# EEL4914 Senior Design I Report -

# Thermal Dune Energy Storage


Department of Electrical Engineering and Computer Science
University of Central Florida - Dr. Chan / Dr. Lei Wei

## Group 9
## Engineering Team:

Filipe Pestana Frances - Computer Engineering
Michael Hernandez - Electrical Engineering
Miguel Baca-Urteaga - Computer Engineering
Tanner Cyr - Electrical Engineering

## Review Committee:

| | | |
|---|---|---|
| Mark Maddox | ECE | Adjunct |
| Dr. Mike Borowczak | ECE | Associate Professor |

# Table of Contents

# List of Tables



# List of Figures

# Chapter 1:Executive Summary

As the world continues its transition to renewable energy sources questions have been raised about addressing the intermittency of renewable energy. The thermal sand battery will be an addition to the approach towards carbon neutrality. The thermal sand battery will be an energy storage device that does not require any sort of professional installation and will perform home heating tasks at a cost that is competitive with lithium ion batteries. The thermal sand battery will be able to be set up in any room in a house and would then be able to heat the room up to a user defined temperature either on a scheduled basis or the user will be able to override the schedule and immediately turn on the battery. The user will be able to set charging times (or override to immediately charge). The purpose of the scheduled charging times will be to give the user the opportunity to charge the battery during off peak electric rates which will allow them to save money on electrical costs over time. The thermal sand battery will compose of an array of sensors to monitor internal battery temperatures to convey a charge state, the battery will have an array of external sensors to detect ambient temperature so as to monitor if the battery should release more heat or not. The thermal sand battery will control an internal heating element with a relay so the battery can charge as needed. Lastly the battery will have an LCD the user can interact with so they will be able to set charge times, heating times, override scheduled charging/heating schedules , and monitor internal temperature to give a sort of relative state of charge. This is all intended to be able to address one of the largest power uses in the average american home while also making it accessible to renters by virtue of the battery needing no special installation, as well as low income americans by virtue of having a thermal battery that will be cheaper per kWh by several orders of magnitude when compared to lithium ion batteries, and having a battery that suffers no type of storage degradation as this thermal battery wouldn't be affected by over time cycle degradation as its not a chemical process like lithium batteries. In an environmental perspective we are also wanting to make a battery that is less environmentally intensive to produce compared to lithium batteries, as there is no special extraction process for sand, and there is a growing market of 'rejected construction sand' which can be used. The philosophy that we used when building this battery is 'if you want to build a dirt cheap battery, build it out of dirt', the benefit of this form of energy storage will be that components that would need replacing in our theoretical 'end of life' stages of the battery would likely be the heating elements which are made of copper, nickel and stainless steel making them very recyclable, and we benefit from just how readily available the heating element market is giving us no limit on potential suppliers as opposed to depending on sensitive international lithium supply chains which can be subject to individual national security interests, tariffs, and trade restrictions. Our biggest advantage aside from per kWh pricing will be that the supply chains we are working with are well established and very stable.

# Chapter 2:Project Description

## 2.1 Project Background and Motivation

The thermal sand battery is intended to electrically heat a home while saving money by utilizing off-peak electric rates. This battery uses a thermally insulated core made of sand to store heat until it is needed later. The inspiration for this project came from exploring alternative energy storage methods for excess solar power. As the power grid transitions to renewable energy sources, there is an increasing focus on the issue of renewable energy intermittency. Although solar and wind energy are significantly cheaper than fossil fuels (even when controlled for industry subsidies), the question of what happens when the wind doesn't blow, and the sun doesn't shine is crucial. The answer is 'energy storage,' but the debate centers on the best form of energy storage.

Currently, lithium-ion batteries are the favored solution. While they are capable and efficient, they have drawbacks that our Thermal Sand Battery can address. Renewable energy intermittency is often discussed alongside 'load shifting.' Solar panels can produce excess power during the day, which is often dumped into the grid. This overproduction can cause net grid demand to drop drastically (and sometimes go negative), jeopardizing grid stability. Solar production causes a significant drop in net grid demand during the day, followed by a sharp increase at sunset when people return home, creating a 'duck curve' in overall grid demand.

This situation has led to the popularity of 'time of use rates' for residential electric users, where electric rates vary depending on projected grid demand. Users pay higher rates during 'peak grid use' (6pm-9pm) and lower rates during 'off-peak grid use' (12pm-6am). This approach encourages 'load shifting,' where users perform energy-intensive tasks during off-peak hours. This strategy promotes a more even distribution of power demand, reducing extreme swings in grid demand. Users with solar battery packs often charge their batteries during off-peak hours if their solar array didn't fully charge the battery during the day or if they lack enough storage to last through the night.

The Thermal Sand Battery aims to provide a form of energy storage comparable in density to lithium batteries, convenient home/room heating, and significantly cheaper energy storage. According to the US Energy Information Administration, 42% of energy usage in the average American home is for space heating, increasing to 45% in single-family homes. While it is possible to run a heater on a lithium-ion battery, the cost is prohibitive. Commercially accessible lithium-ion batteries, including those for residential solar arrays, cost about $300 per kWh. This makes the upfront cost of saving money through load shifting significantly higher. On the other hand, a gallon of fine-grain sand can store roughly 500Wh of power considering a specific heat of 830 J/kg. Given the low cost of sand and its lack of performance degradation over ten years (compared to a 10% capacity

decline every five years for lithium batteries), we believe we can build a battery that is several times cheaper than a lithium-ion battery both short and long term.

The Thermal Sand Battery allows consumers to charge and store heat energy during off-peak hours and discharge it when needed, ensuring maximum comfort. Additionally, it avoids the risks associated with storing highly combustible lithium-ion cells and thermal runaway, offering greater stability. Our project aims to adhere to the saying, "if you want to make a dirt-cheap battery, you have to make it out of dirt."

Lastly, ease of installation is a significant consideration. The $300 price tag for 1 kWh of lithium battery storage covers just the battery. Lithium batteries and their DC power require an inverter with proper fusing and wire sizing. For non-DIY consumers, power stations cost far more than $300 per kWh, and professional installation adds significantly to the break-even cost. Our sand battery is designed to be a no-frills solution, running off a simple 120V outlet, requiring no permitting, DIY knowledge, or concerns about lithium cell fires.

## 2.2 Project Goals and Objectives

After meeting with professor Chan and Professor Wei, this project can be built so as to be able to meet the requirements for senior design. The requirements presented to us were to have a custom PCB board with an integrated MCU, a custom power supply to power this MCU, and to have our custom PCB interface with an array of external sensors. The design of this project aesthetically will be an object that can actually be placed within a room, think along the lines of a coffee table. In terms of the engineering components, this is where it may get a little more difficult. This project will include a main PCB board, this PCB will interface with an LCD with an interactive GUI that will allow the user to change various settings from the thermal battery. The GUI will allow the user to set a specified charge time, set a specified room heating time, set an ideal room temperature to heat the room up to, and to be able to override the room heating schedule and instead start immediately heating the room. The GUI will be controlled by the MCU on our custom PCB board, the average internal battery temperature will be calculated by two thermometers in the MCU and displayed on the LCD for the user to see, the average external room temperature would also be calculated by two outward facing temperature sensors.

### 2.2.1 - Goals

1. The battery must be capable of releasing thermal energy to heat a room to a specified temperature
2. The battery must use a heating element that can fully heat our battery within a reasonable amount of time

3. The battery will be enclosed in an insulated box to prevent heat loss and injury to the user
4. The battery must know when to stop heating and charging
5. The MCU must communicate with temperature sensors to process that information back to the LCD
6. The LCD must have an interface that will let the user control heat settings and charging settings
7. The user must be able to charge the battery on a specified schedule

### 2.2.2 - Stretch Goals

1. The custom-made PCB can communicate via Bluetooth to outer sensors for a more accurate reading of the external room temperature
2. The MCU can store energy usage data and history onto a cloud-based server
3. Create a mobile application to communicate with GUI

### Objectives

1. Determine the best insulator materials required for the battery enclosure
2. Determine a sufficient heating element that is energy efficient
3. Determine a microcontroller (MCU) capable of scheduling and task handling
4. Identify the optimal design for implementing the AC-DC converter
5. Determine which sensors are compatible with selected MCU
6. Determine the power requirements for the heating element and components
7. Determine communication protocols required between peripherals and MCU
8. Determine the best approach for implementing a user-friendly interface
9. Identify components required to realize custom made PCB
10. Identify the best PCB layout design for EMC considerations

## 2.3 Project Description of Features & Functionalities

Because thermal sand batteries are new to adoption as a form of energy storage the consumer/utility sector hasn't quite reached the point of widespread adoption as we've seen with something like flooded lead acid batteries, and lithium-ion batteries. Our research has brought two notable theoretically similar - in that we are utilizing energy storage in the form of heat as opposed to a chemical process- instances of a thermal sand battery.

The first instance of this project comes from a company called 'polar night energy' based in Finland. Their sand battery is a grid scale heating store, their battery is roughly the side of a medium sized corn silo which was connected to their district heating system and was able to deliver a 100kW quantity of heat to the local Kankaanpaa municipality and boasts a storage of 8mWh. Finland has a

high degree of renewable energy adoption in the form of wind energy and polar night energy has been able to utilize low energy rates during peak wind production to be able to charge their thermal sand battery. This gives them a market advantage over on demand heating sources like regular electric heaters or fossil fuel sources. It has been estimated that because of this system the must know when to stop heating and charging a battery will allow the municipality to reduce carbon dioxide emissions from district heating by as much as 70%.
The second market implementation comes from a company called 'batsand'. Conceptually it is a similar idea, using sand to store and distribute heat and controlling it with a thermostat. The implementation of their battery is one of the largest drawbacks, however. The basic specs of their system are a rated power of 14kW of heating and a claimed battery capacity of 12,000 kWh, but the system is a geothermal installation type system. Meaning the sand battery component is a massive installation process which gives the flexibility of being able to store lots of power and be 'out of sight out of mind' but lacks in terms of affordability. The cost of the system alone is already setting back the consumer around $8000, we then need to consider the installation and permitting process of digging a massive hole in the back yard to then be able to run the proper piping into the house to connect to the home radiator and thermostat. This sets their system apart from our project as we intend on making the battery affordable for anyone that wants to heat their home, and easy to install for those that may not even be in a position to dig a massive hole in their backyard.

## 2.4 Engineering Specifications

Table 2.4.1 below shows the important specifications for this project, and we intend to demonstrate 3 of the specifications highlighted in blue.
The external temperature display indicates the current temperature outside of the thermal sand battery system. This information is important because it allows users to understand the environmental conditions and how they may affect the system's performance.

The internal temperature display shows the temperature of the sand inside the thermal battery. This is critical for monitoring the system's performance and safety, ensuring it operates within the specified temperature range of 0°C to 395°C.

The user interface, which consists of an LCD 3.5 inch, facilitates users to set charging times and schedule heating periods, making the system accessible and simple to use. Its significance comes from the fact that it provides a user-friendly interface for customizing the system's operation to meet individual requirements.

| Specification | Description | Range/number |
|---|---|---|
| Charging Time | The time during which the battery is charged is set by the user. | 3-10 hours |

| | | |
|---|---|---|
| Discharge Duration | Duration for which the battery can maintain ambient warmth. | 4-12 hours |
| User Interface | Interface allowing the user to set charging and discharge times. | LCD 3.5” |
| Battery | Shows the percentage of the battery on the screen. | 0-100% |
| Turn on/off | User is able to turn on/off overriding the schedule time. | LCD on/off button |
| External Temperature Display | Display showing the current external temperature. | 0°C to 55°C |
| Internal Temperature Display | Display showing the current internal temperature of the sand. | 0°C to 395°C |
| Temperature Range (Internal) | Range of temperature the internal sand can reach. | ~0°C to 395°C |
| Energy Capacity | Amount of energy the battery can store. | 5KWh |
| Power Input | Power is required for charging the battery. | 120V, 60Hz |
| Power Output | Max current/power drawn | 12A, 500W - 1KW per hour |
| Efficiency | The efficiency of the battery in converting electrical energy to heat. | 80-90% |
| Heat Retention | The ability of the sand to retain heat over time. | Up to 24 hours (if fully charged) |
| Safety Features | Built-in safety mechanisms such as overheat protection. | Overheat shut off at 450°C (internal) |
| Dimensions | Maximum dimensions | 36” x 24” x 24” Excluding wheel height |

*Table 2.4.1: Engineering Specifications*

## 2.5 Block Diagrams

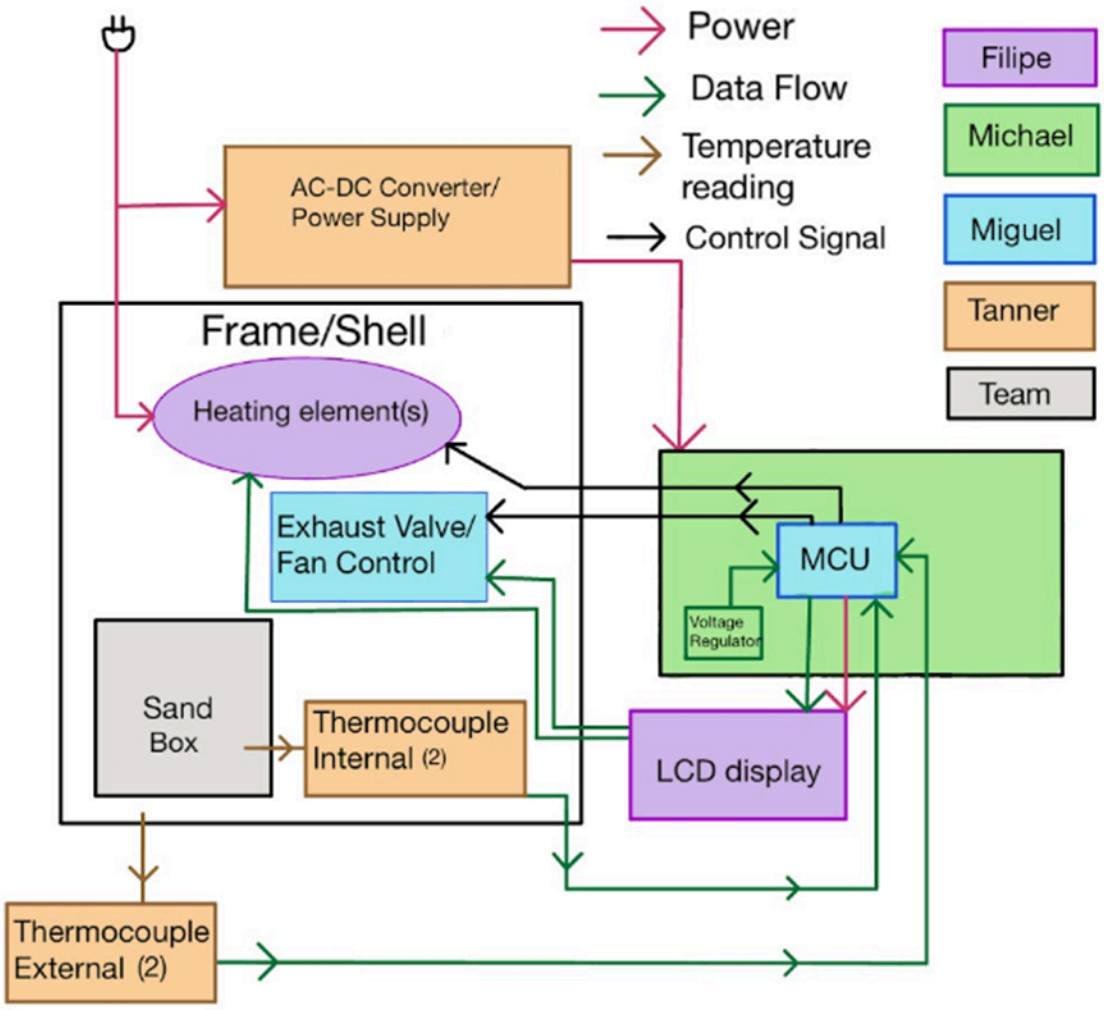



*Figure 2.5.1: Hardware Block Diagram*

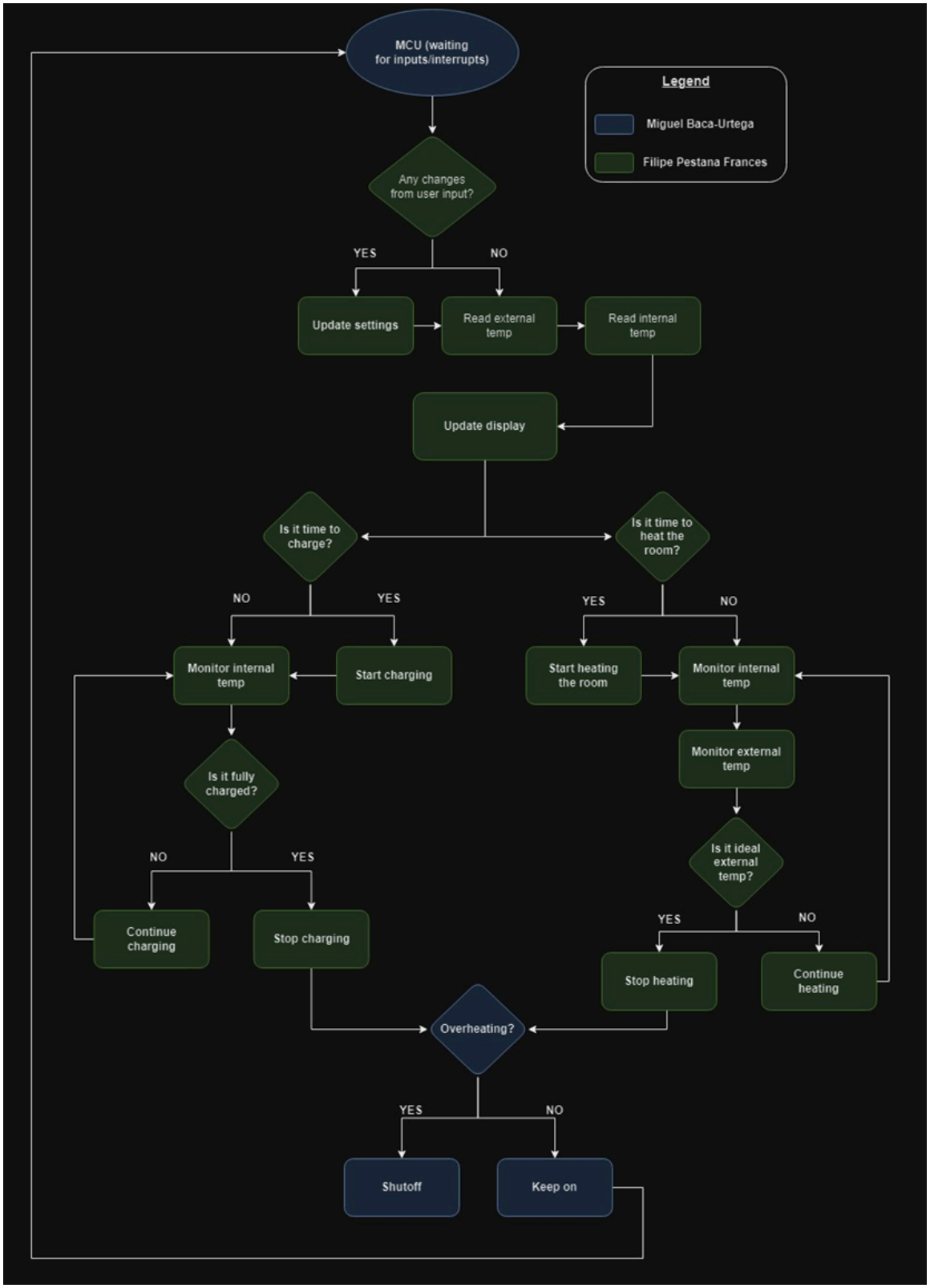



*Figure 2.5.2: Software Diagram*

**Main loop**

● *User Input*

**Purpose:** Determine whether the user has interacted with the system via the LCD.

**Actions:** If user input is detected, update the system settings (for example, change the charging time or heating time).

● *Read External Temp*

**Purpose:** Monitor the temperature of the external environment.

**Actions:** Record the current external temperature for display and decision purposes.

● *Read Internal Temp*

**Purpose:** Monitor the temperature of the sand inside the battery.

**Actions:** Record the current internal temperature to ensure that it falls within the desired range for charging and heating.

● *Update Display*

**Purpose:** Keep the user informed of the system's current status.

**Actions:** Display external and internal temperatures in real-time, as well as charging and heating status on the screen.

● *Charging Time*

**Purpose:** Determine whether it is time to begin the charging process based on the user-defined schedule and real-time clock.

**Actions:** Once the scheduled charging time is reached, begin the process. Continuously monitor the internal temperature while charging. When the desired temperature or time limit is reached, the charging process comes to an end.

● *Heating Time*

**Purpose:** Determine whether it is time to begin the heating process based on the user-defined schedule and real-time clock.

**Actions:** Once the scheduled heating time is reached, begin the process.
Continuously monitor the internal and external temperature while heating. When the desired temperature or time limit is reached, turn off the heat.

● *Overheat Protection*

**Purpose:** To ensure the system's safety by preventing overheating.

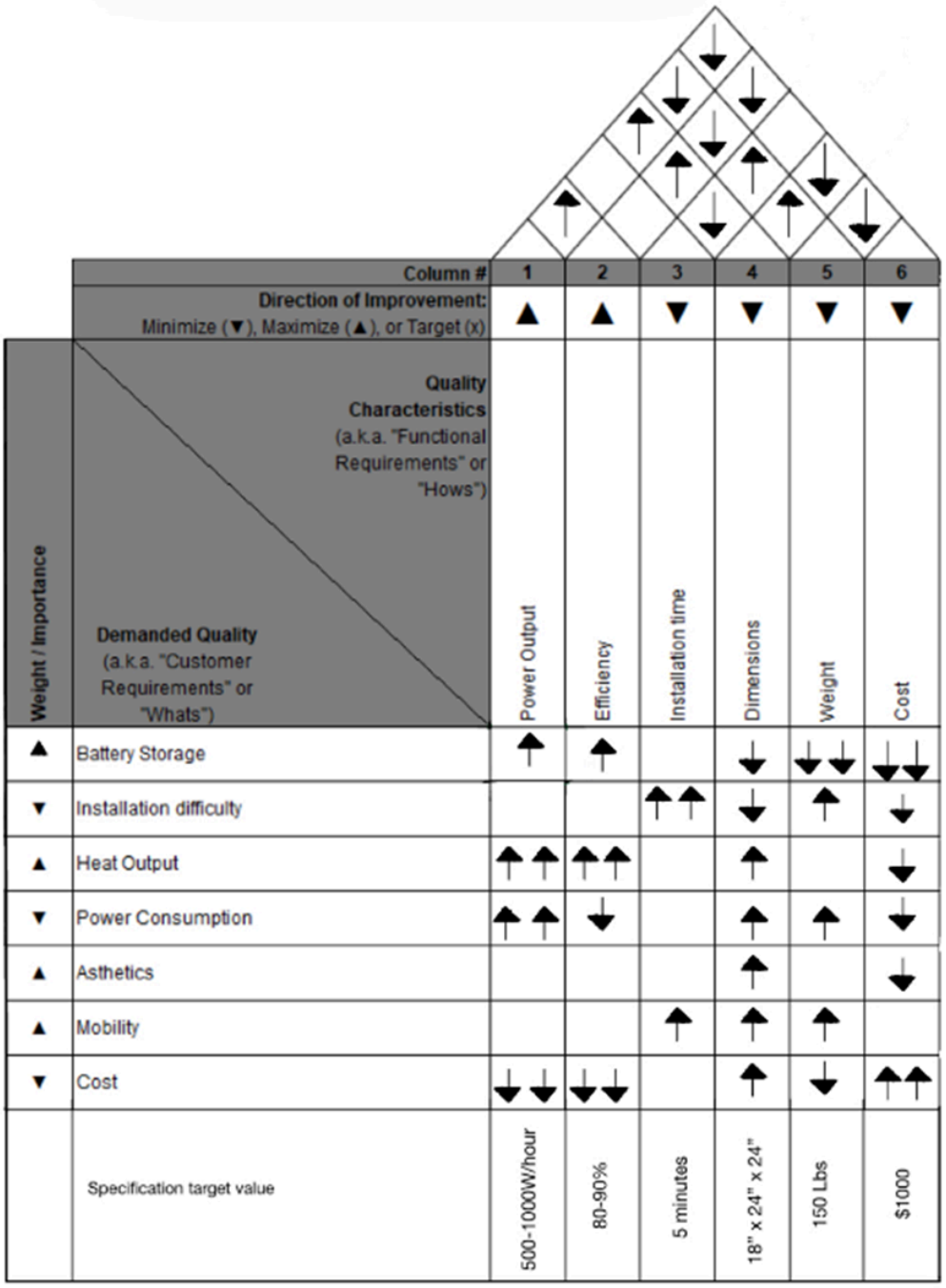
**Actions:** Constantly monitor the interior temperature for indicators of overheating. If overheating is detected: Shut down the system to avoid any harm or dangers.

● *Loop Back*

**Purpose:** Continuously repeat the look to monitor and regulate the system.

**Actions:** Return to the beginning of the main loop to perform the checks and updates.

## 2.6 House of Quality



| Weight / Importance | Demanded Quality (a.k.a. "Customer Requirements" or "Whats") | Power Output | Efficiency | Installation time | Dimensions | Weight | Cost |
|---|---|---|---|---|---|---|---|
| | Column # | 1 | 2 | 3 | 4 | 5 | 6 |
| | Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x) | ▲ | ▲ | ▼ | ▼ | ▼ | ▼ |
| ▲ | Battery Storage | ↑ | ↑ | | ↓ | ↓↓ | ↓↓ |
| ▼ | Installation difficulty | | | ↑↑ | ↓ | ↑ | ↓ |
| ▲ | Heat Output | ↑↑ | ↑↑ | | ↑ | | ↓ |
| ▼ | Power Consumption | ↑↑ | ↓ | | ↑ | ↑ | ↓ |
| ▲ | Asthetics | | | | ↑ | | ↓ |
| ▲ | Mobility | | | ↑ | ↑ | ↑ | |
| ▼ | Cost | ↓↓ | ↓↓ | | ↑ | ↓ | ↑↑ |
| | Specification target value | 500-1000W/hour | 80-90% | 5 minutes | 18" x 24" x 24" | 150 Lbs | $1000 |

*Figure 2.6.1: House of Quality Diagram*

# Chapter 3:Research and Investigation

## 3.1 Technology Comparison and Selection

### 3.1.1 - MCU or FPGA

Although it might be common to think that planning starts with the MCU, our approach actually needed us to work backwards. When we started planning what we wanted our project to achieve, as well as ensuring that stretch goals would be possible, we needed to find the components that would give us the desired final product and then after finding the parts that we would use we then start searching for an MCU that would be able to drive all the components as well as being more capable than what we currently need it for incase the stretch goals do come into reach.

In the final stages of the project planning we were at the crossroads of deciding between an MCU and an FPGA board. The biggest considerations we had was the cost of the overall systems. The benefit of MCU's being cheaper than FPGAs as well as consuming less power helps with the goals that we had been setting. The thermal sand batteries charging ability is one of the biggest priorities we have, so being able to save as much power as possible by having a custom PCB that draws less power was very important so we would be able to redirect as much power as possible into the heating element which we will use to charge the battery. The MCU we would be using would be ideally used for interfacing with an array of sensors, some relays, and an LCD we don't have any extreme calculations that we have to worry about and would much prefer not needing to actively cool the PCB just because we have an unnecessarily powerful FPGA that is also wasting power.

Accessibility and ease of use was another consideration we had for deciding between the FPGA or the MCU, although hardware descriptive languages can give significantly more control to a programmer when designing a custom PCB this adds an unnecessary layer of complexity which is unnecessary given the goals of the project at hand. A prime example would be that MCU's already have many foundational components that we would otherwise need to spend time recreating. With an MCU we would already have the ADCs, DACs, timers, UARTs, I2C, SPI, and PWM controllers. The ease of use of the MCUs would allow us to use the time we would otherwise spend reimplementing this functionality to instead be able to better develop features that we wouldn't be able to achieve if we had to start everything from scratch. MCU's bring us the benefit of extensive documentation on every aspect of the MCU's built in functionality, each MCU that we researched had corresponding programming libraries that would also streamline our software development process. The MCU already uses higher level languages that let us worry about software feature

implementation. The importance of having higher level languages cannot be understated in this project compared to hardware descriptive languages, with this we would be able to debug our programs a lot faster as a simple mistake would be easily corrected as opposed to needing changes to a larger overall FPGA design.

After a fair amount of research we were able to narrow down the MCU' we wanted to use. Although just about every MCU we found was likely capable of performing most of the tasks we had planned for this project we ended up narrowing down the selection by considering other features as well. A big need was to have many GPIO pins as well as a sufficient amount of pins that had ADC's, we also needed an MCU that may have sported features that put it above the other MCU's, considerations were bluetooth and wifi functionality, lastly just a passing glance at the amount of documentation for each MCU. It was fairly common to have older MCU's that may be cheaper that don't reach the computational capacity we need for this project, or even have documentation that is significantly outdated.

### 3.1.2 - PCB Design Software

Engineers and designers in the electronics industry rely heavily on printed circuit board (PCB) design software. These software packages make it easier to create, layout, and simulate printed circuit boards, allowing for the development of complex electronic devices. PCB design software simplifies the design process by providing features such as schematic capture, component placement, routing, and 3D visualization. The complexity of the project, user experience, budget, and specific requirements all go into play in the software selection process. In the following comparison, we will look at the benefits and drawbacks of the three popular PCB design software options: Altium, Fusion 360, and KiCad, to determine which is the best for our project. As we can see in the table 3.1.2, each of the software has advantages and disadvantages, which will depend on the project and user demands. Therefore, to decide which one would fit best for our project, we need to consider what features are the most important for a project to be concluded properly.

For example, Altium is perfect for professional and expert users who require a complete set of tools for complex PCB Design. Fusion 360 is great for projects that require 3D design capabilities and easy connectivity with other Autodesk applications. KiCad is appropriate for users looking for a free, open-source solution with a decent combination of capabilities and easy-to-use.

For our project, we have decided to go with Fusion360 because it is a balance between KiCad and Altium. One of its very important features that is built in for Fusion360 is that it allows us to easily transfer our schematic to the PCB layout. Also, it is strong software with 3D capabilities, offering advanced tools for creating and visualizing complex designs. Another feature is having a cloud-based architecture enables collaboration among the project team

members, which is important. It has a very intuitive and user-friendly interface, which makes it accessible for any person, even if you have less experience in PCB design. Lastly, Autodesk updates Fusion360 regularly, which allows users to have access to the most recent features and improvements. Therefore, by deciding to go with Fusion360, we will have access to a powerful, adaptable, and user-friendly tool that will help in our PCB design process.

| Specification | Altium | Fusion360 | KiCad |
|---|---|---|---|
| Flexibility | Highly adaptable, but may be too much for simple design | Less flexible than dedicated PCB tools | Flexible, but missing some advanced features |
| Complexity | Many features, but it can be too much | With 3D option will increase complexity | Less features than others |
| Features | Good set of features for advanced PCB design | Good selection of features for PCB design | Good set of features for a free software |
| Integration | Excellent integration with different tools | Strong integration with Autodesk tools | Effective integration with other EDA tools |
| Easy to Use | Intuitive interface, industry-standard UI | Intuitive interface, especially for 3D | User-friendly |
| Performance | Requires High-end hardware for max performance | Cloud-based nature, it may be slower | Efficient, and performs well on basic hardware |
| Updates | Frequently updates and new features | Improves regularly with updates and new features | Constantly improves by open-source community |

*Table 3.1.2: PCB Design Software*

### 3.1.3 - Environmental Development Software

Integrated Development Environments (IDEs) are important in software development because they provide programmers with extensive tools for creating software, such as a source code editor, building automation tools, and a debugger. Choosing the right IDE can significantly improve a development project's efficiency and productivity. The eas of use, feature set, compatibility with target platform, and project-specific needs are the most important things to help in the IDE selection.

After researching what IDE would fit better for our project, we have selected 3 IDEs that we would be comfortable to work with and we compared them in the table 3.1.3. As we can see, they all have pros and cons. Arduino IDE is ideal for beginners, it includes support for ESP32, it has a diverse set of libraries, it is simple, and it enables quick prototypes. However, it lacks advanced features for large projects. Visual Studio Code is for intermediate users because it requires some setup, but includes advanced features, and has many customization options and resources. However, it requires an extension for ESP32, and it can be difficult to configure for beginners. IntelliJ IDEA offers powerful tools for code analysis and debugging. Supports many languages and frameworks, which allows development across multiple technologies, and has extensive plugin support. However, it is very complex to set up for ESP32 and to be able to use all the features you must pay for the product, which may not be suitable for a small project.

Considering all the information provided above, and knowing that we will be using ESP32, we have decided to use the Arduino IDE because it is simple and easy to set up, which makes it ideal for our project. Having the built-in support for ESP32 is a big plus because it will allow us to get started quickly with minimal configuration. Also, having many libraries designed specifically for ESP32 and many screens will significantly speed up the development. Therefore, using Arduino IDE is an excellent choice for developing with ESP32 because it checks the balance between simplicity, ease of use, and support.

| Specification | Arduino IDE | Visual Studio Code | IntelliJ IDEA |
|---|---|---|---|
| Easy to Use | Beginner-friendly | Requires some setup | Complex |

| | | | |
|---|---|---|---|
| Resources | Many libraries available | Extensive resources and extensions | Extensive resources |
| Customization | Minimal | Highly customizable | Highly customizable |
| ESP 32 Support | Built-in | Platform IO extension | Via plugins |
| Initial Setup | Simple, quick setup | Requires configuration | Requires plugins and setup |
| Best For | Simple projects | Large projects | Complex projects |

*Table 3.1.3: Environmental Development Comparison*

### 3.1.4 - Programming Language Comparison

Choosing the most suitable programming language is an important decision in any development project, especially when working with embedded systems and microcontrollers. Each language has distinct advantages and disadvantrages that can have significant impact on efficiency, performance, and easy of development. In the following comparison, we evaluated three programming languages to determine which one has the the best suited for our project. To make the decision, we took into account performance, memory usage, easy of learning, and suitability for embedded systems.

We've chosen 3 of the programming languages that we could possibly use for our project, and as you can see in the table 3.1.4, we have compared these 3 to make sure we use the most suitable for our project. Python is readable and easy to learn, which makes it great for rapid development due to its extensive library. However, Python has the slowest performance compared to the other two, and its high memory usage makes it unsuitable for our project. C++ has powerful features like object-oriented programming and an extensive standard library. These features make it suitable for both system and application development. However, the language's complexity compared to C might result in higher memory utilization, and it could slower execution in embedded systems, which is a negative point in resource-constrained applications. C delivers efficient low-level access to hardware and memory, making it suitable for high-performance and resource-constrained applications. Its simplicity and procedural nature enable control and optimization, which is critical for embedded systems. However, C lacks modern programming capabilities such as

object-oriented techniques, making code administration and maintenance more difficult.

Having all these considerations, C is our choice for programming this project because it combines fast performance, low memory usage, and an extensive embedded system library to create a reliable and effective language for this development. While Python excels in high-level programming and rapid development, and C++ provides advanced object-oriented programming tools, C strikes the ideal balance for embedded system and microcontroller programming.

| Criterion | Python | C++ | C |
|---|---|---|---|
| Easy to Use | High, simple syntax | Moderate, complex syntax, object-oriented | Moderate, simple syntax, procedural |
| Library | Extensive for general programming | Extensive for general and system level | Moderate, but extensive for embedded systems |
| Performance | Low, interpreted language | High, compiled language | High, compiled language |
| Memory Usage | High, less efficient | Moderate, efficient | Low, very efficient |
| Control | Low, high-level abstractions | High, OOP features | High, low-level access |
| Development Speed | Fast, many libraries | Moderate, requires more code than python | Slow, requires detailed and specific coding |
| Suitable for Embedded Systems | Low, barely used for microcontrollers | High, widely used for microcontrollers | High, widely used for microcontrollers |

*Table 3.1.4: Programming Language Comparison*

# 3.2 Parts Comparison and Selection

## 3.2.1 - MCU Comparison

Microcontroller unit (MCU) selection is critical decision for the success of any embedded system project. The MCU functions are the device's "brain", where it handle all process tasks, communication with peripherals, and ensure efficient operation. Different MCUs provide varying levels of performance, power consumption, memory capacity, and peripheral support, which leaves the choice based on the project's specific requirements. In order to make the decision, we look into processing power, peripheral support, power consumption, and the overall development ecosystem.

After a long research, we managed to narrow down the MCU selection to just 3 MCU's, as you can see in the table 3.2.1. The first option is MSP430FR6989, which has an ultra-low-power application due to having an energy-efficient design and rapid, long-lasting FRAM memory. It is easy to use because it has very good documentation to help to set up. However, it lacks processing power, and the memory may be insufficient for larger applications. The second option is ESP32, which has the best processing power of the 3 options, and comprehensive peripheral support, which makes it perfect for demanding applications. However, this MCU requires more power, which is a downside. The third and last option is RP2040, which has a balance between performance and affordability. However, it has fewer libraries and resources compared to ESP32-S3 due to being the newest MCU between these 3.

Having all these considerations of pros and cons between these 3 MCUs, we have decided for our project to go with ESP32 because of the strong development ecosystem, comprehensive peripheral support, and high computing power for precision. Also, the cost-effectiveness makes it an appealing option for our project.

When reviewing the documentation there were a few things we wanted to point out which caused us to lean more in favor of the ESP32. According to the RP2040 datasheet the RP2040 has 36 multifunctional General Purpose Input / Output (GPIO) pins, this is notably less than the 48 total pins that are delivered with the ESP32-S3 (notably some of the pins are restricted due to 'allocation for communication with in-package flash/PSRAM and not recommended for other uses', but this only drops our available pins to 41). The almost excessive amount of GPIO pins was vital in our planning for possible future expansion. With all these pins also comes the second consideration we may have, this is a surplus of ADC compatible pins. Within the RP2040 datasheet of the 36 pins that they have 4 are ADC inputs which would be used for us to measure input from the thermometers inside and around the battery, although we didn't really plan on having more than 4 temperature sensors overall this still limits any possible future expansion which may happen as we refine the project. When looking

through the ESP32-S3 datasheet we found that we were offered 20 ADC compatible pins, and if we are using wifi along with the ESP32 then that number only drops down to 10 ADC available pins. When reading more on the ADC system the ESP32 has it seemed like it was split into two different tiers, ADC1_CH… pins were universally available, and it was only ADC2_CH… pins which would be disabled if there is an ongoing wifi connection, and thankfully we didn't have any plans to make our device wifi enabled so we can certainly count on 20 ADC pins being available and 10 at a minimum. With the minimum of 10 available ADC pins we'll be afforded a lot more flexibility should the project need more than the 4 temp sensors we're currently planning on including. At the very least we will be able to add other ADC components

Although the ESP32 does take more power than the RP2040, with its dual core 240 MHz Xtensa processor we were able to give ourselves a sort of computing headroom that we wouldn't have been able to have with the RP2040. When planning the program we were interested in implementing a multithreaded approach to the application which would be responsible for handling all the peripheral components and also driving the LCD. The RP2040 is sporting a 133 MHz dual core ARM Cortex processor with 264 kB of on-chip SRAM whereas the ESP32 almost doubled that amount with 512 KB of on-chip SRAM. We wanted to ensure that we went with a powerful MCU so we would ensure that we would have no issues with driving the over 150k pixels on the touch screen display we have for the user to interact with. By having the 240MHz processor and the larger SRAM capacity we can have confidence that the UI will be able to operate smoothly for the user while also having plenty of computational headroom incase several tasks may be happening at once, or if we get to a point where we can refine our code more to then give room for later updates.

When we started narrowing down our options a big feature we had noticed with the ESP32 that wasn't very prevalent to other MCU's- and that we have alluded to earlier- was the built in wifi and bluetooth capabilities of the ESP32. Although we don't have anything thats exactly planned in terms of using the bluetooth and wifi capabilities it does align with the goal that we've had with the overall MCU selection, that being a low power, modern, and capable MCU that has more than enough features to allow us the ability for future project development/expansion where the opportunity presents itself. The ability of having this sort of wiggle room in our project can also give us the space to adjust the project as we continue the design process.

| Specification | MSP430FR6989 | ESP32 | RP2040 |
|---|---|---|---|
| Processor | 16-bit RISC | 32-bit Dual-core Xtensa LX7 | Dual-core ARM Cortex-M0+ |

| | | | |
|---|---|---|---|
| Clock Speed | Up to 16 MHz | Up to 240 MHz | Up to 133 MHz |
| RAM | 2 KB SRAM | 512 KB SRAM | 264 KB SRAM |
| Flash Memory | 128 KB FRAM | Up to 4 MB external | Up to 2 MB external |
| Power Consumption | Low | Moderate to High | Moderate |
| GPIO Pins | 47 | 45 | 30 |
| Development Ecosystem | TI CCS, Energia | Arduino, PlatformIO | Arduino, MicroPython |
| Price | $8.68 | $3.35 | $7.74 |
| Best For | Battery-powered applications | IoT, Complex tasks | Versatile applications |

*Table 3.2.1: MCU Comparison*

### 3.2.2 - Display Comparison

Finding the appropriate display screen is an important step in creation of any user interface-driven project. For this project, we took into account compatibility with ESP32 chip, touchscreen functionality, power consumption, and voltage requirements. The ideal display must integrate seamlessly with our custom PCB and meet our design criteria for usability, power efficenty and overall performance. Based on these criteria, we've narrowed down to 3 options that we carefully selected for our project. As you can see in the table 3.2.2, we've made a comparison to select the one that would fit best. Researching a display is not an easy task since we have to consider that we are going to be using a custom PCB with ESP32. So, the first thing that we've had to keep in mind is that it should be able to work with the ESP32 chip. The second thing is that we wanted a touchscreen display to allow end users to change the charge and discharge time easily. Lastly, we have to consider power consumption and voltage that would fit great for this project.

Using these parameters mentioned above, we have chosen the Hosyond 3.5inch IPS display because it has an onboard conversion circuit, which makes it compatible with converting from 5V to 3.3V for ESP32, and we will be using SPI communication. The touchscreen being capacitive makes it easy for end users to use for precision and sensitivity, and the power consumption is very low, which will help to save energy for the project.

When researching and reading through the datasheet of the Hosyond there were a few things we found that were very notable which is why we opted to choose this screen. The first and one of the most important aspects was that the maximum power draw of the screen was at only half a watt of power. This is great because the goal of our overall system will be to divert as much power as possible towards the heating element which will be used to charge the battery. The reason we need to be careful with the power utilization of our system was that most 120v outlets in the US are traditionally built with 15 amp circuits, meaning that we will be able to pull a maximum of about 1500 watts if there are no other loads on the same circuit. The system design is also being made so that the system will only pull a maximum of 1000 watts, this will allow other smaller components to run on the same circuit so we don't monopolize a user's entire circuit. Along with the power consumption of the LCD we did also like that the LCD had a level conversion circuit which would make the LCD compatible with both 3.3v and 5v power supplies. This conversion circuit simplifies the overall implementation of the system as we wouldn't need an intermediate buck-boost converter which would just introduce more complexity to our PCB as well as more power losses and waste heat that we would have to manage, and so by having its own ideally optimized 3.3v power conversion this would free up more time for us to focus on overall system design and implementation.

The factor that made this screen stand out for us was one very interesting integration component which is that it has a built-in micro SD card slot. One thing we didn't touch on with the MCU section of the research is that the ESP32 natively supports SD card connections. The ESP32 has an integrated SD device interface that aligns with the 'industry-standard SDIO card specification Version 2.0' which would allow us to use the SDIO bus interface. According to the datasheet for the ESP32 we would be able to directly access the registers of the SDIO interface and also access the shared memory via a 'DMA engine' which maximizes the overall system performance without engaging the processor cores. The built in controller will let us work with an up to 80 MHz clock output in 1-bit, 4-bit, and 8-bit modes. One important consideration we want to have is that the one SD card will be operating at a 1.8V capacity. So with the ESP32's native SD card support and the ability to connect an SD card to the LCD to store images and text fonts for the LCD

The LCD and the built in ST7796S controller supports a maximum display resolution of 320x480 and has a built in GRAM of a345600 bytes. Each pixel will be able to display up to 65,000 colors thanks to the 16 bit color provided by the ST77976S controller. This LCF is centered around the SPI communication

protocol.The C/CX is the built in data control pin for the LCD and will be writing commands when the DCX pin is pulled low and data will be received when high. With the SCL being the SPI bus clock the rising edge will be transmitting 1 bit of data. The LCD will have 14 pins that we have to account for, either in the form of actual break out pins or in the form of a ribbon cable that we will have to figure out how to connect to the PCB we are designing. Although the ribbon cable would greatly minimize the footprint of our overall system design we need to consider how this can make the overall project more delicate, this means that we would have to take a more robust enclosure design for the MCU/thermostat enclosure eo ensure that the ribbon cables are properly secured.

The LCD has 14 pins that we will have to integrate as connections that the custom PCB will have to interface with. The VCC will supply power, the GND will obviously be the ground connection, LCD_CS will be the active low control signal, LCD_RST is the low level reset control signal, LCD_RS is the LCD common and data selection control signal, SDI is the MOSI SPI bus write signal which is actually utilized by the SD card and the LCD screen, SCK will be the SCK bus clock signal which is also used by the SD card reader and the LCD screen, LED is the LCD backlight control signal (this pin is actually optional given if we want to have the ability to control the backlight which we will want to be able to control it), the SDO(MISO) which is the SPI bus read data signal (also used by both the SD card reader and LCD screen), CTP_SCL which is the capacitive touch screen IIC bus clock signal, CTP_RST capacitive touch screen reset control signal, CTP_SDA capacitive touch screen IIC bus data signal, the CTP_INT which is the IIC bus touch interrupt signal which is used when generating touch, and lastly the SD_CS which is the SD card selection control signal. The benefit that we are having is the sheer amount of control this LCD gives us. When we were looking for LCD's that would be able to interface with the ESP32 we found a lot of screens that already had integrated MCU's, the problem with trying to have our PCB interface with essentially another PCB thats driving the screen we're wanting to control is the lack of customizability. This LCD gives us the ability to design and optimize the system, we won't have to hope that an unknown third party implemented an efficient backend.

With something as seemingly simple as the LCD already taking up 14 pins on the ESP32 our decision to get an MCU with more pins than we likely needed continues to prove as the right move. A final benefit to consider is that the ESP32 has 3 different power domains originating from its digital pins. The pins being VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO. The VDD3P3_RTS is also serving as the input supply for the RTC and CPU, VDD3P3_CPU which is also serving as input power for the CPU, and the VDD_SDIO which 'connects to the output of an internal LDO whose input is VDD3P3_RTC', the absolute maximum power output rating from the pins is 1200 mA which far exceeds the 95mA current draw from the LCD backlight, this simplifies our design even more as we would be afforded the option of either just connecting the LCD directly to the 3.3v power supply pin or even giving us the ability to make a 3.3v bus bar that all powered components would need such as the thermocouples, any relays, or valve motors.

| Brand | Waveshare | Adafruit | Hosyond |
|---|---|---|---|
| Model | 4.3inch DSI QLED | 3.5inch TFT Touch | 3.5inch IPS SPI |
| Resolution | 800x480 | 480x320 | 480x320 |
| Display Size | 4.3 inches | 3.5 inches | 3.5 inches |
| Communication | DSI | SPI | I2C, SPI |
| Touchscreen | Capacitive | Capacitive | Capacitive |
| Power Consumption | 1.2 watts | -- | 0.5 watts |
| Price | $45.99 | $39.95 | $18.99 |
| Voltage | 3.3-5V | 3.3V | 3.3-5V |

*Table 3.2.2: Display Comparison*

### 3.2.3 - Thermocouple Comparison

Our goal to measure the internal temperature of the sand came with a few challenges that we needed to overcome, the first being accuracy, response time, and size dimensions. It is easy to measure temperature, but it is difficult to measure temperature accurately. We chose to use the MAX6675 K-type thermocouple to measure the temperature of the sand. K-type thermocouples span approximately -300 to 2300 degrees Fahrenheit of measurable temperature. The k-type temperature range was a convenient range for our project which would reach temperatures between -100 to 1000 degrees Fahrenheit in the product's extreme conditions. The MAX6675 is capable of measuring accurately to 0.25 °Celsius, or 0.45 °Fahrenheit. We compared this to other thermocouples compatible with our ESP32 board such as the MAX31856 and the MAX31865.

The MAX31856 is a precision thermocouple which is accurate to .0078125° Celsius from -210° C to 1800°C. We were considering this thermocouple as it would measure the complete temperature range that our project would operate in, however we ruled out this component because it was a breakout board not containing an actual thermocouple, where the MAX6675 is a thermocouple which could connect directly to the ESP32 board we used. The MAX31856 would involve one more board and be less cost efficient than the MAX6675 thermocouple. The MAX31856 rang in at $27 when we were doing our research compared to the 6675 at $8. The MAX31856 is also less flexible in supply voltage rating 3 to 3.6V, whereas the MAX6675 has a less accurate resolution of .25° Celsius but is able to use a supply voltage from -.3V to 6V. This was very motivating for us as our PCB has a 3.3V layer and a 5V layer, and the flexibility to have either layer of the board power the thermocouple was a large factor for us as we were still settling on the design/size needs of the PCB.

The MAX6675 being SPI (Serial Peripheral Interface) compatible was another contributing factor in deciding on using the MAX6675 thermocouple given that our ESP32 board has 4 SPI ports available. We considered the MAX31865 as well, not to be confused with the MAX31856. The MAX31865 had similar issues to the MAX31856: it was only compatible with voltages between -.3V and 4V, and like the MAX31856, the MAX31865 included anther board that would add extra components to our circuit increasing the footprint. The 31865 measured temperature via a resistance to digital converter (RTD) which often has a slower response time compared to thermocouples, and RTDs are more susceptible to EMI/noise interference. The MAX31865 was the most cost efficient at $1.99 per unit, but the design team concluded that the MAX6675 would be the best choice in the design due to its simplicity, response time, temperature range, and temperature resolution.

| Brand | Maxim Integrated | Maxim Integrated | Maxim Integrated |
|---|---|---|---|
| Model | MAX6675 | MAX31856 | MAX31865 |
| Price | $7.99 | $26.99 | $1.99 |
| Type of thermocouple | K type | K, J, N, R, S, and T type | K, J, N, R, S, and T type |
| Temperature Range | -200ºC to 1024ºC | -210ºC to 1800ºC | -200ºC to 1024ºC |
| Digital Interface | SPI(Serial Peripheral Interface) | SPI(Serial Peripheral Interface) | SPI(Serial Peripheral Interface) |
| Power Supply Voltage | 3.3 or 5V | 3 to 3.6V | -0.3 to 4V |

| | | | |
|---|---|---|---|
| Resolution | 0.25ºC / 0.45ºF | 0.0078125ºC / 0.01283ºF | 0.0078125ºC / 0.01283ºF |

*Table 3.2.3: Thermocouple Comparison*

### 3.2.4 - Onboard Temperature Sensor Comparison

For the ESP32's onboard temperature, we considered multiple sensors which could read an accurate temperature while being financially responsible. The first of the sensors was the Adafruit MCP9808 I2C Temperature Sensor. This sensor involved a breakout with 6 pins. It sees temperature resolution at 0.0625° C or 0.1125°F, and has a temperature range from -40°F to 257°F. The team decided that we did not want the PCB to exceed 70°C (158°F). This sensor communicates via I2C, which is compatible with the ESP32 controller in our project. It has a voltage range from 2.7V to 5.5V which is compatible with both VCC layers of our PCB, it required only 200uA to power, and the cost per sensor was $5 at the time of purchase.

The next board we considered was the DHT11 Temperature Humidity Sensor Module. This sensor operates on 3.3V-5V meaning this sensor is also compatible with both VCC layers of our PCB. It measures humidity in addition to temperature with a humidity range of 20%-95% and a temperature range from 32°F to 122°F. This temperature range does not extend to our desired 158°F, but it covers safe handling temperatures and is close to the top end of the desired range. Its resolution is within 1.8°F and 1% humidity, which would provide enough resolution for its intended use in the thermal sand battery. The dimensions of the DHT11 sensor are 1.1" x 1.1" x 0.47", which is well within reason for our design, and is connected via 3 pins. This device rang in at $2 per unit ($10 for a pack of 5), which is extremely cost effective.

Similar to the DHT11 sensor, we considered the DFROBOT SEN0497 Sensor board. This board is designed to be compatible with the ESP32 and other Arduino/Raspberry Pi boards. It operates on the same voltages as the DHT11 (3.3V to 5V), and it measures temperature alongside humidity. This board communicates via I2C through a 4 pin connector. The downsides to this board were cost-effectiveness, as it cost $18 at the time of purchase, and the temperature/humidity ranges had not been calculated at that time, and we did not have time to experiment.

We examined the Adafruit TMP235 Temperature Sensor Expansion Board. The device's dimensions are 1" x 0.7" x 0.3" weighing less than an ounce, and connects with 3 pins to a JST connector. This sensor is unique in that its output is analog, so it must work with a microcontroller board that has an analog input. Luckily the ESP32 can interpret analog signals. The Adafruit TMP235 is powered by 3V to 5V DC power, perfect for our designed PCB. To read the temperature from this sensor we would need to calculate the voltage output ourselves. The values are from 0V at -58°F to 1.75V at 257°F. This range is perfect for our

intended use of the thermometer as the intended environment ambient temperature would be between 0°F and 125°F. This would involve more overhead in regard to man hours spent in development. The cost per unit for the Adafruit TMP235 was $2.50 at the time of purchasing, which is a competitive price with most of the models we researched.

The last temperature sensor we researched was the Adafruit AHT20 Temperature and Humidity Sensor. This sensor was priced at $4.50, and unlike the Adafruit TMP235, it communicates by I2C with the ESP32. It is rated for 3.3V to 5V as desired, and operates between -40°F and 185°F as well as from 0% to 100% humidity. Its resolution is +-0.55°F over the entire temperature range, with a +-3% accuracy in regard to humidity. These resolutions/ranges made this a promising candidate for the project, not to mention it is relatively the size of a U.S. quarter.

The team decided that the DHT11 sensor would be the best option for our needs. It provides useful temperature and humidity ranges while remaining very cost-effective. This gave us the option to include multiple in different locations within the sand battery at a low cost.

| Model | Adafruit MCP9808 | DHT11 | DFROBOT SEN0497 | Adafruit TMP235 | Adafruit AHT20 |
|---|---|---|---|---|---|
| Type | Digital Temperature Sensor | Digital Temperature / Humidity Sensor | Sensor Board (Analog) | Analog Temperature Sensor | Digital Temperature / Humidity Sensor |
| Price | $4.99 | $9.99 per 5 pack | $17.99 | $2.50 | $4.50 |
| Communication Interface | I2C | Single-Wire Digital | Analog Output | I2C | I2C |
| Temperature Range | -40ºF to 257ºF | 32ºF to 122ºF | -40ºF to 185ºF | -40ºF to 257ºF | -40ºF to 185ºF |
| Temperature Resolution (Accuracy) | 0.1125ºF | 3.6ºF | Variable | 0.18ºF | 0.55ºF |
| Power Supply Voltage | 2.7V to 5.5V | 3.3V to 5.5V | 3.3V to 5V | 3.3V to 5V | 1.8V to 3.6V |
| Humidity Range | N/A | 20% to 95% | N/A | N/A | 0% to 100% |

| | | | | | |
|---|---|---|---|---|---|
| Humidity Resolution | N/A | 1% | N/A | N/A | 3% |

*Table 3.2.4: Onboard Temperature Sensor Comparison*

### 3.2.5 - Sandbox Insulator Comparison

Insulating the sand thermally we knew was an important aspect of the project due to the safety requirements involved, so we chose our insulator very carefully. Our goal was to insulate the sand with a material rated for at least 2000º Fahrenheit. The sand itself is only to reach 1000º Fahrenheit, leaving a 1000º temperature buffer for electronics and sensor inaccuracies, delays, or malfunctions. We considered many types of insulation including fiber ceramic panels, unitherm polypropylene Fabric Insulation, fiber ceramic insulation blanket, brick box, and aluminum coated polyethylene insulation.

We started by looking into the fiber ceramic panels as we thought it could be a good insulator for the sand as it was recommended for applications such as industrial furnaces, kilns, ovens and other devices where high temperatures are utilized. The panels provide great thermal insulation allowing for the sand container to maintain a high heat without dissipating the heat through the panel. The panels are rated for up to 2700º Fahrenheit, almost three times higher than our goal of 1000ºF. These panels would provide a rigid and sturdy support to the sandbox, they would be resistant to thermal shock, and they would be easy for our team to replace while in the development stage should they need to be replaced or altered. Ceramic fiber is very resistant to fire by nature which would reduce fire hazards of the device if any electronic components were to malfunction. These panels were priced at $59.99 per 18 inch x 24 inch by 1 inch and the design would require 4 of these panels to insulate the sandbox if we were to keep our originally proposed dimensions.

We considered unitherm polypropylene fabric insulation as well. This fabric is lightweight, compared to the ceramic panels. We considered this fabric as our product is meant for consumer homes where weight is a concern. The unitherm fabric is flexible and can conform to irregular shapes which was another positive of using this material as the insulator. Polypropylene has moisture resistance inherent traits, so this would be useful for outdoor applications. The polypropylene fabric, while cost-effective, is not heat resistant past 250ºF, meaning we were not able to use it to insulate the sandbox to the 1000ºF required. This material was priced at $40.08 per 5 lbs box. The team estimated we would need two boxes to support our needed amount of insulation for the sandbox, which would become $80.16.

The fiber ceramic insulation blanket was also considered by the team to insulate the sandbox. This “blanket” has many of the same properties as the fiber ceramic panels such as thermal insulation, easy installation, cost effectiveness, and

versatility. The main differences stem from the blanket being less rigid and more flimsy than their counterpart. This turned out to be a useful quality as there were many surfaces of the sandbox that were not squared off. We did not need the aesthetic uniformity of flat panels either as the insulation would only be used inside the main shell/body of the thermal battery. Both types were rated for temperatures far higher than what we needed. This blanket came in many sizes, and the best option for the team was a 60 inch by 24 inch by 1 inch blanket priced at a reasonable $53.99.

Another method we considered was to use brick as the insulator. This method posed multiple issues: brick weighs more than ceramic panels/blankets, it is substantially larger than its counterparts, and brick tends to break down around at a much lower 1200ºF. Brick insulator turned out to be more expensive and time consuming than the ceramic materials, so while it met some of the requirements, it appeared to be a less practical method of insulating the sandbox than some of its counterparts.

Lastly, aluminum coated polyethylene insulation was a material we considered due to its similarities to the fiber ceramic insulation blanket. This product was quickly ruled out as its heat resistivity did not meet the requirements, spanning only -20° C to 90°C. We considered this product for insulation of the air exhaust piping as well due to its self-adhesive quality making it easy to install, but we were able to find a solution with more heat resistivity. Overall the team chose to use the fiber ceramic insulation blanket, as it met the requirements for heat/flame resistivity, installation simplicity, immunity to thermal shock, all while being relatively inexpensive.

| Type of insulation | Fiber Ceramic Panels | Unitherm Polypropylene Fabric | Fiber Ceramic Insulation Blanket | Brick Structure | Aluminum Coated Polyethylene Insulation |
|---|---|---|---|---|---|
| Temperature resistance | Up to 2700ºF | Up to 250ºF | Up to 2700ºF | Up to 1200ºF | Up to 200ºF |
| Fire resistance | High Resistivity | Moderate to Poor Resistivity | High Resistivity | High Resistivity | Poor Resistivity |
| Conformity to shapes | Limited to Flat Surface | Good Maliability | Very Good Maliability | Very Poor | Good Maliability |
| Ease of Installation | Moderate Difficulty | Easy | Easy | Difficult | Easy |

| | | | | | |
|---|---|---|---|---|---|
| Structural support | Poor Support | Very Poor Support | Very Poor Support | Very Good Support | Poor / Moderate Support |
| Cost | $59.99 per 18 inch x 24 inch | $40.08 per 5 lbs | $53.99 per 60 inch x 24 inch x 1 inch | Roughly $50-$70 | $32.49 per 10.8 square feet |
| Flexibility | Rigid | Very Flexible | Flexible | Rigid | Flexible |
| Intended Function | Furnace Lining | HVAC Duct Insulation | Furnace Lining | Structural walls/Fireplaces, Paving / landscaping | HVAC |

*Table 3.2.5: Sandbox Insulator Comparison*

## 3.2.6 - Insulated Cabling Comparison

For the electrical components inside and nearby the sand, we needed insulated wire resistant to heat, reducing the likelihood of components malfunctioning. Standard wires are susceptible to fire/heat damage, oxidization, and metal fatigue (where thermal expansion/contraction causes the wire to become brittle). We researched the following types of insulated wire to prohibit any of these conditions from happening: Mica Glass Fiber Insulated Electronic Wire, Double Insulated Flameproof Cable (2x2.5mm), 12 AWG Tinned OFC Oxygen Free Copper Stranded Wire, and Insulated Stranded Copper- THHN/THWN Wire 10 gauge.

We decided to research cables 12 AWG and under due to the need to be consistent with the 12 AWG standard in residential power outlets for the United States. We began our research with the Mica Fiberglass Insulated wire. We found this type of wire in 11 AWG at 26.3’ length. Wrapped around the Copper wire is a layer of fiberglass, a layer of Mica, and a final outermost layer of fiberglass. Mica tape is known to be an thermal insulator as well as fire resistant and durable, meaning this type of insulation will limit heat exposure of the copper wire while being resistant to cable damage due to bending. We priced the cable as $22.99 per 8 meters (26.3 feet). This cable can withstand up to 932°F Instantaneous temperature and a sustained temperature between -76°F and 660°F, allowing us to run the cable adjacent to the insulation (outside the sandbox). The stranded copper is a better alternative to solid copper as it provides more flexibility, conductivity(if used in high frequency applications, and durability, needless to say The results from researching this product were very promising.

Next we researched the Double Insulated Flameproof Cable 12AWG. This insulated wire included 2 individual wires wrapped together for a positive and negative lead. At $1.13 per meter, this option would allow for customization with less waste of material and installation time/difficulty would both be improved significantly. This cable has more dense strands than the previous option we explored, making it the more rigid option. With a maximum temperature rating of 194°F, we would not be able to run this wire close to the heated sand container, however it is rated for a much higher voltage (300V) which could be useful if the project adapted to have a higher voltage rating than 120V. The maximum short-circuit temperature of this cable is around 480°F, which still does not protect it from the 1000°F sandbox.

We found another 12AWG cable, tinned OFC Oxygen Free Copper Stranded, which we considered as a prospective option for our insulated cables. This option was priced at $29.48 per 50 feet, however it is a single wire so we would have to double the length of cable needed to electronically wire the components in our design. This cable made with resistant silicone coil insulation, is rated for temperatures between -76°F and 392°F. With the copper being stranded with an estimated 660 individual strands, the durability of this product stood out when compared with its competitors. Being stranded wire also made flexibility of the silicon wire stand out. The copper is also coated with tin, making the soldering a quick and easy task and lowering the production time of the sand battery. The cable is rated for 600V and used often for automotive design, power supplies, and grounding rods. At a rating of 99.9% oxygen free wire, this cable is less susceptible to oxidation over time, it should maintain better conductivity, and less oxygen typically means the product will have a better heat resistivity. These characteristics made silicone coated wire an excellent option for its possible applications within our design.

The final wire we researched/considered was the Insulated Stranded Copper THHN / THWN wire 10AWG. This cable priced at $79.97 per 10 feet, included a nylon jacket. Nylon is often used to protect against heat, gasoline, moisture, oils, and gives it flexibility while bending. The cable is only rated for up to 90°C and 600V capacity. It is constructed from 19 individual strands which makes it less flexible than the other stranded cables we considered while maintaining more flexibility than a solid copper cable. Ringing in at ~$8 per foot, with no option to customize cable length, and a limit of 194°F, this cable is not the most cost-effective for our design purposes and would not handle the temperatures needed for the product.

The design team decided the best option to use inside the thermal battery would be the Mica/fiberglass insulated wire due to its superior heat resistivity and its flexibility which allows for design alterations and proximity to hot components of the battery. Due to the small size of the thermal battery, our goal with the cabling was to use less than the 26 feet.

| Type of Insulated Wire | Mica Glass Fiber Insulated Electronic Wire | Double Insulated Flameproof Cable | Tinned Oxygen Free Copper Stranded wire | Insulated Stranded Copper THHN/THWN |
|---|---|---|---|---|
| Gauge | AWG 12 | AWG 12 | AWG 12 | AWG 10 |
| Price | $22.99 per 8 meters | $1.13 per meter | $29.48 per 50 feet | $79.97 per 10 feet |
| Voltage Rating | Variable | 300V | 600V | 600V |
| Temperature Rating | 932°F | 480°F | -76°F to 392°F | 194°F |
| Advantages | Excellent Insulation | Fire Resistance and Durability | Corrosion Resistance and High Conductivity | Oil and Heat Resistance, Durability |
| Intended Function(s) | Electronics, Military and Aerospace equipment | Industrial Machinery and Appliances | Audio Equipment | Commercial and Residential Wiring |
| Conductor Material | Copper | Copper | OFC Copper | Copper |

*Table 3.2.6: Insulated Cabling Comparison*

## 3.2.7 - Exhaust/Intake Valve Comparison

Controlling the airflow from the sand battery is an important part of the project that required controllable valves which could release hot air when the user turned the heating element of the device on. Essentially our design contained heat from the sand while “charging”, and released the heat when instructed. We researched components such as HVAC steel air dampers, PVC Air Volume Control Valves, PVC Exhaust Fan Dampers, and 1 inch Stainless Steel Motorized Water Valve.

From our research we gathered that most electric air valve dampers are offered in 220V, 24V and 12V. The one we researched was the 201 stainless steel electric motorized air valve. It was offered in the following sizes: 80mm, 100mm, 125mm, 150mm, and 200mm, which are approximately 3-8 inch. It is powered via a solenoid at low gas pressures, and is a normally closed device. The device, in layman's terms, twists a valve to let air flow through the pipe when triggered on, otherwise it acts as a cap in the center of the pipe. When the valve is closed, it uses its silicone sealing ring to cut off air flow around the edges. To switch states from on-off or off-on, the device must be powered electronically, the

junction to which is on the outside of the valve where the motor rotates the valve. One valuable piece of information about the device is that it has a 10 second action time. The listed price for a 80mm unit was $19.84 at the time of purchasing. Thankfully stainless steel is capable of reaching roughly 1400°F degrees before degradation, so using this stainless steel for the valve and pipes of the exhaust heat at 1000°F is not an issue. The only issue we saw with the device was that the lowest operable power rating was 12VDC, while our PCB design had only a 3.3V Vcc layer and a 5V layer, so powering the device would require an extra AC/DC converter to supply the 12V, increasing the total budget of the design.

The next part in consideration was the PVC pipe air volume control valve/damper. This device is offered in the following sizes: 75mm, 110mm, 160mm, and 200mm, fitting standard PVC air pipe sizes. It was offered at the time of part purchase at $5.28. Similar to the stainless steel damper listed previously, the PVC device has a value internally that will slowly swivel to close or open the pipe so that the exhaust air can flow freely. The device can be powered with 12VDC, 24VDC, or 24VAC. Again this product does not match either of Vcc options supplied by our PCB. On this model the action time is roughly 8 seconds, meaning that transitioning from fully closed to fully open takes the device 8 seconds. The device has a three pin connection on the outside of the pipe where the valve connection is. Sending a signal to the left pin closes the valve, the right pin opens the valve, and the center pin is sent the zero line (GND) The dimensions of the valve are 145mm x 75mm x 140mm for the 75mm width option and the other sizes can be found in the link to the device. Upon researching PVC piping, we discovered that PVC would reach its boiling point close to 500°F, while rigid PVC's melting point is 185°F. Neither of these temperatures would support our exhaust temperature needs of 1000°F. Although this device would be cost effective to perform the task, it is not a viable component for our design as the exhaust air temperature reaches far higher than compatible with the PVC valve.

Another device we researched was an exhaust fan valve/damper meant for air backdraft. This component was not an electrically controlled device. This valve is normally open, and its intended purpose is to be an air outlet or an air damper. This device is made from galvanized steel with a rubber gasket round the valve which prohibits outside air from entering into the pipe. The device is designed to only allow air to flow out from the pipe when there is a larger air pressure applied to the inside of the valve. This would eliminate the need to run cable to control the valve such as the other devices, however it would require a fan/air propulsion device to create air pressure on the inside of the device in order for the exhaust air to be released. The galvanized backdraft damper was priced at $16.58 which we found to be a competitive price compared to similar components. The dimensions are 3.9" x 3.9" by 3.1" and weighs approximately 1/2 lb. Our design led to the need for a small exhaust pipe due to the temperature hazard, so we did not rule out this component as it met many of our requirements. The only

drawback to this component is that galvanized steel has a maximum recommended temperature of 392°F long term and 660°F for a maximum instantaneous temperature, neither of which does meet the 1000°F, however we considered this component for testing as we were experimenting with output temperatures.

The last component we were considering for the valve was the Smart Water Valve SS304 Full Bore. This device stands out as its diameter is 1 inch. The design team considered using 2 or more of these smart water valves for controlling exhaust air from the device. Having duplicates in the design "back to back" would allow for more airflow out of the exhaust piping. It works by electronically rotating the ball valve 90 degrees inside the 1 inch brass pipe to cut off water/air flow. This device's action time to rotate the valve is between 3-5 seconds, which is the fastest response of the electrical valves we researched for the design. This valve was offered in 1⁄4 inch, 3/8 inch, ½ inch, ¾ inch, and 1 inch options, with the power options of 3.6-6VDC, 12VDC, 24VAC, and 220VAC. The max current draw of the device is 100mA, and its dimensions are 52mm x 66mm x 55mm. The electronics add to the device's size substantially, however it is well within reason for implementing within our project. The device is able to handle close to 500°F, however brass's melting point is 1700°F. The 500°F ceiling was discouraging for our design although as the design was being constructed the team was still experimenting with max exhaust temperature, which kept this device in the running to control the airflow out of our design. This component allowed for smaller design options, so the team kept considering its use. This part was also considered for air intake for the device as a method to close the intake when the device is not in use or turned off.

| Type | HVAC Air Damper | Air Volume Control Valve | Exhaust Fan Damper | Motorized Water Valve |
|---|---|---|---|---|
| Material | Steel | PVC | PVC | Stainless Steel |
| Price | $19.84 | $5.28 | $16.58 | $19.50 |
| Diameter | 3-8 inch | 75mm / 110mm / 160mm / 200mm | 3.9 inch x 3.9 inch by 3.1 inch | 1/4 inch, 3/8 inch, 1/2 inch, 3/4 inch, and 1 inch |
| Temperature Range | Up to 1400ºF | Up to 185°F | 392°F long term, 660°F instantaneous temperature | Up to 500°F |

| | | | | |
|---|---|---|---|---|
| Controls | Manual / Automatic | Manual / Automatic | Manual | Automatic |
| Power Supply Voltage | 220V, 24V and 12V | 12VDC, 24VDC, or 24VAC | Air Pressure Powered | 3.6 to 6V |
| Advantages | Precise Airflow Control | Lightweight | Discourages / Limits Backdraft | Precise Water flow control |
| Intended Function(s) | Commercial and Industrial HVAC | Residential and Commercial HVAC | Exhaust Systems / Ventilation | Water Supply Control |

*Table 3.2.7: Exhaust/Intake Valve Comparison*

### 3.2.8 - Airflow Fan Comparison

Within our design we planned to implement fans to push air through the sandbox which would become the space-heating exhaust air. We looked into the F1238S05B-FSR DC Brushless Fan, the Evercool Medium Speed Fan (EC3007M05CA), the GDSTIME Big Airflow USB Fan, and the UDQFB3E61 Double Ball Aluminum High-Temperature Resistant Cooling Fan. The fans would need intake from outside the unit to cool the fan and create airflow out from the sand battery. Researching this component lead to a multitude of good results/respective component options. While conducting our research we were not worried about noise compared to functionality and cost of the product(s). Internal fans would give our design increased heat distribution by pushing it out from/through the sandbox making the design more efficient internally as well as in the intended space.

First on our list of prospective components was the F1238S05B-FSR DC Brushless Fan. This part was compatible with our 5V PCB Vcc which made design/installation easier. This fan, priced at $18.72, had the dimensions: 120mm x 120mm, roughly 4.7 inches by 4.7 inches. This allowed for ample airflow within the device. The fan has a square frame for easy mount ability/installation, and it is connected electrically with two wires, a hot and a neutral. The Sound Noise was rated at 34dBA, 36dBA below the 70dBA sound safety standard, which we deemed well within reason for our product. The fan weighed in at 8.6 ounces, which added a sizable amount of weight to the sand battery, but negligible given its importance in the design. The fan is capable of 1800 RPM, with an air volume cubic feet per minute (CFM) of 83. This product was very competitive with other 5VDC fans on the market.

The next fan on our research list was the Evercool Medium Speed Fan (EC3007M05CA). This device made our list of fans because it is 5VDC compatible with a three wire connection, and its dimensions are 30mm x 30mm x 7mm, which is roughly 1.2 inch x 1.2 inch. This fan is obviously smaller than the Brushless Fan, however we considered implementing 4 fans in a square formation which would essentially create one larger fan. The small size allowed us to play around with the design and size of the ventilation piping. At $7.95 this product costs more per square inch than the brushless motor, however the device supports a much higher 8000 RPM. The noise level contribution would be <21dBA per fan, which is quiet when isolated to one fan, however the more fans added into the system, the louder their combine sound is, so multiple ~20dBA fans can easily increase the sound to unsafe levels. The drawbacks of this component are that they only produce 3 cubic feet per minute which would produce significantly less airflow than the brushless fan, as well as they are made with (UL 94V-0) plastic which is rated for under 500°F. If placed far enough from the exhaust air at 1000°F the fan would suffice, but they do not produce enough airflow to account for the distance from the sand. Although the combinations/configurations are endless for this component, the lack of airflow through the fan outweighed the advantage of its size.

We researched fans that could connect via USB which would make it easier to connect to certain ESP32 boards quickly. At the time we conducted the research, we were still deciding on which model of the ESP32 we would choose, many of which offer USB ports. The GDSTIME 80mm Router Cooler Fan in the USB controlled fan that we considered. With its operating voltage range from 4.5V to 5.5V, it is compatible with the designed PCB. The dimensions are 82mm x 82mm x 57mm / 3.23 inch x 3.23 inch x 2.24 inch. This is a perfect size for our residential design as it is compatible with common air duct components. Its noise rating at 27dBA would be relatively quiet, especially given its airflow rating just shy of 40 cubic feet per minute. The ratio of airflow to noise is the highest of the components we researched. The life expectancy of the fan is 40,000 hours at 25°C or 77°F. Its maximum RPM is 2500, and the device is made from plastic. This again is a less-desirable material for a fan handling 1000° temperatures, but the amount of airflow produced by the product enabled us to consider placing the fan at a distance where ambient temperature around the fan would be much lower. The cost of the device was $11.99 at the time of research.

Lastly we researched a Fan with an Aluminum frame to potentially reduce the risk of the fan melting/wearing quickly due to high temperatures. This was the UDQFB3E61 Double Ball Aluminum High-Temperature Resistant Cooling Fan. The dimensions of this fan are 1.2 inch x 1.2 inch x 0.4 inch, with a 2 pin power connection, making it easily adaptable/powered. The fan's aluminum frame allows for a higher ambient temperature as aluminum melts far past 1000°F, however the fan blade material was still plastic which has a much lower melting point than aluminum. The square frame with screw holes allows the fan to be mounted easily and quickly. The fan is powered by 5V, perfect for our PCB inside

the sand battery. The aluminum frame fan was priced at $26.38, a few dollars more than its competition but not a deal breaker for the design team given its temperature resistance and easy installation.

| Model | F1238S05B-FSB Fan Brushless | Evercool Medium Speed Fan | GDSTIME Big Airflow USB Fan | UDQFB3E61 Aluminum Double Ball High Temp Resistant Fan |
|---|---|---|---|---|
| Size/Dimensions | 120mm x 120mm x 38mm | 30mm x 30mm x 7mm | 120mm x 120mm x 25mm | 1.2 inch x 1.2 inch x 0.4 inch |
| Price | $18.72 | $7.95 | $11.99 | $26.38 |
| Speed in RPM | 1800 | 8000 | 2500 | Variable with Current |
| Noise Level in dBA | 34 | ~20 | 27 | Not Specified |
| Airflow (CFM) | 83 | <3 | 40 | Variable with Current |
| Power Supply Voltage | 5V | 5V | 4.5 to 5.5V | 5V |
| Intended Function(s) | General Cooling | Small Electronics Cooling | Laptop Cooling | High Temperature Environment Cooling |

*Table 3.2.8: Airflow Fan Comparison*

## 3.2.9 - Heat Transfer Conduit Comparison

To design the sandbox with airflow through it where the sand would not be released. In the design process, we had the idea to have metal tubes running through the sand. We researched metals that would absorb heat from the sand it was running through and would heat the air flowing from the fans through the heated pipes and out through the exhaust piping. We researched a 1 inch Metallic Emt Conduit, 1-2 inch Copper Tubing, Stainless Steel 1 inch Tubing (thick wall), and ¾ inch Aluminum Tubing.

First on our list was the Metal Emt Conduit (1 inch). This product was sold at $17.47 for a 10 foot pipe, $1.75 a foot. It is advertised as a conduit which protects electrical wire from magnetic fields as well as damage/impact. Its interior is coated to help wire slide through the length of the pipe. The pipe is not threaded at the ends, which is one reason why we considered the product as we

believed it would be easier to install through the sand box. It is made from galvanized steel which would be rated for 392°F under long-term conditions and up to 660°F for short-term conditions. This is not conducive for heating the sand up to the 1000°F goal, however if the design only needed a temperature of 400°F we would possibly be able to use this galvanized steel piping throughout the sandbox. We did find that steel is a poor heat conductor, which removed the galvanized steel pipe from our list of solutions.

The second material we considered was copper tubing. We found 2 inch copper pipe for $20.37 per foot and 1 inch pipe for $6.37 per foot. Because copper is a good conductor of heat, our thought behind implementing copper pipes such as HVAC copper condenser piping for the design which would run through the sand. Copper is known for being very durable and resistant to corrosion, so the team decided it would be the ideal candidate as the metal for venting the sand's heat. One of our team members had experience working with HVAC copper tubing with knowledge of how to bevel out copper tubing. The goal with the tube ends was to essentially bend the tips of the copper on either side which would mechanically seal the sand into the sandbox, and create a wider entrance/exit from the pipe which would improve the airflow. Copper has a melting point of 1984°F, just shy of double our goal temperature. This left plenty of room for experimenting with the max temperature of the sand inside the box.

We researched another stainless steel pipe with a 1 inch diameter. The main reason we considered this option after discovering that galvanized steel is a poorer heat conductor is because of the thickness of the pipe wall itself. We later discovered in our research that 304 stainless steel has a much higher temperature rating (800°F – 1580°F short term) than that of galvanized steel. This steel is actually used closer to 1700°F typically when used for high temperature applications due to the fact that it corrodes less at 1700°F than in the 800°F – 1580°F range. The 304 stainless steel thick pipe was priced at $7.85 per foot, and we could customize the specific length of pipe needed without being charged more unused materials. Each with an average lifetime of 50-100 years, the team believed either copper or stainless steel would suffice or even outlive the lifetime of the electronics in the sand battery, however extended exposure to heat can in some cases corrode stainless steel over time which would shorten the lifetime of the sand battery.

The last metal we considered for heat ventilation in the sand box was ¾ inch aluminum tubing. At $4.75 per foot with full order customization, this tubing would have been the most cost effective. Aluminum does have a melting point of 1260°F and becomes soft around 400°F. This is a lower temperature than copper or stainless steel can maintain, however copper is a very good conductor of heat, so while aluminum cannot maintain as high of a heat, we chose to consider using the aluminum as it potentially would allow us to reduce the temperature needed inside the sandbox to produce the same output temperature as the copper or stainless steel pipes. We considered following the same process as we would

have with copper by beveling the tube ends to bend the tips of the aluminum on either side which would mechanically seal the sand into the sandbox, creating a wider opening for air to more easily flow in and out.

Selecting tubing to pipe through the sand was not an easy task as we had three differently beneficial choices of parts. When we did the down-select we chose to use the 1 inch copper tubing because of its heat conductivity, its high temperature compatibility, and its cost-effectiveness (at $6.34 per foot). This metal simplified our design by allowing us to plan to heat the sand to the originally decided 1000°F.

| Type of Tubing | 1 inch Metallic EMT Conduit | 1 inch Copper Tubing | Stainless Steel 1 inch ThickWall Tubing | 3/4 inch Aluminum Tubing |
|---|---|---|---|---|
| Material | Steel | Copper | Stainless Steel | Aluminum |
| Price | $17.47 per 10 feet | $6.37 per foot | $14.80 per foot | $4.75 per foot |
| Size (Diameter) | 1 inch | 1 inch | 1 inch | 3/4 inch |
| Corrosion Resistance | 70% | 60% | 90% | 90% |
| Flexibility | 10% | 40% | 5% | 80% |
| Advantages | Durability | High thermal Conductivity, High electrical conductivity | Corrosion Resistant, Strength | Lightweight, Corrosion Resistance |
| Strength | 50% | 70% | 90% | 20% |
| Intended Function(s) | Electrical Wiring protection | HVAC, Water Supply | Industrial Piping, Structural Component | Structural Component, Electrical Wiring |

*Table 3.2.9: Heat Transfer Conduit Comparison*

## 3.2.10 - Smoke Detector/Alarm Comparison

With the thermal battery reaching temperatures over 700°F, the likelihood of a fire occurring in the design is something worth considering and minimizing. We have mitigated some of these potential issues by choosing components very carefully (examining their combustion temperature and exposure to the high temperature

materials), but with some outcome uncertainty, we decided to implement a fire alarm to alert the user when fire occurs.

First on our list of smoke detection devices researched is the Winsen ZP13 Smoke Detection Module. This option offers gas detection of propane and smoke, with a 5V working voltage. Its warmup time is rated at less than 180 seconds, and response time is less than 20s. This device does require a temperature under 122°F for operation, however it can be in storage at 140°F. It has a lifespan above 5 years, which is adequate but not phenomenal when compared to the 10 year lifespan of a typical home fire alarm. This device would allow for less maintenance over its lifespan as most home fire alarms require new batteries every 6 months, and the Winsen ZP13 runs until it no longer functions. Its connection is through a XH2.54-4P terminal connection and is priced as a 10 count for $22.00. Per unit this device is competitively priced, however the increment of 10 would cause financial waste given our limited budget.

The following device we looked into for smoke detection was the RE46C180S16F. This is an IC Ion Smoke detector sold by Digikey with an operating voltage of 6-12V, above our designed PCB Vcc layers of 3.3V and 5V, which would require more components to operate such as an additional AC-DC converter. This circuit is designed to require very low power, including a procedure where its oscillator turns on the device for 5ms every 10 seconds to check for smoke. This allows the device to remain powered off for a large portion of the device's running time. The device's recommended operational temperature is between 14°F and 140°F which is a larger range than the Winsen ZP13 model. The IC Ion Smoke Detector is a 16 pin CMOS and it includes a timer mode as well as an alarm memory as well, which we may chose to implement in further design iterations. At the time of our research, this component was priced at a very reasonable $1.62 per unit.

Another module we considered was the L-com SRAQ-G004 Gas Sensor Module. It is powered by 5Vdc, perfect for the PCB design we implemented inside the thermal battery. It uses a Mq2 sensor, a component that works well at extreme humidities. A downside to this component is the operating temperature. It is limited to 14°F to 120°F. Its outputs are analog and TTL, and it is connected to via 4 pins: GND, Dout, Aout, and Vcc. Another drawback to this device is its unspecified preheat time. The best statistic provided is that it warms up in under 30 minutes. It is designed to detect flammable gasses as well as smoke, which make it a good candidate for our implementation. It was priced at $6.29 per unit.

Many of these components do not come with built-in audible alarms, however this component's analog and TTL outputs provide communication methods between it and our ESP32 board.

Next on our research list was the Reland Sun NAP-07 Ionization Chamber Smoke Sensor. This device, sold for a competitive $1.98 (plus $4.74 for shipping), has a diameter 0f 37mm. Its small size made it a very appealing component for our design. The component would come with challenges, such as a 9V supply voltage needed to power it, as well as a lack of temperature rating documentation. We did not think we had enough information about this relatively new component to rely on it for such a crucial part of our project. We did occasionally check to see if the manufacturer released any new/updated information about the component throughout our senior design 1 course as it looked like it would yield very promising results. This device also meets the safety standards GB4075, IS02919, and C64444.

The last device in this category which we researched was the First Alert SA303CN3 Smoke Alarm (battery operated). We chose to look into this component because of the manufacturer's reputability in the realm of smoke alarms. As we did not want to haphazardly monitor our project for flaming fire(s), we chose to research a unit with an onboard audible alarm. The alarm we chose to research has a noise output of 85dB where it alarms ("chirps") three times, cycling on for .5s and off for .5s. It requires a 9V battery for power, weighs close to .5 lbs, and it is 4 inch x 4 inch x 2 inch. This means the First Alert Alarm would be the largest of this category of components. It is easily attached to the mounting bracket by pushing the alarm in and twisting, which makes for easy access to the battery or for replacing the device if necessary. There is an easy access door for the battery, meaning we do not need to disconnect the alarm from the bracket to replace the battery, which can be useful in our project with the limited accessibility of the alarm. The alarm utilizes an Ionization Sensor like others we researched, and the alarm has one button which sounds/silences the test alarm. The SA303CN3 adheres to the UL 217 standard and we priced the component at $12.95 (per unit). The First Alert Alarm would not need to be wired to the ESP32 board, which makes adding this part very brief and simple.

| Model | Winsen ZP13 Smoke | RE46C180S16F | L-com SRAQ-G004 Gas Sensor | Reland Sun NAP-07 | First Alert SA303CN3 Smoke Alarm |
|---|---|---|---|---|---|

| | Detection Module | | | Ionization Sensor | |
|---|---|---|---|---|---|
| Price | $22.00 per 10 units | $1.62 per unit | $6.29 per unit | $1.98 (+ S&H) | $12.95 per unit |
| Type of Detection | Propane & Smoke | Smoke | Flammable gases, smoke | Smoke Ionization Sensor | Ionization Sensor |
| Power Source | 5Vcc | 6 -12 Vcc | 5Vdc | 9Vcc | 9V Battery |
| Type of Alarm | Circuitry Component | Circuitry Component | Circuitry Component | Circuitry Component | Wall/Surface Mount |
| Communication with ESP32 | Yes | Yes | Yes | Yes | No |
| Battery Operated | No | No | No | No | Yes |
| Lifespan | 5 Years | 10 Years | --------- | --------- | 10 Years with recommended 6mo battery change |
| Warmup Time | 180 seconds | --------- | 30 Seconds | --------- | 0 seconds |

*Table 3.2.10 Smoke Detector/Alarm*

## 3.2.11 - Sand Composition Comparison

With very little foreknowledge of sand composition and the effects it can have on the ability to heat sand to the temperature specified in our design, we researched a variety of prospective sand types to use in our sand enclosure. The thermal conductivity needed to be very high as thermal conductivity is how heat is transferred from the sand to the venting coils/tubes going through the enclosure.

Heat/temperature capacity was another factor in trying to select the ideal sand composition, because it must be capable of degrees close to 1000°F to meet our design goal and with higher heat capacity, the larger amount of energy our device can store. Our goal for energy storage is 5kWh which composition could affect greatly.

Similar to thermal conductivity, we tried to find sand which would have uniform heating. Chemical stability was a factor we included as well as the sand needed to cause as minimal corrosion as possible with the heating elements/frame of the enclosure (including the pipes running through the enclosure), as well as uniform heating. Uniform heating would allow for consistent thermocouple readings as well as induce less wear on the heating elements. Each sand has a melting point which we also paid attention to when making the selection as we wanted to maintain the integrity of the sand. Luckily sand cannot evaporate, however if the sand in the enclosure did melt it would inevitably change the volume of the sand, which could either expand or diminish too low to problematic levels. As our design would not expand/contract with the changing sand volume, we made an effort to ensure the sand chosen would not be able to vary in volume by choosing the maximum temperature carefully. The types of sand which we researched were: Silica (Quartz) Sand, Calcium Carbonate Sand, Olivine Sand, Garnet Sand, and Polymeric Sand.

Silica Sand is the first sand composition we looked into to use as the sand in the enclosure. The exact type was AquaQuartz-50 20-Grade Silica Sand. The source we researched was selling white silica sand as 50 pound bags, which occupies a volume of 13 inch x 10 inch x 10 inch. At $29.99 per 50 lb bag, this AquaQuartz sand bag is advertised as a sand used for pool filtration. We were able to find Silica's (Quartz) melting point at 3110°F, a temperature more than triple our design's max temperature, leaving plenty of room for fluctuation in temperature in the sand enclosure. It has a higher thermal conductivity, allowing it to transfer heat well. This lends to more flexibility in the design in regard to heating element placement(s). It provides a moderate amount of heat capacity of 0.9 J/g*K and a great amount of thermal conductivity (1-1.3 W/m*K). These qualities lend to the use of silica sand in glassmaking and construction applications.

Calcium Carbonate Sand, sometimes known as Calcite Sand or limestone, was the next type of sand on our list to research. We found a 50 lb bag, the same as the silica sand, of fine-soft ground white limestone. The manufacturer is Ernst Grain and they advertise this product as having multiple uses. True to its marketing, calcium carbonate sand is often used for cement production,

agriculture, and even aquamarine environments. This calcium carbonate sand is rated for lower heat capacity and lower thermal conductivity than silica sand, which means it cannot transfer heat as well as the silica, and with a melting point at only 1510°F, it has a lower heat capacity. This temperature of 1510°F would not cause an issue with the temperature reached for our design, however there is less room for error with the calcium carbonate. The 50 lb bag we found was priced at $44.99, an extra 50% to the cost of the sand compared to silica, which is a large jump up in price given our overall budget of $800.

The next sand researched was Olivine Sand, also known as magnesium iron silicate. This type of sand is sold as 10 lb and 24 lb bags, and the full name of the product is Teton-Black Olivine Fine-Mesh Water-Bonded Foundry Casting Sand by Teton Supply Co. The 10 lb bag is $33.49 and the 24 lb bag is $59.99, with the 24 lb bag being four times as much as the silica per pound. This type of sand has numerous benefits such as its high melting point of 3200°F (which is about 100°F higher than silica sand), which therefore means it has a higher heat capacity. Another positive aspect to the olivine sand is the thermal conductivity higher than that of silica. Common uses for this type of sand are foundry related areas. Ultimately this type of sand is slightly better in most aspects that were useful for our senior design project, however the high cost of the product was a strong factor in the design team wanting another type of sand to enclose for the thermal sand battery.

Garnet sand is another distinct option for our sand enclosure. We found a 55 lb bag of Abrasive Garnet sand. This type of sand, as advertised, is often employed for cleaning, profiling, and for the removal of paint and rust. In our research we discovered that garnet sand has roughly double the thermal conductivity as silica, ringing in at 2.4-2.7 W/m*K. The garnet sand is offered at multiple grit sizes (the number of particles that can fit through a square inch at once), 30/60, 36, 80, and 120. The grit sizes can vary the cost from the RickRidge supplier. We were interested in implementing the 120 grit size sand which was offered at $58.75. This would be just shy of double the price of silica sand, however with almost twice the thermal conductivity, the comparison leads to a hard decision. The heat capacity of garnet sand is roughly the same as silica around 0.9 J/g*K. In regard to size/dimensions of the thermal sand battery, garnet is capable of conducting more heat in a given space than silica, however this comes with the increase in price for the material. This balance gave our team great options while deciding the dimensions.

Next on our research list of sand compositions to consider was Polymeric sand. We sourced a 40 lb bag of Midnight Black DOMINATOR Polymeric Sand with Ceramic Flex Technology. This bag sold for $91.00, with the volume equaling just under 0.6 cubic feet. Our initial project design allowed for slightly less than 1 cubic foot. This product is typically designed for stabilizing pavers and walkways similar to grout. it also makes for a useful binding agent when implemented properly. Its thermal conductivity is lower than the rest of the sand compositions we have researched, clocking in at less than 1 W/m*K. It has a lower heat capacity as well as a significantly lower melting point (230°F – 392°F depending on the mixture). When we compared this type of sand to our other examined options, we decided to not move forward with implementing this type of sand for the thermal sand battery.

The type of sand which we chose to use was the Ernst Grain calcium Carbonate (limestone) sand. This option was appealing to us for our design as the low cost of $44.99 worked better for our budget than any of the other options we researched of pure-composition sands. From what we found in our research, this type of sand is made from a combination of quartz and limestone with the majority being limestone. With the limestone maximum temperature of 1510ºF, we feel confident the sand will meet the requirements in our design. The only draw back to this sand is the weight of 50lbs per 0.7 cubic feet

| Composition | Silica Sand | Calcium Carbonate (limestone) | Olivine Sand | Garnet Sand | Polymeric Sand |
|---|---|---|---|---|---|
| Price | $29.99 | $44.99 | $59.99 | $58.75 | $91.00 |
| Brand/manufacturer | AquaGuartz | Ernst Grain | Teton Supply Co. | RockRidge | DOMINATOR |
| Thermal Conductivity | 1 - 1.3 W/m*K | 2 - 3 W/m*K | 4.5 W/m*K | 2.4 – 2.7 W/m*K | 0.95 W/m*K |
| Heat Capacity | 0.9 J/g*K | 0.82 J/g*K | 0.8-1 J/g*K | 0.8-1 J/g*K | 1.1 – 1.9 J/g*K |
| Volume | .7523 ft^3 | 0.7 ft^3 | 0.296 ft^3 | .31 ft^3 | 0.6 ft^3 |

| | | | | | |
|---|---|---|---|---|---|
| Weight | 50 lbs | 50 lbs | 24 lbs | 55 lbs | 40 lbs |
| Maximum Temperature | 3110°F | 1510°F | 3200°F | 2200°F | 230°F – 392°F |

*Table 3.2.11 Sand Composition*

## 3.2.12 - Heating Element Comparison

One critical component in our design was the heating element(s). It was critical for our sand enclosure to heat the sand to our goal temperature of 1000ºF, and in order to do that we needed to find heating elements which could handle 1000W. These heating elements could be wired in parallel if needed, our goal only focused on the amount of wattage drawn by the elements.

There are multiple reasons why choosing an ideal heating element was an important step in the design. The first important factor which heating elements would affect was the efficiency of our thermal sand battery. Choosing one with a low resistance would raise the efficiency of the battery as more watts would be transferred to the sand. The amount of possible stored energy would be raised in the sand if our elements were to distribute the heat evenly and to a higher temperature. Lastly, with a limited amount of current draw from the AC circuit our device plugs into, it was crucial for our device to maximize its output wattage without drawing 15A to break the circuit. Our self-applied current limit for the device was 12A which includes the heating elements, PCB, and any other devices onboard the unit.

The first potential heating element we researched was the Rheem Protect Copper Heating Element. This component is designed for 120V circuits at a length of 7.65 inches, which was very compatible with our cubic foot sand enclosure. The component was priced at $11.94, which is relatively inexpensive, however we were still researching how many units we would need to heat the sand and $11.94 per unit would have added up quickly in our budget of $800. The device's shape resembled that of a hairpin with the rounded side farthest from the flange. With the heating element made of metal, we knew that the element could handle extremely high temperatures, however we had hesitation with using this copper element as it was designed to be a water heater's element which typically are cooled by the water, meaning in its normal use the exterior of the copper would only reach 212ºF, and most units do not go past 160ºF. The heating element is rated for 2000W, which at 120V would require 16.67A, higher

than our allotted amount of current, so we were going to need to use multiple in series to limit the current through the circuit. The only remaining issue we saw in this component's design was the threads with which the heater is screwed into place. It was beyond our manufacturing scope/tools to retrofit holes in our enclosure to support this type of mount for the heaters.

Our research also included the DERNORD Tri-Clamp Foldback Heating Element. This one is similar to the Rheem Protect copper element, however the DERNORD is made with stainless steel, as well as a locking plug. The shape of the element is similar however this one is built with the round end bent back toward the mount, so effectively it has twice the coils built into the one unit. The component is rated for 120V and 1650W. The price of this unit is $34.99, more than triple that of the copper heating element. With a current draw of 13.75A, again we would need to put at least two of these units in series or configure the load of the circuit differently in order to drop the current pull below the 12A. There were different voltage and wattage rating options for this component, and with its 3 wire plug we believed the installation would be relatively simple, however the much higher cost did steer us toward another heating element for the sand enclosure.

Another potential heating element for the sand enclosure was the Char-Broil Universal electric Heating Element. This component is designed for heating a smoker/grill and it included a dial which would adjust the temperature. This heating element is 10.2 inch by 10.2 inch and the two leads would stick out 3-4 more inches. This was not a quality we were worried about as we could attach the leads outside of our enclosure, though it would not be ideal. It is meant for 110V and could provide 1500W. We would have removed the dial/thermometer from the device and simply used its actual heating element if we were to implement the part. Our thought behind including this into our research was that we could have a heating element in the middle of the enclosure with this element heating sand around the perimeter of the enclosure. Priced at $29.49, we were optimistic about the use of this component. The main drawback for the element was the temperature rating from 50ºF to 500ºF, however we believed those limits to be due more to the plastic dial for controlling the temperature via its intended method. We ultimately chose not to use this component as we believed we found a better configuration for the heating elements at a lower cost.

One unique style of heating element we researched to potentially use was the Camplux Cartridge Heater pipe fittings. This item sold as a set of three 110V heaters. With each fitting capable of 100W output, we would be able to

customize the amount of power drawn by the heaters very easily. These fittings were cylindrical with a length of 1.6 inches and 8mm diameter. To utilize the heaters, our best design idea involved inserting each one just inside the perimeter of the sand enclosure. We were hesitant to use this idea/design because of the risk of not properly sealing off each hole sufficiently in the sand enclosure which would lead to heat leakage and a less efficient design, not to mention the added strain to the components outside the enclosure such as the PCB. At $11.99 for three of these cartridge heaters, we left this component as a backup plan to our leading idea.

The last component we found for our research was the Whirlpool Electric Range Stove Set. This set of elements came as a four pack of stovetop burners in the typical circular coil shape. The set contained two coils with a 4 inch diameter, and two with 6 inch diameters. Whirlpool's advertising said these coils can reach a temperature of 250ºC in thirty seconds, which is almost half of our temperature goal for the sand. For the pack of four, we priced the set at $29.56, equating to $7.39 per burner. Each coil came with its own bracket underneath which we were able to remove. Doing so allowed us to feed our copper ventilation tubes through the center of the elements for maximum heat exposure. This would allow the copper pipes to heat up quickly and allowed the thermal sand battery to release warm air as quickly as possible while still heating up the sand with good heat distribution. These heating elements were simple enough to provide power to as they had two leads extending straight out from the center. Our plan included piping these two circular coil leads through the sand enclosure box so that our wiring would be attached from the outside, limiting the wires' exposure to the heated sand. These coils were rated for 230V at 1500W. We found the resistance of the 4 inch element to be 43ohms, and the resistance of the 6 inch element to be 32ohms. We were able to wire two of the 8 inch coils in parallel to disperse roughly 950W from a 120V outlet in our senior design laboratory. We decided to commit to these heating elements as they provided good heat disbursement, high efficiency, quick heating capability, and cost effectiveness.

| Model | Rheem Protect Copper | DERNORD Tri-Clamp Foldback | Char-Broil Universal electric | Camplux Cartridge Heater pipe fittings | Whirlpool Electric Range Stove Set |
|---|---|---|---|---|---|
| Price | $11.94 | $34.99 | $29.49 | $11.99/ 3 elements | $29.56 / 4 elements |

| Metal Type | Copper | Stainless Steel | Stainless Steel | Stainless Steel | Nichrome inside Steel |
|---|---|---|---|---|---|
| Input Voltage | 120V | 120V | 110V | 110V | 230V |
| Power Rating (W) | 2000 | 1650 | 1500 | 100 | 1500 |
| Resistance (ohms) | 7.2 | 8.73 | 8 | 121 | 32 and 43 |
| Dimensions | 12 inches x 3 inches x 5 inches | 15.1 inches x 2.4 inches x 2.6 inches | 15.6 inches x 10.2 inches x 2 inches | 1.6 inches x 0.32 inches | 4 inches x 4.33 (or 6.33) inches x 0.5 inches |

*Table 3.2.12 Heating Element Comparison*

## 3.2.13 - Ventilation Comparison

We chose to compare different HVAC tubing in our research for the airflow through the device. The tubing would need to hand high temperatures and require little to no maintenance once installed in the design. We also required a tubing type that would be easy to install given our limited manufacturing abilities limited to the UCF senior design laboratory. We looked into each of the following: flexible ductwork, spiral ductwork, and rigid metal pipe ductwork. Each provided different advantages for our design.

First of the researched duct work was the Semi-Rigid Aluminum duct. This option would allow for quick mobility/implementation into our design. This product is designed for bathroom fan venting and is rated up to 435ºF temperatures. This is a number lower than the max heat of the sand enclosure, however when we conducted our research we were unable to calculate/measure the temperature of the air that would be vented out from the copper tubing in the sand enclosure until we had manufactured a bulk of the project. We considered this tubing for its flexibility when we first started researching for the project as the material was highly flexible and simple to install in tight spaces around obstacles.

This type of tubing is also advantageous in regard to weight and cost. The price of a 3 inch by 8 feet length of tubing was $14.82 when we conducted our research. The product did have some drawbacks such as: little durability as the aluminum is prone to tearing or being punctured, it can collapse if not supported properly which would reduce the airflow and cause mechanical/ventilation issues which require maintenance, and it would be difficult to clean should any dust/particles enter the device in its lifetime. One flaw in the design of this piping was its lack of compatibility with the 3 inch valve air dampers we chose to

implement in the design. We needed a solid connecting for the air damper so that it could effectively cut off the airflow in the design, and we were unable to manufacture a joint for the flexible air duct and the air valve.

Second on our research list of ductwork components was the 4 Spiral Pipe duct work. This option is typically used in commercial HVAC systems used to distribute air and ventilate large office spaces as the design has a more aesthetically appealing design than rigid square duct work. The spiral type of duct work also allows for easy maintenance/cleaning as it provides less surface area than square duct work where dust can collect on top of it. Spiral ductwork is much more durable than the semi-rigid aluminum ducts as it is made from 24 gauge steel.

This specific piping for a 5 foot piece of spiral pipe was priced at $39.95 and it is typically implemented in designs with 5 horse power or more, so we were confident it could handle our ventilation needs. Its design allows for leakage resistance while using the spiral pipe as the design requires fewer joints and seems compared to traditional rectangular duct work. The sections are often prefabricated and modular, which typically allows for a faster installation time and less labor cost, however for our design this duct work would actually prove to be more work at each seem due to the HVAC air damper valves we needed to include in the design as they were to be in the middle of the piping we were to install.

Another issue we ran into was in the realm of manufacturing as this spiral pipe has a diameter of 4 inches where our air damper needed 3 inches. We considered trying to fabricate a HVAC joint which would reduce the 4 inch diameter to 3 inch. Also with limited access to power tools, the design team did not feel comfortable with cutting the spiral pipe as we would likely not be able to fabricate a flat cut on the end of the pipe. Even though the spiral HVAC pipe is versatile and very efficient among commercial HVAC piping, instead of trying to proceed with the spiral pipe duct we decided to move forward with another piece of duct work hardware.

The last form of ductwork we looked into during our research was the Master Flow Round Metal Duct Pipe. This product was offered with a 3 inch diameter which is the ideal size for our implementing the air valves seen above which allow us to allow airflow through the device on command. In total there will be 4 physical connections to the air valves, so from a manufacturing standpoint it is advantageous to use a product with the same diameter as the air dampers we installed in the project. This type of duct is similar to the spiral wound duct pipe, however it allows for simpler installation as the pipe can actually be unrolled so it would be easy to cut square which would allow for the joints to be easily formed. It is a more energy efficient design than the semi-rigid aluminum duct as it does not more any corners or pockets where air could be slowed down by, instead it would be a smooth surface on the interior causing less friction and allowing more

airflow. This is crucial as we were limited in horse power of our exhaust fans due to the low voltage available inside our design. We wanted to ensure the heating elements received adequate current to heat the sand, so our fans were limited in power. Made from galvanized steel with gauge options from 26 to 30, we had flexibility in the thickness of tubing we were able to select. We decided 30 gauge would fit our needs better than a lower gauge as the high temperatures from the exhaust air would be less likely to warp/effect the thicker gauged metal. The ends of this pipe are crimped, allowing the pipe to fit inside other 3 inch diameter fittings with easy, which made for easy fabrication in our design. At the price of $5.48 per 2 foot length 30 gauge pipe, this product would cost considerably less than its spiral pipe counter part, however the aluminum semi-rigid pipe cost the most per foot at $14.82 per 8 foot section.

| Model | Semi Rigid Aluminum Duct | Flexible Semi-Rigid Spiral Aluminum Duct | Round Metal Duct Pipe |
|---|---|---|---|
| Price | $14.82 / 8 feet | $39.95 / 5 feet | $5.28 / 2 feet |
| Diameter | 3 inch | 4 inch | 3 inch |
| Metal | Yes | No | No |
| shape | Semi Rigid Cylindrical Pipe with Corrugated Ribs | Cylindrical Pipe | Cylindrical Pipe |
| Flexible | Yes | No | No |
| Durable | no | Yes | Yes |

*Table 3.2.13 HVAC ventilation comparison*

# Chapter 4:Standards and Design Constraints

## 4.1 Industrial Standards

### 4.1.1 - PCB Standards

Manufacturers and fabricators follow specific industry standards for the PCB to be able to adhere to a specified level of functionality. These standards can affect the workmanship quality and performance of the board. Manufacturers may produce PCBs that follow different standards. We have chosen our manufacturer to be JLC PCB. Therefore, we want to thoroughly explore the standards associated with their PCB fabrication. By having an understanding of these industry standards, we are able to learn how to produce a quality PCB for this project.

- IPC-2221: Generic Standard on Printed Board Design
- IPC-6012: Qualification and Performance Requirements for Rigid PCBs
- IPC-A-600: Acceptability of Printed Boards
- IPC-A-610: Acceptability of Electronic Assemblies
- J-STD-001:Requirements for Soldered Electrical and Electronic Assemblies

#### *4.1.1.1 JLC PCB Fabrication Standards*

This manufacturer has certain limitations for fabricating PCBs. These limitations must be considered when designing. Failure to do so will cause any delays in the fabrication of our PCB and this project overall. These limitations include but are not limited to, layer count, max dimensions, board thickness, drill/ hole sizes, and tracing width. The layer count is essential in knowing how many layers we must limit our board to having. Fortunately, this manufacturer has a max board count of 20 layers. The max dimensions of the board is just as important to establish our design constraints. JLC allows a max dimension of 400x500mm. Understanding the drill hole size of the PCB allows us to properly plan how to mount the PCB onto our battery. JLC provides a capability hole size of 0.15-6.30mm. Vias for a single and double layer PCBs have a range of 0.3mm/0.5mm for hole size and diameter respectively. We also have a pad size of 1.0mm minimum. The two most important standards to consider are the minimum clearance and the trace width. Minimum clearance tells us how much distance is needed between traces, routes, and vias. For example, via to via must have a distance of 0.254mm apart. Our pads must have a distance of 0.127mm apart and pad to tracks must have 0.2 clearance between each other. Minimum trace width and spacing is essential for EMC considerations and current considerations. If a trace is too small, we might not be able to drive the current required for certain components. This could be detrimental to providing a working board. Larger traces that are too close to

other traces may cause noise interference. The minimum trace width is 0.127mm for a single and double layer board and 0.09mm for a 4-6 layer board.

| Specification | Capability |
|---|---|
| Layer Count | 1-20 Layers |
| Max Dimensions | 400 x 500 mm |
| Board Thickness | 0.4-2.5 mm |
| Drill Hole Size | 0.15-6.30 mm |
| Min. Via Hole/Diameter Size | 0.15/0.25 mm |
| Pad Size | Min of 1.0 mm |
| Hole to Hole Clearance | 0.5 mm |
| Via to Via Clearance | 0.254 mm |
| Pad to Pad Clearance | 0.5 mm |
| Min. Trace Width (1-2 Layers) | 0.127 mm |
| Min. Trace Width (4-6 Layers) | 0.09 mm |
| Trace Outline | 0.3 mm |

*Table 4.1.1: JLCPCB Fabrication Standards*

### *4.1.1.2 - IPC-2221 Standard*

This is a general standard that creates the overall requirements for the design of printed boards and other kinds of mounting techniques. This document is not meant to be a standard in terms of performance, just in terms of workmanship.This standard generally covers these areas:

1. **Board Size and Shape:** IPC-2221 makes guidelines for board size and shape. This includes the aspect ratio, edge design, size limitations, mounting holes, and thickness of the board. The aspect ratio is defined as the ratio of the board thickness compared to the smallest feature size. The guidelines on edge design are meant to minimize damage during assembly. The size limitation is meant to give manufacturers minimum and maximum dimensions and considerations for tolerances.

2. **Component Placement:** Guidelines are included for how to place SMT and through hole components onto the board. The orientation of components must be placed based on electrical requirements to minimize

EMC and trace path. There is also a major consideration on accessibility. This is so the components are easily replaceable and to avoid board edges.

3. **Trace Spacing and Routing:** There are instructions on routing traces between components to prevent EMC. The width and spacing of the traces is also important to take into account the current carrying capacity and to prevent electrical shorts.

4. **Via Design:** IPC-2221 provides guidance on vias for PCBs. This is meant to cover workmanship standards of how vias should be drilled. This is crucial as vias provide a way for traces to reach different layers of the board. Things to consider include the size, spacing, placement, type of via, and the plating of the via. These considerations are again meant to prevent noise interference and possible shorts.

5. **Thermal Considerations:** Heat can be detrimental to components on the PCB. Heat can seriously degrade the performance of the circuit. By following the guidelines, we can prevent thermal damage and learn how to manage heat dissipation.Factors to take into consideration include selecting the right materials with thermal conductivity, creating thermal relief patterns, selecting the right copper weight, having vias dedicated to heat dissipation, and including heat sinks and fans for more heat dissipation.

Ensuring we choose a manufacturer that follows these standards will increase our chances of having a board that has fewer errors and requires less rework. It will save us on costs and ensure we have a reliable board to work with.

#### *4.1.1.3 - IPC-A-6012 Standard*

IPC-6012 is a performance standard that helps classify the performance and reliability specification of rigid PCBs into three classes. Each class has a different performance and fabrication metric that it must meet. The 3 classes are:

1. **Class 1 - General Electronic Products:** These electronics have general functionality. They are not critical and mainly include low power using and low-signal transmission signals. Reliability and performance is not a hard requirement. These electronics consist of some consumer electronics. There is also a moderate amount of leniency in terms of the cosmetic workmanship of the assembly of the board.

2. **Class 2- Specialized service electronic products:** Class 2 boards must have higher reliability and extended life. They follow a stricter set of standards for their performance and workmanship expectations. However, some cosmetic imperfections are allowed. Uninterrupted service is

preferred but is not required. Class 2 products are not designed to be exposed to extreme environmental conditions. These products might have a moderate amount of power consumption and transmission signals. These boards are used in business machines and communication equipment.

3. **Class 3 - High-performance electronic devices:** Class 3 boards are meant for high performance applications. These boards must have proven high reliability and the highest performance compared to the other classes. They might have high power requirements or possess high speed communications. These boards must have continuous uninterrupted service. Product downtime can not be tolerated. The components and the board itself must be able to withstand harsh environmental conditions. Functionality is critical and time sensitive. There also must be no cosmetic imperfections in the manufacturing and assembly of the board. Applications for these types of boards include military, OEM automotive, and medical applications.

This standard also provides manufacturers with guidance on how to ensure the quality of their PCBs are meeting the necessary requirements
to label these boards in the right class. This includes the materials used to fabricate, design and layout, plating and surface finishes, testing and inspection, and quality control.

#### *4.1.1.4 - IPC-A-600 Standard*

IPC-A-600G is a visual inspection standard that sets the criteria required for the acceptability of bare PCBs. This standard complements IPC-A-6012 by also using the same 3 classes set by 6012. However, IPC-A-6012 is solely meant to set the requirements for performance and reliability of the PCBs. Whereas IPC-A-600 focuses on the cosmetic and workmanship quality of bare boards. This ensures that the board not only is performing as intended, but that customers are satisfied with the fabrication done on the board. The standard provides guidelines on the following visual inspections with acceptability criteria depending on the board's classification. As well as common defects to look out for during these inspections:

1. **Conductor Spacing:** Ensures proper electrical performance and prevents shorts from occurring.

2. **Plating Thickness:** The standard specifies the minimum thickness requirements for any metal features.

3. **Hole Size:** This ensures that a PCB has proper hole sizes for any components, connectors, and vias.

4. **Surface Finish:** The standard defines how the PCB should look when finished to ensure proper conductivity and prevent corrosion. This standard also describes techniques such as conformal coating.

5. **Cleanliness:** The standard ensures how to prevent contamination that would degrade the functionality of the board.
6. **Physical Damage:** Visually inspecting the board to ensure that the board has no cracks or warping that could impact performance.

This standard is helpful in our project because it will help us determine if we have a functional bare board to work with. It will help us identify any common defects in the manufacturing process so that we may take the appropriate steps to correct them before we begin assembling components. This will prevent any delays in the realization of our PCB.

#### *4.1.1.5 - IPC-A-610 Standard*

As a branch of IPC-A-600, this is also a visual inspection standard. However, this standard focuses on the criteria required for the acceptability of PCBs that have been fully assembled. This standard takes into account additional inspections such as soldering quality and component placement. This standard is crucial for us to understand to be able to properly mount our components onboard the PCB. Because we will be doing the assembly portion manually, we must train all team members in how to properly mount all SMT/through hole components to avoid potential damage to the board. The main guidelines from the standard are:

1. **Through Hole Requirements:** The standard addresses what an accepted criteria should look like for components that are through hole. This includes having proper component orientation, spacing, alignment, secure mounting, having enough lead length, and checking for damage to the component leads.

2. **Surface-Mount Requirements:** The criteria is similar to the through hole components, with the added criteria of having proper solder formation, wetting and fillet formation, no bridging or excess solder.

3. **Wire and Cable Assemblies:** There are guidelines on this standard for ensuring how to properly crimp terminations to ensure secure and reliable connections. It also goes over how to properly manage wires and cables to prevent damage.

4. **Soldering Requirements:** This standard goes over the visual inspection to ensure that every component is properly soldered to the board. This includes a detailed breakdown of how a perfect solder joint should look like. This includes solder joint formation, the shape of the joint, solder joint fillet, the cleanliness of the joint, and ensuring the joint has some strength.

5. **Cleaning and Coating requirements:** Ensuring the joints have been cleaned after assembly is crucial. Things to look out for include removing flux residues that could affect reliability, coating the board with conformal coating to protect it from moisture and dust, and checking for the thickness of the coating to ensure it is within an acceptable range.

6. **Marking and Labeling:** The standard goes over guidelines to having accurate labels and markings on the board so we are able to easily track the placement of components. This includes marking the components to identify them, marking the PCB, and marking important locations and orientations.

#### *4.1.1.6 - J-STD-001 Standard*

J-STD-001 is always good to be used in conjunction with IPC-A-610. While the IPC standard is focused on the visual inspection and acceptance of the assembled PCB, J-STD-001 focuses on the soldering process needed during the manufacturing and assembly of the PCB. This standard provides detailed requirements and guidelines to ensure we have high quality solder joints. Taking the extra step and learning more about how to properly solder is crucial for our PCB to ensure we do not spend unnecessary time on reworks. This will also further our understanding on what to look out for when it is time to assemble our board. The standard goes over materials, methods, and the verification criteria required for solder connections for both leaded and unleaded solder. This includes the following guidelines:

1. **Material, Component, and Equipment Requirements:** This standard specifies the acceptable materials for soldering such as what solder alloys to use, the types of fluxes, and what cleaning agents to use after soldering is finished.

2. **Soldering and Assembly Requirements:** There are detailed steps for what soldering techniques to use such as hand soldering, wave soldering, and using a reflow oven for soldering.

3. **Terminal and Wire Connection:**This standard goes over how to make secure and reliable connections with crimping and soldering techniques.

4. **Thermal Management:** The standard discusses heating and cooling rates to use to ensure they match the manufacturer's standards to prevent thermal excursions.

## 4.2 Communication Standards/Protocols

Our board will serve as the central processing unit for the thermal sand battery. Therefore, it is imperative that we are able to establish and maintain communication between the MCU and the peripherals. Our board must be able to constantly monitor the current state of its outside environment and the battery in real time. We have established the use of the following communication protocols that we require due to the selection of our parts:

- Single Wire Digital Communication
- Serial Peripheral Interface (SPI)

### 4.2.1 - Single Wire Digital Communication

The 1 wire single digital communication standard is a communication method that will be used by the DHT11 thermometers when communicating with the MCU. The 1-wire protocol is a single wire interface half-duplex, bidirectional, low speed and power, long distance serial-data communication protocol. Maxim's 1 wire technology is a serial protocol that uses a single line plus a ground connection reference for communication. A 1 wire master initiates and controls the communication with one or more 1 wire master 1 wire slave devices all of which would be connected on the 1-wire bus. The simplicity of this communication standard also shows that there is no use of a clock signal in its use, as the slave devices are generally internally clocked and already synchronized with the signal from the master device. The master device, the MCU in this case, is responsible for read and write operations of the slave devices so they cannot initiate a data transfer on their own. What they can do is to indicate their presence to the master over the bus when the master resets. This wire is usually limited to a maximum speed of 16Kbps. The bus of the wire allows the data wire to transfer power from the master to the slave device, but it might be limited to a certain extent.

Single Wire Digital communication also allows for two different powering modes as well, the first reason being why we see it mentioned in the maximum power settings of the ESP32 GPIO pin. The 'parasitic power mode' would mean that the components, such as the thermometers, just need two wires, the data wire and the ground connection. With the Parasitic power mode the power line is just connected to ground and on the controller we set the 4.7k ohm pull up resistor to be connected to the 1 wire but. This would allow the peripheral device to pull around 1.5 mA when doing temperature conversions, this can add some slight delays of up to 750 ms on the higher end where the controller can't do anything else. We can also use a normal external power supply approach where the peripheral decides to have its data connection wire but also has a dedicated power connection and ground connection. With the 'normal mode' the pull up resistor would still be required on the bus weir. When the bus is free for data transfer, the microcontroller will continuously poll the device that's doing the conversion. The reason we want the microcontroller to continually poll the device is so that we would in a sense always have the most 'up to date' information

possible, this would allow the peripheral to finish its calculation as soon as the device reports are done thus giving the system a lot of time. This is a very important consideration within a real time system because temperature readings and conversions are what we can consider a ‘hard real time system’ as we are operating a project that will reach very high temperatures and so we would need to ensure that all the readings we get are accurate as opposed to garbage information.

The 1-wire communication system also needs each device to include a unique 64 bit “ROM” address, this consists of an 8-bit family code, a 58 bit serial number, and an 8-bit CRC. The CRC will be our second back up for the data we receive from our thermometers as it's used to ‘check the integrity of the data’. The MCU would have to first select the device we want to send a command to be using its unique ROM. The following commands will be responded to by the selected device if it is able to be identified by the ROM that it has hardcoded.

By having the multiple devices accounted for and with their individual data connections to the main board this also affords us a bit more control over the whole system. We can create programs that also address all connected peripheral connections by using a ‘Skip ROM’ command as opposed to needing to address each one individually. This approach is good in being able to simplify the code that we would have to write in the situation were a ‘broadcast’ type application is needed for peripheral devices, and it can make our application even faster and more secure since this would be a natively supported command that is within the standard that was set for the one wire communication protocol.
One important note that we need to consider however would be the power draw of the MCU.
Depending on how restricted the MCU and thermometer power supply is we can possibly operate in a scenario where we only design a power supply that is able to only power the MCU, the LCD, and maybe only a few thermometers at a time. The conscious approach towards power draw can be something that needs to be considered. The reason our thoughts have to go towards power management would be because we wouldn’t want to create a circuit which is somewhat limited in its power draw abilities, because we need this to be a very efficient system, and we can then overload the power supply we create because by using the ‘Skip ROM’ command we actually activate all the peripherals at the same time. By activating all the peripherals at the same time it can be that we do create a power demand spike. Some mitigation tactics for this would be to know the maximum power delivery abilities of the power supply we create and then we would either make sure that we create a power supply that can accommodate the amount of peripherals we want to power all at once, or we can create a similar ‘Skip ROM’ command/function which will be responsible for quickly iterating through all the peripherals we wish to get a reading from and then being able to return it to whatever function we want to get said information from. With this we would be able to emulate the ‘Skup ROM’ function while also being within the constraints of the system we had created and streamlined, and although it may

not be the ideal scenario it is still the best approach we can have to such a situation.

The two main processing steps found within the single wire communication protocol is a ‘conversion’ and then a ‘read scratch pad’. The conversion is a command that is issued from the MCU to tell the device to perform an internal conversion operation. When the command is issued the device will read an internal ADC where the process of reading the information is completed in a sense. The one wire command is noted as a ‘Convert T( 0x44) command’ issued in the OneWire library as a ds.write(0x44) where ds is functioning as an instance of the OneWire class. The length of the conversion process can vary depending on the revolution of the ADC and can be found in the individual datasheet of the ESP32, , which the ESP32 has a 12-bit resolution. For reference a DS18B20 takes roughly 750 ms for a conversion of 9-12 bit ADC readings. A benefit is how flexible the overall system can be, where while a conversion is actively happening the device can be polled again- which would be done with the ds.read() command in the OneWire library- to see if it has successfully performed a conversion. This continuous reading and polling can ensure the programmers that the program we are working with is able to have the most up to date information from its peripherals while also ensuring that we don’t risk our program receiving the wrong data because the conversion and polling was done manually and can risk being ‘mistimed’. In a system like the thermal battery is important because if garbage or incorrect values are read when getting information from the internal sensors we can risk having a battery that exceeds the temperature limits we had set in the code which can risk damaging the enclosure and even be a fire hazard to the area around it.

The Read Scratchpad is for when the data has been converted, scratchpad is a sort of temporary memory that copies the information that is converted so it can be read from. The scratchpad is noted as being accessible to be read at any time without a conversion command to recall the most previous reading.

The approach that is frequently taken with the single wire communication is that of ‘convert, wait, read’ which is simple and secure but can cause a few issues when we’re operating a real time system like the thermal sand battery. The biggest drawback to this approach is that with simple more linear code a peripheral device such as a thermometer will have to ‘hang’ and wait for the conversion to take place if a hardcoded wait time is integrated into the code, this problem is noted as being addressed with a multithreaded approach to our application which is great considering that we did want to take a multithreaded approach to the software that would be running on the ESP32. The reason this is important is because in the instance where the system we are designing has other functions that need to be performed such as user input, and data processing, many types of these functions can just wait for the conversion process to take place. This is why it would be best to have a sort of multithreaded approach with a mutual exclusion lock to ensure that when a function needs a

value the most up to date value is stored in a global variable, and that if the conversion is currently happening then the thread that polls the peripheral devices we would be processing that information to ensure that the global variable only stores the final converted value. The 750ms conversion time is something that can be slow but it is not impossible to work around.

The scalability issues of the ‘convert,wait,read’ approach is also something that we want to consider as well, as its something that can linearly increase the runtime of our program/overall system. In our instance we may have multiple peripherals/thermometers that we are reading data from or writing data to, this will all take time to perform. In the instance where a ‘Skip ROM’ command can be written we can take advantage of that capability, but a multithreaded approach which allows us to keep some global variables up to date with several threads running with the sole purpose of periodically polling and updating something like the temperature information will be important for us in the end.

### 4.2.2 - Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is a synchronous serial data communication protocol which MCU’s use to communicate with one or multiple peripherals. The SPI utilizes a master slave relationship where data is able to be sent and received simultaneously thanks to the separate communication channels the master (MCU) and slave (peripheral device) have connecting them. The SPI standard only allows for one master in a system although multiple slaves can be connected individually or even parallel connections using one SPI pin on the ESP32. The ESP32 datasheet denotes that the SD device interface conforms to the industry standard SDIO Card Specification Version 2.0 and allows a host controller access to the SoC, using the SDIO bus interface and protocol. In this context the ESP32 can act as a slave to the SDIO bus, although the host can access the SDIO-interface registers directly. This process allows a host controller to access the ESP32 via an SDIO bus protocol which allows high-speed data transfer all based around an SPI connection.

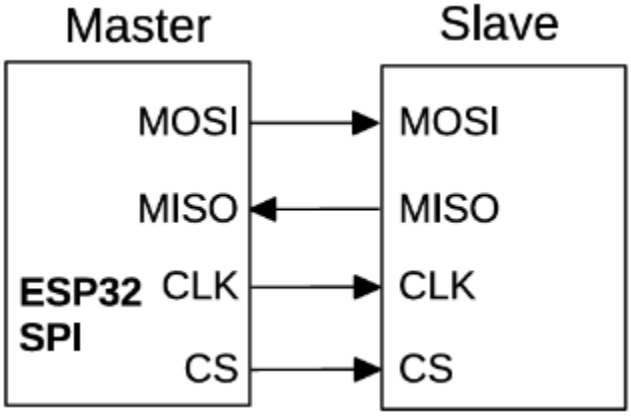


*Figure 4.2.2: Serial Peripheral Interface*

The set up of an SPI communication needs four lines:

- MISO: Master In Slave Out
- MOSI: Master Out Slave In
- SCK: Serial Clock
- CS/SS Chip Select (this is used to select a specific device when multiple peripherals are used on the same SPI bus)

Slave only devices like the display we would be using the pins are labeled as SDO and SDI which aligns with the MISO and MOSI lines.

Because of the above standards the ESP32 is able to interface with two different SPI buses and each bus can connect to three different peripherals. When having multiple SPI devices that use the same MOSI/MISO pins we would need to ensure that the slave devices have their own CS pin. With this the ESP32 can also have multiple SPI peripherals with their own SPI buses where each slave device has their own MISO/MOSI pin as well as their own SCLK and CS pins.

The SPI controller within the ESP32 is compatible with four-line full-duplex/half-duplex communication (MOSI, MISO, CS, and CLK lines) and three half duplex only communication (DATA, CS, and CLK lines) in GP-SPI mode. The SPI1-SPI3 controllers can communicate with other slaves as a standard SPI master. SPI2-3 can operate as both a master or slave. This works perfectly in the context of having systems where there may be commands/data that would have to be sent to our ESP32 device, this will be good because we will be able to take in and process the touch input that is returned by the LCD. The ESP32 master can be connected to three slaves simultaneously at most by default.

The SPI communication is also configured with four line half duplex mode, in this mode the SPI communication supports flexible communication formats such as ‘command+address+dummy phase+received and/or sent data’. The format of this SPI protocol would be in the format of

- Command: length of 0-16 bits MOSI
- Address: length 0-32/64 bits MOSI
- Dummy phase: length of 0-256 SPI clocks
- Received and/or sent data: length of 0-512 bits MOSI/MISO

The GP-SPI three-line half duplex communication differs from the four line half duplex communication in the way that the reception and transmission shares one signal bus and that the communication format must contain the command, address, received and/or sent data. This three-line half duplex communication is software configurable by manipulating the SPI_SIO bit within the SPI_USER_REG register.
The SPI data buffer uses 16 x 32 bits of data buffer to both buffer data-send and data-receive operations. When data is received with the SPI data buffer it is

received from the low byte of SPI_W0_REG by default and the writing ends with SPI_W15_REG. In the instance where the data length is over 64 bytes, the extra part will be written from SPI_W0_REG.
The most important thing when working with our SPI communication would be when the data signals of the ESP32 GP-SPI pin can be mapped to the physical pins as they are seen on the MCU and on the custom PCB we end up designing. The pins will be working with wil be found in either the the form of the IO_MUX or via the IO_MUX and its interfacing with the GPIO matrix. The input signals will be delayed by the two clock cycles when they are passed through the patrix. Output signals will not be delayed within the system that we made.

### 4.2.3 - I2C Interface (I2C)

The ESP32 MCU has two I2C bus interfaces where the MCU assumes the role of master in the I2C master-slave system. The ESP32 I2C system copports the following interfaces:

- Standard mode (100 Kbit/s)
- Fast mode (400 Kbit/s)
- Up to 5MHz, although limited because of the SDA pull-up ability
- 7-bit/10-bit addressing mode
- Dual addressing mode
- Supports both master mode and slave mode
- Supports multi-master and multi-slave communication

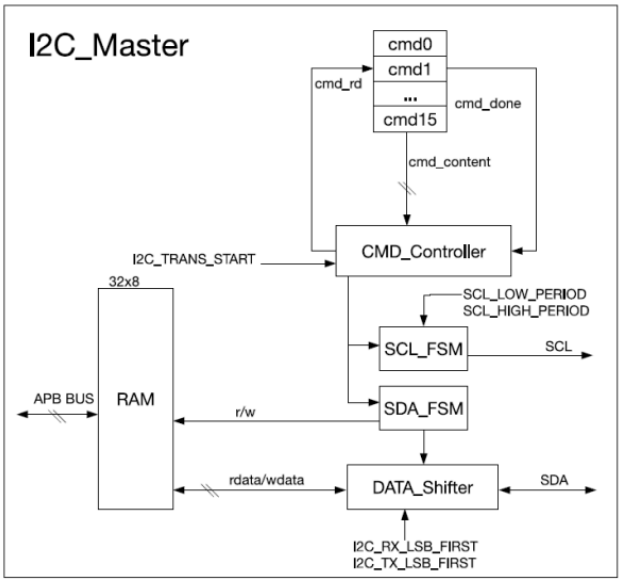



*Figure 4.2.1: I2C Master*

The I2C communication protocol consists of an SDA and SCL line in the form of a two wire bus. The I2C communication protocol uses the SDA and SCL lines to

open the drain output. Within the I2C communication protocol there are more than two devices on these lines, usually having one or more masters and one or more slave connections.

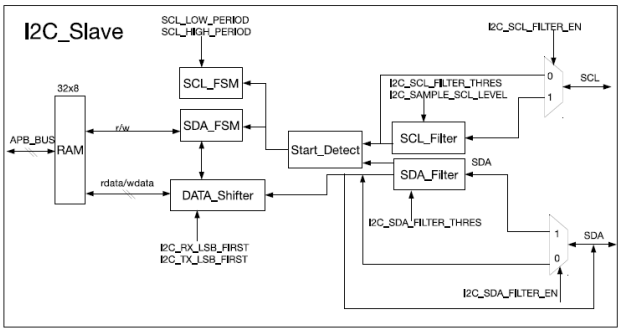



*Figure 4.2.2: I2C Slave*

The I2C communication protocol initializes by having the master send out a start connection message. The I2C communication standard indicates that the SDA line gets pulled low and the SCL line is pulled high. The I2C protocol uses a shift register which is cleared by having eight pulses to shift out the byte consisting of a 7-bit address and a read/write bit. The slave component can answer the master, which is trying to initialize the connection, by then pulling the SDA line low on the 9th clock pulse. In other circumstances the master is capable of sending out more than on 9-bit clock pulse which, depending on the bit that is sent whether its read or write, the device/master will shift out data thats connected to the SDA line. When the I2C connection is transferring data the SDA connection is only changed when the SCL line is in a low state. When the master no longer has information it needs to transmit it sends a stop condition to the slave devices by raising the SDA while the SCL is already high.

The ESP32 realizes all standards put forth to fulfill all I2C standards. The I2C controller is capable of operating in either master or slave mode, and this is controlled by the 'I2C_MS_MODE' register. The I2C controller within the ESP32 consists of 32 x 8 bits of ram and has direct mapping into the address space of the CPU cores, the memory address starting at REG_I2C_BASE+0x100. The I2C protocol stores data in a 32-bit word memory format, this would indicate that the second byte of data would be at +0x104, and the third byte would subsequently be found at +0x108, this information will be set by users using the register I2C_NOFIFO_EN. The I2C controller uses a CMD_Controller, this controller uses 16 command registers (cmd0-cmd15) which is used by the I2C master to control data transmission, this I2C controller thus executes one command at a time. The I2C controller is controlled by the SCL clock which is found with the SCL_FSM register, the I2C_SCL_HIGH_PERIOD_REG and I2C_SCL_LOW_PERIOD_REG are both used to configure the frequency and duty cycle of the signal that is sent through the SCL line. This allows us to be able to conform the I2C communication to exactly the settings that we need for our systems. The State machine that controls the SDA line will be found within

the SDA_FSM register. The ESP32 uses a ‘DATA_Shifter’ register which can convert the needed byte data to an outgoing bitstream, and it is able to convert any sort of incoming bitstream to byte data.the I2C_RX_LSB_FIRST and I2C_TX_LSB_FIRST registers is to be used for determining whether the LSB or MSB is stored/transmitted first. Lastly amongst the I2C controller parts we have the SCL_Filter and the SDA_Filter, this is meant to address some of the issues we may have with the input from the I2C_Slave register, the purpose of these being to filter the input noise from the slave device. The filter can be enabled or disabled by configuring I2C_SCL_FILTER_EN and I2C_SDA_FILTER_EN. The filter can remove any sort of line glitches that have pulse widths less than whats detailed in the I2C_SCL_FILTER_THRES and I2C_SDA_FILTER_THRES ABP clock cycles.

When we set the ESP32 I2C controller as the master the SCL will act as the output signal. When we set the I2C controller in slave mode the SCL becomes the input signal, will this approach the values assigned to the I2C_SDA_HOLD_REG and I2C_SDA_SAMPLE_REG are still valid and accessible in slave mode. To ensure that the host characteristics of the I2C slave are set properly to ensure the slave is able to receive data properly, the pins that are to be set would be the I2C_SDA_HOLD_TIME and the I2C_SDA_SAMPLE_TIME registers. The last thing to consider with the I2C controller set up is that the I2C pads are set in a form of ‘open-drain’ mode, this means that the signal can take longer to transition from a low signal to a high signal level. The transition length is determined by the combination of the pull-up resistor and capacitor. The output frequency of SCL is relatively low as a result of being put in the drain mode that is the default approach of this system.
The I2C controller has the industry standard cmd Structure which is only active when the I2C controller is in the master mode. The op_code of the command is used to indicate what actual command is being used.

The Master-Slave transmission process follows the I2C protocol as well. The first byte the master sends on the transmission wires will always be the destination Slave address. The first byte of our RAM unit is holding the slaves 7-bit address with the 1 last bit being reserved as the 1-bit read/write flat, this indicates that when the flag is zero we will be operating with under the premise of a write operation and that with a one we are operating under the premise of a read operation. For the actual master controller to begin operation the bus must have the SCL line in an active high position indicating that it is not being utilized by any other device on the I2C bus. When the data that needs to be transmitted exceeds the value in the I2C_NONFIFO_TX_THRES then an interrupt will be generated and stored within the I2C_TX_SEND_EMPTY_INT register. When the interrupt is detected the TXFIFO_END_ADDR value in the EXFIFO_ST_REG can be read by the internal software, this can get us the last address of the data in the RAM and refresh the old data in the RAM.

When the master sends the ‘START’ bit the slave starts listening and receiving the address so it can compare it to its own value. In the instance where the Masters sent address doesn’t match the slaves built in address then the slave ignores the following transmission. If the sent address does match that of the Slave then it will receive the rest of the data and store it in its built in RAM. The register RX_FIFO_START_ADDR is refreshed only one time throughout the entire transmission time. The I2C master component can transmit up to (14*255-1) butes of valid data when the END command is not used, and the ‘cmd’ unit ends up being populated with RSTART+14WRITE+1 STOP

The standard for I2c addressing as its also operating within the ESP32 I2C controller uses a 7-bit addressing, but the ESP32 I2C controller is also capable of 10-bit addressing. When the ESP32 is acting as the master controller of the I2C system the ESP32 would send a second I2C address byte after the first address byte is sent. The slave component would then have its address registers activated with a 10-bit addressing mode setting. The Slave address setting would be changed using the ‘I2C_SLACE_ADDR register, this register is what we would want to use to configure whether the slave would be using a 10-bit or a 7-bit address.

The I2C standard for data transmission from the Slave to the Master also depends on the address settings that are set up within the master and slave componends of the overall system. When the slave has a 7-bit address the master starts by sending the 7-bit address to the slave using the ‘cmd1’ WRITE command, the byte the command sends with its information is a slave address plus the R/W flag which would be a 1 in this case indicating that the master will be using a read operation. After the byte indicating the address and R/W operation is sent the slave starts sending the data to the master so long as the master address matches the address that was sent by the slave component. The master then returns ‘ACK’, according to the I2C standard the ack_value within the red command is there upon receiving every byte.

With all this the the two I2C standards that the ESP32 offers you allow the ESP32 to be operated in master or slave mode, the I2C protocol allows for full duplex and half duplex communication modes. The I2C communication protocol can also be configured to operate with 8/16/32/48/64 bit resolution as either an input or output channel. BCK clock frequency can be adjusted from 10kHz to as much as 40MHz. When one or both the I2S interfaces can be cinfogured when in the master mode where th emaster clock can be the output to the external DAC/CODEC. Both I2C communication standard instances have dedicated DMA controllers. PDM and BT PCM interfaces are also supported within the ESP32.

## 4.3 PCB Design Constraints for EMC Considerations

EMC, or electromagnetic compatibility, is a state where all electronic components on a system are able to be functional and be reliable without interfering with each

other. The goal of EMC is to reduce Electromagnetic Interference (EMI) inside and outside of the system. To ensure that we are able to successfully realize our PCB, it is important that we take EMC into consideration. By doing this, we are ensuring that proper zone placement is implemented and each component is properly placed in a section of the PCB that will maximize performance and minimize interference. It is also the most cost-effective method to be able to consider EMC constraints at the early stages of design. It may be more costly and time consuming trying to fix an EMC problem on a board that has already been fully assembled. This in turn will be detrimental to our other major constraint which is time. Therefore, we must thoroughly investigate all possible design constraints before we begin designing our PCB.

### 4.3.1 - Components to Use for EMI Reduction

There are three main sources that can cause problems for EMC. This includes a source of electromagnetic energy, some kind of receiver that is unintentionally being affected by EMI, and a trace path between them that is coupling that energy from the source to the receiver. This energy can be coupled conductively from an electric current, capacitively from an electric field, inductively from a magnetic field, or radiated from an electromagnetic field. To combat these scenarios, we can make use of passive components that can negate some of the effects of EMI. This includes the use of resistors, capacitors, inductors, and diodes.

1. **Resistors:** Resistors are an excellent way to be able to keep signal integrity and have line termination on driver inputs and outputs. By placing a resistor on outputs, we are able to help with the impedance matching of the input and output. This is essential when designing as a mismatch in line impedance can cause reflections. This is where we have signals bouncing back and forth on a trace that could interfere with a set voltage level. This could potentially cause some circuits to misbehave if another component is checking for an incoming voltage level to be able to turn on or off. Line termination is going to be more essential with high speed circuits such as digital circuits. We must also take into account the type of resistor to use. There are many resistors available for us to use, each having their own characteristics. For example, surface mount resistors tend to be used over leaded resistors due to low parasitic properties. Metal film resistors are more suitable for applications involving a high power density or circuits that require precise accuracy. This resistor should not be used at low frequencies as it tends to create parasitic elements. An important constraint to look out for is if we need to amplify any signals. Very high frequencies can increase the parasitic inductance on the resistor. This means we must have our gain resistors as close as possible to our amplifying circuit to minimize this inductance.

2. **Capacitors:** We will have to deal with noise interference from the systems we have in place and the outside environment our PCB is in. Capacitors are a good option for filtering system noise. When picking our capacitors, we must take into account the type of capacitor to use and how we will implement using them. It is important to use a capacitor that has a low equivalent series resistance (ESR). This will give us a higher attenuation overall. Having bypass capacitors is a functional way to create a path to ground that can clean up our lines. This should be considered around noisy areas such as our power supplies. We can also use capacitors to decouple noise to ground and provide a source of DC power for digital circuits like our MCU. This is beneficial as many digital circuits create noise from switching at high speeds. Similar to resistors, we must also take into account the type of capacitor to use for the right application. However, unlike resistors, capacitors can greatly differ in frequency response due to their dielectric materials. Aluminum and tantalum capacitors make good choices for bypass capacitors and in low voltage applications. While ceramic capacitors can be used for decoupling situations and are low in parasitic inductance. Feed through capacitors can cover a wide range of filtering applications like power decoupling and high frequency decoupling in our data, clock, and control lines.

3. **Inductors:** Inductors are another way to filter EMI in conjunction with capacitors.They are also useful in stabilizing any kind of voltage regulation. The most practical way to use inductors is with capacitors to create LC filters. We can use LC filters to separate any unwanted signals. However, we must also take into account what arrangement to put our LC circuits and determine the right application to use the for. The different configurations we can look for include the common LC circuit, a PI-Circuit, or a T-Circuit. The LC circuit is most commonly used when our source and load impedances differ largely in value. A PI-circuit is used when we need high attenuation and both the source and load have very high impedances. The T-Circuit should be used when we also need high attenuation but our source and load impedances are relatively low. In general we must know to place our inductor towards the lower impedance side and capacitors towards the higher impedances.

4. **Diodes:** Diodes will be essential in ESD protection and helping us control any transient spikes in our PCB. As with the other components, there are many diodes to choose from. We must determine which iteration to use in our design. Some common diodes include the rectifier, Schottky, zener, and TVS diodes. Schottky diodes can be useful when we have fast transient spikes. Zener diodes can be used for over voltage protection.

### 4.3.2 - PCB Components Placement

We must strategically place our components during the PCB layout phase of our design. This will help us in being able to easily assemble our board and pass EMC constraints. Before we begin placement, we must first know the locations of fixed pieces of our board like what enclosure we will use to fit on the board, where the mounting holes and connectors will go, and anything that has to interface with the peripherals of our system. Because these locations are fixed on the board, we must be able to build off of that in terms of components placement. We must also be aware of the noisiest circuits on our PCB. This tends to be the power supply system such as voltage regulators and DC-DC converters. As well as the MCU due to clock and switching characteristics. In general, there are guidelines to building segregation zones of where we can place components to minimize interference. The general zones are as follows:

**Analog Zone:** here we can place any analog devices that will be used on our board. To be even more specific, we can separate sensitive analog devices from others.

**Digital Zone:** Here we can place our digital circuits such as our MCU. It is a good rule to put high speed digital devices in the center of our digital zone to prevent interference between the analog zone and power supply zone. The MCU will be a big generator for noise. This noise usually comes from the clock of the MCU.

**Power Supply Zone:** Here we can attach VCC and any power switching and high current circuits. This would prevent interference to the digital and analog zones.

**Filter Zone and Connections:** Incoming signals from our peripherals can be placed here to be able to filter them from any noise that was received before being sent out to their respective destinations. It is a good rule of thumb to place any input and output devices and power connectors along one edge of the board. The farther apart they are, the more EMI noise voltage we could get. We should not place circuity between connectors and any peripherals should be as close as they can be to the connection onboard the PCB.

### 4.3.3 - PCB Layer Stack Ups

Determining the amount of layers required for our PCB will help us speed up the process of designing and help us determine other placement requirements such as vias. Generally we should take into account the allowed board size, the total components on the board, the device packages, and the total number of nets to be routed. For this project, we will not take into consideration anything more than 4 layers. It is essential to know that we must never have an odd number of layers. We must always shoot for an even number. We take the following into consideration when creating our design:

**1-2 Layer PCB:** This type is used for breakouts or simple circuits, we should avoid using 1-2 layers if we plan on using high speed connections such as USB or ethernet connections.

**4 Layer PCB:** This tends to be the standard for low to medium simple boards and breakouts. This type of PCB gives us more layers to be able to potentially apply a VCC copper layer, improving signal integrity and EMC performance. However, we must have a manufacturer that can create blind vias to make this work. A common design to look into would be having the top layer as a short trace, the 2nd layer as a ground plane, the 3rd layer as a power plane, and the 4 layer as another short trace. A large core must be put between the power and ground plane to reduce interference.

### 4.3.4 - Grounding System

When grounding our system, we will make use of the ground plane. There are other options to choose from such as minimal grounds and ground structures, but taking into consideration EMC constraints, a ground plane will be the most efficient. The ground plane is essential to and our first step in creating EMC on our PCB. We must take into account the ground returns from our high speed digital circuits and any low level analog circuits. They should never be mixed. In general, we must ensure that our analog, digital, and power ground return paths do not flow through each other's circuits. Keeping our ground leads short can keep our impedance low.

### 4.3.5 - Traces and Routing

PCB traces will serve as a way to deliver power and signals to all the components on board the PCB. It is imperative to follow certain design constraints to maximize efficiency and minimize EMI. The best common practices we must follow include: avoiding sharp right angles to prevent capacitance, keeping signal groups separate. This means we must keep high speed traces away from low speed traces. Analog traces must be separated from digital traces. We must keep all return paths the shortest they can be. If we use vias, we must use them sparingly as they can add some inductance and capacitance. If we do use vias, we can not use them in differential traces. Because we will be focusing on analog and digital circuits on our board, we take a more detailed look at specific routing guidelines for them:

**Digital Circuits:** We must control the rise and fall times, the duty cycle, and the frequency of the signals to minimize any reflections. If we are not using an IC input, we should terminate it to prevent noise generation.If our MCU has a crystal, it is best practice to run traces next to them as they also produce a lot of noise. It is also best to run high frequency traces next to a ground trace. We must also keep the high frequency traces away from the edge of our board. Differential

signals need to be routed adjacent to one another to create a cancellation of their magnetic fields.

**Analog Circuits:** It is recommended to keep analog circuits close to the I/O connector and away from any high speed digital, high current, or power electronics. We must route low level analog circuits in a dedicated section only. It is a good practice to apply a low pass filter to all analog inputs to clear out any high frequencies that are on the line.

### 4.3.6 - I/O Filtering and Cabling

As the main area of traffic on the PCB, we should ideally have some kind of shielded connectors. This way, we do not get unintentional signals delivered to the peripherals or any of our PCB zones. Some ways of shielding can include the use of shield cans to create a faraday cage. However, this would only block electric fields. To be able to filter magnetic fields, the use of ferrite beads can be implemented. It is best practice to have a shielded cable, connected to a shielded connector. From there the traces are routed through an EMI filter like any of the LC circuits described in 4.3.1.

## 4.4 Time Constraints

Starting our senior design project in the summer has considerably reduced the amount of time we have for every phase of this project. A regular semester has on average 17 weeks to be able to assemble a design team, gather reviewers, finalize a project idea, conduct research, and complete the design report with prototyping done at the end of Senior Design I. We were given around 11 weeks. This takes away 6 weeks worth of time to be able to complete the same requirements and phases. From the start of this project, time has always been a constraint. Because of this, we have had to manage our time efficiently and communication within our group has been a key component in being able to meet deadlines and milestones. Time considerations have always been included in every decision that we have made for this project. In a way, it has constrained our creativity for the project. We had many aspirations when the brainstorming for this project began. However, ideas had to be eliminated and our final goals for this design were decided with the vision to make this project feasible within our given time constraints.

To mitigate any risks associated with this constraint, we have always had at least one team member think one step ahead. Project management has been essential to ensure we don’t stay stuck in one area. When one milestone has been met, we meet to discuss the results of that milestone and ensure all team members understand where their focus is needed to meet the deadline for the next milestone. This has worked for the most part and has been met with little resistance from the team.

Luckily, most of the parts that have been required for this project are in stock and have a lead time of maximum 2 weeks. We use this knowledge to our advantage to give us some breathing room in being able to properly decide which components are required to minimize the chance of reworks. PCB fabrication lead time is also around 3 weeks. We keep this lead time in mind to be able to plan for future design finalization in Senior Design II. Our plan to make a few different iterations of our board (at least 3) is feasible even with our time constraint.

Perhaps the most affected design step from this constraint will be our software design. Our goal is to implement a user interface on the LCD display to be able to give a user friendly experience. This not only involves being able to establish communication between the peripherals and MCU, but also being able to code the interface. This will involve extensive coding, debugging, and testing. Lack of experience for some members will affect this step as well. To mitigate the risks of not being able to accomplish this goal, we have our computer engineers focused on this part of the project. We will also coordinate tag ups to be able to have all hands on deck for any barriers that our computer engineers may find. Having all 4 minds on this is better than having a single person working on it.

## 4.5 Other Standards and Constraints

### 4.5.1 - Heated Material Evaluation Standard

The first standard that helped guide our research was one that examined copper, stainless steel, and ceramic flow meters. Its title is Understanding Flow Meters, and the goal of this standard was to be able to understand and optimize the flow of temperature, pressure, and fluids through these materials. The standard used critical flow venturis (CFVs) to conduct measurements. These devices are traditionally used for flow measurement, energy management/optimization, and process control (moderating the flow of fluids).

As the bulk of our design relied on airflow over the heated copper tubes/coils through the sand enclosure, we used NIST's data in understanding temperature distribution applied to the copper, stainless steel, and ceramic devices. This NIST document shows clearly how, when the same heat is applied to copper, stainless steel, and ceramic material, copper has a thermal conductivity of 380 W/m*k, stainless steel was 16 W/m*k, and ceramic had 1.46 W/m*k. With copper having roughly 21 times the conductivity of stainless steel, and ceramics providing the least useful medium for heat of the examined materials, we were able to use this evidence to guide our decision of using copper conduit through the sand enclosure to transfer the heat from the sand to the air flowing through the conduit.

### 4.5.2 - Air Quality Standard

Within the sand heating process there are many components which heat can affect at 1000ºF. The degradation of parts, in this design, has the potential to affect the air quality maintained by the device. With environmental protection and public health in mind, we researched the following standard in order to make conscientious decisions with the materials involved in the design: SRM 2860 Phthalates in Polyvinyl Chloride.

This standard takes a look into PVC and the current toxicity research being conducted in phthalates, a substance in many plastics and other materials like PVC (Polyvinyl Chloride) which is a product used in a multitude of industries such as construction, electronics, and healthcare tools. The guided research is focused on the phthalate compound's hazard to human development and reproduction. This substance is most dangerous to children under the age of 5. Currently only 0.1% of any child toy is allowed to be PVC (by weight) because of the danger of children ingesting phthalates.

With this issue in mind, our design team chose to use as little PVC in the parts of the design that would involve high temperatures in an effort to limit the possibility of phthalates being released. We chose to use copper and stainless steel for ventilation/piping throughout the design as these metals do not release toxins at high temperatures. Copper can cause headache, nausea, and abdominal pain if burning copper is inhaled, however our project should not approach temperatures near that of burning copper.

### 4.5.3 - Thermocouple Calibration Standards

Thermocouples are a vital component of this design project. Our thermocouple uses include: reading internal temperature of the sandbox, providing temperature readings while the device is powered on. We will use the thermocouple(s) to monitor the sand and to establish a maximum temperature reading from the thermocouple which our ESP32 board will interpret. The ESP32 will then power down the heating element(s) when the sand has reached its maximum temperature. We chose the Thermocouples Calibrations Services standard to help guide our studying in understanding how thermocouples are calibrated and how accurate they are at a variety of temperatures, resulting in a better understanding of how to calibrate our thermocouples to maintain safe temperatures inside the sand enclosure.

This standard chose to use thermocouples with varying wire lengths with the shortest being 70cm, however for our project we will be using wire under 50cm. In our design project, we intend to use K type thermocouples. This standard measured types K, N, B, R, and S and compared their calibration(s) at different temperatures. For K type they measured the calibration voltages at temperatures -200ºC to 500ºC in increments of 100ºF. The largest calibration disparity was at 500ºC at .5uV which would roughly be our project's highest temperature reached. Compared to the other thermocouple types studied, this calibration voltage is

considered a small error. Our design required thermocouple calibration with the MAX6675 K type thermocouple, however the thermocouple's factory calibration allowed us to easily fine-tune the calibration for the device for our use.

### 4.5.4 - Fire Dynamics and Protection Standard

Safety is a large concern for the sand battery space heater in regard to fire hazards. This design is intended for residential homes built in climates of -30ºF to 65ºF where a large number of homes are constructed with wood components. In the United States of America, it is estimated that 94% of homes are built with a majority of wooden products. We made a large effort to research/abide by the following fire protection standards to verify that the device would be safe to operate in small residential and commercial spaces.

### 4.5.5 - Manufacturing Constraint

We considered the constraint of manufacturability for the project, especially as undergraduate students with limited access to tools and equipment. We were allowed to use a lab with related equipment such as oscilloscopes, signal analyzers, 3-D printers, and soldering benches, and we were able to use any power tools already owned by a member of the group. Awareness of this constraint limited the team to keep the design's construction relatively simple.

A positive from this limitation was that we were able to conduct more parts research allowing for us to find better parts which worked more cohesively with our design. A large factor in manufacturing was material availably and lead time. Although we were able to purchase parts better suited for our design, often we ran into an issue where parts were discontinued or lead time on the product(s) was too long for the senior design 1 & 2 timeline. We were also allowed to outsource the actual construction of our designed PCB with a manufacturer outside of UCF, a practice which is commonplace in the engineering field, which saved the team a large amount of effort in manufacturing. With availability of materials and the complexity of interacting systems, working within this manufacturing constraint tested and improved our ability as engineers to produce a complex design from start to finish within the timeline given limited tools.

### 4.5.6 - Fire Constraint

One main notable constraint we had for the design is the fire constraint. We would not be allowed for the design to create a safety hazard such as a fire in the sand enclosure when completed. The design needed to be functioning without any components breaking and without causing a fire. The design team chose to include fire safety measures when constructing the thermal sand battery considering the risk of fire or smoke at temperatures past 700°F where smoke and component-fires can become an issue. To put this in perspective, skin tissue can receive second degree burns from temperatures of only 162°F (flame, not

water), and it is vitally important that this product should not put any user in danger/at risk. It is estimated that 358000 house fires occur annually in the United States alone, causing more than 2000 civilian deaths, and our design included precautions to ensure safety around the device.

With smoke alarm type information from “Smoke Alarm Performance In Kitchen Fires and Nuisance Alarm Scenarios”, we were able to focus our research in finding the optimal ionization alarm to use. This document also clarified that such residential smoke detectors’ response time is inversely proportional to the distance from the smoke alarm to the source of the smoke. We kept this in mind while deciding the best location/placement for the smoke alarm attached to the thermal sand battery.

# Chapter 5: ChatGPT Comparisons

Today, many industries are beginning to implement Artificial Intelligence (AI) as a tool to be able to help engineers achieve results faster. However, this still comes with many limitations. AI may produce false information, especially when it comes to asking it to solve technical problems. This anomaly is known as “Hallucinations”. Professionals must take this into account if they plan to use AI for a project and take extra caution in verifying results for accuracy. We decided to ask ChatGPT some questions relating to our project to be able to compare and contrast the differences we found in the inaccuracy of its generated response. To ensure we are getting the best results that AI can give, we decided to use ChatGPT-4o. This is OpenAi’s most advanced model to date. The other platform that we compare the results with is Microsoft's newest AI platform, Co-Pilot.

**Case Study 1:** What is a thermal sand battery and how is it implemented?

ChatGPT 4o: A thermal sand battery is a type of energy storage system that uses sand as the primary material to store heat. It operates on the principle of storing thermal energy in sand during periods of excess energy production (such as from solar panels or during off-peak electricity times) and releasing it when needed, such as for heating purposes. This system leverages the high heat capacity and thermal conductivity of sand to store and retain heat efficiently.

### How It Works

1. **Heating Phase:**
    - Excess electricity is used to heat the sand to a high temperature, typically using resistive heating elements.
    - The sand's high thermal mass allows it to store large amounts of heat.
2. **Storage Phase:**
    - The heated sand is stored in an insulated container to minimize heat loss.
    - The sand retains the heat for extended periods due to its thermal properties.
3. **Discharge Phase:**
    - When heating is needed, the stored thermal energy is extracted from the sand.
    - This heat can be transferred to air or water and then distributed to the heating system of a building or industrial process.

### Implementation

1. **Selection of Sand:**
    - The type of sand chosen should have high specific heat capacity and thermal conductivity.

    - Fine-grained sand is preferred due to its higher surface area for heat transfer.
2. **Heating Elements:**
    - Electrical resistive heaters are typically used to heat the sand. These can be powered by renewable energy sources like solar panels or wind turbines.
3. **Insulation:**
    - The storage container must be well-insulated to minimize heat loss. Common insulation materials include fiberglass, mineral wool, and other high-temperature resistant materials.
4. **Heat Exchange Mechanism:**
    - A heat exchanger is used to transfer the stored heat from the sand to the medium (air or water) that will carry it to the point of use.
    - This can involve blowing air through the heated sand or using a water loop that circulates through pipes embedded in the sand.
5. **Control System:**
    - A control system is necessary to manage the charging (heating) and discharging (heat extraction) cycles.
    - It ensures that the sand is heated when excess energy is available and that heat is extracted efficiently when needed.

## Benefits

- Efficiency: Thermal sand batteries have high energy storage efficiency due to the excellent thermal properties of sand.
- Sustainability: They can be integrated with renewable energy sources, reducing reliance on fossil fuels.
- Cost-Effectiveness: Sand is an abundant and inexpensive material, making these systems cost-effective.
- Scalability: They can be scaled to different sizes, from small residential systems to large industrial applications.

## Sources:

ChatGPT could not find the proper sources with the URLs, therefore we can not possibly conclude that what the AI is saying is true. When asked to properly cite its sources, it just gave us some general places to look for the information like the Duke Energy website and Energy Storage

Microsoft Co-Pilot: A thermal sand battery is an innovative energy storage solution that uses sand to store and release thermal energy. Here's a brief overview of how it works and its implementation:

**How It Works**

1. Heating the Sand: The sand is heated using renewable energy sources like wind or solar power. This is done through resistive heating, where electricity heats the air, which in turn heats the sand[12].

2. Storing the Heat: The heated sand is stored in an insulated silo, where it can retain heat at temperatures around 500-600°C for several months[12].

3. Releasing the Heat: When needed, the stored thermal energy is released by circulating air through the hot sand. This hot air can then be used for various applications, such as heating homes, offices, or industrial processes[12].

**Implementation**

- Location: The world's first commercial sand battery was implemented in Kankaanpää, Finland. It consists of a 7-meter steel container filled with 100 tonnes of sand[12].

- Usage: The stored heat is used to provide heating for homes, offices, and even a public swimming pool. It can also be directed to generate industrial steam[12].
- Efficiency: The system is highly efficient, with very little energy loss, as long as the heat is not transported over long distances[12].

Conclusion: ChatGPT was more thorough in the implementation of the thermal sand battery, while Co-Pilot was able to successfully cite its sources. Both Ai's successfully showed accurate information about how to implement and what a thermal sand battery is. However, this could only be verified because we know sufficient information about the technology associated with sand batteries. If a regular user asked these questions, there would be no way to know if ChatGPT was accurate if no sources were given. Therefore, Co-pilot is the better option in this case study done.

**Case Study 2:** I am using the ESP-32WROOM to be able to implement a PCB to control the thermal sand battery. However, I must know what standards and constraints to take into account when building this board. Tell me some standards and constraints I should worry about and research to be able successfully build a working board. Cite your sources by giving me the URL of where you found the information

**ChatGPT4o:** When designing a PCB to control a thermal sand battery using the ESP32-WROOM, you need to consider various standards and constraints to ensure the board is reliable, efficient, and compliant with industry practices. Here are some key standards and constraints to take into account:

**Standards:**

1. IPC Standards:
    - IPC-2221: Generic Standard on Printed Board Design.
        - Covers design principles for PCB layout, ensuring manufacturability, and reliability.
        - URL: IPC-2221 Overview
    - IPC-6012: Qualification and Performance Specification for Rigid Printed Boards.
        - Specifies performance and qualification requirements for rigid PCBs.
        - URL: IPC-6012 Overview
    - IPC-A-610: Acceptability of Electronic Assemblies.
        - Defines the acceptability criteria for electronic assemblies.
        - URL: IPC-A-610 Overview
2. IEC Standards:
    - IEC 61000-4-2: Electrostatic Discharge (ESD) Immunity Test.
        - Specifies testing and performance criteria for ESD protection.
        - URL: IEC 61000-4-2 Overview
    - IEC 61000-4-4: Electrical Fast Transient/Burst Immunity Test.
        - Specifies requirements for immunity to electrical fast transients.
        - URL: IEC 61000-4-4 Overview
3. ISO Standards:
    - ISO 9001: Quality Management Systems.
        - Specifies requirements for a quality management system to ensure consistent product quality.
        - URL: ISO 9001 Overview

**Constraints:**

1. Electrical Constraints:
    - Power Supply: Ensure the power supply meets the voltage and current requirements of the ESP32-WROOM and associated components.
    - Signal Integrity: Maintain signal integrity by following proper trace routing, impedance control, and minimizing noise.
    - Thermal Management: Implement adequate thermal management techniques to dissipate heat generated by components.
2. Mechanical Constraints:
    - PCB Size and Shape: Ensure the PCB fits within the physical dimensions of the intended enclosure.
    - Component Placement: Optimize component placement for ease of assembly, maintenance, and signal routing.
3. Environmental Constraints:

    - Temperature Range: Design the PCB to operate reliably within the expected temperature range of the application.
    - Humidity and Dust: Consider conformal coating or encapsulation to protect the PCB from environmental factors.
4. Regulatory Constraints:
    - RoHS Compliance: Ensure the PCB complies with the Restriction of Hazardous Substances Directive.
    - EMC Compliance: Design the PCB to comply with electromagnetic compatibility (EMC) regulations to minimize electromagnetic interference (EMI).
5. Safety Constraints:
    - Isolation: Implement proper electrical isolation to protect users from high voltages.
    - Fault Protection: Design the PCB with protection mechanisms such as fuses and transient voltage suppressors to handle faults safely.

Microsoft Co-Pilot: When designing a PCB for the ESP-32WROOM to control a thermal sand battery, there are several standards and constraints you should consider:

**Key Standards and Constraints**

1. PCB Layer Design:
    - Four-layer PCB: Preferred for better performance. Layers typically include:
        - Layer 1 (TOP): Signal traces and components.
        - Layer 2 (GND): Dedicated ground plane.
        - Layer 3 (POWER): Power traces and some signal traces.
        - Layer 4 (BOTTOM): Minimal signal traces, no components[1].
    - Two-layer PCB: Can be used but with more constraints on layout and performance[1].
2. Power Supply:
    - Ensure proper routing of power traces, preferably on the inner layers.
    - Use multiple vias for power traces to ensure stable connections[1].
3. Ground Plane:
    - Maintain a continuous ground plane to minimize noise and interference.
    - Avoid splitting the ground plane, especially under the RF and crystal areas[1].
4. Antenna Placement:
    - Position the module’s antenna outside the baseboard if possible.

- Ensure a clearance of at least 15 mm around the antenna area to avoid interference[1].

5. Component Placement:
    - Place critical components like the crystal oscillator close to the ESP-32WROOM module.
    - Minimize the length of high-frequency signal traces to reduce signal degradation[2].
6. **Thermal Management:**
    - Consider the thermal properties of the PCB material, especially since the thermal sand battery may generate significant heat.
    - Use thermal vias and heat sinks if necessary to manage heat dissipation[2].
7. **EMI/EMC Considerations:**
    - Implement proper shielding and filtering techniques to minimize electromagnetic interference.
    - Follow guidelines for trace width, spacing, and grounding to ensure compliance with EMI/EMC standards[2]

Conclusion: In this case study question, ChatGPT was able to successfully reference the standards that are associated with PCB designs. It listen some

# Chapter 6:Hardware Design

## 6.1 Controller Block Diagram

Our project will consist of two main PCBs built. With an additional PCB that will be optional. One will be the AC-DC converter which will provide us with a 3.3V power supply to the board by converting 120VAC from a power outlet. The second will be our main board. The optional board will be a simple LED board that will give the user visual indicators of charging and discharging status. However, this board will be a stretch goal as it is not required to have a feasible design. This was thought to just improve the overall user experience. The main board will house the ESP32-WROOM and will serve as the central processing unit for all the peripherals and control units required. To make the design process flow smoother, we had to decide what functions would be required onboard to make our PCB work. We started with building subsystems. These are illustrated by the block diagram below. Each subsystem will house its own circuit and schematic that will perform a specific function to ensure we have a working board. In Chapter 8, we combine all the different schematics and lay them out onto one master board.

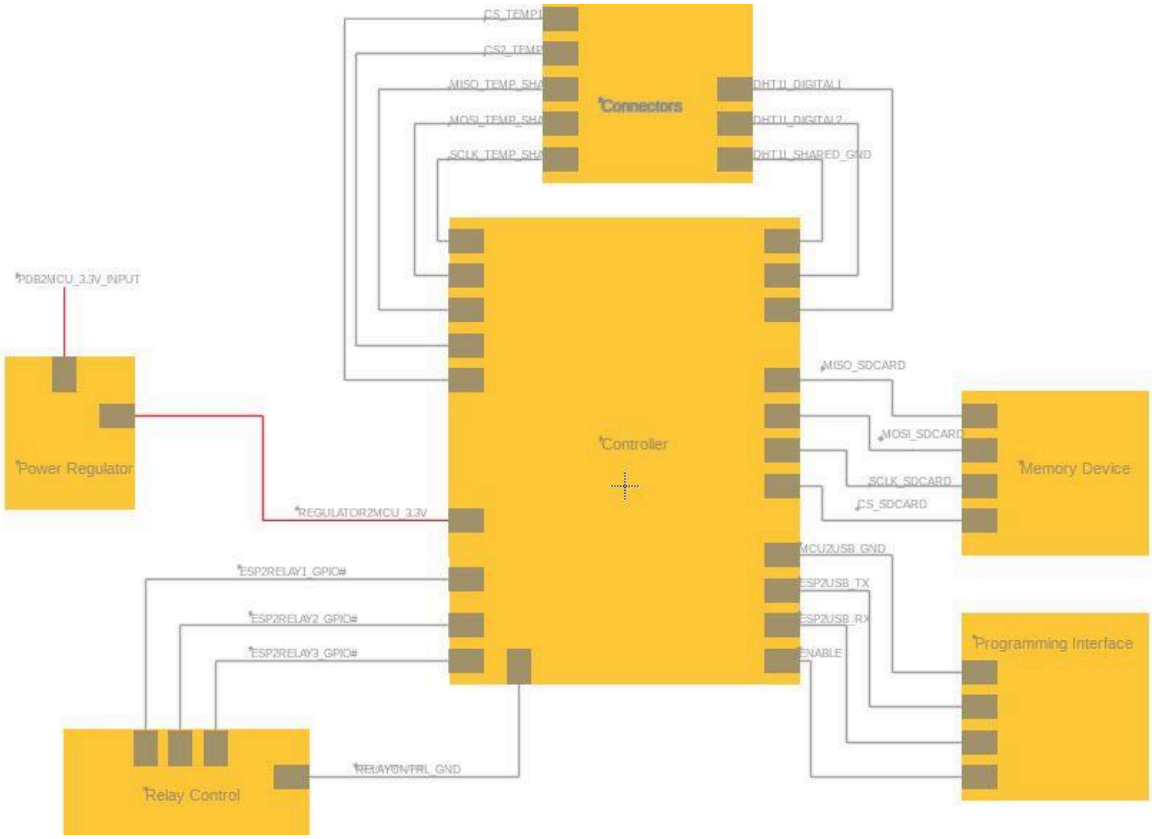



*Figure 6.1: Subsystem Block Diagram*

1. **Controller** - This block diagram consists of the ESP32-WROOM and any digital electronics that would be needed to accomplish communication to the peripherals.

2. **Programming Interface** - The ESP32 chip is going to need a way to be programmed. Therefore, we created another schematic for a circuit that will allow us to interface with the ESP32. This will allow us to simply connect to our computer and open up an IDE such as Ardunio sketch to implement the code used from prototyping.

3. **Power Regulator** - While we will have another PCB that will convert 120VAC to the desired 3.3V DC, it is a good practice to have additional components to serve as a way to keep our voltage input stable. This way, we can protect our components from damage.

4. **Connectors and Relay Controls** - These connectors serve as a way to provide a reliable connection from the peripherals to the ESP32. Our relay controls will also be used to be able to control the ON/OFF setting on the thermal sand battery for overheating and user control settings.

5. **Memory Device** - This block is for now optional. It is not a requirement to be able to accomplish our stated goals and have the battery work as intended. However, if we can make our stretch goals, we will need a memory unit to be able to store data and transfer that data to a cloud via the ESP32's wifi module. By thinking ahead, we are allowing ourselves more time to make our stretch goals feasible if time constraints allow.

## 6.2 Controller Schematic

The controller schematic holds our ESP32. To simplify each circuit, we made the use of labels that will connect our traces to the other schematic sheets. The ESP32 will make use of both of its SPI connections for our thermocouple sensors and I2C single wire digital for the DHT11 sensors. Because of this, we must take into consideration the distance that these communication protocols can take. SPI is only recommended for short distances. To solve this problem, we have decided to attach IC buffers and level shifters to be able to drive our signal through the lines. We are still in the process of trying to find an IC buffer for the SPI connections that will match our voltage and current specifications. For driving our I2C communication, we are looking at using an I2C differential driver. This will improve our range and our signal integrity of the signal line. This is imperative as we can not lose communication to the peripherals. The signal also must arrive on time without delay so our ESP32 can make a precise adjustment when heating or charging.

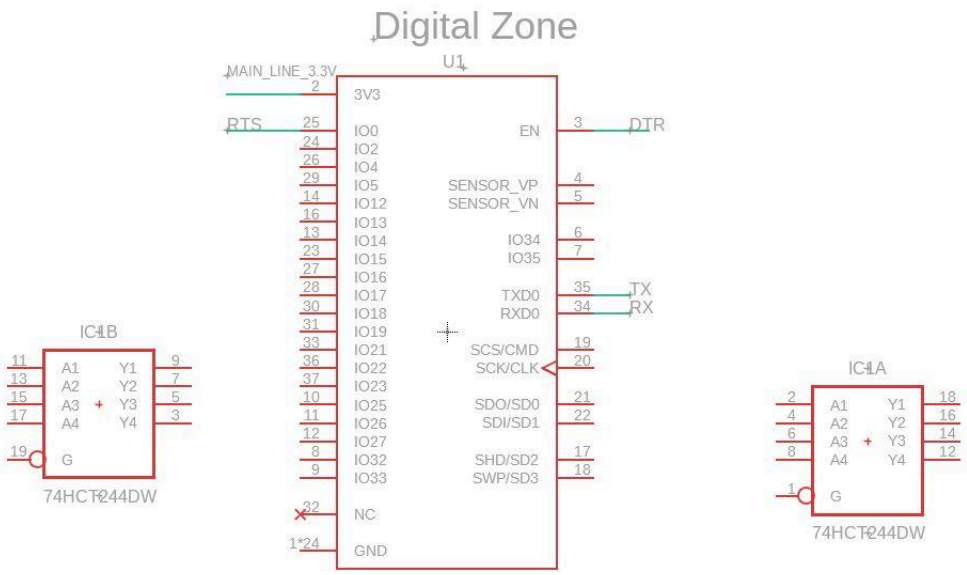



*Figure 6.2: Controller Schematic*

## 6.3 Programming Interface

We wanted to use a USB to be able to program the ESP32 chip. Therefore, we started to look at what would be required to be able to successfully connect to this. Unfortunately, the ESP32-WROOM uses UART to be able to program it. This gave us an opportunity to learn more about how to convert USB signal to UART. Our first idea was to simple create a port that would connect the UART lines on the ESP32 to an external FTDI cable. This cable already has the circuitry on board tha would allow us to seamlessly connect via USB. However, we wanted our board to be more modular. Not everyone will possess an FTDI cable to program the board and we wanted any user to have the ability to simply connect with a regular USB type B cable to a laptop. The next option was to have a USB to UART Bridge onboard our PCB. This IC chip will convert our D+ and D- from the USB to TX and RX signals and establish a reliable connection to the board. We decided to go with the CP2102 USB to UART Bridge as this is the most compatible chip for the ESP32. Our computer engineers also have all the drivers required for that particular chip which will speed up the process of creating a connection. For ESD protection, we added some Schottky Diodes that will keep signal integrity of the data and power lines. Another interesting feature of the CP2102 is that it has an internal LDO regulator that will convert the 5V from the USB port to 3.3V to power the ESP32. This is an excellent addition we wanted to have since it will give us multiple ways to power our board. To allow us to feed 3.3V to the MCU, we were required to add a 4.7uF and 0.1uF capacitor to the VDD line per the data sheet. This serves as a way to decouple the line and stabilize our voltage. After doing some research, we were able to identify the 4 main lines required to establish communication to the ESP. These include the TX and RX lines, DTR, and RTS. The Data Terminal Ready Line is used to signal whether or not our ESP is ready to communicate. This lets the CP2102 know

when it is time to transmit data to our MCU. This line will be connected to the EN pin of the ESP32 to reset the chip every time a new program is being installed. The RTS line or the Ready to Send is meant to be our flow control line. This line controls the flow of data and will either pull high or low depending on whether our ESP32 is ready for more data or can not accept more data and to pause transmission. We connect this pin to the IO0 pin of our ESP32 that will pull it high or low if we want the MCU to go into boot mode. While the TX and RX lines are fairly easy to connect, we are spending more time trying to find the right configuration for the RTS and DTR lines. We have decided to put transistors on the output of these lines to be able to act as a switch for the CP2102. This will help us keep signal integrity and will pull the lines low or high to tell CP2102 whether the ESP is ready or not and if the CP2102 must pause transmission. Transistors were decided because they are excellent when wanting to switch at high speeds for digital communication. We are still working on the right configuration that is needed to ensure this works properly. We are taking into consideration adding pull up resistors or some multiplexers that would be more simplified in routing a signal.

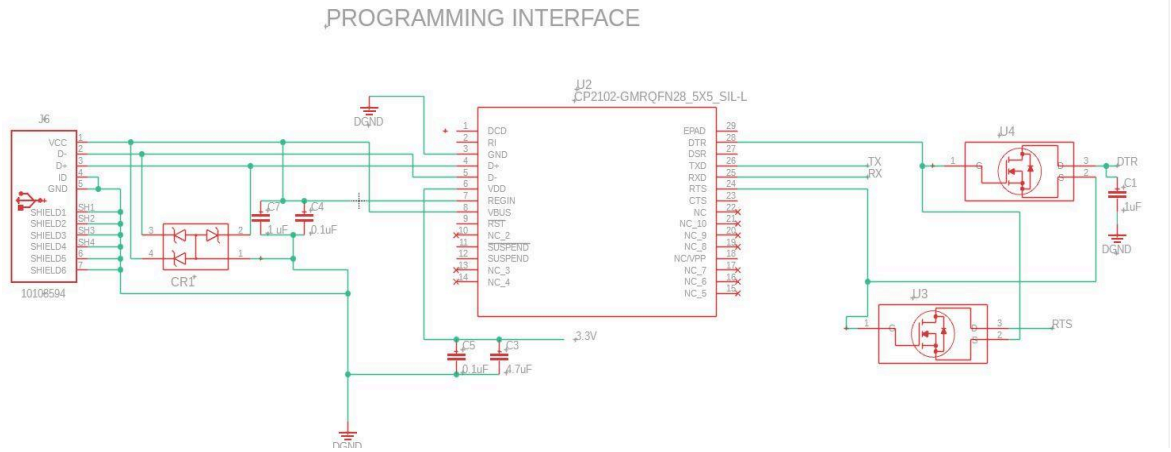



*Figure 6.3: Programming Interface*

## 6.4 Power Regulator

The main source of the input will come from our custom built AC-DC converter. This converter will have an LDO that will regulate the voltage to a 3.3V level. However, we want to ensure that we have effectively cleaned up our power to be able to protect the ESP32 and any components sensitive to noise. Our power will be traveling through a cable to be able to reach our main PCB. This can create an opportunity for noise to inject in the signal. Therefore, we have added some EMI filtering on J1 power input. We first filter our power through ferrite beads. This is excellent in first attenuating any high frequencies that our signal may have. We have used a ferrite bead at 100 ohms @ 100MHz. The signal will then be filtered through two capacitors. These capacitors will help filter out any lower

frequencies and stabilize the fluctuations in our voltage. We then reach two Schottky Diodes. These diodes were put in place because we will have two sources of power on a single line. One from our main power source and the other from our USB connection. These sources of course will not be powered on at the time. However, when one source is powered on, we want a way to prevent that power from leaking into the other circuit. This could cause some irratic behavior and could unintentionally cause damage to them. We decided to go with Schottky diodes due to their very low voltage drops. This particular model has a voltage drop of less than 0.2V and a built-in enable line that will turn on the diode when enabled high. The way this works is that when one source is turned on, it will also enable high, allowing the diode to be forward biased and current to flow through. If the enable is low, then the diode will simply block voltage from both sides, essentially creating an open circuit and protecting the circuit.

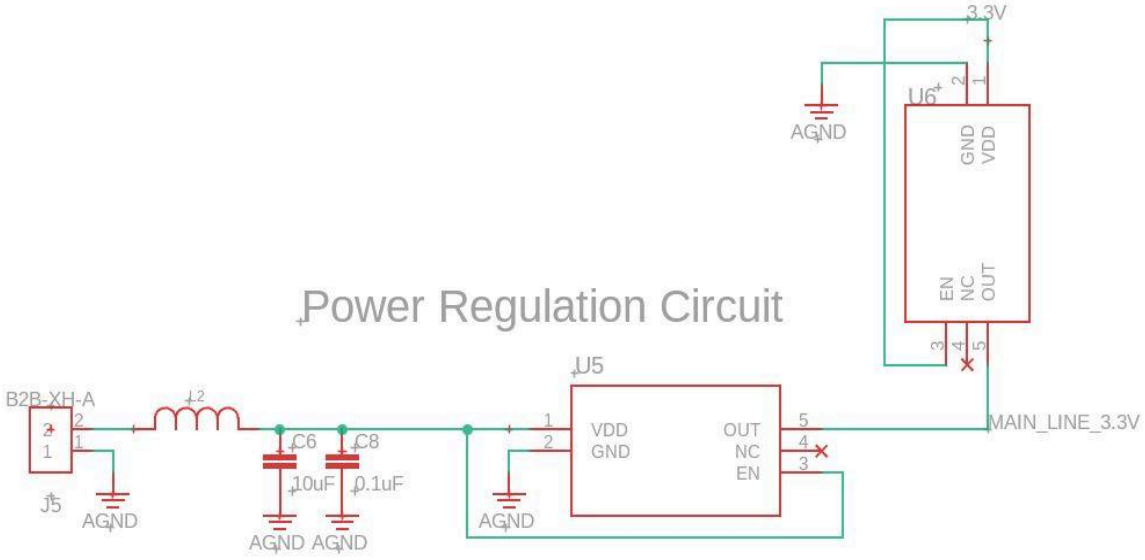



*Figure 6.4: Power Regulator*

## 6.5 Connectors and Relay Control

These Connectors will serve as the main line between the PCB and our sensors which will be offboard. We have established connectors for the 3 Relays we will use to control the main power of the sand battery and the two valves that will open and close depending on whether the battery is heating or charging. We will also have a 14 FFC cable that will connect to the LCD display to be able to interface with the ESP32. We then have our 4 pin JST connectors that will be used for the external thermocouples and sensors. These connectors were chosen as they are able to be clipped in place to make a secure connection. This schematic is still being worked on, but our plan is to add EMI filtering to each

connections to help filter out any noise that may have been accumulated from the outside environment.

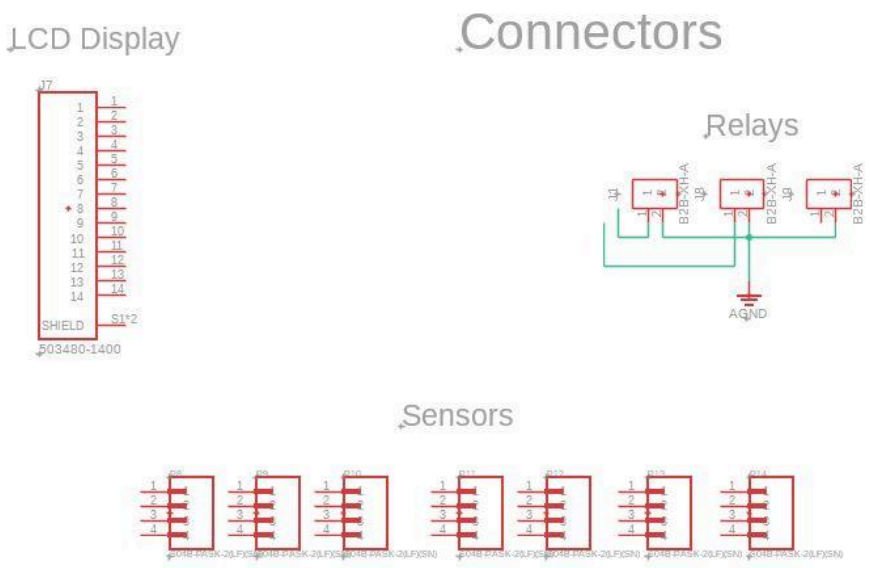



*Figure 6.5.1: Connectors Used*

## 6.6 AC-DC Converter PCB

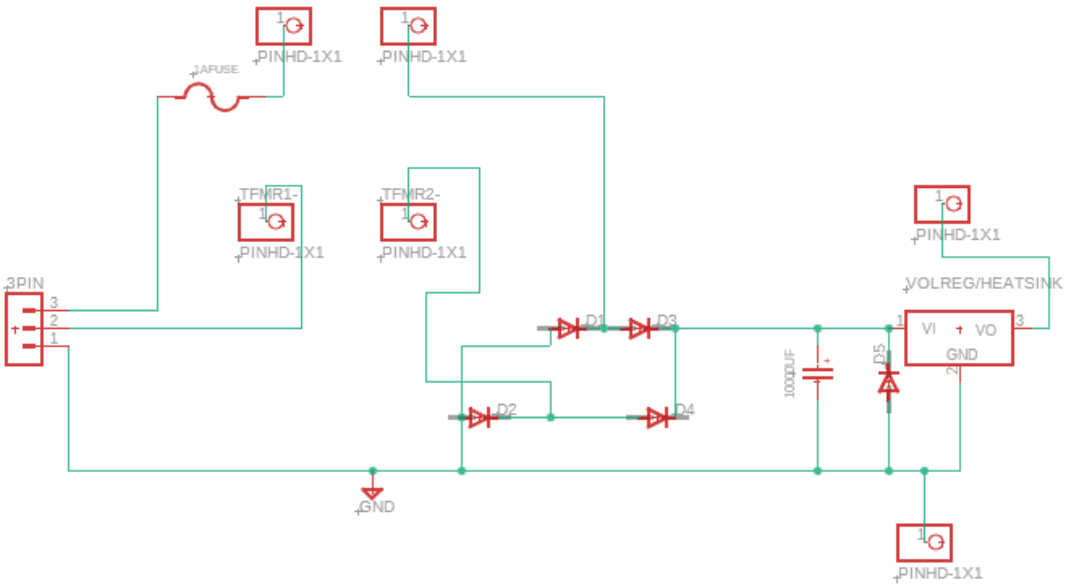


*Figure 6.6: AC/DC Converter Circuit*

## 6.7 Frame/Sand Enclosure

Our project will consist of two main boxes, the interior cube will be the sand enclosure with heating elements, thermocouples, copper tubing, and with the HVAC exhaust ventilation connected on the top and bottom of the enclosure. The outer box/frame will enclose the sand enclosure/HVAC tubing, all electronics onboard such as the PCB and AC/DC converter, the insulation around all heated mechanical parts (sand enclosure and HVAC piping), and will contain all wiring between electrical components. Below the frame are 4 wheels rated for 200 lbs, and on the top of the frame we attached the wooden table top.We piped two copper tubes from the bottom of the sand enclosure box through the top of the enclosure. We made sure to install the 1 inch copper piping after we installed the heating elements in the enclosure. This was in order to install the copper through the center of the heating elements. We chose the placement through the circular heating elements would allow for the elements to heat the sand localized close to the copper tubing first and the heat would radiate outward as time when on. This is useful for when the user needs immediate heat while the unit has not been charged. As the heating elements can reach temperatures over 200ºC in under 30 seconds, we utilized their proximity to enable the unit to output heat as quickly as possible. The motivation behind this aspect of the project was to be able to heat a room/cabin with an efficiency/speed of a furnace. Copper was the best option for thermal conduction and structural strength at high temperatures. Copper can be easily bent and shaped, which was advantageous to the team as the ends of the pipe sticking out of the enclosure would need to be beveled outward. This beveling would increase airflow through the two 1 inch copper pipes as air could be pushed through a larger surface area while entering/exiting the sand enclosure.

The frame itself will consist of a a 36 x 24 x 24 inch rectangular cube with a wooden board on top of the frame. This size allowed for 6 inches on 4 faces of the sand enclosure cube for components such as the HVAC pipes, thermocouple wiring, mounts, and insulation. These dimensions also allowed for our PCB, AC/DC converter, and our LCD mount to be placed at one of the ends of the design furthest from the center of the sand enclosure. This extra distance was critical as these components are rated for much lower temperatures than the sand/heating elements, therefore it was important to mount them as far from the main heat source as we could.

We also insulated each piece of metal that would be heated to a high temperature, ensuring no heat remains outside the sand enclosure box inside the main unit’s frame. To support the sand enclosure, we used steel brackets and braces in order to lift the enclosure 6 inches. This enabled us to fit the HVAC pipes underneath and left room for us to insulate the enclosure as well as the piping. We included the same brackets along the top of the enclosure to provide

support to the top of the top of the frame (the “coffee table surface”). This allows the device to have a larger load bearing capacity so that the consumer can store heavy objects on top of the coffee table.

We installed exhaust ventilation covers on both openings to the HVAC pipes to block large objects from entering the pipes as an obstruction to the airflow with flammable material could result in a fire. The ventilation covers were made from stainless steel and included metal mech behind the main opening to provide extra protection from larger objects entering the pipes. Behind the ventilation cover, we places the exhaust fan and oriented it to push air coming through the vent through the rest of the ventilation pipes of the device, increasing the rate for heat to be expelled from the device.

The frame rests on four wheels rated for larger weight in order to move the heavy unit easily in the consumer’s home. This will also help the user by allowing them to heat different rooms with the unit or charge the unit with different circuits in the home. Our design will utilize 11-12A on the circuit it is plugged into, so in certain scenarios the user may need to distribute power to different circuits within the home to not exceed their 15A breakers. The team installed handles on the long sides of the frame to be able to lift the unit comfortably and ergonomically. Another benefit to our design is the device’s ability to radiate heat after being unplugged. If the air valves are open when the user unplugs the device, the thermal sand battery would slowly dissipate its heat through the ventilation. This is not the intended use for the device, however it can be useful during power outages in cold climates.

We wanted the sand battery coffee table to be a multifunctional piece of furniture combining comfort and aesthetics. The tabletop of the thermal sand battery was attached with brackets connecting the wood and metal surfaces. This is a 48 inch by 30 inch board for the consumer to use as a table. Thinking of most furniture configurations in American homes, we designed the thermal sand battery so that the exhaust air would be exiting on the smaller side of the coffee table. In future designs we will build exhaust venting in all four directions low enough in the frame so the exhaust air cannot warp the wooden tabletop over time. Our simple rectangular cube design allows for the surfaces to be easily cleaned. We decided on a wood tabletop to add warmth to the design and to soften the design of the piece of furniture. The wood would hopefully inflict less damage than metal if the user were to bump into the table.

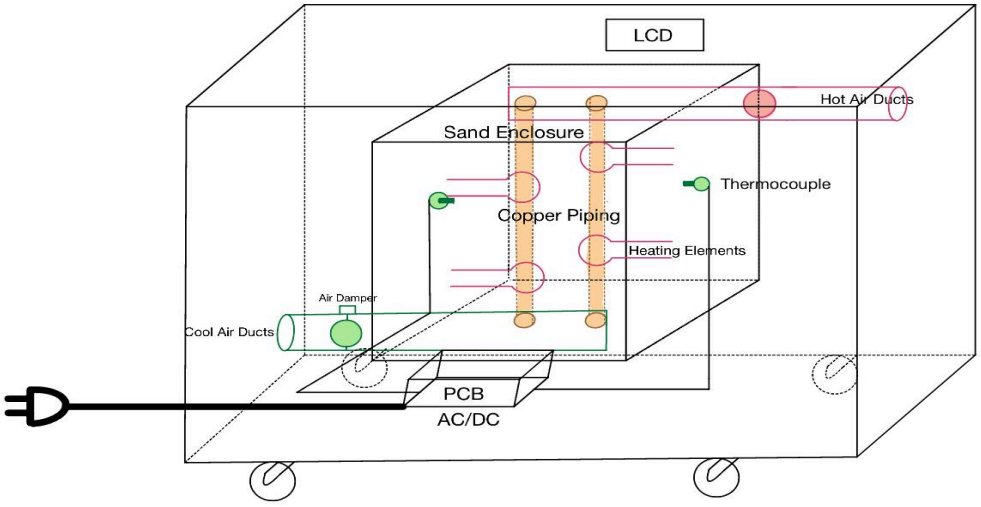



*Figure 6.6: Frame/Sand Enclosure*

# Chapter 7:Software Design

## 7.1 Flowchart

The flowchart diagram illustrates the logical steps and decision points in a system's main loop, which is intended to manage user interactions, temperature monitoring, and charging/heating processes. The main loop keeps the system running smoothly and safely by continuously checking for user input, reading temperatures, updating the display, and maginging charging/heating operations. It is also includes overheat protection to prevent potential hazards. In the figure 7.1, shows the ilustration of the project idea of the user's interaction, and here it is the explanation of each step:

**Start:** Where the system begins executing the main look, after system's initialization.

**User input: Check for interaction:** Checking if the user interacted with the system via LCD.

**Update Settings if input detected:** If user input is detected, system needs to update the settings accordingly. After the setting are updated, the system will go to the next step.

**Read External Temperature:** System reads the current external temperature from a sensor, records the temp for display.

**Read Internal Temperature:** System reads the current internal temperature from a sensor, records the temp to check if falls within the desired range for safe operation.

**Update Display:** The system updates the LCD to display real-time external and internal temperatures.

**Check Charging Time:** Check if time to start charging. If the scheduled charging time is reached, the system will start charging, and if it is not time to charge, the system will check heating time.

**Start Charging:** The system will start charging.

**Monitor Internal Temperature while Charging:** Keep tracking the internal temperature while charge for safety reasons.

**Stop Charging if temp/time limit reached:** Check if it is time to stop charging, if temperature or time limit is reached, it will stop.

**Check Heating Time:** Check if time to start heating. If the scheduled heating time is reached, the system will start heating, and if it is not time to heat, the system will check for overheat protection.

**Start Heating:** The system will start heating.

**Monitor Internal Temperature while Charging:** Keep tracking the internal and external temperature while heating for safety reasons.

**Stop Heating if temp/time limit reached:** Check if it is time to stop heating, if temperature or time limit is reached, it will stop.

**Overheat Protection: Shut down if overheating:** The system will keep monitoring for overheat condition, if overheat detected the system will shut down, otherwise, it will return to check user input.

**End:** The system completes the current interaction of the main loop, and transitions back to checking user input to repeat the loop.

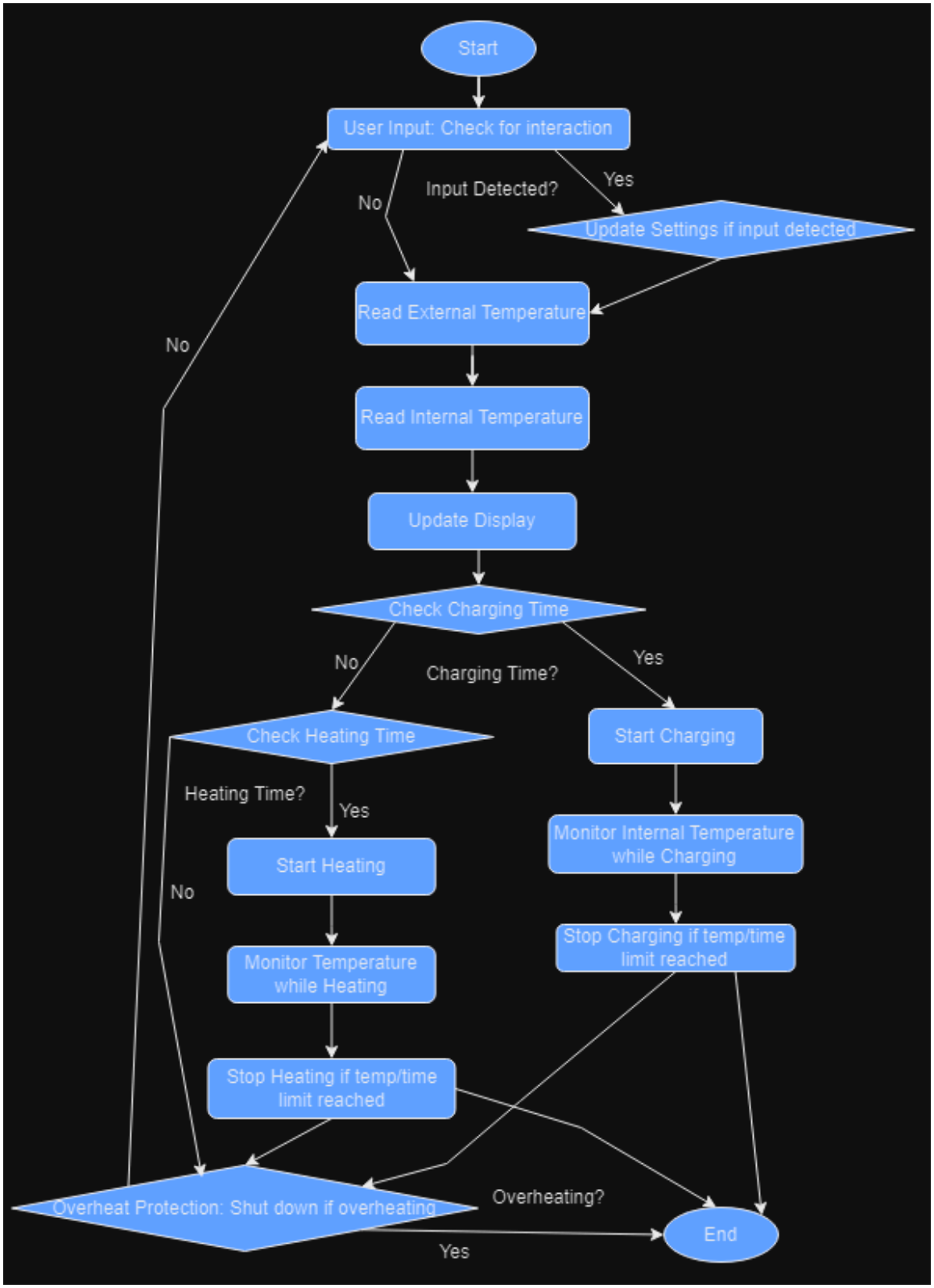



*Figure 7.1: Flowchart*

## 7.2 Case Diagram

The use case diagram is a significant visual representation of how users interact with the system, illustrating the various functionalities and processes involved. The idea of this type of diagram is to provide understanding the actions that a user take and how the system responds to those actions. It serves as a blueprint for the system's behavior, outlining the roles of the user and the system in key operations such as user input validation, temperature reading, display updating, charging/heating process, and overheat protection. In the figure 7.2, shows the representation of use cases and actors helps the successful interaction and design of the system's requirements and functionalities.

### Use Cases:

1. **User Input**
   Actor: User
   Description: User interacts with the system via the LCD to update settings.

2. **Check User Input**
   Actor: System
   Description: The system checks if there is an update in the settings by the user.

3. **Read External Temperature**
   Actor: System
   Description: The system reads the external temperature from a sensor.

4. **Read Internal Temperature**
   Actor: System
   Description: The system reads the internal temperature from a sensor.

5. **Update Display**
   Actor: System
   Description: The system updates the display with the latest temperature.

6. **Start Charging**
   Actor: System
   Description: The system initiates the charging process based on the user-defined schedule.

7. **Monitor Charging**
   Actor: System
   Description: The system monitors the internal temperature during the charging process.

8. **Stop Charging**
   Actor: System
   Description: The system stops the charging process when the desired temperature or time limit is reached.

9. **Heating Charging**
   Actor: System
   Description: The system initiates the heating process based on the user-defined schedule.

10. **Monitor Heating**
    Actor: System
    Description: The system monitors the internal and external temperature during the heating process.

11. **Stop Heating**
    Actor: System
    Description: The system stops the heating process when the desired temperature or time limit is reached.

12. **Overheat Protection**
    Actor: System
    Description: The system suits down to prevent damage if overheating is detected.

## Actors:

1. **User:** Interacts with the system to update settings and schedules.

2. **System:** Perform various operations such as reading temperatures, updating the display, starting and stopping charging/heating processes, and providing overheat protection.

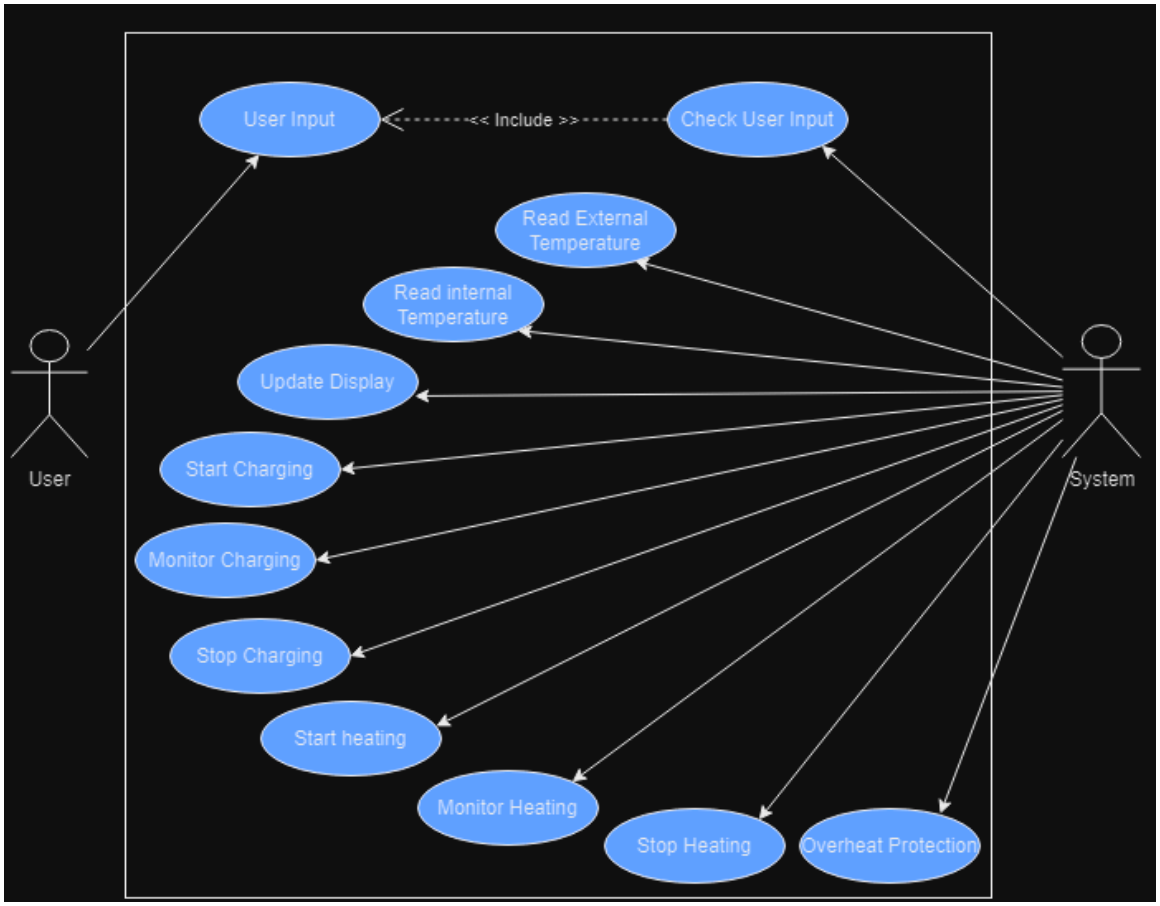



*Figure 7.2: Case Diagram*

## 7.3 State Diagram

The state diagram illustrates various states and transitions in the system. It shows how the system transitions betwee states, such as idle, temperature reading, display updating, charging, heating, and overheat protection, in response to specific triggers and conditions. The state diagram helps us understand the systems' workflow by mapping out these states and transitions, ensuring that each component operates properly and efficiently. This detailed visualization is critical for both systems design and throubleshooting because it shows how the sytesm should respond to various events and conditions.

In the figure 7.3, shows the visual representation of the states and transitions for the system. Here is the explanation of the states and its transitions.

**States:**

Idle: The system waits for user input or scheduled events.
Reading Temperature: The system reads temperature data from sensors.
Updating Display: The system updates the LCD display with the latest information.

Charging: The system is charging, monitoring the internal temperature.
Heating: The system is heating, monitoring both internal and external temperatures.
Overheat Protection: The system detects overheating and shuts down to prevent damage.

**Transitions:**

From idle to Reading Temperature: Triggered by a timer event.
From Reading Temperature to Updating Display: Triggered when temperature data is read.
From Updating Display to idle: Triggered when display is updated.
From idle to Charging: Triggered by a scheduled time or user initiation.
From idle to Heating: Triggered by a scheduled time or user initiation.
From Charging to idle: Triggered when charging completes or user stops it.
From Heating to idle: Triggered when heating completes or user stops it.
From Charging to Overheat Protection: Triggered by overheat detection.
From Heating to Overheat Protection: Triggered by overhart detection.
From Overheart Protection to idle: Triggered when the system cools down.

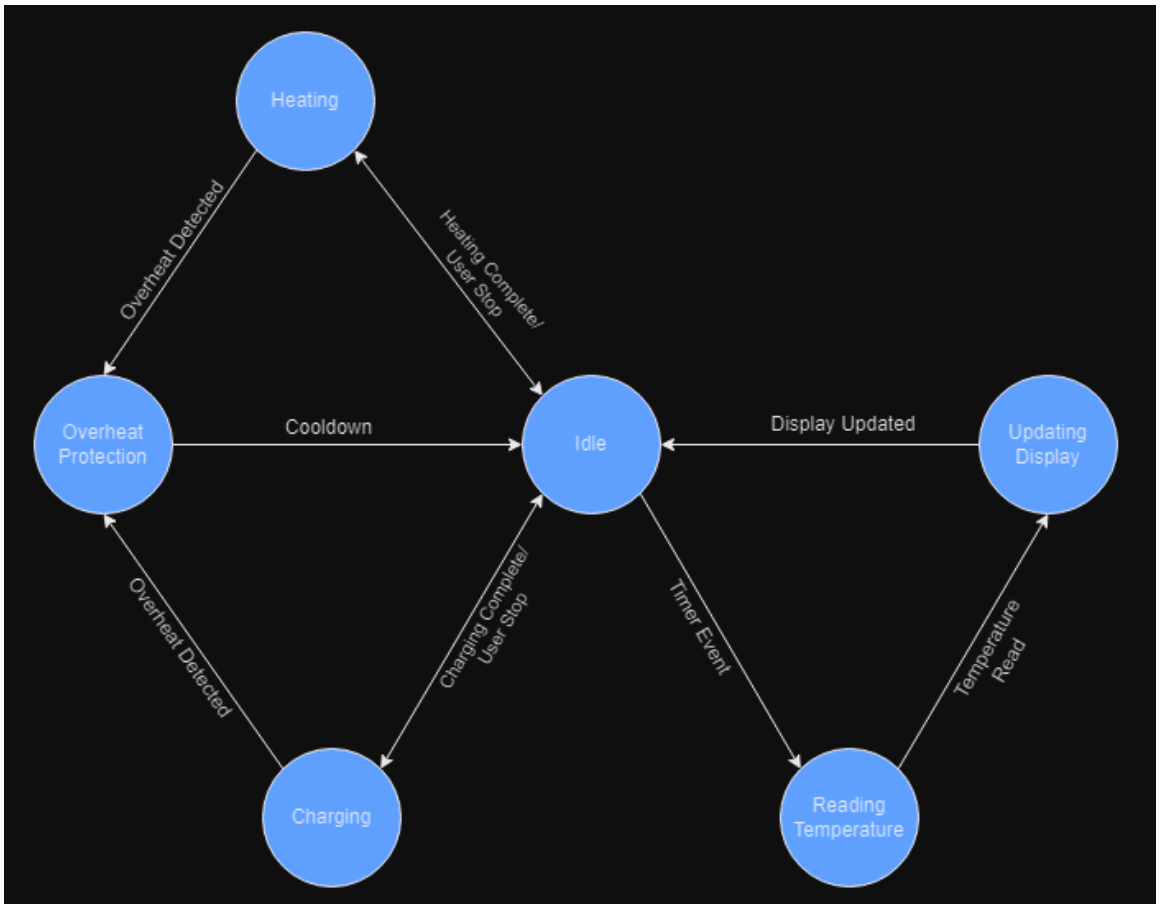



*Figure 7.3: State Diagram*

## 7.4 Class Diagram

The class diagram depicts the various classes, their attributes, methods, and the relationships between them, giving a detailed view of the system's architecture. This diagram demonstrates how many components interact with another to achieve the desired functionality. It is mostly used in object-oriented programming and system modeling because it provides a picture of the system's architecture. In the figure 7.4, shows how the systemControl, inputControl, tempSensor, displayContro', chargeControl, heatControl, and overheatControl classes integrate in order to manage user inputs, monitor temperatures, update displays, and ensure safe charging and heating processes.

**Classes:**

1. **systemControl:** Manages the overall system.
2. **inputControl:** Manages the user interaction with the system.
3. **tempSensor:** Reads and store temperature data.
4. **displayControl:** Updates the screen with the current information.
5. **chargeControl:** Manages the charging process.
6. **heatControl:** Manages the heating process.
7. **overheatControl:** Ensures safety by monitory temperatures.

**Attributes and Methods:**

1. **systemControl**
   Attributes:
   userInput: inputControl - Manages user input settings.
   externalTemp: tempSensor - Reads the external temperature.
   internalTemp: tempSensor - Reads the internal temperature.
   displayStats: displayControl - Updates and manages display information.
   chargeManager: chargeControl - Manages the charging process.
   heatManager: heatControl - Manages the heating process.
   overheatManager: overheatControl - Monitors and manager overheat.

   Methods:
   mainLoop() - Runs the main operational loop of the system.
   updateSystem() - Updates the system state based on input and sensors.

2. **inputControl**
   Attributes:
   userSettings: settingsMap - Store user-defined settings

   Methods:

checkUserInput() - Checks for and processes user inputs.
updateSettings(settings: settingMap) - Updates the user settings.

3. **tempSensor**
   Attributes:
   temperature: float - Represents the current temperature reading.

   Methods:
   readTemp() - Reads the current temperature.
   getTemp() - Returns the current temperature value.

4. **displayControl**
   Attributes:
   currentDisplay: str - Holds the current display content.

   Methods:
   displayUpdate(externalTemperature:float,internalTemperature:float, status:str) - Updates the display with the latest temperatures and system status.

5. **chargeControl**
   Attributes:
   chargeStats: str - Indicates the current status of the charging process.

   Methods:
   startCharge() - Initiates the charging process.
   monitorCharge() - Monitors the charging process.
   stopCharge() - Stop the charging process.

6. **heatControl**
   Attributes:
   heatStats: str - Indicates the current status of the heating process.

   Methods:
   startHeat() - Initiates the heating process.
   monitorHeat() - Monitors the heating process.
   stopHeat() - Stop the heating process.

7. **overheatControl**
   Attributes:
   isOverheating: bool - Indicates whether the system is overheating.

   Methods:
   checkOverheat() - Checks for overheating conditions.
   shutDownSystem() - Shuts down the system to prevent overheating.

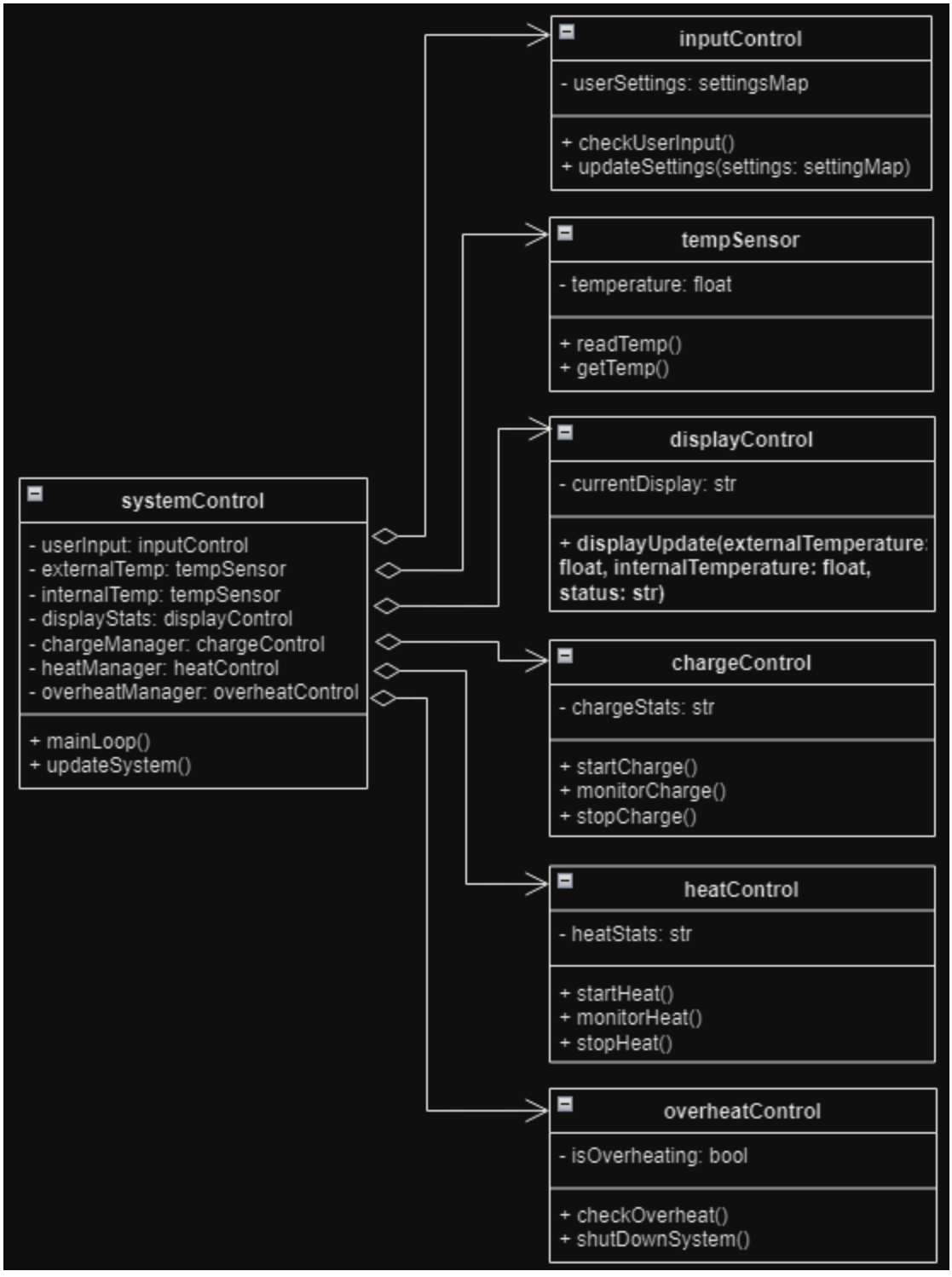



*Figure 7.4: Class Diagram*

## 7.5 User Interface

The user interface (UI) of a system is critical in allowing effective interaction between the user and the system. It functions as a bridge, converting user commands into system actions and returning system status to user. Creating a well-designed user interface allows users to easily access and control the system's features and functions, which improves usability and satisfaction with the product. This system's user interface is intend to be intuitive and informative, providing real-time updates on critical parameters suchs as external and internal temperatures, system status, and user-programable settings. The interface provides simple navigation options, allowing users to seamlessly switch between monitoring system performance and adjusting settings to meet their specific requirements.
In the figure 7.5.1, shows the system status display, and here it is the explanation of the interface elements:

- **External Temperature(Blue Circle):** Display the ambient temperature obtained from the sensor.

- **Internal Temperature(Red Circle):** By touching the red circle it will allow to display either the internal temperature or the percentage of charged battery.

- **System status(Yellow, Green, Red light):** Provides real-time of status of charging, heating, and overheat condition.
    - **Yellow light:** It is charging if the light is on and not charging if the light is off.
    - **Green light:** It is heating if the light is on and not heating if the light is off.
    - **Red light:** If the light is on, It is overheating; if the light is turned off, it is not. The system will have a range limit before shuting off on its own, and if the user sees the red light, he must manually shut it off.

- **Settings button:** Allows the user to navigate to the user settings(figure 7.4) to configure system parameters.

- **Start/Stop Charging:** Allows the user to manually control the charging process.

- **Start/Stop Heating:** Allows the user to manually control the heating process.

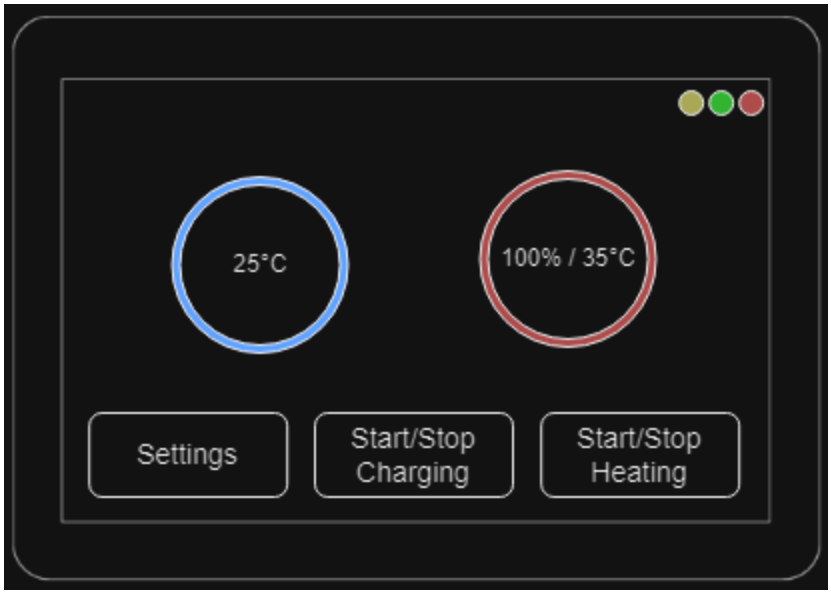



*Figure 7.5.1: System Status Display*

In the figure 7.5.2, shows the system's user settings display, and here it is the explanation of the interface elements:

- **Charging Time:** Field to set the timer at which the charging process should start and stop.

- **Heating Time:** Field to set the timer at which the heating process should start and stop.

- **Ambient Temperature:** Field for user to set the maximum ambient temperature.

- **Temperature Scale:** Field for user to select if he would like to see the temperature in celsius or Fahrenheit.

- **Save button:** Saves the user-configured settings.

- **Up/Down button:** Used to help setting the charging/heating time, select the maximum ambient temperature, and select the temperature scale.

- **Back button:** Returns to the system status display(figure 7.3) screen without saving the changes.

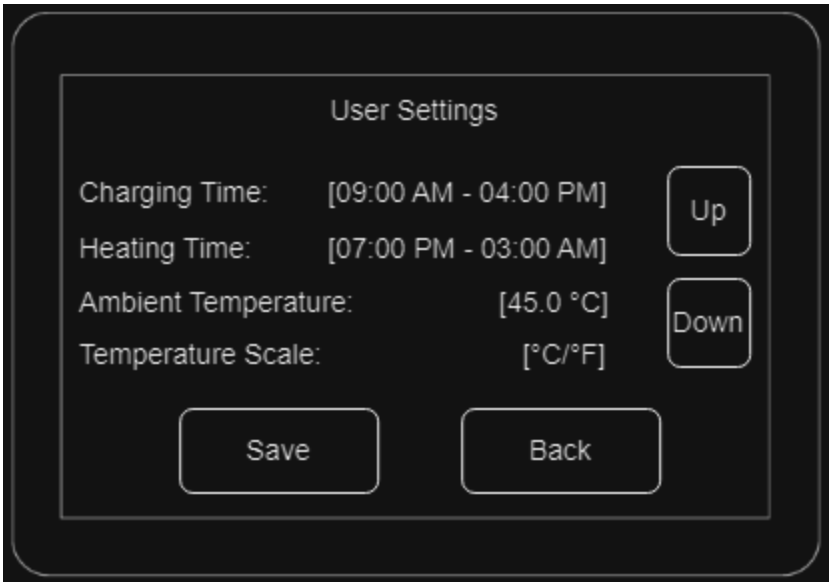



*Figure 7.5.2: User Settings Display*

## 7.6 Data Transfer Diagram

The data transfer diagram represents the flow of information through the system, including interactions between components. Starting with the user interface, the diagram indicates how the input control processes user inputs before updating the system control. The system control component manages temperature sensor data, controls charging and heating processes, and monitors overheat protection. The display control updates the user interface to reflect the current system status, ensuring real-time feedback and efficient system operation. In the figure 7.6, shows the data pathways and below it is the respective data transfer process in the system.

1. **User Input:**
    - Data: User settings and commands.
    - Source: User Interface (User input)
    - Destination: InputControl

2. **Settings Update:**
    - Data: Update settings.
    - Source: inputControl (Update the settings)
    - Destination: systemControl

3. **Temperature Readings:**
    - Data: External and internal temperatures.
    - Source: tempSensors External and Internal (Reads the temperatures)
    - Destination: systemControl

4. **Charging Control:**
    - Data: Charging commands.
    - Source: systemControl  (Triggers the charge)
    - Destination: chargeControl
    - Data: Status updates, stop signals.
    - Source: chargeControl (Stops the charge)
    - Destination: systemControl

5. **Heating Control:**
    - Data: Heating commands.
    - Source: systemControl (Triggers the heat)
    - Destination: heatControl
    - Data: Status updates, stop signals.
    - Source: heatControl (Stops the heat)
    - Destination: systemControl

6. **Overheat Protection:**
    - Data: Overheat Status.
    - Source: overheatControl (Triggers the overheat protection)
    - Destination: systemControl
    - Data: Commands for managing overheating.
    - Source: systemControl (Monitors the overheat)
    - Destination: overheatControl

7. **Display Update:**
    - Data: Current status and temperatures.
    - Source: systemControl (Sends data to display)
    - Destination: displayControl

8. **User interface update:**
    - Data: Updated display information.
    - Source: displayControl (Updates the display)
    - Destination: User Interface

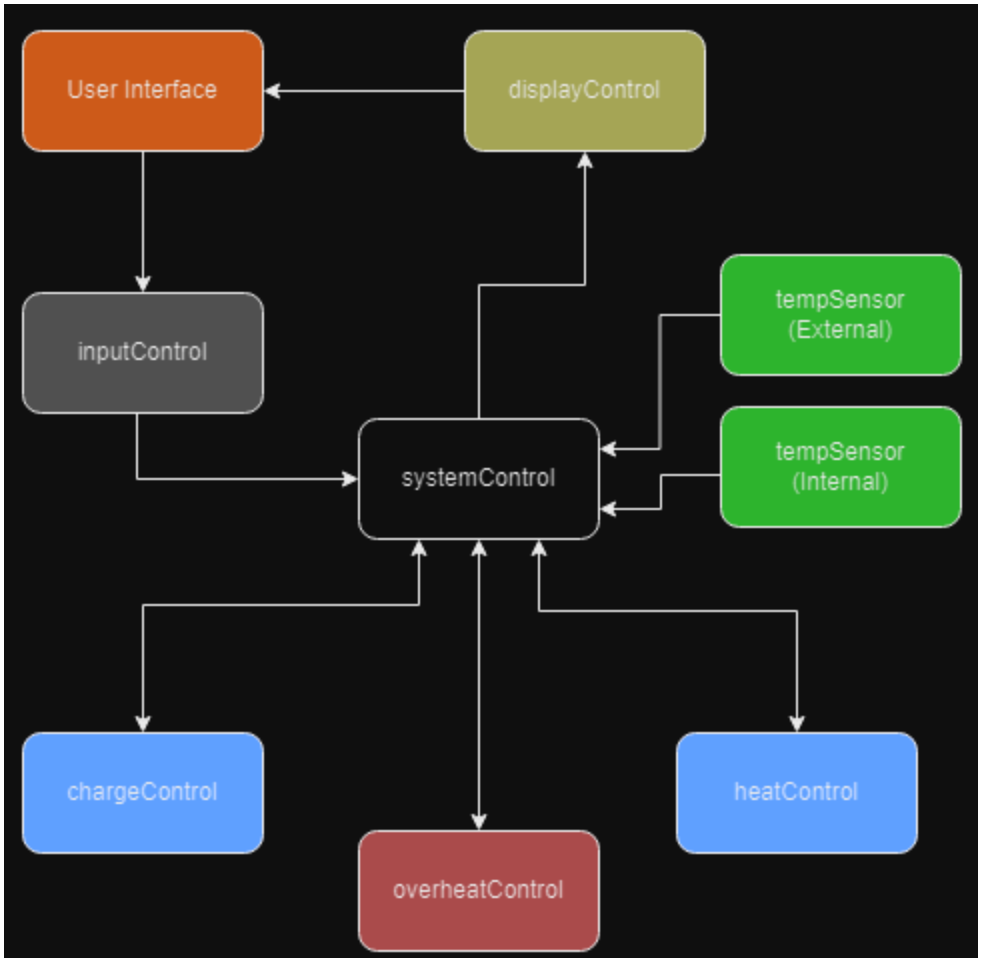



*Figure 7.6: Data Transfer Diagram*

# Chapter 8:System Fabrication/Prototype Construction

## 8.1 Main PCB

Once the schematics were captured, we then proceeded to begin the PCB layout. We first had to take into consideration what would be the best layout for EMC based on the standards established in Chapter 4. To do this, we first decided to separate our PCB into different zones. This is a good practice as it helps us visually see where each component needs to go based on the functionality of it. This not only helps shorten trace lines, but helps us identify each section on the board and separates components with high noise from those sensitive to higher noises. Our two noisiest zones will be the digital zone and the power zone. The connectors must be isolated from this noise as we do not want to accidentally inject it into our communication lines. Below is a diagram of our established zones.

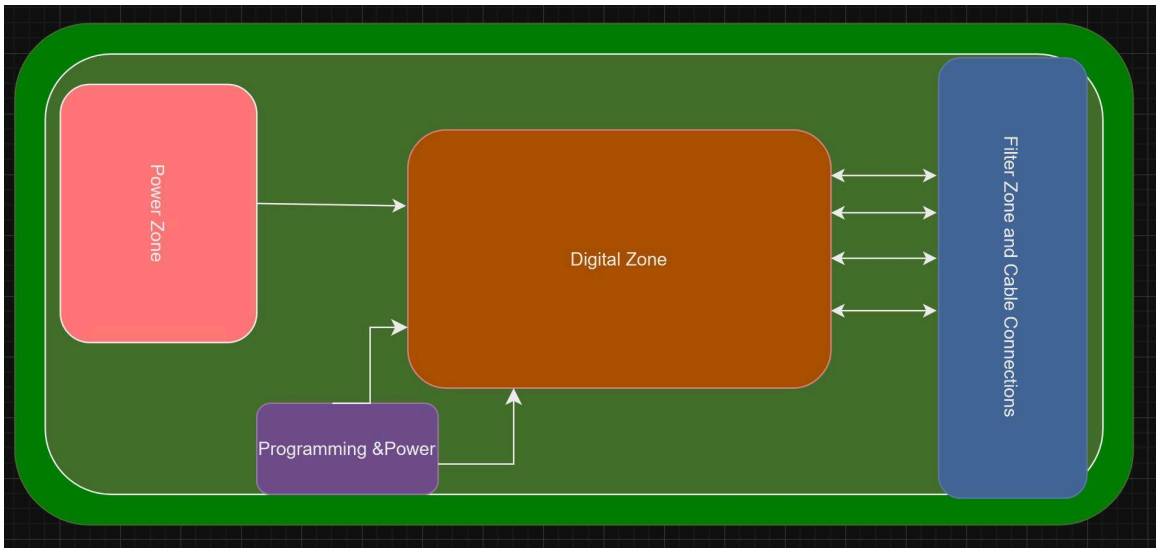



*Figure 8.1: Segregation Zones for Main PCB*

## 8.2 Custom AC/DC Converter

We created each of our PCB designs using Fusion 360 software. We carefully considered all via dimensions/trace widths to provide enough surface area for up to 1A at roughly 10 mils for the traces. The AC/DC converter we designed (figure 8.1) provides 3.3VDC from the voltage regulator for our components which need 3.3V such as the ESP32 which we used for our CPU. This board required us to mount the transformer to the chassis of the housing as it risked compromising the structural integrity of the board due to its weight.

We created 4 holes, one at each corner, to allow for the AC/DC converter to be mounted to the chassis or the main PCB in the design via screws. The design includes via stitching to improve the heat dissipation for the board as well as for improving the circuit's grounding. Another reason we used via stitching was to improve the EMI shielding of the board without the addition of necessary components. The design is common in AC/DC converters, having four diodes create the full wave rectifier with a capacitor to smooth the voltage ripple and a voltage regulator to create the steady 3.3V needed. We left room on the AC/DC converter's board to utilize a heat sink on the voltage regulator, ensuring the board will not overheat when powered on. In our design we created pins for the transformer and the voltage regulator output to connect these devices easily once the PCB arrived.

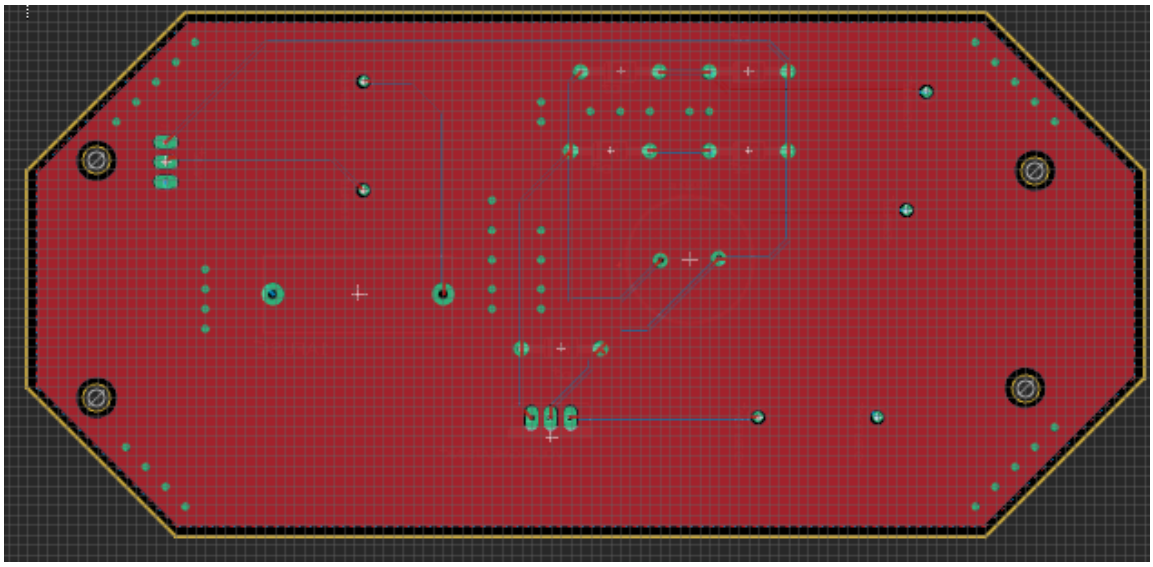


*Figure 8.2: AC/DC Converter PCB v1*

## 8.3 Sand Enclosure and Frame

For the sand enclosure, we utilized a stainless steel box with dimensions of 12 inch x 12 inch x 12 inch. Inside this, we placed our sand, heating elements, thermocouple, and copper ventilation tubes running through the center of the steel cube. This design allowed for ventilation through the center of the sand enclosure with a metal surface to mount our heating elements and thermocouples onto.

Our design includes HVAC ducts of 3 inch diameter for airflow through the device. The air flows into one "exhaust" vent, through the ductwork and airflow fans used to circulate the air within the machine, through the copper piping of the same enclosure, and out through more duct work. On either side of the sand enclosure in the HVAC duct work we implemented electronic HVAC dampers which allows the device to start or stop the air circulation in the pipes, allowing for

the device to retain its heat while in charging mode or to dispel the heat when heating the room. We wrapped the enclosure and the duct work with the fiber ceramic insulation blanket. This would ensure the heat from the sand enclosure and the ventilation would not overheat the areas of the design surrounding the enclosure such as the frame, support mounts, PCB/circuitry, and other components which would deteriorate at high temperatures. With each heating element requiring 2 holes to mount, 2 holes for either side of the copper tubes, and two holes for the thermocouple, we chose stainless steel due to its strength under high temperatures to increase the lifespan of the design. Below is our design's overall schematic.

# Chapter 9:System Testing and Evaluation

## 9.1 Hardware Design and Testing

The ESP32-WROOM development board has many conveniences for final developers although there are many tests that we will have to implement to ensure that the systems we're designing will be resilient to all possible inputs. The ESP32 will have an automated LED light to indicate that it is receiving power and operational. On initial set up will be able to test each needed GPIO pin with a standard multimeter/voltmeter device. The development board also has two buttons that can be tested as well, for example the reset pin will noticeably rest any code that is currently running and displaying information on the Serial Monitor within the Arduino IDE, this would be the most ideal form of checking whether the reset pin works or not.

## 9.2 DHT11 Thermo couple

The project will have roughly 2 DHT11 temperature modules, the temperature modules can be tested by using the GPIO pins that are currently built into the ESP32 dev board. We are able to power the DHT11 temperature modules using the 3.3v output pin from the ESP32 dev board and read the data from it using the third pin on the module using the ESP32 GPIO pin. We are able to interface with both the DHT11 modules and read their current readings to have them printed on the serial monitor within the arduino IDE that we are using to program the ESP32 development board. We are able to corroborate the temperature module readings by comparing with a thermometer we purchased from the store. We are also able to test polling frequency with the arduino ide as well as. With the thermocouples we can also test having different variables for the reading to ensure the accuracy and just how much deviation/how sensitive the thermometer can be. This sort of testing will also indicate just how detailed we can be with our readings in the instance where we use float values or we use integer values. For environmental testing we can start by taking an indoor room temperature reading, with this will be able to compare both the readings from both DHT11 temperature modules so we can calculate the average deviation between both modules. We will take the thermometers to a colder environment so we can see how long the DHT11 takes to accurately read the colder environment as well as the average deviation between both the sensors. We will also be able to see just how capable the ESP32 power pins are with the DHT11 pins, we want to ensure that the ESP32 power pin is capable of powering multiple thermocouples and being able to manage the sudden power demand if multiple thermocouples were to turn on at the same time/overlapping times. With this testing we would also be able to indicate the kind of compatibility we can expect from the libraries we use, upon the testing we've seen that there are outdated libraries that may pertain to the DHT11 and with this testing we were able to find the most modern version of

the library giving us the dht11.h library which seems to have built on the original dht.h library.

## 9.3 K Type High Temperature Thermocouple

The K type temperature sensor can follow a similar testing approach to that of the DHT11 only that the extremes will be a bit more difficult to test. We would want to start with a simple basic connectivity test where we have the K type temperature thermocouple connected to the SPI interface board. When we connect the K type high temperature thermocouple we can then connect it to a thermocouple amplifier so it can interface with the ESP32 development board.With this testing we will be able to identify the best library for interfacing with the thermocouple. The thermocouple will also need to be tested in different environments, being tested in a typical indoor environment and compare the readings with another thermometer, being tested in a cold environment such as a refrigerator or an ice bath to see if the temperature falls accordingly, as well as a hot environment where the sensor can be placed in hot water or something similar. In terms of testing the extremes of this thermocouple we will also want to make sure that it can withstand the extreme heat of the sand battery, because of this we will put the thermocouple within the sand and try to get it up to the maximum temperature we have set within out ESP32. When we have the thermocouple in the super heated sand we can verify its accuracy by measuring the sand temperature with an IR thermometer and then ensuring that the thermocouple would also still be able to accurately read the temperatures within the room temperature environment, the cold environment, and the warm environment. To ensure product safety we wanted to make sure that the thermocouple was accurate and calibrated if needed. With this we compared the readings of the k type thermocouple with the DHT11's and a store bought thermometer. The K Type High Temperature Thermocouple was rated to withstand temperatures of up to 1000C and we made sure that we could reach 500C while also having accurate temperature measurements. Another important factor given the range of temperatures this thermocouple is going to experience was testing the sensor response time, how long does this sensor hold onto the head/cold that it experiences and how does it affect its readings in changing environments. The thermal sand battery will range between 100C and 500C this thermocouple will have to show how quick it reports new temperatures as we the internal temperature fluctuates. When setting up the thermocouple we did also ensure to have the correct polarity for the connections so as to ensure proper connections and readings from the thermocouple. One of the biggest considerations we had when testing this thermocouple was ensuring that the thermocouple would get the power that it needs to give correct readings, with insufficient power input the thermocouple would deliver incorrect power readings that would be dangerous to the overall system as these thermocouples would be responsible for indicating whether the MCU should cut off power to the battery. With our testing and corroboration of the data sheet we see that the supply current of the overall thermocouple system typically draws 0.7 mA of current

while drawing a maximum 1.5 mA of current, this 1.5mA seeming like its done when a reading is needed. Because 1.5mA was such a small amount its not a cause for concern for our overall system as the ESP32 can comfortably deliver that power. We did also want to ensure that when testing that the K type thermocouple wasn't subject to any kind of signal noise/interference during testing as well as what different kinds of errors noise/interference would have the greatest effect on the readings from the actual temperature modules. With the temperature module fully tested we then want to ensure that the thermocouple we used is up to the task as well.

## 9.4 MAX6675 SPI Module

The MAX6675 is capable of performing cold-junction compensation and is able to digitize the signal from a type-K thermocouple. When the type-K thermocouple is being read by the ESP32 we are able to read it as a digital 12-bit resolution output with an SPI compatible read-only format, this simplifies the pinout of the actual MAX6675 module as we only had to account for an SCK pin, CS pin, and the SO (Master In Slave Out pin), this is notably missing the master in pin which is not necessary as we are dealing with a read only SPI device, so with this we start a basic connectivity test, we used the max6675.h library to initialize the connection between the ESP32 and then proceed to create a loop function which can loop and get new readings from the K-type thermocouple we're using. the using the max6675.h library we had a somewhat similar approach to the DHT11.h library only with the MAX6675 we had to initialize the connections between the MISO pins, the chip select pin, and the SCK, this being different to that of the DHT11 which just needs to initialize the GPIO pin. We would also want to test the thermocouple with the interface that we have connected by doing the same room temperature readings in a typical room, cold environment testing, and then the hot environment testing. We would want to ensure that we can test the SPI communication between our custom PCB and the SPI module. With this we were able to use a little oscilloscope (or logic analyzer) to verify the SPI signals during data transmission and ensure that all the data is transmitted to the ESP32 and processed correctly.

## 9.5 CG Solid State Relay

The solid state relay would involve needing to test the electrical and mechanical functionality to ensure our system is able to switch correctly between states and handles the input from the ESP32 as well as the managing the AC component of our system. The CG solid state relay has a 3-32VDC input and an output 24-480VAC output with a maximum current of 25A, this would indicate that we can theoretically control a power throughput of 12kW. We checked the inherent coil resistance as well as the overall insulation resistance to check if we need to make any special enclosures to protect surrounding components or any users that enters the hardware enclosure, but with an insulation resistance of 1000M Ohm/500VDC as well as the protective plastic cap for the terminals, we would

just need to protect users and other components from the slightly exposed terminals. We then wanted to test this relay DC input terminals as most arduino compatible relays operate wth a V+, GPIO pin, and GND pin. However this relay only had a V+ and ground terminal on the DC side. With this we would test if the activation coil toggles correctly. We ended up connecting the coil to GPIO19 which is capable of outputting a 3.3v ‘HIGH’ value, because this 3.3v GPIO pin output is over the 3v minimum threshold of the relay we wanted to make sure that the toggle time was still low and that we would be able to toggle it at all. After we had set up the whole core we wrote a code that would just toggle the output state of GPIO pin 19 every second so that we could see the relay switching on and off and we saw that the relay did actually end up toggling along with the GPIO pin indicating that the ESP32 GPIO pins were more than capable of driving the DC control pins of the DC-AC relay. Next we wanted to test the AC side of the relay, we took little precaution and connected a cable to the AC side that would be able to connect to a wall port. When we has connected the AC side fo the relay we needed to make sure that we put the relay on the hot wire of the AC connection, this way we would be able to have the sort of switching effect, for safety we also tested connecting the relay on the ground wire of the AC cable and found that the relay was able to respond quickly to the GPIO output on the DC side of the connection.

During testing of the AC side of the relay we have the terminals connected to the heating elements in a testing sand pit to ensure that we had a load on the AC wires. When we connected everything we also have a kilowatt reader between the AC connections and the wall port, this was capable of telling us if the plug was drawing power or not and we were able to determine that the relay was accurately toggling the AC side of the relay and that there were no noticeable delays nor heating up of the relay despite driving 1kW of power. For safety testing we also wanted to see what would happen if the system lost power, we wanted to make sure that if the thermal sand battery was disconnected from wall power that it would still be able to fall to a resting state that would be safe to have in the home. When the ESP32 development board lost power all the GPIO pins naturally fell to a lot state causing the relay to toggle to an off resting state thus cutting off the AC power. This sort of safety check is important because if the relay were to stay in its current activated setting when the ESP32 loses power then we would have a situation where the sand battery never stops charging, in the best case scenario if this were to happen where the AC connection remained active would be that the heating elements burn out, but in the worst case scenario we would be in a situation where if the container got too hot and the thermal insulation started failing or wires started heating up to a degree that exponentially degrades their capacity we would be in the risky situation where we can start a fire, and the last thing we need our thermal battery to do is compromise the structural integrity of the battery and start dumping super heated sand onto the floor of the end user.

## 9.6 MP15YA 6” & MP21YA 8” Heating Elements

When we were designing the heating section of the sand battery we wanted to ensure that we had an accurate measurement of the resistance provided by the resistive heating element to corroborate the rated power output of the two differently sized heating elements. We recorded the larger heating element using a standard voltmeter at a flat 32 ohms while the smaller ‘less powerful’ heating element was 40 ohms. The 32 and 40 ohm resistance values tracked with the rated power output that was provided by the supplier indicating that the 6” coil was rated at 1500 watts at 230v and 2100 watts for the 8” coil at 230v as well. Although we didn’t have a method of plugging in these coils to a 230/240v source we were able to connect it to a standard 120v AC wall plug and got a returned power output value of roughly half the rated power output of the rated 230v rating. Although we wanted to maximize the power output we did also need to consider some of the constraints provided from our report meetings as well as standard electrical practices within the US. Most generic US home room circuits are rated at 15A 110v/120v, this means that we don’t want to pull more than 1650-1800 watts from a single circuit, on top of this most power outlets are only rated for a power draw of 10 amps at the 120v rating, thankfully when testing our system we had wired our heating elements so as to ensure a power draw below but close to 1000 watts. The reason we want to limit our power draw to 1000 watts, aside from not wanting to heat up a power receptacle to a degree which risks damaging/burning dry wall, is because often times users will have multiple components connected to the same power circuit, living rooms will have TV’s and sound systems connected so we want to ensure that the user can still enjoy their typical daily peripherals while also being able to adequately charge and utilize their battery. The supplier indicated that the heating elements take 30 seconds to reach a temperature of 250C, although we weren’t able to get an accurate measurement at the time we were able to see that even at half the power output we were able to evaporate water off it within a minute of active power, and even when testing it while they were submerged in the sand the area within a 3ft radius of the sand battery had already become several degrees warmer than the rest of the room. We were also able to start testing the internal temperatures with the K-type temperature module being able to gather the average time it takes the heating elements to warm our demo bucket of sand several degrees, this would give us to corroborate the calculated charge time given the specific heat of the sand, assumed perfect insulation of the environment, and the calculated power output of the heating elements we will be using within the sand battery. We wanted to test the power cycling abilities/reaction from the heating elements as we will be having a system that frequently toggles the heating elements on and off, to do this we connected the ESP32 development board to toggle the DC input connection on the relay, we would toggle the relay every 2 seconds, and with this we found that the heating elements had an ideal amount of power output and had no form of delay to start drawing power. Another consideration we had was changing resistance given different thermocouple temperatures, we wanted to make sure that power draw would remain consistent as the heating element

reached higher temperatures, the reason we wanted to check this was to ensure that there was no form of reduced resistance at higher temperatures and that as a result there's no risk of heating element degradation through constant heating/cooling of the system. We started with the room temperature sand battery and measured the power draw from the wall which we measured as having a consistent 120V supply. When the sand battery had been heating for an hour the sand temperature in the small demo bucket seemed to stabilize at about 300ºC given that the battery was reaching an equilibrium of radiating just as much heat as it was being given from the heating elements. In the process of heating the power draw had only deviated by less than 10 watts, which on a power draw of 1100 watts after we had finalized the parallel/series connections of the heating elements, this represented a 0.90% standard deviation of internal resistance. We had then double checked this information by disconnecting the relays and AC power and measuring the final resistance once the battery had reached its equilibrium temperature, with this measurement we were able to indicate that the final resistance value of the heating elements had remained within the same <1% deviation from the initial rating manufacturer/cool temperature resistance readings. The simplicity of this system heating design, in using an array of heating elements, gave us the opportunity to explore different approaches to wiring the overall heating system while also being able to control the system with multiple relays. Needing more power we would be able to active relays that wire the system in such a way that draws more power and creates more heat, and then being able to switch which relays are active and then creating a sort of connection that is giving us a different heating element connection format which changes power draw and heat output. The benefit of this approach is that multiple relays can be connected to one GPIO pin meaning that we can have entire heating element set ups operating entirely off a handful of GPIO connections which are driving the very ‘native off’ relays we discussed earlier in the chapter. Later on if we are trying to push the boundaries of this heating element we will likely explore options to heat the sand past the initial 500C we had discussed earlier as the thermocouples and insulation is reportedly able to withstand temperatures up to 1025C, this would likely include a process where we use sacrificial heating elements that we heat to the point where we do see degradation or even a sort of catastrophic failure so as to have a better idea of just where the limits of our system can be.

## 9.7 ESP32-WROOM-32 MCU

To test the ESP32-WROOM-32 MCU we wanted to verify most of the functionality that we had read within the datasheets as this thermal dune sand battery project will be maximizing the amount of data this MCU can handle as well as maximizing the amount of connections it has. For development environment testing of the ESP32-WROOM-32 MCU we utilized the Arduino IDE which we installed along with all the necessary libraries we would need to drive the very components we had talked about earlier. When connecting the ESP32 to the computer we wanted to ensure that the development board would receive the

power that it needs to be able to operate correctly as well as establish a COM5 data connection with the IDE, with the Arduino IDE we were able to test the built in LED by initializing the pin 2 as an output pin and toggling it with the simple looping function. This assured us that we were able to set pin 2 as an output pin. We then moved on and wanted to test the GPIO pins as well, to test the digital I/O connections we connected an LED and a simple resistor to one of the GPIO pins (in our case we used GPIO pin 2) where we were able to set the pin GPIO output to toggle every 1000 milliseconds, we would use the digitalWrite() functions to set the pin to a high and low value, this would toggle the LED, to save time we wanted to test some of the other GPIO pins we knew we would be using (for example to toggle the relays) to all toggle together, sending a high then low signal and looping. We actually ended up going pin by pin and reading the voltage outputs and confirming that the ESP32 would output a 3.3v high signal when it's activated. We also tested the GPIO pins by running a pulse width modulation test, with a PWM to output an analog signal using digital means, with this we were able to test the speed and responsiveness of the GPIO pins that we had been using. To run the PWM test we had initialized GPIO4 and created a ledcSetup() function call to set Channel to 0, 5kHz frequency and an 8-bit resolution, we then ran some for loops that would increment the duty cycles using the ledcWrite() function, and then another for loop to then decrease the brightness. Because we have many thermocouple components, some of which will have to utilize the internal ADC converter, we also wanted to test the analog input ability of the ESP32, for this test we connected a potentiometer to the analog input in and wrote a short script to get a reading from it. When setting up the code we set GPIO pin 34 to be the pin that would connect to the output pin of the potentiometer, the Vcc and ground were connected to the respective 3.3v power supply pin on the ESP32 and the ground pin of the ESP32. We started the potentiometer test by initializing the Serial connection at a 115200 baud rate and would use the built in 'analogRead()' function that the arduino IDE/ESP32 offers us, we would read this as a plain integer value and print it to the Serial Monitor, and with this approach we would be able to see the integer values changing according to the output value of the potentiometer and its correlation to how we twist the dial. The continuation of the peripheral interface tests would next be with the I2C device, the I2C test is usually done with an OLED display or sensor so we had used the ESP32 to connect to some thermocouple peripherals using the I2C protocol and we were able to get an accurate reading from the temperature sensors that was corroborated with the use of reading the room temperature with a 3rd party store bought thermometer. We were then able to test the SPI communication protocol with the k-type thermocouples and their dedicated SPI conversion SPI PCB boards which was also able to provide a good measurement. Lastly we wanted to run a power consumption test on the ESP 32, with this connection test we ran the code that we had been working on for the overall system and powered it with a USB cable that was connected to a standard wall plug that we connected to the AC wall outlet power monitor. What we found was a vindication of our decision to utilize a powerful yet low power draw MCU as the amount drawn by the MCU and any connected peripherals was

less than 250 mA of power, although this number is given to change as the project continues, develops, and possibly expands already having most of the foundational components connected to the ESP32 and seeing that its power draw can be considered negligible is a reading that we gives us lots of space for expansion.

## 9.8 Software Testing

Software testing is an important step in the software development lifecycle because it ensures that the developed system meets the requirements and functions properly. The goal of software testing is to find and fix bugs, verify performance, and ensure that the system works as intended. In order to create a strong testing framework for the system, we will need to go use different types of software testing, and methodologies.

### 9.8.1 - Types of Software Testing

- **Unit testing:** Consists of testing individual components or modules to ensure they perform as expected.

- **Integration testing:** Focuses on testing the interaction of integrated modules to ensure that all components work properly.

- **System testing:** Consists in checking the entire and integrated software application to ensure that it meets the requirements.

- **Regression testing:** Consists in checking the existing software functionality to ensure that it works as expected after changes, as well as detecting new bugs in existing functionality.

- **Performance testing:** Consists in checking the system's performance under load to identify bottlenecks. Load testing, stress testing, and scalability testing are all types of performance that we can be used on our system to check how it perform.

### 9.8.2 - Methodologies

- **Manual testing:** Consists in creating manual test cases, which provides flexibility and intuition. However, it could be time consuming and easy to have human error.

- **Automated testing:** Consists in using tools to execute test cases, which results in increased speed, repeatability, and minimize human error.

For our specific system, we will begin by using the unit testing individual components like inputControl, tempSensor, displayControl, chargeControl,

heatControl, and overheatControl, to ensure all methods like checkUserInput(), readTemp(), displayUpdate(), startCharge(), startHeat(), and checkOverheat() are working properly. Next step, we will conduct integration testing to verify the interactions between systemControl and the sub components to ensure proper data flow and command execution between systemControl, inputControl, tempSensor, displayControl, chargeControl, heatControl, and overheatControl. Then we will execute system testing, which simulates real-world scenarios and user interactions to ensure that all components work as intended.

After all these tests, we will run performance testing to evaluate the system's performance under different loads to find bottlenecks and optmize the performance of the system. Therefore, following this test guideline we will be able to develops a full software testing strategy that ensures the system's reliability, performance, and accuracy.

### 9.8.3 FreeRTOS Multithreading

When planning the overall system design for the ESP32 logic and code we had been running into a few issues that were arising from trying to create a more linear iterative approach to the overall program. Having this amount of external sensors, an LCD screen with a touch component, fan/valve control, and toggling several relays we needed a new approach that would allow for independent polling and manipulation of variables which would also allow for a simplification of the code by essentially sectioning off elements of the code. The benefit to the FreeRTOS multithreading library is that we will be able to create threads that would be responsible for each section/specific external component. The reason this approach would be best as opposed to an iterative/linear coding approach is that errors in any part of the code can be more sectioned off and errors would essentially be sectioned off to the exclusive thread thats running as opposed to collapsing the entire program from just one error, as a result error handling when working on the code would be far simpler as we would also be able to ensure that a failure/error in any component can be correctly handled and even mitigated within the final product. When setting up the overall code we would start by integrating the necessary libraries such as Arduino.h for the arduino ide and the FreeRTOS.h library as well. We started testing the code by ensuring that we were able to set up global variables that would be accessible by all of the threads, these variables would be responsible for storing things such as temperature readings from all of the thermo couples, relay status' (which would also pertain to the activation status of the heating elements) , valve and fan conditional states, as well as any form of LCD touch interfacing which would need to be accessed by multiple threads. When testing this all we also wanted to ensure that we had the proper mutual exclusion locks on sensitive information that is stored in variables to prevent any possible race condition that could happen, for example the thermometer integers storing the values of each of the thermocouples would need to have a mutual exclusion approach so that the room temperature management thread would read a 'sure value' from the

thermometer variable which would give us the ability to be confident in saying that we avoid race conditions where the room would be heated to a degree higher than the user had set, or that the room is left colder than the user had originally set.

Mutual exclusion as a safety protocol for internal heat management would be the most important safety protocol to have for the overall system as we will be dealing with the super heated sand. During testing we had set a maximum temperature for the internal sand at 500C which we are confidence can be achieved by the given heating elements, but in the situation where the variables responsible for storing the read temperatures of the internal thermocouples is being updated and it then gets read by another thread that is responsible for the managing the internal charge of the battery, the mutual exclusion would ensure that we aren't reading a garbage value just because its being updated which would then likely cause the battery to overcharge and thus risk the structural integrity of the battery and the fire safety of the home the battery is currently stored in. Because there will be so many shared resources, like in the earlier example where the system needs to securely measure the internal temperature, we would also have other threads that- for example- communicate with the LCD to display an active room temperature state or charge state of the battery. This sort of information being provided to the user would be very important because we would have to ensure that the user is seeing the correct information and that the battery is meeting the use requirements that had been set forth by the user as well. For testing mutual exclusion locks we had initialized a 'SemaphoreHandle_t' variable within the IDE with a 'sharedResource' variable which is to be interacted with by the different threads to ensure that the shared resources are accessed one at a time. We would define the task functions to use the mutexes by instantiating the threads with an infinite while loop that also has an initial if condition that checks the semaphore condition and if possible locks the variable in the case where it does have to read/write to it. In our testing we created tasks that access shared resources which aren't currently reading anything too important, in many cases the threads were just incrementing a variable but with every test we were able to confidently say that the variables were only being accessed by one thread at a time. We verified this mutex functionality by monitoring the serial output to ensure the tasks accessed the shared resources in a controlled manner by having serial.println() lines within the threads that we had set up. In the serial output we had each thread pring whether it was accessing a given shared resource and we were able to find that initial implementation was a success. Stress testing the MUTEX involved us introducing additional tasks to increase demand of a shared resource which only solidified out confidence in the MUTEX implementation we had put in place. The next step to our testing was to create a more advanced approach by being able to create tasks with different priorities to test the overall impact on the mutex acquisition. This is important because higher priority tasks should be able to superseded lower priority tasks which would have to wait for the more important thread to finish. An example of a 'higher priority' task, or 'hard real time system'

within our code would be the thread that is responsible for managing the internal temperature of the thermal sand battery. This is a critical component of the overall system as dealing with super heated sand is something that doesn't give us room for error, if the battery is actively heating and the variable responsible for storing the internal battery temperature is being hogged by the LCD thread or any other thread we risk overshooting the internal temperature as the internal temperature management thread would not be able to toggle the charging status of the battery because it may not have the most up to date value for the internal temperature. So on our testing code we were able to create tasks with different priority levels which will be taken into consideration when accessing the mutex, and at the moment the highest priority threads will be a value of 3 for the thread responsible for reading and updating the internal temperature value variable, and a value of 2 for the charge status management thread which will be responsible for reading the system clock/time and then being able to toggle whether the relays responsible for managing the charging of the battery are in an active or deactivated state, and everything else is currently at a priority value of 1 as none of the other threads have any sort of importance as it pertains to reading the internal temperature value, for example the LCD thread which reads the internal battery temperature is not a hard real time system that would have catastrophic effects on the project if it were to take longer to get a reading, thankfully the LCD doesn’t have to have the most up to date temperature reading a all times. Moving to a second example of a priority list for multiple threads would be having the LCD thread gain a higher priority for reading the system/local clock as well as the ambient temperature by reading the ambient temperature variable to then display it on the LCD, but even in this situation its not as important as the discharging status variable which is responsible for keeping track of recorded discharge times and ensuring that the ambient temperature falls within the desired range by the user by toggling the fans and valves which are their own variables and threads in this situation.

With the testing of these threads we did also try introducing simulated delays, these simulated delays would simulate long and short access times to the shared resources. This would allow us to monitor task execution and mutex handling to ensure proper program synchronization and ensure that there are no delay restrictions/errors we may have from our threads.

The testing of potential deadlocks is also a very important situation to consider. Deadlocks are a common issue in multithreaded systems. When we are dealing with multiple temperature sensors and and LCD display, as well as many other elements we would need to ensure that each thread is correctly locking a resource and unlocking a resource as long as its not needed. The last thing we would want is to cause two important variables to be locked because two threads locked their variables and indefinitely waiting for the other resource to become available only for it to wait infinitely or until the entire thermal battery system collapses. In a system with multiple thermometers where each sensor has its own variable the tasks would likely need to share common resources such as the

I2C buses, SPI interfaces, or logging mechanisms. To best approach this situation we would want to ensure that we can avoid any of these situation by avoiding nested locks where tasks need to lock multiple mutexes simultaneously, use timeouts where a task can fail/retry gracefully as opposed to sort of freezing the entire system, having a mechanism/thread that can detect the deadlocks and thus resolves it- although this approach might be a bit more complicated as we need to track the state of mutexes and tasks-, and lastly we would use a lock hierarchy which establishes a strict order in which the locks are to be acquired, the lock hierarchy would essentially build on the priority lock approach that individual threads can utilize but instead applying it to the MUTEX level.

By implementing these strategies we can prevent deadlocks from occuring in our multithreaded application as well as preventing race conditions from damaging the integrity of the overall system.

# Chapter 10:Administrative Content

## 10.1 Budget and Financing

The budget for this project will be limited to USD 800 and is set by our group members. This amount will help the group to stay true to one of our goals, which is to make this device affordable should it ever be mass-produced. We have allotted larger portions of the budget toward the transformer, thermistor sensors, microcontrollers, and the frame of the device.

The measure of the goal for the project will be with the intent of a per-kWh of storage benefit. The main benefit of this project being the value proposition, a raw lithium consumer battery averaging at about $300 per kWh of energy storage is what we seek to beat, the goal of this project will be to achieve a per kWh storage price several orders of magnitude lower than a basic lithium battery which our system would even have the benefit of being ‘the whole package’ whereas the lithium batteries would still need costly inverters and heating elements.

| Systems | Budget |
|---|---|
| Frame/Sand Storage | $250 |
| Microcontrollers/Sensors | $150 |
| AC-DC Converter/Heater | $200 |
| Valves/Fans/Aesthetics | $200 |
| TOTAL | $800 |

*Table 10.1.1: Initial Budget and Startup Costs*

## 10.2 Project Milestones

To ensure the timely completion of this project, we have established extensive milestones to track the team's progress and meet specified deadlines. Team formation and initial brainstorming for the project took place during Senior Design I (Documentation Phase), where we will research the feasibility of the project to confirm its viability. Most of the planning, research, and initial designs will occur over the summer, to have a working prototype ready before Senior Design II (Validation Phase). In the next phase, we will conduct extensive testing and validation before releasing the final design for inspection. Below is the tentative schedule for both Senior Design I and II.

| Task | Start Date | End Date | Description |
|---|---|---|---|
| Team Formation/Brainstorm | 5/13/24 | 5/17/24 | Teams will be formed, and the project is initially agreed upon. |
| Individual Research/ Feasibility determination | 5/17/24 | 5/22/24 | Roles and responsibilities are assigned to team members. Each team member breaks off and investigates the feasibility of their respective system. |
| Initial Divide and Conquer Report (Chapter 2 and Chapter 10) | 5/22/24 | 5/31/24 | Initial research, costs, specifications, and project descriptions are gathered and presented for approval. |
| 30 Page Milestone (Chapter 1 and Chapter 3) | 5/31/24 | 6/14/24 | A formal executive summary of the project and extensive engineering research is conducted, building upon initial research. |
| 60 Page Report Milestone (Chapter 4 and Chapter 5) | 6/14/24 | 6/28/24 | Using gathered research, design constraints are constructed for each system of this project. Standards are formally written and implemented. |

| | | | |
|---|---|---|---|
| 90 Page Milestone (Chapter 6 and Chapter 7) | 6/14/24 | 7/5/24 | System schematics are refined utilizing block diagrams. The architecture of each system will be thoroughly evaluated. The software team will implement the user interface architecture and communication protocols. |
| 120 Page Milestone (Chapter 8 and Chapter 9) | 7/5/24 | 7/23/24 | PCB layout is finished utilizing ECAD (Fusion 360). System test procedures (STP) are created. Performance parameters are evaluated based on established standards. |

*Table 10.2.1: Project Report Milestones SD1*

| Task | Start Date | End Date | Description |
|---|---|---|---|
| Schematic Design | 6/10/24 | 6/26/24 | Begin the initial schematic design that will be used to designate individual component functions on the PC. |
| Begin building GUI Software Interface | 6/10/24 | 6/26/24 | The software team will begin coding the GUI. |
| Parts Order | 5/31/24 | 6/21/24 | Order the necessary parts required for SMT and breadboard prototyping. |

| | | | |
|---|---|---|---|
| Begin PCB Layout Design | 6/15/24 | 7/12/24 | Once the schematic is complete, translate it into a PCB layout. Focusing on EMC and Layers considerations. |
| Breadboard/Dev board Prototyping | 6/21/24 | 7/12/24 | Prototyping the designs using a breadboard and a dev board to catch potential points of failure and redesigns. |
| Initial System Testing and Function Validation | 7/12/24 | 7/19/24 | Systems are initially tested and validated to meet engineering specifications and standards. |
| Finish PCB Layout Design | 7/19/24 | 7/23/24 | The PCB layout is completed and Ready for fabrication. |

*Table 10.2.2: Project Design Milestones SD1*

| Task | Start Date | End Date | Description |
|---|---|---|---|
| PCB manufacturing and fabrication | 7/23/24 | 8/6/24 | The PCB will be fabricated and SMT components will be placed. |
| PCB validation and testing | 8/6/24 | 9/06/24 | PCB is rigorously put through the same validation and testing from SD1 to ensure functionality. |
| GUI testing and validation | 8/6/24 | 9/06/24 | GUI is tested to ensure user-friendliness and no bugs are missed. |
| System Integration and Testing | 9/06/24 | 10/07/24 | All systems are assembled, connected, and tested to ensure specifications are met. |

| | | | |
|---|---|---|---|
| Practice Demo Initiated | 10/07/24 | 10/28/24 | The finished battery system is demoed. Ensuring all goals are met and objectives are completed. |
| Refine Final Documentations | 10/28/24 | 11/22/24 | All documentation is edited and refined to account for last-minute findings. |
| Final Presentation | TBD | TBD | The finished product is presented to the review committee for approval. |

*Table 10.2.3: Project Design Milestones SD2*

## 10.3 Workload Distribution

| Student | Student Major | Responsibility |
|---|---|---|
| Michael Hernandez | Electrical Engineering | PCB design/layout |
| | | PCB-Sensor communication protocols |
| Miguel Baca-Urteaga | Computer Engineering | MCU-Hardware communication |
| | | MCU/embedded system programming |
| | | Hardware Implementation |
| Filipe Pestana Frances | Computer Engineering | LCD-MCU communication |
| | | MCU user interface programming |
| Tanner Cyr | Electrical Engineering | AC-DC converter design |

| | | |
|---|---|---|
| | | Thermocouple and Hardware implementation |
| | | PCB-Hardware considerations |

*Table 10.3.1: Workload Distribution Table*

# Chapter 11:Conclusion

In conclusion, integrating the thermal sand battery into a versatile coffee table/space heater provides more cost effective heating while providing an aesthetically appealing piece of furniture. With the ability to plug into any common 15A residential circuit and its portability, the space heater allows for localized heat for the consumer in place of a bulky furnace. With its installation time of less than a minute, this device can quickly provide heat to the room it is in with a very user friendly interface much like that of a typical HVAC system. Combined with the ESP32 board's advanced capabilities for monitoring and control, the sand battery offers a sustainable and cost effective way to maintain comfortable indoor temperatures. This combination not only optimizes energy consumption but also allows for smart home integration and precise temperature regulation. As a result, the design streamlines the realm of home heating systems for a fraction of the cost compared to whole-home HVAC systems or furnaces. As this device's heat is localized to its respective living space/room, this system would supply more cost efficient heating due to its ability to target the heat toward occupied rooms versus the entire home.

The innovative design would allow builders in the northern part of the United States to utilize space much more efficiently. Without the need for ductwork from a furnace or HVAC system, it allows for better use of the architectural design of newly manufactured homes as HVAC ductwork can be very bulky in design. With minimal to zero maintenance required for the thermal sand battery, it is much more cost effective than traditional HVAC systems which should be replaced every 15 to 20 years.

The ESP32 is particularly well suited for temperature monitoring and management due to several key features and capabilities, first of which is the ability to connect via its multiple I/O ports. The ESP32 has a wide range of input/output ports, enabling it to interface with various temperature sensors and other peripherals. This flexibility allows for the integration of multiple sensors to monitor the PCB temperature and sand enclosure's temperature. It offers very low power consumption compared to other boards with various sleep modes that can be applied. The ESP32 max current draw is about 250mA which allows for the onboard heating elements of the thermal sand battery to utilize more power from the circuit it is connected to. It is equipped with a dual-core processor and can handle complex tasks and algorithms for temperature management such as data logging, analysis, and real-time system applications. The ESP32 is very cost effective and allows for the manufacturers (our design team) to customize their product with ease.

Our custom PCB and AC/DC converter allow for the ESP32 to receive adequate power from household circuits and were designed to reduce the amount of maintenance hours spent in the system's design. The PCB was designed to hold all major computing components such as the ESP32 along with the connections

to/from peripherals to the PCB. Our AC/DC converter provides a steady voltage of 3.3V for the ESP32 and peripherals connected, along with simple pin connections for the chassis mounted transformer and screw mounts to connect the AC/DC board to the system. This converter provides safety and isolation between the input and output voltages and minimizes power loss during the conversion process. This compact design allows for the user to easily and quickly heat their home as well as utilize power during hours of the day when electricity is less expensive.

# References


[1] J. Ellsmoor, "Forbes," 10 December 2021. [Online]. Available: https://www.forbes.com/sites/jamesellsmoor/2019/06/15/renewable-energy-is-now-the-cheapest-option-even-without-subsidies/?sh=7114dd575a6b.

[2] I. Hubschmann, 28 August 2020. [Online]. Available: https://www.nabto.com/guide-to-iot-esp-32/.

[3] Luke, 7 September 2023. [Online]. Available: https://riverdi.com/blog/discovering-the-magic-of-3-5-inch-displays-size-and-appli.

[4] Energy Information Administration, "Energy Information Administration," [Online]. Available: https://www.eia.gov/energyexplained/use-of-energy/electricity-use-in-homes.php.

[5] Energy Information Administration, "Energy Information Administration," [Online]. Available: https://www.eia.gov/energyexplained/use-of-energy/homes.php.

[6] C. Liang, "Instructables," 25 February 2021. [Online]. Available: https://www.instructables.com/Select-Color-Display-for-ESP32/.

[7] S. Santos, "Random Nerd Tutorials," 5 December 2019. [Online]. Available: https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/.

[8] S. Santos, "Random Nerd Tutorials," 15 March 2022. [Online]. Available: https://randomnerdtutorials.com/esp32-k-type-thermocouple-max6675/.

[9] SourceForge, "Altium designer vs. Autodesk Fusion 360 vs. KiCad Eda Comparison Chart," [Online]. Available: https://sourceforge.net/software/compare/Altium-Designer-vs-Fusion-360-vs-KiCad-EDA/. [Accessed 23 June 2024].

[10] Software Radius, "KiCad vs altium designer - which one is best? in [2024]," [Online]. Available: https://www.softwareradius.com/kicad-vs-altium-designer/. [Accessed 23 June 2024].

[11] StackShare, "Arduino IDE vs IntelliJ IDEA VS Visual studio: What are the differences?," [Online]. Available: https://stackshare.io/stackups/arduino-ide-vs-intellij-idea-vs-visual-studio. [Accessed 25 June 2024].

[12] Digital Shack, "Replacing the arduino ide with VS code & platformio - digital shack," [Online]. Available: https://www.digitalshack.org/replacing-the-arduino-ide-with-vs-code-platformio/. [Accessed 25 June 2024].

[13] G2, "Arduino Ide vs. Visual Studio 2022 vs IntelliJ IDEA | G2," [Online]. Available: https://www.g2.com/compare/arduino-ide-vs-visual-studio-vs-intellij-idea. [Accessed 25 June 2024].

[14] Wikimedia Foundation, "ESP32," [Online]. Available: https://en.wikipedia.org/wiki/ESP32#ESP32-S3. [Accessed 20 June 2024].

[15] Texas Instruments, "MSP430FR6989 data sheet, product information and support | ti.com," [Online]. Available: https://www.ti.com/product/MSP430FR6989. [Accessed 20 June 2024].

[16] D. Reynolds, "RP2040 vs. ESP32: How to choose the right microcontrollers," 25 January 2024. [Online]. Available: https://www.xecor.com/blog/rp2040-vs-esp32. [Accessed 20 June 2024].

[17] Waveshare, "4.3inch QLED display, DSI interface, 800 × 480, integrated structure, toughened glass panel," [Online]. Available: https://www.waveshare.com/4.3inch-dsi-qled.htm. [Accessed 27 June 2024].

[18] JavatPoint, "C vs C++ VS Python vs Java - Javatpoint," [Online]. Available: https://www.javatpoint.com/c-vs-cpp-vs-python-vs-java. [Accessed 1 July 2024].

[19] GeeksforGeeks, "Comparing python with C and C++," 29 June 2022. [Online]. Available: https://www.geeksforgeeks.org/comparing-python-with-c-and-c/. [Accessed 1 July 2024].

[20] Espressif Systems, "ESP32 technical reference manual," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. [Accessed 2 July 2024].

[21] Espressif Systems, "ESP Hardware Design Guidelines - ESP32 - - ESP Hardware Design Guidelines latest documentation," [Online]. Available: https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32/index.html. [Accessed 2 July 2024].

[22] Espressif Systems, "Schematic Checklist - ESP32 - - ESP Hardware Design Guidelines latest documentation," [Online]. Available: https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32/schematic-checklist.html. [Accessed 2 July 2024].

[23] LCDwiki, "MSP3525&MSP3526 3.5inch IPS TFT SPI Display Module Specification," [Online]. Available: http://www.lcdwiki.com/res/MSP3525_MSP3526/MSP3525_MSP3526_Specification_EN_V1.0.pdf. [Accessed 3 July 2024].

[24] LCDwiki, "MSP3525&MSP3526 3.5inch IPS TFT SPI Display Module Specification," [Online]. Available: http://www.lcdwiki.com/res/MSP3525_MSP3526/3.5inch_SPI_Module_MSP3525_MSP3526_User_Manual_EN.pdf. [Accessed 3 July 2024].

[25] Maxim Integrated, "Max6675 cold-junction-compensated K-thermocouple," [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max6675.pdf. [Accessed 2 July 2024].

[26] DigiKey, "SEN0497 DFRobot | Development Boards, Kits, Programmers | DigiKey," [Online]. Available: https://www.digikey.com/en/products/detail/dfrobot/SEN0497/15283080. [Accessed 30 June 2024].

[27] "Simond Store Ceramic Fiber Insulation Board - 2300F - 1" x 18" x 24" - Ceramic Thermal Insulation Board for Wood Stoves Fireplaces Furnaces Forges Kilns Boilers Pizza Oven Insulation - Pack of 1 : Tools & Home Improvement," [Online]. Available: https://www.amazon.com/Ceramic-Insulation-Individual-Stoves-Furnaces/dp/B07H1HSXD3. [Accessed 28 June 2024].

[28] "UniTherm Polypropylene Fabric Insulation Supports 600-in - Low Thermal Conductivity & Excellent Heat Insulation," [Online]. Available: https://www.lowes.com/pd/UniTherm-Ceramic-Bulk-Fiber-5lb-Box/5013070129. [Accessed 28 June 2024].

[29] "smart water valve ss304 full bore BSP NPT dn15 dn8 dn10 cr05 5wire 5v 12v 5s cwx15n motorized water valve," [Online]. Available: https://www.alibaba.com/product-detail/smart-water-valve-ss304-full-bore_60812190270.html. [Accessed 29 June 2024].

[30] "UDQFB3E61 30mm Fan 5V 0.07A Double Ball Aluminum Frame high Temperature Resistant Cooling Fan," [Online]. Available: https://www.amazon.com/UDQFB3E61-Aluminum-Temperature-Resistant-Cooling/dp/B09TW18JMD. [Accessed 1 July 2024].

[31] "Tubing Stainless Steel 1" Thick Wall," [Online]. Available: https://www.sailrite.com/Tubing-Thick-Wall-Stainless-Steel-1-1-Foot. [Accessed 3 July 2024].

[32] JLCPCB, "PCB Manufacturing & Assembly Capabilities," [Online]. Available: https://jlcpcb.com/capabilities/pcb-capabilities. [Accessed 30 June 2024].

[33] NextPCB, "IPC-2221: The Standard for Printed Circuit Board Design," [Online]. Available: https://www.nextpcb.com/blog/ipc-2221. [Accessed 30 June 2024].

[34] NextPCB, "IPC 6012 Guidelines: Classifying, Designing, and Fabricating PCBs," [Online]. Available: https://www.nextpcb.com/blog/ipc-6012. [Accessed 30 June 2024].

[35] IPC, "IPC-2221B Generic Standard on Printed Board Design," [Online]. Available: https://www.ipc.org/TOC/IPC-2221B.pdf. [Accessed 1 July 2024].

[36] NextPCB, "IPC-A-600: The Standard for Printed Circuit Board Inspection," [Online]. Available: https://www.nextpcb.com/blog/ipc-a-600. [Accessed 1 July 2024].

[37] IPC, "IPC-A-610D - sumnic," [Online]. Available: https://dl.sumnic.com/download/IPC-A-610D_-_Acceptability_of_Electronic_Assemblies_-Proposed.pdf. [Accessed 2 July 2024].

[38] NextPCB, "IPC-A-610: The Standard for Acceptability of Electronic Assemblies," [Online]. Available: https://www.nextpcb.com/blog/ipc-a-610. [Accessed 2 July 2024].

[39] Sierra Circuits, "IPC J-STD-001 Standard for Soldering," 8 April 2024. [Online]. Available: https://www.protoexpress.com/blog/ipc-j-std-001-standard-soldering-requirements/. [Accessed 3 July 2024].

[40] R. Hu, PCB Design and Layout Fundamentals for EMC, Independently, 2019.

[41] NIST, "Understanding Flow Meters," 6 April 2021. [Online]. Available: https://www.nist.gov/programs-projects/understanding-flow-meters. [Accessed 5 July 2024].

[42] NIST, "Material Evaluations," 17 August 2020. [Online]. Available: https://www.nist.gov/mml/csd/organic/material-evaluations. [Accessed 5 July 2024].

[43] NIST, "Fire Dynamics," 2Fire Dynamics July 2024. [Online]. Available: https://www.nist.gov/el/fire-research-division-73300/firegov-fire-service/fire-dynamics. [Accessed 5 July 2024].

[44] NIST, "Fire Protection," 26 June 2018. [Online]. Available: https://www.nist.gov/el/fire-research-division-73300/firegov-fire-service/fire-protection. [Accessed 5 July 2024].

[45] NIST, "Thermocouples Calibrations Services," 16 November 2019. [Online]. Available: https://www.nist.gov/pml/sensor-science/thermodynamic-metrology/thermocouples-calibrations-services. [Accessed 6 July 2024].

[46] NIST, "AC-DC DIfference," 26 February 2024. [Online]. Available: https://www.nist.gov/programs-projects/ac-dc-difference. [Accessed 6 July 2024].

[47] NIST, "Smoke Alarm Research," 2 June 2021. [Online]. Available: https://www.nist.gov/el/smoke-alarm-research. [Accessed 6 July 2024].

[48] Winsen Electronics, "Winsen ZP13 Smoke Detection Module," [Online]. Available: https://shop.winsen-sensor.com/products/zp13-smoke-detection-module?variant=42756689100992. [Accessed 7 July 2024].

[49] Digikey, "RE46C180S16F Microchip Technology | Integrated ... - Digikey," [Online]. Available: https://www.digikey.com/en/products/detail/microchip-technology/RE46C180S16F/2775880. [Accessed 7 July 2024].

[50] "L-Com," [Online]. Available: https://www.l-com.com/smoke-sensor-moldulemq2-sraq-g004. [Accessed 7 July 2024].

[51] "First Alert SA303CN3 Basic Battery Operated Smoke Alarm," [Online]. Available: https://www.firstalertstore.com/store/products/sa303cn3-basic-smoke-alarm.htm. [Accessed 8 July 2024].

[52] "AliExpress," [Online]. Available: https://www.aliexpress.us/item/3256802436687394.html. [Accessed 29 June 2024].

[53] "AliExpress," [Online]. Available: https://www.aliexpress.us/item/2251832840428651.html. [Accessed 8 July 2024].

[54] Digikey, "F1238S05B-FSR Mechatronics Fan Group | Fans, Thermal Management | DigiKey," [Online]. Available: https://www.digikey.com/en/products/detail/mechatronics-fan-group/F1238S05B-FSR/8119949. [Accessed 1 July 2024].

[55] Lcdwiki, "320RGB x 480 dot 262K Color with Frame Memory Single-Chip TFT Controller/Driver," [Online]. Available: http://www.lcdwiki.com/res/MSP3525_MSP3526/ST7796S-Sitronix.pdf. [Accessed 3 July 2024].

[56] Lcdwiki, "3.5inch IPS SPI Module ST7796 - LCD wiki," 19 July 2023. [Online]. Available: http://www.lcdwiki.com/3.5inch_IPS_SPI_Module_ST7796. [Accessed 27 June 2024].

[57] Garnet Abrasive RockRidge, "Garnet Abrasive - RockRidge," [Online]. Available: https://shop.surfaceprep.com/garnet-abrasive-rockridge.html. [Accessed 9 July 2024].

[58] Lee Supply, "1" X 20' COPPER PIPE M HARD," [Online]. Available: https://www.gotolee.com/buy/product/1-x/3970. [Accessed 3 July 2024].

[59] A. Industries, "Adafruit AHT20 - Temperature & Humidity Sensor Breakout Board," [Online]. Available: https://www.adafruit.com/product/4566. [Accessed 30 June 2024].

[60] A. Industries, "3.5" TFT 320X480 + touchscreen breakout board w/microsd socket," [Online]. Available: https://www.adafruit.com/product/2050. [Accessed 27 June 2024].

[61] A. Industries, "Adafruit MCP9808 High Accuracy I2C Temperature Sensor Breakout," [Online]. Available: https://www.adafruit.com/product/5027. [Accessed 30 June 2024].

[62] A. Industries, "Adafruit PT100 RTD temperature sensor amplifier - MAX31865," adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/3328. [Accessed 2 July 2024].

[63] A. Industries, "Adafruit TMP235 - Plug-and-Play STEMMA Analog Temperature Sensor," adafruit industries blog RSS, [Online]. Available: https://www.adafruit.com/product/4686. [Accessed 30 June 2024].

[64] A. Industries, "Adafruit Universal Thermocouple Amplifier MAX31856 Breakout," adafruit industries blog RSS, [Online]. Available: https://www.adafruit.com/product/3263. [Accessed 2 July 2024].

[65] "Holland 8-in L X 4-in W X 2-in H Red/charcoal concrete paver in the Pavers & Stepping Stones Department at Lowes.com," [Online]. Available: https://www.lowes.com/pd/Holland-Red-Charcoal-Concrete-Paver-Common-8-in-x-4-in-Actual-7-75-in-x-3-88-in/3010214. [Accessed 28 June 2024].

[66] "EVERCOOL 30X30X7MM MEDIUM SPEED 5V DC FAN EC3007M05CA WITH 3 WIRE/PIN CONNECTOR," [Online]. Available: https://www.coolerguys.com/collections/dc-5-volt-fans/products/evercool-30x30x7mm-medium-speed-5v-dc-fan-ec3007m05ca-with-3-wire-pin-connector. [Accessed 1 July 2024].

[67] "Double insulated flameproof cable 2x2.5mmq 1 meter FG16OR162X2.5," [Online]. Available: https://www.elettronew.com/en/cables-and-pipes/cable-double-insulation-flame-resistant-2x25mmq-1-meter-fg16or162x25-14910.html. [Accessed 29 June 2024].

[68] PCBasic, "Global high-mix volume high-speed Shenzhen PCBA manufacturer," [Online]. Available: https://www.pcbasic.com/blog/best_pcb_file_design_cad_softwares.html. [Accessed 23 June 2024].

[69] "Aliexpress," [Online]. Available: https://www.aliexpress.us/item/2255800441357126.html. [Accessed 29 June 2024].

[70] "1-in x 10-ft Metallic Emt Conduit," [Online]. Available: https://www.lowes.com/pd/Metal-EMT-9-98-ft-Conduit-Common-1-in-Actual-1-163-in/5013672209. [Accessed 3 July 2024].

[71] "Motuka 15.7in x 26.6ft 198mil single reflective insulation foam thermal panel foam core heat radiant barrier 34 sqft aluminum foil Finsh white foam heat shield material - Amazon.com," [Online]. Available: https://www.amazon.com/MOTUKA-Reflective-Insulation-Aluminum-Material/dp/B0BHST44PM. [Accessed 28 June 2024].

[72] "40 Pound Midnight Black DOMINATOR Polymeric Sand with Revolutionary Ceramic Flex Technology for Stabilizing Paver Joints/Gaps, 1/8” up to 4”, Professional Grade Results - - Amazon.com," [Online]. Available: https://www.amazon.com/Pound-Midnight-Black-DOMINATOR-Revolutionary/dp/B09SVLX132. [Accessed 9 July 2024].

[73] "5pcs DHT11 Temperature and Humidity Sensor Module for Arduino MEGA 2560 AVR PIC Raspberry Pi 2 3 4B (Black) : Appliances," [Online]. Available: https://www.amazon.com/Organizer-Temperature-Humidity-Arduino-Raspberry/dp/B07TDFPKMW. [Accessed 30 June 2024].

[74] "AquaQuartz-50 Pool Filter 20-Grade Silica Sand 50 Pounds, White," [Online]. Available: https://www.amazon.com/Fairmount-Minerals-Filter-20-Grade-Silica/dp/B00JJ5GXSK. [Accessed 9 July 2024].

[75] "ATIKIL 26.3Ft 11AWG High Temperature Wire, -60-350 Degrees Celsius Mica Glass Fiber Electronic Wire Insulated Heat Resistant Electronical Flexible Cable for Lamp Boiler Heater, White," [Online]. Available: https://www.amazon.com/PATIKIL-Electronic-Insulated-Temperature-Electrical/dp/B0CXTDLCHS. [Accessed 28 June 2024].

[76] "Ceramic fiber Insulation blanket (24.1"x15"x1") Fireproof Insulation Kaowool for Dishwasher, Forge, Fireplace,Kilns, Stoves, Furnace 1500C/2700F : Tools & Home Improvement," [Online]. Available: https://www.amazon.com/Ceramic-Insulation-Fireproof-Dishwasher-Fireplace/dp/B0CBLQ2J18. [Accessed 28 June 2024].

[77] "DERNORD 120V 1650W Tri-clamp Foldback Heating Element Stainless Steel Immersion Water Heater with 3-Wire Electrical Locking Plug (1.5 Inch Tri clamp) - Amazon.com," [Online]. Available: https://www.amazon.com/Tri-clamp-Foldback-Stainless-Immersion-Electrical/dp/B07BXK5JWG. [Accessed 10 July 2024].

[78] "Fine-Soft Ground White Limestone | 50 lbs | Multiple Uses | Calcium Carbonate (50 lb) : Patio, Lawn & Garden," [Online]. Available: https://www.amazon.com/Fine-Soft-Limestone-Multiple-Calcium-Carbonate/dp/B09CHDTJ2F. [Accessed 9 July 2024].

[79] "GDSTIME Quiet 80mm Fan, Router Cooler Fan, 80mm x 25mm DC 5V USB Brushless Cooling Fan," [Online]. Available: https://www.amazon.com/GDSTIME-Airflow-Cooling-Computer-Electronics/dp/B07S2VCYFP. [Accessed 1 July 2024].

[80] "GearIT 12 Gauge Silicone Wire 600V (Black - 50 Feet) 12AWG Tinned OFC Oxygen Free Copper Stranded Soft Flexible Wires for Primary, Electrical, Car/Auto, Trailer, Amp - 12ga 50ft," [Online]. Available: https://www.amazon.com/GearIT-Gauge-Silicone-Wire-Black/dp/B0B74BJ8CX. [Accessed 29 June 2024].

[81] "tatoko 8 x 40mm 100W Cartridge Heater Electric Heating Pipe Fittings AC110V, 3Pcs : Tools & Home Improvement," [Online]. Available: https://www.amazon.com/tatoko-Cartridge-Electric-Heating-Fittings/dp/B07Q5R7GV6. [Accessed 10 July 2024].

[82] "Teton-Black™ Olivine Fine-Mesh Water-Bonded Foundry Casting Sand (24lb) : Arts, Crafts & Sewing," [Online]. Available: https://www.amazon.com/Teton-Black-Olivine-Fine-Mesh-Water-Bonded-Foundry/dp/B07N2GH8C7?th=1. [Accessed 9 July 2024].

[83] "Universal Electric Smoker and Grill Heating Element Replacement Part with Adjustable Thermostat Cord Controlle for Masterbuilt Smokers & Turkey Fryers 1500 Watts : Patio, Lawn & Garden," [Online]. Available: https://www.amazon.com/Universal-Replacement-Adjustable-Thermostat-Masterbuilt/dp/B0BR9148XM. [Accessed 10 July 2024].

[84] "Upgraded MP22YA 4 Pack Electric Range Burner Element Unit Set, 2 x MP15YA 6" and 2 x MP21YA 8" Stove Burners Replacement for Kenmore, Hardwick, Maytag, Norge Whirlpool Electric Range Stove : Appliances," [Online]. Available: https://www.amazon.com/LXun-Electric-Included-Replacement-Whirlpool/dp/B08BLJYTD4. [Accessed 10 July 2024].

[85] "100 Feet (30 Meter) - Insulated Stranded Copper THHN / THWN Wire - 10 AWG, Wire is Made in the USA, Residential, Commercial, Industrial, Grounding, Electrical rated for 600 Volts - In Black," [Online]. Available: https://www.walmart.com/ip/100-Feet-30-Meter-Insulated-Stranded-Copper-THHN-THWN-Wire-10-AWG-Made-USA-Residential-Commercial-Industrial-Grounding-Electrical-rated-600-Volts-In/1556002787. [Accessed 29 June 2024].

[86] "4 in. Galvanized Back-Draft Damper with Rubber Seal," [Online]. Available: https://www.homedepot.com/p/VENTS-US-4-in-Galvanized-Back-Draft-Damper-with-Rubber-Seal-KOM-100-U/205715168. [Accessed 29 June 2024].

[87] "Rheem PROTECH 120-Volt, 2000-Watt Copper Heating Element for Electric Water Heaters SP10874GH," [Online]. Available: https://www.homedepot.com/p/Rheem-PROTECH-120-Volt-2000-Watt-Copper-Heating-Element-for-Electric-Water-Heaters-SP10874GH/205650205. [Accessed 10 July 2024].

[88] "Master flow 3 in. x 2 ft. round metal duct pipe BCP3X24," [Online]. Available: https://www.homedepot.com/p/Master-Flow-3-in-x-2-ft-Round-Metal-Duct-Pipe-BCP3X24/100129266. [Accessed 20 July 2024].

[89] "Everbilt 3 in. x 8 ft. Semi-Rigid Aluminum Duct MFX38PHD," [Online]. Available: https://www.homedepot.com/p/Everbilt-3-in-x-8-ft-Semi-Rigid-Aluminum-Duct-MFX38PHD/313386903. [Accessed 20 July 2024].

[90] "4 inch x 5 foot dust collection ductwork spiral pipe," [Online]. Available: https://www.oneida-air.com/4-inch-5-foot-dust-collection-spiral-pipe. [Accessed 20 July 2024].

[91] OpenAI, "ChatGPT: July 2024 version," [Online]. Available: https://www.openai.com/. [Accessed 22 July 2024].

[92] "How a sand battery could transform clean energy," 14 April 2023. [Online]. Available: https://www.bbc.com/future/article/20221102-how-a-sand-battery-could-transform-clean-energy. [Accessed July 21 2024].

[93] "World's first giant 'sand battery' shows how energy solutions can be simple," 18 July 2018. [Online]. Available: World's first 'sand battery' can store heat at 500C for months at a time. Could it work in Australia? - ABC News. [Accessed 21 July 2024].