Battery-Free Handheld Game

DESIGN DOCUMENT

Team Number: 18
Client: Henry Duwe
Adviser: Henry Duwe
John Brose, Chief Engineer of Power Systems
Jake Larimore, Chief Engineer of Hardware Integration
Franklin Bates, Chief Engineer of Microcontrollers
Shivam Vashi, Chief Engineer of Software
Daniel Lamar, Test Engineer
Team Email: jbrose@iastate.edu
Team Website: https://sddec21-18.sd.ece.iastate.edu/

Revised: 12-6-2021 / v4

Executive Summary

The goal of our project is to develop a handheld battery-free gaming device that provides users with fun, enjoyable experience while utilizing energy harvesting to supply the energy required for operation. The device will provide users with a unique experience of interacting with the game in a manner that supplies energy in both a single and multi device environment.

Development Standards & Practices Used

- Agile software development
- IEEE 12207-1996 - ISO/IEC International Standard - Information Technology - Software Life Cycle Processes
- IEEE C63.5-2019 - American National Standard for Electromagnetic Compatibility--Radiated Emission Measurements in Electromagnetic Interference (EMI) Control - Calibration and Qualification of Antennas (9 kHz to 40 GHz) - Corrigendum 1

Summary of Requirements

- The device shall run with power being supplied solely from energy harvesting.
- The energy harvesting shall be a part of the gameplay.
- The device shall support some form of multiplayer functionality.
- The device shall be as portable as a standard laptop.
- The game shall be reasonably intuitive to pick up and play by anyone who has played a game before.
- The game shall run through at least one room challenge state after energy harvesting.
- The device shall show a low power screen when it is running out of battery, telling the user to energy harvest.

Applicable Courses from Iowa State University Curriculum

EE 201, EE 230, EE 330, EE 333, CPRE 288, CPRE 488, CPRE 388, COMS 309, COMS 319

New Skills/Knowledge acquired that was not taught in courses

- Schematic design in KiCad
- PCB design in KiCad
- C++ best practices
- Experience with Code Composer Studio
- Interdisciplinary Collaboration

Table of Contents

List of figures

1 Introduction

1.1 Acknowledgement

Professor Henry Duwe was our technical advisor for this project, providing us with technical knowledge and access to labs.

Vishak Narayanan, a graduate student at Iowa State University, provided us with guidance and advice on energy harvesting systems.

1.2 Problem and Project Statement

Design a gaming device that does not have a battery component. The device will use power harvested from the user. Other features include incorporating energy consumption into gameplay and utilizing multiplayer functionality in certain elements of the game.

Batteries are an easy solution to powering devices that cannot have a steady supply of generated power at hand. Although, as new technology is emerging, the downsides to batteries is becoming more and more significant. Batteries can only hold a limited amount of energy, and will either need to be recharged or disposed of. They also take up significant space, have a limited lifespan, and can be expensive depending on the application. Battery-free devices are becoming increasingly common in the industry, and our gaming device is taking another step in this direction. This is a proof of concept that will help further the knowledge in this field and help show if a battery-free device is a viable alternative to a battery powered device.

1.3 Operational Environment

The handheld game will be used in an indoor setting with a regulated temperature and calm weather conditions. It will be able to withstand reasonable wear and tear while being played by the user.

1.4 Requirements

The device shall run with power being supplied solely from energy harvesting.
The energy harvesting shall be a part of the gameplay.
The device shall support some form of multiplayer functionality.
The device shall be as portable as a standard laptop.
The game shall be reasonably intuitive to pick up and play by anyone who has played a game before.
The game shall run through at least one room challenge state after energy harvesting.
The device shall show a low power screen when it is running out of battery, telling the user to energy harvest.

1.5 Intended Users and Uses

The intended use of the project is to have a device that can provide the user a brief, entertaining multiplayer experience. The intended audience for this device is primarily college faculty, staff, and students.

1.6 Assumptions and Limitations

Assumptions:
The users will be reasonably careful with the device.
The users will understand the basic operations of a dungeon crawler game.
The users will be physically able to operate the energy harvesting devices.

Limitations:
The game must have interaction between at least two devices.
The device must have energy harvesting as the main source of power.
The game must feature energy harvesting as a part of the gameplay.
The device must be able to be hand held.

1.7 Expected End Product and Deliverables

The expected end product will be a device that includes an E-ink screen, energy harvesting push buttons, dip switches, and a hand crank generator all into a small handheld unit. The energy harvesting push buttons along with the hand crank generator will provide the entirety of the power needed to run the game, and will be incorporated into the gameplay. The push buttons, dip switches, and hand crank generator will form the core tactile interaction with the user while the E-ink screen will provide visual cues and video game information.

This device will allow you to play a dungeon crawler game that features energy harvesting elements and multiplayer. The main gameplay loop will be switching between players deciding between rooms to explore, to challenges in those rooms the players need to complete. If the device reaches a low power state, we will store the game's state, including things like the player's stats and any rooms that have been completed, into the device's memory, and we will then restore the game state from the memory once the device has been sufficiently powered by energy harvesting.

2 Project Plan

2.1 Task Decomposition

1. Gameplay Design:
    a. Define the category of game that will be designed.
    b. List out the required aspects of the game play.
    c. Set “stretch goals” to implement once required goals are completed.
2. Energy Harvesting:
    a. Research the resources available to generate power for the gaming device.
    b. Decide how the user will operate the energy harvesting components to interact with the gameplay.
    c. Measure Part Specifications
    d. Harvesting Detection Design
3. Hardware Design:
    a. Research and determine what microcontroller, display unit, and other hardware elements will be used.
    b. Design Power System
    c. Breadboard isolated blocks for testing
    d. Make PCB
    e. Debug PCB in blocks to ensure all parts are functioning to expected standards
    f. Assemble Final Device Prototype
4. Software Design:
    a. Graphical User Interface for use with E-ink display
    b. Environment generation algorithm
    c. Character creation and motion feature
    d. Room challenges feature
    e. Multiplayer feature
    f. Save/Load state on while on low power or returning from a low power state
5. Microcontroller Interface Design:
    a. E-ink display
    b. Button & RPM detection
    c. Implement low power mode
    d. Implement save states
    e. Energy harvesting detection
    f. Multiple device communication
6. Test Design:
    a. Energy harvesting & power generation
    b. Display & User Interface
    c. User input detection
    d. Environment Algorithm

e. Room Challenge Algorithm
f. Multiplayer gameplay
g. Boss fight logic
h. Saving/loading states saved to fRAM
i. Room Challenge power production produces enough energy for room challenge loop
j. Enough energy stored when boss fight is reached

2.2 Risks And Risk Management/Mitigation

Gameplay Design Risks: 0.1
Energy Harvesting Risks: 0.4
Hardware Design Risks: 0.7

> *Mitigation*: Review the design documents and datasheets for each hardware component to ensure all connections are correct. If so, reevaluate the uses for each component and ensure each component is receiving the right amount of power. If so, consider replacing certain hardware components that seem to be causing the problem.

Software Design Risks: 0.8

> *Mitigation*: Debug all errors in the program. Ensure the error is occurring in the software, not the hardware. Review datasheets for the microcontroller to debug any miscommunication between the hardware and software.

Test Design Risks: 0.6

> *Mitigation*: If the test fails was it due to hardware/software or the design of the test. If test failure was due to the design of the test reevaluate test design. Otherwise, use hardware/software mitigations.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Gameplay Design: The game will be a dungeon crawler that is intuitive to play, and does not lose data or prevent the user from continuing during it's full gameplay cycle.

Energy Harvesting: The users will be able to generate 100% of the necessary power for the proceeding turn by completing the current task. The device will turn off once it loses the necessary power to remain on. The user will be able to generate enough energy to turn the device back on in under 5 seconds.

Hardware Design: The hardware will connect the power producing energy harvesting components to the supercapacitor, microprocessor, and E-ink display. This will allow the powering of the device as well as the ability to tell how fast the user is producing power. The hardware will have a constant current consumption in the nanoamp range while not updating the display to minimize power consumption.

Software Design: The software will be able to run the full game from start to finish, generating rooms to give player's stats through challenges, and display all of this to the user through an intuitive user interface. The software should also be able to successfully save and load the game state during low power without any data loss.

Microcontroller Interface Design: A fully charged supercapacitor on the microcontroller will be able to update the e-ink display, and generate a new room once. When the microcontroller is in the “saved state” mode and it receives enough power to resume gameplay, it will reboot to the last saved UI and gameplay state in under one second. Multiplayer communications will occur with less than 500 ms delay.

2.4 Project Timeline/Schedule

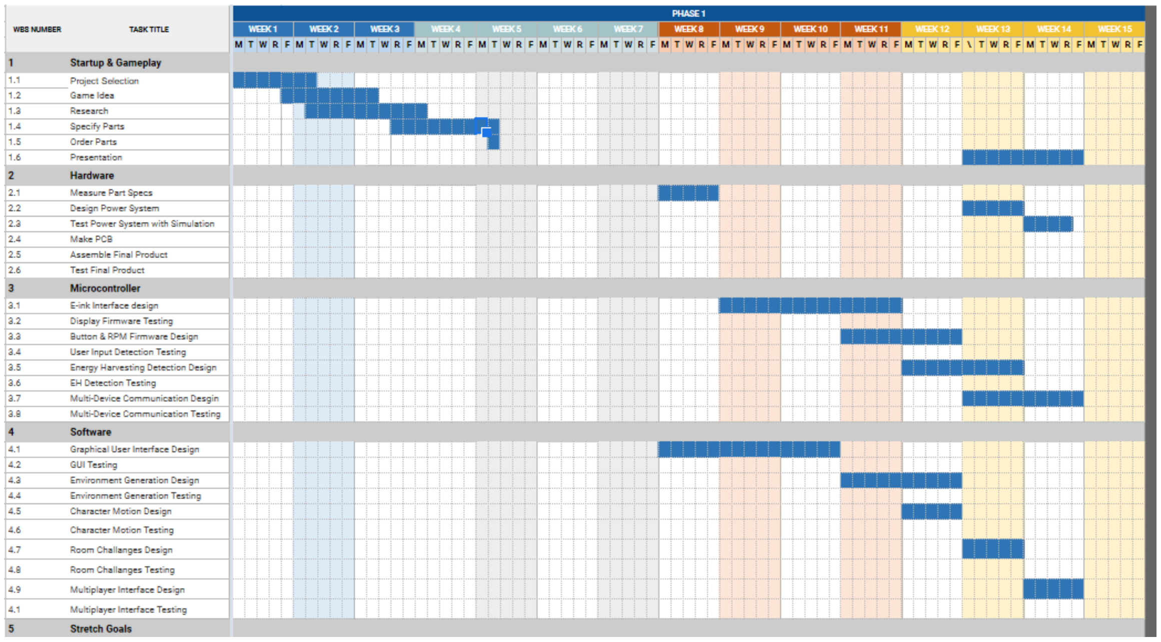


Figure 2.1: Spring 2021 Ghant Chart

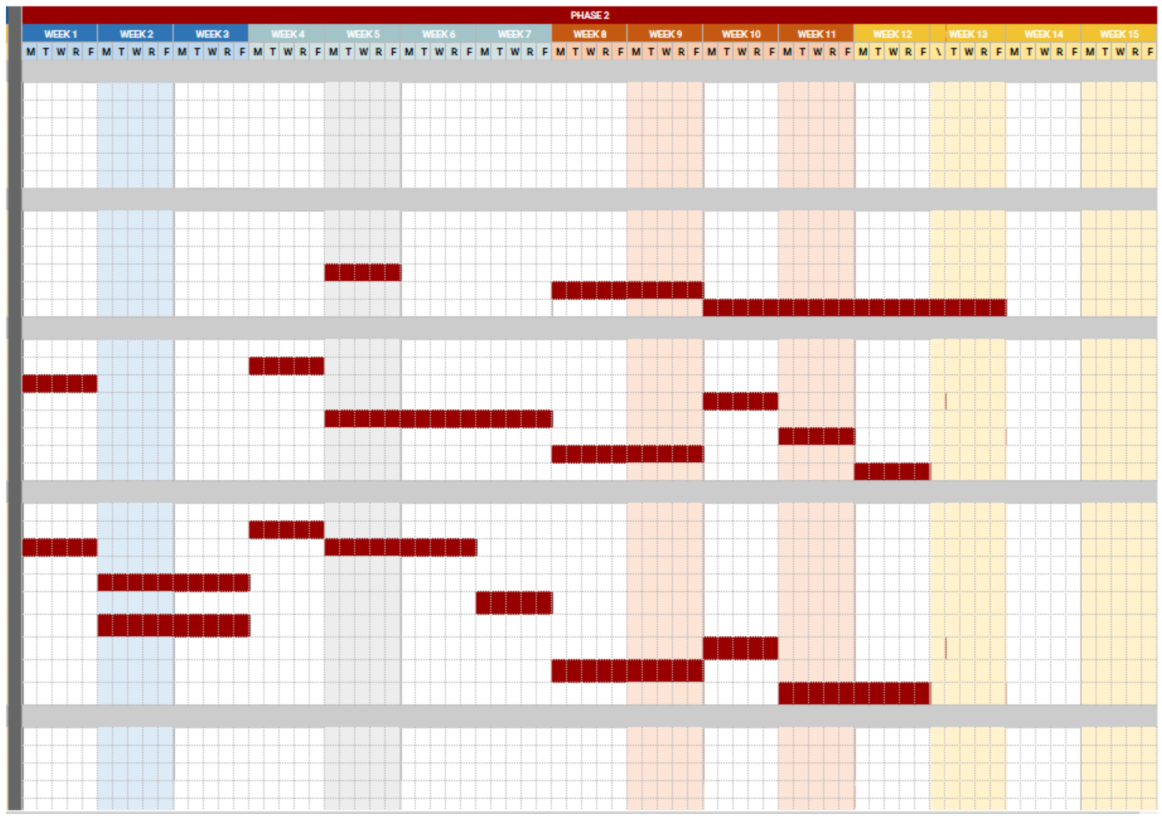


Figure 2.2: Fall 2021 Ghant Chart

2.5 Project Tracking Procedures

The group has used Trello to communicate what tasks need to be completed, what tasks are being completed, and what tasks have been completed. The team also used a Discord server as a means of communication and posting related information. Lastly, Google Drive and Github were used to store files and programs used in the design and documentation of the project. We will continue to use these platforms as the project continues.

2.6 Personnel Effort Requirements

Tasks	# Hours Req	Details
Gameplay Design	5-10 hours	What kind of game we want to create, how it will implement energy harvesting, how it will function.
Energy Harvesting	25-35 hours	Find out how we can collect energy with devices, and how we can

		implement these devices in our overall design.
Hardware Design	30-40 hours	Creation of the schematic connecting microcontroller, circuits, and all devices together. Layout of the PCB based off of the schematic.
Software Design	30-40 hours	Developing GUI and gameplay.
Test Design Software	40-50 hours	Testing gameplay and functionality, debugging, ensuring no errors are found.
Test Design Hardware	40-50 hours	Testing voltages and currents at certain nodes in schematics, measuring power dissipation across components, ensure the capacitor is charging and discharging correctly.

2.7 Other Resource Requirements

Lab equipment (oscilloscope, multimeter, dc power supply), KiCad PCB software, LTSpice simulator, Spectre simulator, solder station, PTC Creo, Code Composer Studio, CLion, 3D printer.

2.8 Financial Requirements

Below are the core components needed to create the batteryless handheld game and their financial breakdown:

Part	**Price**	**Quantity**	**Price for Quantity**
Bottom board buttons	$0.37	4	$1.48
Push Button	$0.15	1	$0.15
32 kHz Crystal	$0.86	1	$0.86
Energy Harvestor Controller	$4.93	1	$4.93

Boost Regulator	$2.94	1	$2.94
MSP430FR5994	$9.79	1	$9.79
Zener Diode	$0.17	1	$0.17
3-phase rectifier	0.62	1	$0.62
4 OR gates	$0.37	1	$0.37
Bridge Rectifier	$1.40	4	$5.60
Hand Crank Connector	$0.12	1	$0.12
100mF Super Cap	$1.96	1	$1.96
Kinetic Button Harvestor	$11.15	4	$44.60
E-ink	$19.50	1	$19.50
Hand Crank Generator	$9.99	1	$9.99
Capactiors			
0.1uF	$0.10		$0.00
10uF	$0.28		$0.00
1nF	$0.11		$0.00
10nF	$0.11		$0.00
22pF	$0.10	2	$0.20
4.7uF	$0.10	1	$0.10
1uF	$0.10		$0.00
Resistors			
10 MOhm	$0.13	10	$1.33
1.69 MOhm	$0.04	10	$0.38
5.62 MOhm	$0.03	10	$0.25
4.53 MOhm	$0.03	10	$0.25
0.39 MOhm	$0.02	10	$0.17
9.76 MOhm	$0.10		$0.00
1.4 MOhm	$0.03	10	$0.25

6.98 MOhm	$0.03	10	$0.25
1 MOhm	$0.10	7	$0.70
180kOhm	$0.02	10	$0.17
47kOhm	$0.02	10	$0.15
Inductors			
22 uH	$1.43	1	$1.43
2.2uF	$0.40	1	$0.40
PCB			
PCB	$8.00	1	$8.00
Stencil	$10	1	$10
Total			$127.11

3 Design

3.1 Previous Work And Literature

The Battery-Free Game Boy is an example of something that comes close to realizing the vision of a batteryless game. However, the batteryless game boy does not have any mechanics which implement energy harvesting as a core aspect of the game, nor are there multiplayer aspects to the game. While this was an interesting exercise in converting a battery powered game into batteryless, our battery-free handheld game uses the lack of batteries and energy harvesting as a core aspect of the game, not as a supplement to normal batteries.

3.2 Design Thinking

One of the biggest limitations on our project is the fact that the game needs to have interaction between multiple devices. Because of this, we need to make sure that the multiplayer aspect of the game is present from the beginning, and that it's engaging. The other limitation is that the energy harvesting needs to be a part of the gameplay, and there needs to be an engaging way to make sure the user performs the energy harvesting.

After figuring out our problem and the limitations on our project, we began to list out several possible game ideas, and everyone brought their own takes on what would be fun and engaging gameplay that involved energy harvesting. One of our first ideas was a dungeon crawler, where players would be tasked with going through a dungeon and defeat a boss at the end. We improved upon this idea by adding challenges to the rooms that require energy harvesting, and making the final battle an RPG battle system. Our next idea was a card game, where two players go head to head with decks, and could earn cards by energy harvesting. We improved upon this idea by adding in a way for the player to interact with their opponent when it was not their turn, using energy harvesting to create more resources for them when it became their turn. Our next idea was a party game, similar to Mario Party, where we would have several minigames that involve energy harvesting with a board game feel to the game. Our last idea was a laser tag game, where users would have light gun-like devices they could tag other people with on campus, charging shots with energy harvesting. After discussing with the others, we also added the idea of objectives to capture around campus, as well as a website to login to track scores. We decided to pick the dungeon crawler as that was the idea most people were interested in.

3.3 Proposed Design

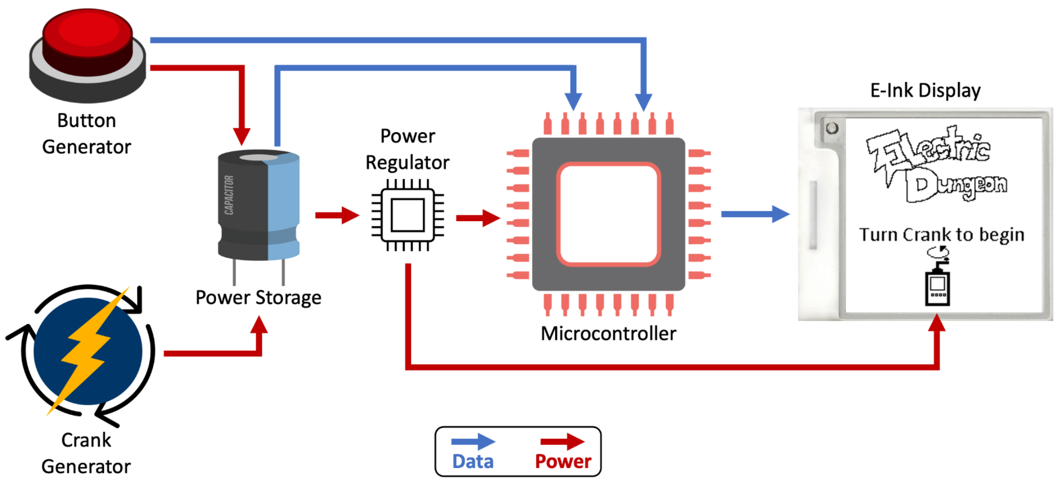


Figure 3.1: Functional Diagram

The design of the hardware consists of five main components shown in *Figure 3.1*: the hand crank, energy harvesting buttons, power management & storage, microcontroller, and E-Ink display. The energy harvesting devices power the microcontroller which enables all aspects of the gameplay to take place. This system also transmits input signals to the microcontroller to be read for gameplay purposes which allows the microcontroller to integrate energy harvesting into the gameplay so that the system is continuously powered as the user progresses through the game (see the *Power System* section for more details on energy harvesting). This system is the sole power source for every component in our design.

3.3.1 Hand Crank

The hand crank itself is a simple brushless motor with 3 phase power. The standalone motor was not appealing to turn on its own, so our team designed a handle with gear implementation to make cranking easier while also maximizing power generation from the component (See section 4 for picture of crank). With the modifications we made our team was able to meet the expectations we had for this device in being the backbone of our power supply. Figure 3.2 shows the schematic we have for this device. The hand crank produces 3-phase power when our system can only accept positive voltages, so our design includes a 3-phase rectication circuit using diodes so that all voltages output from the hand crank come in the form of a positive wave. This is not shown in Figure 3 but we do have a zener diode next to the power IC to prevent hight voltage from the crank damaging the power IC.

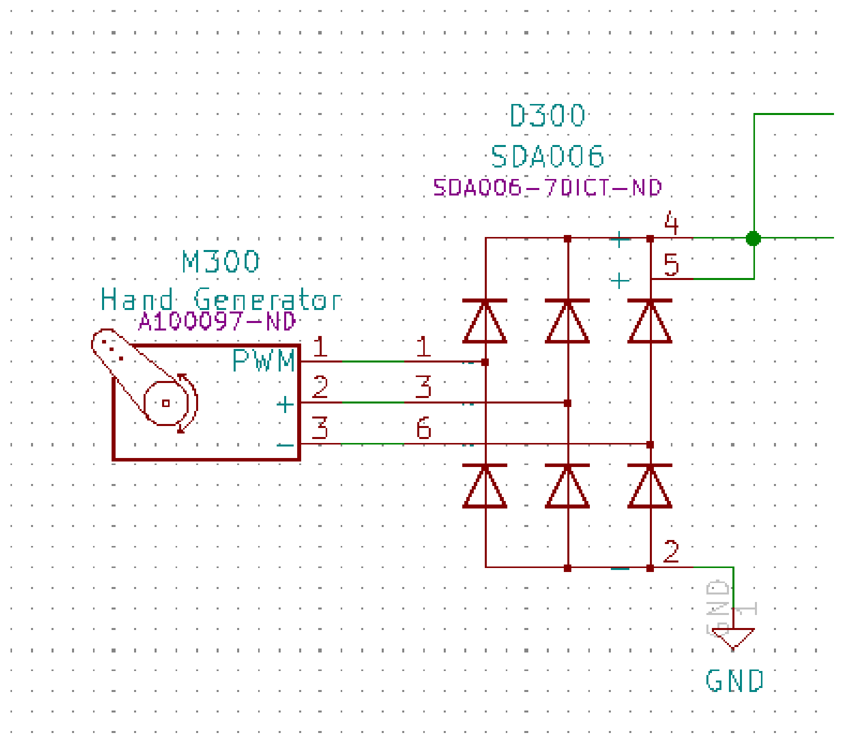


Figure 3.2: Hand Crank Schematic

3.3.2 Energy Harvesting Buttons

The buttons need to be implemented for power generation and GPIO input for gameplay. The combination of these two prompted some interesting design challenges for us to overcome.

The very first challenge we came across with these power generating buttons was the confusing configuration between the 4 pins it had. The datasheets held little information on the button itself and it was somewhat difficult to measure the magnitude of power extracted from these buttons as well, so it took our team some time to figure out how to maximize output from these devices.

The buttons held another minor inconvenience to us in the fact that half of the power output from the button came in the form of a negative wave when the button was released. This was a problem since all our power handling devices only accepted values greater than zero, although this problem was easily handled with low forward active bridge rectification to make the all power output from the button into positive waveforms. This is seen in Figure 3.3

The final issue we encountered with the buttons was how to properly connect them to the GPIO. The first design we came up with was directly connecting the corresponding GPIO pin to the output of the rectified button output. This was the most direct approach and since the GPIO pins have very high impedance there would be little power directed away from our power storage unit. However, our team did end up going for a new approach. To completely neglect the possibility of power being redirected through any GPIO, our team implemented a completely separate GPIO button connected to the 3.3V boost board output that triggered at the same time the power button is pushed. The power buttons had actuators that were able to physically press a GPIO button when the power buttons were pressed (Figure 5.3 in Implementation shows this better). This method completely separates the power generation and GPIO functionality into two separate portions so our circuitry was cleaner. It is important to note that this method also gives the GPIO a constant 3.3V pulse instead of a sloppier waveform that is typically seen on the output of the button.

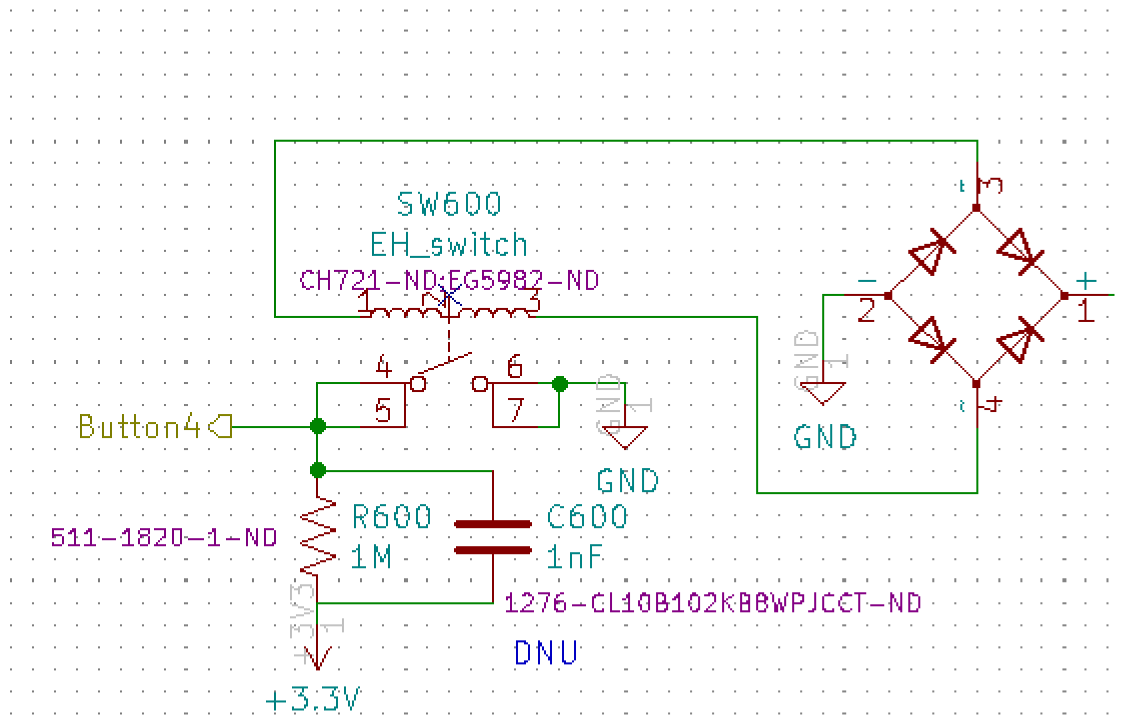


Figure 3.3: Button Circuit (Note that we have four of these circuits in our overall schematic, one for each button)

3.3.3 Power Management IC

Despite the fact that our system is a battery free device, we still need some type of subsystem for temporary storage and control. A super capacitor will be used for briefly storing incoming power from our energy harvesting devices. To manage power generation and power storage, we decided to use the BQ25504RGTR. This IC would intake power ranging from 0-5.5V and up to 200-300mA, then allocate this input onto some storage device, which for is the 100mF super cap, and output current at some voltage depending on the current voltage seen on the super cap.

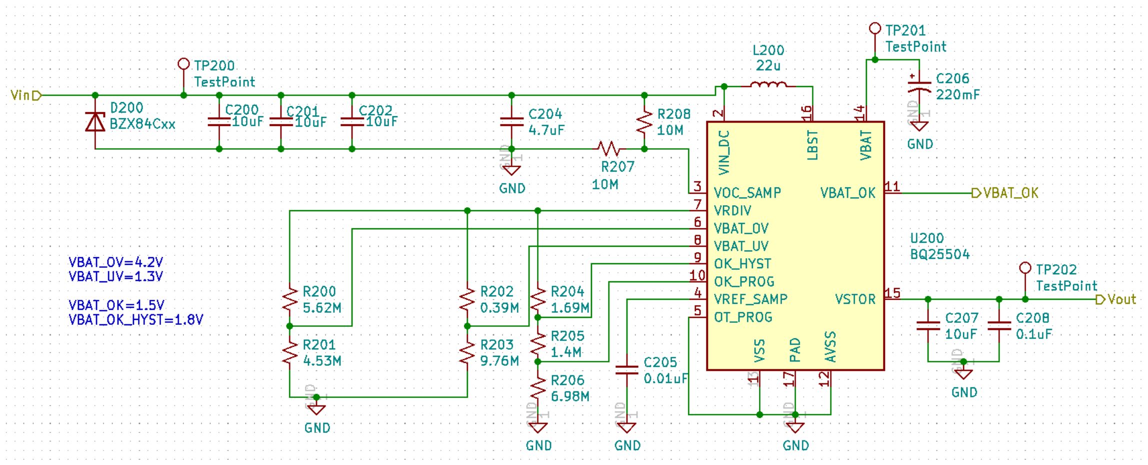
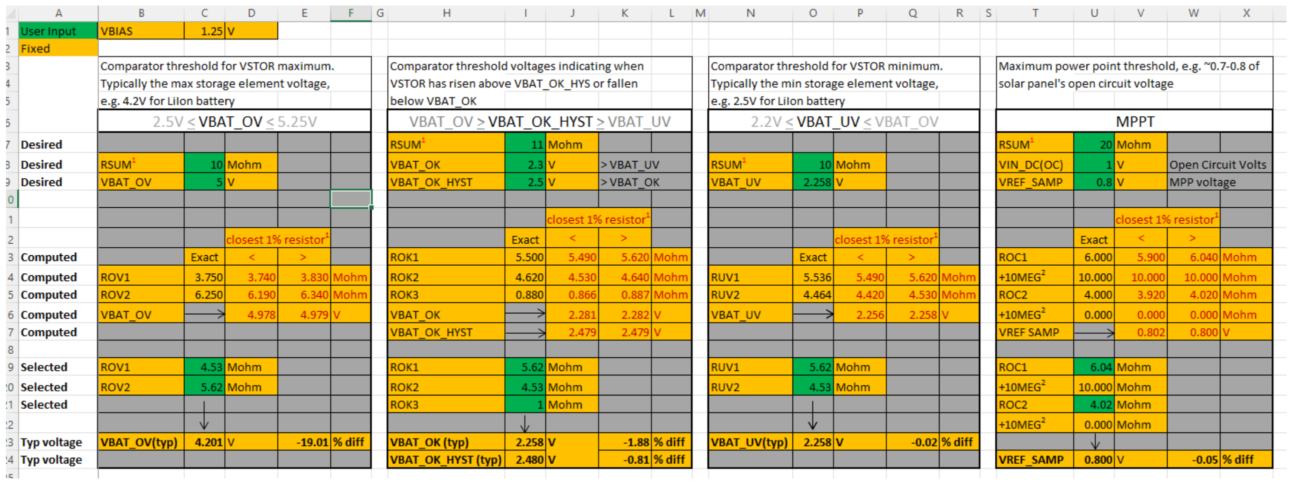


Figure 3.4: Power IC Schematic

From what can be seen in the schematic, Vin is our input, VBAT is where the supercap is connected, and VSTOR is our output. We also have other pins connected to implement UVLO among other things. We also have a 4.2V zener diode attached to the input to prevent the voltage on our input going too high and damaging our power IC.



	A	B	C	D	E	F
1	User Input	VBIAS	1.25	V		
2	Fixed					
3		Comparator threshold for VSTOR maximum.				
4		Typically the max storage element voltage,				
5		e.g. 4.2V for LiIon battery				
6		2.5V ≤ VBAT_OV ≤ 5.25V				
7	Desired					
8	Desired	RSUM[1]	10	Mohm		
9	Desired	VBAT_OV	5	V		
10						
11						
12				closest 1% resistor[1]		
13	Computed		Exact	<	>	
14	Computed	ROV1	3.750	3.740	3.830	Mohm
15	Computed	ROV2	6.250	6.190	6.340	Mohm
16	Computed	VBAT_OV	→	4.978	4.979	V
17	Computed					
18						
19	Selected	ROV1	4.53	Mohm		
20	Selected	ROV2	5.62	Mohm		
21	Selected		↓			
22						
23	Typ voltage	VBAT_OV(typ)	4.201	V	-19.01	% diff
24	Typ voltage					

	H	I	J	K	L
3	Comparator threshold voltages indicating when				
4	VSTOR has risen above VBAT_OK_HYS or fallen				
5	below VBAT_OK				
6	VBAT_OV ≥ VBAT_OK_HYST ≥ VBAT_UV				
7	RSUM[1]	11	Mohm		
8	VBAT_OK	2.3	V	> VBAT_UV	
9	VBAT_OK_HYST	2.5	V	> VBAT_OK	
10					
11			closest 1% resistor[1]		
12		Exact	<	>	
13	ROK1	5.500	5.490	5.620	Mohm
14	ROK2	4.620	4.530	4.640	Mohm
15	ROK3	0.880	0.866	0.887	Mohm
16	VBAT_OK	→	2.281	2.282	V
17	VBAT_OK_HYST	→	2.479	2.479	V
18					
19	ROK1	5.62	Mohm		
20	ROK2	4.53	Mohm		
21	ROK3	1	Mohm		
22		↓			
23	VBAT_OK (typ)	2.258	V	-1.88	% diff
24	VBAT_OK_HYST (typ)	2.480	V	-0.81	% diff

	N	O	P	Q	R
3	Comparator threshold for VSTOR minimum.				
4	Typically the min storage element voltage,				
5	e.g. 2.5V for LiIon battery				
6	2.2V ≤ VBAT_UV ≤ VBAT_OV				
7					
8	RSUM[1]	10	Mohm		
9	VBAT_UV	2.258	V		
10					
11					
12			closest 1% resistor[1]		
13		Exact	<	>	
14	RUV1	5.536	5.490	5.620	Mohm
15	RUV2	4.464	4.420	4.530	Mohm
16	VBAT_UV	→	2.256	2.258	V
17					
18					
19	RUV1	5.62	Mohm		
20	RUV2	4.53	Mohm		
21		↓			
22					
23	VBAT_UV(typ)	2.258	V	-0.02	% diff
24					

	T	U	V	W	X
3	Maximum power point threshold, e.g. ~0.7-0.8 of				
4	solar panel's open circuit voltage				
5					
6	MPPT				
7	RSUM[1]	20	Mohm		
8	VIN_DC(OC)	1	V	Open Circuit Volts	
9	VREF_SAMP	0.8	V	MPP voltage	
10					
11			closest 1% resistor[1]		
12		Exact	<	>	
13	ROC1	6.000	5.900	6.040	Mohm
14	+10MEG[2]	10.000	10.000	10.000	Mohm
15	ROC2	4.000	3.920	4.020	Mohm
16	+10MEG[2]	0.000	0.000	0.000	Mohm
17	VREF SAMP	→	0.802	0.800	V
18					
19	ROC1	6.04	Mohm		
20	+10MEG[2]	10.000	Mohm		
21	ROC2	4.02	Mohm		
22	+10MEG[2]	0.000	Mohm		
23		↓			
24	VREF_SAMP	0.800	V	-0.05	% diff

Figure 3.5: Power IC Calculations

Figure 3.5 shows the calculations we made to determine the overvoltage and undervoltage trip voltages for the power IC. The calculations consisted of finding a variety of resistor combinations to show the IC certain voltages for it to compare to. Figure 3.5 is an automated excel document created that will determine the resistors required based on undervoltage and overvoltage numbers the user enters. The document also automatically selects the closest one percent resistor so that real world resistor values can be easily found.

3.3.4 Boost Converter

On the output of the BQ25504RGTR we inserted the TPS61201DRCT boost regulator. This device is a necessary part to our system as the voltage output to the Power IC is directly correlated to the voltage currently seen on the super cap. There are several parts in our system that need a reference voltage to be beyond a certain point or at a fixed voltage. The buttons need a voltage source greater than 2.6V in order to trigger a GPIO HIGH and the E-ink display needs a constant 3.3V source in order to remain powered.

There is also an under voltage lockout functionality of the boost regulator which shuts off the output if the input voltage goes too low. This is important because without the UVLO when the supercapacitor is initially charging, the load of the system is too great and the user is never able to generate enough power to overcome the system load. The UVLO circumvents this by allowing the system to store enough charge for the large load transient of connecting the system.

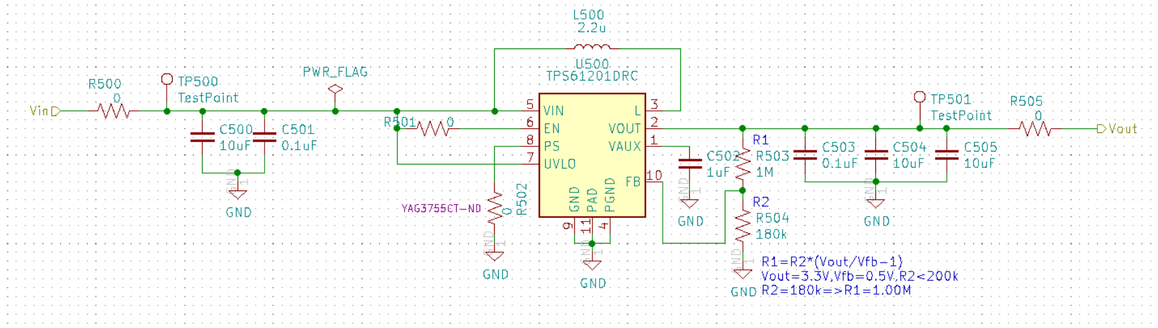


Figure 3.6: Boost Regulator Schematic

3.3.5 E-ink Display

The E-Ink display is the only user interface output for the device and instructions will be written using a 4-wire SPI data communication. The E-Ink display will be rewritten using the charge stored on the supercapacitor. The data displayed will describe the current state of the device and information pertaining to changing to another state. This event occurs every time the user enters a new room, starts a new game, or the microcontroller loses the required power to run the game.

3.3.6 uC Low Power Mode

The low power operation of the microcontroller is key to the success of the system. The MSP430 when all peripherals are turned on has a relatively large amount of quiescent current. In order to work around this the software puts the microcontroller into low power mode, which uses minimal quiescent current, as often as possible by only coming out of low power based on interrupts. The general approach to the software works around the principle of starting in low power, some user input triggers an interrupt which wakes the MSP430 from low power, the microcontroller performs some task based on the user input, and finally the microcontroller goes back to low power mode. In this manner the gameplay can still function but the microcontroller uses much less power overall.

3.3.7 Multiplayer Functionality

Multiplayer states are communicated between two devices over USB using a UART communication protocol. This allows both devices to transmit current state information and receive next state information between the two devices with minimal delay and power loss.

3.3.8 Entire Schematic

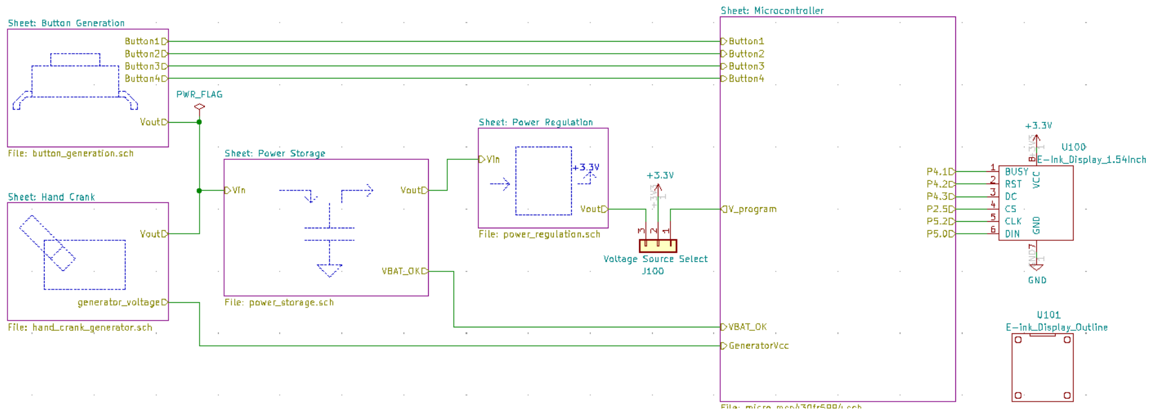


Figure 3.7: Top Level Schematic

As shown in Figure 3.7, we took all the subsystems and combined them together to match up with our functional diagram, Figure 3.1. Along with integrating all subsystems together we also implemented a number of testing functionalities that will be crucially fundamental to debugging the final PCB. We installed test points at several different places within the schematic to measure certain voltages and also so we can rebuild a portion of the PCB off a breadboard or protoboard if a certain function doesn't work. We also inserted zero ohm resistors into several portions of the PCB to separate it into its subsystems. This way if any subsystems do not work they will not impact the rest of the PCB.

3.3.9 Software

The software controls user interface outputs and deterministically changes the state of the game based on user input and power supply status. Critical information about the current state is saved and utilized to maintain the state of the game through a low power event and restoration of power supply. This allows players to maintain advancements throughout the game without the requirement of a constant power supply to the device.

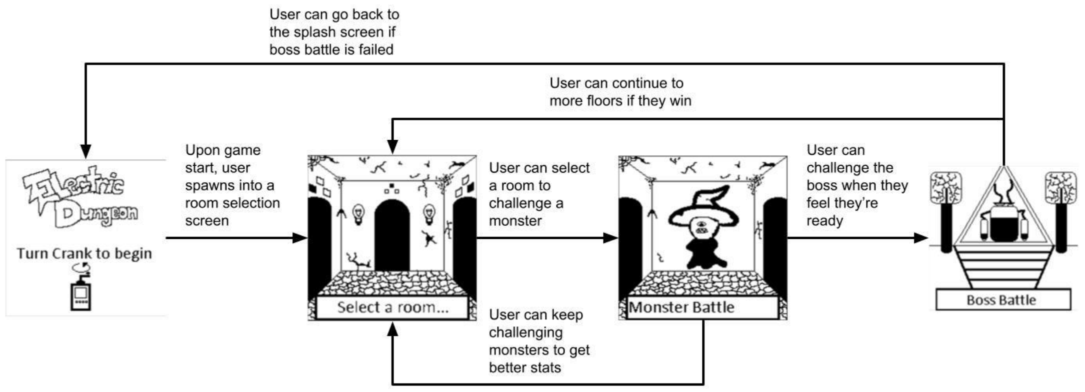

User can go back to the splash screen if boss battle is failed
User can continue to more floors if they win
Upon game start, user spawns into a room selection screen
User can select a room to challenge a monster
User can challenge the boss when they feel they're ready
User can keep challenging monsters to get better stats
Electric Dungeon
Turn Crank to begin
Select a room...
Monster Battle
Boss Battle

Figure 3.8: Software Flow Diagram

On game startup, the software flow (as seen in *Figure 3.7*) shows a splash screen, detailing a title as well as a prompt that asks the user if they want instructions for the game. If they do, the player will enter a tutorial room that walks through the basics of gameplay, like making choices with DIP switches and energy harvesting basics.

Once a player is ready to begin gameplay or if they don't want tutorial instructions, they are sent into the first room. The room selection UI will give the player the choice between three rooms. Once they select a room, a room challenge will begin. The room challenge will require a user to do an energy harvesting challenge, and if they are not able to generate enough power, they get the failure state and are sent back to the splash screen.

If a challenge is successfully completed, the player gains rewards that will make the boss fight at the end easier, and are sent back to the room selection UI. After a software defined number of challenge room loops, currently three, the player will see one of the rooms in the room selection UI changed to a boss lobby. The player can choose not to enter the boss lobby, and can go through the room selection UI/room challenge loop until the number of loops equals a maximum number of loops, currently 10, and the player is forced to choose the boss fight lobby as the next room.

The boss fight lobby provides the player a choice to engage in multiplayer, where they will wait for another player's device to enter a multiplayer state. *Figure 3.8* shows the communication of state information during multiplayer mode. The player can also choose to fight the boss singleplayer. The combat in the boss rooms will be a turn based RPG, where players will select moves from a list, using the DIP switches to make their choice. After they select their move and damage the boss's health, the boss will take a turn to fight back, decreasing the player's health. This will continue until all players' health reaches zero or the boss is defeated. Defeating the boss in multiplayer mode will allow players to continue a multiplayer version of the game loop where challenges will be tailored to multiple player interaction. Dying during the boss encounter in single and multiplayer modes will return the player to the splash screen.

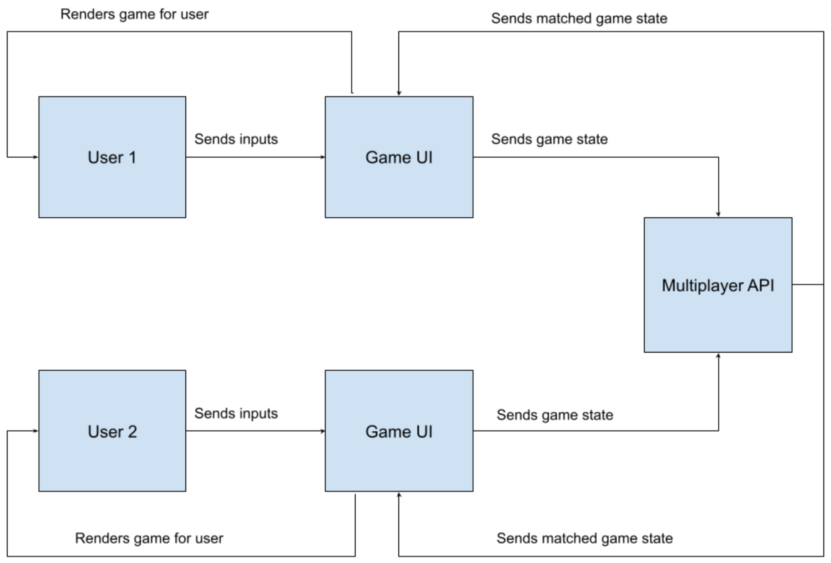


Figure 3.9: Multiplayer Communication Flow Diagram

When the MCU detects a low power event (as shown in *Figure 3.9*), software will save the state of the game to the fRAM before the device enters a low power state. In this state, a message will instruct the user to perform energy harvesting. As soon as a sufficient power threshold is achieved, the device will read the DIP switches and restore the saved state of the game, including rooms beaten, player character information, and current room state information.

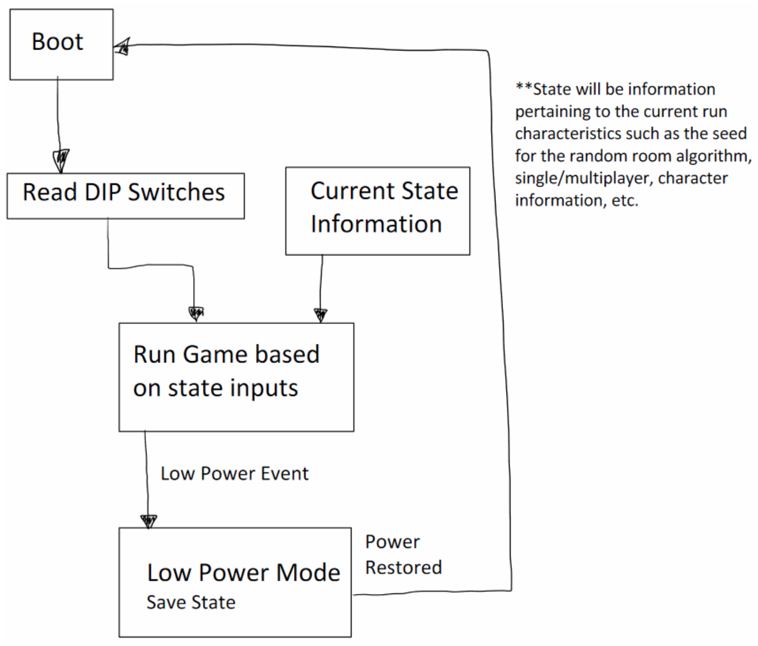

Boot
Read DIP Switches
Current State Information
**State will be information pertaining to the current run characteristics such as the seed for the random room algorithm, single/multiplayer, character information, etc.
Run Game based on state inputs
Low Power Event
Low Power Mode
Save State
Power Restored

Figure 3.10: MCU State Flow Diagram

3.4 Technology Considerations

Due to lack of batteries in the project there were many trade-offs made in technological devices chosen. The two main devices discussed were the display and the energy harvesting instruments. The display had to take as little energy as possible since the only energy available was from the energy harvesting, thus an E-ink screen was chosen since it takes practically no power in standby and moderate power when changing. A major downside of the E-ink screen is that it cannot be brought into direct sunlight which limits the device's operational environment to indoors only. However, due to the low power consumed overall it was practically the only option available.

Choosing the energy harvesting instruments was a more difficult task due to the various forms of energy harvesting available. As a base, kinetic button harvesting was a necessity since the user would almost certainly need to be able to press buttons to move around. However, the kinetic button harvesters did not produce enough energy to power the system over multiple screen changes and other forms were looked into as supplements. Solar cells were a consideration due to their low cost and wide spread use, however one downside of using solar cells is that the energy production is mostly based on the surface area of the cell. Since the game had to be handheld the overall size of the solar cell needed to be reasonably small and thus not produce an abundance of energy. Upon testing we concluded that solar cells would not be able to contribute a reasonably useful amount of energy and therefore we would not use them in our design. Alternatively, the small AC motor crank generator would serve as a capable source to meet the majority of our design's power demands, and it would also serve as a new component to integrate into the device's gameplay.

3.5 Design Analysis

The completed design works as intended with almost all requirements met. However, not all the parts worked right away and many had to be analyzed extensively to create a working design. One such example is the rectification of the hand crank generator and the rectification of the kinetic energy harvesters. Both of these components worked as expected, however one issue that came up was a possibility of the voltage produced by these components being too high if the user cranks the shaft very fast. If the voltage is above 5.1V the BQ25504 control chip will burn out which would be catastrophic to the operation of the device. The way this issue of the design was fixed was to ensure the voltage stays below 5.1V by using a zener diode set to 5V in parallel with the sources to sink any excess voltage.

Another analysis we did was on the boost regulator. We found out that we had the feedback pin hooked up incorrectly, and we also found that we needed to implement the under voltage lockout of the boost regulator so that when the system started there was not a load surge that prevented the user from ever being able to provide enough power to the system from energy devices. In order to minimize the number of setbacks we faced, we tested each module immediately after it was integrated with the microcontroller and the gameplay software. We incorporated zero ohm resistors and probe points into the design to allow for easy management of iteration of the current design.

3.6 Development Process

We used the waterfall development process for hardware because most of our tasks are only achieved if the tasks before them have been completed. It would be difficult to try to develop our project any other way due to this cascading depence of tasks.

In terms of software development, we used the agile development model. We first started off by having a minimum viable product in terms of gameplay. This first iteration only iterated between rooms, with placeholder methods for our peripherals that were not implemented. Each later iteration focused on adding one functionality, such as displaying rooms, or having working buttons, until we developed our final product.

3.7 Design Plan

The design plan for this device incorporated two teams working in tandem. The Power Systems team was responsible for developing the energy harvesting circuitry that will supply power to the device and provide physical interaction to the user. The microcontroller team developed the firmware for interpretation of physical I/O and developed software enabling a user interface and gameplay.

The teams needed to coordinate deliverable deadlines so that dependencies are produced before they are required. The power system team designed a user input detection circuit incorporating energy harvesting with buttons and crank generator. During this time the microcontroller team designed a firmware interface for the E-Ink display and software for a graphical user interface. These components enabled user interaction with the device and were invaluable to testing the system as components are added.

The next stage was for the power system team to design circuitry to regulate and power the device without external assistance. The microcontroller team designed gameplay features including random environment generation, character mobility, and room challenges. These components fulfilled the requirements for the device being batteryless and having a unique gameplay experience that incorporates energy harvesting.

The final stage was developing multiplayer functionality, which we didn’t get time to implement. Both teams would have worked together designing a method of communication between two devices and software that allowed two users to interact in a common environment to meet the requirements of multiplayer functionality.

4 Testing

4.1 Unit Testing

Energy Harvesting

1. Crank Generator & Rectification
    a. Setup crank generator in a circuit with full-wave 3-phase rectification
    b. Connect capacitor to output of rectifier
    c. Have a user crank generator at different RPM rates including smallest, average, and maximum RPM.
    d. With an initial state of zero volts on the capacitor, crank generator at different speeds for x amount of time.
    e. Measure the final voltage (V) on the capacitor when the time limit is reached.
    f. Obtain the energy produced using $E = 0.5CV^2$ and the resulting average power using $P = \frac{E}{time}$.
    g. Repeat steps a through f using different speeds as well as different users to obtain a set of power outputs.
    h. Average the set of power produced from each trial to obtain a final average power for the crank generator.
2. Energy Harvesting Button & Rectification
    a. Set up an energy harvesting button into a full-wave diode rectifier.
    b. Connect the output of the rectifier to a capacitor (C).
    c. Start at zero volts on the capacitor and push the button as fast as possible for x amount of time.
    d. Measure the final voltage (V) on the capacitor after time has stopped.
    e. Obtain the energy produced using $E = 0.5CV^2$ and the resulting average power using $P = \frac{E}{time}$.
    f. Perform steps a through e for multiple trials using various capacitance values and amount of time to obtain a set of power produced.
    g. Average the set of power produced from each trial to obtain a final average power for the energy harvesting button.

Hardware

3. Speed Detecting Circuit
    a. Connect speed detecting circuit output to voltmeter.
    b. Ensure as the speed of the generator increases, the voltage read by the voltmeter also increases.

    c. Ensure maximum output voltage is below 3.3V (for voltage protection to the microcontroller).

4. E-Ink Display
    a. Determine the amount of energy needed to load an image onto the display. Specifically, what percentage of the super capacitor's capacity is used every time a screen update occurs.
    b. Determine the maximum amount of screen writes we can achieve with a fully charged supercapacitor.

Microcontroller

1. GPIO:
    a. Set all GPIO pins to inputs.
    b. Provide a logic high to all GPIO pins to confirm read functionality.
    c. Set all GPIO pins to outputs.
    d. Set GPIO pins to set all pins to logic high for 1 second then all pins to logic low for 1 second.
    e. Confirm write functionality of all GPIO pins by using a voltmeter to probe the voltage at each pin and observe the oscillation between logic high and logic low.
2. SPI:
    a. Hook up microcontroller SPI signal wires to a slave device.
    b. Perform a writing to the slave device and confirm correct sending of data to test SPI is functioning correctly.
3. UART:
    a. Connect the UART communication wires from the microcontroller to a UART compatible device.
    b. Send a test signal to a device from the microcontroller and from a device to the microcontroller to confirm correct sending of data and the working of the UART.
4. CTPL:
    a. Run a program on the MSP430 that slowly decreases the blink rate of the onboard LED using GPIO.
    b. Disconnect MSP430 from power.
    c. Check that the LED remains off to confirm that MSP430 no longer has access to power.
    d. Reconnect the MSP430 to power.
    e. Confirm the MSP430 enters the saved state on power loss by observing that the LED blink rate is the same as when the power was disconnected in step b.

Software

1. GUI: User input and software output testing

    a. Make sure each room and button prompts render correctly with all of their components and check components positions.
    b. Test and confirm each input combination for the room selection UI.
2. Environment Generation: Random room testing
    a. Test for paths from the start room to the end.
    b. Validate each room has the correct components.
3. Room Challenge:
    a. Test success state for each room and check if players are updated.
    b. Test failure state for each room and check if players are updated.
    c. Make sure each room challenge is rendered correctly.
4. Multiplayer API:
    a. Validate messages sent via UART are the same as messages received via UART.
    b. Check after two players finish their turn that the boss health and player health is the same on both devices and is calculated correctly from both player's turns.
    c. API is able to send game state as a package.
5. Power loss:
    a. Check game state is sent to fRAM on power loss.
    b. Check that the game state, including the players' stats, room information, boss stats, and current turn in the boss fight, can be restored from fRAM upon recharging from power loss, and are exactly the same as the game state before power loss.

4.2 Interface Testing

1. E-Ink SPI:
    a. Perform a write to the E-ink display from the MSP430 using SPI communication.
    b. Confirm the E-Ink receives the SPI signal properly by observing that the display matches the intended image as detailed by software.
2. Multi device UART:
    a. Connect two MSP430 microcontrollers together via UART communication.
    b. Send a series of data between the microcontrollers and confirm that the data matches expected values using comparisons via code as well the debug window in Code Composer Studio.
3. Power to Capacitor:
    a. Crank generator and press buttons
    b. Confirm each component is providing power to system by observing voltage increase on super capacitor
4. Power to System:
    a. Fully charge the supercapacitor using a DC power supply.
    b. Slowly drain the energy harvesting system (the power system of our device) by connecting a load (simulated system) to the output.

    c. Confirm that 3.3V is output to the load over the full working range of the supercapacitor.
5. Speed Detecting Circuit with Microcontroller:
    a. Connect the speed detector circuit to an ADC input on the microcontroller.
    b. Use the ADC to measure the voltage from the detector circuit and confirm correctness with voltmeter.
6. Buttons:
    a. Setup energy harvesting buttons into their correct GPIO pin.
    b. Confirm the rising edge reads by the microcontroller of the energy harvesting buttons.

4.3 Acceptance Testing

The functional requirements consist of three main pieces, the device is powered only from energy harvesting, energy harvesting is part of the gameplay, and the device supports some multiplayer functionality. Energy harvesting being part of the gameplay is clearly demonstrated as shown in the room challenges with the tasks that involve spinning the hand crank generator and pushing the buttons. The device being powered only from energy harvesting can be demonstrated by showing to the client that there is clearly no battery or charging port as part of the hardware, meaning the power must be coming from the energy harvesting devices. If the multiplayer requirement was met it would have been clearly shown in the core game when the final boss battle was encountered and collaboration with another player could be used.

The nonfunctional requirements have tests designed for each case. Portability shall be tested by seeing if the device can fit into a normal backpack. Intuitiveness will be tested by seeing if a user who has not played the game before can play the game from start to finish with little to no outside help. For the game running through a room challenge state, we will run at least ten tests where we start a room challenge. After energy harvesting, we should be able to get through the room challenge state and be sent back to the room selection UI at least nine times out of ten. Lastly, for the low power screen, every time the device reaches low power, it should switch to the low power screen.

4.4 Testing Results

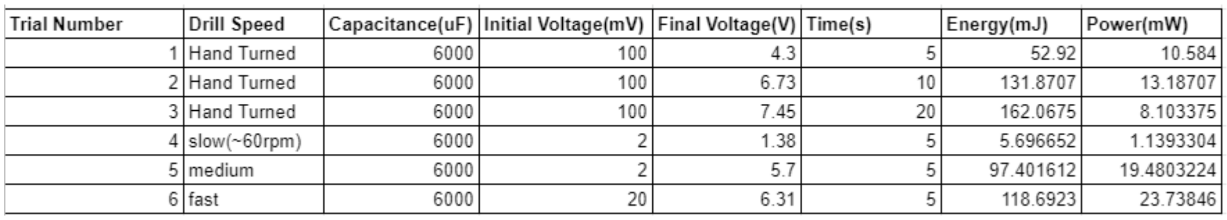
The information below reiterates the test procedures, our data collected, and our results. The main takeaway from this initial testing is the proof of feasibility of the battery free handheld game.

In order to prove feasibility we took the route of being able to update the E-ink display as confirmation that the making of the battery free game design we had was feasible. The reason being that the E-ink display, by a large margin, will be the largest draw of power for our system. So if we are able to comfortably update the E-ink display often strictly from energy harvesting devices then that proves our design is feasible.

Based on the data below the energy harvesting buttons produce an average of 1 mW and the hand crank generator produces an average of 10.6 mW while being used. From these values the energy harvesting button will be able to update the E-ink display every 50 seconds and the hand crank generator will be able to every 5 seconds. Thus, while the energy harvesting buttons may not be super useful in generating the majority of our power, the hand crank generator will be able to easily produce enough power which confirms that our design is feasible. Moreover, if needed we can introduce a gear ratio to increase the speed and thus power production of the hand generator for even more power production which further confirms the proposed design with function.

4.4.1 Hand Crank Generator

Test Procedure: Set up hand crank generator into a full-wave 3-phase diode rectifier. The output of the rectifier is connected to a capacitor. Start at 0V on the capacitor and turn the generator using drill or hand at various speeds for x amount of time. Measure the voltage after time has stopped. The average power production of the generator is equal to energy divided by time (in seconds).



Trial Number	Drill Speed	Capacitance(uF)	Initial Voltage(mV)	Final Voltage(V)	Time(s)	Energy(mJ)	Power(mW)
1	Hand Turned	6000	100	4.3	5	52.92	10.584
2	Hand Turned	6000	100	6.73	10	131.8707	13.18707
3	Hand Turned	6000	100	7.45	20	162.0675	8.103375
4	slow(~60rpm)	6000	2	1.38	5	5.696652	1.1393304
5	medium	6000	2	5.7	5	97.401612	19.4803224
6	fast	6000	20	6.31	5	118.6923	23.73846

Figure 4.1: Hand Crank Generator Tests

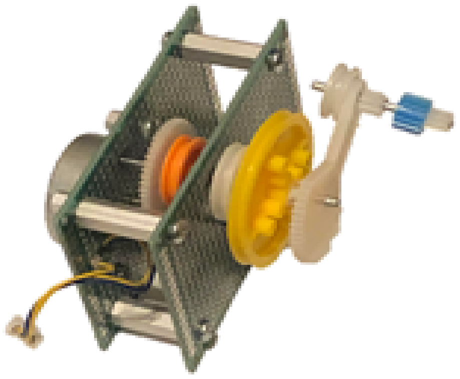
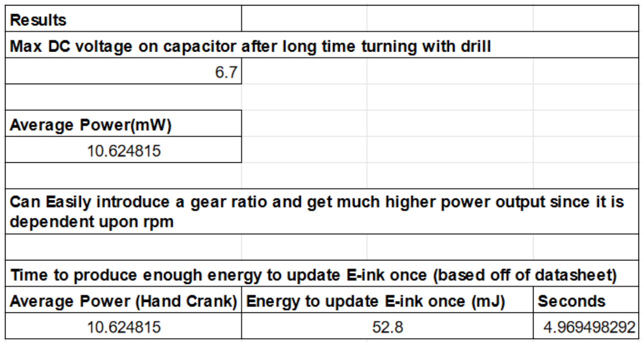


Figure 4.2: Hand Crank Schematic

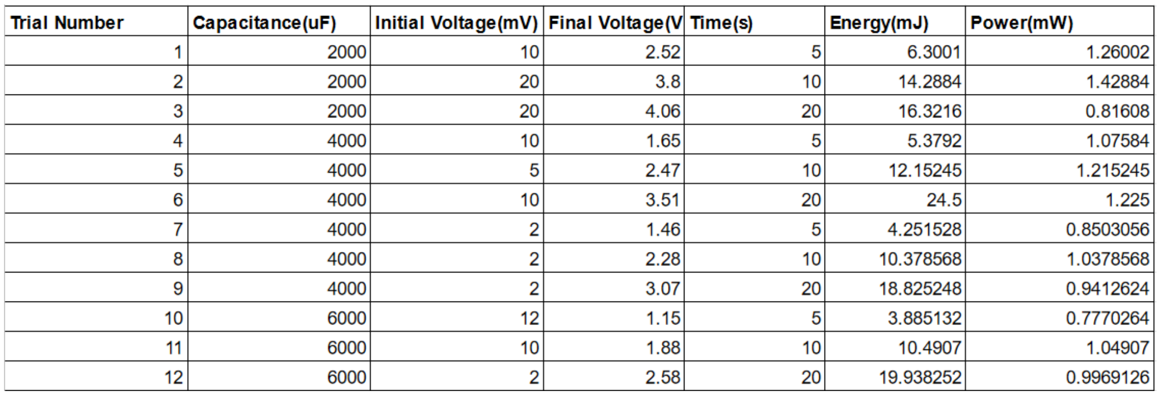


Results		
Max DC voltage on capacitor after long time turning with drill		
6.7		
Average Power(mW)		
10.624815		
Can Easily introduce a gear ratio and get much higher power output since it is dependent upon rpm		
Time to produce enough energy to update E-ink once (based off of datasheet)		
Average Power (Hand Crank)	**Energy to update E-ink once (mJ)**	**Seconds**
10.624815	52.8	4.969498292

Figure 4.3: Hand Crank Test Results

4.4.2 Energy Harvesting Buttons

Test Procedure: Set up an energy harvesting button into a full-wave diode rectifier. The output of the rectifier is connected to a capacitor. Start at 0V on the capacitor and push the button as fast as possible for x amount of time. Measure the voltage after time has stopped. The average power production of someone pressing the button fast (our use case for a skill check) is equal to energy divided by time (in seconds).



Trial Number	Capacitance(uF)	Initial Voltage(mV)	Final Voltage(V	Time(s)	Energy(mJ)	Power(mW)
1	2000	10	2.52	5	6.3001	1.26002
2	2000	20	3.8	10	14.2884	1.42884
3	2000	20	4.06	20	16.3216	0.81608
4	4000	10	1.65	5	5.3792	1.07584
5	4000	5	2.47	10	12.15245	1.215245
6	4000	10	3.51	20	24.5	1.225
7	4000	2	1.46	5	4.251528	0.8503056
8	4000	2	2.28	10	10.378568	1.0378568
9	4000	2	3.07	20	18.825248	0.9412624
10	6000	12	1.15	5	3.885132	0.7770264
11	6000	10	1.88	10	10.4907	1.04907
12	6000	2	2.58	20	19.938252	0.9969126

Figure 4.4: Energy Harvesting Button Tests

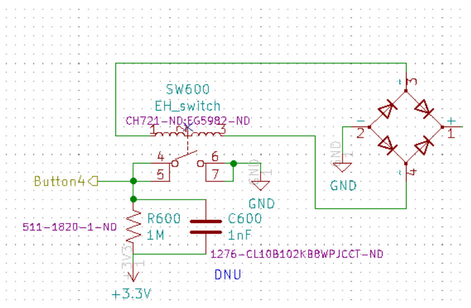
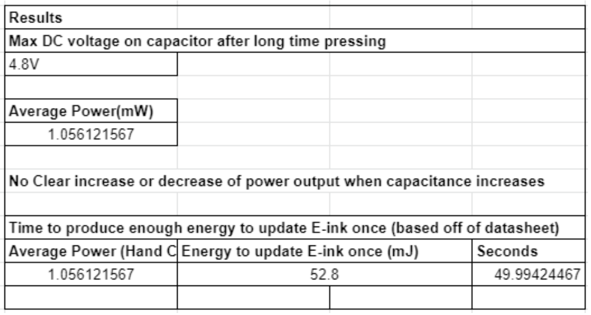


Figure 4.5: Button Schematic



Results			
Max DC voltage on capacitor after long time pressing			
4.8V			
Average Power(mW)			
1.056121567			
No Clear increase or decrease of power output when capacitance increases			
Time to produce enough energy to update E-ink once (based off of datasheet)			
Average Power (Hand C	Energy to update E-ink once (mJ)		Seconds
1.056121567	52.8		49.99424467

Figure 4.6: Button Test Results

4.4.3 Breakout Boards

To appropriately test the functionality of our ICs (the BQ25504RGTR power IC & the TPS61201DRCT boost IC), we designed breakout PCBs for each IC. The breakout boards would allow us to pinout each IC so that it can be placed in a breadboard and tested using lab equipment. Figure 4.7 and 4.8 show their layouts and the physical boards.

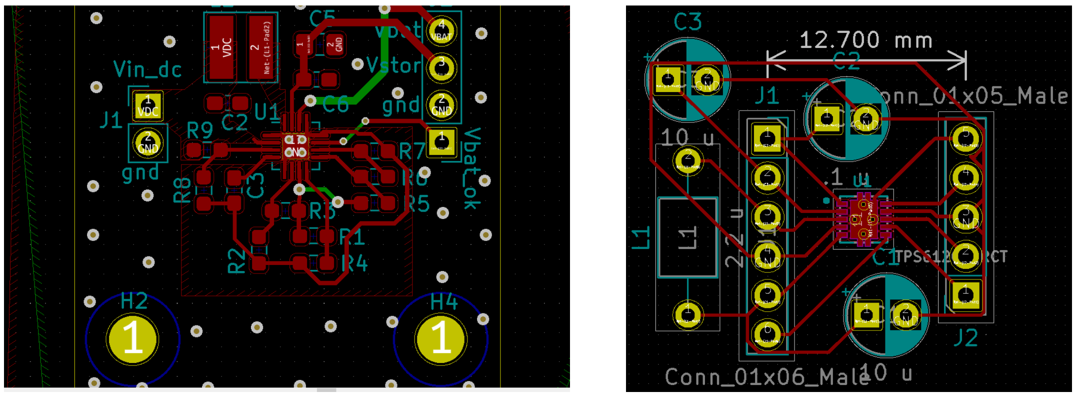


Figure 4.7: Breakout Board Layouts

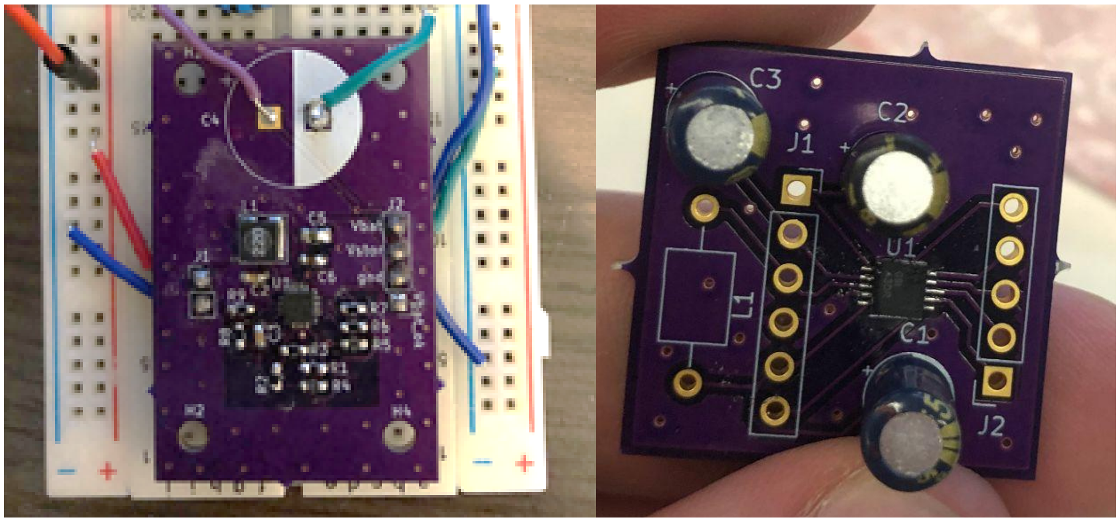


Figure 4.8: Physical Breakout Boards

4.4.4 Capacitor Charging

One of the most important factors we have to take into consideration is the exact size of our super cap. A large cap will take an incredibly long amount of time to charge and discharge while a small cap will charge and discharge too quickly. The following test shows a comparison of different sized capacitors and how long it took to fully charge them.

Test Procedure: use power IC breakout board with external super cap and crank generator to charge super cap. Time how long it takes to reach max voltage (~4.2V as limited by Power IC). Discharge the capacitor to 0V before the next trial.

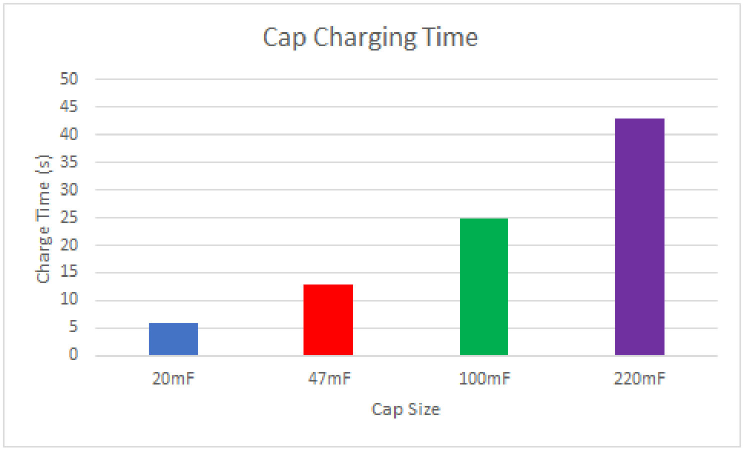


Figure 4.9: Capacitor Charging Time

The results seen in Figure 4.9 would at first lead us to pick the 47mF capacitor, as 100mF and 220mF have incredibly long charge times. However, something that we did not take into consideration during design but found out during implementation was that the power IC has a cold start, meaning that when starting with 0V on the cap, it will take a much longer time to charge. So these results will more resemble a user picking the game up when it has not been played for a very long time (enough time to bring cap voltage below 600mV as specified by power IC datasheet), and it should charge significantly faster during gameplay. Unfortunately we did not find the time to go back and redo these experiments while considering cold start as we found out about the cold start too late into the semester. However, we did make the final decision to use the 100mF cap instead of the 47mF cap.

4.4.5 E-Ink Display

The biggest factor in power consumption in our entire system is the E-Ink display when it is performing a full refresh. Since this event is the most power consuming event, we decided to measure how many full refreshes we could get off a single fully charged super cap.

Test Procedure: use a fully charged super capacitor to write a full-screen refresh to the E-ink display until the microcontroller enters low power mode.

When using the 100mF, we were able to successfully full refresh the E-ink display 5 times before running out of charge.

When using the 220mF, we were able to successfully full refresh the E-ink display 10 times before running out of charge.

4.4.6 Boost UVLO

Using the boost IC breakout board, we could measure when the boost IC's undervoltage lockout kicked into effect, which should drop the voltage from 3.3V to ~0V. This behavior is represented when the power IC is no longer able to supply enough voltage to the system for it to work effectively, so instead of supplying our devices with undervoltage and potentially damaging them, the boost effectively goes to 0V. The threshold voltage for the UVLO is 2.52V, so we put this to the test with the voltmeter and power supplies in Coover's labs.

Test Plan: Set up boost IC. Set 5V on the power supply and hook up the voltmeter to the output of the boost IC. Decrease power supply voltage by 500mV and measure output voltage on boost IC. Results can be seen in Figure 4.10

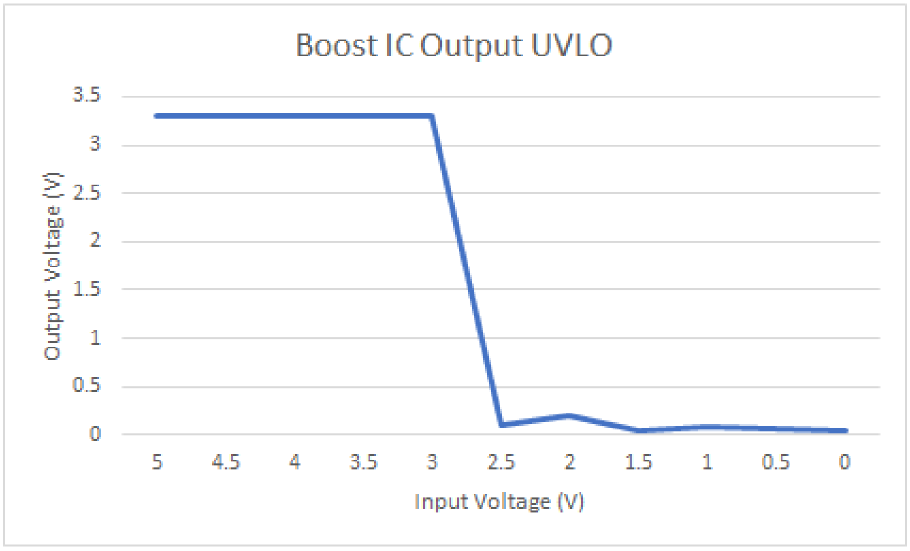


Figure 4.10: Boost IC UVLO

4.4.7 Software Testing

Software testing mainly focused on modular testing of the various peripherals the MSP430 has that we required. The software for each peripheral was created, then uploaded to the MSP430 development module, and functionality finally confirmed. One major finding from tests included figuring out that the E-ink display module took a large amount of quiescent current unless all SPI data lines were turned off via software. Another issue solved through software testing was how to save the correct states for ctpl so that when the device turned back on the user could continue playing without issue. Lastly, software testing revealed a need for a different way of obtaining button presses. The kinetic energy buttons by themselves had such a short pulse when actuated the system had a hard time detecting it. This led to the hardware development of being able to push a pushbutton on the bottom side of the pcb through a cutout using the actuator of the kinetic energy harvester.

5 Implementation

Beginning this semester we finalized all our unit testing, completed PCB designs based on hardware performance in each unit test, and began integrating each component one stage at a time. Our integrated design was then used to determine the ideal specifications to allow our system to store enough power while not taking away from the user experience. Throughout this process we encountered several issues that allowed us to fine tune our design to perform as expected. In the end we were able to combine all components into a fully-functional hand held device. Below Shows our final implementation with each major part outlined in relation to our functional diagram.

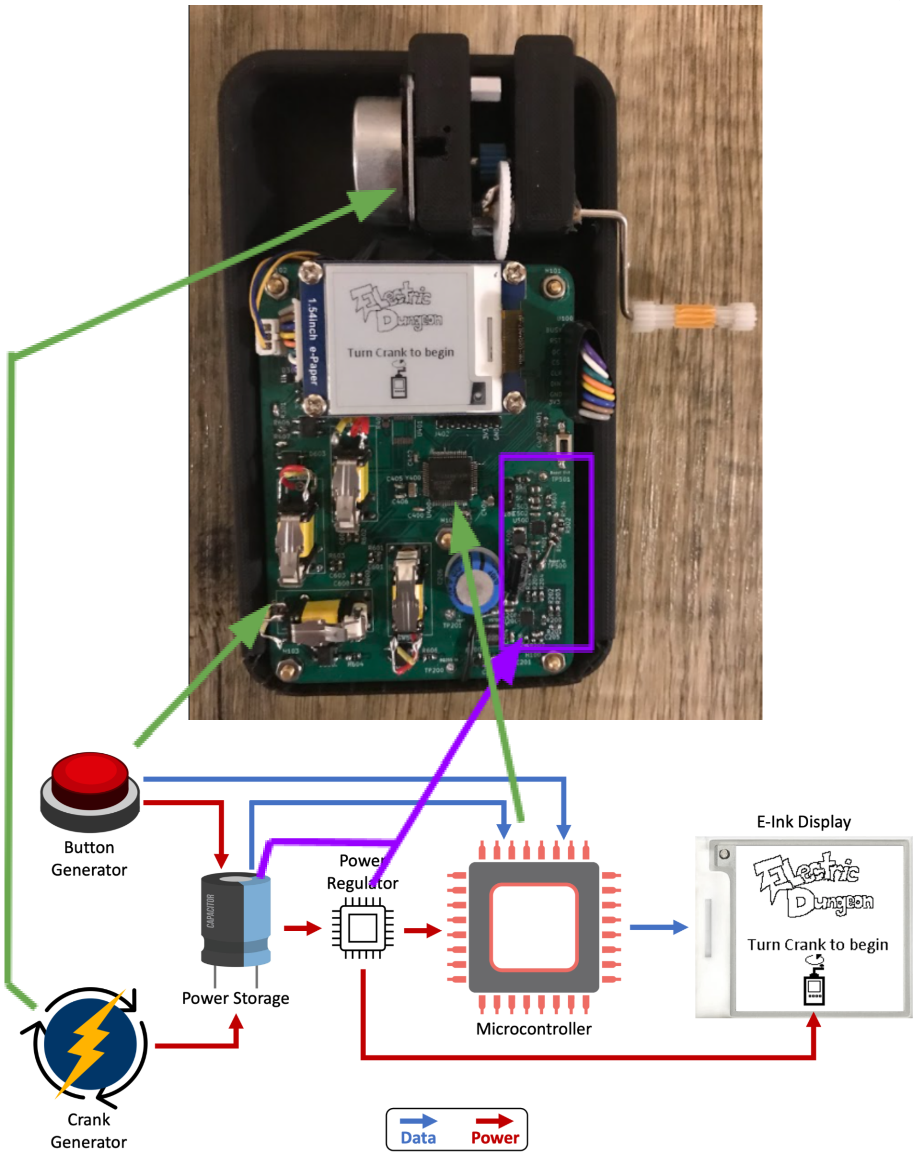


Figure 5.1: Functional Diagram to Finished Product

5.1 Hardware Implementation

Throughout the implementation quite a few issues were encountered and documented to give way to a better design in the future. One such issue we encountered was the cold start of the BQ25504 power storage module. When the system initially starts storing energy onto the supercapacitor there is a period of high inefficiency where the IC needs to get enough initial charge to start the high efficiency boost regulator. Due to this the charging of the supercapacitor on the board takes closer to ~25 seconds rather than an ideal ~10 seconds if there is no cold start. In a future revision one possible solution would be to add some sort of capacitor banking at the input to the power storage IC so that it is always in the high efficiency boost state. Another solution would be to select a different power storage IC.

Another issue encountered during the implementation of the final board was accessing the correct voltages to determine when the system should go into low power state. Initially the thought was to use the 3.3V rail to the microcontroller and monitor when this started to drop meaning the system was out of energy. However, since the E-ink display needs quite a large amount of energy to change screens, and we want to change screens before power off, this did not work. Instead we switched the microcontroller to measure the voltage on the supercapacitor so that we could clearly tell exactly how much energy the system had left and make decisions based on this.

Knowing the supercapacitor voltage also turned out to be a better measure of how quickly a user generated power when doing a hand crank game. Initially the intent was to measure the hand crank generator voltage to determine the speed of the generator, however the power IC module was so efficient that the generator voltage remained roughly constant no matter the speed. Thus the voltage on the supercapacitor was measured instead and how quickly this increased determined if the user was spinning the crank fast enough.

5.2 Software Implementation

Software development of the game happened in parallel with the hardware development. The development strategy used was the Agile development model. This process allowed the core functionality of the game to be implemented and tested over time while adding degrees of complexity throughout the development cycle. We began by developing the main gameplay loop with user input being hard coded data. From there, we developed user interface displays and integrated them with the e-ink display functionality for testing. The next stage was developing the random room generation algorithm and integrating it for testing. The third stage was developing user input detection for controlling the game. This was integrated with the energy harvesting power system for testing. Lastly, we implemented the compute through power loss library to save the user's progress even if the device turned off. While there were plans for multiplayer, unfortunately though we did not have the time to properly follow through with those plans.

6 Closing Material

6.1 Conclusion

This semester we implemented our design plan into a product. We were able to create a functional game that does allow the user to continue onwards with their progress if the game loses power. The game also mainly runs on the hand crank generator power. While we were not able to implement multiplayer, we achieved most of our goals for this project. Our risk mitigation methods helped to reduce the amount of refactor time we would need. In addition, our plan to concurrently develop hardware and software allowed us to implement as much of our design plan as possible with the relatively short development window we had.

6.2 References

Mastellari , Alessandro. “A Quick Guide to Powering Smart Objects with Kinetic Energy Harvesting.” *Avnet*, 22 Jan. 2019, www.avnet.com/wps/portal/abacus/resources/article/a-quick-guide-to-powering-smart-objects-with-kinetic-energy/.

“MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers.” Texas Instruments, Jan. 2021. https://www.ti.com/lit/ds/symlink/msp430fr5994.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1614352864965

“MSP430 SPI Peripheral.” *ARGENOX*, www.argenox.com/library/msp430/msp430-spi-peripheral-chapter-9/.

“Rectification of a Three Phase Supply Using Diodes.” *Basic Electronics Tutorials*, 3 Sept. 2018, www.electronics-tutorials.ws/power/three-phase-rectification.html.

“200x200, 1.54inch E-Ink Display Module.” *Waveshare*, Apr. 2021, www.waveshare.com/1.54inch-e-paper-module.htm.

“1.54inch e-Paper Module.” *1.54inch e-Paper Module - Waveshare Wiki*, Apr. 2021, www.waveshare.com/wiki/1.54inch_e-Paper_Module.

“Wireless Switches – Generator .” Mouser Electronics, 30 Apr. 2015.https://www.mouser.com/datasheet/2/833/Energy_Harvesting_Generator_US-1075748.pdf

“BQ25504Ultra Low-PowerBoostConverterWith BatteryManagementFor EnergyHarvesterApplications.” Texas Instruments, Nov. 2019. https://www.ti.com/lit/ds/symlink/bq25504.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1619382959060

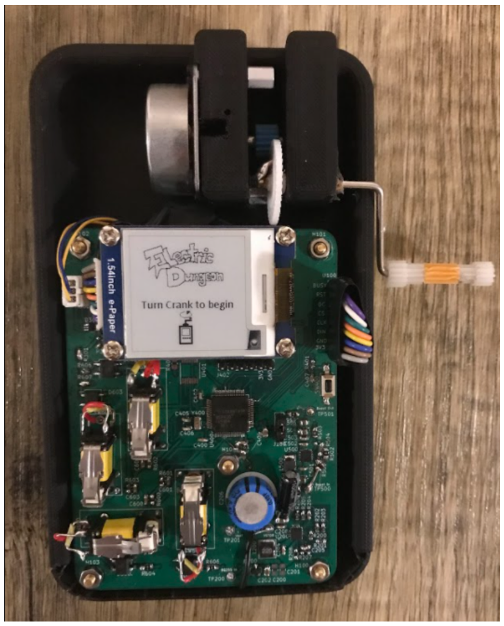
“TPS6120xLow Input VoltageSynchronousBoostConverterWith 1.3-A Switches.” Texas Instruments, Mar. 2007. https://www.ti.com/lit/ds/symlink/tps61202.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1619382985859

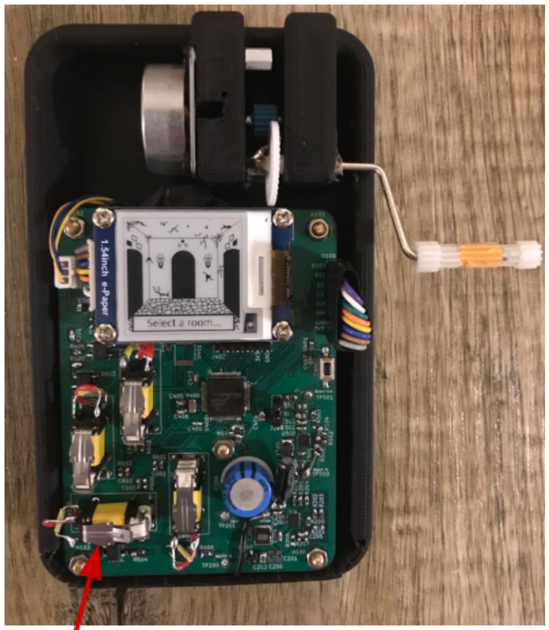
6.3 Appendices

Appendix I - User Guide

Core Game Loop:

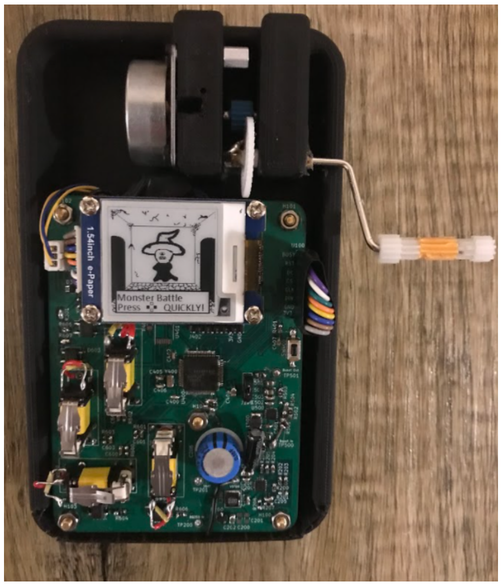
Step 1: Turn Crank until screen changes



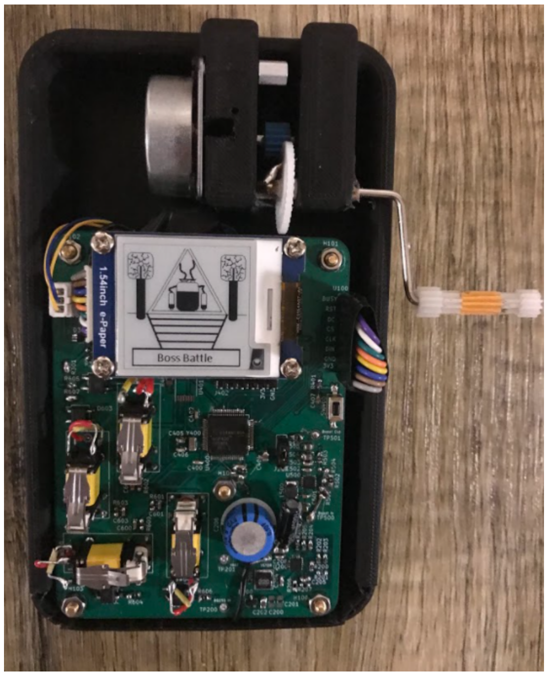


Step 2: Select a room using the push buttons shown above

Step 3: Fight the monster by following the on screen prompts (shown below)

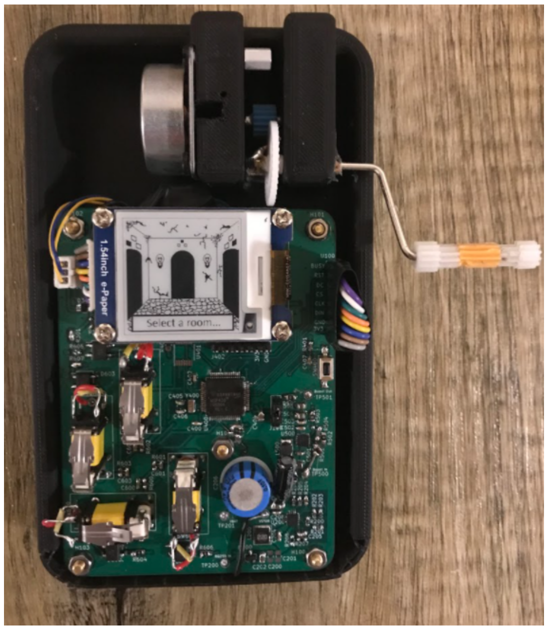


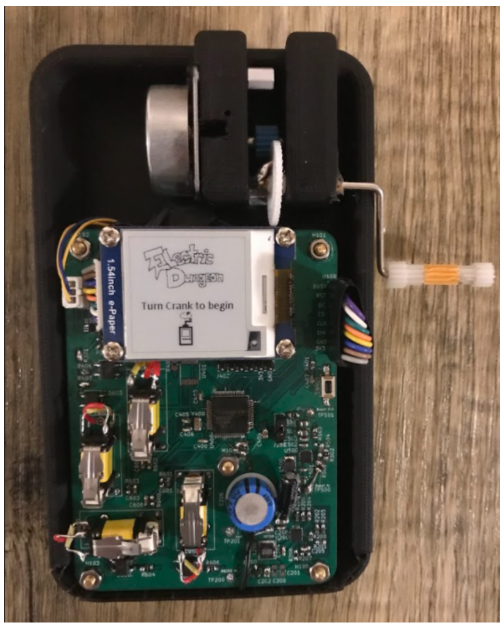
Step 4: Repeat Step 2 and Step 3 until the boss room is encountered



Step 5: Fight the boss by following the on screen prompts
Step 6: Once beaten restart from Step 1 to play the game again

Intermediate Power Handling:
If at any point the system does not have enough energy stored on the capacitor to play the game the state will change to the low power screen.
Step 1: Screen changes from game to low power screen





Step 2: Turn the crank generator until the game turns back
Step 3: Continue play

Appendix II - Code

https://git.ece.iastate.edu/svashi/ccs-senior-design