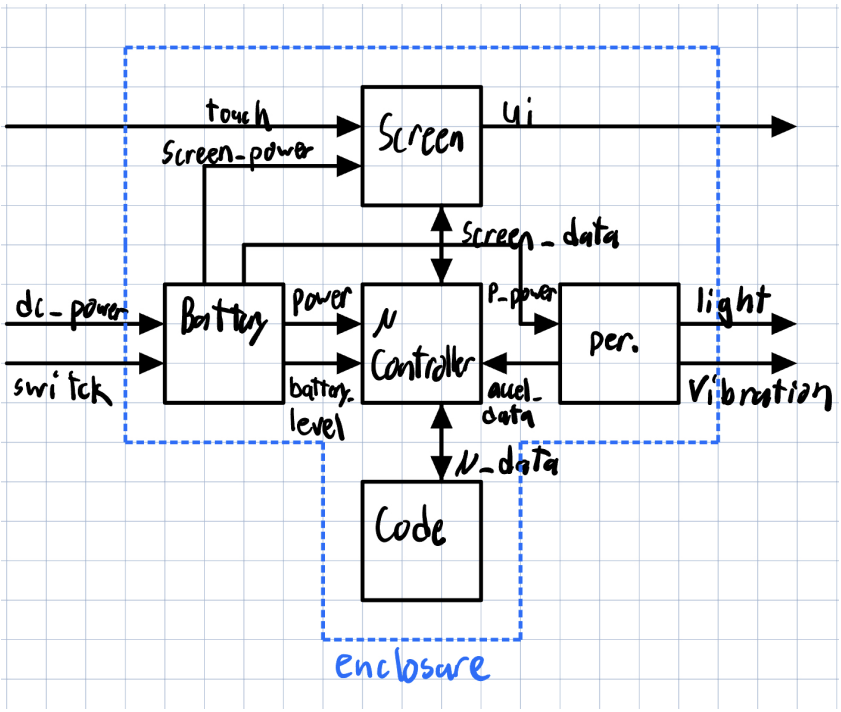


Fig. 1: Block Diagram

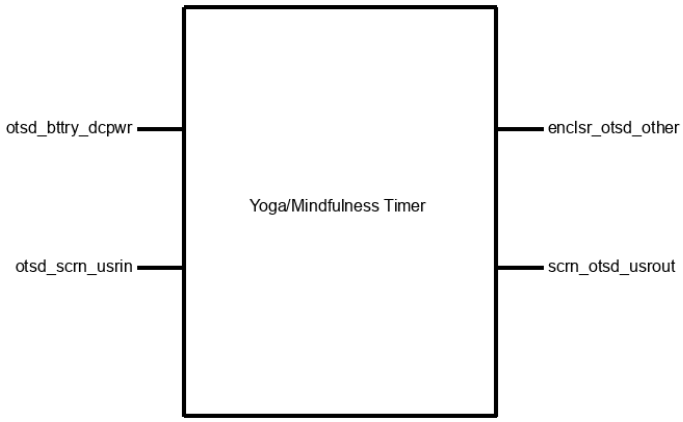


Fig. 2: Black Box Diagram

3.2 Block Descriptions

This table lists the description of each block in the system, their contributions to the system, and the project team member in charge of each block.

Name	Description
Enclosure Champion: Rory Patterson	The Enclosure will contain all the components and provide access to the inputs. It will also feature a belt-clip, to allow the whole device to be worn.
Code Champion: Craig Brod	The code block will contain a UI for the user to interact and control the unit's timers to their needs. Furthermore, this block will control what the microcontroller will output to the screen and other peripherals as well as read in values from some peripherals and the battery.
Microcontroller Champion: Rory Patterson	The microcontroller is designed to connect the code to the other elements of the project. It allows the code to communicate with the screen and the touch panel to communicate back; using an I2C connection to update the screen and an SPI connection to receive touch data. It uses a series of digital I/O to interact with the peripherals block. And it uses an ADC to provide information about the battery to the controller. Additionally, the microcontroller's clock is how the whole project keeps track of time.
Peripherals Champion: Bryan Gaona Herrera	The peripherals block will include components that output light and vibrations as well as the necessary circuitry to switch those on and off using the microcontroller.
Battery Champion: Bryan Gaona Herrera	The battery block will contain a battery cell connected to a student designed charge regulation module. A DC-DC converter will connect the battery output to the rest of the system.
Screen Champion: Craig Brod	The screen will receive power from the battery unit, receive UI display data from the microcontroller, receive input data from the user, relay the user's input to the microcontroller, and output a UI for the user.

3.3 Interface Definitions

This table lists every interface and associated properties for each of the blocks in the system.

Name	Properties
otsd_bttry_dcpwr	• **Inominal:** 500 mA • **Ipeak:** 550 mA • **Vmax:** 6 V

	• **Vmin:** 4 V
otsd_scrn_usrin	• **Timing:** Take 2 inputs per second and user input should take less than a second • **Type:** Single-touch based with fingers • **Usability:** Input should be understandable by at least 9/10 users
enclsr_otsd_other	• **Other:** 3.2x2 mm opening for LED • **Other:** Belt clip that will clip onto surfaces that are 1/4 inch • **Other:** 6.9x1.85mm opening for micro USB port • **Other:** Max outer dimensions: 2.5x3.5x1 in • **Other:** Max Internal dimensions: 2.4x3.4x0.9 in
mcrcntrllr_cd_data	• **Datarate:** frequency of 80MHz for SPI connection and 100Kbit/s for the I2C connection • **Messages:** Screen communication, battery level, peripheral values • **Protocol:** Configures SPI for sending data to the screen, I2C for receiving data from the screen and accelerometer, an analog pin for battery level, one digital pin for input, and three digital pins for output.
mcrcntrllr_prphrls_comm	• **Datarate:** 100 Kbit/s • **Messages:** Accelerometer data • **Protocol:** I2C
mcrcntrllr_scrn_data	• **Datarate:** frequency of 80MHz for SPI connection and 100Kbit/s for the I2C connection • **Messages:** Location data of user input will be received and UI elements will be sent to the screen • **Protocol:** Uses SPI to receive data from microcontroller and I2C for sending data to the microcontroller

prphrls_mcrcntrllr_dsig	• **Logic-Level:** Active high • **Vmax:** 3.6 V • **Vmin:** 0 V
bttry_mcrcntrllr_asig	• **Other:** Sampling Rate: 100 kSPS • **Vmax:** 0.750 V • **Vrange:** 0.0V - 0.750V
bttry_mcrcntrllr_dcpwr	• **Inominal:** 30mA • **Ipeak:** 500mA • **Vmax:** 3.6V • **Vmin:** 3V
bttry_prphrls_dcpwr	• **Inominal:** 20mA • **Ipeak:** 80mA • **Vmax:** 3.5V • **Vmin:** 3.0V
bttry_scrn_dcpwr	• **Inominal:** 60 mA • **Ipeak:** 80 mA • **Vmax:** 3.5 V • **Vmin:** 2.9 V
scrn_otsd_usrout	• **Other:** The backlight will be visible to at least 4 meters away to 9 out of 10 users. • **Type:** Will show UI elements to the user in the form of lit up pixels on a screen • **Usability:** 9 out of 10 users will say it is readable from 1 meter away.

3.4 References and File Links

3.4.1 References

3.4.2 File Links

ESP32-C3 documentation Get Started - ESP32-C3 - — ESP-IDF Programming Guide latest documentation

3.5 Revision Table

3/10/2023 - Craig, Bryan	Section created and subsections populated from online tool
4/28/2023 - Craig	Added a file link
5/11/2023 - Bryan	Updated interfaces to match online tool. Added more detail to battery block.

4 Block Validations

4.1 Screen Block

4.1.1 Description

This block is composed of a single capacitive touch screen module that will send data to and receive data from the user and the microcontroller. The screen itself will track user input via the touch screen and relay this information to the microcontroller so that the user can control the UI. For a majority of the time, the screen will use its backlight to be illuminated for the user, but the backlight will be turned off when inactive. From the microcontroller, the screen will get information about what it should display. Furthermore, the screen will be taking in power from the battery management block.

4.1.2 Design

This block will consist of the screen module “2.8" TFT Display - 240x320 with Capacitive Touchscreen” from Adafruit. In this module there is a backlit LCD display portion that communicates over SPI, and a capacitive touch layer that communicates with I2C [1].

As the name implies, the whole module is 2.8 inches diagonally with 2.7 inches lengthwise and 2 inches widthwise [1]. This display has a backlight that can be set to on or off and can display an array of colors [1]. The display portion specifically can communicate using SPI, 8-bit serial, and 16-bit serial, but for our purposes we are sticking to SPI [2].

The digitizer portion of the display is a capacitive touch layer, meaning that it uses capacitance to measure where a user touches the screen. This type of layer is also commonly used on many smart devices for its accuracy [3]. How capacitive touch accomplishes this is by using the human body’s natural ability to absorb electrostatic charge [3].

This block connects to other blocks and the outside world with a number of interfaces. It connects to the outside world for both user input via touch and user output via the screen’s pixels. The power block supplies the screen with voltage to keep its backlight on. On top of this, the microcontroller tells the screen what to display and the screen sends the user input to the microcontroller for processing.

The screen will physically connect to other components in the system via a ribbon cable. This ribbon cable will connect to a 50-pin quick connect slot, also found on Adafruit’s shop [4]. With this connector, assembly and screen module

swapping will be much simpler. They also offer a breakout board for simplified testing by turning the ribbon cable conn

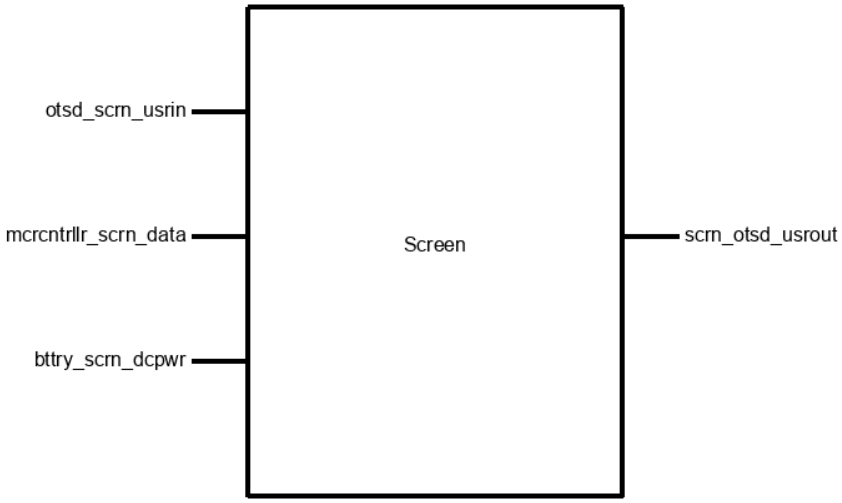


Fig. 1. Black Box Diagram

1. **otsd_scrn_usrin:** This interface represents the user's input into the screen. The user will be able to give input simply by touching the screen with their finger. While a select few other objects will work, a finger will be the most consistent [3].
2. **mcrcntrllr_scrn_data:** This interface represents the data that is being received from the microcontroller and the data that the screen is sending back. The data is being sent to the screen via SPI, while the data from the screen will be sent to the microcontroller via I2C. The reason this is one interface instead of two is because splitting the two would overcomplicate how the two blocks converse [2]. For the purposes of this block, this interface is bidirectional.
3. **bttry_scrn_dcpwr:** This interface is where the battery will be supplying power for the screen in order for it to operate. The screen will normally use 3.3 volts at 60 milliamps, but these values can differ depending on the situation. According to the datasheet, the screen can still operate with voltages ranging from 2.9 volts to 3.5 volts [2].
4. **scrn_otsd_usrout:** This interface where the screen will turn on its backlight and change its pixel colors to show a user text or a timer. Furthermore, the backlight will change its brightness based on whether the user is currently using it or not. When the system is turned off, the screen will not output anything.

4.1.3 General Validation

Some of the design for this block comes based off of the project partner's requirements. The key requirements that influence this block are that the system needs to be small and that the system must use a capacitive touch screen. I chose this particular screen because of not only its size, being that it is only 2.7

inches by 2 inches, and its touch capabilities, but also for a couple other features that allow it to be easily tested and for its ease of integration.

This screen is actually a combination of two parts. The first part is an LCD color display. I wanted the display to be full color so that I could use color in the user interface. This is important as a simple black and white display can be hard on the eyes. I also wanted the display to be a dot matrix and not a segmented display so that we were not limited to simple text. The second part of the screen is the capacitive touch layer. Even if it were not a requirement, we would still have gone with this touch panel due to its precision and the fact that it is hard to influence with anything but a finger [3].

Additionally, the screen can use an SPI to receive display information and I2C to send touch information. This is great for us because it means that we do not have to worry about timing issues for communication. Both SPI and I2C are easy to implement as most, if not all, accessible microcontrollers have support for them. They are also well documented such as these [5], [6].

Furthermore, I was looking for a screen that could be efficiently tested. This particular screen has a breakout board and ribbon cable connections that will make testing and integrating as simple as unhooking a latch [4], [7]. This also means that if we somehow break the screen, replacing it does not require any unsoldering and resoldering.

The screen is also accompanied by an Arduino library from Adafruit with examples, allowing us to have plenty of documentation, like this [8], to accurately use it.

4.1.4 Interface Validation

otsd_scrn_usrin : Input

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
Timing: Take 2 inputs per second and user input should take less than a second	I do not expect the user to give more than 2 inputs per second and that they will remove their finger in a short period of time. This is based on how many interact with smart devices.	The screen I have chosen is capable of handling much more than 2 inputs per second due to the timings listed on the datasheet and the speed of I2C [2], [5].

Type: Single-touch based with fingers	I do not expect a user to use more than a single input at a time. Furthermore, they will only be using their fingers to give the input.	Listed in the datasheet in section 2 it says that the touch input is a single-point capacitive touch panel [2].
Usability: Input should be understandable by at least 9/10 users	A key component of our project is to make sure that the system should be user friendly.	Since the average person has used a touch screen device in some capacity, understanding how this touch screen works should be intuitive.

mcrcntrllr_scrn_data : Input

Messages: Location data of user input will be received and UI elements will be sent to the screen	In order for the system to function properly, it needs to communicate with other components in the system. These messages are meant to be simple and not get in the way of other block connections.	I know that my design fits this since I am using an I2C connection to send data to the microcontroller and receive data over SPI. Since these two will be on separate data lines, they will not interfere.
Protocol: Uses SPI to receive data from microcontroller and I2C for sending data to the microcontroller	The I2C protocol is the only way the touch panel can communicate, while SPI was chosen since it required the least amount of wires.	Both of these types of communication are supported by this screen module according to its datasheet [2].
Datarate: Frequency of 80MHz for SPI connection and 100Kbit/s for the I2C connection	These values are based on the base rates for both SPI and I2C connection.	Since I have chosen to use both SPI and I2C, there base rates for data transmission should be easily obtainable when set up by the microcontroller [5], [6].

bttry_scrn_dcpwr : Input

Inominal: 60 mA	This is what the screen should be using in terms of average current.	It is expected that the device will be able to perform with this value based on the datasheet values of the screen's backlight in section 8 [2].
Ipeak: 80 mA	This is the maximum possible current that the screen should use during a limited time such	It is expected that the device will be able to perform with this value based on the datasheet

	as when it first turns on.	values of the screen's backlight in section 8 [2].
Vmax: 3.5 V	This is the maximum voltage that the screen and its backlight should operate at. Going beyond this may cause damage.	It is expected that the device will be able to perform with this value based on the datasheet values of the screen's backlight in section 8 [2].
Vmin: 2.9 V	This is the minimum amount of voltage required to keep the screen and its backlight functioning.	It is expected that the device will be able to perform with this value based on the datasheet values of the screen's backlight in section 8 [2].

scrn_otsd_usrout : Output

Other: The backlight will be visible at least 4 meters away to 9 out of 10 users.	The screen needs to be bright enough to be able to understand what it is presenting on screen.	
Type: Will show UI elements to the user in the form of lit up pixels on a screen	The screen will be the main source of user output and must be robust enough to supply the user with information easily.	The screen I chose for my design accomplishes this by having a 240 by 320 color pixel display [1].
Usability: 9 out of 10 users will say it is readable from 1 meter away.	The user needs to be able to understand what is being presented on the screen.	Due to the screen's resolution, text will be very legible for users with adequate sight.

4.1.5 Verification Process

User Input Testing

1. Connect the screen to a breakout board using the ribbon cable connector.
2. Connect the breakout board to a power supply providing 3.3 volts.
3. Connect the SPI and I2C lines on the breakout board to a microcontroller (a Teensy 3.2 or Arduino Uno will work).
4. Connect the microcontroller to a computer.

5. Run test program on the microcontroller. This program will simply have a prompt that says touch, changing color when touched and sending the microcontroller a signal.
6. Open a serial monitor on the computer to read the upcoming data from the microcontroller. This data will be whether or not the microcontroller received data from the screen along with where the input was if it did receive data.
7. The tester will then perform a couple tasks: taping the screen as fast as they can and varying the length of presses they use.
8. The test is successful if the computer can see input data at least 2 times per second from a single finger.

Microcontroller Data Testing

1. Connect the screen to a breakout board using the ribbon cable connector.
2. Connect the breakout board to a power supply providing 3.3 volts.
3. Connect the SPI and I2C lines on the breakout board to a microcontroller (a Teensy 3.2 or Arduino Uno will work).
4. Run test program on the microcontroller. This program will display text on the screen and wait for user input. After user input is received, an LED on the microcontroller will light up to show that it received data.
5. The microcontroller will send data to the screen telling it to display some information.
6. If the connection is working, the data will be accurately displayed on the screen.
7. The screen will then ask for a tap as input.
8. The tester will tap the screen.
9. The screen will send data to the microcontroller.
10. The test passes if both the screen and the microcontroller are able to receive the appropriate data. This also confirms that both protocols are working at the speed that the microcontroller has set them to.

Power Testing

1. Connect the screen to a breakout board using the ribbon cable connector.
2. Connect the breakout board to a power supply that can control current and voltage.
3. Set the power supply to 3.3 volts.
4. Put a digital multimeter in series with the screen input voltage line in order to measure current. This can be skipped if the power supply can also read current.
5. Connect the SPI and I2C lines on the breakout board to a microcontroller (a Teensy 3.2 or Arduino Uno will work).
6. Run test program on the microcontroller. This test program will simply count numbers on the display.
7. Lower the power supply to 2.9 volts for 1 minute.
8. Raise the power supply to 3.5 volts for 1 minute.

9. The test passes if the screen works as intended for the entire time, keeps a nominal current of 60 milliamps, and never exceeds 80 milliamps.

User Output Testing

1. Connect the screen to a breakout board using the ribbon cable connector.
2. Connect the breakout board to a power supply providing 3.3 volts.
3. Connect the SPI and I2C lines on the breakout board to a microcontroller (a Teensy 3.2 or Arduino Uno will work).
4. Run test program on the microcontroller. This program will simply have the screen display a block of text.
5. Have testers read the block of text.
6. This test passes if the output is readable by the tester.

4.1.6 References and File Links

[1] A. Industries, “2.8’ TFT display - 240X320 with capacitive touchscreen,” adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/2770#technical-details. [Accessed: 11-Feb-2023].

[2] “Adafruit Industries.” [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/091/676/original/AHT20-datasheet-2020-4-16.pdf?1591047915. [Accessed: 11-Feb-2023].

[3] Admin, “Why most smartphones use capacitive technology,” *Nelson Miller*, 27-Jul-2020. [Online]. Available: https://nelson-miller.com/why-most-smartphones-use-capacitive-technology/#:~:text=Whether%20they%20run%20Android%20or,this%20type%20of%20touchscreen%20technology. [Accessed: 11-Feb-2023].

[4] A. Industries, “50-pin 0.5mm pitch top-contact FPC SMT connector,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/1773. [Accessed: 11-Feb-2023].

[5] S. Campbell, “Basics of the I2C communication protocol,” *Circuit Basics*, 14-Nov-2021. [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed: 11-Feb-2023].

[6] S. Campbell, “Basics of the SPI communication protocol,” *Circuit Basics*, 14-Nov-2021. [Online]. Available: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/. [Accessed: 11-Feb-2023].

[7] A. Industries, “Adafruit 50 pin 0.5mm pitch FPC adapter,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/1492. [Accessed: 11-Feb-2023].

[8] Adafruit, “Adafruit/ADAFRUIT_ILI9341: Library for Adafruit ILI9341 displays,” *GitHub*. [Online]. Available: https://github.com/adafruit/Adafruit_ILI9341. [Accessed: 11-Feb-2023].

4.1.7 Revision Table

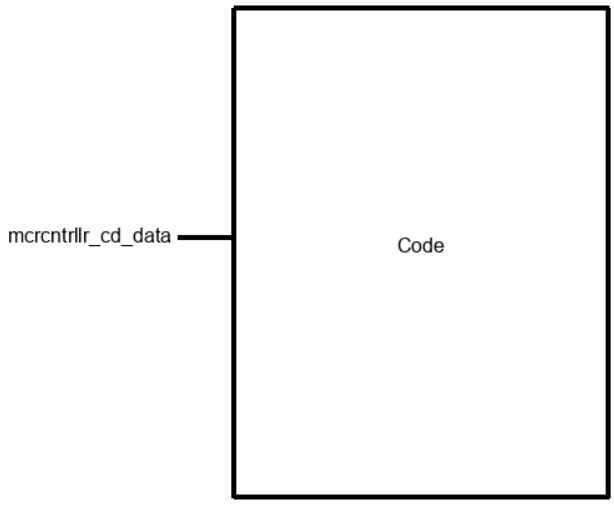
1/19/23, Craig	Created the document and filled out most of the sections with content or placeholders.
1/25/23, Craig	Finished filling out verification plans, added placeholders for in-text citation until references section was complete
2/6/23, Craig	Reworked every section to give a more detailed paper based on feedback, labeled figures and table
2/10/23, Craig	Finished filling out references and file links
2/11/23, Craig	Added an additional interface for the microcontroller data line
5/11/23, Bryan	Updated a scrn_otsd_usrout property to match online tool

4.2 Code Block

4.2.1 Description

The code block will contain a UI for the user to interact and control the unit’s timers to their needs. Furthermore, this block will control what the microcontroller will output to the screen and other peripherals as well as read in values from some peripherals and the battery.

4.2.2 Design



This block will use Arduino as its code base. The code is being broken down into five major sections: screen change, get input, pedometer, peripherals, and timer. Furthermore, the code sets up communication between the microcontroller and the other blocks in the system.

For the screen change function the code will make use of a if statement followed by a switch statement. The if statement will check a boolean value for whether or not the screen needs to be changed. If it is false, it returns from the function without doing anything. Otherwise it leads into a switch statement based on what the screen should be displaying right now. From here it loads the corresponding bitmap.

The get input function will contain a large switch statement. This switch statement will be based on the current screen the system is on. From here it will record input and follow up with any necessary processing.

In the pedometer function the code will keep track of the data coming from the accelerometer. With this data it will look for footsteps and add them to a global counter.

The peripherals function will keep track of and control the miscellaneous peripherals on the board. In our system’s case this includes a LED, a vibration motor, and a simple switch.

Finally, the timer function will use the inputs from the user to create a set of timers. Then once the timers are done, it will return the user back to the main menu.

4.2.3 General Validation

For this block I specifically chose to use Arduino for the code base. This is because of its vast amount of support online and for how robust it has become over the years. Using Arduino also allows us to be able to both test our code on other platforms as well as change platforms if we think that our current one is not working.

Within the code itself, I opted to try and simplify certain actions that are repeated throughout the code into their own smaller independent functions. This allows me to make use of more space on the device since it can simply jump to the function during execution rather than repeating the same set of instructions over and over.

4.2.4 Interface Validation

mcrcntrllr_cd_data : Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Datarate: frequency of 80MHz for SPI connection and 100Kbit/s for the I2C connection	This interface is this value because that is the default rate for both.	My design meets this value since it sets the rate to default for both communication types.
Messages: Screen communication, battery level, peripheral values	This interface is this value because these are the types of messages we plan to send and receive.	My design meets these expected values as I have picked out specific libraries that create these connections and manage them.

Protocol: Configures SPI for sending data to the screen, I2C for receiving data from the screen and accelerometer, an analog pin for battery level, one digital pin for input, and three digital pins for output.	This interface is this value because these are the only ways to communicate with some of our chosen parts.	My design meets these values since the libraries being used manages it in the background.

4.2.5 Verification Process

4.2.6 References and File Links

4.2.7 Revision Table

Craig 3/14/23	Created section and filled out

4.3 Battery Block

4.3.1 Description

The battery block will contain a battery cell connected to a student designed charge regulation module. This will provide all the power to the rest of the system, and it will regulate the voltage level from the battery to meet the needs of the other blocks. This block ensures that the power draw from the rest of the system can be met while ensuring the system requirement for how long the system can be used in a single charge is met.

4.3.2 Design

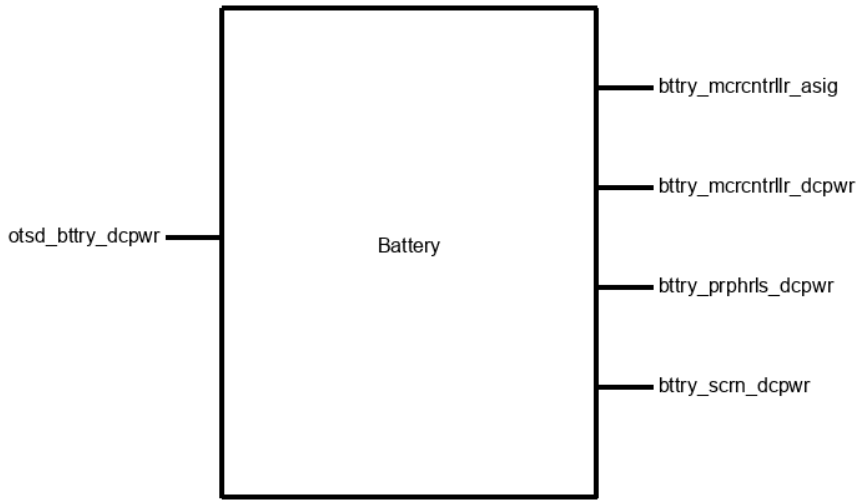


Figure 1: Black box diagram of the Battery Block.

The battery block will consist of a 2500mAh lithium ion polymer battery and a custom designed PCB board that will be similar to Lithium Polymer Charging and Boost Converter Circuit Layout from one of the recitation sessions in ECE 441. This custom designed PCB board will need to ensure that the charge coming from an external power source, which will be a 5V DC USB input in our case, will be regulated and prevent the battery from overcharging. This external power supply will feed into the block through the otsd_bttry_dcpwr interface.

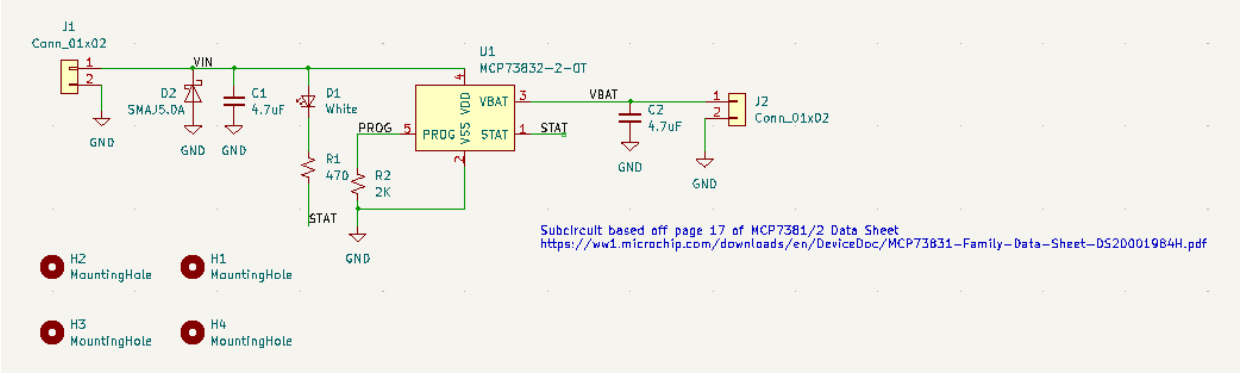


Figure 2: Schematic clip of charge regulation module.

The MCP73832T-2DCI/OT chip used in the Lithium Polymer Charging and Boost Converter Circuit Layout will be used for the Battery Block as well. The Battery Block's custom designed PCB board will also feature a DC to DC converter, however a step down DC to DC converter will need to be used instead of the previous step up DC to DC converter in order to achieve an output of 3.3V that is required by the rest of the system.

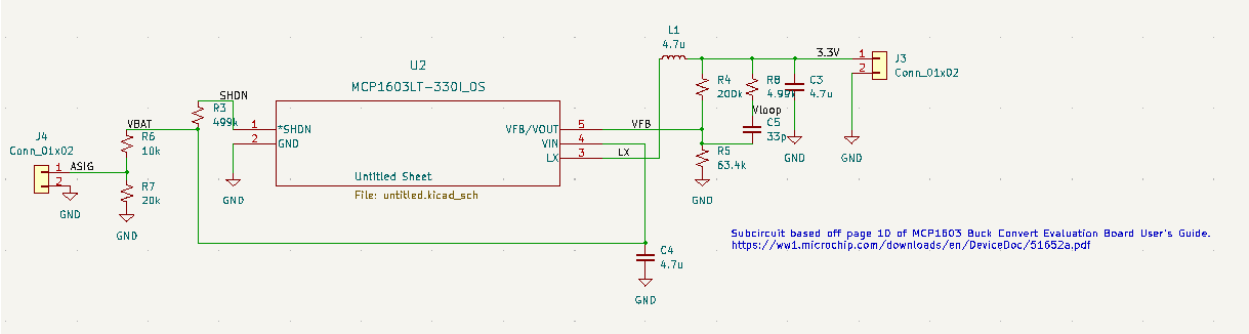


Figure 3: Schematic clip of DC to DC converter module.

The MCP1603LT-330I/OS buck regulator chip was chosen for this design. The design for the buck converter was based on the MCP1603 Buck Converter Evaluation Board. This configuration from the board maintains a 3.3V output voltage using a resistor network. Basing the design of the evaluation board allows for the correct sourcing of components when it comes to parameters like the inductor current rating.

This 3.3 voltage will be outputted from the block and supply power to the screen block through the bttry_scrn_dcpwr block, to the peripherals block through the bttry_prphrls_dcpwr interface, and to the microcontroller block through the bttry_mcrcntrllr_dcpwr interface.

An additional output, the bttry_mcrcntrllr_asig interface, will come from the battery using a voltage divider to ensure the voltage is less than 0.75V. The voltage divider will consist of a 10K ohm resistor and a 20K ohm resistor. The microcontroller pin will be connected to the positive end of the 20K ohm resistor. This voltage divider will ensure that input voltage will be scaled down for the microcontroller to read the battery's voltage level without overloading the pin. A 10K ohm resistor and 20K ohm resistor were chosen to limit the amount of current going through the voltage divider circuit.

4.3.3 General Validation

When deciding on a battery to use for the system, two specifications needed to be considered. The first would be the size of the battery. Our system is limited in the weight and size of the final product, and the battery would be one of the most likely candidates to contribute the most to those two measurements. The second thing to consider would be the total capacity of the battery. Our system has requirements on the amount of time that it can run off of a single charge during idle and active use of the system. These two specifications often relate to each other inversely, which made it important to choose the right rechargeable battery.

A 2500mAh lithium ion polymer battery was chosen because it met the right balance between size and capacity. The battery is slim enough to fit comfortably within our final enclosure, and it will not contribute too much weight to the system.[1] 2500mAh was chosen after some calculations for the possible current

draws from our system. More specifically, the screen, microcontroller, and peripheral's power consumptions were considered. During the system's lowest power consumption state, the system would only need to draw 25mA, which would allow the system to run for around 100 hours. For a worst case scenario of power consumption, the system would need to draw 260mA of current, which would allow the system to run for a little under 10 hours. However, this maximum power consumption state assumes that the microcontroller is running at max power, the screen is turned on, and all peripherals are turned on consistently, which is unlikely to happen for too long if at all.

For the charge regulation portion of the block, the MCP73832T-2DCI/OT chip was chosen due to the ability to reuse parts of the design from the Lithium Polymer Charging and Boost Converter Circuit Layout. The circuit was designed for lithium polymer batteries, so it should work well with our battery.[2] A step down DC to DC converter is necessary in order to maintain an output voltage level of 3.3V. The screen and microcontroller require around 3.3V to operate, so this will help ensure the voltage level stays consistent. The LED and vibration motor in the peripherals block also need around 3.3V, so they can also accept this output from the battery block. The MCP1603LT-330I/OS chip was chosen for a multitude of reasons. Its input voltage can range from 2.7V to 5.5V. The battery's voltage will stay within 3V and 4.2V, so there is no issue with the input voltage falling out of range. The output voltage is fixed at 3.3V by the resistor network, so maintaining a 3.3V output voltage should not be difficult. Finally, the chip can handle up to 500mA of output current, which is above the maximum possible current draw of the system.[3]

4.3.4 Interface Validation

This interface validation table lists the justifications for every property of every interface for the battery block. The needs of the system and the characteristics of the battery, charge regulation chip, and buck converter chip were all examined to build these justifications.

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
otsd_bttry_dcpwr : Input		
Inominal: 500 mA	According to the datasheet for the charge regulator module, the output current is equal to 1000V divided by the R_{PROG} resistor. In	R_{PROG} is set to 2K ohms in the design, so this current should be guaranteed. Additionally, there is no requirement for the charge time

	this case the R_{PROG} resistor is 2000 ohms, so the output current should sit around 500mA.[2]	of the system, so there is no requirement on the current going into the battery.
Ipeak: 550 mA	According to the datasheet for the charge regulator module, this is the maximum current it can provide to the battery.[2]	R_{PROG} is set to 2K ohms in the design, so this current should be guaranteed and below the maximum of 550mA.
Vmax: 6 V	According to the datasheet for the charge regulator module, this is the maximum voltage it can receive.[2]	The charge regulator chip in the design will be the first object that receives the outside power.
Vmin: 3.75 V	According to the datasheet for the charge regulator module, this is the minimum voltage it can receive.[2]	The charge regulator chip in the design will be the first object that receives the outside power.

bttry_mcrcntrllr_asig : Output

Vmax: 0.75 V	According to the datasheet for the microcontroller, pins need voltages between 0V and 0.75V to properly read a voltage.	The voltage divider will ensure that the voltage is scaled down. The maximum possible voltage after the voltage divider is 0.73V.
Vrange: 0V - 0.75V	According to the datasheet for the microcontroller, pins need voltages between 0V and 0.75V to properly read a voltage.	The voltage divider will ensure that the voltage is scaled down. The maximum possible voltage after the voltage divider is 0.73V, so any other voltage will fall below that.

bttry_mcrcntrllr_dcpwr : Output

Inominal: 30mA	According to the microcontroller's datasheet, this is the minimum amount of current the microcontroller will draw after all unnecessary functionality is turned off.	The maximum discharge current, according to the datasheet for the battery, is 1500mA. This is a lot more than needed.
Ipeak: 500mA	According to the microcontroller's datasheet, this is the maximum amount of current the microcontroller will draw at start	The maximum discharge current, according to the datasheet for the battery, is 1500mA. This is a lot more than needed. Additionally,

	up before any unnecessary functionality is turned off.	the buck converter chip can supply this amount of current as well.
Vmax: 3.6V	This is the maximum possible voltage the microcontroller can take to operate according to its datasheet.	The step down converter will be set to regulate the output voltage level to around 3.3V. This is below the maximum of 3.6V.
Vmin: 3V	This is the minimum possible voltage the microcontroller can take to operate according to its datasheet.	The step down converter will be set to regulate the output voltage level to around 3.3V. This is above the minimum of 3V.

bttry_prphrls_dcpwr : Output

Inominal: 20mA	The LED in the peripherals block draws 20mA. Having the LED on by itself is one of the modes of operation.[4]	The maximum discharge current, according to the datasheet for the battery, is 1500mA. This is a lot more than needed.
Ipeak: 80mA	The LED in the peripherals block draws 20mA while the vibration motor draws 60mA at 3.3V. Having both peripherals on is one of the modes of operation.[4] [6]	The maximum discharge current, according to the datasheet for the battery, is 1500mA. This is a lot more than needed.
Vmax: 3.5V	The LED and the vibration motor in the peripherals block both operate with 3.5V. Resistors can be used for voltage dividers as well.[4] [6]	The step down converter will be set to regulate the output voltage level to around 3.3V. This is below the maximum of 3.5V
Vmin: 3.0V	The LED and the vibration motor in the peripherals block both operate with 3.0V. [4] [5]	The step down converter will be set to regulate the output voltage level to around 3.3V. This is above the minimum of 3.0V.

bttry_scrn_dcpwr : Output

Inominal: 60 mA	The screen draws 60mA during normal operation according to its datasheet.[6]	The maximum discharge current, according to the datasheet for the battery, is 1500mA. This is a lot more than needed.
Ipeak: 80 mA	The screen draws 60mA during	The maximum discharge current,

	high power operation according to its datasheet.[6]	according to the datasheet for the battery, is 1500mA. This is a lot more than needed.
Vmax: 3.5 V	This value comes from the datasheet for the screen.[6]	The step down converter will be set to regulate the output voltage level to around 3.3V. This value is below the maximum of 3.5V.
Vmin: 2.9 V	This value comes from the datasheet for the screen.[6]	The step down converter will be set to regulate the output voltage level to around 3.3V. This value is above the minimum of 2.9V.

4.3.5 Verification Process

This verification plan lists the steps that should be taken to verify that the battery block is capable of operating at, or supplying the listed values. Steps are given for every property of every interface for the block. A DC power supply, a multimeter, and a configurable load resistance is required to perform many of these tests.

Otsd_bttry_dcpwr: Inom = 500mA

1. Connect a power supply set to 5V DC to the input of the block with a multimeter in series to measure the current while ensuring that the battery is discharged.
2. Turn on the power supply and observe the current measured by the multimeter.
3. If it is around 500mA, then it meets the specification.

Otsd_bttry_dcpwr: Ipeak = 550mA

1. Connect a power supply set to 5V DC to the input of the block with a multimeter in series to measure the current while ensuring that the battery is discharged.
2. Turn on the power supply and observe the current measured by the multimeter.
3. If it is at or under 550mA, then it meets the specification.

Otsd_bttr_dcpwr: Vmax = 6V

1. Connect a power supply set to 6V to the input of the block while ensuring that the battery is discharged.

2. Turn on the power supply and observe if current is being drawn out of the power supply.
3. If current is being supplied, then it meets the specification.

Otsd_bttr_dcpwr: Vmin = 3.75V

1. Connect a power supply set to 3.75V to the input of the block while ensuring that the battery is discharged.
2. Turn on the power supply and observe if current is being drawn out of the power supply.
3. If current is being supplied, then it meets the specification.

Bttry_mcrcntrllr_asig: Vmax = 0.75V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for the asig signal.
2. If the asig voltage is below 0.75V, then it meets the specification.

Bttry_mcrcntrllr_asig: Vrange = 0V - 0.75V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for the asig signal.
2. If the asig voltage is within 0V and 0.75V, then it meets the specification.

Bttry_mcrcntrllr_dcpwr: Ipeak = 500mA

1. Ensure that the battery is charged and connect a multimeter between the output and a resistance closest to 6.6 ohms. Ensure the multimeter is measuring current.
2. If the multimeter is around 500mA, then it meets the specification.

Bttry_mcrcntrllr_dcpwr: Inom = 30mA

1. Ensure that the battery is charged and connect a multimeter between the output and a resistance closest to 110 ohms. Ensure the multimeter is measuring current.
2. If the multimeter is around 30mA, then it meets the specification.

Bttry_mcrcntrllr_dcpwr: Vmax = 3.6V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for this interface.
2. If the output voltage is below 3.6V, then it meets the specification.

Bttry_mcrcntrllr_dcpwr: Vmin = 3V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for this interface.
2. If the output voltage is above 3V, then it meets the specification.

Bttry_prphrls_dcpwr: Inominal = 20mA

1. Ensure that the battery is charged and connect a multimeter between the output and a resistance closest to 165 ohms. Ensure the multimeter is measuring current.
2. If the multimeter is around 20mA, then it meets the specification.

Bttry_prphrls_dcpwr: Ipeak = 80mA

1. Ensure that the battery is charged and connect a multimeter between the output and a resistance closest to 41 ohms. Ensure the multimeter is measuring current.
2. If the multimeter is around 80mA, then it meets the specification.

Bttry_prphrls_dcpwr: Vmax = 3.5V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for this interface.
2. If the output voltage is below 3.5V, then it meets the specification.

Bttry_prphrls_dcpwr: Vmin = 3.0V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for this interface.
2. If the output voltage is above 3.0V, then it meets the specification.

Bttry_scrn_dcpwr: Inominal = 60mA

1. Ensure that the battery is charged and connect a multimeter between the output and a resistance closest to 55 ohms. Ensure the multimeter is measuring current.
2. If the multimeter is around 60mA, then it meets the specification.

Bttry_scrn_dcpwr: Ipeak = 80mA

1. Ensure that the battery is charged and connect a multimeter between the output and a resistance closest to 41.25 ohms. Ensure the multimeter is measuring current.
2. If the multimeter is around 80mA, then it meets the specification.

Bttry_scrn_dcpwr: Vmax = 3.5V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for this interface.
2. If the output voltage is below or at 3.5V, then it meets the specification.

Bttry_scrn_dcpwr: Vmin = 2.9V

1. Ensure that the battery is charged and use a multimeter to probe the voltage at the output for this interface.
2. If the output voltage is over or at 2.9V, then it meets the specification.

4.3.6 References and File Links

[1] X. Nie, “Li-polymerbatterytechnologyspecification.” [Online]. Available: https://cdn-shop.adafruit.com/product-files/328/LP785060+2500mAh+3.7V+20190510.pdf. [Accessed: 21-Jan-2023].

[2] “MCP73831/MCP73832 data sheet - microchip technology.” [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf. [Accessed: 21-Jan-2023].

[3] “MCP71603/B/L data sheet - microchip technology.” [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/22042B.pdf. [Accessed: 11-Feb-2023].

[4] “Chip nano pico and Z leds,” Evan Designs. [Online]. Available: https://evandesigns.com/collections/hobby-leds/products/chip-nano-pico-leds?variant=32158378885168. [Accessed: 20-Jan-2023].

[5] “产 品 规 格 书.” [Online]. Available: https://cdn-shop.adafruit.com/product-files/1201/P1012_datasheet.pdf. [Accessed: 21-Jan-2023].

[6] “MbedAdafruitLCD/SPEC-DT280QV10-ct_rev.b.pdf at master · Adamgreen ...” [Online]. Available: https://github.com/adamgreen/mbedAdafruitLCD/blob/master/docs/SPEC-DT280QV10-CT_Rev.B.pdf. [Accessed: 21-Jan-2023].

4.3.7 Revision Table

1/20/2023 Bryan	Document created & initial draft procured.

2/10/2023 Bryan	Updated format based on new interfaces and added notes for area needing more detail.
2/11/2023 Bryan	Completed empty entries in interface validation table. Added details on buck converter chip in validation and design sections. Added details to description.
3/10/2023 Bryan	Updated DC DC converter description with new chip.
4/28/2023 Bryan	Updated buck converter sections for new chip.
5/11/2023 Bryan	Fixed references ordering to match IEEE standards

4.4 Peripheral Block

4.4.1 Description

The peripherals block will include components that output light and vibrations as well as the necessary circuitry to switch those on and off using the microcontroller. An accelerometer is also contained within this block which allows for the system to track the user's steps.

4.4.2 Design

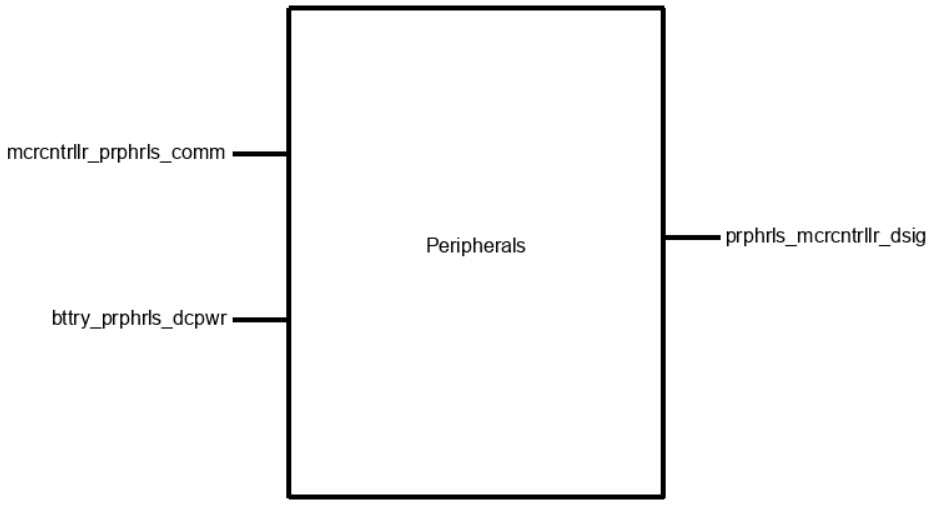


Figure 1: Black box diagram of the Peripherals Block.

The peripheral block will be located on the main central PCB for the system. It will consist of a vibration motor, LED, accelerometer, a single pole single throw switch, and a few resistors and transistors for switching on the LED and motor using the microcontroller.

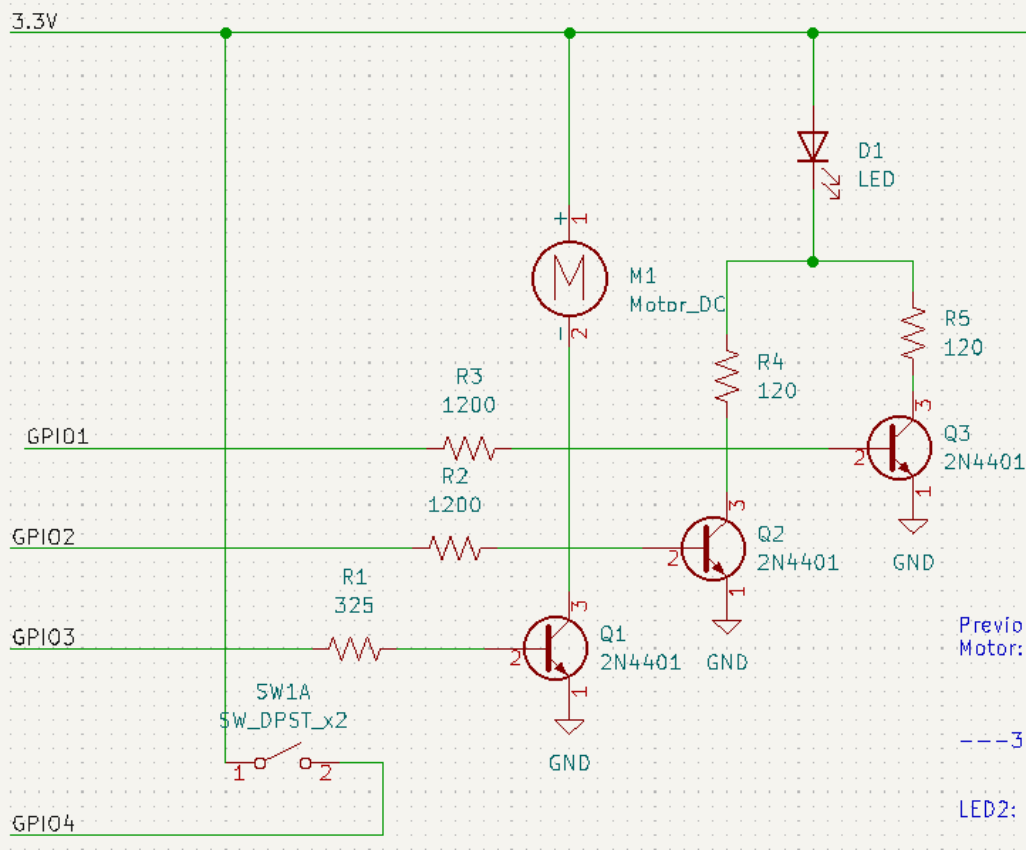


Figure 2: Schematic clip of motor, LED, switch, resistors, and transistors

The microcontroller pins will be used to drive the base of each NPN 2N4401 BJT (MMBT4401 equivalent SMD) to 3.3V and thus allow current to flow through the collector. The vibration motor can be turned on by itself with one of the microcontroller pins. It can also be turned in in conjunction with the LED. The vibration motor draws around 60mA when powered by a 3.3V line. [1] A vibration motor was used in order to provide haptic feedback when the user wishes to select that mode. Its small size and power draw were chosen to fit the small enclosure of our system and not draw too much from the battery when it is in use. A 325 ohm resistor is connected to the base of the transistor to limit current coming from the microcontroller pin and link with the current through the collector with a factor of around 10.

The LED is of size 3.2mm across and was chosen for its small size to fit easily inside the enclosure. It was also chosen since it can work with around 3.3V without reaching its maximum possible input voltage. [2] The LED is connected to

two transistors each with a 500 ohm resistor connected to the collector to limit the current draw. 2 microcontroller pins are to be used to adjust the brightness of the LED based on whether the user wants to operate in a discrete or standard mode. 1200 ohm resistors are connected to each base as well to limit current draw from the microcontroller pins.

The peripherals block also houses the terminals for a single pole single throw switch that will be attached to the enclosure. This switch connects or disconnects a microcontroller pin to the 3.3V power line so that the microcontroller can switch between a discrete and standard mode of operation for the system. An accelerometer is also part of the peripherals block. The accelerometer is the ADXL343 module from Adafruit. The peripherals block will contain 4 connection points for the accelerometer: VIN which will connect to the 3.3V line, GND, and 2 I2C lines that feed back to the microcontroller. [3]

4.4.3 General Validation

When deciding on how to generate light and haptic feedback for the timers, an LED and vibration motor seemed like good choices. One of the main concerns was that the LED and motor needed to be small enough to fit inside of the enclosure. From there, both components also needed to be able to take around 3.3V like the rest of the system. The LED was more than small enough for the system and operated around that voltage level. The motor also operated around that voltage level and was small enough as well. The power draw for both components was also reasonable enough, especially considering how both components won't be active all the time. With both components turned on, it can be expected that less than 100mA of current will be drawn. This won't make too much of an impact on the battery life.

A simple single pole single throw switch was chosen for this block since not much complexity is involved with switching the mode the system operates in in terms of peripherals. Our microcontroller will read a pin and see whether or not it's at 3.3V to determine what mode to set. The switch will also be easy to flip when mounted on the enclosure to easily switch between the modes.

The ADXL343 accelerometer was chosen to meet the requirements of having a pedometer in the system. Accelerometers are commonly used to create pedometers, and this module was also small enough to fit in our enclosure. Our microcontroller uses I2C to communicate with other parts of the system, so the fact that this accelerometer could communicate with I2C worked well for our needs.

4.4.4 Interface Validation

This table contains each of the interfaces for the peripherals block, as well as descriptions on the need for the values of each interface and how they are ensured through the design.

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
mcrcntrllr_prphrls_comm : Input		
Datarate: 100 Kbit/s	I2C communication is typically done in 100Kbit/s.	I2C lines will be connected from the accelerometer to the microcontroller through connection points on the central PCB.
Messages: Accelerometer data	The accelerometer is the only module that will be communicating with the microcontroller that doesn't involve just switching something on. Its data will be read.	The block contains an accelerometer module.
Protocol: I2C	The microcontroller and accelerometer both can communicate in I2C.	The accelerometer being used in this block can communicate in I2C.
prphrls_mcrcntrllr_dsig : Output		
Logic-Level: Active high	The voltages on the bases of the BJTs need to be higher than the emitter which is connected to ground.	The microcontroller pins connect to the bases of the BJTs through a resistor and can output 3.3V to turn them on.
Vmax: 3.6 V	The microcontroller pins can in theory output no higher than this voltage.	The BJT can take in a 3.6V input to the base since there is a resistor that limits the amount of current fed into it.
Vmin: 0 V	The microcontroller pins can be set to 0 volts at the minimum.	The BJT will be off when both the base and emitter are set to 0.

bttry_prphrls_dcpwr : Input

Inominal: 20mA	The LED draws around 20mA when on at full power by itself.	The LED being on by itself is one of the modes of operation for the system peripherals.
Ipeak: 80mA	Having the LED and the motor on draws around 80mA.	The LED and the motor being on is one of the modes of operation for the system peripherals.
Vmax: 3.5V	Both the LED and the motor can take up to 3.5V.	The motor and LED are in series with resistors and transistors, so there should be no worry about both components having too much voltage.
Vmin: 3.0V	Both the LED and motor can operate on a minimum of 3.0V.	The transistors have voltage requirements for collector-emitter and base-emitter, so the LED and motor circuits should operate if the input is too low.

Figure 3: Interface Validation Table

4.4.5 Verification Process

This verification plan lists the steps that should be taken to verify that the peripherals block is capable of operating at, or supplying the listed values. Steps are given for every property of every interface for the block. A DC power supply, a multimeter, a configurable load, and a microcontroller is required to perform many of these tests.

mcrcntrllr_prphrls_comm: Datarate = 100kbit/s

1. Connect the block to a microcontroller capable of communicating in I2C.
2. Collect accelerometer data using the microcontroller after setting up I2C communication to run at a 100kbit/s datarate.
3. If data is transferred successfully, then it meets the specification.

mcrcntrllr_prphrls_comm: Messages = Accelerometer Data

1. Connect the block to a microcontroller capable of communicating in I2C.
2. Collect accelerometer data using the microcontroller after setting up I2C communication.
3. If acceleration is measured, then it meets the specification.

mcrcntrllr_prphrls_comm: Protocol = I2C

1. Connect the block to a microcontroller capable of communicating in I2C.
2. Collect accelerometer data using the microcontroller after setting up I2C communication.
3. If acceleration is measured, then it meets the specification.

prphrls_mcrcntrllr_dsig: Logic-level = Active high

1. Connect a power supply set to 3.3V DC to one of the BJT base inputs, either the motor or LED.
2. Turn on the power supply and observe if the motor or LED turns on.
3. If it turns on, then it meets the specification.

prphrls_mcrcntrllr_dsig: Vmax = 3.6V

1. Connect a power supply set to 3.6V DC to one of the BJT base inputs, either the motor or LED.
2. Turn on the power supply and observe if the motor or LED turns on.
3. If it turns on, then it meets the specification.

prphrls_mcrcntrllr_dsig: Vmin = 0V

1. Connect a power supply set to 0V DC to one of the BJT base inputs, either the motor or LED.
2. Turn on the power supply and observe if the motor or LED stay off.
3. If it stays off, then it meets the specification.

bttry_prphrls_dcpwr: Inominal = 20mA

1. Connect a power supply set to 3.3V DC to one of the LED BJT base inputs. Connect another one to the power line of the block.
2. Turn on both of the power supplies and observe the current measured by the multimeter coming from the power line.
3. If it is around 20mA, then it meets the specification.

bttry_prphrls_dcpwr: Ipeak = 80mA

1. Connect a power supply set to 3.3V DC to one of the LED BJT base inputs and to the motor BJT base input. Connect another one to the power line of the block.
2. Turn on both of the power supplies and observe the current measured by the multimeter coming from the power line.
3. If it is around 80mA, then it meets the specification.

bttry_prphrls_dcpwr: Vmax = 3.5V

1. Connect a power supply set to 3.5V DC to the power line of the block. Connect another 3.3V DC power supply to one of the BJT base inputs of either the LED or motor.
2. Turn on the power supplies and observe if current is being drawn out of the power supplies.
3. If it is, then it meets the specification.

bttry_prphrls_dcpwr: Vmin = 3.0V

1. Connect a power supply set to 3.0V DC to the power line of the block. Connect another 3.3V DC power supply to one of the BJT base inputs of either the LED or motor.
2. Turn on the power supplies and observe if current is being drawn out of the power supplies.
3. If it is, then it meets the specification.

4.4.6 References and File Links

[1] “产 品 规 格 书.” [Online]. Available: https://cdn-shop.adafruit.com/product-files/1201/P1012_datasheet.pdf. [Accessed: 12-Mar-2023].

[2] “Chip nano pico and Z leds,” *Evan Designs*. [Online]. Available: https://evandesigns.com/collections/hobby-leds/products/chip-nano-pico-leds. [Accessed: 12-Mar-2023].

[3] “Digital MEMS accelerometer data sheet ADXL343 - Adafruit Industries.” [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/070/556/original/adxl343.pdf?1549287964. [Accessed: 12-Mar-2023].

4.4.7 Revision Table

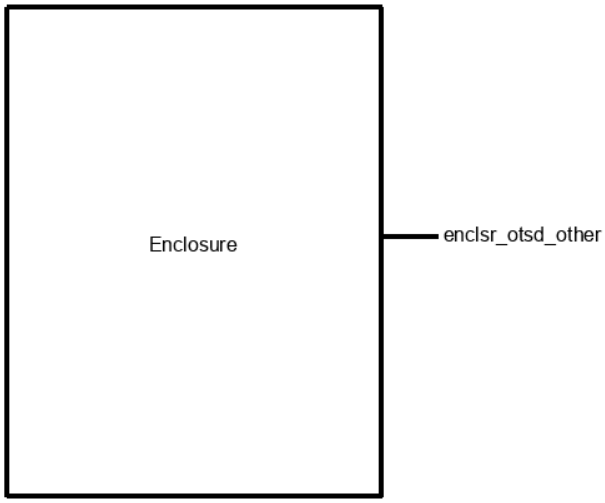
3/10/2023 Bryan	Document created. Design section started.
3/11/2023 Bryan	Design section, general validation, and interface validation finished.
3/12/2023 Bryan	References section finished.
4/28/2023 Bryan	Updated resistor values.
5/11/2023 Bryan	Updated references order to match IEEE standards

4.5 Enclosure Block

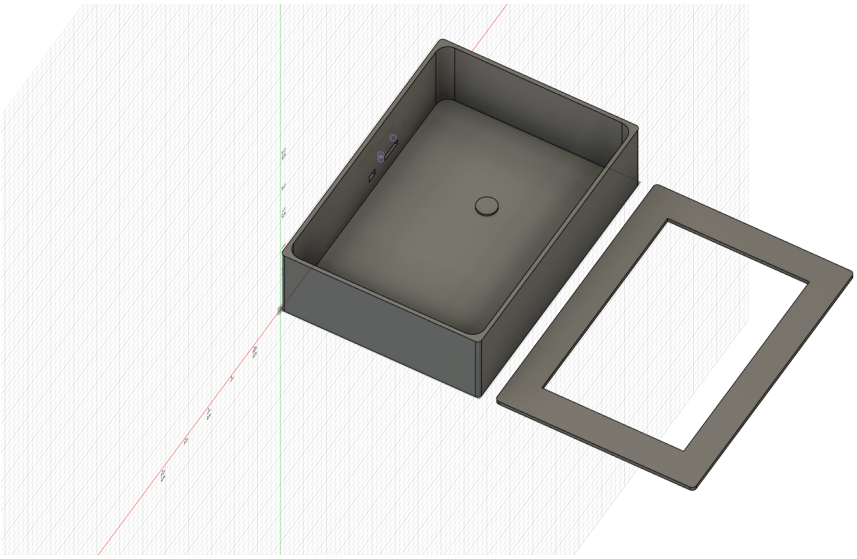
4.5.1 Description

The Enclosure will contain all the components and provide access to the inputs. It will also feature a belt-clip, to allow the whole device to be worn.

4.5.2 Design



The enclosure will be a hollow rectangular prism with 3 rectangular cutouts: 2.74x1.78in, 6.9x1.85mm, 3.2x2mm.



4.5.3 General Validation

There are three main reasons we need an enclosure block: to hold all the parts together, and to make the finished product easy to handle, and to allow it to be clipped onto a belt. A simple 3D print is sufficient for these functions.

4.5.4 Interface Validation

enclsr_otsd_other : Output

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Other: 6.9x1.85mm opening for micro USB port	This hole is slightly larger than a micro usb port to allow for small errors.	All dimensions of the enclosure are designed in CAD and then 3D printed, allowing for precise control of dimensions.
Other: Max outer dimensions: 2.5x3.5x1 in	The outer dimensions are a system requirement.	All dimensions of the enclosure are designed in CAD and then 3D printed, allowing for precise control of dimensions.
Other: Max Internal dimensions: 2.4x3.4x0.9 in	The inner dimensions should be at most slightly smaller than the max outer dimensions.	All dimensions of the enclosure are designed in CAD and then 3D printed, allowing for precise control of dimensions.
Other: 3.2x2 mm opening for LED	These are the dimensions of the LED we intend to use.	All dimensions of the enclosure are designed in CAD and then 3D printed, allowing for precise control of dimensions.
Other: Belt clip that will clip onto surfaces that are 1/4 inch	¼ inch seemed like a reasonable estimate of a particularly thick belt.	The belt clip was taken from a tape measure, which easily stretched open ¼ inch.

4.5.5 Verification Process

All the verification will be done by measuring the enclosure, with the exception of the belt clip which will be demonstrated by stretching the clip before measuring it.

4.5.6 References and File Links

4.5.7 Revision Table

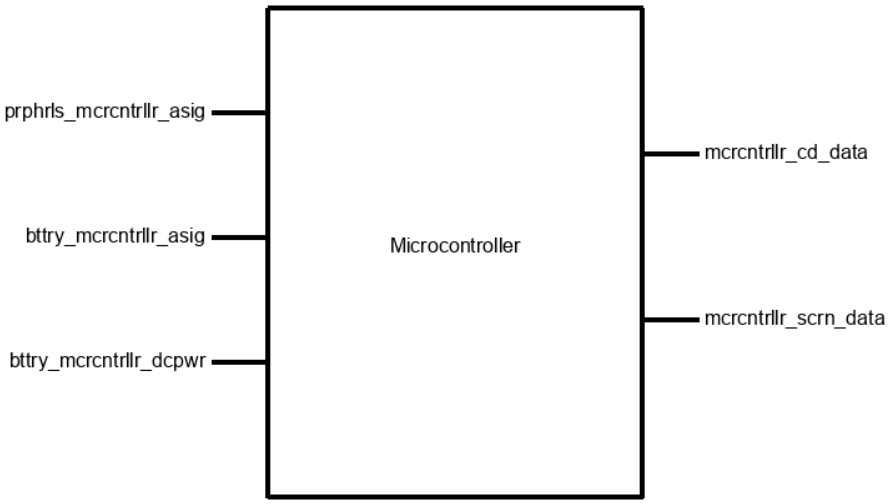
2/27/2023	**Rory Patterson: write draft**

4.6 Microcontroller Block

4.6.1 Description

The microcontroller is designed to connect the code to the other elements of the project. It allows the code to communicate with the screen and the touch panel to communicate back; using an I2C connection to update the screen and an SPI connection to receive touch data. It uses a series of digital I/O to interact with the peripherals block. And it uses an ADC to provide information about the battery to the controller. Additionally, the microcontroller’s clock is how the whole project keeps track of time.

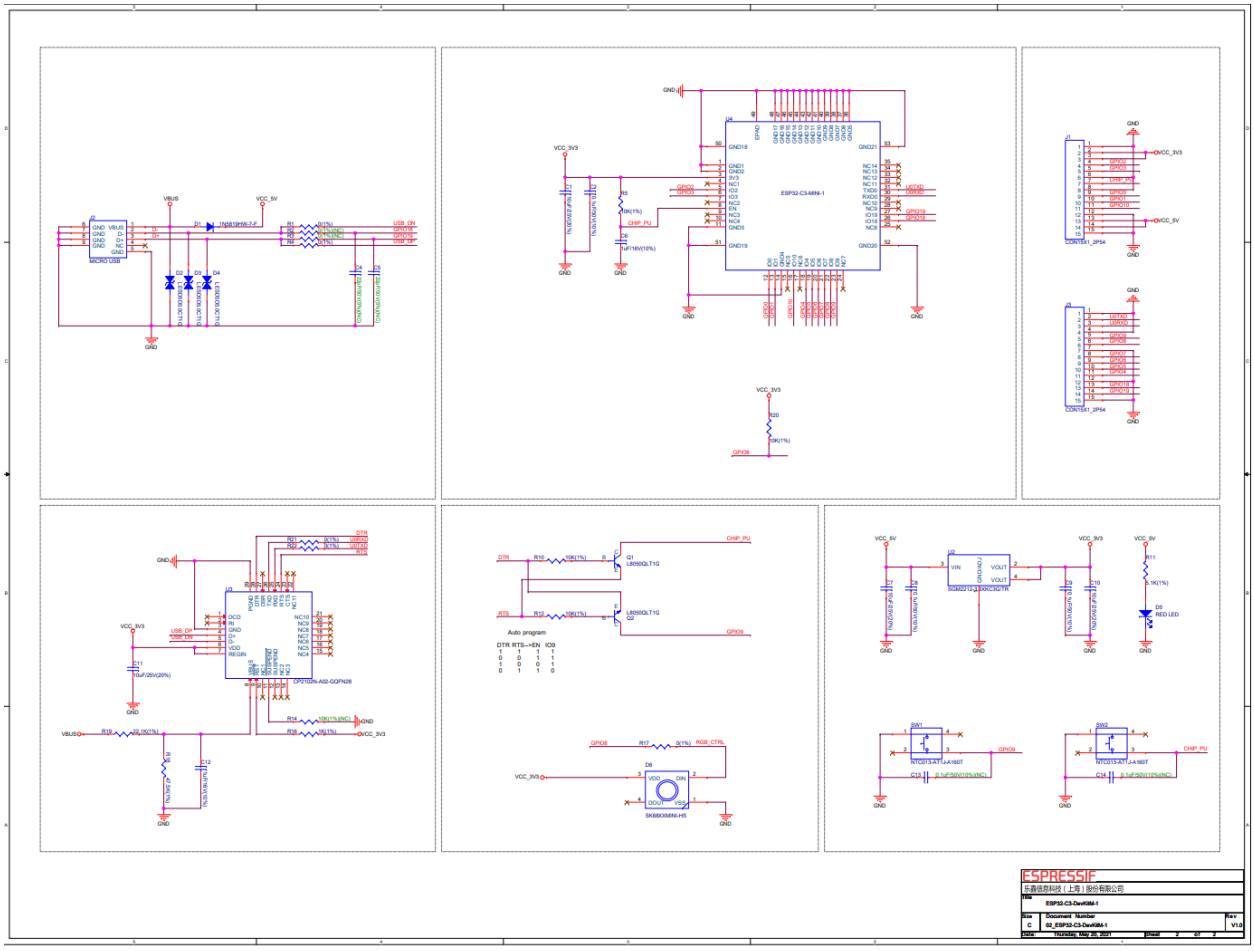
4.6.2 Design



The microcontroller block will be based around a ESP32-C3 microcontroller. It will have a small module contained within this block to regulate the power. This isn't strictly necessary, but since it's one of the most delicate parts of the project we chose to err on the side of caution. It will also contain a series of I/O connections for SPI and I2C which are needed for communicating with the screen, and an ADC which is used for reading the current battery level. The block will use a USB programming chip to make it much easier to upload the software to the controller. Finally there will be a series of digital I/O for interacting with the peripherals board.

For the verification we will be using an ESP32-C3-DevKitM-1, but the final version will use an ESP32-C3 mounted to a custom PCB. This PCB will house the microprocessor, the USB programming chip, and a series of capacitors to regulate fluctuations in the power supply.

Below are the schematics for the ESP32-C3-DevKitM-1 we will be using for verification.



4.6.3 General Validation

The project needs some kind of microcontroller to control the display, touch interface, peripherals, and to keep time. The ESP32 was selected because the team has experience working with one such controller from fall's recitations, and as that project has a PCB design that can be adjusted to fit these needs.

The selected screen was Adafruit's "2.8" TFT Display - 240x320 with Capacitive Touchscreen" as a cost effective capacitive touch screen. The chosen screen communicates in several ways, but the code block's champion elected to use the combination of I2C and SPI for sending the images to the screen and receiving the touch inputs. Because of this the microcontroller will need to be able to use those communication protocols. The EPS32 is capable of both. In master mode the ESP32-C3 operates at 80MHz, since this channel will only be transmitting data to the screen we only need to be concerned with the master speed. The I2c connection operates at 100 Kbit/s in standard mode and at 400 Kbit/s in fast mode.

Some of the components the system uses will require more current then the microcontroller can output. To get around this limitation the peripherals block was created to use transistors to provide the required current to the components such as a vibration motor and LED, the base current can be controlled by connecting to microcontrollers digital I/O pins to digital low controlling the transistor's collector and emitter currents. It will also monitor a switch as a user input with one of the digital I/O pins.

While not integral to the system's function, it would be nice to be able to display current battery levels to the user. The microcontroller contains a 12-bit ADC, which will accomplish this handily. Although lithium batteries' charge is difficult to measure from its voltage, this can still serve as a low battery indicator. The ranges of the ADC are lower than the battery will supply, so an extremely high impedance voltage divider will allow the ADC to measure a known fraction of the battery voltage, and use that fraction to calculate the total battery voltage.

Within the PCB, there will be a small power management system. The battery block already controls the power output, but it only costs a few cents extra to add some capacitors to smooth out the input. Power regulation is also used for the USB to UART chip which allows the chip to be programmed much more easily, which will save a lot of headache during testing. There will be a series of low profile connectors to allow the microcontroller to be plugged into the other components. Thinness is important for this design since it will be sharing space inside the enclosure with a screen, battery, power control module, and peripheral module.

The ESP32 is capable of clock gating and lowering its cores used and clock speed which will allow the device as a whole to use less power and allow the battery to last longer. It also has a "modem sleep" function which turns off the wireless components; this will dramatically reduce the nominal current draw. While in modem sleep it can also operate at 2 different clock speeds: 160MHz and 80MHz. Since the difference in current draw is relatively similar between the two speeds, we will likely use 160MHz which will hopefully make the display more responsive.

4.6.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?

mcrcntrllr_cd_data : Output

Datarate: frequency of 80MHz for SPI connection and 100Kbit/s for the I2C connection	**These data rates were determined by what the ESP32-C3 is capable of. Both for SPI and I2C**	**These are listed in the ESP32-C3 datasheet as the frequency for SPI in master mode, which we will be using since it's the microcontroller sending data to the screen, and the standard mode data rate for its I2c interface.**
Messages: Screen communication, battery level, peripheral values	**Text and screen layout sent to the screen module. This will ultimately be determined by the finished code block.**	**The content of the code will be determined by the code.**
Protocol: Configures SPI for sending data to the screen, I2C for receiving data from the screen and accelerometer, an analog pin for battery level, one digital pin for input, and three digital pins for output.	**These values are determined by the screen we chose to use, the decision to add battery monitoring, and the peripherals.**	**The ESP32-C3 datasheet lists SPI and I2C in the "peripherals and sensors" section. It also lists bitrates and operating frequencies which informed the data rate values.**

mcrcntrllr_scrn_data : Output

Messages: Location data of user input will be received and UI elements will be sent to the screen	**Our main means of allowing the user it interact with the device will be through the touch screen, so we need to display information on, and read inputs from, the screen**	**This data is sent by the screen and read by the code, all the microcontroller has to do is deliver those messages.**
Protocol: Uses SPI to receive data from microcontroller and I2C for sending data to the microcontroller	**These values are determined by the screen we chose to use. The screen takes its inputs from an SPI connection and outputs data from the touch screen over I2C.**	**The ESP32-C3 datasheet lists SPI and I2C in the "peripherals and sensors" section**

Datarate: frequency of 80MHz for SPI connection and 100Kbit/s for the I2C connection	**These data rates were determined by what the ESP32-C3 is capable of**	**These are listed in the ESP32-C3 datasheet as the frequency for SPI in master mode, which we will be using since it's the microcontroller sending data to the screen, and the standard mode data rate for its I2c interface.**

prphrls_mcrcntrllr_dsig : Input

Logic-Level: Active high	**The digital pins will be pulled to high (vcc) to activate the peripherals.**	**This is based on the design of the peripherals block.**
Vmax: 3.6V	**This is determined by the voltage range of both the microcontroller and the power supply.**	**This is the listed max input voltage listed in the ESP32-C3 datasheet in the "DC Characteristics" section.**
Vrange: 0-3.3V	**This is determined by the voltage range of both the microcontroller and the power supply.**	**This range is in the ESP32-C3 datasheet in the "DC Characteristics" section. Additionally, the power supply will provide this range of voltages.**

bttry_mcrcntrllr_asig : Input

Vmax: 0.750 V	**We selected attenuation mode 0, which will give the most accurate readings. This mode can read voltages in a range from 0-.75 V.**	**The information about the ADC's attenuation modes is listed in the ESP32-C3 datasheet in the "ADC Calibration Results" section.**
Vrange: 0.0V - 0.750V	**We selected attenuation mode 0, which will give the most accurate readings. This mode can read voltages in a range from 0-.75 V.**	**The information about the ADC's attenuation modes is listed in the ESP32-C3 datasheet in the "ADC Calibration Results" section.**
Sampling Rate: 100 kSPS	**This is the sampling rate listed in the ADC section of the ESP32-C3 datasheet**	**This is the sampling rate listed in the ADC section of the ESP32-C3 datasheet**

bttry_mcrcntrllr_dcpwr : Input

Inominal: 30mA	**We will use “modem sleep” mode to disable the wireless functions and dramatically reduce the current the micro controller draws.**	**The ESP32-C3 datasheet lists the typical current of modem sleep as 28mA when the CPU is running at 160MHz.**
Ipeak: 500mA	**This is the recommended operation current when all the wireless components are running.**	**The ESP32-C3 datasheet lists the recommended current as .5 A**
Vmax: 3.6V	**This value is determined by the range listed in the ESP32-C3 datasheet in the “Recommended Operating Conditions” section.**	**This value is determined by the range listed in the ESP32-C3 datasheet in the “Recommended Operating Conditions” section.**
Vmin: 3V	**This value is determined by the range listed in the ESP32-C3 datasheet in the “Recommended Operating Conditions” section.**	**This value is determined by the range listed in the ESP32-C3 datasheet in the “Recommended Operating Conditions” section.**

4.6.5 Verification Process

mcrcntrllr_cd_data

1. Connect microcontroller and screen to power
2. Connect microcontroller and screen interfaces
3. Load test code into microcontroller
4. Run test program showing input and output

bttry_mcrcntrllr_asig

1. Connect microcontroller to a bench-top power supply
2. Connect ADC to separate bench-top power supply
3. Load test code onto microcontroller and establish serial connection with laptop
4. Adjust voltage of the power supply connected to the ADC
5. Transmit ADC data over serial to display on the laptop

It will be relatively easy to use a serial connection to display the measured voltage on the laptop rather than giving the microcontroller its own means of displaying that data. Using a PWM signal to vary the brightness of an LED was considered, but was deemed too imprecise.

bttry_mcrcntrllr_dcpwr

1. Connect microcontroller to bench-top power supply
2. Load test code onto microcontroller and establish serial connection with laptop
3. Adjust voltage of power supply
4. Transmit lopping data over serial to show the microcontroller is operating at various voltages

The current being supplied by the bench-top power supply can be double checked with a multimeter.

Prphrls_mcrcntrllr_asig

1. Connect microcontroller to bench-top power supply
2. Load test code onto microcontroller and establish serial connection with laptop
3. Connect test transistor and LED to test pins
4. Have the controller blink the LED and monitor current through the transistor's base using a multimeter

The test code could simply turn the led off or on at regular intervals, or the state of the GPIO pins could be controlled from the laptop over a serial connection.

4.6.6 References and File Links

"ESP32C3 series Datasheet," *ESP32C3 series - espressif*. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf. [Accessed: 11-Feb-2023].

SCH ESP32-C3-devkitm-1 v1 20210 - espressif. [Online]. Available: https://dl.espressif.com/dl/schematics/SCH_ESP32-C3-DEVKITM-1_V1_20200915A.pdf. [Accessed: 11-Feb-2023].

A. Industries, "2.8' TFT display - 240X320 with capacitive touchscreen," *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/2770. [Accessed: 11-Feb-2023].

4.6.7 Revision Table

1/20/2023	**Rory Patterson: write first draft**

Group 9 Yoga Timer Project Document
Craig Brod, Bryan Gaona Herrera, Rory Patterson

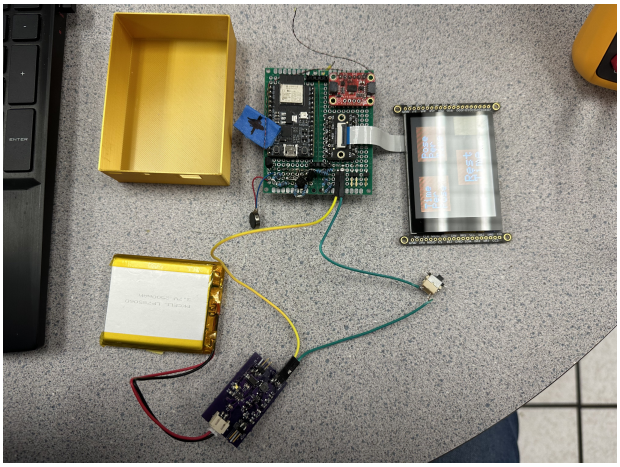
2/10/2023	**Rory Patterson: revise first draft**

5 System Verification Evidence

5.1 Universal Constraints

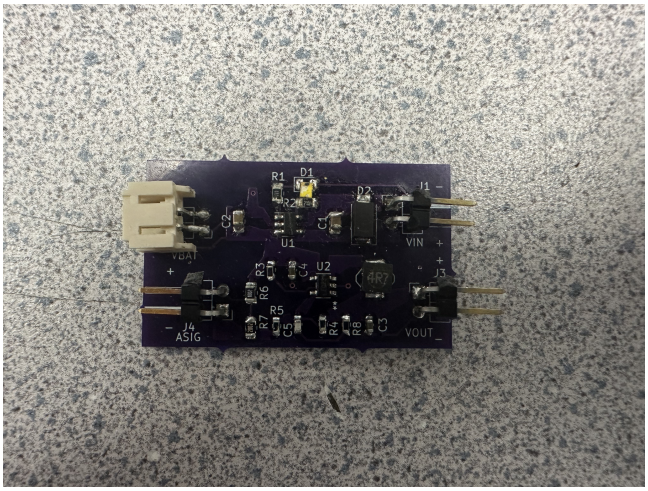
5.1.1 The system may not include a breadboard.

Our project uses a student made PCB and a protoboard for implementation. Originally we were going to use two student made PCBs but due to issues with the second PCB and time constraints, we decided to use a protoboard for a working product. Pictured below is the system.



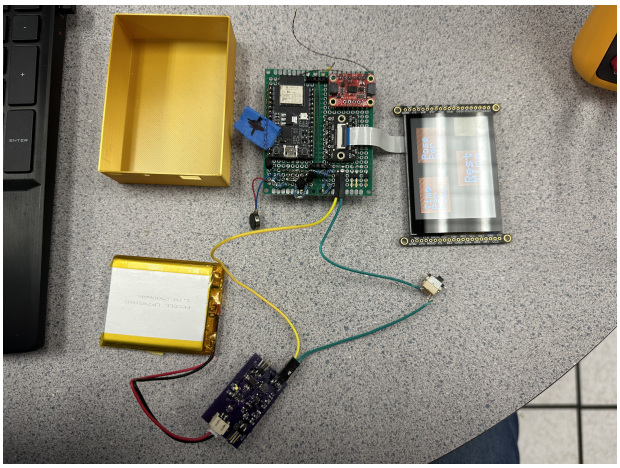
5.1.2 The final system must contain a student designed PCB.

Our system makes use of a modified version of the LiPo PCB that we designed in ECE 441. The board that we designed has 51 pads. Pictured below is the board.



5.1.3 All connections to PCBs must use connectors.

Our system has four main circuitry components. The main board, the battery board, the battery, and the screen. The battery is connected to the battery board via a connector, the battery board is connected to the main board via jumper wires, and the screen is connected to the main board via a ribbon cable. Pictured below is the system with the connections highlighted.



5.1.4 All power supplies in the system must be at least 65% efficient.

Buck Converter Chip Datasheet:
https://ww1.microchip.com/downloads/en/DeviceDoc/22042B.pdf

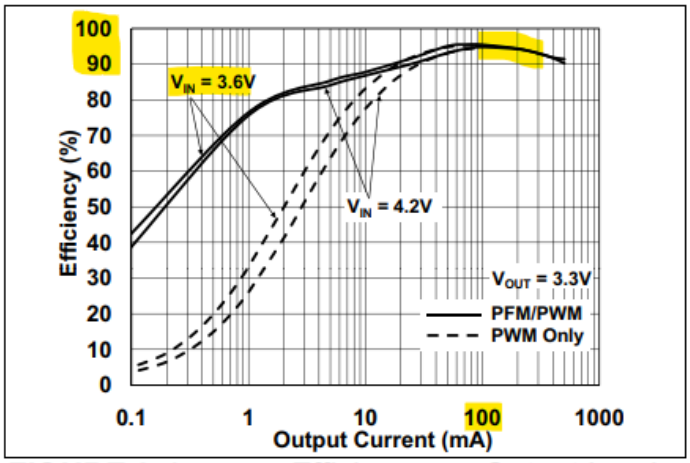


FIGURE 2-14: *Efficiency vs. Output Load (V_{OUT} = 3.3V).*

Figure 1: MCP1603 Efficiency vs. Output Current Graph

Our power supply is using the MCP1603 Chip for our Buck Converter. This connects the battery as an input and the rest of the system as our load. Our battery sits around 3.6V and the output sits around 3.3V. Going off of the corresponding line for those values, we can see that for loads between 100mA and 300mA, which we are expecting our system to stay within, The efficiency is 90-100%. This is well above the 65% required efficiency for our power supply.

5.1.5 The system may be no more than 50% built from purchased 'modules.'

Block	Built/Modified or Bought with reason
Enclosure	Built The enclosure has been designed and 3D printed by us.
Microcontroller	Bought We used the ESP32-C3 development board for our microcontroller.
Screen	Built The screen is originally a bought module but needed to be modified to fit our design.
Code	Built Other than the use of libraries provided by Adafruit, the code is completely written by us.
Peripherals	Built The peripherals are soldering into the main board.
Battery	Built The battery block is built due to it containing our own designed PCB and then connects the battery to it.

Total percentage built from purchased modules is 16.7%.

5.2 Requirements

5.2.1 Battery Operated

5.2.1.1 Project Partner Requirement

The unit must be battery powered and rechargeable.

5.2.1.2 Engineering Requirement

The system will operate for up to 2 days in standby mode on a single charge with 5 hours of active use.

5.2.1.3 Verification Process

1) While the system is on, measure the current that is exiting the battery block using a DMM. Write down the nominal current.
2) Turn the system off, measure the current that is exiting the battery block using a DMM. Write down the nominal current.
3) Calculate the total hours that could be used in each mode based on the 2500 mAh battery.

5.2.1.4 Testing Evidence

While active, our device is pulling 109 mA nominal and while off 0 mA. Our battery has 2500 mAh so therefore we are getting 22 hours while active and more than 5 days while inactive.
Video of the test itself: https://youtu.be/nGgXI287y8o

5.2.2 Custom Timers

5.2.2.1 Project Partner Requirement

The unit must be able to accurately run multiple timers that are based off of user input.

5.2.2.2 Engineering Requirement

The system will run at least 2 timers at the same time that will be defined by user input and be accurate to 0.1 seconds.

5.2.2.3 Verification Process

1) Power on the system.
2) Select rest time and input 0 seconds.
3) Confirm the selection.
4) Select timers by poses.
5) Input 10 seconds per pose.
6) Input 10 poses for the session.
7) Start a stopwatch at the same time as the timer starts.
8) Stop the stopwatch when the final timer stops.

5.2.2.4 Testing Evidence

With the addition of human error in reaction, our stopwatch should be within 99.5 to 100.5 seconds. From our test, we found that our timer matched with 99.62 seconds.

Video test: https://youtu.be/TXOZ6V5TFxY

5.2.3 Discrete and Indiscrete

5.2.3.1 Project Partner Requirement

The unit must allow the user to select between a discrete and indiscreet modes.

5.2.3.2 Engineering Requirement

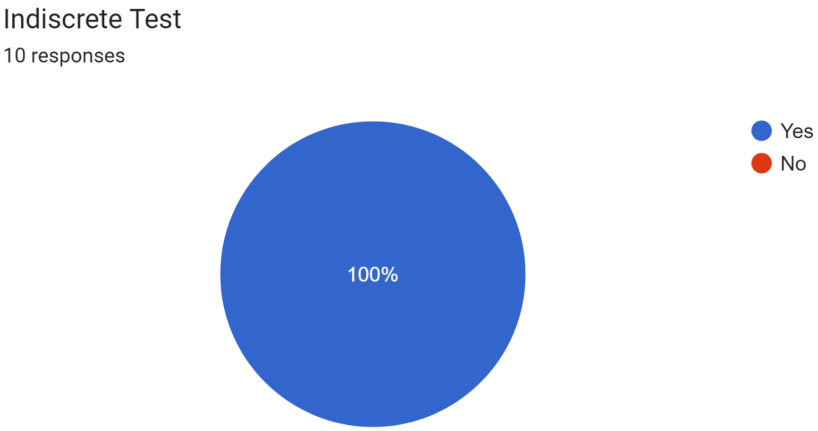
The system will have an indiscrete mode that will use a visual and haptic indicator that will be noticeable up to 20 feet away, and a discrete mode that has a low intensity visual indicator that is noticeable up to 5 feet by 9 out of 10 users.

5.2.3.3 Verification Process

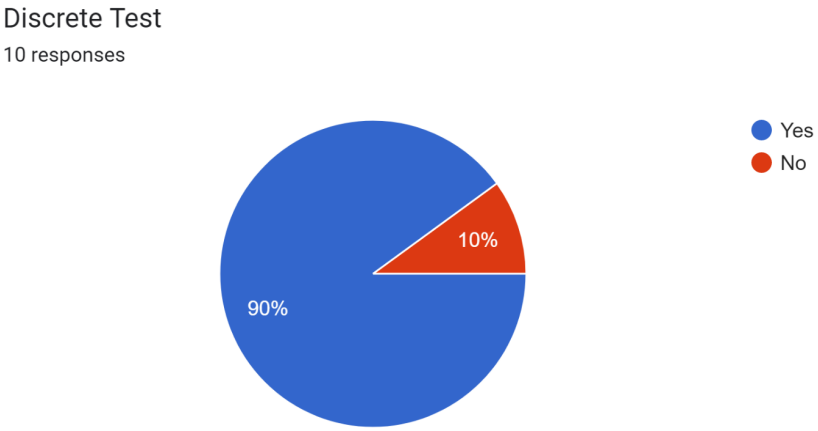
1) Place the system 20 feet away from the user in indiscrete mode.
2) Select timers by pose.
3) Input 10 seconds per pose.
4) Input 1 pose.
5) Let the timer finish.
6) Move the system to be 5 feet from the user and set it to discrete mode.
7) Repeat steps 2 through 5
8) Have the user fill out a google form that asks them 2 questions on whether the two modes were noticeable.

5.2.3.4 Testing Evidence

Below are the test results of the indiscrete test.
Video: https://youtu.be/DkzZxRot5oU



Below are the results of the discrete test.
Video: https://www.youtube.com/watch?v=r3mB0SJVCC0



5.2.4 Friendly UI

5.2.4.1 Project Partner Requirement

The unit must have a UI that is user friendly.

5.2.4.2 Engineering Requirement

The system shall be considered easy to use by 9 out of 10 users with only basic written instructions.

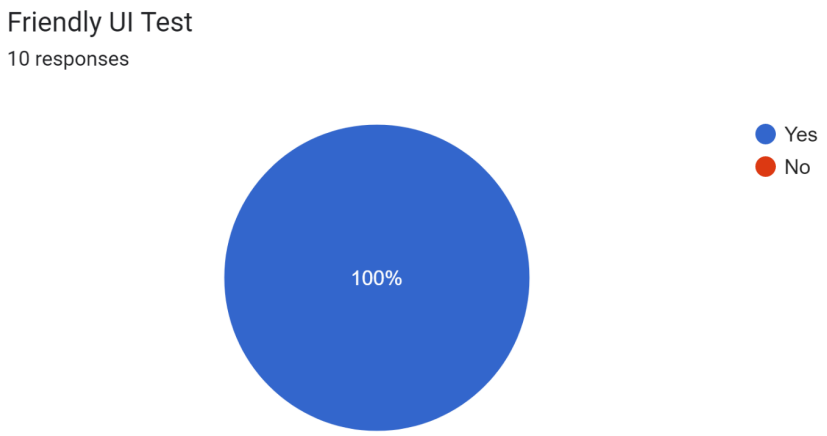
5.2.4.3 Verification Process

1) Give the system to a user along with a short instruction list that gives a quick run through of the system.

2) Have the user follow the instruction list.
3) Have the user fill out a google form that asks if the system UI (user interface) is easy to use.

5.2.4.4 Testing Evidence

Link to test process video:
https://youtu.be/K8xlnqLd8C0



5.2.5 Lightweight

5.2.5.1 Project Partner Requirement

The unit must be lightweight.

5.2.5.2 Engineering Requirement

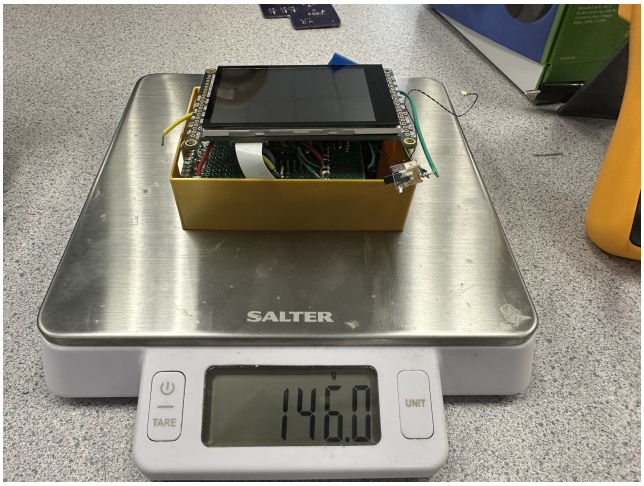
The system shall weigh less than 300 grams.

5.2.5.3 Verification Process

1) Weigh the entire system all at once on a scale in grams.

5.2.5.4 Testing Evidence

Our system weighs 146.0 grams. Well within our threshold.



5.2.6 Pedometer

5.2.6.1 Project Partner Requirement

The unit must be able to track the steps of the wearer.

5.2.6.2 Engineering Requirement

The system shall report the user’s steps and be accurate within 10% every 100 steps.

5.2.6.3 Verification Process

1) Attach the device to the side of a user's pants or their belt.
2) Have the user walk 100 steps.
3) Check the number of steps on the device.

5.2.6.4 Testing Evidence

We are not passing this requirement.

5.2.7 Size

5.2.7.1 Project Partner Requirement

The system must be small.

5.2.7.2 Engineering Requirement

The system shall be no bigger than 2.5” x 3.5” x 1”.

5.2.7.3 Verification Process

1) Measure the dimensions of the enclosure that the system is contained within.

5.2.7.4 Testing Evidence

We are not passing this requirement.

5.2.8 Touch Screen

5.2.8.1 Project Partner Requirement

The unit must have a capacitive touch screen as its main I/O.

5.2.8.2 Engineering Requirement

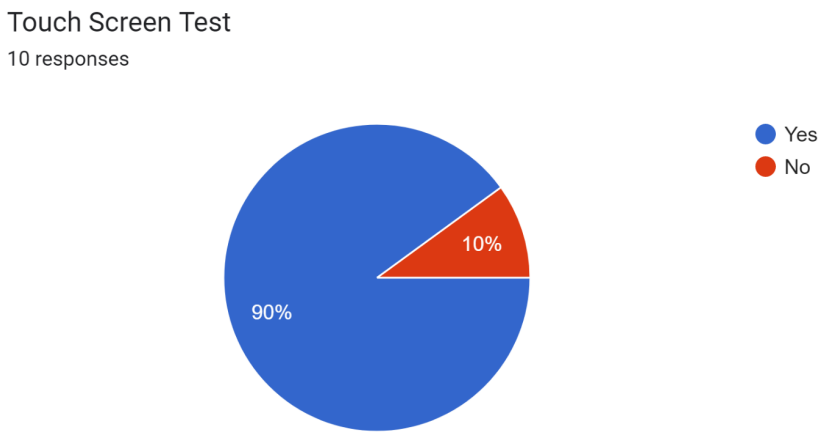
The system will feature a touch interface and use it to display timers and to be considered easy to use by 9 out of 10 users with only basic written instructions.

5.2.8.3 Verification Process

1) Give the system to a user along with a short instruction list that gives a quick run through of the system.
2) Have the user follow the instruction list.
3) Have the user fill out a piece of paper that asks if the system is easy to understand.

5.2.8.4 Testing Evidence

Link to test process video:
https://youtu.be/K8xlnqLd8C0



5.3 References and File Links

5.3.1 References

5.3.2 File Links

Add links to videos

5.4 Revision Table

3/12/2023 - Craig	Section created and subsections populated from other project documentation.
3/14/2023	Test results for two requirements added

6 Project Closing

6.1 Future Recommendations

6.1.1 Technical recommendations

1) The code base can be severely cut down. The microcontroller has limited memory and the library [1] that controls the screen takes up most of it.
2) The power control electronics could be incorporated into the main PCB [2] . We chose to build them separately to make building and testing easier, the connectors and wires alone take up unnecessary space.
3) As a product it would be faster and cheaper on a large scale to use injection molding instead of 3D printing [3].
4) The buck converter was very difficult to get working. It would be easier and more efficient to buy one premade [4].

6.1.2 Global impact recommendations

1) Try to source parts from as few locations as possible, it cuts down on order costs and makes logistics much easier to track. We wound up ordering form: mouser, digikey, adafruit, and some parts from tekbots. Fewer sources reduce the number of shipments being made, which is both cost effective [5] and less environmentally damaging.
2) Try to source cheap parts. We encountered a screen with a control module that was cheaper than an equivalent screen without any control board [6]. By

using cheaper parts the overall cost of the finished product will be lower; making it more accessible to users.

6.1.3 Teamwork recommendations

1) Use dedicated team management software to track communication and documents. It seems easy to use a discord server and a google drive, but those both get messy fast. Consider: slack, trello, and github [7].
2) Keep some kind of meeting notes [8]. Even if they're just a list of the subjects, it's better than nothing. And the act of writing information down helps you remember it.

6.2 Project Artifact Summary

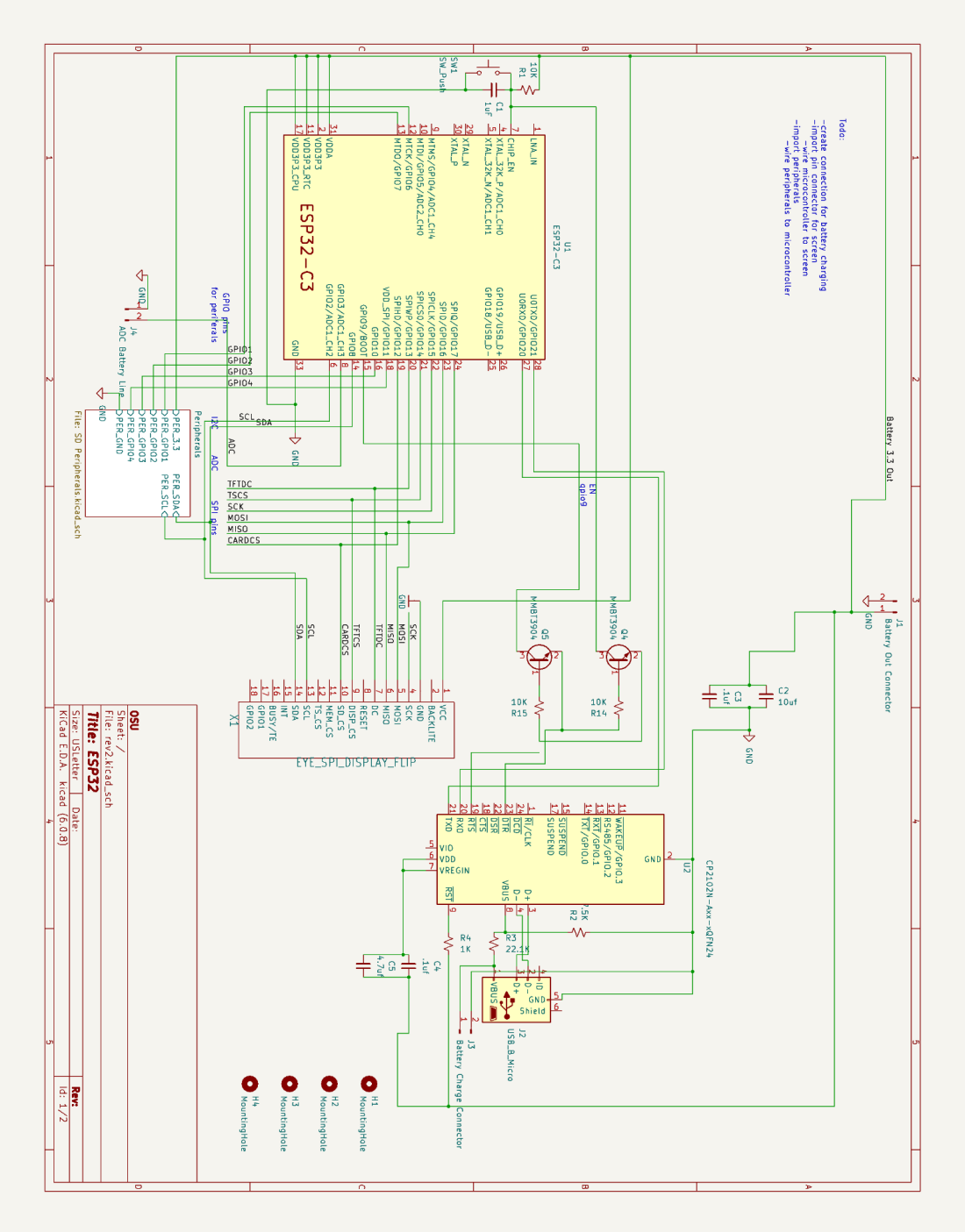


Figure 1: Main board schematic

This is the schematic for the main PCB board in our system. It houses the microcontroller, peripherals, connectors to other blocks, and associated circuitry. The peripherals connections are hidden in a sub-schematic.
https://drive.google.com/file/d/1K6CrAgWpSbq3BrCvrxzNi6s0Kl6bVm1d/view?usp=share_link

Group 9 Yoga Timer Project Document
Craig Brod, Bryan Gaona Herrera, Rory Patterson

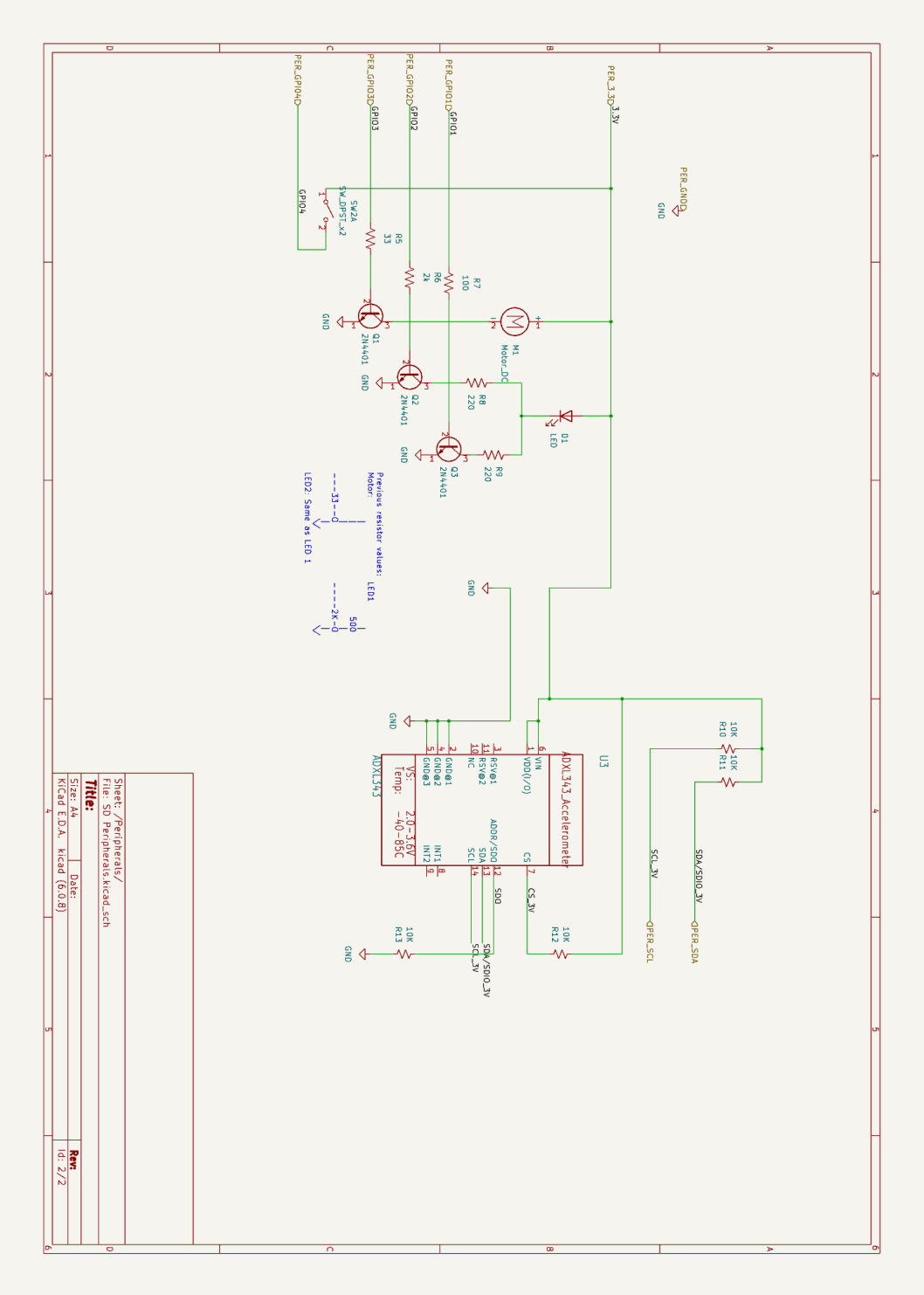


Figure 2: Sub Schematic of main board, SD Peripherals

This is a schematic showing how peripherals components are connected. This schematic contains the accelerometer, LED, motor, switch, and necessary transistors.https://drive.google.com/file/d/1cVJUMEeYp7KYIbKsmot5TQAKJGGp8nRU/view?usp=share_link

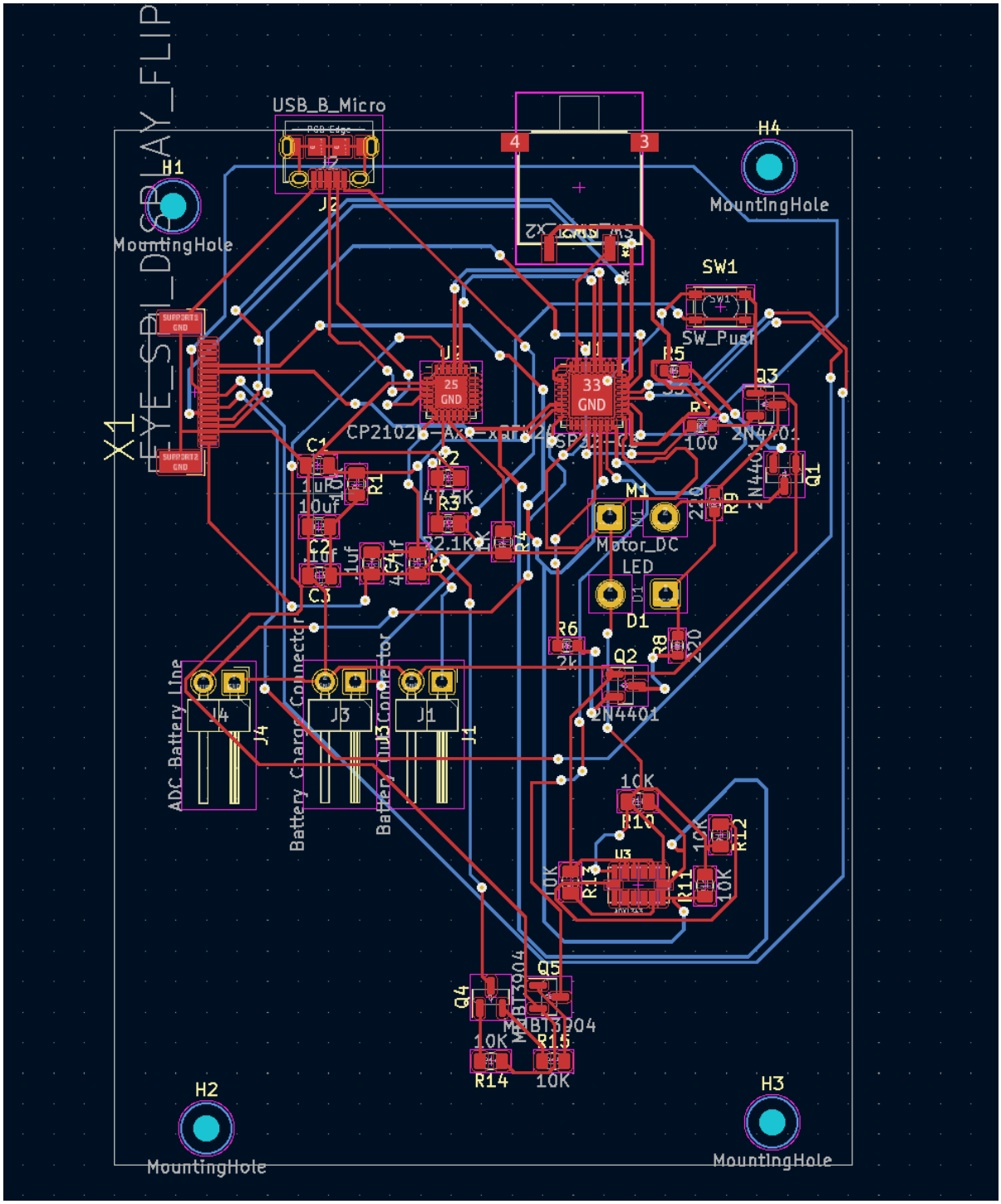


Figure 3: Main Board PCB

This is an image of the main PCB in our system housing the microcontroller, peripherals, connectors to other blocks, and associated circuitry. This is the hub of our system and is shaped to fit tightly within our enclosure.https://drive.google.com/file/d/1DlW6ceX6ywXF92561JppvauarW77Z-Ma/view?usp=share_link

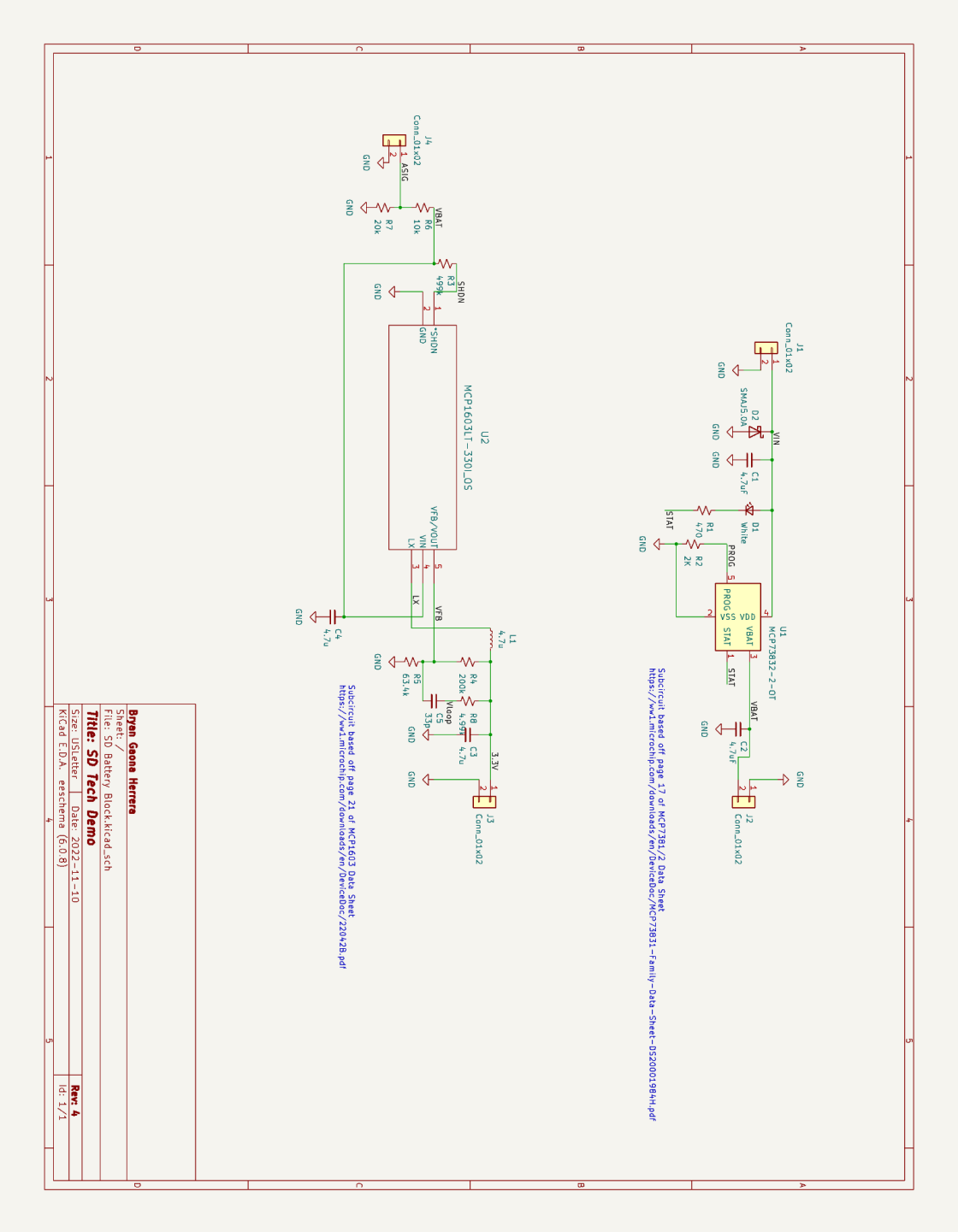


Figure 4: Battery Block Schematic

This is the schematic for the power supply PCB in our system. It contains the charge controller components (on the top) that control power going into the battery during charging, and it contains buck converter components (on the bottom) that ensure a 3.3V output goes to the rest of the system.

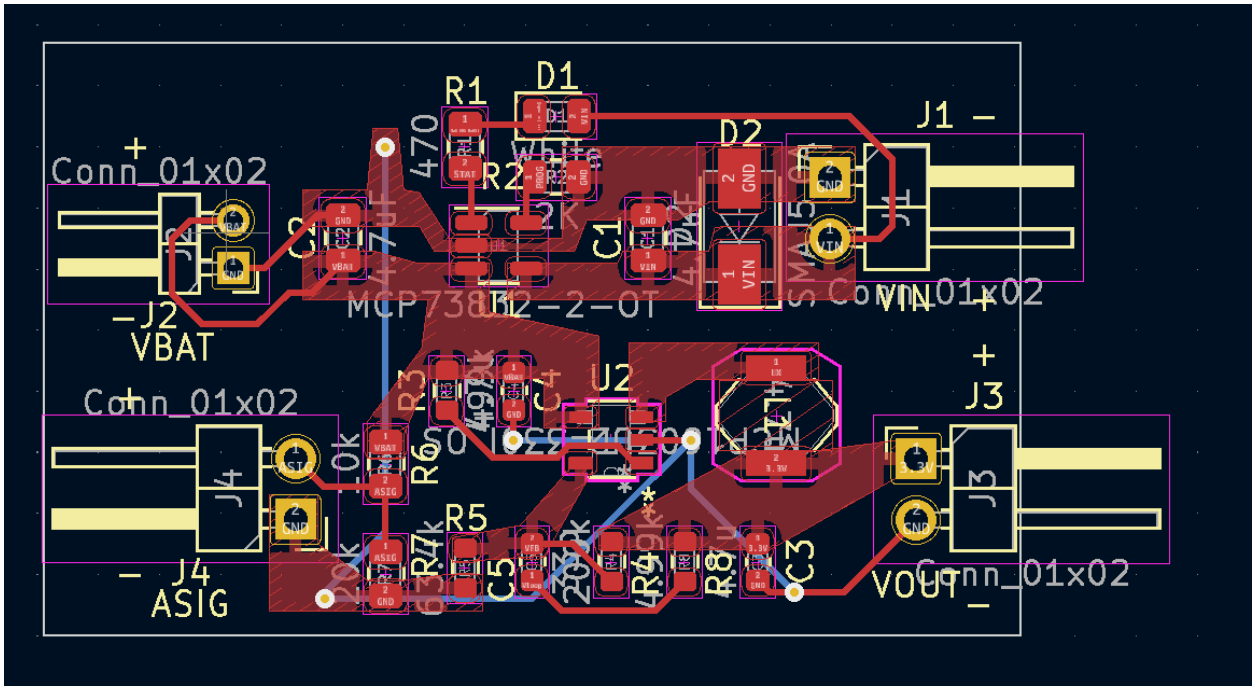


Figure 5: Battery Block PCB

This is an image of the power supply PCB in our system. Charge controller components are on the top half while buck converter components are in the bottom half. There are also connectors to the main PCB board and the actual battery itself.

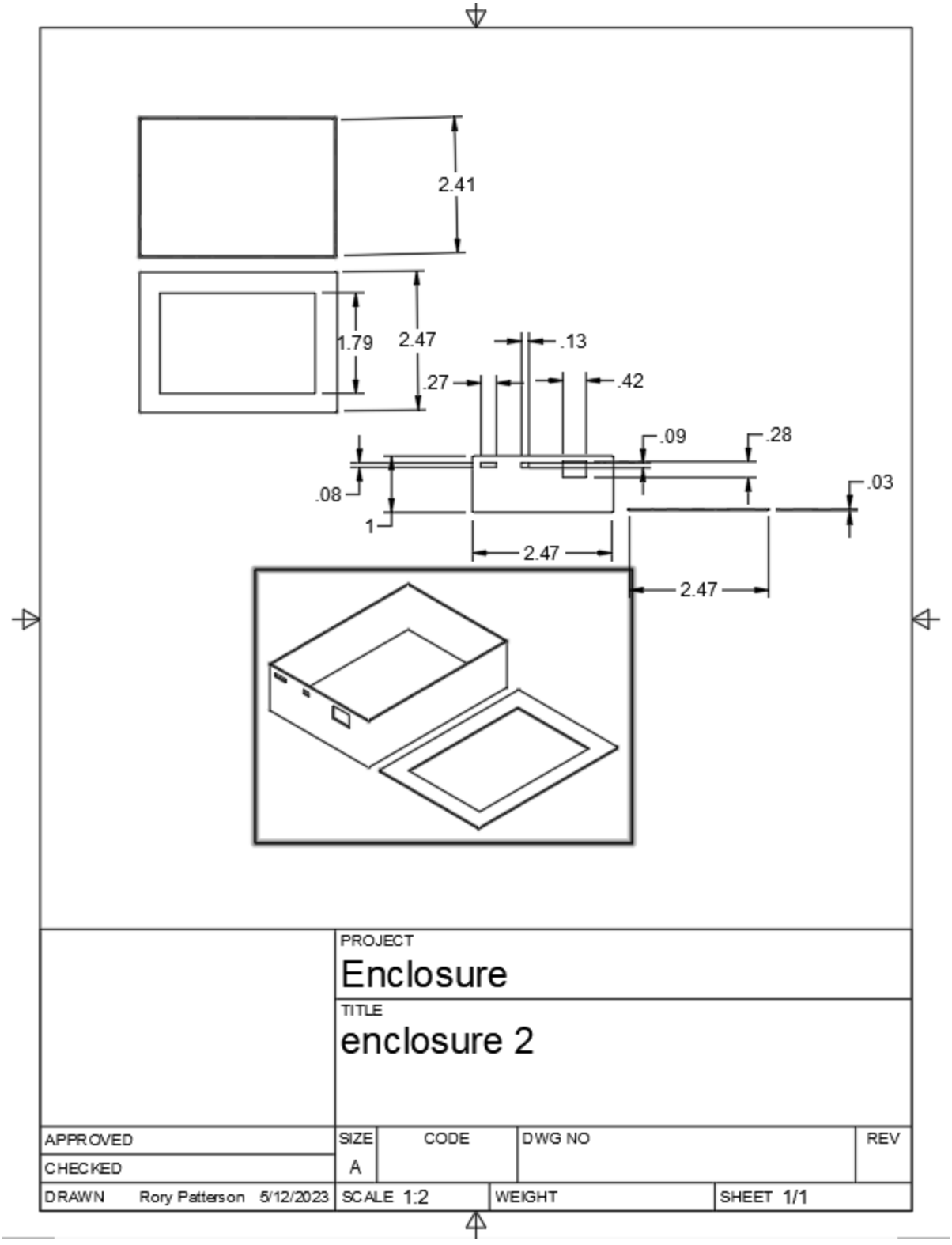


Figure 6: technical drawing of enclosure

This is the mechanical drawing for our enclosure. Dimensions of the enclosure are given in inches.

Microcontroller Code:
https://github.com/blitzschwert/YogaTimer

This is the code used in our system to design the timers and GUI. It works with our microcontroller to communicate with the screen and operate the peripherals at the appropriate times.

6.3 Presentation Materials

Project Poster Expo Project Poster.pdf

Project Showcase Site:
https://eecs.engineering.oregonstate.edu/project-showcase/projects/?id=A6FILm7g0rDGkC4h

6.4 References and File Links

[1] Adafruit, “Adafruit/Adafruit-GFX-Library: Adafruit GFX Graphics Core Arduino Library, this is the ‘core’ class that all our other graphics libraries derive from,” GitHub, https://github.com/adafruit/Adafruit-GFX-Library (accessed May 11, 2023).

[2] Rob, “What are the advantages of smaller PCBS? – clarydon blog,” Clarydon Electronic Services, https://www.clarydon.com/advantages-smaller-pcbs (accessed May 11, 2023).

[3] “What is the 3D printing vs injection molding cost-per-unit breakeven?,” Xometrys RSS, https://www.xometry.com/resources/injection-molding/injection-molding-vs-3d-printing/ (accessed May 11, 2023).

[4] “MCP1603EV - Microchip Technology,” MCP1603 BUCK CONVERTER EVALUATION BOARD, https://www.microchip.com/en-us/development-tool/MCP1603EV (accessed May 11, 2023).

[5] G. Miller, “As Asia-US shipping rates rise, so does skepticism on staying power,” FreightWaves, https://www.freightwaves.com/news/trans-pacific-spot-shipping-rates-are-rising-and-so-is-skepticism (accessed May 11, 2023).

[6] A. Industries, “2.8" Tft lcd with Cap touch breakout board W/microsd socket,” adafruit industries blog RSS, https://www.adafruit.com/product/2090 (accessed May 11, 2023).

[7] CodeClouds, “The Pros and cons of using github for repository management,” CodeClouds, https://www.codeclouds.com/blog/advantages-disadvantages-using-github/ (accessed May 11, 2023).

[8] N. Kahansky, “The difference between meeting notes and meeting minutes (and how to write them),” Hypercontext, https://hypercontext.com/blog/meetings/meeting-notes-vs-meeting-minutes (accessed May 11, 2023).

6.5 Revision Table

4/28/23 Bryan, Craig, Rory	Creation of section. Recommendations filled out. Project artifacts put in.
5/11/2023 Bryan, Craig Rory	More artifacts inserted. Descriptions added to artifacts. References section created and references included.
5/12/2023 Bryan, Craig, Rory	Project artifacts finalized. Revision table section created. Poster Picture finalized.