

SunChill, The Smart Sustainable Cooler



Group 17

Anya Legaspi - Electrical Engineering
Adam Shanholtzer - Electrical Engineering
Ethan Garcia - Electrical Engineering
Lana Perkins - Computer Engineering



University of Central Florida
Department of Electrical Engineering and Computer Science
Professors: Dr. Lei Wei and Dr. ChungYong Chan
Reviewers: Dr. Wei Sun, Dr. Michael Borowczak, and Dr. Morris Beato
Senior Design II – Spring 2024

Table of Contents

1. Executive Summary	5
2. Project Description.....	6
2.1 Introduction	6
2.2 Motivation.....	6
2.3 Goals and Objectives.....	6
2.3.1 Basic Goals.....	7
2.3.2 Advanced Goals.....	8
2.3.3 Stretch Goals: (In order of priority).....	9
2.3.4 Accomplishments	10
2.4 Functionality.....	10
2.5 Existing Products / Related Work.....	12
2.6 Requirements and Specifications.....	12
2.7 Block Diagrams	13
2.7.1 House of Quality.....	13
2.7.2 Hardware Diagram	14
2.7.3 Software Diagram	16
3. Research	18
3.1 Technology Investigation.....	18
3.1.1 Display Options	18
3.1.2 Cooling Options.....	20
3.1.3 Light Options.....	23
3.1.4 Voice Communication Options.....	24
3.2 Part Comparison and Selection.....	25
3.2.1 Temperature Sensor Comparisons	25
3.2.2 LED Lights Comparisons.....	28
3.2.3 Speaker Comparisons	31
3.2.4 Analog Buttons Comparisons.....	34
3.2.5 Keypad Buttons Comparisons	38
3.2.6 Electronic Locks Comparisons.....	43
3.2.7 USB Ports Comparisons.....	47
3.2.8 Rechargeable Battery Comparisons	51
3.2.9 Solar Panel Comparisons	53
3.2.10 Software Comparisons.....	59
3.2.11 Microcontroller Comparisons.....	60
3.3 Part Selection Summary	64
4. Standards and Design Constraints	66
4.1 Standards.....	66
4.1.1 Bluetooth Standards.....	66
4.1.2 Software and Hardware Validation Standards.....	66

4.1.3 USB Standards	67
4.1.4 LED Light Standards	67
4.1.5 Sensor Standards	68
4.1.6 Solar Panel Standards	69
4.1.7 Speaker Standards	69
4.1.8 Standard for Rechargeable Batteries	70
4.1.9 Standards Impact on Design	70
4.2 Design Constraints	71
4.2.1 Health and Safety Constraints	71
4.2.2 Environmental Constraints	73
4.2.3 Budget & Time Constraints	74
5. ChatGPT	77
5.1 Pros of AI Resources	77
5.1.1 Basic-Information	78
5.1.2 Proofreading	79
5.2 Cons of AI Resources	80
5.2.1 False Positives	80
5.2.2 Prompt Engineering v.s Learning	82
5.3 Conclusion	82
6. Hardware Design	83
6.1 Overall Schematic	85
6.2 Hardware General Block Diagram	85
6.3 Power Delivery System	86
6.4 Solar Umbrella and Solar Controller System	89
6.5 Lid Locking Mechanism	90
6.6 Speaker and Audio Amplifier System	93
6.7 Temperature Sensor System	95
6.8 LED Light System	96
7. Software Design	97
7.1 Software Inputs	97
7.1.1 Sensors	97
7.1.2 Keypad and Lock	98
7.1.3 Bluetooth App	98
7.2 Application Interface	99
7.3 Coding Design	100
7.3.1 Light Code Plan	100
7.3.2 Speaker Code Plan	101
7.3.3 Light and Speaker Plan	101
7.3.4 AI Voice Plan	101
7.3.5 Temperature Sensor Plan	103
7.3.6 Keypad Code Plan	103

7.3.7 Total Software Part Integration.....	106
8. System Fabrication/Prototype Construction.....	107
8.1 PCB Design.....	107
8.2 Cooler Configuration.....	110
8.2.1 Overall Construction.....	110
8.2.2 Internal Design.....	110
8.2.3 Front View Design.....	112
9. System Testing.....	113
9.1 Hardware Testing.....	113
9.1.1 LED Lights.....	114
9.1.2 Temperature Sensor.....	114
9.1.3 USB 2.0 Breakout Board.....	115
9.1.4 ESP32.....	115
9.1.5 Keypad Testing.....	116
9.1.6 Lock Testing.....	117
9.1.7 Speaker and Audio Amplifier.....	118
9.1.8 Rechargeable Battery.....	119
9.1.9 Solar Panel and Solar Charge Controller.....	120
9.1.10 Button Testing.....	121
9.2 Software Testing.....	121
9.2.1 MIT App Testing.....	122
9.2.2 Arduino Testing.....	122
9.3 Overall Integration.....	122
10. Administration.....	124
10.1 Budget and Financing.....	124
10.2 Initial Milestones.....	125
10.2.1 Senior Design 1.....	125
10.2.2 Senior Design 2.....	126
11. Conclusion.....	127
Appendix.....	129
Appendix A: Code and Data Sheets.....	129
Appendix B: Figure Index.....	140
Appendix C: Table Index.....	143
Appendix D: References.....	144

1. Executive Summary

Every year, the beaches in Florida get hotter, and our goal was to find a way to assist people in having a convenient way to keep their drinks cool, safe, and fun to access. Most coolers on the market have Bluetooth apps with smart features, but most don't have a full package of features. Our team's goal is to have a cooler with smart features that aren't usually combined, with new features that are rarely seen in the market. Our team's goal is to also involve an AI voice to allow a seamless user experience and use some of the newest technology in the market.

Our team proposed the SunChill, a smart cooler with multiple features all accessed through a Bluetooth phone application. Through the phone application, a user is greeted with a screen to unlock their cooler, and then have access to a multitude of features not seen on many coolers. During the day, while taking a cooler out to the beach a user has a solar powered device that can access the temperature and current battery percentage. It isn't just an application, but a solar powered cooler as well that connects to a battery that powers all the features the application can access, with no need to plugging it into an unnecessary outlet. A user can also connect their music to the application to play music from the cooler, as the cooler is also an easy to use speaker. During the day (or night) the user can light up the night with a lightshow placed on the outside of the cooler.

The more intense features that are also included are the AI voice and music reactive lights connected to the cooler. The AI voice now involves a LLM, or a Large Language Model, from a commercial vendor programmed into the application for a user to speak to, and will respond with features from the cooler so if the user doesn't want to press on the application menus, they can just ask for them instead. For the light show, a user can connect their music and the lights together for a reactive light show. This is done when the music is connected a variable will change, and give the user the option to go into music reactive mode. Then, a signal is sent to the sound sensor to activate, and the light function will switch to music reactive mode. These parts are both software intensive, and is the main reason for the switch from a hardware to a more software based project. But with these features involved, it leads to a more simple, smart, and enjoyable user experience.

The research and development of the SunChill is detailed in the sections below. It begins with the basic description, and how this idea comes to be. It shows what we want to accomplish, and how our project transitions from an electrical project to a more software-based design. Then it leads to a research section, where it details the ideas of how solar panels work, light shows function, and how AI voices are generated. It also details the constraints of how we switch from a compressor to an ice-based cooling design. It then leads to our actual design methodology. The hardware components of all the sensors and parts are shown in detail and how each one functions with the others. The software section explains how our main software, the Bluetooth application, is going to be programmed and how each standalone 'function' will be programmed. The next section is the physical testing of each component, and the implementation of the PCB for how the main MCU communicates with every part through Bluetooth to the application. The final section explains the scheduling for the teams and further resources that assist us during the process.

2. Project Description

2.1 Introduction

For everyone living in Florida, we can all agree that the heat here is no joke. We all feel the sun rays beaming down at us at all hours of the day, draining us of energy. Whether we're sunbathing at the nearby beaches, swimming at the pool, going to class, or simply walking to our car, we can definitely feel it. Just recently, Florida broke its past record and set a new high for the hottest July on record (July 2023). Living in the hottest state in the USA means we have to prioritize staying hydrated. This is why being able to bring cold drinks is an important facet of being outside. No one wants to swim at the beach and come back to a lukewarm bottle of water that's been sitting out on the sand! Everyone should be able to enjoy the outdoors and refresh themselves with a crisp and cold drink. With our cooler, we are able to bring water and drinks while keeping them cool and refreshing. Spending time outside, relaxing, and enjoying the outdoors is an amazing experience, and with these record high temperatures, we have to make sure we are ready and well-equipped to take on this heat.

2.2 Motivation

By creating a cooler with many different smart features, we are innovating and redesigning a seemingly basic object into something that solves modern problems and adds creature comforts of today. The reason we love this project so much is that we can add so many features to it that have a useful and meaningful purpose, it's just not adding things to add them. We believe with this project we can turn a 'dumb product' into something very smart and appealing to most people. When looking at most smart coolers we notice none of them have true portability, as in they always need to be plugged in to run any smart features. With the addition of PV charging capabilities, this allows our project to be used anywhere. This smart cooler also becomes the perfect tailgate cooler. When there's lots of students and a busy crowd around, a person doesn't want just anyone walking past being able to open their cooler and grab a drink, especially if it is alcoholic. Locking the lid allows a person to keep their drinks safe from strangers and also prevents alcoholic beverages from getting into underage hands. The basic cooler just doesn't cut it anymore; we want to change and revolutionize it to something even better. With the addition of our solar umbrella, this opens us up to a whole new world of what we can do. With this umbrella, we not only can replace a typical beach umbrella, we are also able to increase the longevity of our cooler's functionality while giving users the ability to charge phones, power portable fans, and other devices that users may not have had the ability to use before. The solar umbrella on its own is such a powerful and useful tool for the beach, and pairing it with our smart cooler will make this duo unbeatable in both usability and functionality.

2.3 Goals and Objectives

We have decided to split up the possible features into three categories labeled 'Basic', 'Advanced', and 'Stretch Goals'. Features under 'Basic' are at minimum our main focus of implementation and should be on the final product. 'Advanced' features are ones we wish to include for convenience for the user and/or as a challenge for us as engineers. 'Stretch Goals' are meant for any final touches

that we would want to add to our project to polish and make it a commercial-ready device. These features will only be targeted if there is extra time to work on our project.

2.3.1 Basic Goals

Temperature Sensor

When a user uses a typical cooler, they just throw in ice and hope things stay cold the whole day. Some days they come back home and most of the ice is still there; on other days, the ice may be completely gone. This is a gamble; if a freshly caught fish is still kept at safe temperatures and if their food is still safe to eat. Adding a temperature sensor is a safe and simple way to track our cooler's internal and external temperatures. This provides users with useful real-time accurate data on if their cooler is keeping items at a desired temperature. Our temperature sensors are a noninvasive safe design and provide us with powerful data throughout cooler usage time. If time is not an issue, we will be able to implement an external temperature sensor on the exterior of our cooler to provide real-time outside temperature updates to the user. With this sensor, we open up more data to be available and used in other applications for our cooler.

LED Lights

Since we have Bluetooth functionality, we want to have light integration using LED addressable lights. When a user is using the cooler, we want it to be an all in one machine, so having the cooler and projecting a light show can add to the use of it to be used during the evening. For food and safety reasons, we want to have the LEDs on the outside of the cooler. It will be bright enough to illuminate the area and the content within the cooler. We also want it to not just be lights, but a selectable light show. For our main goal, we want to have a programmable light show and with the application, allow a user to change between different shows and colors.

On/Off Switch

Since we are using a portable and rechargeable battery, every second matters for our smart cooler in terms of conserving energy and being as efficient as possible. One of the ways we can protect this idea is to incorporate a simple On/Off switch onto our smart cooler. This ensures that power is cut off from the system and the battery so the battery does not excessively drain at all times, but instead, only when it is needed. This switch is extremely cheap and simple to implement, and also extremely simple to test that it is working properly. The switch that we choose needs to have some degree of protection from the elements (i.e., water, dust, dirt, etc.), since it will be attached to the outside of our smart cooler.

Keypad Buttons

We are using a keypad device to allow the user to input a device passcode in order to lock and unlock the smart cooler. Keypads typically have twelve buttons (numbers, star, and hashtag symbols), but there are some that are manufactured to have sixteen buttons that can typically be used as auxiliary buttons or macro keys, to be used for any purpose we may need or want in the future. The keypad that we chose has some degree of protection from the elements (i.e. water, dust,

dirt, etc), since it is attached to the outside of our smart cooler.

Speaker

Having a speaker allows for audio output from the cooler. Since we use voice activated commands, the speaker itself is able to provide feedback. For example, if we lock the cooler we are able to audibly hear the cooler state that it is locked or some kind of locking sound to indicate it. In addition, the speaker can also be used to play music when their phone is connected to the application.

USB Charging Ports

Since we are taking advantage of our smartphones through our Bluetooth application to control some cooler functionality we want a way for users to be able to charge those devices. We are providing two weatherized USB type A charging ports. We decided to choose this because almost all mobile devices use this interface. This allows the user to charge their devices like any other place they would charge. We are also ensuring that these ports are weatherized so they do not short out and are protected in transit and in use.

2.3.2 Advanced Goals

Locking the Lid w/ Passcode

We have implemented an electronic lock (solenoid lock) into our smart cooler that works in tandem with the keypad device mentioned above. This prevents any unwanted user from accessing the inner contents of our smart cooler if they do not possess the passcode selected by the owner of the smart cooler. This smart locking mechanism is built heavily and sturdily enough to withstand the average attempt of getting into a locked system. It also is small enough to fit on the inside of the lid or smart cooler to leave room for the actual contents of the smart cooler to fit (i.e drinks and food).

Rechargeable Battery

Having a rechargeable battery means that our cooler is as portable as can be. We do not want a cooler that can't be brought around, that defeats the main purpose of using the cooler. With this, our cooler still has all of the functionality and effects while still being able to bring it around wherever a user goes. This battery is capable of powering the entire cooler and its peripherals.

Bluetooth App

To control the cooler, instead of a remote or using a display on the cooler, we have integrated a Bluetooth application to work with the cooler's functionalities. For every part, there is a functionality on the phone application to access the data. The plan is to be able to check the battery, temperature, unlock/lock, connect the speaker and turn on and off the lights. This is intended to be a basic application, but as a stretch goal would be more functionality on the application and more

user interface options.

Smart LED Lights

Having not just a light show, but one timed to the music connected to the speakers, is one of our main advanced goals. There would be no need for the user to get a portable speaker if they just use this product which is a speaker and a cooler all in one. The plan was to connect the lights to the Arduino with an audio detector that connects to the speaker. Then having each addressable led synced to a certain part of the music (treble, bass, etc), it allows variable light shows to any kind of music. The application detects the music on the phone, then broadcasts it to the Arduino, and then it syncs the music to both the speaker and the lights for a congruent show and entertainment for the user.

Voice Activated Commands (AI Voice)

Having to not need to even press a button on the application but talk into it to turn on a function is a key advanced goal. The user would just need to press the application and then talk, with an AI voice responding back to the user and the cooler doing as told. For simple functionality, there will be certain phrases implemented to ensure certain functions work. A further work on this is to allow more commands and then just allowing complete AI control with it knowing what the user wants by them just asking for it.

Solar Panel

Working in tandem with the rechargeable battery, this solar panel is able to provide supplementary power to the battery and extend the cooler's life. We want the cooler to be in the shade, and what better way than to implement solar panels on top of the cooler. This is a dual function because now it provides shade to the cooler while also providing charge to the battery. The solar panel is portable as well, similar to a typical beach umbrella. By having the solar panels arranged in a configuration, we are able to fold and compact the solar panel down for storage as well as unfold and expand it for its purpose.

2.3.3 Stretch Goals: (In order of priority)

Design our own AI voice

A key stretch goal is to not just use an ai voice, but make our own to use in the system. Using the latest AI technology it is possible for a programmer to use voices to create their own voice and use LLMs to create the backbone of an AI chatbot. We want to use our voices as a basis combined into one voice to ensure a unique experience for the user.

Compressor

If we have the time and desire to continue making our smart cooler more smart than we currently plan it to be, we will begin on implementing a compressor unit into the smart cooler in order to

bring the interior of our smart cooler to the same temperatures measured inside of refrigerators and freezers used for kitchen purposes. We plan on using micro compressor units that are typically found inside many commercially available miniature fridges as a base for our cooling system. We will then need to supply power to the compressor in order for it to run and cool the temperature of the cooler down. With this, we can extend the longevity of items in the cooler.

2.3.4 Accomplishments

Below will indicate our accomplishments we have created. In general, our basic and advanced goals were accomplished, but had some bugs along that journey that are still being worked on to this day. For our stretch goals, due to time constraints and limited space on the cooler and ESP, we could only accomplish the basic and advanced goals.

Basic Goals

For the basic goals, all of them work successfully. For the simple ones that were accomplished, the temperature sensor senses the temperature inside the cooler easily with a few lines of code and the keypad and USB ports were simple to integrate. However, the lights and speaker were finicky in the beginning. By creating an addressing system along with coding tricks, the LED lights and speakers can work together without any issues. If there is any issue, it is that the speaker doesn't give the clearest output, but we attempted to alleviate as much as we can by modifying our amplifier.

Advanced Goals

For our advanced goals, we were able to accomplish all of them, but with some caveats. For locking the cooler and the rechargeable battery, those were able to be accomplished. Locking the lid, however, we decided it would be smart to have a safety feature of making it unable to change the passcode on the cooler while the phone app can be changeable. This would be if somehow the main user forgot the code on the cooler they wouldn't be locked out of it. For the bluetooth app, it was accomplished without issue due to the ESP32's easy bluetooth capabilities. For the Smart LED, it mostly works. The main issue is the connection to the sound sensor is finicky with the wiring so the data is not always consistent. Also due to board issues, sometimes playing the function breaks the system, so that had to be debugged. For the final build, it is still being debugged to this day to ensure its proper functionality. For the AI voice, they work mostly, but due to the AI, it sometimes gets confused whether we are asking it a cooler question or a general question, so that is still being debugged. Finally, for the solar panel umbrella unfortunately it couldn't be made collapsible. We did make it as an umbrella, but its functionality is limited.

2.4 Functionality

This cooler has three main purposes and functionalities: Voice activated commands, locking/unlocking mechanism, and the usage of a Bluetooth application to control all these features. In addition to the cooler, the utilization of a solar panel umbrella will provide supplemental electricity to run all of these features.

Our cooler has a plethora of user input/output features that we believe are all quality of life changes that serve a real purpose and are not for show. Starting with our Bluetooth connectivity application we are able to bring the smart of a phone to the smart of the cooler. Our phones are such powerful devices that we use every single day. We believe the simplest way to achieve the best user input/output is experience is through that device. With our Bluetooth application we will be able to use our cooler in ways that were never thought of. The features integrated with the application will be temperature tracking, audio output through speakers, remote locking and unlocking, LED light controls, and voice activated commands.

We aim for the user to be able to view and use these features with ease, we will also implement voice command prompts to control cooler features. With this in mind we plan on using AI to potentially add another layer to our coolers functionality. By having voice activated commands, we are able to provide more ease of use with the simple use of a couple phrases to activate a desired command.

We also do not want to forget our physical hardware for input/output. We plan on having basic USB charging functionality for small personal devices and basic analog on/off control for basic LED functionality.

With locking the lid, our cooler will be vastly more secure and safe compared to other coolers on the current market. In doing so, we will be able to keep underage participants safe from accessing alcoholic beverages. Anything they don't want to be publicly accessed can be easily locked with a pin code or using our mobile application. Since there are a lot of smart features embedded within the cooler, we want to make sure we have the ability to keep it safe. No one wants to steal an unusable cooler, and having a physical or virtual pin code provides a simple yet effective deterrent against it.

Solar charging is another useful functionality which will allow us to broaden and extend the longevity of our cooler's functionality. By having a PV panel implemented, our smart cooler now offers a green renewable solution that isn't commonly seen on smart coolers today. Since we have many electronic components that need to be powered, we want to make sure the cooler is able to stay on when used throughout the day. With the addition of a PV panel, this will allow for supplemental electricity to be used and will prolong the cooler staying on for maximum benefits and use. To configure these panels into a solar umbrella is where things get very interesting. Not only are we making a 1 for 1 exchange with a typical beach umbrella, we are able to take that and add much more. With the Solar Umbrella (SU) we will be able to charge our coolers' batteries while being able to run higher power consuming products such as fans and battery charges with no worry as long as the sun is out.

All these combined features into our smart cooler creates a different and revolutionized cooler that hasn't been seen before. In addition, this cooler provides a basic structure that can be expanded upon with our stretch goals. We are not limited to only these features, we have limitless ideas that we can implement here to make our smart cooler useful and beneficial for everyone. We truly think that if needed, this cooler could be customized to an individual's personal needs and wants.

2.5 Existing Products / Related Work

Initially for our project we wanted a self cooling portable cooler without ice that has time tracking, temperature sensing, and configurable lights all in one product. For example, the Outdoor Refrigerator Dual Zone cooler from Vevor is a smart cooler that has limited smart features. Using a Bluetooth application it advertises battery and temperature, yet there can be much more that can be done. We planned to expand that with our application that could control temperature, lights, check battery and control the time. Another project related to our senior design project controlled LED lights with little to no temperature control or monitoring. Ours has dynamic and intelligent light control, with temperature tracking and monitoring.

Another example we can look at is the Dometic CFX3 75DZ Powered Cooler. This is one of the best coolers out in the market with a price point of \$1,249.99. It has cooling capabilities, an application, and lighting. This was one of our inspirations for our project. It has great reviews for the product and has above average cooling capabilities. However, its application is limited only from the temperature changes. Our team's goal was to do a little less of the cooling aspect, to budget constraints, and expand on that application. It advertises temperature control and temperature tracking but nothing for lights or other smart features. If we can leverage the smart features then for different reasons our product can be comparable to this.

One of the biggest features of our cooler is the smart locking system. From our research, an example such as the Igloo Tan IMX 70 Qt Lockable Insulated Ice Chest Injection Molded Cooler. It is able to lock the lid and keep cool, but it is a physical lock. Having one controlled by a pin pad or an application hasn't been seen in any coolers researched. Having the cooler be lockable keeps away unwanted users and underage drinkers if alcoholic beverages are present in the cooler. Having this in our project, while not being explored before in a cooler, is one of our highlights and a feature we believe many people will cherish.

2.6 Requirements and Specifications

Below is a *Table 1* listing out all of the components we plan to use in our project in one way or another. All of these components are necessary for the success of our project of building a smart cooler because they all play a unique and specific role that brings the whole "Smart Cooler" together. We have designated four components in particular that we are confident in being able to test during the final demonstration of our project in Senior Design two. These components have been selected because of their ease of access on the smart cooler, quick response from the testing, and relative ease in testing. The temperature sensor is a device we wanted to prove accuracy of plus or minus one degree celsius and we compare the results with a physical mercury based thermometer. The LED lights need to work when prompted so a visual cue will determine the success of them activating nearly every single time. The same principle applies to the keypad that is attached to the outside of the cooler. Only difference is we tested all programmed buttons and ensured they provide the correct data to our PCB. And finally, the locking mechanism on the cooler lid works whenever power is applied therefore we will physically test the lock to see if it works after the battery is connected.

Commented [1]: move to end or diff section entirely

Commented [2]: actually he likes this here so imma leave it

Component(s)	Parameter	Specification
Temperature Sensor	Accuracy	+/- 1°C
Battery Life Sensor	Accuracy	≥ 95%
Lid Status Sensor	Accuracy	≥ 95%
LED Lights	On/Off	≥ 95%
Keypad	Provide Proper Input	≥ 95%
Portable Battery	Discharge Time	~ 5 hours
Solar Panel	Power Output	≥ 10 Watts
Cooler	Weight/Size (Empty)	≤ 30LBS/~32QT
Bluetooth App	Connects to Cooler	≥ 95%
AI Voice	Collects input/Gives proper Output	≥ 95%
USB Ports	Charge external devices	~ 1 hour
Locking Lid	Successfully locks and unlocks lid	≥ 95%
Speaker	Outputs sounds/AI Voice	≥ 95%
System Functions	Startup Time	~ 30 seconds

Table 1: Project Specifications

*The highlighted requirements are able to be tested for the Final Demonstration

2.7 Block Diagrams

2.7.1 House of Quality

The House of Quality, shown in *Figure 1*, correlates the engineering requirements with the customer requirements. This diagram provides us a clear view of specific customer requirements and how the product specifications affect those. This helps us understand what we need to focus on for our project to be satisfactory for the customer. With this, we can look at our Smart Cooler from two perspectives: the engineering side and the marketing side.

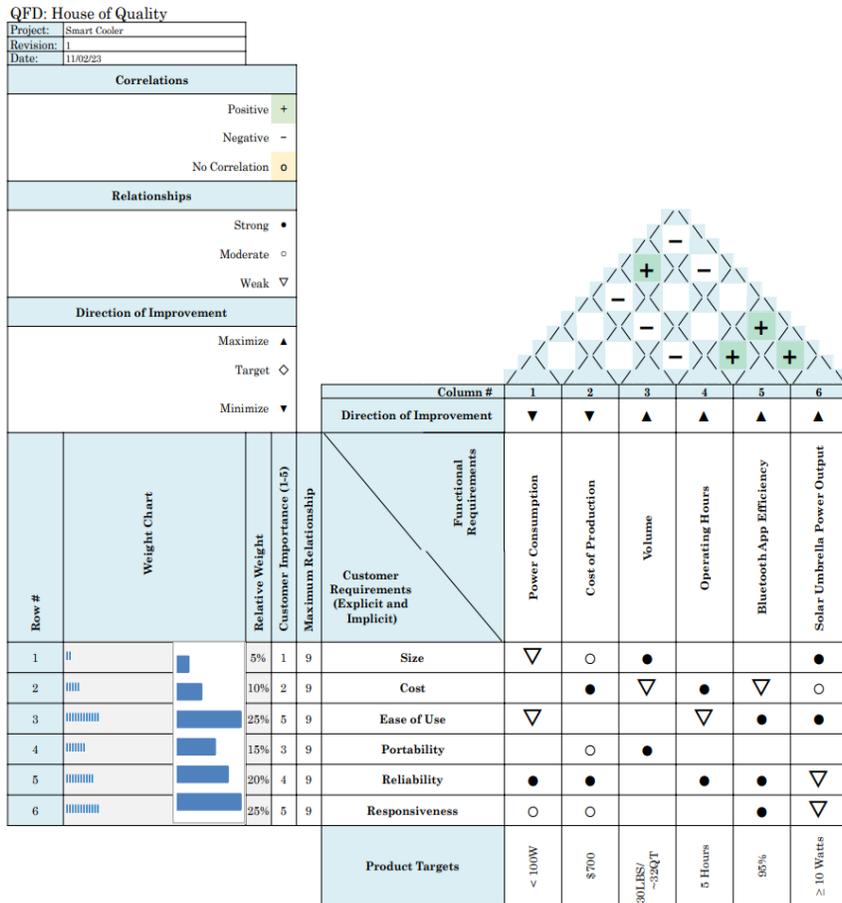


Figure 1: House of Quality

2.7.2 Hardware Diagram

This represents a basic hardware diagram of the electricity flow throughout the system. First we receive power from our photovoltaic panels. This DC power then goes to our Solar Charge Converter which will show us our incoming voltage and current in real time; this device will be used to control the flow of power into our system for over-current/voltage protections. All power in this system is then derived from our battery and provided to all our other systems. The USB ports have the most direct line of power since they are strictly power-in and no data is being sent too and from. Then all remaining power is sent to our mcu which will handle power delivery to all other systems in our cooler. This system showcases how we can use the power of the sun to cleanly

harvest energy to use and store within our system. The details of this plan is shown below in Figure 2.

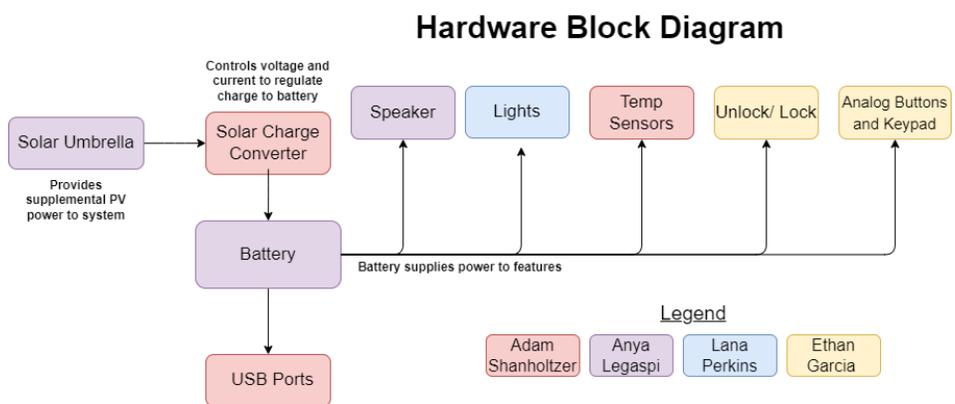


Figure 2: Basic Hardware Block Diagram

2.7.3 Software Diagram

Below is *Figure 3*, a diagram of how the software will interact with all the functionalities required for the cooler.

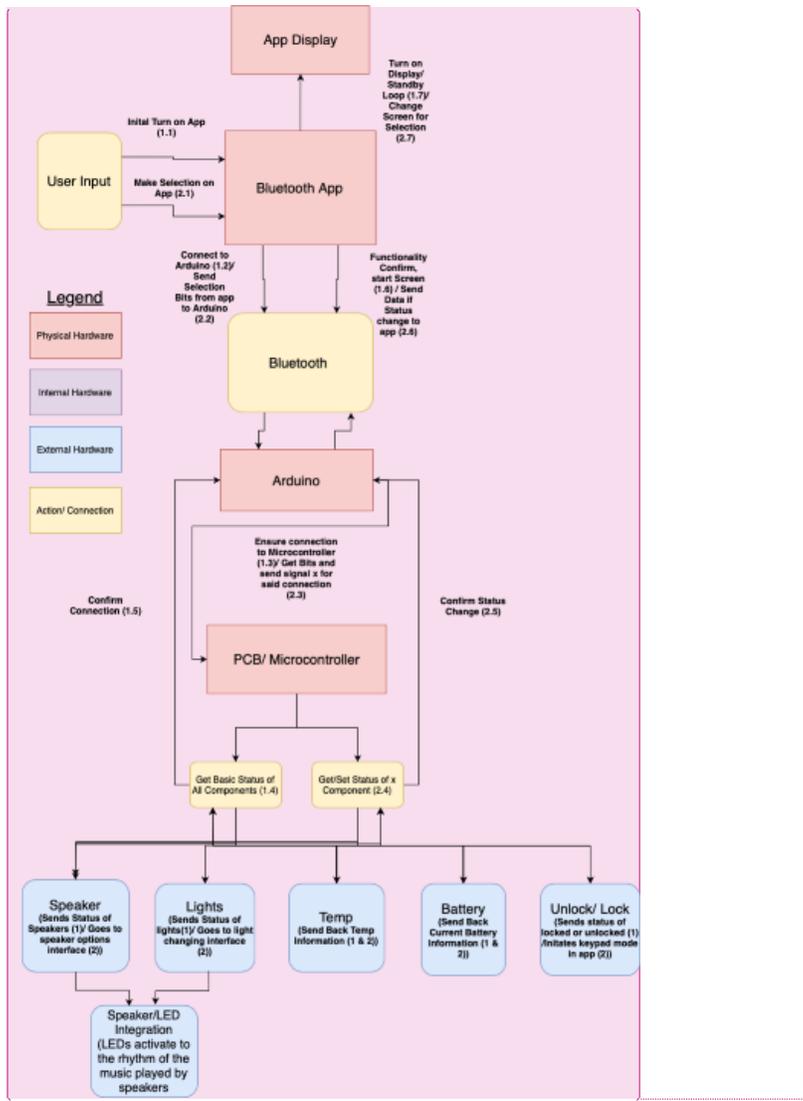


Figure 3: Software Diagram

Commented [3]: i just added the adjustment here @anya cuz we were dmng about it

The smart features of the cooler are accessed from the user pressing on the application. Upon startup, the Bluetooth application on the phone will begin and immediately ask for a Bluetooth connection to the Arduino. Once it obtains that connection, then the Arduino will send a message to the PCB to obtain the current status of all the smart components on the cooler. The PCB sends back a confirmation signal if there isn't anything wrong to the Arduino and then the Arduino will send a Bluetooth response back to the application where it then will display the current data to the user on the application screen. Then there will be buttons for the user to press such as the speaker, lights, temp, unlocking/locking the cooler, and battery status. When the user presses a button, that button will have a value and it is sent to the Arduino and sent to a port to send to the PCB. The PCB will obtain that number and it will be the selector for what smart feature is accessed.

The Battery and Temp will just send a signal back without any further user input. As they will not be messed with and only checked, the user will see the data on a screen and the next push of any buttons will only remain in the application for those 2 features. For the lights, this will involve further steps from the Arduino to the microcontroller where the first signal to check the lights will just be the first cycle in the loop. It will show the status of the lights and further options. For the time being, it is planned to be changing light patterns, turning on or off, or changing brightness. The light patterns will further be subdivided to just a normal sequence or to match with the music. The user will not be able to manually configure the music pattern. The unlock/lock will be a simpler feature. If the user presses the button, it will lead to a screen that will have a passcode to input. For the time being, the passcode is hardcoded so the user must know that code and press it in the application. If it passes then a signal will be sent to where the motor will unlock the cooler. The final is the speaker. If the speaker button is pressed then a screen will appear with 2 options. One is to change the speaker volume and the other is to sync the music. Speaker volume will be a screen with a slider going up and down for the volume. The sync screen will then wait until a user begins playing music, and that will send it to the speakers. Then those speakers will interact also with the lights for the music patterns.

3. Research

3.1 Technology Investigation

3.1.1 Display Options

The Cooler's smart features need a user interface to interact with the smart features. There are three avenues that were considered: an LCD touchscreen assembled into the cooler, an LCD screen without touch capability using a keypad, and a Bluetooth application. The following information will regard each of the advantages, disadvantages, and ultimate conclusion. Below is *Table 2*, where it mentions the options that can be employed.

Method	Cooler Power Consumption	Microcontroller Necessity	Popular in Industry	Price
Touchscreen	Most	At least Raspberry Pi	Moderate	\$20-\$100
Non Touchscreen LCD	Moderate	Any Microcontroller	Least	\$10-\$20
Bluetooth App	None	Needs Bluetooth	Most	N/A

Table 2: Display Options

Touchscreen

A touch screen allows an interface directly on the cooler with emphasis on UI design and functionality. This route would have a 480 x 320 display where the user interacts with the smart features of locking, temp control, and others by pressing on the screen. As this was the main connection between user and cooler, a GUI with extensive features would have been created. Programming a touchscreen, however, would be difficult for a simple microcontroller unit to handle. This route would involve an extra microcontroller, and the best fit would be a Raspberry PI. As it was already owned, a Raspberry Pi3B would have been used, due to the CPU and OS capabilities. Having an OS run a locally hosted web application on the PI would have allowed multiple customization features to the application, and a display with touchscreen functionality is already seen in the market.

To test touch screen coding, The Hosityond 3.5 Inch 480x320 Touch Screen TFT LCD SPI Display Panel for Raspberry Pi A, B, A+, B+, 2B, 3B, 3B+,4B was used on a Raspberry Pi3B and was tested. The installation process was simple, however there were some drawbacks. A simple display that was used was cheap, but to get one that has a touchscreen and is waterproof ranged in the \$60+ price range. On another note, the coding was simple, acting like a coordinate system like the joystick control from the embedded systems class project. It wasn't perfect, and the one tested tended to be too finicky to get the exact coordinate to correlate with touch. The area on the screen for each section needed to be big to account for this error, but as the screen was too small, there

wasn't enough room to do it properly. Another issue would be power, as the Raspberry Pi requires 5V compared to a Arduino's 3.3V, and would cause unnecessary power to the display.

To sum it up, the touchscreen would lead to more time coding a GUI and allowed more experimentation in the CS coding realm. However, with the cost, excessive power of the Raspberry Pi, and its difficult configuration, this route was not pursued any further.

Non-Touch LCD Display with Keypad

Another avenue considered was a small embedded screen without touchscreen capability and used a keypad to control what the user sees. It would involve simple C code to program and requires no GUI. The user would press the buttons to select which smart feature to use, and use the up and down keys to change the temperature. This route would involve a 4 inch display and only showed necessary information. Because of this, a simple microcontroller would be used to interface with the display like the Arduino Uno. This can also lead to a display needing just a waterproof enclosure, so there would be no need for an expensive waterproof display like the touchscreen. The locking feature could be easier implemented with the buttons. The keypad would be used to lock/unlock the cooler lid using a 4-digit passcode inputted and stored on the embedded system, leading to a more satisfying user experience. This could all be achieved by using an Arduino kit and solenoid lock.

The key issue with this idea would be the space required for the display and for the buttons. The touchscreen only required the microcontroller and the display, but this design would need another component implemented. There would be less space for the actual smart features that the display and buttons would now take. Also this route leads to little UI and GUI design, using only a simple display which would look cheaper and more primitive compared to touchscreen.

This design was originally considered over the touchscreen, as the simpler coding, implementation, and less power usage overshadowed the space issue. However, an issue that was considered was battery depletion, and market necessity. So in the end, this wasn't implemented.

Bluetooth App

A Bluetooth application would involve an external device that would take away any unnecessary space and power from the cooler. Also such as the Vevor 1.27 cu. ft. Outdoor Electric Compressor Cooler w/Application Control & Wheels, and most coolers in the market, Bluetooth functionality is integral to the smart cooler audience. Phones are everywhere, and having all the smart features easily controllable from the phone is what's in demand.

This route would require an external device to connect Bluetooth with the Bluetooth application. A Raspberry Pi doesn't work well with IOS, but Arduino apps can work with most modern phone types. To test Bluetooth functionality, the DSD TECH HM-10 Bluetooth 4.0 BLE iBeacon UART Module with 4 PIN Base Board for Arduino UNO R3 Mega 2560 Nano was bought, used and tested. 4.0 BLE works with both IOS and Android, and has a simple wired connection to the Arduino Uno. Simple tests were employed and Bluetooth functionality was realized using an iPhone 13.

The biggest hurdle will be the actual application development. Arduino is the easiest compared to most systems, but the team have mainly had electrical experiences. Doing an application as a central point would cause most of our focus to shift away from electrical, and more towards understanding application development. However, with the advent of new AI technology and the added bonus of using AI apps and user customization, this can lead to time investing on topics popular in industry. Another issue is battery depletion. If the battery that connects the Bluetooth receiver dies, then there is no way that the application can connect to a dead cooler.

Due to commercial smart cooler features, power usage, and space efficiency, the Bluetooth route was taken. An application will be created interfacing with an Arduino interface, and to fix the issue with the battery dying, a keypad will be installed into the cooler so there will always be a way to access it with or without the application.

3.1.2 Cooling Options

A cooler is meant to keep drinks cold, so a way to cool the items inside of the cooler is a necessary feature to the project. There were three options considered: Thermoelectric Cooling, Compressor Cooling, or Ice. The following information will regard each of the advantages, disadvantages, and ultimate conclusion. Below is *Table 3*, where it mentions the option we chose.

Method	Power	Temperature	Efficient	Price
Compressor	Yes	≤ 40°F	Yes	~\$200
Peltier	Yes	40°F lower than ambient temperature	No	~\$15
Ice	No	0°F - 32°F	No	~\$2

Table 3: Cooling Options

Thermoelectric Cooling

Thermoelectric or Peltier cooling is cooling that uses conductive plates made of semiconductors placed in series to pull heat from one pad and transfer that heat to the other pad - making the first pad much cooler. This is achieved by simply applying a small voltage to the system which begins to transfer the electrons within each pad in a cyclical manner. Combining this technology with heatsinks and fans can improve the effectiveness of the overall cooling potential by cooling the air around the “cool pad” and expelling the hot air from the “heat pad”. This type of cooling is popular in multiple DIY projects, and is cheap to create compared to Compressor cooling. The plan for this would have the plate placed on the top of the cooler to cool the items inside of the cooler.

Commented [4]: add in how peltier is used in very small scale applications such as electronic chips

Commented [5]: add in example about its application in wine coolers

The most prevalent integration of thermoelectric cooling using Peltier devices can be found in the wine cooling industry. Many wine coolers have adopted Peltier cooling systems because they are extremely cheap to manufacture, maintain, and operate. Wine coolers also need to have as little humidity as possible so as not to interfere with the corks and wine itself. Compressors have the ability to cool air quickly and efficiently, yes, but are also susceptible to massive humidity exposure, whereas Peltier devices do not have this issue.

The main deterrents toward this method is the cooling effectiveness and smart implementation. Most projects have the thermal cooling to work in seldom, with little smart functionality shown in prior projects. A solution to this is to have the thermal cooling be separate, but then it would make changing the temperature from an application an issue. The other issue is that the cooling capabilities are poor compared to other options. Thermoelectric is known to make a cooler only about 40 degrees colder, and the amount of electricity required for a small amount of temperature change is more expensive and less efficient than compressor cooling. Another factor to consider is the time it takes to achieve this final cooling temperature as well as the total volume of air that Peltier devices can cool before the system peaks past the curve of efficiency based on how much voltage is applied to the system.

The main factor for this method is the ease of implementing this alone, but that is also why this outcome was not picked - in order to get a better cooling factor and higher efficiency, we could simply add more Peltier devices, heatsinks, and fans in parallel with each other. This works because as more and more Peltier devices are added in parallel, the overall current capacity increases due to more cells sharing the load. Therefore, more current can be applied to the system which can increase the overall flow of electrons from the hot plate to the cold plate, thus increasing efficiency without the risk of overloading the individual cells. Another benefit to stacking Peltier devices in parallel is the overall stability of the entire system increases. This is because there is no reliance on any particular cell maintaining proper functionality in order for the other cells to function properly. The greatest analogy to this idea is classic Christmas string lights from the 20th Century; if one bulb goes out, then all subsequent bulbs will not receive current to light up. This will not happen on a parallel stacked Peltier system, therefore, the overall stability of the cooling system increases. Typically when any type of unit is placed in parallel, there is a decrease in overall resistance to the system. In our case with Peltier devices it is the same. This is another benefit due to the common idea that resistance is inefficient because it releases energy from the system as heat. So by building the cooling device with parallel stacked Peltier devices, we can lower the overall resistance load which will make our system more efficient since less energy will be expended as heat.

If the project's focus was on cooling, then a compressor would be picked, and as smart features are the project's main objective, having thermoelectric is only going to cause unnecessary extra steps with little cooling outcome.

Commented [6]: i feel like this stat is considered good not poor. gonna find other stats that show it only goes about 10/20 degrees below ambient temperature

Commented [7]: can expand more on this explaining that ties back to peltier is best at cooling when the cold plate is directly touching the device

find some formula or the conductivity of air vs metal saying how hard it is to cool a volume of air vs metal

Commented [8]: Expand on what happens when peltier is wired in series/parallel

Say why wiring it in either configuration is not good (lots of watts and current draw that is not comparable to the cooling it will achieve)

Compressor Cooling

Compressor cooling involves 3 parts: a evaporator , a compressor, and a condenser. Cooling is done by having refrigerant pass the evaporators expansion valve, where the refrigerant expands, boils and evaporates. The liquid turns into a gas, and the compressor recompresses the gas and turns it into a liquid. This action causes it to rapidly boil and then suddenly decrease to temperatures generally around of below 20C. This technology is used in commercial cars, refrigerators, and most smart coolers.

The main arguments towards this technology is the extensive documentation and cooling capabilities. The cooler can easily cool most objects and can be used in big or small coolers. Also most smart coolers, if not none, use compressors to cool and easily wire to the microcontroller that works with a Bluetooth application. If this would be used, a section on the side would hold the compressor that cools the remaining area.

The issues that come with this technology are price and space. A compressor that would suit a big cooler would take ¼ the space of the cooler, and most compressors range in the 100\$+ ranges. With the space being taken, there would be less area for smart features to be implemented. Another issue is the technology itself. Compressors have moving parts, and any cheap compressor would be difficult to be titled or rotated. This means less portability.

Additionally, compressor units come with the difficulty of maintenance in order to operate properly throughout time. Compressor units deteriorate over time due to the friction and thermal expansion of its parts, gaskets, joints, and mechanical features. All of these require inspections, lubrications, replacements, and possibly alterations. Not to mention the coolant liquid will eventually need to be replaced over a certain period of time, and also there is a possibility of leaking due to factors mentioned above.

Ice

Due to the features considered on this cooler, ice was considered the simplest yet effective method. By keeping the cooling methods quite straightforward, it allows us to direct more time and energy towards the smart features of the cooler. Since our goal is to create a smart cooler with Bluetooth and voice activated features, choosing ice over the other cooling technologies makes this more productive. Limiting the usage of large power drawing electronics greatly reduces the weight of the cooler in its entirety, allowing for more storage space inside and makes it easier to transport. In doing so, we are also able to downscale the size of the battery. This becomes safer in turn for both our team and consumers as it lessens the voltage and current being drawn throughout our entire electrical circuit. Minimizing risks is important, as we want to prevent any possible malfunctions that could interfere severely with the cooler. Fluid leakage from the compressor could occur which would be detrimental to the project. Any leaks from rain/drinks inside that get into those large cooling components would be a big issue as well. Taking all of these factors into consideration, sometimes choosing the simplest option is actually the best option.

Commented [9]: def can expand more on this and talk about issues with actually implementing the compressor

ex.

1. large power draw / wattage means bigger battery and other factors
2. use of refrigerant fluid and hazards
3. placing the coils within the cooler wall and how we need to keep it sealed. ensure no leaks get into those electrical components
4. needs adequate ventilation and a fan to make sure internal components/cooler does not get too hot and cancel out the cooling it does

3.1.3 Light Options

Having lights on the cooler can go in multiple directions based on what we can accomplish. For simplicity's sake it could just be lights to stick in the cooler, and more advanced such as music reactive LED lights. Below will describe the multiple options available and what we will pursue throughout this project. Below is *Table 4*, where it mentions the options that can be employed.

Method	Power Consumption Comparison	Coding	Features Implemented	Time to Implement*	Price
Non Music Reactive	Least	0- less than 100 lines of code	1 Feature	1-2 Weeks	<\$20
Non Music Reactive, Light Show	Moderate	100-1000+ lines of code	2 Features	2-3 Weeks	\$10-\$30
Music reactive	Most	100-300 lines of code	3 Features	2-3 Weeks	\$20-\$40

Table 4: Light Options

Non Music Reactive

This option is the most limiting and simple option. All that would be required are lights that are waterproof, can be wired to a rechargeable battery, and have a simple way to turn on and off. An example of simple lights can be shown through Water Changing Color LED Lights. These lights are simple to place onto the cooler, and have simple wiring to configure easily onto the cooler. This option would make the Bluetooth application just press it to be one and off, or just cycle through a few colors if the lights allow it.

Non Music Reactive, Light Show

The goal for the lights on the cooler is that through the phone, the user can change the lights pattern and color. However, the extent to how much a user can change the lights depends on 2 factors, coding and extra hardware. If there is no music and there are just patterns, then all it would require is a LED light band and extensive code for each pattern. There is also music reactive, which very little coding needs to be done but would need a sound sensor, and an equalizer/ relay if they do not have a light controller box.

For lights that just change color, there would just need to be lights that are LED and address addressable. Then when connected to the microcontroller, there are multiple libraries (Fastled, Xlights, etc). That will allow the programmer to change the color on any of the specific lights. An example can be found in the video “How I made a simple Christmas light show with an Arduino” . As the video states, he used multiple lights (12V 20A power supply), so taking that in consideration, and with simple code, he had 5 light patterns in a loop that cycled through. If just

changing it to a button to change the cycle, implementation would be easy. However, this way would not allow it to react to music, but it can be customizable to however complex the light patterns we want it to be.

Music Reactive

On the other hand, having lights react to music would require a sound sensor, more circuitry, but easier coding. Instead of making lights that we would have to program the patterns, it can be automatically done by splitting the lights into 3-7 bands based on the parts (base, tenor, exc). Then certain lights will turn on and off reacting to the volume of that part. A simple example can be shown in this video here [How To Make DIY Music Reactive RGB LED Strip \(WS2812B\) - YouTube](#). This can be done either by getting one set of lights for each part, or getting a relay or equalizer to split different lights. As shown here [DIY Automatic Musical Christmas Light Show \(Bluetooth + MSGEQ7 + Arduino\) - YouTube](#) he split into 7 parts and using a MSGEQ7, gave a light effect when connected to Bluetooth music.

3.1.4 Voice Communication Options

As our project has an emphasis on being a smart cooler with a Bluetooth application, we want to implement a feature where the user can talk into the phone and get a response. As this is locally from their phone, it can't use the internet and must be native to the application. If it has access to the internet, then there might be personal information sharing issues, which we do not want the user to worry about.

Talking to the application is a popular subject, and can be easily done using multiple libraries using either the Arduino IDE or other vendor offerings. The main topic will be the application speaking back to the user. There is Alexa and Siri, but those have real people being used to say multiple responses. What only has been recently popularized is the use of AI and AI chatbots. Having these can allow users to make their own voice to answer questions or make a voice themselves. The team wants to explore these options, and we are focusing our attention on either creating our own AI voice, or using Commercial AI voices. Below is *Table 5*, where it mentions the options that can be employed.

Method	Time to Develop	Customizability	Coding
AI Voice	2 Weeks- 1+Months	Limitless	1000+ Lines of code
Commercial Options	Less than a week	Little to none, as developed not by us	<100 Lines of Code

Table 5: AI Voice Options

AI Voice

As of late 2022, AI generated voices have been popularized in the media. Unlike synthesized voices, where they try to make computer noises to emulate human speech, Ai voices are trained on real vocal data. The person would feed in some voice samples into a computer program, and

then a LLM algorithm will use weights to create a voice model. Then whenever the program outputs a sentence to the model the model would be able to accurately say the words. There are many tools that can do this such as Tortoise and Piper, and many others which are still being developed to this day.

The only problem with using AI voices is making a program that can accurately sample and use the models. Creating and finding voices are simple, as a user would just need to find them on the Hugging Face website or use piper to sample our own voice to a model. The program would need to know how to read the voices, run the weights through a LLM algorithm, and know how to feed out accurate responses. There are many tools that have the capabilities to use these, such as whisper for Raspberry Pi's and Arduinos voice libraries, but those either need a sufficient amount of RAM or a subscription.

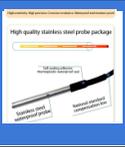
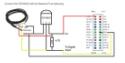
Commercial Options

Another option for voices is using voices in the market that are easily usable on most embedded platforms. These options consist of Google assistant, Dicio and Hound. These are models that have already been optimized by LLM tools and are in a form that most tools can read. An example are both the MIT application creator Google Assistant Plugin and the Arduino voice library. These plus other examples across the internet have used these tools. This option sacrifices customizability for portability. As it already has the voice trained and modeled, very little RAM is required, allowing it to work on multiple systems. However they are pretrained, so the actual voices have little to no customization.

3.2 Part Comparison and Selection

3.2.1 Temperature Sensor Comparisons

For the cooler one of our main components is temperature sensing of the inside cooler. We need this internal data so the user can be sure that the internal temperature of the cooler is desired. Below is *Table 6*, where it mentions the option we chose.

Name	Water Proofing	Rating	Material	Programmability	Cost	Picture
GikFun	Yes	4.6/5	Stainless Steel	Low	\$11.58	
HiLetgo	Yes	4.5/5	N/A	Low	\$10.99	

GikFun with pull-up board	Yes	4.4/5	N/A	Low	\$13.48	
---------------------------	-----	-------	-----	-----	---------	--

Table 6: Temperature Sensor Options

Since this system will either be inside and outside the cooler, we need a weatherized waterproof temperature sensor that is tough yet accurate. Being waterproof is the most important requirement since that will be a common component that the sensor will be interacting with. Since the sensor will be in an aqueous environment it will get wet at some point, so it will need to be able to perform in these conditions. Water is a huge killer of electronics when power is running through them and since our sensor will be active at all times when the cooler is running we need to ensure this will not occur at any point in this system. If we lose a temperature sensor we lose a massive amount of critical data that is vital to our system and cooler usage.

With this said we cannot trade durability for accuracy as it's implausible to have an inaccurate sensor. For our sensor we only want a margin of +2 or -2 degrees centigrade since we are going to be tracking and recording temperature overtime this margin of error will still allow us to be accurate and accomplish our goals. We will also be observing the temperature ranges of these products as they need to meet our ranges (H2O freezing temperature and up to 120 degrees fahrenheit) to ensure our sensor doesn't plateau at variable temperature ranges. Lastly we need a sensor that can undoubtedly be added to our cooler.

We cannot afford to have a bulky sensor as this will lead to reduced usable cooler space and make it so additional design practices need to be taken to ensure the sensor's durability. The more compact the better, in not only the size but shape of the sensor to figure out what will be the best to implement for our design.

Since we are using an Arduino based system, we need to make sure our probe can communicate with an Arduino system. With that said the chip that was most widely available was the DS18B20 "Programmable Resolution 1-Wire Digital Thermometer". This chip can monitor temps from -55 celsius to +125 celsius (-67F to 257F) which is well within our ranges for a cooler.

The DHT11 is able to record humidity and temperature, however it was not weatherized. This will be a good backup option and could possibly be used both internally and externally if weatherized. DHT11 products are not weatherized and we would need to develop and most likely 3D print weatherization and protection of those products. This is why we have chosen to go with the DS18B20 probe products since they are waterproof and there is a stainless steel cap covering the probe. This means that the probe will not rust and will be more rugged and durable to conditions inside of the cooler. This material and design type will prove effective outside the cooler especially when exposed to the salt-water environment of the beach. The probe-like nature of these devices means that we can snugly hide and secure these devices with ease. More specifically when further thinking into the development of the solar umbrella or a side/radio antenna type attachment to the outside of the cooler; The cylindrical nature of these devices should integrate easily into any construction for outdoor ambient temperature monitoring.

Gikfun DS18B20 Waterproof Digital Temperature Sensor with Adapter Module for Arduino (Pack of 3 Sets) EK1183 is our first part we will be looking at for our temperature sensor. This product is currently listed for 13.48\$ for a pack of three making it 4.49 a piece. It has an operating voltage of 3.3V to 5 volts DC which is relatively easy to achieve as these sensors have very little current draw. Our wires are very important as well as we have yellow being our data wire, red being our VCC and black is our ground, there is no clear indication of wire length which is a concern. This product is unique for 2 reasons: it comes with a dedicated pull-up board and it has a pre-built mount to connect to that connector. With the pull-up board it is a much cleaner approach than adding a 4.7kohm resistor to our circuit via soldering and then making sure that resistor is weatherized. With future thinking we believe that if possible moving our pull-up to board on our PCB and with the premade attachment it would make for a very nice plug and play and we could swap out the adapter without the need to hardwire our sensor. No other product we looked at came with a pull-up board, though one did come with the resistors, but we see this as almost a non factor since we can easily acquire the resistor at any given moment. This is not without its downsides, using this product means we can only use it without taking additional steps to make other sensors have the same connection piece, or without messy soldering to the potential PCB integration or pull-up board. With this said we don't think this will be a massive issue as we can overcome any connectivity issues, but we would like to address it here so it's not forgotten in the future and when comparing other products. Looking at the amazon store page itself we noticed a discrepancy with pictures 1 and 2 being different colored tips. We are not sure if this is just showcasing the product without the metal covering , but we have confirmed via user review that the product comes with the metal weathering tip which was our only concern. Once again we want to point this out in case it's important for further review and decision making. The product has a decent product rating of 4.4 stars out of 69 current reviews and 50+ have been bought within the last month. Upon reading these reviews we can conclude that this is a good product for our means and it comes with some useful additional features that our other products do not have. But this product is missing a lot of important product data.

Looking at this product for the price it is a steal, it's also from Gikfun and has the exact same sensor and design as our other product, but with more reliable data it seems. This product comes in at 11.58\$ for a 5 pack of sensors making them 2.32\$ a piece. They are the exact same design as the first part but include much more info. This part clearly states it is waterproof and has "rust prevention" as it is made out of stainless steel. Stainless steel, although not impervious to rust, is rust resistant and our cooler being in a saline environment will be exposed to more oxidizing materials via saltwater. This is an extremely important qualification for us as we need a resilient sensor so that we won't have to worry about water getting into the sensor from pure design or material degradation. This product provides both of those so we have no concerns regarding product durability. The wire is 100 centimeters or 1 meter long which will be plenty of wiring to work on within our cooler, and is advertised as a "self-sealing adhesive thermoplastic waterproof seal". Although this seems a little bizarre and far-fetched, it's just a simple wire covering/ protection. We have the same wire configuration as the other Gikfun product with yellow being our data wire, red being our VCC and black is our ground. Going more into detail about the sensor itself, it has an accuracy of plus/minus .5 degrees celsius between the range of -10 celsius to 85 celsius. This is within our accuracy requirements of plus/minus 2 degrees celsius, and within our

range for what temperatures our cooler will be exposed to inside and outside. Looking at the product page on amazon we have a great rating of 4.6 stars out of 436 reviews currently which is very positive with 100+ bought within the last month. These reviews are much stronger than our other Gikfun product with over 6 times the reviews it has a higher rating of .2 stars. With all that said this product is simply the better buy, even when bringing the pull-up board into the equation the 5 pack is a better price and has far better data, product page, and consumer reviews. Below is *Figure 4*, it is the HiLetsgo Sensor that is preferred in our project.



Figure 4: HiLetsgo Sensor

HiLetgo 5pcs DS18B20 Temperature Sensor Temperature Probe Stainless Steel Package Waterproof 1M is the next product we will be considering. This and our previous product are nearly identical in all areas. Coming in at a 10.99\$ (2.20\$ each), there is a price difference of 12 cents per sensor, as they both come in a packet of 5. This product is rated 4.5 stars out of 5, .1 stars less than our previous product with 255 reviews. This product however provides an extremely helpful wiring diagram and recommends using a 4.7k ohm resistor between the data wire and ground to improve accuracy. However we find it almost comical that the product doesn't seem to include 4.7k ohm resistors like our other product. Without repeating the same information twice we believe that this is an acceptable product to use on our product and at a very reasonable price and there is nothing holding us back from buying it if needed.

Outcome

We decided to go with Gikfun DS18B20 Temperature Sensor Waterproof Digital Thermal Probe Sensor for Arduino (Pack of 5pcs) EK1083, this is the best value for the money in our eyes and with 5 sensors we can do more robust testing.

3.2.2 LED Lights Comparisons

Our goal is to have the application be able to configure lights on the outside of the cooler. For the time being, it is still being decided if the lights will be reactive to music or not, however we do want to have some sort of light show to be configured on them. For the ones found, most if not all of them would be able to be configured with a device connected to the Arduino IDE and use libraries specifically for light shows that can address each address on the leds. For the constraints,

the three main factors are price, ease of coding, and power consumption. Below in *Table 7* are the options and which one the team chose.

Name	Water Proofing	Current Rating	Voltage Rating	Programmability	Cost	Picture
Alitove	Yes	900mA (Max)	5V	High	\$16.99	
Odlamp	Yes	60mA (Op.)	5V	High	\$13.99	
Daybetter	Yes	5A	12V	Medium	\$19.99	
Water Changing LED Lights	Yes	800-1200mA	N/A	Low	\$21.98	

Table 7: LED Light Options

For price, there is a little leeway as addressable lights tend to be expensive, but instead of making them ourselves buying ones that work just as needed will suffice. There are plenty of lights like this on the market, but finding one with the cheapest cost will help put our funds towards other necessary areas. The other thing necessary is the ease of coding. Due to the nature of being mostly electrical engineers, we are looking for lights which already have a considerable community and multiple libraries to easily configure the lights. It's understandable that it will take time to make a good light show, but it would take longer to try to understand the electronics intricacies and then get into the actual program. The final issue is power consumption. More than the Microcontroller it is expected that this will take a lot of power, but finding ones that will lead to less waste would be best to help save the battery life of the rechargeable portable battery.

Figure 5, as shown below, is the ALITOVE WS2811 Addressable LED Pixel Light 50pcs 5V. This option is one of the most popular devices for running light shows music reactive and non music reactive. This device is already owned by one of the developers, and has very simple set up and implementation for coding. The only issue with this is the need for a level shifter/ sound equalizer chip to work in tandem with the microcontroller, lights and speaker to actually implement the features. However, if just for a light show this would be a safe option.



Figure 5: Alitove 50pc lights

The next option is the Odlamp 50pcs WS2811 Dream Color Changing RGB Addressable LED Pixel String Light Waterproof 12mm DC5V for Christmas Party Advertising Board Decoration (DC5V 50pcs) . This device is similar to the ALITOVE lights with similar capabilities, but uses less power and costs less than the competitor. The only issue would be the less documentation on this device, which will cause more issues for the coding department. However, with how similar these devices are the coding shouldnt be an issue as it should translate easily from one device to the other.

Figure 6, as shown below, is the DAYBETTER Led Strip Lights 16.4ft Waterproof Color Changing Led Lights with Remote Controller. This option is a combination of easy coding, but has an issue with power consumption, price and lack of flexibility. There are about 150 LED controllable lights which come with a prebuilt controller, is waterproof, and has preprogrammed colors to go with it. The voltage level of this is 12V and the price is \$19.99. If this option was chosen it would be difficult to reconfigure the pattern already programmed into it. This would be chosen if we still want to do lights, risk the power consumption, but want an easy from the box light show.



Figure 6: DayBetter Led strips

Figure 7, as shown below, is the Water Changing Color LED Lights. This option has the easiest coding configuration out of the other options. These are easily stuck on lights that can change colors from a prebuilt program in the device and can be powered by self contained batteries. This should be simple to implement by taking out the batteries, hooking it up to the microcontroller, and remapping the settings to the phone application. The only issue is that it would be difficult to add our own code. This option would be chosen not for the price, but for the ease of use option.



Figure 7: Water Changing Color LED Lights

Outcome

For the LED lights, our main goal is to be able to program them with special patterns, music or no music. As such, we need lights flexible enough to be programmable without music interactions while having the capabilities for it. Price is also a key factor, and after putting them under consideration the one that was chosen was the Alitove lights. There is an expansive library with multiple light show programs to experiment with, along with plenty of documentation from the developers. The final nail was that these lights were already bought by a team member, and as they have experience with those lights, those were chosen.

3.2.3 Speaker Comparisons

Choosing a speaker comes down to two factors: the protection on the speaker itself and the power handling (wattage) along with the decibels it is able to produce. The speaker's use is important because it provides ease of use to the consumer when controlling its features since the speaker provides sound output that is tied to the application and voice commands. Below in *Table 8* are the options and which one the team chose.

Name	Operating Voltage (V)	Speaker Ohm (Ω)	Speaker Wattage (W)	Cost	Picture
Degraw	5	4	3	\$11.99	
Kibbeh	n/a	8	3	\$8.99	
Hzipowen	n/a	8	10	\$12.99	
Acxico (module)	3-12V, Optimally $\geq 5V$	n/a	5-10W, Optimally 8W	\$8.99	

Table 8: Speaker Options

First, we want to ensure that the speaker is protected and weatherized. These speakers will mainly be used outside wherever the cooler goes, and these speakers will be exposed to many different environmental elements. These include very hot temperatures, rain, humidity, and UV rays. We need to ensure that our speakers are well equipped for wherever we want to bring our Smart Cooler. We must choose ones that have an enclosure or some sort of protection around it, since the diaphragm of the speaker is sensitive and can easily be broken when accidentally grabbed or touched too hard. In addition, the speakers must be waterproof or water resistant to protect against rain and drink spills. For this, IPX5 speakers and above will be enough to withstand. Since the speakers will also be in the sun, we want to make sure our speakers can handle heat and UV rays. Heat can affect the voice coil within the speaker and can even blow if hot enough. It may damage the magnets and other delicate electronics inside, and even the plastic casing around it may melt. To combat this, speaker covers would provide ample dirt, sand, and water protection as well as UV protection but it must be paired with a sufficiently weatherized speaker for the best coverage against elements. We must also be able to weatherize a standard non weatherized speaker ourselves if it is hard to find a specific part that includes this.

Secondly, power handling of the speaker is related to the decibels it can produce. The wattage rating shows the maximum amount of power the speaker can withstand without being damaged. Decibels are a measure of the sound the speaker can produce when driven by a specified amount of power. We need to make sure that the speaker is loud enough to be heard in an outdoor setting, since it may be playing music or giving application feedback to the user. Knowing this, the speaker we choose must be loud enough to project sound even when people are talking around it. A detailed diagram is shown in *Figure 8*.

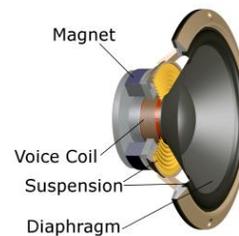


Figure 8: Speaker Part Diagram

Another thing to note for speaker selection is the difference between active and passive speakers. Active speakers have an amplifier within its housing and a speaker as well. This requires AC power. Passive speakers require a separate amplifier because it will not work correctly without amplification. The AC current is an amplified audio signal. If DC current is fed through the speaker, it will break the voice coil of the speaker drivers. AC current has waves that will bring it high and low, using DC power will push the speaker but have no way to bring it back. The driver will not be able to convert it to anything else besides heat which will fry the coil. Because our battery is 12V DC power, we will need to use a passive speaker. This means we will need to create or attach a circuit that amplifies the signal given, and then feeds it through to the passive speaker. We will be choosing an Arduino shield with an audio amplifier circuit and passive speakers with specifications suitable for our needs. We must further test the products in person to determine the best speaker, as it is hard to choose based on the listed specifications.

Figure 9 shows the Degraw DIY Speaker Kit - PAM8403 5V Amplifier + 2Pcs 4 ohm, 3 watt Speakers. This Degraw speaker kit would work for our project as it consists of the amplifier circuit and two speakers already. This kit will be able to produce sound and connect to the Arduino. The

only issue is that without further testing we will not know if the speaker is loud enough for our intended usage.



Figure 9: Degraw DIY Speaker Kit

Figure 10 shows the Kibbeh 2" 8 ohm 3w Arduino Speaker Full Range Speakers for Arduino. This product is 2 speakers and it must be paired with the audio amplifier module. This will work for our project, we just have to test how many dB the speaker can clearly produce without sound distortion.



Figure 10: Kibbeh Speakers

Figure 11 shows the HZPOWEN 10W Mini 3" HiFi Full Range Speaker 8 Ohm. This product is 2 speakers and must also be paired with the audio amplifier module. This will work just as well as the previous speaker listed and since it is a higher wattage it means it can be driven louder than the 3W speaker without damaging any of the internal components.



Figure 11: HZPOWEN Speakers

Figure 12 shows the LM386 Audio Amplifier. This product consists of just the LM386 audio amplifier module. There is no speaker included in this item. This product would work well due to the wide operating voltage range as well as the speaker wattage ranges. This means we have a lot more options on which speaker to pair with this audio amplifier module. In addition, this module seems to amplify much more than the Degraw speaker kit, making it a more suitable choice.



Figure 12: LM386 Audio Amplifier

Outcome

We chose to use the Acxico LM386 audio amplifier shield along with the Hzpowen 8 Ohm 10W speaker pair. This combination of parts will provide the clearest sound quality and loud dB range. In addition, it is compatible with the other operating voltages and will do its job considering our budget.

3.2.4 Analog Buttons Comparisons

When deciding a button to be chosen there are only a few key features that are noteworthy. They are its implementation, cost, and protection from water. This is because a button is not something that can be revolutionized very much since it is a technological device that provides a system with an input of either ON or OFF. There are buttons out there that do serve more than those two functions, however for the application on our smart cooler. We only need a button that performs the simple task of sending a signal to our MCU, and then cuts off the signal. Below in *Table 9* are the options and which one the team chose.

Name	Operating Voltage	Current Rating	Waterproof	Cost	Picture
Daoki Miniature Button	250V	50 mA	NO	\$0.05/Count	
SPST-NO Standard Panel Mount	125V	16A	NO	\$1.32/Count	
Momentary Push Button Switch	220V	3A	YES	\$7.89/Count	

DaierTek ON/OFF Round Rocker	12V	6A	MAYBE	\$1.40/Count	
------------------------------------	-----	----	-------	--------------	--

Table 9: Switch Options

Having considered these basic required features and functionalities of our button/switch, we have narrowed down our buying options to the four buttons/switch shown above in *Table 9*. These four vary in price, durability, build quality, and features, which allows us as a group to make the best purchase decision possible after comparing and contrasting all four of these buttons and switches.

When comparing and contrasting the four options above, the first thing we looked at is their operating voltage. The output voltage is typically defined as a range of input voltages for a device that guarantees the specified output voltage. Therefore, the wider the range the button has as an operation voltage, the more diversified its purpose and application become – which is a benefit.

Next, we considered the current rating of each button or switch. The current rating is the maximum current a load can withstand for a specified duration of time before the load deteriorates. Therefore, searching for a button or switch with a higher current rating allows us the leniency and cushion in the unlikely chance we make a miscalculation or plug the wrong wire into the wrong spot because there is a higher chance that the current we apply to our system will be below the current rating of the button or switch.

Furthermore, we have evaluated the build quality of the switches, which to us, are crucial for ensuring our purchased device can go through possible mishandling in our non-ideal conditions. Thus, factors such as materials, visual strength, and user reviews informed our decision-making process. While not requiring heavily machined build materials, the button or switch we choose must withstand multiple presses, accidental impacts, and be properly secured to our smart cooler for reliable accessibility during testing or transport.

The button or switch we have will have to withstand water, whether light rain, splashing, or submersion is the cause of water exposure. The button or switch will also need the ability to withstand dust/debris. These are important factors because these buttons will be implemented onto a smart cooler that is expected to travel with the user to locations that will inevitably be exposed in some way or another to the elements. For example, taking a cooler to the beach is common, at the beach there is a lot of water, sand, salt (Which is corrosive) and tons of miscellaneous debris like sweat, food, etc. When deciding on which button to choose, we need to ensure that it has some level of protection from these elemental concerns so that their overall reliability increases.

The last major focus that determines whether or not we chose a button was its cost. This cost depends on things such as overall package cost, or per unit cost, which are two different things so we took them both into consideration. Overall package cost is what we describe as the cost for the total online listing even with shipping included. This overall package cost also takes into account if the package delivers with more than one button/switch unit. For example, below the first button on our list comes in a total package of 100 units. Which is definitely a benefit because it means we

have more buttons to use to implement onto our cooler, and also it means if we make a mistake and break our button/switch, then we can easily replace the broken part instead of needing to purchase a replacement and waiting for it to arrive which is costly to our overall schedule. Therefore, to simplify things, we also took into account the per unit cost or cost “per count”. We did this to get a better idea of the individual cost of the single button/switch to help make easier comparisons between all of our purchase options. Ultimately though, we looked at both costs because they both have their own benefits and drawbacks to use when making comparisons.

Figure 13 shows the Daoki Miniature Momentary Tactile Push Button (100Pcs). The Daoki miniature momentary tactile push buttons are the perfect example of simplicity meets effectiveness. These tiny analog buttons only have one feature and that is to be pressed down to send a signal onward to be interpreted by a computer. This device gives us the perfect option for customization without much complication. This purchase also comes with 100 pieces which allows us to have up to 100 distinct assignments for each button if necessary, but more importantly for our case, it provides us the confidence that if a component breaks that we will have a backup on hand and will not need to make another purchase.



Figure 13: Miniature Push Button

One of the biggest issues with these buttons, however, is their implementation onto our embedded system or PCB. That is, these buttons are completely exposed to the elements and are used for simple cases in classrooms or at-home labs in the garage where elemental concern is nonexistent. If for example our cooler is at the beach there are issues with sand, salt water, drinks, and heat that can come in contact with our button and jam it up, or short circuit the pins and/or board altogether. We could design a work around by encasing these buttons in a rubber mold to completely protect it from all angles but that would be extra cost, design, and time that may not be worth the effort if there are better options.

Figure 14 shows the Pushbutton Switch SPST-NO Standard Panel Mount, Snap-In - Part # PR144C1900. The PRC144C1900 is a small simple device that performs the same functionality as the Daoki miniature momentary tactile push buttons, however, it is only sold as a single unit. This is because it is a bigger product overall and therefore more durable and possibly even reliable. Instead of having pins for a breadboard style system to connect to devices like the Daoki miniature momentary tactile push buttons, the PRC1441900 has plates with hole cut-outs that can allow for wire-wrapping, or plugging into a proprietary port to connect the plates to a device. Since this listing only ships with one unit, if the unit breaks or gets lost it will be tedious to replace due to the manufacturer's turn around time of 11 weeks so it might be advisable to make an initial purchase of multiple units. Additionally, if multiple functions are required in the overall project, more units will need to be purchased.



Figure 14: Pushbutton Switch SPST-NO

This button has the benefit of having a greater separation between the electrical pins and the physical button which can increase its feasibility in a smart cooler that is going to presumably be exposed to the elements when used by the consumer. For example, the button can be placed in rubber o-ring attached to the side of the cooler and the electrical pins will theoretically be protected from the elements. Now the only concern left would be sand or liquid accessing the electrical components from the physical button itself on the outside. This can be easily prevented by attaching another o-ring on the inside of the button unit housing, however this poses a threat of breaking the entire button if we are not careful when disassembling and reassembling the device.

Figure 15 shows the Momentary Push Button Switch. The switch performs the same as the previous two listings in terms of its basic functionality. However, some distinct characteristics that set it apart from the rest of the field are its build quality (fitted with metal components and rubber gaskets), and IP67 water resistivity. This push button device is described on the purchase page as being splash resistant which is a key feature to have for a cooler that comes in contact with liquids a lot. This push button connects to a device by using its two screw terminals to attach and secure wires into place.



Figure 15: Momentary Push Button Switch

This button is by far the best choice for our specific use case on the cooler device - turning the system on and off. It is built very sturdily, it comes with rubber o-rings on the outside for the actual button mechanism, and the ability to screw the button into position gives us the option to coat the threads with liquid rubber sealant like Loctite or WaterWeld, or use any generic thread tape in order to completely seal off our electrical components from the elements.

Figure 16 shows the DaierTek On/Off Round Rocker is a very simple button that allows us to designate our electrical system into two modes - on or off. This button will give the user of our device (and ourselves while building) the visual representation of the system's mode without having to guess or remember the last time they clicked the button to determine if the system is on or off.



Figure 16: DaierTek ON Off Round Rocker

This button has very similar benefits as the button mentioned above in regard to its protection from the elements. The button can easily be protected by some rubber sealant. Additionally, this button comes with wires attached that are sealed in silicon to protect them from elements so that we can then route them away from any possible elemental exposure. Therefore, the only real difference between this button and the previous is their functionality. This button displays the current output of itself to the system because it is a flip switch that is labeled unlike the previous button that is a push-button. This only requires a minor adjustment in the code to properly integrate into our cooler system. Also, the switch can be programmed to two specific modes needed for any application as opposed to being confined to ON and OFF functionality. This allows us as a team to have more flexibility with the switch and possibly even use multiple switches in a sequence that performs some action while being able to keep track of every position easily since it is a switch and not a button.

Outcome

After much deliberation and careful consideration between all possible buttons and switches, we concluded to make a purchase of two devices because we feel these two will be capable of fulfilling all of our needs and can complement each other. We have decided to move forward with the DaierTek ON/OFF Round Rocker, and the Momentary Push Button Switch.

3.2.5 Keypad Buttons Comparisons

When deciding a keypad to be chosen there are only a few key features that are noteworthy. They are its implication, cost, and protection from water. This is because keypads are very simple to implement and design functions from – typically used to provide an input value to be stored onto an embedded system that can then be processed and provide an output. Therefore, it is difficult to revolutionize the function of a keypad. There are keypads out there that do serve more than those functions, however for the application onto our smart cooler, we only need a keypad that performs the simple task of sending a signal to our computer at the press of a button in order for its value to be stored and processed through an algorithm or calculator. Below in *Table 10* are the options and which one the team chose.

Name	Operating Voltage	Current Rating	Waterproof	Cost	Picture
------	-------------------	----------------	------------	------	---------

Tegg 1PC 3x4 Keypad	Unlisted	Unlisted	Splash	\$9.99/Count	
12 Key Membrane Switch Keypad	35V	100mA	YES	\$3.49/Count	
Songhe 4 x 4 Matrix Array Membrane Switch Keypad	35V	100mA	YES	\$1.40/Count	
Digilent PmodKYPD - 16-button Keypad	Unlisted	Unlisted	NO	\$14.99/Count	

Table 10: Button Options

Having considered these basic required features and functionalities of our keypads, we have narrowed down our buying options to the four keypads shown above. These four vary in price, durability, build quality, and features, which allows us as a group to make the best purchase decision possible after comparing and contrasting all four of these keypads.

When comparing and contrasting the four options below, the first thing we looked at is their Operating Voltage. The output voltage is typically defined as a range of input voltages for a device that guarantees the specified output voltage. Therefore, the wider the range the keypad has as an operation voltage, the more diversified its purpose and application become - which is a benefit.

Subsequently, we evaluated the “current rating” for each keypad, which indicates the maximum current a load can endure within a specified duration (usually instantaneous), before experiencing deterioration. Thus, opting for a keypad with a higher current rating provides a suitable margin of error for our needs, which can offer us some flexibility in the case of possible miscalculations or unintentional connection errors. This is advantageous to our project because the likelihood of the applied current exceeding the keypad's “current rating” is low, which will enhance the overall safety and reliability of our system.

Next, we considered the build quality and durability of the keypads because we need to know that the device we purchase is capable of withstanding improper user handling at times (due to us not working in confined laboratories). Simple things such as build materials, visual strength and durability, and user reviews were contributing factors to our decision making process. Keypads do not need to be made of bullet proof material, however, they do need to be able to withstand multiple buttons presses, they need to be able to withstand the constant banging and hits from outside sources on accident, and they need to be able to be secured to our cooler in a properly fastened manner so that they are reliably accessible at all times (not falling off our PCB in the middle of testing or transporting).

A subsection of the paragraph above is the keypad's ability to withstand water, whether light rain, splashing, or submersion is the cause of water exposure AND the keypad's ability to withstand dust/debris. These are important factors because these keypads will be implemented onto a smart cooler that is expected to travel with the customer to locations that will inevitably be exposed in some way or another to the elements. For example, taking a cooler to the beach is common, and at the beach there is tons of water, tons of sand, tons of salt (Which is corrosive) and tons of miscellaneous debris like sweat, food, etc. When deciding on which keypad to choose, we need to ensure that it has SOME level of protection from these elemental concerns so that their overall reliability increases.

The last major focus that determines whether or not we choose a keypad was its cost. This cost depends on things such as overall package cost, or per unit cost, which are two different things so we took them both into consideration. Overall package cost is what we describe as the cost for the total online listing even with shipping included. This overall package cost also takes into account if the package delivers with more than one keypad unit. For example, below the first keypad on our list comes in a total package of 2 units. Which is definitely a benefit because it means we have more keypads to use to implement onto our cooler, and also it means if we make a mistake and break our keypad, then we can easily replace the broken part instead of needing to purchase a replacement and waiting for it to arrive which is costly to our overall schedule. Therefore, to simplify things, we also took into account the per unit cost or cost "per count". We did this to get a better idea of the individual cost of a single keypad to help make easier comparisons between all of our purchase options. Ultimately though, we looked at both costs because they both have their own benefits and drawbacks to use when making comparisons.

Figure 17 shows the Tegg 3x4 Keypad is a simple 12 button matrix keypad that can be easily paired with an Arduino Raspberry Pi to allow our cooler to receive inputs used for multiple functions. This specific keypad is not water resistant unless the board itself is protected. The buttons themselves are protected by a single layer or silicon matting so there is no way for water or dust to get into the buttons. This device needs to be coded and mapped by the consumer, however, there is no data sheet nor wiring/pinout mapping provided by the manufacturer. This will require us as a team to use trial and error to map out the correct buttons to the pins ourselves.



Figure 17: Tegg 1PC 3x4 Keypad

The biggest benefit with this specific keypad is the rubber mold encasement surrounding the actual board and buttons below because we need to keep our electronics away from possible elemental danger exposed to the cooler. The pins shown in the picture can easily be tucked behind a rubber seal placed into the cooler, so overall, this device is a great solution to a possible concern we have with our electronics. Additionally, the keypad encompasses all of the basic features we need without being too overly complicated to enact.

Figure 18 shows the 12 Key membrane switch keypad. It is a very simply designed “tape” style keypad. It is flat and rubber plated to protect the buttons and wiring underneath. It has 12 buttons just like the previous device listed. It is unsure if this device ships with any diagrams to support us as we determine the proper pinout and button configuration but since it is a device purchasable on Amazon, it can easily be returned if the device does not meet our expectations. This device can easily pair with an Arduino device if needed.



Figure 18: 12 Key Membrane Switch Keypad

Figure 19 shows the Songhe 4x4 matrix array membrane keypad is nearly identical to the device listed above however with the added functionality of 16 keys instead of 12. This ultimately gives us as a team the opportunity for more creative solutions to problems that may arise that are now unseen. Similar to the previous purchase listing, this purchase listing does not specify if the manufacturer provides any diagrams or schematics of how to configure the pinout with the buttons. Fortunately, this purchase listing comes with 5 pieces and has more available buttons for use which is overall a more cost effective decision. This device pairs with an Arduino easily if needed.



Figure 19: Songhe 4 x 4 Matrix Array Membrane Switch Keypad

The previous two keypads are very similar in nature in regard to their components, safety, elemental protection, etc. However, one key difference is the number of usable buttons; from looking at the photos, there are obviously more buttons on the second keypad as opposed to the first which can allow us to implement more possible features into our cooler if needed. It is always better to have more than not enough in this situation.

Figure 20 shows the digilent PmodKYPD is an amalgamation of all of the previous purchase listings in some way or another. It has the same silicon matted keypad design as the Tegg 3x4 Keypad but it has the added benefit of 16 total accessible keys similar to the Songhe 4x4 matrix array membrane keypad. Unlike the previous two purchase listings, this device can register multiple inputs at once. This device is also easily mounted on a PCB if needed. Favorably, the digilent PmodKYPD has provided a full manual complete with a library and example code for us to use as guidance to configure the device to our needs for the cooler.



Figure 20: Digilent PmodKYPD - 16-button Keypad

Although the easiest to implement with software, the most durably built, and the most mainstreamed, the biggest issue with this keypad device is its protection (or lack thereof) from potential elemental dangers that may be present if integrated onto our cooler. It would require a lot of design and physical work to integrate this keypad onto our cooler and completely seal it off. It will also be protected from the most common elements our cooler will be potentially in contact with.

Outcome

After much deliberation and careful consideration between all possible keypads, we concluded to make a purchase of just one device because we feel the Songhe 4 x 4 Matrix Array Membrane Switch Keypad will be capable of fulfilling all of our needs at a great price. We have decided to move forward with this keypad because it is mostly water proof, has the most possible buttons, is

tied with the easiest implementation onto our computers, and is durable/malleable enough to easily attach to our smart cooler.

3.2.6 Electronic Locks Comparisons

When deciding a locking mechanism to be chosen there are only a few key features that are noteworthy. They are its implication, cost, and protection from water. This is because locking mechanisms are very simple to implement and design functions from - typically used to receive an input voltage and perform a locking/unlocking action as the “output”. Therefore, it is difficult to revolutionize the function of a locking mechanism. There are locking mechanisms out there that do serve more than those functions, however for the application onto our smart cooler, we only need a locking mechanism that performs the simple task of receiving a signal and converting that voltage into a mechanical lock to prevent anybody from entering our smart cooler unwarranted. Below in *Table 11* are the options and which one the team chose.

Name	Operating Voltage	Current Rating	Waterproof	Cost	Picture
Rotating Door Lock DC Motor	5V	400mA	NO	\$2.50/Count	
QWORK Electromagnetic Solenoid Lock (4Pc)	12V	350mA	Splash	\$3.99/Count	
Metal Fail Secure Mode Multifunctional Mini Electric Cabinet Lock	12V	100mA	NO	\$30.06/Count	
TOPLEE Electronic Cabinet Lock	12V	2A	NO	\$13.99	

Table 11: Lock Options

Having considered these basic required features and functionalities of our locking mechanisms, we have narrowed down our buying options to the four locking mechanisms shown above. These four vary in price, durability, build quality, and features, which allows us as a group to make the best purchase decision possible after comparing and contrasting all four of these locking mechanisms.

When comparing and contrasting the four options below, the first thing we looked at is their operating voltage. The output voltage is typically defined as a range of input voltages for a device that guarantees the specified output voltage. Therefore, the wider the range the locking mechanism has as an operation voltage, the more diversified its purpose and application become.

Following that, we assessed the "current rating" of each locking mechanism, denoting the maximum current endurance within a specified duration before deterioration occurs. Choosing a locking mechanism with a higher current rating affords a beneficial cushion, allowing flexibility in the event of miscalculations or inadvertent connection errors. This proves advantageous to our team, as the probability of surpassing the keypad's "current rating" with the applied current is minimal, which in turn bolsters the safety and reliability of our system as a whole.

Following that, we deliberated between the different standards for the build quality and durability of the locking mechanisms, deciding which is essential to guarantee our purchased device's resilience amid occasional improper handling in less-than-ideal conditions. Decision-making factors included materials, visual strength, and user reviews. While not necessitating unbreakable materials and manufacturing processes, the buttons and switches must endure multiple multiple locks and unlocks, they need to be able to withstand the constant banging and hits from outside sources on accident (or on purpose), and they need to be able to be secured to our cooler in a properly fastened manner so that they are reliably accessible at all times (not falling off our PCB in the middle of testing or transporting, or falling off the cooler).

A subsection of the paragraph above is the locking mechanism's ability to withstand water, whether light rain, splashing, or submersion is the cause of water exposure and the locking mechanism's ability to withstand dust/debris. These are important factors because these locking mechanisms will be implemented onto a smart cooler that is expected to travel with the customer to locations that will inevitably be exposed in some way or another to the elements. For example, taking a cooler to the beach is common, and at the beach there is tons of water, tons of sand, tons of salt (Which is corrosive) and tons of miscellaneous debris like sweat, food, etc. When deciding on which locking mechanism to choose, we need to ensure that it has some level of protection from these elemental concerns so that their overall reliability increases. However, due to use most likely implementing the entire locking mechanism on the inside of the cooler, we can protect it more from the elements and therefore the waterproof factor is less of a concern but not completely negated.

The last major focus that determines whether or not we choose a locking mechanism was its cost. This cost depends on things such as overall package cost, or per unit cost, which are two different things so we took them both into consideration. Overall package cost is what we describe as the cost for the total online listing even with shipping included. This overall package cost also takes into account if the package delivers with more than one locking mechanism unit. For example, below the second locking mechanism on our list comes in a total package of 4 units. Which is definitely a benefit because it means we have more locking mechanisms to use to implement onto our cooler, and also it means if we make a mistake and break our locking mechanism, then we can easily replace the broken part instead of needing to purchase a replacement and waiting for it to arrive which is costly to our overall schedule. Therefore, to simplify things, we also took into

account the per unit cost or cost “per count”. We did this to get a better idea of the individual cost of a single locking mechanism to help make easier comparisons between all of our purchase options. Ultimately though, we looked at both costs because they both have their own benefits and drawbacks to use when making comparisons.

Figure 21, shown below, is a Rotating Door Lock DC Motor. This device is a very simple DC motor that runs on very low power to rotate the axis by 90 degrees. The one thing to note is that it takes a long time to rotate the complete 90 degrees (roughly 5 seconds), which means to lock and unlock the cooler, we will need to allocate a total of roughly 10 seconds into the routine. Additionally, this device is only a motor and not a complete locking system. Meaning we as a team will need to attach some device to the small rectangular indent shown in the picture above in order for the complete locking mechanism we desire to be achieved. An example of this is attaching a small metal rod in the rectangular slot made by us which will complete the locking and unlocking mechanism.

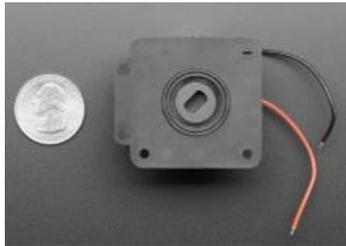


Figure 21: Rotating Door Lock DC Motor

Figure 22 is the QWORK Electromagnetic solenoid lock shown above is the perfect simple device to perform our desired lid locking goal. It is a complete package device that requires only 12 volts of direct current power in order to fulfill the locking and unlocking performance. Its small stature and mounting brackets allow for easy implementation into a lid. The only thing the team will need to provide is the opposite bracket that the solenoid device locks into. This will provide a unique creative challenge for the team.



Figure 22: QWORK Electromagnetic Solenoid Lock (4Pc)

Building off of the previous purchase listing of the solenoid lock, *Figure 23* is the Metal Fail Secure Mode Multifunctional Mini Electric Cabinet Lock. It provides the complete package of a locking mechanism for our needs but at the cost of \$30 per unit. This device uses a solenoid system as well through electromagnetic fields. Out of all the purchase listings so far, this device is the

most durable and has the most heavy duty locking capabilities, but an issue with this specific device is the seemingly lack of quality control between units as many purchase reviewers consistently state that the device is very inconsistent and/or fails to work after so many uses.



Figure 23: Metal Fail Secure Mode Multifunctional Mini Electric Cabinet Lock

Many reviews online mention instability in its performance. However, when it does work, it works exactly as promised from the manufacturer. This is something the team needs to consider when making a purchase decision because we need to have equipment and components that work properly at nearly all times so that the fail rate is near zero percent.

Figure 24 is the TOPLEE Electronic Cabinet Lock is a simple solenoid based cabinet locking mechanism found on Amazon as a commercially available device for home/project uses. The build quality is heavily focused on by the manufacturer because the outer shell is composed of hardened steel, while the inner mechanisms and components are carbon steel. The lock is not energized when it is in the “Lock” position, it is only after a voltage signal is applied to it that it unlocks itself. The manufacturer ensures that this lock can operate up to 500,000 individual locks and unlocks over its lifetime which is a high durability and reliability rating. The latch that comes with this purchase is able to be configured on the side (as opposed to directly in front) which allows us as a team to make a more adaptable decision down the road when building our smart cooler on how we want to design and orient the locking mechanism with the lid of the smart cooler. Nowhere in the manufacturer’s notes nor the user reviews does safety from water come into question, so either it is not an issue or it has not been tested, but for safety, we will try to separate the lock from water as much as possible.



Figure 24: TOPLEE Electronic Cabinet Lock

From user reviews we have learned that this lock distinctly requires exactly twelve volts and at least one amp in order to operate - no less. Thankfully, as a failsafe if for some reason power is unable to be provided to the locking mechanism (i.e. battery died), there is a mechanical switch on the back that can activate (unlock the mechanism) to release it from the latch.

Outcome

After much deliberation and careful consideration between all possible keypads, we concluded to make a purchase of just two devices because we feel the QWORK Electromagnetic Solenoid Lock and the TOPLEE Electronic Cabinet Lock will be capable of fulfilling all of our needs at a great price. We have decided to move forward with these locking mechanisms; they both have great build quality, reliability, simple enough implementation, and reasonable electrical requirements.

3.2.7 USB Ports Comparisons

USB Type A will be our main use for power delivery when charging mobile devices with our smart cooler. Due to the environments our cooler will be in we need to weatherize these charging ports so sand and water does not enter the USB socket. Below in Table 12 are the options and which one the team chose.

Name	Water Proofing	Current Rating	Weatherized Cover	USB TYPE	Cost	Picture
PENGLIN	Yes	500mA	Yes	Type A 2.0 Female-Female	\$14.99	
ICESPRING	Yes	900mA	No	Type A 3.0 Female-Male	\$8.50	
Duttek	Yes	500mA	No	Type A 2.0 Female-Male	\$11.58	

Table 12: USB Port Options

Figure 25, shown below, shows the PENGLIN 2Pack USB 2.0 Panel. This product is the perfect thing we needed to weatherize our external USB charging ports. This product will be paired with the male adapter chosen. Coming in at \$14.99 for a 2 pack making it \$7.50 a piece this is perfect since we decided on having only 2 USB 2.0 charging ports. This product will be mounted to the

cooler after we have drilled out the correct hole size then using that we will use the products pre-cut holes to secure it via screws to our cooler. With the water/dust proof rubber cap this will be perfect for a cooler and the environments that a cooler would be in, allowing users to cover charging ports when the cooler and charging ports are not in use. We would like to mention that this port is not strictly powered and has the ability to transfer data. The body/case holding the USB Female connector is made out of metal which will be more resistant to cracking when we mount it to our cooler. Looking at the amazon page the product only has 8 reviews with a rating of 4.2 stars, with only 2 actual written reviews. This product has been on amazon for over a year, from what we have researched it seems that these products are not very commonly bought, but across the board it seems that products from the seller PENGLIN are decently rated but not very commonly bought. With all said, we believe that this will be a decent product for our project.



Figure 25: PENGLIN 2Pack USB 2.0 Panel

Figure 26 shows the ICESPRING USB 3.0 Male to Female AUX Flush Mount Extension Cable for Car Truck Boat Motorcycle Dashboard Panel -(3 Feet - 1 Meter) is the next female connector we will be looking at. The product is currently \$8.50 and 11% discount off the original price of \$9.50. This is a little different from our PENGLIN product as it is Female to Male which should still prove useful for our application. This product is USB 3.0 which is far better than USB 2.0. USB 3.0 is able to put out 4.5watts of power while USB 2.0 can only put out 2.5 watts of power. Although USB 3.0 is far better than 2.0, we have current draw concerns in our system as it may be an issue powering both USB 3.0 ports at the same time. This product has a wire length of 1 meter and seems to be adequately protected from the elements, but covering the actual port itself may be difficult due to the way we plan to implement these to the outside of our cooler. The amazon reviews on this product are very good with a rating of 4.7 stars out of 3919 reviews. The only things holding us back from buying this product would be that it is not female to female and we may have to worry about the wire not being interchangeable, and that we don't think that we can easily cover the port without additional construction on the outside of the cooler.



Figure 26: ICESPRING USB

Our next product is very similar to our last except we decided to go with the USB 2.0 option and it has a much better mounting mechanism for our dust/water covering. *Figure 27* shows the Duttek. This product comes in at a price of \$11.85 for one. We like that although this product has a cylindrical shape it has a very simple rubber cover that can be taken on or off easily. The only downside of this product is it's a little pricey and the wire is quite short at 1 foot. Due to it being female to male we would have to extend the wire using another wire. Looking at the amazon page for this product it is well reviewed with a 4.4 rating with 112 ratings. This product would be a little interesting to work around and weatherize a wire connection but we believe it will work.



Figure 27: Duttek

Outcome: Our first product is what we will be choosing, for the price and way it is designed it is perfect for what we need, in terms of usability and construction.

Female USB Ports: For hooking up power via PCB or wherever else it will be needed. This product is important because it will be a bridge from any power source to our chargers. We just need a simple interface that we can solder the wires to our power disbursement circuit or they will be integrated into our PCB. Below in *Table 13* are the options and which one the team chose.

Name	New Materials	Rating	Voltage Rating	USB TYPE	Cost	Picture

MELIFE 10pcs	Yes	4.6/5	5V	Type A 2.0 Female	\$6.99	
Walfront 10pcs	Yes	4.4/5	5V	Type A 2.0 Female	\$5.49	
Akozon 10pcs	Yes	4.5/5	5V	Type A 2.0 Female	\$6.29	

Table 13: Female USB Port Options

Figure 28 shows the MELIFE 10pcs. This comes in at \$6.99 for ten pieces making them \$0.70 each. This is just a simple interface to connect our USB charging to our PCB or circuit that regulates the power output to the USB's. Since we are hooking up only power we don't need anything fancy we just need to connect up our power. The amazon page of this product is very good with a 4.6 star rating out of 179 reviews. We see no issues using this product.



Figure 28: MELIFE 10pcs

The next option is the Walfront 10pcs. This product comes in at \$5.49 for ten pieces making them \$.59 each. Just like our previous product this is a simple interface. The product page claims "The USB breakout board is made of 100% brand new and high quality material ensures longer use", as long as this is true this is a good thing for us. If this is made with new and not recycled materials that means that the materials they use will be more pure, as recycled materials contain a large percent of trace materials. The product has a rating of 4.4 stars out of 66 reviews which is solid. There is nothing holding us back from purchasing and using this product.

The final option is Starting with the Akozon 10pcs. price this one comes in at \$6.29 or \$0.63 each making it the middle ground of our 3 parts. Upon further inspection the product page has multiple spelling errors which is a red flag for us. With that said the product is rated quite well with a 4.5 star rating out of 71 reviews. However we do not trust a product page with incorrect spelling. We will not be purchasing this product for that reason.

Outcome

We went with MELIFE, and although it was more expensive the product page was more reliable and had better ratings when compared to the other two.

3.2.8 Rechargeable Battery Comparisons

Choosing a battery is based on multiple parameters such as the power output (W), battery capacity (Ah), type of battery, and weight (lbs). Selecting the proper battery is essential because it is delivering power to all electrical components. All sensors, circuitry, lights, speaker, USB ports, etc will draw power from this battery. We must choose the battery that takes all these factors into consideration. Below in *Table 14* are the options and which one the team chose.

Name	Voltage Rating (V)	Capacity (Ah)	Battery Type	Weight (lbs)	Dimensions(in)	Cost	Picture
Mighty Max	12	7.2 Ah	Lead Acid SLA	5V	6.25 x 2.75 x 4.25 in	\$19.99	
UPlus	12	8 Ah	Lead Acid AGM	5V	5.95 x 2.56 x 3.68 in	\$25.99	
Vatrer	12	7 Ah	Lithium Ion	12V	5.94 x 2.54 x 3.74 in	\$27.99	

Table 14: Rechargeable Battery Options

For our rechargeable battery, the optimal rated voltage we want is 12V. All components used are not drawing large amounts of power, therefore a higher rated voltage is not needed. We want a voltage that has a high compatibility with other components to make it easy to work with and integrate into our system. Because we are not using any cooling technology such as a compressor or Peltier cooling system, we are able to use a lower rated battery. None of the other components will be drawing that much power, and this extends the longevity of the Smart Cooler operating time.

After looking at the power consumption from all other electronic parts, we can give an estimate for the battery’s required capacity. The higher the ampere-hours rating is, the longer the battery is able to run. As a requirement, we want the cooler to be able to run for at least 5 hours. This means that our battery should be rated for higher than that to account for normal battery degradation and other factors that may diminish the battery’s capacity. Knowing this, our battery’s target should be around 7-8 Ah.

The type of battery also impacts our decision. The battery types we will be focusing on are lead-acid and lithium-ion. Lead acid batteries can operate within a large temperature range and can keep

up that performance even in sudden temperature changes. However, there is a specific charge and discharge cycle that can affect cycle life. Lithium-ion batteries on the other hand, are less sensitive to that cycle which makes it a more suitable option when taking solar charging into consideration. This battery is also more efficient than lead acid, which means its lifespan will be longer. This battery can completely discharge and still power efficiently vs lead acid, which will degrade more beyond a certain point. Lithium-ion batteries are more expensive than lead-acid, so we will have to weigh our options appropriately to decide which is better suited all around.

The battery's weight is also a factor that should be taken into consideration. The battery is easily the heaviest singular component inside this cooler, and heavier the battery the harder it is to bring the cooler around. An important facet of a cooler is that it should be easily portable and the entire weight cannot be over 30 lbs, so a big heavy battery conflicts with those requirements.

Another thing to note is that this battery will be recharged using the Solar Umbrella. This means that the battery will be going through quick and repeated cycles which affects its cycle life. Large discharges will decrease the battery's cycle life, and since the umbrella is charging through an uncontrollable method (solar), we have to ensure we can monitor the battery's discharge and cycle time to maintain it. This will be further discussed within the Solar Charger section.

Figure 29 shows the Mighty Max Battery. This lead acid battery meets all the baseline requirements for the rated voltage, capacity, and size. SLA stands for sealed lead acid. Sealed means the produced gasses will be contained within the battery itself, unless the battery is under high stress or pressure. This battery will require a bit more maintenance. This battery will definitely work for what we need, but there are better options to consider.



Figure 29: Mighty Max Battery m17-12 - 12 Volt 7.2ah SLA Battery

Figure 30 shows the UPLUS battery. This battery is also lead acid, but is a different type. AGM stands for absorbed glass mat. This means that inside the battery is a glass separator between the positive and negatives that absorbs the battery acid produced. This keeps the battery maintenance free. This also means that this battery is better suited for recharging than the previous SLA battery.



Figure 30: UPLUS LP12-8 12 Volt 8Ah Rechargeable AGM Battery

Figure 31 shows the Vatrer Power battery. This lithium ion battery works for all the requirements. Lithium ion is more efficient and stable than lead-acid, which already makes it a more optimal choice. In addition, it's rated for recharging and even has an additional battery management system within it. This 12A BMS will protect the battery from overcharging and deep discharging. It also cuts off charging at a temperature of 167 °F (75°C). This is also the lightest battery option, which also corresponds with our requirement of making the cooler light and portable. This battery is also IP65 rated which is water resistant.



Figure 31: Vatrer Power 12V 7AH LiFePO4 Battery Deep Cycle Lithium Battery

Outcome

For this, we have chosen to go with the lithium ion battery. It is the most efficient and long lasting battery, even taking price into consideration. In addition, it has an integrated battery management system. This is an additional feature that makes our entire project much more safe. This battery meets all requirements and even goes above what we had initially wanted.

3.2.9 Solar Panel Comparisons

Selecting a solar panel is based on three main factors: size, weight, and output wattage. Since the solar panel is to be integrated into a portable Solar Umbrella, size and weight are critical in ensuring the Solar Umbrella can both work as its intended use and still be able to fold/collapse. Below in *Table 15* are the options and which one the team chose.

Name	Water Proofing	Size	Wattage	Weight	Cost	Picture
Topsolar	Yes	13.4x 13.0 x 0.8 inch	20W	4.8lb	\$27.99	
SOLARPER K	Yes	21.3 x 13.6 x 0.7 in	30W	5lbs	\$68.13	

SOLARPER K	Yes	15 x 13.2 x 0.7 inches	20W	3lbs	\$49.99\$	
------------	-----	------------------------	-----	------	-----------	--

Table 15: Solar Panel Options

The size and weight are rough estimates, since we will have to create and integrate the Solar Umbrella. We will not know the best dimensions until we are able to physically build and test it out that way. The size of the solar panel should at least be able to completely cover and protect the cooler from the sun. In addition, we can create a solar umbrella configuration that implements multiple solar panels and expands. Knowing this, we can select a smaller solar panel and just use more of them to reach that surface area coverage. A typical beach umbrella is about 7 ft in diameter which means about 38 sq ft.

Output wattage is essential since it determines the amount we are able to recharge the battery. Our battery is lithium ion which will be capable of handling the charge and discharge cycles. We also do not need a very high wattage rating because it would be overkill. The battery is fairly small and the amount of power consumption from our electronics is low as well. The solar panel is simply to be used as supplemental power to extend the longevity of the battery, versus the idea of completely running off solar power entirely.

This 20 watt panel from SOLPERK, as shown in *Figure 32*, is our lightest panel coming in at 3 pounds. It comes with a 21-30 percent efficiency rated panel. It includes the following 1 x alligator clips, 1 x SAE Cable, 1 x 360° Adjustable bracket, 1 x User Manual, 1 x controller charger, 1 x 20W Mono solar panel, 2 x Mount Screws and comes in at 21.3 x 13.6 x 0.7 in (Length x Width x Height). It provides a maximum voltage output of 18 volts and 1.11 amps of current. This solar panel is claimed to be “100% waterproof and rust-proof, and provide a prolonged lifespan of up to 25 years”, this is due to its low-iron tempered glass surface and the aluminum frame which is resistant to rust and corrosion. It can also withstand up to 2400pa of wind pressure and 5400pa of snow load. This panel for the size and power output having a price tag of \$49.99 is quite expensive compared to other products, that being said this is a higher quality panel we can buy but we believe our money would be better spent in something less expensive or with a higher power output such as the next panel we will be looking at. The solar control that comes with this product is also on the cheaper side only being able to handle 8 amps of current when most panels we looked out for this power rating came with a 10 amp control.



Figure 32: 20 watt SOLPERK Panel

This 30 watt panel from SOLPERK, as shown in *Figure 33*, is our biggest and most powerful yet, not to mention the most expensive. But this doesn't come without its perks, this product comes with a 21-30 percent efficiency rating it includes the following 1 x alligator clips, 1 x SAE Cable, 1 x 360° Adjustable bracket, 1 x User Manual, 1 x controller charger, 1 x 30W Mono solar panel, 2 x Mount Screws. This panel as mentioned above is the most expensive we will be looking at coming in at \$68.13 and comes in at 21.3 x 13.6 x 0.7 in (Length x Width x Height), weighing approximately five pounds. This product is rated 4.4 stars with 2202, and is listed as the number one best seller in solar panels as of 11/12/2023. This solar panel is claimed to be "100% waterproof and rust-proof, and provide a prolonged lifespan of up to 25 years", this is due to its low-iron tempered glass surface and the aluminum frame which is resistant to rust and corrosion. *Figure 34* shows the proof of the math for the following conversions. It can also withstand up to 2400pa of wind pressure and 5400pa of snow load. Upon further research this means that this panel can withstand a wind speed of 2400 newtons per square meter, which is about the equivalent of 216 kilometers / 134 miles per hour. A 5400 pa snow load equates to 5400 newtons per square meter or when converting pascals into pounds per square foot we get the following.



Figure 33: SOLPERK Panel 30 Watt

To convert the snow load from Pascals to pounds per square foot (psf) and then estimate the snow depth in inches, you would need to consider the density of the snow. The density of snow can vary widely, but a common assumption is that 1 Pascal is approximately equal to 0.0208854 pounds per square foot (psf) of snow load.

Given that the snow load is 5400 Pa, you can calculate the equivalent load in psf:

$$\text{Snow Load (psf)} = 5400 \text{ Pa} \times 0.0208854 \text{ psf/Pa}$$

$$\text{Snow Load (psf)} \approx 112.7076 \text{ psf}$$

Now, if you assume a specific snow density (e.g., 15 pounds per cubic foot is commonly used for design purposes), you can estimate the depth of the snow:

$$\text{Snow Depth (inches)} = \frac{\text{Snow Load (psf)}}{\text{Snow Density (psf/ft}^3\text{)}}$$

$$\text{Snow Depth (inches)} \approx \frac{112.7076 \text{ psf}}{15 \text{ psf/ft}^3}$$

$$\text{Snow Depth (inches)} \approx 7.5145 \text{ feet}$$

Figure 34: Conversion Explanation Example for Solar Panels

With all that said this means that this panel can take a beating from the outside elements and should be extremely dextrous in normal use cases. It has a max current output of 1.67A with a max voltage output of 18 volts of power.

This panel from top solar Top Solar, as shown in *Figure 35*, comes in at an output of 12 volts DC and a power output of 20 watts. The \$27.99 package comes with a 1pc 20 watt mono solar panel, 1pc 10A solar charger controller, 1pc 6.5ft extension cable with Battery Clips, 1pc 6.5ft extension cable with O-Ring Terminal. This panel has an advertised efficiency of 24 percent, and a 12 month “worry-free” warranty. It is advertised to have a corrosion resistant aluminum frame which fits our specifications for weatherized products that are resistant to saline environments. It has a maximum current output of 1.11 Amps and maximum voltage output of 18 volts which equates to our 20 watts of power. It weighs 4.8 pounds which is perfect for our solar umbrella not too heavy.



Figure 35: Top Solar 12 Volt Solar Panel

Figure 36 shows us the layout of our panel and how it is made. We first start with the aluminum frame, this is advertised to have some pre-drilled holes which may prove useful, but this also tells us the panel frame is strong enough to be used as a mounting point. We then see that it is followed by low iron-tempered glass then EVA film. After that we have our monocrystalline solar cells followed by another sheet of EVA film. The solar charger controller will be a very useful item we can use for our project and avoid having to design advanced power delivery. We lastly have a TPT backsheet to round off the back. When compared with our other panels it weighs the most for being the smallest in size, this however is not a bad thing since we need a more durable panel but we need to point that out from a comparison perspective. It has a solid rating of 4.3 stars out of 2444 reviews.



Figure 36: Panel Layout

3.3.10 Cooler Comparisons

Our cooler is the most crucial piece of our project since it is our project so picking one will be extremely difficult, but we have some criteria. It needs to be a well established brand, light weight, and come with wheels. Most importantly it needs to be able to fight our internal components and have plenty of space for drinks. Below in Table 16 are the options and which one the team chose.

Name	Water Proofing	Size	Wheels	Weight	Cost	Picture
Coleman 316	Yes	100 qts	Yes	15.4 lbs	\$115.99	
Coleman 316	Yes	62 qts	Yes	12.3 lbs	\$84.99	

Igloo Latitude	Yes	90 qts	Yes	22.4 lbs	\$119.99	
----------------	-----	--------	-----	----------	----------	--

Table 16: Cooler Options

This product, as shown by *Figure 37*, is a very large Coleman cooler with a capacity of 100 quarts, it is advertised as being able to hold 80 cans with 50 pounds of ice. This cooler weighs 7 kilograms or 15.4 pounds when empty. With its dimensions of 17.3"D x 36.9"W x 18"H this cooler will be plenty to encapsulate all over our necessary components.



Figure 37: Coleman 316 1

This is identical to our first cooler but with a smaller sized configuration, as shown by *Figure 38*, we wanted to include this because we may downsize our cooler if we find that we have adequate space in the 62 quart 28.2 x 15 x 18 inches. Priced at \$84.99 it is on the expensive side as well but should do the job. The cooler weighs a nice 12.3 pounds and it's advertised to hold 50 cans with 26 pounds of ice.



Figure 38: Coleman 316 2

Lastly we have the igloo latitude 90 quart cooler coming in at \$119.99 in *Figure 39*. This cooler can hold 137 cans, no mention if that includes ice or not, and weighs 22.4 pounds it comes in at a size of 31.84" L x 17.84" W x 19.5" H. This one is quite heavy when compared to the rest so that will need to be factored in.



Figure 39: Igloo Latitude

Outcome: We are not entirely sure what to pick. These coolers were far more expensive than we thought they would be but we believe that the Coleman 100 qt will currently best suit our needs simply because of its size and relatively lightweight.

3.2.10 Software Comparisons

An application is integral for our project, therefore having a software to allow us to easily make an application for the phone would be ideal. It will need to have the ability to make a GUI for the consumer to press buttons, and go into menus to select the specific settings for the cooler. It doesn't need to be a complex tool, just something that will allow us to connect an Arduino device through Bluetooth straight to Bluetooth through the user's phone. As such, we also are looking for an environment to host the code on the microcontroller. Our main constraints for this are price, simplicity, and effectiveness. Below in *Table 17* are the options and which one the team chose.

Name	Pre made Libraries	Pre Made Assets	Customizability
Android Studio	1000+ Pre Made Libraries	<100 Premade Assets	Limitless, but has less than 10 programming languages
React Native	500-1000 Pre Made Libraries	N/A	Limitless, but only 1 programming language
Arduino IDE	10,000+ Pre Made Libraries	N/A	Limitless
MITApplication Inventor	100+ Premade Libraries	100+ Premade Assets	Limited to the what is in MITApp Creator

Table 17: Software Option

For price we do not want to use a software that requires a fee or a license. An example of this is making an IOS application, and that would require an Apple License or a hack around that will take too much time. Also for the environment we don't need one that codes for us but something simple like the Arduino IDE, that has all the libraries and examples to put our project into motion. Another thing is simplicity. Our goal isn't to have a state of the art application, it's to have a device

that can connect to the state of the art cooler. Having a simple application that can be easily created on the fly is our goal. Finally its effectiveness. We want an application that works, so we are going to test similar coding using each platform to see which one works the best and has the least amount of latency through Bluetooth between device and microcontroller.

Android Studio is an IDE specifically for Android application development. It has a wide variety of features, such as showing the output of code in a UI and has multiple languages to be coded in. For the constraints, it is open sourced and very effective. However, it isn't a simple tool to understand, and requires extensive knowledge of the required languages and application development to be used effectively.

React Native is a JavaScript tool that allows users to create apps for IOS and Android. This application is similar to Android Studio's UI, but is more concise in having a single language with multiple libraries to work with. For the constraints, the cost is free, and is extremely flexible. The only issues would be an intermediate learning curve, but can be learned in about a week or so.

The Arduino IDE is the main way for a user to put code into the Arduino from a computer. This isn't just useful for Arduino's, but other embedded systems that can be programmed under this IDE. For constraints it is open source, useful for any programming needed for Arduino functionality, and is simple to use with its c like language.

MIT application creator is an easy pre-built tool meant for making Android apps with little coding knowledge required. It has an easy to use UI that can allow the user to click and move around buttons onto the screen, and allows user customization by if statement logic. For price and simplicity, it is very simple to use and it's open source. The only problem is that it is limited in what it can do. A user has to follow its conventional design and works well with buttons and switches. If there is an idea for a nice ui or complex options, then this wouldn't allow a user to use it.

3.2.11 Microcontroller Comparisons

Our goal is to have a microcontroller that can easily connect through Bluetooth to a phone, and not steal too much power from the battery powering the smart features. The Microcontroller is just linked to the PCB's mcu and the application interface. However, it does have to deal with computing lights and music at the same time. So, in short, the 3 main factors are cost, power consumption, and flexibility. Below in *Table 18* are the options and which one the team chose.

Name	Bluetooth Type	Wi-Fi Type	SRAM	Voltage Supply	Cost	Picture
Arduino Uno (with HM-10)	Bluetooth 4.0 BLR	N/A	32 KB	7-12V + 3V	\$33.17	

RaspberryPi-4b	Bluetooth 5.0	802.11ac Wi-Fi	4-8 GB	4.7V-5.25V	\$65.69	
MSP 430	N/A	N/A	512 KB	1.8V-3.6V	\$54.95	
ESP32	Bluetooth LE 4.2	802.11ac Wi-Fi	520 KB	2.3V-3.6V	\$15.88	

Table 18: Microcontroller Options

For the reasons for the criterias, first the cost is derived from how the smart features don't require a complex microcontroller. The main reason it can't be done just through the PCB is that Bluetooth connection, which would be difficult to configure with just the PCB, and a microcontroller would make the process more streamlined. For the power consumption, we want to have a battery in order to power all of the smart features. However, having powered lights and speakers is already draining the battery so having a microcontroller as the main monitoring device taking most of the charge wouldn't be wise. The final thing is flexibility. We don't just want a mcu, we need something that is able to change between multiple functions with the least amount of latency, along with having the capability of having lights and music playing at the same time.

Figure 40 shows the Arduino Uno Rev3. This device is the most powerful Arduino used and has a voltage level of 7-12V and a cost of roughly \$27.95. This is not the cheapest device, but it makes up for it with its all around capabilities and usability of the device. As it is one of the most popular microcontrollers, there is plenty of documentation on the device and has multiple projects of smart coolers using this. The only issue is the unnecessary price point and the power of the device. This would require more power to run this device, and running less may cause problems and fry the board. However, due to the capabilities of the device this would be one of the best options due to the unlimited documentation that will help us implement and control our design more easily.



Figure 40: Arduino Uno Rev3

Figure 41 shows the Raspberry Pi 4b. This device is not just a microcontroller, but a full on single board cpu that also runs at a lower voltage of 4.7V-5.25V. Even with a high 2.5Z and \$65.59 price tag, the flexibility on this device is endless and has the capabilities of what we need plus more.

This option also has Bluetooth and Wi-Fi support, so there would be no need to get extra devices. The only issue for this is that its price point and current rating is higher than most options, and would end up wasting power in the long run while it just runs in standby. This option would have been explored more if a screen was connected to the cooler but as that isn't planned, this is likely not going to be chosen.



Figure 41: Raspberry Pi 4b

Figure 42 shows the MSP 430. This microcontroller runs at 100-2675 microamps and has a voltage of 1.8-3.6, making this one of the most efficiently powered microcontrollers out there. There is also the fact that this was used extensively in a required class, making the team already knowledgeable with the specs and capabilities of the device. However, this does not have Bluetooth capabilities and has a price point of \$54.95 dollars. There is also little documentation on projects relating to ours using the device.



Figure 42: MSP 430

Figure 43 shows the ESP32. This device is said to be low powered, easy to program, and has built in Wi-Fi/ Bluetooth capabilities. It has a low voltage count, and cost at max 6V and \$15.88 making it recourse efficient compared to most microcontrollers. It also has the Arduino ide for simple programming of the device. The only problem would be its capabilities, as it is low powered it would need a way to do multiple tasks at a time. The most intensive thing in the project would be the light display with music, but according to sources this has been done, but will be a little tricky to implement because of the limiting device. This is essentially equivalent to Arduino Nano and already has Bluetooth in it.

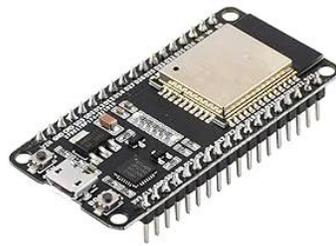


Figure 43: ESP32

Figure 44 shows the bluetooth module to assist a normal Arduino, the HM-10. For the options that doesn't have Bluetooth, this would have been used for connecting it to Bluetooth, as Wi-Fi would be unnecessary. This runs at a low 50mAs, 3.3V, and costs only \$5.27. This works easily with Arduino based devices, and we would pair this with the Arduino R3 as that microcontroller does not have Bluetooth on it. This would alleviate any Bluetooth issues but the problem will be that it will cause more strain on the battery pack. However, if an option that doesn't have Bluetooth is chosen, this would be the best outcome.

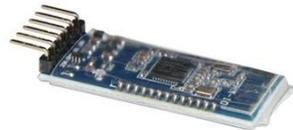


Figure 44: HM-10 Bluetooth Module

Outcome

We chose to use the ESP32. It is an Arduino wrapped in an easy to use package, and already has Bluetooth and Wi-Fi integrated in it. There would be no need for extra modules for its functionality and there are multiple sources of more complex projects using the small device. It also has one of the best price points and powers efficiently than most of the others on the list. Below in *Table 19* are all the options the team chose for continuing the project..

3.3 Part Selection Summary

Name	Part	Price	Picture
GikFun	Temperature Sensor	\$11.58	<p>High quality stainless steel probe package</p> <p>High stability, high precision, corrosion resistant, waterproof and oxidation proof</p> <p>High quality stainless steel</p> <p>Stainless steel waterproof probe</p> <p>National standard compensation</p>

Alitove 50 Pc LED Lights	Lights	Free (Already Own)	
Hzipowen	Speaker	\$11.99	
Acxico	Audio Amplifier Module	\$8.99	
PENGLIN	USB Port	\$14.99	
MELIFE	Female USB Port	\$6.99	
Vatrer	Rechargeable Battery	\$27.99	
Topsolar	Solar Panel	\$27.99	
Coleman 316	Cooler	\$115.99	
Arduino IDE	Software (microcontroller)	Free	NA
MITApplication Creator	Software (app)	Free	NA
ESP32	Microcontroller	\$15.88	

DaierTek ON/OFF Rocker	Rocker Switch	\$6.99	
Songhe 4x4 Matrix Keypad	Matrix Keypad	\$6.99	
TOPLEE Electronic Cabinet Lock	Electronic Lock	\$13.99	

Table 19: Part Selection

4. Standards and Design Constraints

This section will classify the standards and constraints that are necessary for our project.

4.1 Standards

When making a device, there are certain standards that need to be followed. Standardization occurs in order for consumers to understand the bare minimum of a product and quality for anything sold. Below are standards noted for our design such as Bluetooth for our Bluetooth application and the speaker to ensure it has quality output. Standards are also done to ensure that there are no safety hazards to the product. Such as the battery, if there is a leakage that can hurt the product or the consumer then that will not satisfy consumers, and such, the standards. Below will discuss these topics in more detail along with how these standards were used for the design of the cooler.

Commented [10]: can add in solar panel standards from ieee as well

Commented [11]: can also add in battery standards (aka why 12V is standardized)

Commented [12]: Added the battery standards

4.1.1 Bluetooth Standards

The standards for Bluetooth and other similar communication protocols are in IEEE 802.15.1 . Bluetooth is the sharing of wireless communication for short range data transfer. Bluetooth operates in 2.4 GHz ISM. The range of frequency allowed is between 2400 MHz to 2483.5 MHz, with few countries such as France deviating from these numbers. For the power specifications, it needs to operate between 100mW and 1 mW for power control. Due to the nature of our project other communications would be unnecessary, such as constant Wi-Fi taking more power than necessary for the microcontroller and battery. Below in *Table 20* shows the Bluetooth power standards.

Power class	Maximum output power (P_{max})	Nominal output power	Minimum output power ^a	Power control
1	100 mW (20 dBm)	N/A	1 mW (0 dBm)	$P_{min} < +4$ dBm to P_{max} Optional: P_{min}^b to P_{max}
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (-6 dBm)	Optional: P_{min}^b to P_{max}
3	1 mW (0 dBm)	N/A	N/A	Optional: P_{min}^b to P_{max}

Table 20: Bluetooth Power Standards

4.1.2 Software and Hardware Validation Standards

The standards for Software and Hardware Validation can be found in the IEEE 1012-2016 documentation. It regards the Validation and Verification process where the software and hardware interact for the sole purpose of solving the necessary problem. It is also the process of creating, reviewing and revising software and hardware interaction during the design process. This process ensures the correct methodology about going through the design process, and that much can be said for our project. Our project has constant hardware and software interaction, so following the V&V process is integral for a successful product.

4.1.3 USB Standards

USBs, or universal serial buses, are ports allowing connectability between electronic devices. There are multiple types of USBs, each with different connectability and power output for each type. The USB standards began in 1997 and has subsequent revisions for each release following. Below in *Table 21* shows the USB standards.

Name	Speed Range	Power	Connectors
USB 1.0	1.5 Mbps - 12 Mbps	2.5W	Type A, Type B
USB 1.1	1.0 Mbps - 12 Mbps	2.5W	Type A, Type B
USB 2.0	280 Mbps - 480 Mbps	2.5W	Type A, Type B, Type C, Mini, Micro A, Micro B, Type C
USB 3.0	3 Gbps - 5 Gbps	4.5W	SuperSpeedUSB Type A, SuperSpeedUSB Type B, Type C, SuperSpeed Micro A, SuperSpeed Micro B
USB 3.1	5 Gbps - 10 Gbps	4.5W	SuperSpeed+ Type A, SuperSpeed+ Type B, Type C, SuperSpeed Micro A, SuperSpeed Micro B
USB 3.2	5 Gbps - 20 Gbps	4.5W	Type C
USB 4.0	20 Gbps - 40 Gbps	7.5W	Type C

Table 21: USB Standards

4.1.4 LED Light Standards

The LED light standards are defined in the IEEE 1789-2015 . The main issues regarding lights are their brightness, and their flickering which can be a hazard for people with epilepsy and people sensitive to strobing lights. Compared to other light options, LEDs have a lower risk compared to others, but the recommended practices for flickering and light brightness is shown in *Figure 45* below

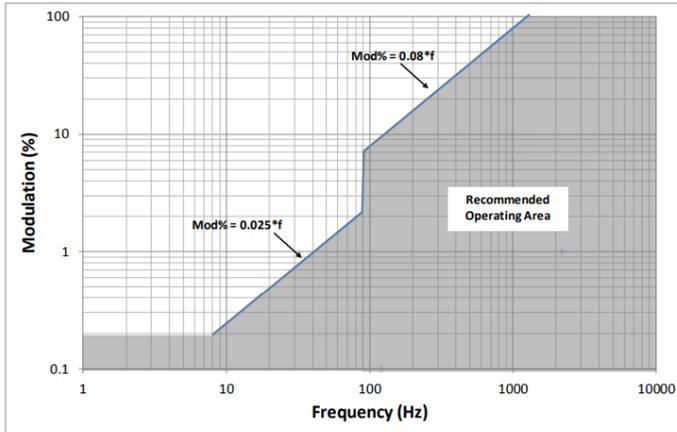


Figure 45: Light Frequency to Modulation Standards

4.1.5 Sensor Standards

The standard for sensors is located in the IEEE 2700-2017 publication. Our project requires a sensor for temperature, so it is necessary to check if it falls under normal regulations. For it to fall under standards, it needs to accurately tell temperature between a -3 to +3 sigma limit. This also states that it needs an awareness system of the actual temperature around the device, not just the temperature of the device. The exact conditions of the Absolute Temperature Error are stated in the Figure 46 below.

Definition		Temperature measurement accuracy over the entire measurement range.
Unit of measure		°C
Conditions		At RT (25 °C)
		For operating temperature range
		At operating voltage
		After PCB assembly (place and reflow)
Distribution	Minimum	-3 sigma limit
	Typical	Mean
	Maximum	+3 sigma limit
	Dispersion	Standard deviation

Figure 46: Sensor Standards

4.1.6 Solar Panel Standards

Solar Panel and solar charging standards are found in the IEEE 1562-2007 guide. Due to using solar panels, there can be concerns regarding the safety of the panels. This is due to the solar radiation and how the energy saved onto the panels can affect the materials surrounding the panels. For our project these standards should be satisfied due to the limited amount of radiation the size of our panels will emit. If we were using multiple metered sized panels then there would be an issue.

4.1.7 Speaker Standards

Standards for Speakers can be found in the IEEE 269-2019 documentation. As our speakers are used for music, the acoustic output level needs to be considered. Our plan was to use an analog passive speaker paired with an audio amplifier module. Below shows the measurements required for analog systems. The details can be shown in *Figure 47* shown below.

Measurement	Signal direction	Digital terminal	Digital subsystem	Analog terminal	Analog subsystem
Frequency response	Receive	12.2.2	12.2.2	12.2.2	12.2.2
	Send	12.2.3	12.2.3	12.2.3	12.2.3
	Sidetone	12.2.4		12.2.4	
	End-to-End	12.2.5		12.2.5	
Output level	Receive	12.3.2	12.3.2	12.3.2	12.3.2
	Send	12.3.3	12.3.3	12.3.3	12.3.3
	Sidetone	12.3.4		12.3.4	
	End-to-End	12.3.5		12.3.5	
Conversational gain	Receive	12.4	12.4	12.4	12.4
Tonal balance	Receive	12.5	12.5	12.5	12.5
	Send	12.5	12.5	12.5	12.5
Balance and polarity	Receive	12.6	12.6	12.6	12.6
Echo	Electrical	12.7		12.7	
Self-noise	Receive	12.8.1	12.8.1	12.8.1	12.8.1
	Send	12.8.2	12.8.2	12.8.2	12.8.2
Level-dependent dynamics	Receive	12.9.4	12.9.4	12.9.4	12.9.4
	Send	12.9.5	12.9.5	12.9.5	12.9.5
	Sidetone	12.9.5		12.9.5	
	End-to-End	12.9.5		12.9.5	
Distortion and induced artifacts	Receive	12.10.2	12.10.2	12.10.2	12.10.2
	Send	12.10.3	12.10.3	12.10.3	12.10.3
	Sidetone	12.10.4		12.10.4	
	End-to-End	12.10.5		12.10.5	
Speech quality in quiet environment	Receive	12.11	12.11	12.11	12.11
	Send	12.11	12.11	12.11	12.11
Speech quality in noisy environment	Receive	12.12	12.12	12.12	12.12
	Send	12.12	12.12	12.12	12.12
Mute leakage	Receive	12.13.1		12.13.1	
	Send	12.13.2		12.13.2	
Device delay	Receive	12.14.1	12.14.1	12.14.1	12.14.1
	Send	12.14.2	12.14.2	12.14.2	12.14.2
	Sidetone	12.14.3	12.14.3	12.14.3	12.14.3
	End-to-End	12.14.4		12.14.4	
Stability loss		12.15	12.15	12.15	12.15
Howling		12.16	12.16	12.16	12.16
Max acoustic output	Maximum	12.17.1	12.17.1	12.17.1	12.17.1
	Peak	12.17.2	12.17.2	12.17.2	12.17.2
Impedance	Receive				12.18.1
	Send				12.18.2

Figure 47: Speaker Frequency and Specifications Standards

4.1.8 Standard for Rechargeable Batteries

For the battery, it needed to satisfy the standards of the casing, the actual cell, and the adaptor that charges it. It needed to satisfy that whenever it had charged, that it is to 100% capacity of the cell. IF the cell couldn't be fully charged or has been limited to less than maximum threshold, then it was liable to be broken or needs to be redesigned. A consumer being sold a product advertised for full capacity yet is limited in any way is not allowed to be sold. Furthermore, the actual pack needs to be adjusted to where it can not harm items around it when sold and further wear and tear. Below in *Table 22* shows the minimum requirement for the subsystem standards.

Table 2—Minimum subsystem requirements

System component	Standard
AC/DC adaptor or charger	IEC/UL 60950-1 or IEC/UL 62368-1 and IEC 62368-3 or appropriate safety standard of destination country
DC/DC adaptor or charger	IEC/UL 60950-1 or IEC/UL 62368-1 and IEC 62368-3 or appropriate safety standard of destination country
Battery pack	UN Manual of Tests and Criteria, Section 38.3 Lithium Batteries and appropriate safety standard of destination country
Cell	UN Manual of Tests and Criteria and appropriate safety standard of destination country, refer to 5.7.4

Table 22: Subsystem Requirements

4.1.9 Standards Impact on Design

USB

We decided to use USB 2.0 Type A due to the lower power consumption of it and Universal usage to charge small mobile devices. Since we are using two ports we would have either had a max power draw of 5 watts or 9 watts. The concern for USB 3.0 becomes the total power draw on our system if all ports are being utilized, in combination with our other ambient functions. We wanted to preserve the longevity of our system where we are not having to worry about rapidly draining the battery while we are charging a mobile device. We also are able to fully run our USB charging off of pure solar energy at the 5 watt power point. This will allow us to do so many things; we will be able to charge our battery and use both charging ports at the same time, we can run our system and utilize both ports with only solar without engaging charge from our battery. In our eyes it's about being efficient with the power we have to utilize it in a meaningful way it doesn't mean anything if we charge our phone in 5 minutes then our cooler becomes inoperable 5 minutes later. USB type A can be integrated into so many things and if we wanted to we could base our whole electrical system off of that power type. This will allow us to charge almost any device since we are using a Female Type A adapter 99% of mobile chargers use a Type A Male connect to a Type A female connector so it's a no brainer to use this as our power transport and connection mechanism.

Bluetooth

There are multiple types of Bluetooth options and some options are limiting on what devices that can be connected. For example, if we had gone the route of basing our application on IOS, we would have to worry about that connection. Most Bluetooth modules such as the HM9 or 8 would

work perfect for Android, but wouldn't connect to IOS systems. Thus we would have had to choose the HM-10, or other modules that support Bluetooth 4.0 LE. Due to IOS having limited standards, we decided to work with Android and use the Bluetooth LE 4.2. This allows us to work with windows as well, so we can test apps using emulated programs.

4.2 Design Constraints

Making sure the battery and components fit into the cooler in a non-invasive way is going to be very important to our design and design philosophy as we want everything to be contained inside the system. With this in mind we still want plenty of space for ice, drinks, food, and other items we may want to store inside the cooler. This made it so we could not downsize our cooler to a small size which will limit our cooler selection to bigger more expensive products. We also need to design our internal components in a way such that they do not take up too much space and are a uniform box shape. In addition to this we will need to ensure all wires are unable to be displaced or unplugged from their intended location.

Our lid must be able to still close properly to allow for an adequate seal even with the Lid Status Sensor: Depending on the type of cooler, and type of lid we are going to be working with, we need to determine the most accurate and robust way to detect when the lid is open or closed. This system will have to be tied to the locking mechanism, but we do not want these systems to be intrusive into the system or can be easily damaged by the user.

The issue with this is high current/power draw from the battery will lead to high temperature inside the sealed electronic portions of the cooler. With our current plans we do not think it will be a major issue, but it's something we need to plan ahead for. Dissipating heat inside the coolers electronic systems will be the biggest design/physical construction task we may have to face. This should only become an issue if our electronics produce an ample amount of heat or our battery generates too much heat with additional current draw.

Construction within the cooler will be a challenge with drilling holes into the cooler. We need to be concerned about cracking or unlevelled drill holes. This is why we will need to measure twice and cut once. With a cooler there is not a lot of room for errors when drilling as fixing or creating any errors will reduce overall strength of the cooler body. Not only that but working in the cooler confined space will prove a challenge as we need to think before we add and secure components. This is going to be an important construction practice as we may need to move, repair or replace components. Permanently securing an item may prove fatal as if that component needs to be moved it could cause major damage to the interior of the cooler and the component itself. Construction will have to be loosely tested and thought out before permanently implemented as we will not be able to reverse certain aspects of it.

4.2.1 Health and Safety Constraints

Our project is a Smart Cooler, so it is imperative that the food and items in our product meet the regulations as to where storing food inside of it won't cause a health risk to the consumer. To ensure this, there are standards in place that include electronic device contact and temperature correctness.

Having food, liquids, and ice contacting electronics and PCBs becomes a hazard that can lead to electric shock drowning. This can be fixed by ensuring most if not all the electronics are stored in a separate compartment in the cooler. The design is that the storage compartment and electronics are stored in two separate compartments, with the only connection being the wiring for a small stainless steel temperature sensor. Even then, the wiring for the temperature sensor will be extremely small and sealed off to reduce these two compartments' access to one another. The electronic components are not just in a separate compartment, but in one that is sealed from the elements. We will be waterproofing and sealing off this compartment entirely. By having some combination of a rubber seal, insulation, and other materials, we are able to create this compartment. This is because having dirt or outside contaminants accessing the electronics can cause the cooler to not work as intended and could cause electrical hazards.

This also caused us as a team to change the design to a cooling option utilizing ice instead of the compressor. The compressor would have caused strain on the electronics and further interactions between the food compartment and the electronics. We would have had to further seal off more compartments and ensure it is sealed correctly. This is because the compressor works by having the coils inside the actual insulated interior walls. The compressor has coils that expand and contract the gas inside and in turn, this leads to the coil reducing its temperature. If we had to cut open the cooler wall and create additional compartments around the perimeter of the cooler to insert these coils in, it would have led to additional sealing issues that would have affected our health and safety.

In addition, the use of the refrigerant gas in the compressor technology would have posed another issue. Inhalation of refrigerant gas especially in large quantities can make people sick and lead to brain damage. Refrigerant fumes such as freon can be dangerous and potentially deadly depending on the amount inhaled. Accidental exposure can also happen if there is a leak anywhere within the compressor system. If we had utilized this compressor technology, we would've had to take additional extra care in sealing it. Leaking refrigerant is bad not only for us but for the environment as well. Having leaking gas near electronic components is dangerous, especially because it can be ignited. Because we have so many electronic wirings in its vicinity, any spark/static could potentially lead to ignition of leaking gas and cause a fire/explosion. Especially when all the components are contained within the cooler with a 12V battery, we want to minimize the possibilities of all hazards.

We also considered the electronics on the outside of the cooler. For locking the cooler, if the user does not wish to use the application, then the electronic lock is available to use. To ensure that the electronics will not interact with food, all wiring and keypad will be placed in front of the electronic compartment. For the LED lights, they will only be used on the outside of the cooler.

Another safety hazard is the correctness of the temperature sensor, as if it is faulty then the user could believe that food inside is a safe temperature when it is not. To ensure this, the temperature through the microcontroller is constantly checked at regular intervals and checks to see if there is a sudden spike or no change after a long period of time. If there is, it would alert the microcontroller and send a signal to the application in which when the user checks the temperature, it will display that there may be an error in reading the temperature. The sensor also does not check the exact temperature of the food but the general temperature of the cooler. It is under the user's discretion

that the item is the right temperature. We may also implement a system where we have temperature sensors running independently to check with each other if they are correct. With the use of smart coding we can add a system that alerts the user if the sensors are at a

General construction of the cooler is a hazard in itself since we will be utilizing power tools. Power tools can be very dangerous if not correctly used with the proper personal protective equipment. We will need to implement the use of heavy gloves, safety glasses and knowledge of the tools we are using. Setting up a safe work station with another member supervising when power tools are in use is critical for safety.

Our battery is also a point of safety we have to consider as it is the largest component of the internal system. Securing the battery so it cannot move while the cooler is in use is a top safety concern. A loose battery can lead to damage to our electronics, inside of the cooler, and battery itself. Loose or disconnected battery terminals are extremely dangerous as they can shock and injure users of the cooler. Heat coming from the battery may also pose a danger to food integrity, our electronic components, or the battery itself. This is why temperature monitoring of our battery is extremely important and setting up a “kill” switch if temperatures get too hot will be a major safety feature.

4.2.2 Environmental Constraints

As our project is a cooler, it needs to be able to move around in harsh environments while still being able to function as a smart cooler. Either just outside someone's house to the beach, we have adjusted our design with the solar panel, cooling mechanism, and the battery. Electrical components must be completely sealed off from any food/drinks for protection. This is essential because we want a clear separation between the cooler and electronic components. We want to make it like a normal cooler as if a user could not tell the electronics were present on the inside. This is also important because we don't want to expose our electronics to outside weather variables, mostly moisture and any sort of dirt or sand that could become present inside the system. With further thinking, we need to develop an access point so we can maintain and access our electronics; This will most likely be a hinge or a tension sealing lid with latch configuration.

Since we are going to be introducing our cooler to a saline environment it needs to be rust and corrosion resistant. This means we must implement materials such as stainless steel and aluminum when constructing internal and external components. Saltwater is another thing we need to protect against. This means all external metal components will need to be resistant to rusting and corrosion, so this limits us to what components we will be able to use. Choosing materials and finding parts of that material will limit the scope of parts and products that we may be able to use. The grade of materials will have to be for outside or industrial use as these will be less prone to breaking and malfunctions in these environments.

Our environment is very harsh and we are susceptible to all weather conditions. That means we cannot get away with just building an unprotected system and running it. We need to make sure that every part of our system is resilient to all types of weather. The main environments we will be dealing with are the beach and grass/dirt environments. Constraints in the beach environment are sand and saltwater. Undoubtedly our cooler will get sand on it and need to be washed off like any other cooler. Our cooler will have to be able to not only not let sand be introduced to the electronics but also make it so that water does not intrude when being washed off. This makes it so that we

have to water tight seal every covering of our electronic component. Water introduced into the system in any way would be a critical failure of the system. To ensure this doesn't happen, we will have to introduce a system to check if any moisture is able to seep into the electronic holding area.

Components installed on the outside (touchscreen/keypad) should be able to withstand outside weather factors to a basic degree. Analog switch and pressure buttons are usually the most interacted with features on a system; these buttons need to be all of the above: resilient, durable, weatherized, and with tactile feedback. In order for keypads and buttons or switches to be resistant to the elements of the environment, there needs to be a certain level of encasement or separation with the electronic equipment. Separations such as silicon (or any rubber) based housings, o-rings, and sealed connectors.

4.2.3 Budget & Time Constraints

Our project is being funded by the team, so we had to limit our design to a budget of under \$700. Budget is a huge constraint for us as a team because we are all college students living in a situation where we need to work part-time jobs to make enough money to live properly. For this reason we have tried to keep our budget as low as possible so that the per-group-member cost is greatly reduced. Additionally, in order to further offset the costs of this Senior Design project coming from our own pockets, we have ventured into the possibility of receiving scholarships, grants, and/or sponsorships. Constraints that we have found and ran into is that not many people or companies are willing to donate a few hundred dollars to students working on a project this early in the process, so as of now, we have no outside income assistance for this project, and we are paying for everything on our own as a group. In the market, smart coolers are priced around \$300-\$1200 dollars, so we wanted to balance functionality with cost to ensure that the user gets a good product at a good price. The main points that are affected by the budget was the cooling mechanism, design emphasis, and microcontroller, and additional and unforeseen costs and components.

In our project, we had originally had an emphasis on the cooling capabilities and electrical enhancements of our product. For cooling, we desired to create our own cooling mechanism, and we had a desire to do lights in multiple areas inside and outside of the cooler. However, after research we discovered the best case would be a compressor and that would put a heavy strain on the budget we had. Compressors have a range of \$250-\$650 dollars and that would take away from other user functions such as the locking and user functionalities, which is why we chose to use ice and focus our efforts into other areas. The lights also would be difficult to balance inside of the cooler. It could lead to a health hazard and having more lights would put a strain on the battery, causing us to find a more expensive battery which would put more strain on the budget. We also had a desire to add more electrical parts on the cooler such as a LCD screen, but that would require a more expensive microcontroller.

The problem is having more electronic devices would only strain our budget more, so instead we decided to switch focus from an electric project to more of a software based one. Instead of placing multiple electronics on the cooler, we decided to place more electronics onto a Bluetooth application. We forgone the Compressor and LCD route as our focus shifted towards the application, phones already have displays and we wouldn't need to focus on the cooling aspect. In the market, more people are looking for smart features on a phone instead of just having a powerful

cooler. This change also allowed us to use a microcontroller, which was cheaper, and with the phone application allowed for easy connections through Bluetooth with the Arduino IDE.

Time is something that we all wish we had more of. With many of us working, having multiple classes and other obligations making additional time could prove difficult. Trying to find a time everyone is available can also be difficult as plans change and life happens. Having to dedicate more time into something a person thought would only take a set amount of hours can be frustrating and lead to fatigue as more and more time needs to be dedicated to the task. Finding matching times to work and meet is an absolute must and can either be the success or failure of a project or goal. Being able to quickly either dedicate more or less time is crucial for staying on task and staying efficient. Time wasting is a killer that must be avoided, it is always better to stay in front of the curve instead of waiting until a deadline is looming. Time management skills and planning must be used for project success. For our group we always make sure to have 2 times a week that we can schedule meetings via in-person or online. This way we can keep in touch and touch base on important topics, deadlines, and other important activities. This is something we have already planned ahead for in the testing phase as we will need to meet in-person to test components as we have overlapping systems that rely on each other. We have planned 2 times a week that we can meet in person to test our components and are already setting deadlines for part acquisition. Having a premade schedule with set deadlines and especially goals is an excellent way to stay on track. Holding yourself to these goals is the difficult part, when deadlines are far away it is very easy to relax and put it off until it sneaks up on a person at the last minute. Setting intermediate smaller goals and deadlines is what we found to be very effective in keeping ourselves on track.

The testing phase will be the most time consuming as we are going to be looking at many aspects that will take hours to fully test. Our temperature sensors will need to be insured to accurately work for multiple hours. Our battery will need to be tested to see if it can remain charged for multiple hours or days. Our solar panels will need to be active for multiple hours to see if they can charge our batteries and run our electronics. The testing phase and part buying phase will be the most time consuming as we will need to run and monitor these tests for hours. To ensure proper scientific testing ethics we will need to run multiple tests and controls to validate accuracy and data legitimacy. Creating code for our test data will also be time consuming as with code there are bugs and with bugs there are hours of debugging. Weeding out faulty components is also a big loss as if one relies on another we may be in a holding pattern until that part is replaced.

We will need to do adequate research on how to properly test each component so that when it comes time to test the components we purchased, we will not make the mistake of accidentally breaking our own components. For example, if we purchase a button or switch, we need to understand that it is not simple enough to plug a button into a power supply and press it to test if it works. Instead, for this example, we will need to design and build a circuit that supports the desired action of the button and allows us to properly test the button without the risk of blowing it up.

Construction of our cooler itself will prove very time demanding as we are physically building our smart cooler from the ground up. It may take up to an hour to drill a single hole because we don't have the correct drill bit and need to acquire it at another location and return to the job site. Wiring and soldering components together may take hours depending on the quality of soldering needed

and the weatherization of wires needed. Securing components in place is a necessity and will prove time consuming. Having done projects from scratch such as robotics and coding we know that these things take a lot longer than we have already planned for. Being flexible is a must with these things as when a person is 5 hours in and they have only completed a portion of the goal it will prove frustrating.

There may be instances where we cannot secure a component directly to our smart cooler and will need to improvise with a creative solution. One such solution could be to buy a commercially available bracket from a store. If that method does not work for us we can design a bracket and send the designs to a company that can build it for us out of rigorous materials. And finally, we could design a bracket and decide to 3D print the bracket ourselves and hope that the material we use and the design we created is enough to withstand the building constraints of our components. Either route we choose to go down has its own benefits and drawbacks and we will need to make a decision for the best efficiency at that time.

5. ChatGPT

ChatGPT is an AI chatbot using trained language models that talks back to the user, answering any questions from coding to asking it a story, and even just giving the user answers to homework. Chat GPT is not the only one, as this is a popular topic spoken about everywhere in the world. There is privateGPT, that a user can run at home without touching OpenAPI, and a copilot that will write code for a user in an IDE. They are taking the world by storm as it can take away hours of work. However, knowing this, they are not perfect.

5.1 Pros of AI Resources

AI resources are a great tool, not a replacement for learning and understanding. They can help elevate a person's understanding by finding info and elaborating on it. For more general understanding it would be a good way to introduce the user to coding. These are not replacements, but a good way to make learning easier. Below will describe some of the helpful benefits for summarization, basic information, and proofreading.

5.1.1 Summarization

They can help facilitate learning by, for example, helping a user find info from a 1000 page document that they need to understand one specific piece of information. Data sheets are important to understand, but sometimes to help them move forward with understanding and reading through most of the document, there is 1 key info that a user would need to grasp. Below in *Figure 48* is an example of trying to find the power output of a ESP32:

The image shows a chat interface with a document snippet on the left and a chat window on the right. The document snippet is titled "esp32_datasheet_en.pdf" and contains the following text:

3.3.3 Power Management Unit (PMU)

With the use of advanced power management techniques, ESP32 can switch between different power modes.

- **Active mode:** The chip is powered up. The CPU can access, transmit, or listen.
- **Modem sleep mode:** The CPU is suspended and the clock is configured. The Wi-Fi/Bluetooth hardware and radio are disabled.
- **Light sleep mode:** The CPU is powered. The RTC memory and RTC peripherals, as well as the L3P controller are running. Any wake-up events (BAC, SERI host, RTC WAKE, or external interrupt) will wake up the CPU.
- **Deep sleep mode:** Only the RTC memory and RTC peripherals are powered. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The L3P expansion is backed.
- **Hibernation mode:** The external I/O pin controller and I/O expansion are disabled. The RTC memory memory is powered down. Only one RTC hour on the slow clock and optional RTC CPU is active. The RTC clock or the RTC CPU is not active on the chip from the Hibernation mode.

Table 3-2. Power Consumption by Power Modes

Power mode	Description	Power Consumption
Active (CPU working)	Wi-Fi is powered	Pluses refer to Table 4-4 for details
	Wi-Fi OFF	100 mA
Modem sleep	Wi-Fi OFF	100 mA
	Single core (F400)	100 mA
	Single core (F800)	100 mA
	Normal speed (80 MHz)	100 mA
Light sleep	The L3P expansion is powered up	100 µA
	RTC clock	100 µA
Deep sleep	RTC clock	100 µA
	RTC memory	100 µA
Hibernation	RTC memory only	10 µA
	RTC clock	10 µA

The chat window shows a question: "what is the recommended power input and what is the power output of the ESP32?" and an answer: "The recommended power input for the ESP32 is 5V DC, and the power output varies depending on the specific application and power mode. The ESP32 offers fine-resolution power control through a selection of clock frequency, duty cycle, Wi-Fi operating modes, and individual power control of internal components. It has five power modes designed for typical scenarios: Active, Modem sleep, Light-sleep, Deep-sleep, and Hibernation. The power consumption in Deep-sleep mode is 10 µA."

Figure 48: ChatPDF Summarization Example

This shows where from the document the info is from but the explanation from the document. This is a way to help avoid the caveats of generative AI. As that uses information from hundreds of documents and uses word prediction to find the best word, this uses only the document given to

spew out info that is less likely to be false. This can also be proven with the page numbers, so the user can check it themselves. Doing this also allows a user to understand how they should even read the document, as if the power output was beyond necessity then, they can find a different tool and there was no point in trying to understand it.

This also requires understanding of what a person may want, and what is a reasonable answer for these questions. Generally when trying to read a document a user already knows what they are looking for, and having the proof with it makes this tool, ChatPDF, a way to help further understanding rather than hindering it. Using it without understanding what they are looking for would just make this tool useless, so it's hard to take advantage of it or glean much false information.

5.1.1 Basic-Information

Giving Chat-GPT a specific question will likely lead to hallucinations, or a confident false answer. If they ask it something general, then it's very helpful and can help a user learn more about what they're looking for. For our project, there is only one instance using Chat-GPT which is *Figure 34*. This was done to do some basic math and for general knowledge. Another example will be shown below. For the UCF Computer Engineering program, we use C and Java in our classes, and not much experience with C++ which was only used in embedded. To help a user understand it for coding our application, they asked about the general differences between the syntax of input and output and reading from a file. This is just general knowledge, and is something that is very difficult for chat gpt to get wrong, so it spewed out a response, and allowed the user to understand the basic differences. This can be shown in *Figure 49*.

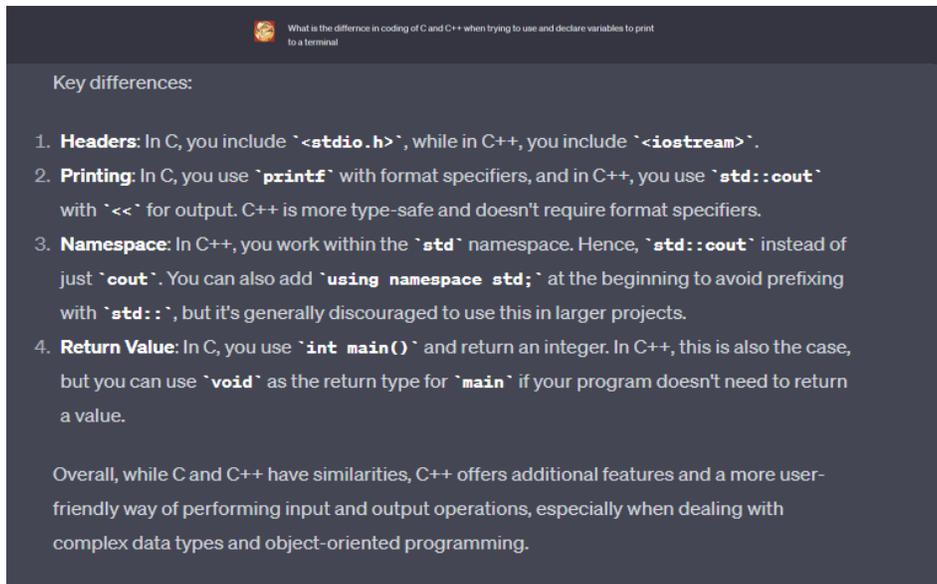


Figure 49: Chat GPT Coding Example

5.1.2 Proofreading

Another useful thing is that it is very helpful with proofreading work. Asking it to write an entire paper is just cheating, but asking it for a section of a person's work to help fix grammar is a different story. There are grammar fixers implemented in google docs and word, but sometimes even that can't fix the issues. This is using it as a tool to ensure that it is of the level of what the person's audience is looking for. Here is an example of an excerpt from the old false positive section fed into ChatGPT. This is just used as a reference and is not used in the document for *Figure 50*.

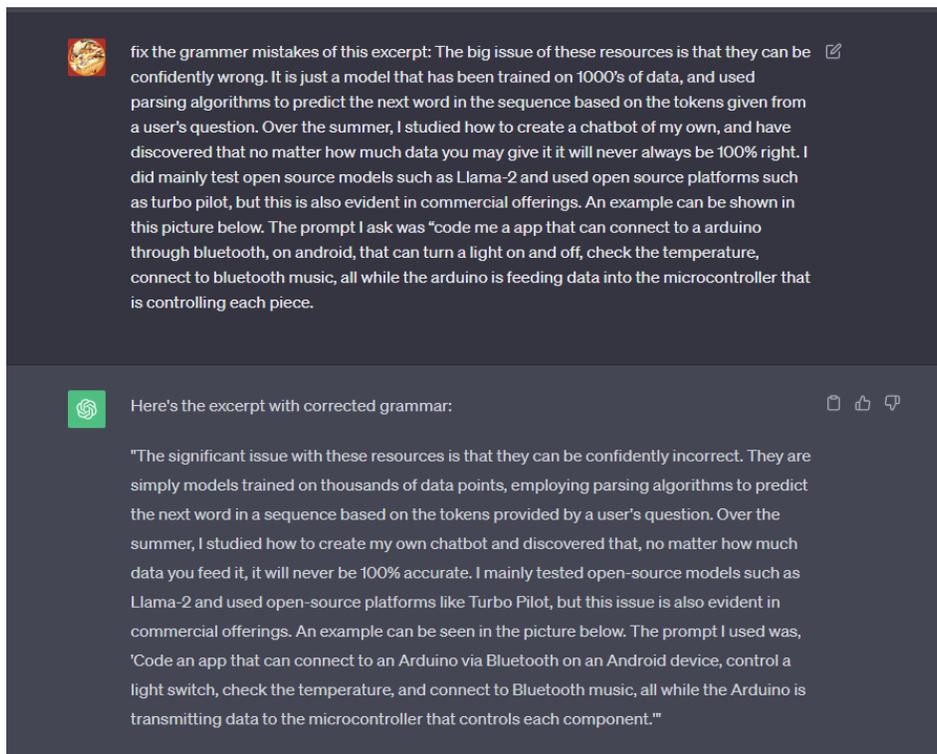


Figure 50: Chat GPT Grammar Correcting Example

This allowed the user to understand their mistakes and use it as a reference for further understanding on how to employ that kind of language throughout the paper.

5.2 Cons of AI Resources

AI tools are far from perfect, and can lead to more negativity than good. Directly copying from a basic prompt to asking it to code their entire application will take away any learning possibilities and can lead to a user falsely believing code should work but in actuality it isn't right at all. Many

people use these tools thinking it will solve all their problems, however one of the two big issues are false positives and lack of learning.

5.2.1 False Positives

The big issue of these resources is that they can be confidently wrong. It is just a model that has been trained on thousands of data, and uses parsing algorithms to predict the next word in the sequence based on the tokens given from a user's question. Over the summer, a member of the project studied how to create a chatbot of their own, and discovered that no matter how much data a user may give it, it will never always be 100% right. They did mainly test open source models such as Llama-2 and used open source platforms such as turbo pilot, but this is also evident in commercial offerings. An example can be shown in *Figure 51* below. The prompt they asked was "code me a application that can connect to a Arduino through Bluetooth, on Android, that can turn a light on and off, check the temperature, connect to Bluetooth music, all while the Arduino is feeding data into the microcontroller that is controlling each piece. Its response is below.

```
1. Set up your Arduino:
You'll need an Arduino board with Bluetooth capabilities, such as the Arduino Uno with an
HC-05 or HC-06 Bluetooth module. Write code for the Arduino to control the lights and
read temperature. Here's a basic example:

cpp Copy code

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(2, 3); // RX, TX

void setup() {
  BTSerial.begin(9600);
  pinMode(4, OUTPUT); // Control the light
}

void loop() {
  if (BTSerial.available()) {
    char command = BTSerial.read();
    if (command == '1') {
      digitalWrite(4, HIGH); // Turn on the light
    } else if (command == '0') {
      digitalWrite(4, LOW); // Turn off the light
    }
    // Add temperature reading code here
  }
}
```

Figure 51: Chat GPT Error Coding Example

It does not give the user the environment, the needed ports, or all the code they asked for. If they placed this in their Arduino's IDE and ran it then nothing would happen. This is also true for the part about the application. Here is what it says in *Figure 52*.

2. Create the Android App:

You can use Android Studio for this. Here's an outline of what the app should do:

- **Pair with Arduino:** Use the Android device's Bluetooth capabilities to connect to the Arduino. You'll need Bluetooth permissions in your AndroidManifest.xml.
- **Control Lights:** Create a user interface with buttons or switches to send '1' or '0' to the Arduino for turning the light on and off.
- **Check Temperature:** Display the temperature data received from the Arduino in a TextView or similar UI element.
- **Bluetooth Music:** You can use Android's Media Player or a library like ExoPlayer to handle music playback.

Figure 52: Chat GPT App Example

It did not create an application, and in the environment I was using AndroidManifest.xml isn't going to even be in the project. This also leads to another issue that will be addressed in the next section.

5.2.2 Prompt Engineering v.s Learning

Students are leaning towards prompt engineering, or learning how to construct a question to get the most out of the ai tools, instead of actually learning the material they are searching. Prompt engineering is a powerful way to use these tools, but it takes away the key thing for learning, especially senior design. An example is trying to ask it "write me a detailed plan on how to make a smart cooler." It will give a person a plan which is fake and won't work, and the students would blindly follow it. This can lead to them not actually understanding what they are doing and inevitably fail senior design.

5.3 Conclusion

These tools are one of the most popular things being researched, but they are tools not replacements for understanding and experimenting. Senior design is meant to help students learn how to work in a team and build a product so they can use this in their professional careers. If they just used ChatGPT or other AI tools to do it for them, then what is the point for paying and taking the class? If it is just for the degree, then really there is no point in going to college. Students are well aware of plagiarism and copying, so it is their job to use the tools to help learn and not replace it. If used for that purpose, then AI tools really are powerful tools.

6. Hardware Design

This chapter will include all the details associated with the hardware section of this project. This can include subsystem block diagrams for a specific part, schematic diagrams, wiring diagrams, and structural illustrations. The focus is not on the function, task, or code of each part, but more of physical wiring connections and how to turn on that component to get the task that a person wants completed. Each section is labeled for each hardware component and will incorporate associated figures.

Below is *Table 23*. It shows the labeling, describing, and indicating connections of all of the pins on the ESP-WROOM-32. There are a total of 30 pins on our particular ESP32 board.

ESP-WROOM-32 Pin 1-30	Description	Connection
1 (VIN)	5V Input	LED Lights (+) Temperature Sensor (+)
2 (GND)	Ground	Speaker (GND), LED Lights (GND))
3 (T4, ADC14, GPIO13)	Internal Capacitive Touch Sensors, Analog to Digital Converter, MAX 40mA,	Temperature Sensor (Data)
4 (T5, ADC15, GPIO12)	Internal Capacitive Touch Sensors, Analog to Digital Converter, MAX 40mA,	N/A
5 (T6, ADC16, GPIO14)	Internal Capacitive Touch Sensors, Analog to Digital Converter, MAX 40mA,	N/A
6 (T7, ADC17, GPIO27)	Internal Capacitive Touch Sensors, Analog to Digital Converter, MAX 40mA,	N/A
7 (DAC2, ADC19, GPIO26)	Digital to Analog Converter, Analog to Digital Converter, MAX 40mA,	Speaker (+)
8 (DAC1, ADC18, GPIO25)	Digital to Analog Converter, Analog to Digital Converter, MAX 40mA,	Speaker (-)
9 (T8, ADC5, GPIO33)	Internal Capacitive Touch Sensors, Analog to Digital Converter, MAX 40mA,	N/A
10 (T9, ADC4, GPIO32)	Internal Capacitive Touch Sensors, Analog to Digital Converter, MAX 40mA,	N/A

11 (ADC7, GPIO35)	INPUT ONLY, Analog to Digital Converter, MAX 40mA,	N/A
12 (ADC6, GPIO34)	INPUT ONLY, Analog to Digital Converter, MAX 40mA,	N/A
13 (ADC3, GPIO39)	INPUT ONLY, Analog to Digital Converter, MAX 40mA,	N/A
14 (ADC0, GPIO36)	INPUT ONLY, Analog to Digital Converter, MAX 40mA,	N/A
15 (RESET, EN)	3.3V Regulator Enable, Pulled HIGH	N/A
16 (GPIO23, VSPI MOSI)	MAX 40mA, Output Master	Keypad Column 3
17 (GPIO22, I2C SCL)	MAX 40mA, I2C SCL Channel	LED Lights (Data Cable)
18 (GPIO1, TX0)	MAX 40mA, Transmit Through UART	N/A
19 (GPIO3, RX0)	MAX 40mA, Transmit Through UART	N/A
20 (GPIO21, I2C SDA)	MAX 40mA, I2C SDA Channel	N/A
21 (GPIO19, VSPI MISO)	MAX 40mA, Input Master	Keypad Row 1
22 (GPIO18, VSPI SCK)	MAX 40mA,	Keypad Row 2
23 (GPIO5, VSPI SS)	MAX 40mA,	Keypad Row 3
24 (GPIO17, TX2)	MAX 40mA,	Keypad Row 4
25 (GPIO16, RX2)	MAX 40mA,	Amplifier Output 1, Keypad Column 1
26 (GPIO4, ADC10, T0)	MAX 40mA, Analog to Digital Converter, Internal Capacitive Touch Sensors	Amplifier Output 2, Keypad Column 2
27 (GPIO2, ADC12, T2)	MAX 40mA, Analog to Digital Converter, Internal Capacitive Touch Sensors	N/A
28 (GPIO15, ADC13, T3)	MAX 40mA, Analog to Digital Converter, Internal Capacitive Touch Sensors	Keypad Column 4
29 (GND)	Ground	Temperature Sensor (GND)

30 (VDD 3V3)	3V Input	N/A
--------------	----------	-----

Table 23: ESP32 Pinout and Connections

6.1 Overall Schematic

The overall schematic is a more detailed diagram that shows connections between components with a circuit. It incorporates resistors, capacitors, inductors, etc to show what each hardware part is made up of. This schematic will allow us to see connections on a higher level. This is simply a representative of the circuit, and not an accurate or true to size diagram. With this, we can truly understand how each component works. Each component is included within this diagram; *Figure 53* shows a schematic of our total system as a whole.

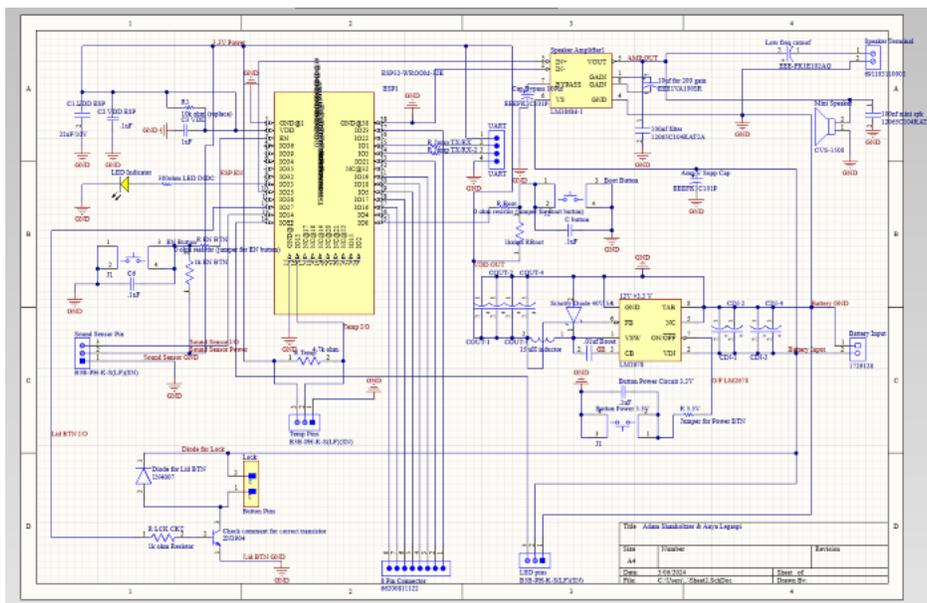


Figure 53: Overall Schematic

6.2 Hardware General Block Diagram

The hardware general block diagram is an indication of how each component is powered and connected together. This diagram is a high-level view which indicates how each part is connected to one another. This is a more detailed circuit in which all components and peripherals will be listed instead of grouped. In addition, all voltages required will be listed. This diagram is especially important in keeping track of which component is wired to where, as different components require different voltages. This block diagram is simply for connections and ensuring everyone has a clear understanding of where each component receives its power. These are all components that will be

drawing power from the Solar Umbrella and Battery. One thing to note is that this diagram will not show flow of information or data connections. For example, even though the temperature sensor and MCU are connected together, the Hardware General Block Diagram will not show that since this is a diagram of power and connections. The temperature sensor is in fact connected to the MCU through a data pin, however in this diagram it must be turned on first. This diagram will focus on highlighting instances of that sort.

As we go through the block diagram, we start with the Solar Umbrella, which is our first hardware component. This will send power to the battery through the Solar Charge Controller. From there, the battery will be wired to the PCB. There will be 3 different voltages: 12V, 5V, and 3.3V. From those 3 voltage levels, we can use that wide range to power all of our hardware components. The audio amplifier module and the solenoid lid locking mechanism requires 12V. The LED lights and the USB ports require 5V to work. The temperature sensor and the MCU require 3.3V to work. This can be visualized in the figure, *Figure 54*, below.

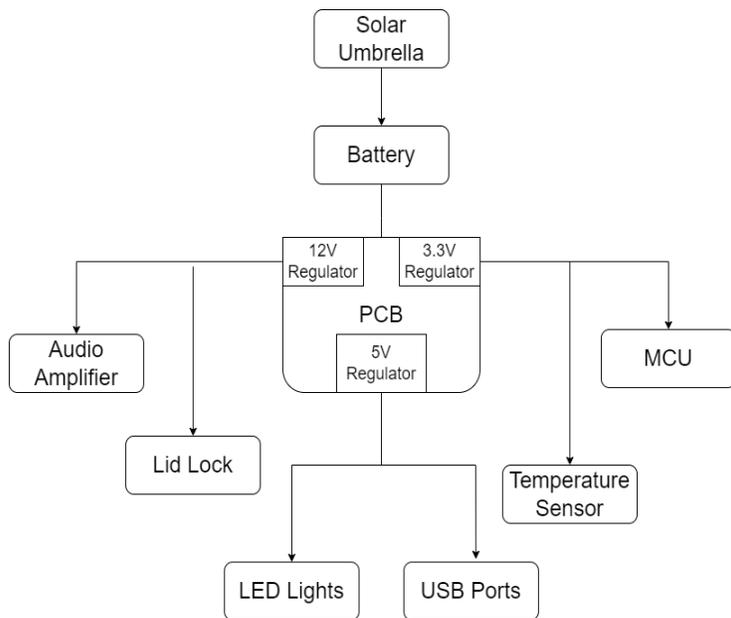


Figure 54: Hardware General Block Diagram

6.3 Power Delivery System

This diagram shows the power delivery system for our entire Smart Cooler. With this, we can see the flow of power throughout the system starting from inputs to outputs. In addition, we can also see how the power and voltage is being regulated throughout. We have also implemented layers of protection This is also showing the protection schemes embedded within to prevent any overcurrents, overvoltages, frying of components, and to maintain the overall safety of our team.

We can also use this block diagram to try and isolate faults. There is a solar charge controller that contains internal fuses for circuit protection since the circuit will be handling many different voltages. Another layer of protection is the battery itself. There is a built-in 12A battery management system. This will protect the circuit by balancing the charge and discharge current and cycles of the battery. In addition, since solar power is very unpredictable, we can have very high voltages at one point and at the next point very low voltages. We want to make sure this power is regulated throughout. An additional level to protect is the implementation of a voltage sensor. This is a simple voltage divider. By wiring this voltage sensor in parallel, it will receive the same voltage as the other branches. With the voltage divider, it will be able to step down the voltage and read it, then send its data over to the MCU. This will be used in conjunction with the software section and AI Voice to indicate low battery and manage the lid locking system. Now on the PCB itself, it will be taking in a lot of voltage and current. We also want to make sure this is protected, so we will have included three different regulators. There will be a 12V regulator, 5V buck regulator, and 3.3V buck regulator. *Figure 54* contains the 3.3V regulator, in which the other regulators will model after. This will power all of the peripherals and everything else needed within the cooler, so this must be double checked to ensure safety and accuracy. If this specific regulator / output fails, all components connected and tied to those pins are at risk of failure. In the event of any overcurrents and overvoltages, the team is able to locate the issues a lot quicker when we have diagrams to sort through. If we have an issue concerning one section within the 12V regulator, this makes it easier for us to isolate and protect the other equipment and components while simultaneously identifying the component's faultiness at hand. The figure below, *Figure 55*, shows the diagram of the power system. Below that is *Figure 56*, where the schematic for the power delivery system is shown.

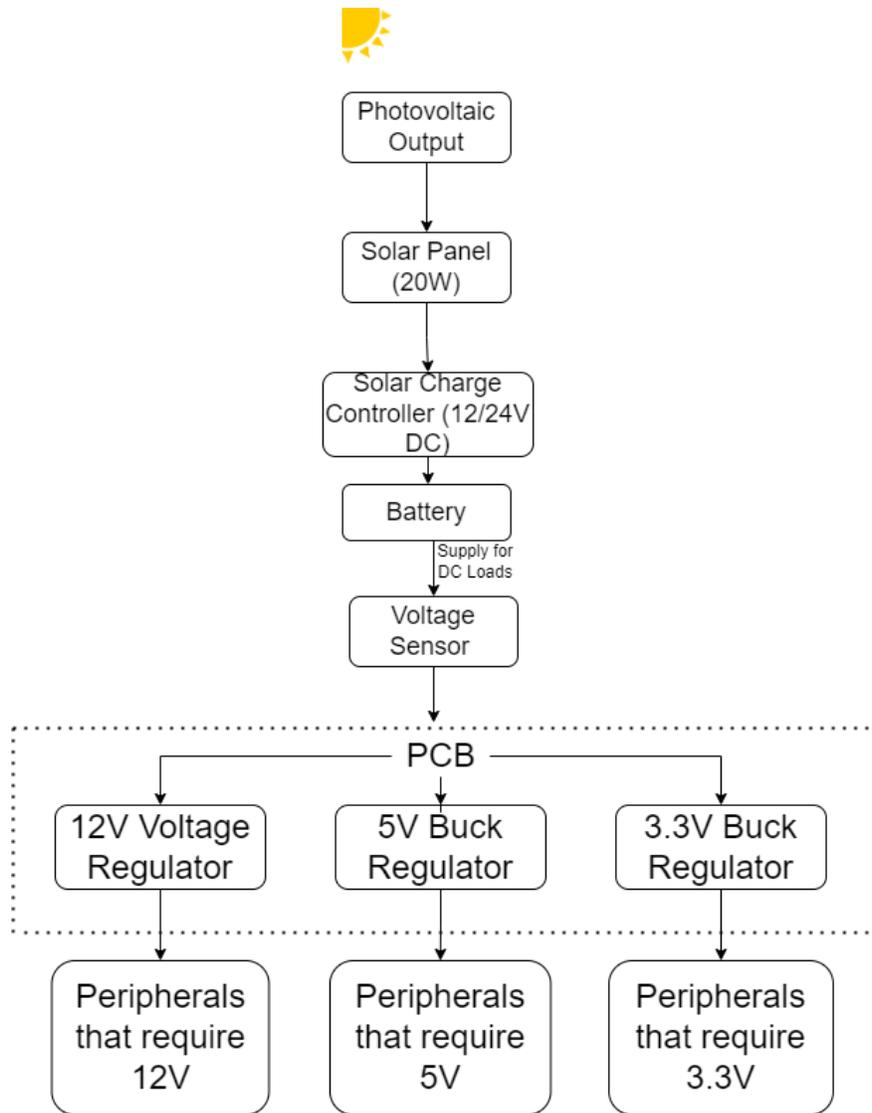


Figure 55: Power Delivery System Block Diagram

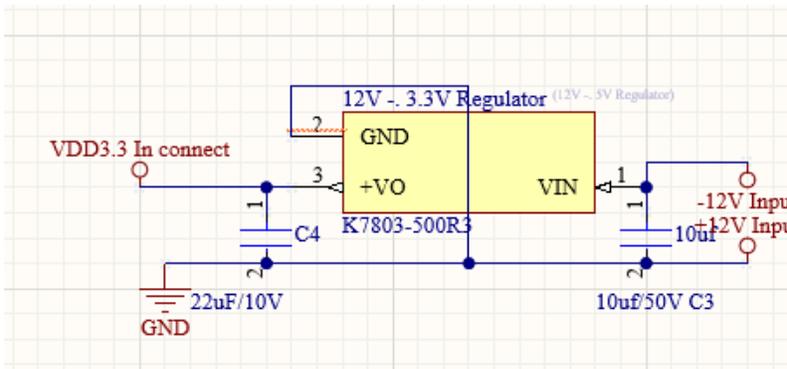


Figure 56: Voltage Regulator Schematic

6.4 Solar Umbrella and Solar Controller System

The figure below, *Figure 57*, indicates a diagram of how the Solar Umbrella and Solar Controller should be wired. This wiring diagram is especially important because it not only indicates connections, it also indicates the physical wiring itself. When constructing, it is very important to understand and remember the terminals and rated voltage levels you are handling. In addition, this system contains the highest rated voltage levels in our entire system.

In addition to those wiring connections, this diagram is important for knowing the steps to connect these components together. There is a very specific way to connect and disconnect the system to prevent overcurrents and overvoltages concentrated in a specific area. All of the load and stress being placed in one area can lead to faulty connections and damaged equipment. The steps to connect are listed below.

1. Screw in and connect the positive and negative terminals of the 12V Battery to the Solar Charge Controller.
2. Screw in and connect the positive and negative terminals of the Topsolar Solar Panel to the Solar Charge Controller.
3. Connect any load to the positive and negative terminals of the 12V Battery. For our project, these two wires will lead into the voltage buck regulators on the PCB.

These steps are to connect the entire system together. To disconnect, you must follow these steps in reverse for the most optimal circuit system protection.

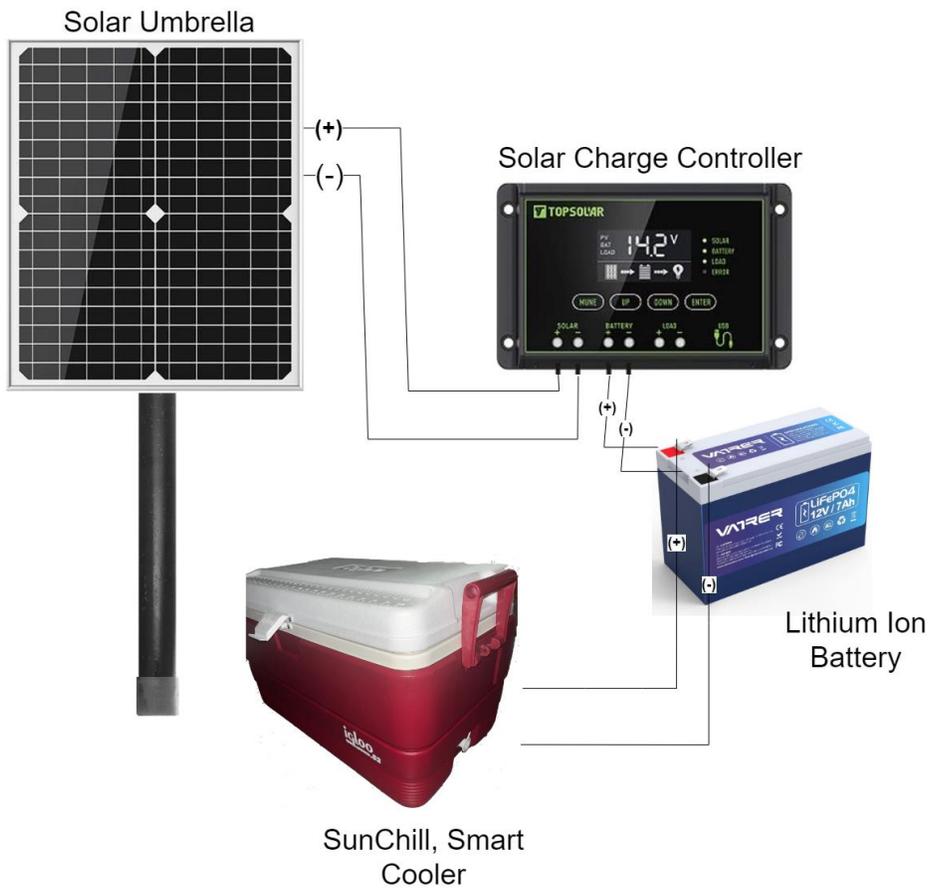


Figure 57: Solar Umbrella Wiring Diagram

6.5 Lid Locking Mechanism

The lid locking mechanism is a simple yet elegant design that is being implemented onto our smart cooler. There are two variants to how our lid will be locked, unlocked, and customized. The first is through the use of the physical keypad that will be attached to the side of the cooler, the second is through the use of an application downloadable on a smartphone that resembles the interface of a sixteen digit four-by-four matrix styled keypad. No matter which interface is used to unlock the smart cooler's lid, the logic between them both remains the same and is the following:

The user presses a trigger button to tell the system that a passcode is about to be entered and be prepared to read and compare data. If the passcodes match, send a signal (12V) to the locking

mechanism from the battery to unlock the lid for a given period of time until that time has elapsed in which case the locking mechanism will relock itself. If the passcode is incorrect, keep the locking mechanism locked. This can be shown in the figure below, *Figure 58*.

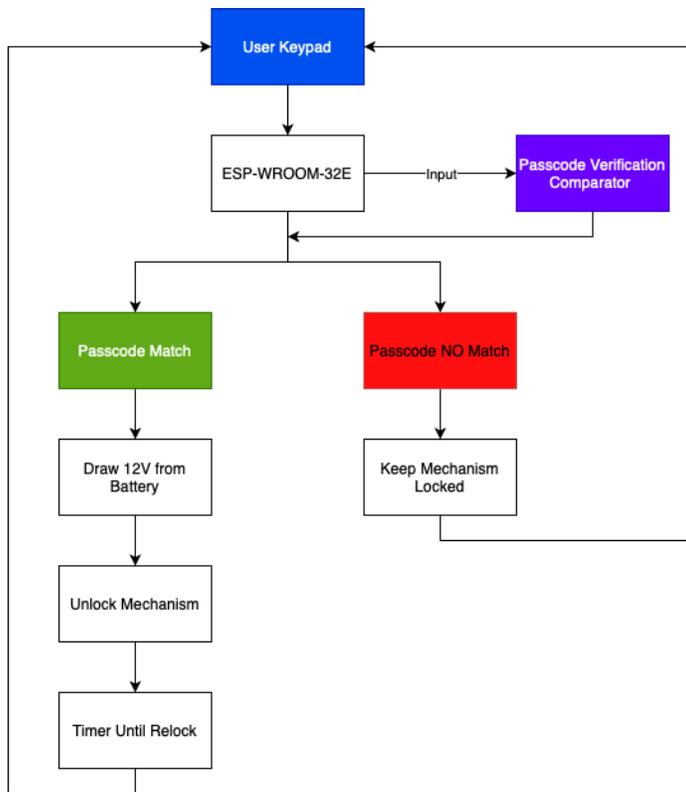


Figure 58: KeyPad Hardware Block Diagram

The diagram below, *Figure 59*, demonstrates the pin layout circuitry of the locking mechanism. As a person can see, there are 8 pins that connect the sixteen digit four-by-four matrix keypad to the ESP-WROOM-32E. The pin connection points match the array logic in our test code and correspond to the correct row and column internal pins of the keypad. Following the input from the user on the keypad and after approval from our code within the ESP32, the GPIO1 pin will signal the voltage regulator to allow 12V to be drawn from the rechargeable battery. At this point, the 12V will be directed toward the locking device which will unlock the latch from the provided hatch (also shown in the diagram for convenience). This is the completed circuit of the entire locking mechanism. It is very straightforward and thankfully has been backed by many

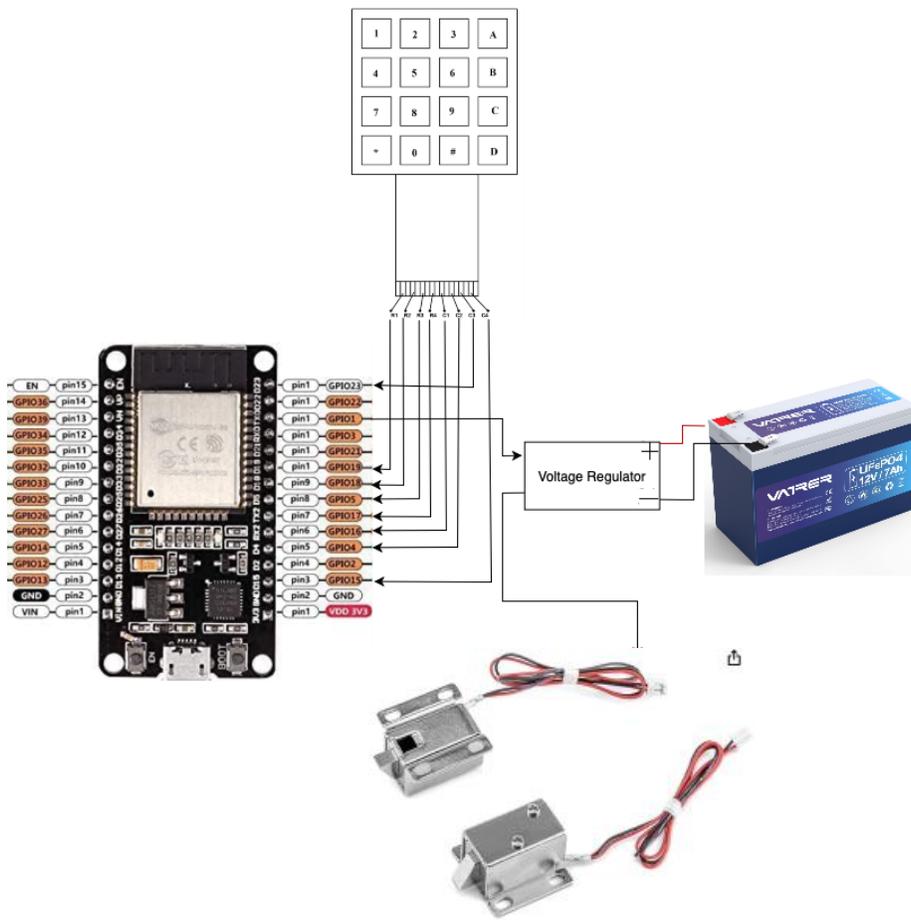


Figure 59: Hardware Physical Diagram

users in the past that have tried similar designs, so we as a group can be confident that this design will work when we begin our in-depth component testing section. In order for this whole process to work, the system needs to receive one key input at a time or else the ESP will buckle and not accept any digits. If this happens then the locking device will not receive any power and it will read as a failed attempt at unlocking the entire mechanism. And as stated in a future chapter, after so many failed attempts at unlocking the entire mechanism, the system will temporarily block the user from trying again until a certain amount of time has lapsed.

6.6 Speaker and Audio Amplifier System

In this schematic, we are able to see which pins are used by the MCU, the op amp and pins of the LM386, and how those pins are connected to the actual speakers. The MCU GPIO data pins will be outputting an audio signal as a waveform. This is connected to the LM386 audio amplifier positive input, pin 3. Since this audio signal is simply a waveform, the negative input pin 2 can simply be connected to the MCU's ground. The audio amplifier will be powered on through the Supply Voltage, Pin 6 and the Ground, Pin 4. Even though Pin 2 and Pin 4 are both connected to a ground, we have connected them to separate grounds. This is for added hardware and MCU protection. Since the audio signal is extremely low voltage and only an input, the negative input tied to ground has relatively low electricity flowing through it when compared to the Ground on Pin 4. That pin is used to ground the Supply voltage, which will be 12V. This is a very large discrepancy, so we are going to want to separate those two wires and not tie them together.

The audio signal is very quiet, so the waveform needs to be amplified through the operational amplifier in order to be heard. The passive speakers require this amplification. Passive speakers do not require any power to be turned on, they simply convert the electrical waveform into mechanical energy. This part is the voice coil, and that voice coil is able to vibrate the air which creates a sound wave from the original audio input signal. We will use the output, Pin 5 to connect our passive speakers through.

Pins 1 and 8 are Gain. This will adjust how much the signal is being amplified by up to 20X. In the audio module, there is an included gain knob already attached for ease of usage. Pin 7 is the bypass capacitor. This will divert the noisy frequencies away from the sensitive components of the circuit. This is especially important to retain the purest audio input and reduce buzzing being heard. This will improve the signal's integrity and allow the signal to be transmitted and received at its best. Since the gain can be amplified by so much, we need the bypass capacitor to be able to handle those signals. Buzzing may not be heard at low volumes, but once amplified it can prove to be an issue so we want to be able to combat that within our PCB design.

For further hardware wiring, we will use two audio amplifiers for each set of speakers for that signal integrity. We do not want to split the voltage and use two wires at one pin, as we want minimal additional connections. In doing this, we can isolate the amplifier and speaker and see each as a pair. This will allow for the best audio output. This diagram is shown in full in *Figure 60* which is shown below.

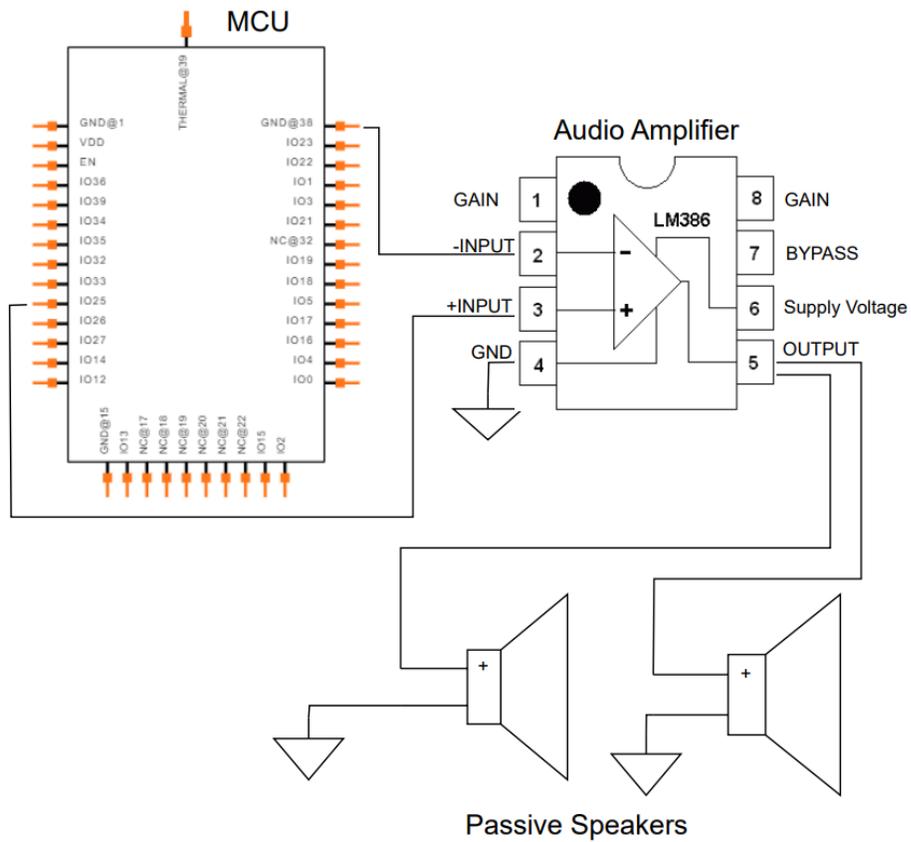


Figure 60: LM386 Low Voltage Audio Power Amplifier Chip Integration

6.7 Temperature Sensor System

Figure 61 shows us a small diagram of our temperature sensor circuit and how it is connected to our ESP32--WROOM--32E. We have 3 pins in our DS18B20, power, ground, and data. In our circuit it is necessary to have a 4.7k ohm resistor to bridge our power and data nodes together in series. This needs to be done so we can correctly collect temperature data, as proven in testing without it not present data is not recorded correctly. The sensor will not be on the PCB itself but this shows where the traces will be so they can be connected when soldering. We can dedicate any data pin to this.

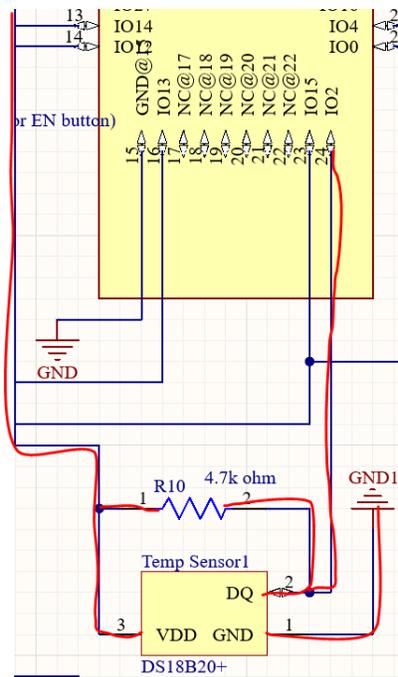


Figure 61: Temperature Sensor Schematic

6.8 LED Light System

Since the ESP32--WROOM--32E does not have a 5V output pin we will need to provide our lights with 5V of power. We believe this would be necessary even if we had the 5V output since we want to limit the amount of power going through the ESP32--WROOM--32E. This is a safe and easy way to provide power to our lights. With the addition of the regulator we could also power these lights with solar energy from our Solar Panel since they draw so little current. This is shown in *Figure 62*.

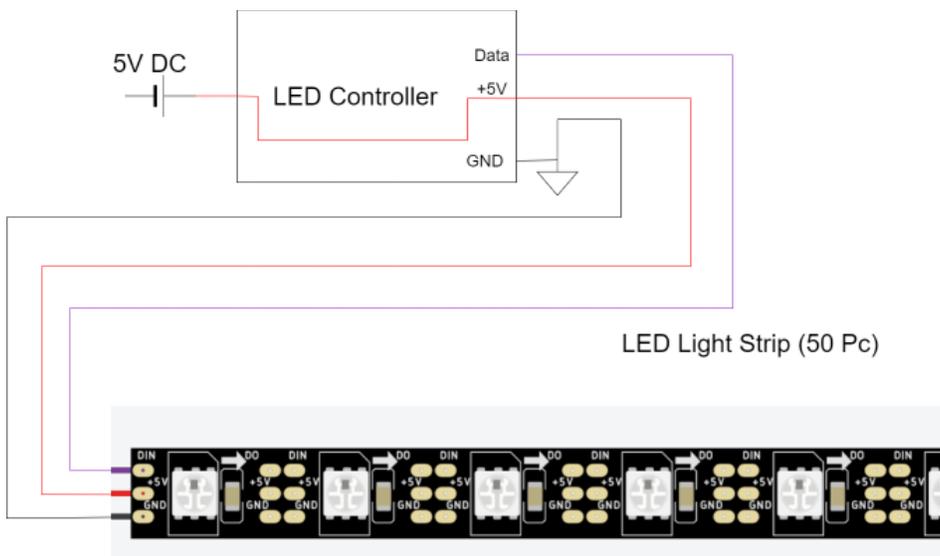


Figure 62: LED Lights Connected to 5V Output Regulator

7. Software Design

One of the project's key features is the Bluetooth application. From downloading the application, the user will be interfaced with a screen that will show off a variety of features to the users. From showing the temperature to playing music from the user's phone, all of these must be programmed in for the optimal user experience. Below will detail the coding plans for the application for each feature, along with any other details.

7.1 Software Inputs

The application has 3 kinds of inputs into the program. This involves the application's actual touchscreen, the sensors, and the keypad. The lights and speaker are just outputs where the application only gives a signal to confirm if the action has been done, so those will not be discussed in this section. But the remaining pieces will explain how each part will obtain data, how it's sent back, along with diagrams to help with explanation.

7.1.1 Sensors

Sensors that are involved in the cooler are the temperature sensor, and the voltage detection module. There is also the sound sensor, but that is a piece in between the lights and the speaker, so that will be discussed in a further section. Both will work in a similar fashion, so the temperature sensor will be used as an example. When power is given to the system, the microcontroller will give a signal out every x interval to the sensor. The sensor will then check the current temperature and send that data back to the microcontroller. This is a continuous loop, and the application will just check in to see the current status of that loop. The application is only taking the data and not sending anything back, as there is no need with these sensors. This process is shown in *Figure 63*.

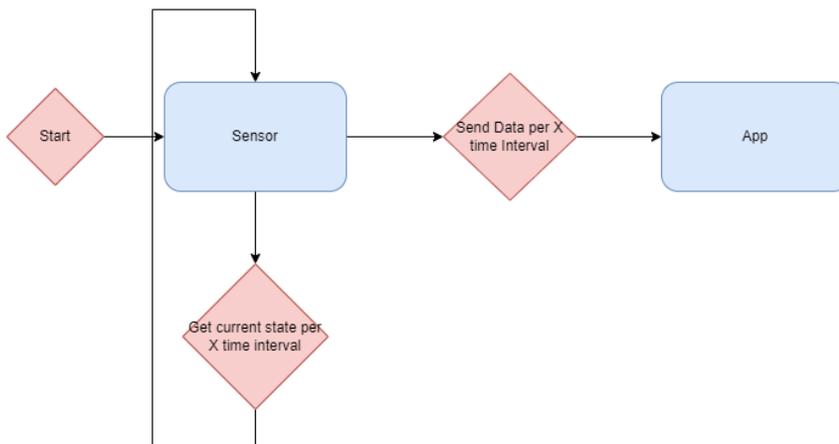


Figure 63: Sensor Block Diagram

7.1.2 Keypad and Lock

The keypad and lock's software have the same functions in both just the microcontroller software and the application. According to the diagram, the user will either type the passcode into the application or the keypad, and the correct pin will send a signal to the microcontroller to unlock the lock. This will be unlocked until a signal is sent to lock it, and the next time the cooler is closed the lock will remain locked. This is, unless the cooler loses power. If the cooler is about to run out of power, then it will send a signal to keep the cooler unlocked. This will ensure nothing necessary will be locked inside and if, for example, a user leaves their phone inside of the cooler and the cooler loses power, they can get their phone out. This process can be shown in *Figure 64*.

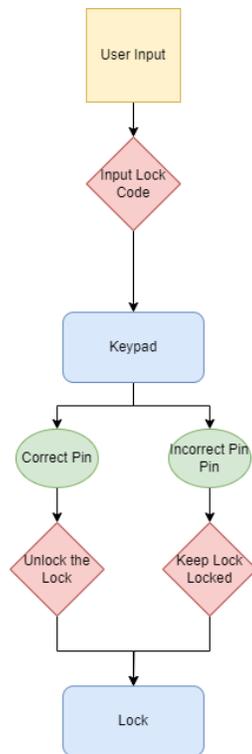


Figure 64: Keypad and Lock Basic Block Diagram

7.1.3 Bluetooth App

The final user input is the main one, the Bluetooth application. Whenever anything wants to be changed or accessed by the user, other than the keypad to unlock it, it is done through the application. The user will input which feature to check or access on the cooler and through Bluetooth it will send the desired feature to the microcontroller. The picture above is a simplified

version, but it's the general purpose loop. Note that the microcontroller will only send data without the user input to the application when only checking the battery. Other than that, the user needs to signal they want it, and the application alone is only the display to those features. This process can be seen in *Figure 65*.

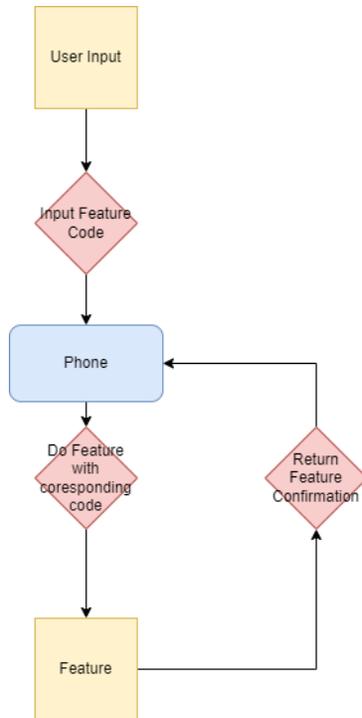


Figure 65: Bluetooth App Input Block Diagram

7.2 Application Interface

Our application is planned to be extremely straightforward and simple for any user to interact with no matter their level of experience with our application or any other computer application in the past. In short, anybody and everybody should be able to pick up our application and navigate it with ease. We will have simple menus that are not cluttered with options, buttons, or extravagant graphic designs that fill up the user interface. The main menu will have the following: Our SunChill Logo, a display icon presenting the user with the current status of the battery attached to our smart cooler that powers the whole device, a display icon presenting the temperature inside the cooler at all times for the convenience of the user, a music icon to control the built-in speakers allowing the user to play personally selected music, a light icon that gives the user the ability to control the built-in LED lights of the user, an AI Voice icon where the user can ask the AI voice

certain questions, and a lock interface that deals with the built-in locking mechanism of the smart cooler.

If the temperature icon is pressed, the user will be taken to a menu that displays the current temperature reading inside the cooler in real-time, along with a button that can trigger the AI voice to produce a current weather announcement of the user's current location.

If the music icon is pressed, the user will be taken to a menu that displays volume controls, an on/off button, and a display presenting the user with the current status of the connection of music to the smart cooler.

If the light icon is pressed, the user will be taken to a menu that displays a color wheel for the user to control the LED's output colors directly or a button called "Light Show" that displays multiple colors at random.

If the lock icon is pressed, the user will be taken to a menu that displays options to lock and unlock the cooler from a distance and also change the passcode stored on the cooler from a distance.

7.3 Coding Design

Below will be the sections detailing how the code will work with each of the features. The application's interface will be the front that will then intersect into each section where separate code on the application and the microcontroller will be run. Each section has its own interface, and function of code that the microcontroller will access. Most of the code will be the libraries used in the Arduino and the multiple cases that will switch to separate functions. The actual application is more just a front layer, so there is less coding for the application part and most of it will focus on the Arduino in the microcontroller. Note the light and speaker are separate into separate and together as the coding for the intertwined functionalities are entirely different.

7.3.1 Light Code Plan

For the lights, once the sections for lights are chosen then a display will show with an icon of Light options at the top of the screen. Right below it will be a button to turn on or off the lights and right below that a brightness slider. Then under that will have 2 sections that lead to further menus. One will be called Non changing mode and the other will be party mode. Party mode will be the mode that will allow the lights to change to multiple arrays of color patterns. There will be 3-5 options to choose from with multiple color patterns preprogrammed into the lights. They can be changed on the fly from pressing one button to the other to change into a different light pattern. The Non changing mode will have stagnant color options with multiple colors to choose from and ones with multicolored options.

The code for this section that is not display related will be regulated to a separate light function. There will be a main function that runs forever unless a user presses a button. This will change a variable where each function will be called. When the button is pressed for the lights, the variable will change and a forever loop will begin in that menu. In that menu, if the user wants to change the lights for when the user moves the slider a for loop will be done calculating the distance changed from the old location to the new location for the brightness. For the lights it just sends a

signal out to change the color and returns to the loop. For the light show, there will be another function embedded that will have a switch statement for each of the shows. When a button is pressed the variable in the switch statement is made and that code of the show will begin. Having a switch statement will make it easily able to switch between other programs so even if one is interrupted, it won't ruin the functionality.

7.3.2 Speaker Code Plan

The speaker display is simpler compared to the lights, as it is just an icon that shows the current music status and a simple menu. Before beginning the application, it needs to sink to Bluetooth, so the Bluetooth part of the speakers is already functioning. When a user begins playing music on the phone, the icon that says “waiting for music” will then change to “music currently playing”. When that button is pressed there will be a slider that just allows the user to change the volume of the music.

The code is dependent on the BluetoothA2DPSink library. When the application is connected a static function obtaining the baud, sample, and channeling rate is configured. Then when the `a2dp_sink.start` function is activated then a signal is sent to the application showing that the music is playing. With the slider menu the brightness is changed similarly to the volume, changing in a for loop with the old value of the slider and new value being compared.

7.3.3 Light and Speaker Plan

This part of the code is simple, as it has done many times in other projects. For the display, separate from the lights and speakers will be a mode called “Party Mode.” If there is no music playing, then this part will remain grayed out. If there is music, it won't be and if pressed it will then be sent to the microcontroller to turn on the sound sensor and begin the light and music reaction mode. The lights will react to the music being played on the speakers and the lights will change depending on the music. The music volume can be changed in the music mode, but if it gets to a quiet volume a banner will appear stating that the light may not be accurate.

For this, it will use the sound sensor, and use the Music Reactive Multicolor LED Lights library. For when the function begins, the Arduino sound sensor gets a threshold value, and if it doesn't meet the value (the volume isn't loud enough) then the lights will not display. But each light will gain a random value between 1 to 6 and with 6 different thresholds a light will light up. Each threshold has a list of lights it can become and a random value assigned. This will cause each song to truly be random and have a specialized lightshow whenever a song plays.

7.3.4 AI Voice Plan

The first thing we do is find a medium for which to perform all artificial intelligence related tasks and functionalities. There were many choices but we wanted one that was easily integratable into the app that we built through the MIT app inventor. After selecting the MIT App inventor, and the ChatBot extension, we bought access to use an API key through OpenAI services for ChatGPT functionality. we will need to integrate into our code a way for our ESP-WROOM-32E to communicate with the artificial intelligence on the app and the speakers. The basic idea is that when an action occurs on our smart cooler, for example, the cooler lid unlocks or the smart cooler's

battery reaches 20%, we initiate the code that writes key words (“Lid Unlocked”, “Battery at 20%”) into storage for the artificial intelligence to read from and speak out loud through the built-in speakers. In order for this to work, of course, we need to define a set list of actions or commands that we wish to integrate into our smart cooler. A few have been mentioned already, but others can include “Music play/paused”, “The weather today is ___”, and “Battery fully charged”. Once this list of commands are complete, and an action occurs we will write our code to call a function that sends the corresponding command to our text-to-voice library application programming interface (API). One thing we need to consider is allowing our code and artificial intelligence to be asynchronous so that the interaction with the user does not become bogged down by a tight pipeline of information. Most likely we will need to trigger a request using the ESP32’s ability to connect to Wi-Fi that requests from an artificial intelligence voice library api to receive the information that is displayed through the speakers or the device of the user. This process can be shown in *Figure 66*.

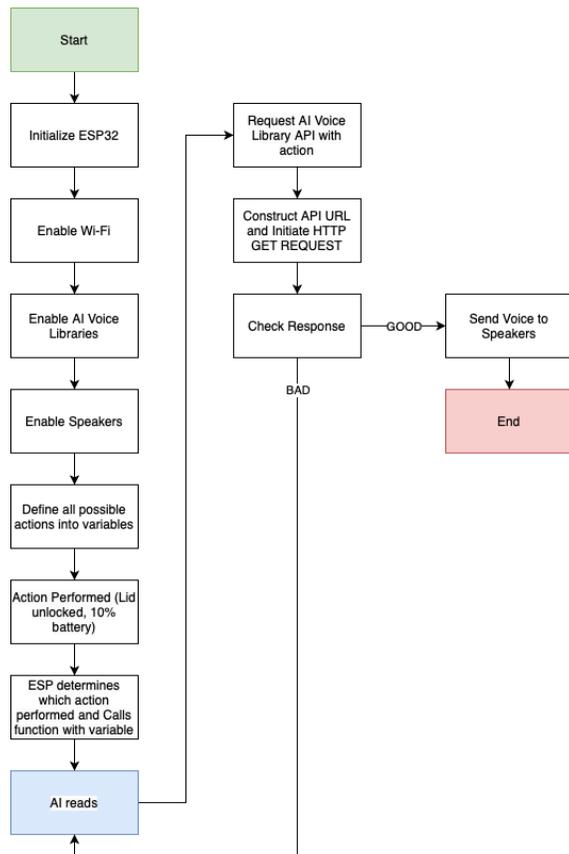


Figure 66: AI Voice Flowchart

7.3.5 Temperature Sensor Plan

The display for the temperature sensor will involve only a single menu. When the button is pressed for temperature, it will switch to the next menu where it will display the last temperature it scanned for. If the user is not happy with it or thinks it's wrong, they can press the refresh button for it to check the temperature again. The code for this involves the DallasTemperature library. With this, it goes to the requestTemperatures function and gains the temperature from the sensor in the MSP, and sends that number to the application where it prints out on the display.

7.3.6 Keypad Code Plan

In order to use the keypad as a device that can store a passcode used to lock and unlock our cooler lid, we need to first identify how the keypad works in tandem with the code.

At default (no power and no button input), all of the row pins are held LOW and the column pins are held HIGH as seen in *Figure 67*. Once a button is pressed, the column pin connected to the button press is pulled LOW shown in *Figure 68*. The Arduino code then sweeps through each row to HIGH and reads the column pins to determine which one gets pulled back to HIGH demonstrated in *Figure 69*. Once the Arduino sees both a row and column being pulled to HIGH, it can determine which button was pressed to make that occur. That is the very basic premise of how the button pressing works on a 4x4 matrix keypad. *Figure 70* confirms which button on the keypad has been pressed once the procedures above have been completed.

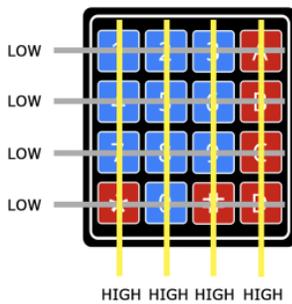


Figure 67: Buttons General Picture

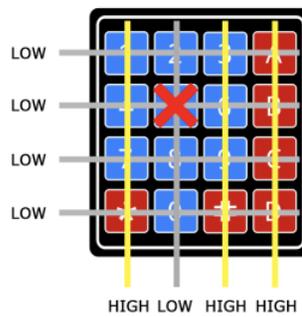


Figure 68: Buttons General Picture 2

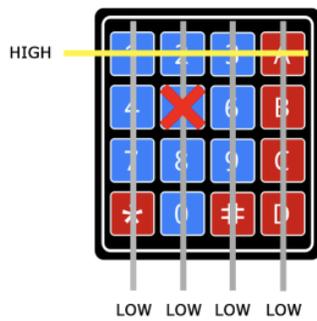


Figure 69: Buttons General Picture 3

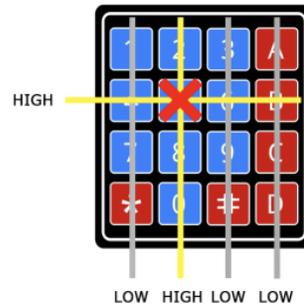


Figure 70: Buttons General Picture 4

Then, we can identify the purpose of the buttons on the keypad within our code. For example, we can use the meaningless star or ampersand symbols as trigger buttons. Our code will always be looking for the input of those two buttons. One will trigger a passcode that is about to be entered that is already stored on the ESP-WROOM-32E, and the other can trigger that a new passcode is about to be inputted to replace the old passcode. For the star button we simply need to read the input of the button that calls a function to read and store the following numbered inputs into a value labeled "Passcode". Once this passcode was inputted, we can prompt our AI voice to ask the user to repeat the passcode for verification. The AI voice will then alert the user that the passcode has been stored and is ready for use. If the ampersand button is pressed, the AI voice will announce to input the old passcode in order to move on to storing a new passcode. If the passcode is verified, our code will route to the same steps of setting up a new passcode mentioned above. Overall, once a passcode is defined, we can begin writing the logic for utilization of the passcode such as reading the inputted numbered keys and comparing those to the stored passcode. If they match, then call for the lid to be unlocked and trigger the AI voice, if they do not match, play an error sound and keep the device locked. If after the passcode is entered correctly, the logic should only allow the lid to be unlocked for a certain duration of time both for security purposes and efficient battery allocation. And finally, the code must keep track of the number of passcode attempts given in a specific time frame and/or time in between each attempt as a preventative measure against brute force attacks on the smart cooler and its locking mechanism. If there are too many attempts made unsuccessfully within a given time frame (say five unsuccessful attempts made within one minute), then the ESP-WROOM-32E will lock the entire system (disallowing input, or rather, disregarding input from the keypad) for a time period of five minutes. This error will also be announced through the artificial intelligence voice system to the user so they are aware of the issue and do not continue

attempting to unlock the device using the keypad on the smartphone or the physical keypad that is attached to the side of the smart cooler. This process can be seen in *Figure 71*.

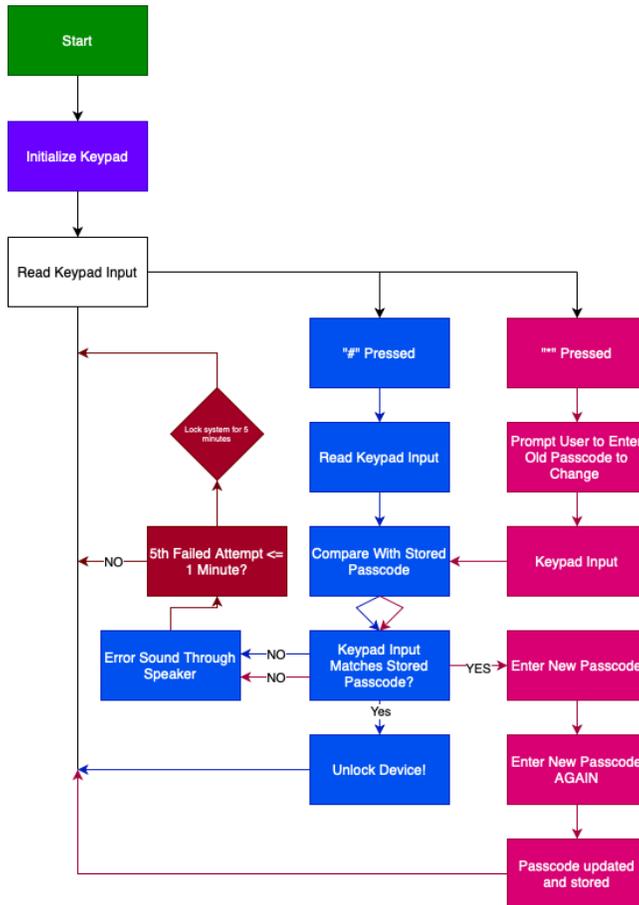


Figure 71: Keypad Detailed Code Flowchart

7.3.7 Total Software Part Integration

All of the parts described before need a central function that allows all of the other functions to work as one cohesive program. The application's display works under a central application using MIT Maker, but the actual backend is the key for how the microcontroller functions. This section will describe the hierarchy of how all the parts will work together.

The first part will be the initialization. The AI voice, speakers, Bluetooth, and battery connected will be initialized in a static void function. After the initialization is the forever loop faze in the main function. At the top will have the baud rate initialized as it may change for light and ai interactions. Then there will be a variable to be the state variable. This will be used like a menu, with that value being the key between which state of the menu. There will then be a variable to control which option the user chose. The main variable is just to control how many menus embedded the user is in so if any sudden changes or glitches occur, the microcontroller can adjust and return to the beginning state for any possible bugs. There is also a variable to control the selection the user made, and is reset after leaving the selections. These will be kept under switch statements, as if statements can lead to errors and double states. Also having 2 variables instead of one to keep tabs on the other states will ensure less bugs will occur.

Each statement on the menu is when all baud or other changes for the libraries will occur. Instead of having the changes embedded in the separate functions, having it done at the start will be best for experimenting and being able to change on the fly any issues from testing. It also allows for better testing of specific functions so when a developer is coding instead of said feature, they won't need to worry about messing any other features up or the initialization, just the actual use.

This function is also integral for when the power almost reaches zero. For if the cooler's battery goes to zero or somehow power is suddenly taken out, it needs to be able to ensure the lock unlocks and it motions to the user that the cooler is out of power. The main function will, no matter what sub function the user is in, always check the battery every x intervals. If the battery hits 10% or less, it will motion to the user in a banner that the battery is low on power, and that banner will remain until the battery goes above 10 percent or if the battery dies. When the cooler hits 3% or less, this will motion to go to the battery depletion function in the main function. This will then cause that function to automatically unlock the cooler and for as long as that battery is that low, there would be no way to relock the cooler for safety reasons. It will also cause another banner to appear in red. It will state that, in bold letters, that the power is critically low and to charge immediately. It will also state that locking will be unavailable. As mentioned previously, the AI voice will also speak through the speakers of the charging in case the user does not check the application during this time.

8. System Fabrication/Prototype Construction

In this section, we will be going over basic PCB design and other details related to the construction of our Smart Cooler Prototype. This will show our PCB layout, the internal and external configuration of our coolers construction.

8.1 PCB Design

During PCB creation, we decided to go with an Altium designer. Since group experience in PCB design was minimal, our circuit design was adequate enough so this would be the best application for us to use as we could refer to help from peers and other resources for basic setup. The hardest part about designing the PCB was first learning the software itself. A member of the team personally spent 12-16 hours working and designing the PCB. With all new software the biggest learning curve is learning a new user interface as things that may seem standard to one software may require special keybinds in another. The first thing that was done was try to find the closest ESP32 that matched our own. We did find the exact ESP32, but it was not in-stock and we cannot know when it will be back in stock. So the decision was made to go with the ESP32--WROOM--32E by Espressif Systems. Below is *Figure 73*, the ESP32-WROOM schematic.

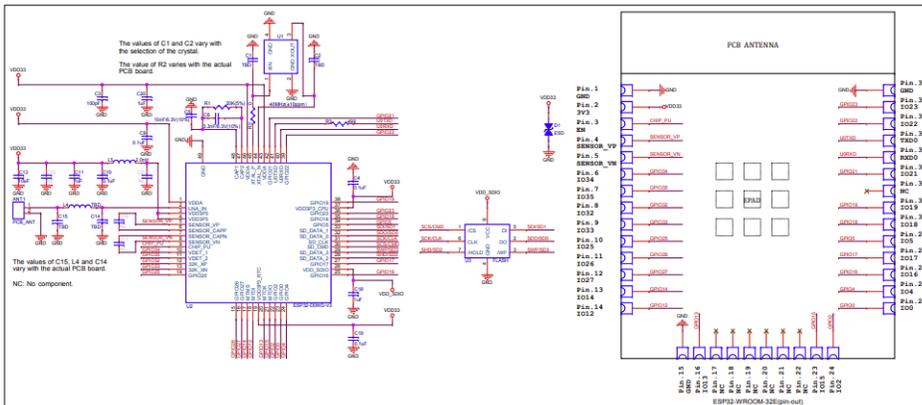


Figure 73: ESP32--WROOM--32E Schematic

The next thing was to look over the data and determine which pins and peripherals we would need for our chip to use it for our purposes. What we wanted to do was make something very similar to the ESP32 chip we used in testing while retaining and adding more functionality and usability than we had before. The first step of the design process was adding the ESP32--WROOM--32E schematic, footprint, and 3D model. This was done by downloading the files from SnapMagic formerly known as Snapeda. After this was done, these files were extracted and brought into the workspace in Altium in the proper locations. After this step the footprints were made and 3D models were attached to that footprint. When making the schematic referring to the Datasheet was essential as there were many useful tips and pinout info that was needed to create our PCB. Such as adding the peripherals such as our buttons, voltage inputs, grounds, voltage regulator, UART and JTAG connections, etc. This was done by looking up the correct part in Altium then referring

to the datasheet and connecting the correct pins. An RC delay circuit was added to the EN pin of our ESP32 as it was recommended by the manufacturer. Other additions are the voltage regulator circuit. Below is *Figure 74*, where the layout of our PCB Design is shown.

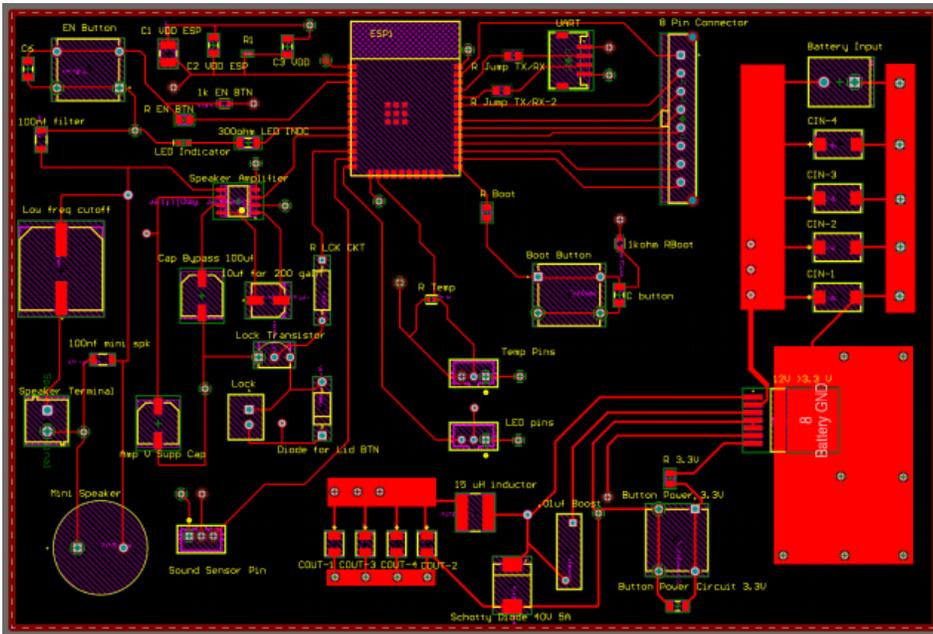


Figure 74: 2D Layout of the PCB Design

After the 2D layout was complete, all changes were ported over to the PCB section of our design in Altium where we began moving all our added components into a proper PCB layout. This was done by taking each component and moving it into the desired space with all other related components. The most difficult part of this was the ground was automatically being shared, which isn't actually an issue but we want to keep things as close to the 2D diagram as we can.

After this was done, we first used Altiums clever auto-routing trace tool to clean up the traces for us and that is how we got our first result. However, we quickly realized that manually routing each trace was essential. This allowed us to have more control over each route and properly allocate enough space for testing. The most important item is that manually routing allowed for all signal traces to be on the top layer only. Auto routing even with specific rules was not able to do this. Our finished 3D render from Altium is shown in *Figure 75*. Below that, we have the completed and fully soldered PCB shown in *Figure 76*.

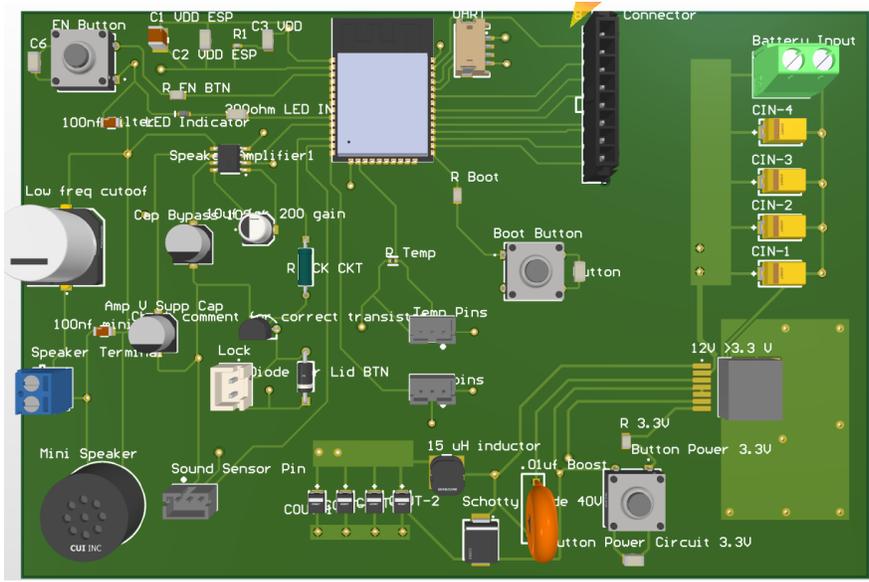


Figure 75: Completed 3D render of PCB Design

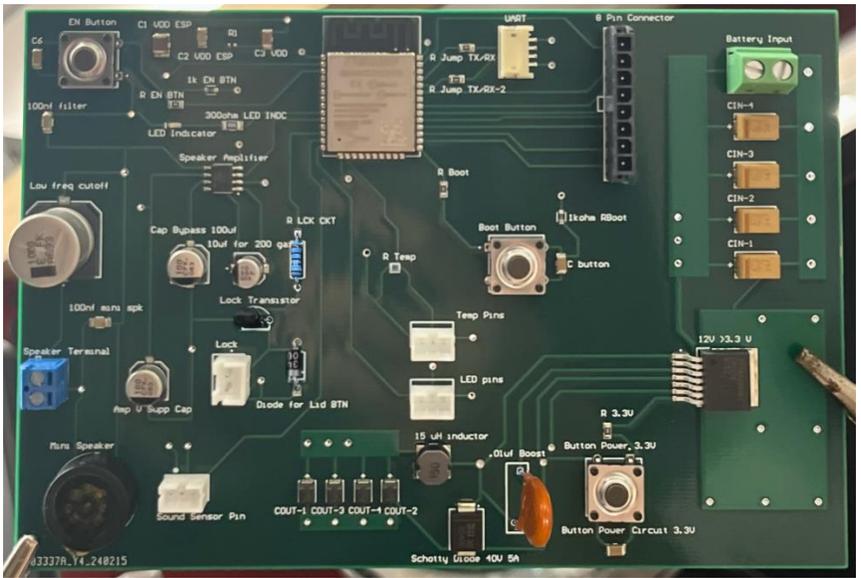


Figure 76: Completed Physical PCB

8.2 Cooler Configuration

We made a completely internal system with only minimal and necessary components on the outside. This was done through various means, as different components required a differing build plan even if they are in the same general location. We decided to take this route since the cooler will already protect our electronics from the outside elements and we only had to ensure proper sealing and wiring methods on the inside. This also gives us the benefit to monitor internal temperature of not only our storage but electronics and battery to ensure safe operation.

8.2.1 Overall Construction

In this section, we have illustrated the interior and exterior construction of our cooler and how we plan on attaching our components to our cooler. It's important for us to show how we plan on implementing the components into our cooler and what techniques we may use to do it. This is an important step for our prototype as we may find or be advised to pursue different methods of construction. We will be using an Igloo Legend 52 Quart cooler as our prototype and testing many of our design and construction practices in it to ensure they hold up in our final design. The cooler used is shown below in *Figure 76*.



Figure 77: Igloo Legend 52 Quart

8.2.2 Internal Design

Here, we will be showing basic diagrams of the inside of our cooler and how we plan to implement our hardware on the inside. To begin with any cooler we use we will be placing the electronics on the side opposite to the cooler drain plug as we want to retain that functionality to the user for normal use. As seen in *Figure 77*, we will have our internal electronics on the opposite side of the drain plug. This is just a reference to determine which side all major components will go on. This side view provides us with insight as to how we plan to mount the solar charger controller, battery, and circuit box.



Figure 78: Internal View

We will have 3 separate areas, as we can see in *Figure 78*. The first being our electronic holding area, the second being our water/air tight seal to keep things separated and dry, and lastly our user area or basic cooler. These sections illustrate how much space we plan on using for our prototype, and how we have divided up the space in our cooler section. Since our battery is the largest component and that is relatively small, we believe that we can allocate less space to our electronic holding area and give more to the user section. Section three takes about two thirds of our cooler while section one and two and about a third. Section two is our most important section as it will literally be the glue that holds together our cooler. This will be made of two sided insulated non-conductive material so that we can prevent any electrical discharge into section three of our cooler. Since our cooler already has a built-in ridge, this will make mounting that panel much simpler and secure as it will not be able to move as freely due to that ridge.



Figure 79: Cooler Internals

8.2.3 Front View Design

The figure below, *Figure 79*, shows the front design of the cooler. This front design view is to show how we will mount our last batch of components that will be placed on the front of our cooler. These include our speakers, USB charging port, and our lights. This is how we envision we will be placing these components but all is subject to change. All these components are front facing for ease of access and useability. Originally, we thought of placing our speakers on the side, but decided it would be better for the sound to go directly in front, behind, or to the sides of the user. This was decided since the front of the cooler typically always faces the user no matter where the cooler is placed. To start we will be placing the USB charging ports as high as we can, they are also placed on the side opposite of our drain plug where our component holding area is to reduce wiring lengths. This is also to avoid any additional outside elements from entering the ports. The same thing goes for our speakers; they will be placed higher to avoid any sand or dirt from entering them, but we have to space them out to provide the best audio volume output on the front of the cooler. The LED light will be mounted to a groove on the outside of the cooler that dips inward. This just happens to be the perfect height where we would want it and provides more space so the leds do not poke out as much. With that all being said all front mounted exterior components will be as flush-mounted as possible to provide a smooth exterior design.



Figure 80: Front View of cooler

9. System Testing

Our project, being a smart cooler, required multiple parts to make it function correctly. For SD1, our concern is more of the actual connectivity, with the extra features being more researched in SD2. Below will show the testing of the lights, buttons, locks, temperature sensor, speaker, and rechargeable battery. The section will also detail the ESP32 and how effective the software works. The final section will conclude with all the parts being put together to ensure that there are no disturbances with the pinouts and the power running through each component and its features.

9.1 Hardware Testing

The following section provides detailed explanations of all the hardware that has been rigorously tested, and will be used for our SmartChill cooler.

9.1.1 LED Lights

One of the first things to be tested were the ALITOVE WS2811 Addressable LED Pixel Light 50pcs. This is a 50 light strip that has each light connected to an address to work as a light show. For testing, we powered it using a ALITOVE 5V 3A 15W AC 100V~240V DC Power Supply Adapter Converter 5.5X. To ensure the device does not burn up, an Adafruit Accessories Quad Level-Shifter was bought but as there is only one strip, there was no need to use it. We used the Arduino IDE to allow the lights to turn on. For sequence testing, the Adafruit_NeroPixel library was used along with the FastLED library to test that all the LEDs function and that if a light show is possible. By testing a simple color changing example code, we saw that it worked as intended and all the lights turned on. Further tests will be made later on using the ESP32 and the speakers in sequence with Bluetooth functionality and speaker light integration. This testing is shown below in *Figure 80*.



Figure 81: LED Lights Testing

9.1.2 Temperature Sensor

The temperature sensor testing is shown below in *Figure 81*. The first thing that was verified was accuracy of the part data on the product page. All which was true based on testing. To start, sample code was run to confirm the sensors worked; After testing all five filters were confirmed to work even after one ground wire was damaged and additional wire was soldered on. The sensors had a very responsive response rate to temperature changes and we could see in real time temperature changes which would prove extremely useful for our usage. The sensor was also proven to be waterproof as the probe was fully submerged and provided accurate readings and no signs of leakage was apparent and the sensor remained functional after it was removed from that environment. The sensor requires a 4.7k ohm resistor within the circuit to function accurately. This resistor placed in series with the data line and connecting the same node as the power line is how this is obtained. It was also confirmed that all 5 sensors registered about the same ambient temperature, which confirmed that none of the sensors had an inaccurate reading. For the code that is referenced for this, look for the Temperature Sensor Code in Appendix A. For how the Temperature sensor is wired, reference *Figure 81* in Appendix A.

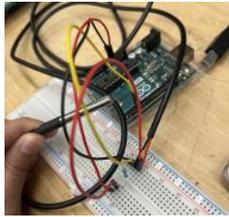


Figure 82: Temperature Sensor Testing

9.1.3 USB 2.0 Breakout Board

The first thing that was confirmed was that all power traces on the USB were intact and actually transferred power. This was done by soldering two wires to the D+ and D- traces, however any two pins could have been used since we are testing basic power. After this was done, power was then hooked up using our battery which we confirmed was putting out about 13.25 V, and that is the resulting voltage we got when we measured the output pins across the trace of D+ and D-. This power was confirmed via battery power and multimeter DC voltage testing. This was evident that our traces were intact and that we can receive power into this module for disbursement in or out of the system.

9.1.4 ESP32

As shown in *Figure 82*, the ESP32 was tested with the Arduino IDE. The ESP32, unlike other Arduino boards, requires more setup than the other boards. First, most computers do not have the driver to sense the Arduino, so the CP210x USB to UART Bridge VCP Drivers needed to be installed. Also, due to the board using a universal USB-A charger and not coming with one, it was difficult to find a wire that works. Most wires at home don't have the data wires configured, so a separate one needed to be obtained that had those wires soldered. The final hurdle was actually sending data to the device. If the driver is installed and if the right USB-A charger is used, the user needs to hold down the BOOT button while connecting in order for data to transfer. After that, it was simple for most of the other devices. This device was checked with the buttons, lights, and temperature sensor to ensure they worked. The results were successful so for the ESP#2's that were obtained, there were little issues with the pins. If there was an issue, the only one would be that the D2 data pin doesn't work well with inputs. This was shown when testing the 4x4 matrix buttons, where it shared outputs of what it was supposed to along with the pins of D19. So instead of showing the column lines of 369#, it showed 149*. This issue is a common occurrence with esp and the D15 line is also an issue, so both of those ports will be avoided unless necessary.

Commented [13]: Figure 80 and 81 have been removed



Figure 83: ESP32 Testing of Bluetooth Light

Another thing that was tested was the Bluetooth functionality, which is a key feature for the cooler. A similar process was employed of pressing the BOOT button and having the drivers installed, and it was found to be extremely efficient compared to a normal Arduino. A normal Arduino involves the use of an external Bluetooth module, and from originally using it to test the lights, it required external apps to get the module to function. For the ESP32, it only needed the Bluetooth serial library to be installed. After that was installed to the IDE and onto the ESP, the phone easily connected with the Bluetooth from the ESP (look at the picture for reference). As the ESP32 only used Bluetooth LE 4.2, that is incompatible with iPhones Bluetooth classic. This was already accounted for during testing, but it is something to note for continuing the project.

9.1.5 Keypad Testing

Before purchasing the keypad, we had the basic idea that in order to test the part we had to simply hook up a power source to the keypad and 8 LEDs to each pinout that would be used to tell us if each button was being pressed. At first thought, that is simple enough and it makes sense if we were testing if the pinouts and buttons on the keypad were working properly. However, after some thought, that method of testing neglects to include the “16” total buttons on the keypad – meaning we could test that we are getting signals from each pinout from the keypad, but we would have no clue which buttons actually work leading to those pinouts without trial and error and logging. To improve on our testing method, we decided to go the typical route which is to set up a code within the Arduino IDE and that specifically channels each button press through an array and assigns values to those buttons to be translated into actual information for the user (I.E., pressing button “1” translates to the computer reading and storing a value of “1”).

This method worked out because we got to simultaneously test coding our keypad onto the ESP-WROOM-32 that we purchased as well. The first step was to write a program that would receive input from the keypad through the registers on the ESP-WROOM-32, store those values using an array, and then print the values in the built-in terminal for us to confirm that the button pressed corresponded with the character displayed on the screen - confirming that the buttons, buses, and code all work in tandem with the ESP-WROOM-32.

We started off by referencing the ESP-WROOM-32 Pin Diagram to understand which pins would be best suited for our needs. As shown below, we have highlighted in YELLOW which pins we have decided to use for this testing.

After choosing those pins we needed to understand how the keypad functions and how each of its 8 pins needed to be routed once they are understood. The keypad is broken down into four rows and four columns that are appropriately labeled R1,R2,R3,R4,C1,C2,C3,C4 respectively.

All we had to do now was to connect our keypad row/column pins to the correct IO pins of our ESP-WROOM-32, write the code in the Arduino IDE, and test each button with all 5 keypads to make sure every button and keypad worked properly. This is shown in *Figure 83* below.

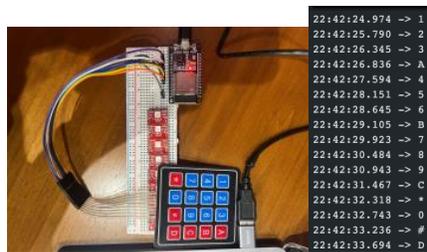


Figure 84: Button Testing

One thing to note is that while every button worked properly, there was an error ever so often with the second row of buttons on every keypad. This error only occurred rarely, but consistently; meaning once it occurred for a single button in Row 2, it would continue to occur for all buttons in Row 2 until the entire keypad was readjusted in orientation. We believe this comes down to low quality manufacturing of the keypads. To confirm this, we bought a second set of keypads from another company but from the same chinese manufacturer through Amazon. The second set of keypads had the exact same outcome, so we have concluded that this is a manufacturing error. All in all, it is not a major issue since it happens rarely and can be fixed by readjusting the keypad slightly.

9.1.6 Lock Testing

The lock we ended up testing was the ATOPLEE Electronic Cabinet Lock. It functions as a normal locking mechanism that holds onto the end of a bracket when powered off. However, once it is supplied 12V, in an instant, the hook that holds onto the bracket releases the bracket and “unlocks”. So testing this device is very straightforward and takes little effort. All that was needed to be completed was to hook up the positive and negative (ground) terminals of the lock to a power supply unit and turn the power on and off to demonstrate the ability and correct functionality. Below in *Figure 85* of the setup demonstrating the bracket being locked while no power is being supplied.



Figure 85: Lock Testing

If there is no ability to supply power to the locking mechanism and the bracket is fastly secured to the locking unit, there is a mechanical fail-safe that is designed into the locking mechanism to manually unlock the device and allow the bracket to be freed from the unit. In the testing, we demonstrated multiple times that this manual lever works. We did this by first securing the bracket to the locking mechanism and activating the lever without any power supplied to the locking mechanism to prove that it works. We then held the lever down and tried to refastened the bracket into the locking mechanism and as expected, the bracket did not secure (again without any power supplied). It is important to note that although we tested the lock shown above in early development of this project, we have since decided on selecting the QWORK Electromagnetic Solenoid Lock because it fits our needs more in smaller size.

9.1.7 Speaker and Audio Amplifier

In *Figure 85*, the speaker and audio was first tested with a wired connection. The speakers are passive speakers, meaning they do not need a voltage applied to it in order to function. The speaker will use the audio signal and transform it into sound, however that volume is very low. This means we need to use the audio amplifier. The components were first tested in stages to ensure each part worked. In this stage, a team member supplied a 12V DC power to turn on the amplifier. They then attached its output + and - pins to the + and - of the speaker. With a wired aux cable for input, they connected it to a phone and played music. The speakers were able to output a clear sound, and the gain knob was able to turn up the volume. However at loud volumes, the speaker started making a buzzing sound. When the gain knob was tuned to 50% -75%, the music was unintelligible and we had to turn it down to hear clearly. After different tests, we learned that we have to ensure stable connections at the pinouts to prevent it. We also have to ensure a better input signal, because the aux cord was not new and affected the audio output. For the second stage, they added two wires to the amplifiers output screw terminals to connect two passive speakers. By using the multimeter, we saw that two speakers split the voltage between the two.

For the second stage shown in *Figure 86*, we tested using the speakers wirelessly. With this test, we forgoed the audio amplifier and connected the speaker inputs directly to the ESP32 data pins. This is to ensure that the ESP32 data pins will correctly transmit audio output and to also test the

sound level. In doing so, we learned that the audio amplifier is necessary, since the volume of the speaker at its maximum was like a whisper, about 35 dB. The wireless input was also similar to that of the wired aux input, meaning that wireless inputs will not have a noticeable difference in sound quality.

For the last stage, we tested by connecting the audio amplifier in between the speaker and the ESP32. Since the ESP32 is simply outputting an audio signal, the audio module in between should be able to amplify the sound to the speakers. Since this is how the audio in the cooler will be connected, this stage was very important. By desoldering the aux input from the audio amplifier, we were able to use the pins from the LM386 and connect that to the ESP32 data pins. This allows a wireless input that also amplifies the sound to the speaker. In doing so, we found that it buzzed considerably more than Stage 1, with the gain knob only at 40%-50%. This is due to the desoldering of the aux input along with a faulty gain knob. Since we desoldered the aux input, we did not have direct access to the input pins, which means the connection was not as direct as could be. We had to use the traces in the amplifier's PCB. Knowing this, in our PCB we will be able to create a more stable input, direct connections, and the least amount of wire to prevent audio losses so that the speaker works as intended.

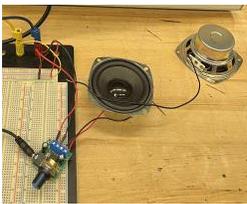


Figure 86 Stage 1: Testing speakers and Audio Amplifier with Wired Connection



Figure 87 Stage 2: Connecting speakers directly to the ESP 32

9.1.8 Rechargeable Battery

To test the rechargeable battery, we used the digital multimeter at its two terminals. This battery is rated for 12V, so a reading of 13.26V is normal and a good indication that the battery is operating at full charge. A typical 12V battery will read at 12.6V or below when it is at 50% or lower capacity. To test the battery underneath a load, we used it in conjunction with the speaker and audio amplifier. The audio amplifier has a wide input voltage, so the battery is perfectly capable of powering this system. When the digital multimeter was put between the two terminals, it still read a voltage of 13.26V. This can be seen in *Figure 87*.

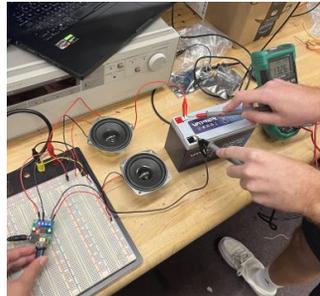


Figure 88: Battery Under Speaker Load

9.1.9 Solar Panel and Solar Charge Controller

Testing the solar panel and the solar charge controller are important to ensure that the system will be powered correctly. Since the solar panel and battery will be providing power to the entire cooler, making sure that it is performing as expected is crucial. To do this, we tested the solar panel and charger in three different stages. In *Figure 89*, we are measuring the open circuit voltage of the solar panel. Using a digital multimeter set to the specified range, we can see that it is measuring at 22.56V. This is well within the expected parameters of the 20W solar panel. The rated solar panel output voltage is 12/24V. In addition, we are able to confirm the correct voltage through the solar charge controller. Since they are the same value, both the multimeter and the solar charge controller are measuring solar panel voltage correctly.

The second stage of testing this solar panel and controller is analyzing the current flowing through. In *Figure 89*, we are measuring the current output of the solar panel. This 20W Top Solar Solar Panel is rated for an operating current of 1.11A at optimal solar conditions. We can see that with the digital multimeter it is showing an amperage of 1.03A. This was when the solar panel was angled for the best direct sunlight. The rated current for this panel is listed as 1.11A, so this is within range for the solar panel to be working correctly. Since this is rated as a 20W solar panel, our average power wattage using our recorded values can be calculated. Power is calculated as voltage multiplied by current, so $22.56V * 1.02A = 23.01W$. This is the solar panel without load, so as we connect more and more components that will draw power away from the solar panel and controller, it will be well within the rated ranges for this solar panel.



Figure 89: Solar Panel Voltage Testing

Figure 90: Solar Panel Current Testing

For our third stage, we connected the 12V battery to the solar charge controller to complete the power delivery system. We confirmed that the voltage readings for both the multimeter and the solar charge controller are accurate. In addition, we were able to confirm that the solar panel was charging the battery. Initially, the voltage reading across the battery terminals was 13.26V. When we let the solar panel charge the battery, we saw that it had risen to 13.37V, a clear indication that the battery is being charged by the solar panel. This proves that the solar panel is capable of providing supplementary power to the battery.

9.1.10 Button Testing

Buttons are possibly the simplest things to test on the list of components we need for our smart cooler device. The buttons we switched were ON/OFF rocker switches and came in a pack of 5. We tested each button switch with the same method to determine if there were any faulty components. The testing method was quite simple, we hooked up the positive and negative (ground) terminals of the switches to the appropriate terminals of a Rohde & Schwarz HMC8012 Digital Multimeter and put the multimeter in Continuity Range/Mode to determine if the buttons are adequately resisting current when flipped into the ON position. Continuity mode tests if there is a continuous flow of current through a system (which means it is working). So for a button switch such as the ones we purchased, if they are placed in the OFF position, there should be no continuity since the switch has made the system into an open circuit. However, once the switch is placed in the ON position, there will be a small resistance which signals there is continuity and therefore a proper flow of current through the system. We set our multimeter to alert us with an audio notification whenever the current met a resistance of above 200 ohms which is very small but signals that there is at least some continuity.

9.2 Software Testing

Below are examples of the software required for this project, the MIT application and the Arduino software. They have been used throughout the project to test specific parts and functionality, so below will be a few instances of how they were tested for demonstration sake.

9.2.1 MIT App Testing

The initial testing of the application is shown in *Figure 90*. The MIT App software is an easy-to-use application maker that can be used to make apps for Android phones and interact with Arduino. For testing, a team member created an application that turned a light on by pressing a button and pressing it again turns it off. They used a LED light with a 330 ohm resistor, put it to digital pin 5, and began the testing. First they needed to connect the application to their phone, and that was easily done by making the application and just sending it through email to their phone. Android phones allow this process to be seamless as for when they placed the application on their phone, it instantly began the app installation process and the UI popped up. Then once they placed the Arduino code to their ESP32, they connected Bluetooth and began sending the signal to turn the light on and off. From this process, it was successful.



Figure 91: MIT APP ESP32 Physical Testing

9.2.2 Arduino Testing

For most of the items obtained for the ESP32, they were all individually tested with the ESP32 and Arduino. The details are mentioned in prior sections, but the short version of it is that each functionality needed a library to be installed. Once it was, then as long as the pinout was correct, the item performed as it was intended. There were issues with adding some code, but it was mainly from the Arduino IDE and Arduino Lab IDE both using different programming languages: modified C and modified python respectively. Lessons were learned, and the Arduino IDE will continue to be used to flash data on the ESP32, and for it to connect with the items necessary for the cooler.

9.3 Overall Integration

Now that we have verified all our parts independently on our PCB, we then had to test if they all worked together. The first thing we did was connect more of our parts on the PCB and verify each was working correctly. We then proceeded to troubleshoot and alter any features that were not working correctly. Troubleshooting was done either with physical circuit design or debugging or the addition of more optimized code. For the physical part we had to make sure everything fit uniformly in the cooler. This was completed by first making rough drafts of what we wanted the cooler to look like then drilling and adhering the parts to make this possible. As this process went on parts such as the speaker and USB charging ports were verified that they fit correctly. After these parts were confirmed to fit holes for wires were made so that externally mounted parts such as the LED lights and Solar Panel wires could be connected inside the internal housing. These parts were all secured through various means and tested within the cooler itself to verify correct functionality. *Figure 91.*



Figure 92: Shows all major components being used together with PCB in cooler

10. Administration

10.1 Budget and Financing

Overall, the total cost of the project between Senior Design 1 and Senior Design 2 to be \$470. Split between all four of us without any sponsors will cost \$117.5 per person. Below, *Table 24* shows an itemized cost of everything required for our project.

Component	Quantity	Cost
Cooler	1	\$10
LED Strips	1	\$30
Arduino ESP-32 Dev Board	1	\$16
Keypad	1	\$7
Analog Buttons	4	\$15
Electronic Locks	2	\$30
Temperature Sensors	1	\$12
Sound Sensor	1	\$12
Battery	1	\$28
Solar Panel	1	\$28
Speaker	1	\$13
Audio Amplifier Module	1	\$9
Solar Umbrella Parts	1	\$5
USB Ports	1	\$15
Custom PCB Order	1	\$72.34
PCB Parts #1 (Digikey)	1	123.86
Waterproof Flex Sealant	1	\$11
Plastic Divider and Container for Cooler	1	\$15
PCB Parts #2 (Digikey)	1	\$17.42

Table 24: Actual Parts Prices

10.2 Initial Milestones

10.2.1 Senior Design 1

Below in *Table 25* shows the plan for Senior design 1 and how things were broken up to ensure that everything went smoothly.

Milestone	Description	Duration	Due Date
Group Formed	Find group members	<1 Week	August 25th
Project Idea	Brainstorm, Divide and Conquer 10 page	3 Weeks	September 15th
Initial Research	Find components, research	3 weeks	September 15th
30 Page Draft	Starting Research	3.5 Weeks	October 11th
50 Page Draft	Revising 30 Page draft, finalizing research	2 Weeks	October 22nd
60 Page Draft	Revising 50 page draft, finalizing research	2 Weeks	November 3rd
Parts Tested	All parts bought and Tested	2 Weeks	November 17th
90 Page Draft	Revising 60 page draft, get good grasp on testing materials	2 Weeks	November 17th
Initial PCB Design Finished	Initial Design Finished for testing for product	3 Weeks	December 5th
Final Document	Revising 120 page draft, finalizing document	4.5 Weeks	December 5th

Table 25: Senior Design 1 Plan

10.2.2 Senior Design 2

Below in *Table 256* shows the plan for Senior design 2 and how things were broken up to ensure that everything went smoothly. Note: Multiple milestones overlap depending on each team member's work distribution.

Milestone	Description	Duration	Due Date
PCB Design	Finish PCB Schematic and Layout	6 Weeks	2/9
App Design	Code each feature and test via breadboard	8 Weeks	2/23
Build PCB	Create, solder, and verify a working PCB	2 Weeks	3/9
Initial Test	First test for components utilizing PCB and app	3 Weeks	3/30
App Integration	Integrating all working codes into one concise app	1 Week	3/30
Build Cooler	Physically mount all components and cable manage on the cooler	1 Week	4/8
Finalize Design	Completed working model and anything missing from before	5 Days	4/12
Debug	Troubleshoot any issues occurring with the cooler's final design and app	1 Week	4/15
Final Report	Complete report	1 Week	4/16
Final Presentation	Complete presentation	1 Week	4/16

Table 26: Senior Design 2 Plan

11. Conclusion

In conclusion, after months of research, development, troubleshooting, and construction we have reached our desired goals for building our SunChill. Due to all of the hard work and preparation we have completed this entire project, we believe that we are in a solid position to present our Senior Design 2 project. This is not to discount all roadblocks we encountered, however, with time and effort we have overcome multiple MCU failures due to unknown causes and completed our task.

Our initial budget we planned for was grossly more than the final bill of materials came out to be. We had initially all agreed that the total price for all components, materials, and miscellaneous devices would summarize to roughly \$750. Although this is a hefty price, we were willing to evenly split it amongst the team, in the hopes that our price approximations were too harsh and that we would find sponsors to lighten the financial load. Unfortunately, as of the end of Senior Design 2, we have yet to secure any financial sponsorships. Our team continued to strive on self funding the project. Having said all that, we are fortunate to report that our actual cost of the bill of materials was a bit over half the initial estimated cost, coming out to be around \$470. This significant price reduction gives us a bit of joy as even though we had bought multiple extra parts and backups, we planned correctly for what we would need .

Having secured a solid foundation for Senior Design 2 we still had to lock-in and focus to achieve our goals and stay on time. This included finding the capabilities of our PCB and what it could and could not do, designing and building a PCB was one side of the coin Senior Design 2. There were multiple instances where we had to think on the fly and come up with new ideas to reach our end goals.

The majority of our goals have been met this semester during Senior Design 2. We have finalized a PCB and cooler design and have been working toward that goal ever since its approval early on in the semester. When trying to implement some of our goals we are still cautious about implementing into the overall final build of our smart cooler. This was due to the high power draw of these components and didn't want to risk damaging other types of functionality. However, we were still able to design, and develop successfully while implementing all our parts to make our project whole. Even so, this is still on our radar to improve the longevity of our project's whole functionality. This is because our sensitive components take time and effort to implement properly, and could burn up in a flash. Doing this improperly or unexplained component failure causes us the loss of money, time, and or other resources if we are not careful enough.

Having completed an entire semester of Senior Design 2, we have come to realize the purpose of this project and university capstone. We are four individuals that have been tasked with coming together to fulfill a commonly desired goal; building a smart cooler. This improves us not only as future engineers, but also as people with the need to work in groups. Working in a group required basic communication, trust, equal effort, and reliability. It is a constant push and pull dynamic that works to balance out the entire road map of a project. This was a necessary skill for us to acquire for when we enter the engineering business after graduation.

Additionally to our development as group members, we have furthered our development as individuals throughout this entire semester of working on Senior Design 2. Before beginning this semester, we all arrived with our own individualized skill sets and knowledge. We all had differing levels of knowledge or expertise in fields such as embedded systems, robotics, programming, and electronic design. There were many topics with which none of us have had experience in that we were determined to research and familiarize ourselves with. Artificial intelligence, designing, and implementation are all examples of unfamiliar concepts we had to understand. After completing Senior Design 2, we have greatly increased our knowledge on all topics for the cooler as we have all been forced in one way or another to dip into every single topic discussed in making our final project. If we had not done so, this project would have never been created and been a small shadow of what it currently is.. Only through teamwork and dedication did we reach this point of completion and success.

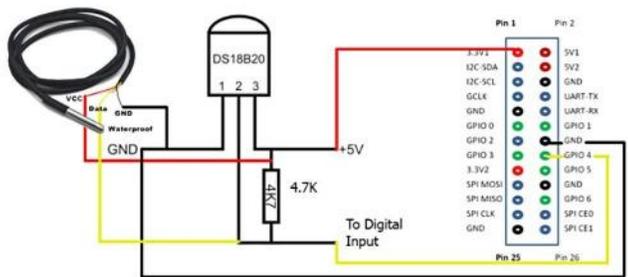
Finally, as we look toward the end of Senior Design 2 and the beginning of our future careers, we bring with us many bits of useful information we have learned as a group throughout this semester that will help us reach for success in the real world. We are determined to retain our knowledge and dive deeper into topics we have enjoyed when making this project. We have held each other to a high standard of accountability, responsibility, and capability throughout the remainder of this semester, through the spring break, and throughout the entirety of the spring semester until our final demo for this project.

Appendix

Appendix A: Code and Data Sheets

Connecting Banana Pi

Connect the DS18B20 with the Banana Pi as following:



Temperature Sensor Code:

```
#include <OneWire.h>
#include <DallasTemperature.h>
// Data wire plugged into pin 2 Arduino
#define ONE_WIRE_BUS 2
// Setup a oneWire for communication with tempensor
OneWire oneWire(ONE_WIRE_BUS);
// oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);
void setup() {
  // Start serial communication baud rate 9600
  Serial.begin(9600);
  // Initialize the DS18B20 sensor
  sensors.begin();
}
void loop() {
  // Request temperature data from all DS18B20 sensors on the bus
  sensors.requestTemperatures();
  // Get the temperature in Celsius
  float temperatureC = sensors.getTempPCByIndex(0);
  // Check if the temperature reading is valid
  if (temperatureC != DEVICE_DISCONNECTED_C) {
    // Print the temperature to the Serial Monitor
    Serial.print("Temperature: ");
    Serial.println(temperatureC);
    Serial.println(" °C");
  } else {
    Serial.println("Error reading temperature!");
  }
  // delay before taking the next reading
  delay(1000);
}
```

Commented [14]: adam do we need dis pic

Commented [15]: MOVE TO APPENDIX OR GET RID OF . No pinout pictures

Commented [16]: btw didnt mean to be capitalized

Commented [17]: I will remove after i am done

Commented [18]: gotcha. I have this saved in the appendix so remove when ya want

Commented [19]: Yo code cant be a picture so this would need to be moved to the appendix or deleted

ESP32 DataPins

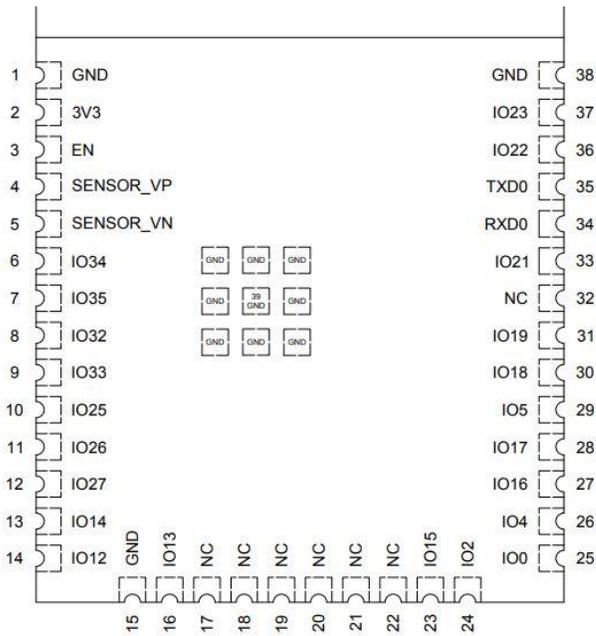


Figure 3: Pin Layout (Top View)

This photo illustrates the circuit used for testing the probe

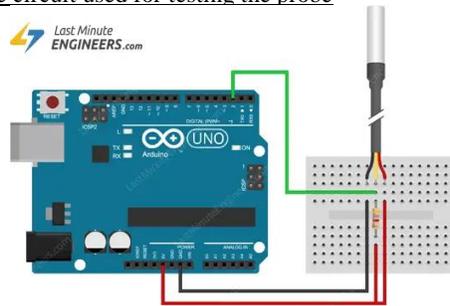
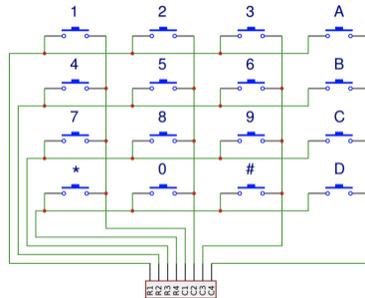


Figure 90: Button Schematic



Code Used for Testing the Parts:

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// Released under the GPLv3 license to match the rest of the
// Adafruit NeoPixel library
#include <Keypad.h>

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif
#define ROW_NUM 4 // four rows
#define COLUMN_NUM 4 // four columns
#include "BluetoothA2DPSink.h"
#include <OneWire.h>
#include <DallasTemperature.h>
// Data wire plugged into pin 2 Arduino
#define ONE_WIRE_BUS 4
// Setup a oneWire to communicate with tempsensor
OneWire oneWire(ONE_WIRE_BUS);
// oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);
BluetoothA2DPSink a2dp_sink;

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN 22 // On Trinket or Gemma, suggest changing this to 1

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 50 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
```

```

// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
int p = 0;
byte pin_rows[ROW_NUM] = {19, 18, 5, 17}; // GPIO19, GPIO18, GPIO5, GPIO17 connect
to the row pins
byte pin_column[COLUMN_NUM] = {16, 4, 23, 15}; // GPIO16, GPIO4, GPIO0, GPIO2
connect to the column pins
Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
void setup() {
  static const i2s_config_t i2s_config = {
    .mode = (i2s_mode_t) (I2S_MODE_MASTER | I2S_MODE_TX |
I2S_MODE_DAC_BUILT_IN),
    .sample_rate = 44100, // corrected by info from Bluetooth
    .bits_per_sample = (i2s_bits_per_sample_t) 16, /* the DAC module will only take the 8bits
from MSB */
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = (i2s_comm_format_t) I2S_COMM_FORMAT_STAND_MSB,
    .intr_alloc_flags = 0, // default interrupt priority
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = false
  };

  a2dp_sink.set_i2s_config(i2s_config);
  a2dp_sink.start("MyMusic");
p = 0;
Serial.begin(9600);
sensors.begin();
// These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
// Any other board, you can remove this part (but no harm leaving it):
#ifdef __AVR_ATtiny85__ && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
// END of Trinket-specific code.

pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

```

```

void loop() {
  char key = keypad.getKey();
  //Serial.println(key);
  if(p==0){
    Serial.begin(9600);
    char key = keypad.getKey();
    Serial.println(key);
    if (key=='1')
    {
      p=1;
    }
    if (key=='2')
    {
      p=2;
    }
  }
  else if (p==2)
  {
    for(int i=0; i<1000; i++)
    {
sensors.requestTemperatures();
// Get the temperature in Celsius
float temperatureC = sensors.getTempPCByIndex(0);
// Check if the temperature reading is valid
if (temperatureC != DEVICE_DISCONNECTED_C) {
  // Print the temperature to the Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" Â°C");
} else {
  Serial.println("Error reading temperature!");
}
// delay before taking the next reading
delay(1000);
}
}
else if (p==1)
{

pixels.clear(); // Set all pixel colors to 'off'

// The first NeoPixel in a strand is #0, second is 1, all the way up
// to the count of pixels minus one.
for(int i=0; i<NUMPIXELS; i++) { // For each pixel...

```

```

char key = keypad.getKey();

// pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
// Here we're using a moderately bright green color:
pixels.setPixelColor(i, pixels.Color(0, 150, 0));

pixels.show(); // Send the updated pixel colors to the hardware.

delay(DELAYVAL); // Pause before next pass through loop
}
p=0;
}
}
}

```

Lightshow Testing Code:

```

// @file ColorPalette.ino
// @brief Demonstrates how to use @ref ColorPalettes
// @example ColorPalette.ino

#include <FastLED.h>

#define LED_PIN 26
#define NUM_LEDS 50
#define BRIGHTNESS 64
#define LED_TYPE WS2811
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];

#define UPDATES_PER_SECOND 100

// This example shows several ways to set up and use 'palettes' of colors
// with FastLED.
//
// These compact palettes provide an easy way to re-colorize your
// animation on the fly, quickly, easily, and with low overhead.
//
// USING palettes is MUCH simpler in practice than in theory, so first just
// run this sketch, and watch the pretty lights as you then read through
// the code. Although this sketch has eight (or more) different color schemes,
// the entire sketch compiles down to about 6.5K on AVR.
//
// FastLED provides a few pre-configured color palettes, and makes it
// extremely easy to make up your own color schemes with palettes.
//
// Some notes on the more abstract 'theory and practice' of
// FastLED compact palettes are at the bottom of this file.

```

```

CRGBPalette16 currentPalette;
TBlendType currentBlending;

extern CRGBPalette16 myRedWhiteBluePalette;
extern const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM;

void setup() {
  delay( 3000 ); // power-up safety delay
  FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection( TypicalLEDStrip );
  FastLED.setBrightness( BRIGHTNESS );

  currentPalette = RainbowColors_p;
  currentBlending = LINEARBLEND;
}

void loop()
{
  ChangePalettePeriodically();

  static uint8_t startIndex = 0;
  startIndex = startIndex + 1; /* motion speed */

  FillLEDsFromPaletteColors( startIndex);

  FastLED.show();
  FastLED.delay(1000 / UPDATES_PER_SECOND);
}

void FillLEDsFromPaletteColors( uint8_t colorIndex)
{
  uint8_t brightness = 255;

  for( int i = 0; i < NUM_LEDS; ++i) {
    leds[i] = ColorFromPalette( currentPalette, colorIndex, brightness, currentBlending);
    colorIndex += 3;
  }
}

// There are several different palettes of colors demonstrated here.
//

```

```

// FastLED provides several 'preset' palettes: RainbowColors_p, RainbowStripeColors_p,
// OceanColors_p, CloudColors_p, LavaColors_p, ForestColors_p, and PartyColors_p.
//
// Additionally, you can manually define your own color palettes, or you can write
// code that creates color palettes on the fly. All are shown here.

void ChangePalettePeriodically()
{
  uint8_t secondHand = (millis() / 1000) % 60;
  static uint8_t lastSecond = 99;

  if( lastSecond != secondHand) {
    lastSecond = secondHand;
    if( secondHand == 0) { currentPalette = RainbowColors_p;      currentBlending =
LINEARBLEND; }
    if( secondHand == 10) { currentPalette = RainbowStripeColors_p; currentBlending =
NOBLEND; }
    if( secondHand == 15) { currentPalette = RainbowStripeColors_p; currentBlending = I
LINEARBLEND; }
    if( secondHand == 20) { SetupPurpleAndGreenPalette();      currentBlending =
LINEARBLEND; }
    if( secondHand == 25) { SetupTotallyRandomPalette();      currentBlending =
LINEARBLEND; }
    if( secondHand == 30) { SetupBlackAndWhiteStripedPalette(); currentBlending =
NOBLEND; }
    if( secondHand == 35) { SetupBlackAndWhiteStripedPalette(); currentBlending =
LINEARBLEND; }
    if( secondHand == 40) { currentPalette = CloudColors_p;      currentBlending =
LINEARBLEND; }
    if( secondHand == 45) { currentPalette = PartyColors_p;      currentBlending =
LINEARBLEND; }
    if( secondHand == 50) { currentPalette = myRedWhiteBluePalette_p; currentBlending =
NOBLEND; }
    if( secondHand == 55) { currentPalette = myRedWhiteBluePalette_p; currentBlending =
LINEARBLEND; }
  }
}

// This function fills the palette with totally random colors.
void SetupTotallyRandomPalette()
{
  for( int i = 0; i < 16; ++i) {
    currentPalette[i] = CHSV( random8(), 255, random8());
  }
}

```

```

// This function sets up a palette of black and white stripes,
// using code. Since the palette is effectively an array of
// sixteen CRGB colors, the various fill_* functions can be used
// to set them up.
void SetupBlackAndWhiteStripedPalette()
{
  // 'black out' all 16 palette entries...
  fill_solid( currentPalette, 16, CRGB::Black);
  // and set every fourth one to white.
  currentPalette[0] = CRGB::White;
  currentPalette[4] = CRGB::White;
  currentPalette[8] = CRGB::White;
  currentPalette[12] = CRGB::White;
}

// This function sets up a palette of purple and green stripes.
void SetupPurpleAndGreenPalette()
{
  CRGB purple = CHSV( HUE_PURPLE, 255, 255);
  CRGB green = CHSV( HUE_GREEN, 255, 255);
  CRGB black = CRGB::Black;

  currentPalette = CRGBPalette16(
    green, green, black, black,
    purple, purple, black, black,
    green, green, black, black,
    purple, purple, black, black );
}

// This example shows how to set up a static color palette
// which is stored in PROGMEM (flash), which is almost always more
// plentiful than RAM. A static PROGMEM palette like this table
// takes up 64 bytes of flash.
const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM =
{
  CRGB::Red,
  CRGB::Gray, // 'white' is too bright compared to red and blue
  CRGB::Blue,
  CRGB::Black,

  CRGB::Red,
  CRGB::Gray,
  CRGB::Blue,
  CRGB::Black,
}

```

```
CRGB::Red,  
CRGB::Red,  
CRGB::Gray,  
CRGB::Gray,  
CRGB::Blue,  
CRGB::Blue,  
CRGB::Black,  
CRGB::Black  
};
```

```
// Additional notes on FastLED compact palettes:  
//  
// Normally, in computer graphics, the palette (or "color lookup table")  
// has 256 entries, each containing a specific 24-bit RGB color. You can then  
// index into the color palette using a simple 8-bit (one byte) value.  
// A 256-entry color palette takes up 768 bytes of RAM, which on Arduino  
// is quite possibly "too many" bytes.  
//  
// FastLED does offer traditional 256-element palettes, for setups that  
// can afford the 768-byte cost in RAM.  
//  
// However, FastLED also offers a compact alternative. FastLED offers  
// palettes that store 16 distinct entries, but can be accessed AS IF  
// they actually have 256 entries; this is accomplished by interpolating  
// between the 16 explicit entries to create fifteen intermediate palette  
// entries between each pair.  
//  
// So for example, if you set the first two explicit entries of a compact  
// palette to Green (0,255,0) and Blue (0,0,255), and then retrieved  
// the first sixteen entries from the virtual palette (of 256), you'd get  
// Green, followed by a smooth gradient from green-to-blue, and then Blue.
```

Appendix B: Figure Index

- Figure 1: House of Quality, **14**
- Figure 2: Basic Hardware Block Diagram, **15**
- Figure 3: Software Diagram, **16**
- Figure 4: HiLetsgo Sensor, **28**
- Figure 5: Alitove 50pc lights, **30**
- Figure 6: DayBetter Led strips, **30**
- Figure 7: Water Changing Color LED Lights, **31**
- Figure 8: Speaker Part Diagram, **32**
- Figure 9: Degraw DIY Speaker Kit, **33**
- Figure 10: Kibbeh Speakers, **33**
- Figure 11: HZPOWEN Speakers, **33**
- Figure 12: LM386 Audio Amplifier, **34**
- Figure 13: Figure Miniature Push Button, **36**
- Figure 14: Pushbutton Switch SPST-NO, **37**
- Figure 15: Momentary Push Button Switch, **37**
- Figure 16: DaierTek ON Off Round Rocker, **38**
- Figure 17: Tegg 1PC 3x4 Keypad, **41**
- Figure 18: 12 Key Membrane Switch Keypad, **41**
- Figure 19: Songhe 4 x 4 Matrix Array Membrane Switch Keypad, **42**
- Figure 20: Digilent PmodKYPD - 16-button Keypad, **42**
- Figure 21: Rotating Door Lock DC Motor, **45**
- Figure 22: QWORK Electromagnetic Solenoid Lock (4Pc), **46**
- Figure 23: Metal Fail Secure Mode Multifunctional Mini Electric Cabinet Lock, **46**
- Figure 24: TOPLEE Electronic Cabinet Lock, **47**
- Figure 25: PENGLIN 2Pack USB 2.0 Panel, **48**
- Figure 26: ICESPRING USB, **49**
- Figure 27: Duttek, **49**
- Figure 28: MELIFE 10pcs, **50**
- Figure 29: Mighty Max Battery ml7-12 - 12 Volt 7.2ah SLA Battery, **52**
- Figure 30: UPLUS LP12-8 12 Volt 8Ah Rechargeable AGM Battery, **53**
- Figure 31: Vatrer Power 12V 7AH LiFePO4 Battery Deep Cycle Lithium Battery, **53**
- Figure 32: 20 watt SOLPERK Panel, **55**
- Figure 33: SOLPERK Panel 30 Watt, **55**
- Figure 34: Conversion Explanation Example for Solar Panels, **56**
- Figure 35: Top Solar 12 Volt Solar Panel, **56**
- Figure 36: Panel Layout, **57**
- Figure 37: Coleman 316 1, **58**
- Figure 38: Coleman 316 2, **58**
- Figure 39: Igloo Latitude, **59**
- Figure 40: Arduino Uno Rev3, **62**
- Figure 41: Raspberry Pi 4b, **62**
- Figure 42: MSP 430, **62**
- Figure 43: ESP32, **63**
- Figure 44: HM-10 Bluetooth Module, **63**
- Figure 45: Light Frequency to Modulation Standards, **68**

Figure 46: Sensor Standards, **68**
Figure 47: Speaker Frequency and Specifications Standards, **69**
Figure 48: ChatPDF Summarization Example, **77**
Figure 49: Chat GPT Coding Example, **78**
Figure 50: Chat GPT Grammar Correcting Example, **79**
Figure 51: Chat GPT Error Coding Example, **80**
Figure 52: Chat GPT App Example, **81**
Figure 53: Overall Schematic, **84**
Figure 54: Hardware General Block Diagram, **85**
Figure 55: Power Delivery System Block Diagram, **87**
Figure 56: Voltage Regulator Schematic, **88**
Figure 57: Solar Umbrella Wiring Diagram, **89**
Figure 58: KeyPad Hardware Block Diagram, **90**
Figure 59: Hardware Physical Diagram, **91**
Figure 60: LM386 Low Voltage Audio Power Amplifier Chip Integration, **93**
Figure 61: Temperature Sensor Schematic, **94**
Figure 62: LED Lights connect to 5V Output Regulator, **95**
Figure 63: Sensor Block Diagram, **96**
Figure 64: Keypad and Lock Basic Block Diagram, **97**
Figure 65: Bluetooth App Input Block Diagram, **98**
Figure 66: AI Voice Flowchart, **102**
Figure 67: Buttons General Picture, **103**
Figure 68: Buttons General Picture 2, **103**
Figure 69: Buttons General Picture 3, **103**
Figure 70: Buttons General Picture 4, **103**
Figure 71: Keypad Detailed Code Flowchart, **105**
Figure 72: Battery/AI Code Flowchart, **106**
Figure 73: ESP32--WROOM--32E Schematic, **108**
Figure 74: 2D Layout of the PCB Design, **109**
Figure 75: Completed 3D render of PCB Design, **111**
Figure 76: Igloo Legend 52 Quart, **110**
Figure 77: Side View, **111**
Figure 78: Cooler Internals, **112**
Figure 79: Front view of cooler, **113**
Figure 80: LED Lights Testing, **114**
Figure 81: Temperature Sensor Testing, **115**
Figure 82: ESP32 Testing of Bluetooth Light, **116**
Figure 83: Button Testing, **117**
Figure 84: Lock Testing, **118**
Figure 83: Power Confirmed Via Battery Power and Multimeter DC Voltage Testing, **115**
Figure 84: Testing Across USB Traces, **115**
Figure 85 Stage 1: Testing speakers and Audio Amplifier with Wired Connection, **119**
Figure 86 Stage 2: Connecting speakers Directly to the ESP32, **119**
Figure 87: Battery Under Speaker Load, **120**
Figure 88: Solar Panel Voltage Testing, **121**
Figure 89: Solar Panel Current Testing, **121**

Figure 90: MIT APP ESP32 Physical Testing, **122**

Figure 91: Shows all major components being used together with ESP32, **123**

Appendix C: Table Index

Table 1: Project Specifications,	13
Table 2: Display Options,	18
Table 3: Cooling Options,	20
Table 4: Light Options,	23
Table 5: AI Voice Options,	24
Table 6: Temperature Sensor Options,	25
Table 7: LED Light Options,	29
Table 8: Speaker Option,	31
Table 9: Switch Options,	34
Table 10: Button Options,	39
Table 11: Lock Options,	43
Table 12: USB Port Options,	48
Table 13: Female USB Port Options,	50
Table 14: Rechargeable Battery Options,	51
Table 15: Solar Panel Options,	54
Table 16: Cooler Options,	57
Table 17: Software Options,	59
Table 18: Microcontroller Options,	61
Table 19: Part Selection,	65
Table 20: Bluetooth Power Standards,	66
Table 21: USB Standards,	67
Table 22: Subsystem Requirements,	70
Table 23: ESP32 Pinout and Connections,	83
Table 24: Actual Parts Prices,	124
Table 25: Senior Design 1 Plan,	125
Table 26: Senior Design 2 Plan,	126

Appendix D: References

1. VEVOR 1.27 cu. ft. Outdoor Refrigerator Dual Zone Car Fridge Freezer Electric Compressor Cooler w/App Control & Wheels in Gray FTBZPE36L110VELLHV1. (n.d.). The Home Depot. Retrieved December 5, 2023, from <https://www.homedepot.com/p/VEVOR-1-27-cu-ft-Outdoor-Refrigerator-Dual-Zone-Car-Fridge-Freezer-Electric-Compressor-Cooler-w-App-Control-Wheels-in-Gray-FTBZPE36L110VELLHV1/321820802>
2. July was globe's hottest month on record and 11th-warmest July on record in US. (n.d.). USA TODAY. <https://www.usatoday.com/story/news/nation/2023/08/08/july-earths-hottest-month-on-record/70547096007/>
3. Electronic Enclosure Cooling Thermoelectric vs. Compressor-Based Air Conditioners | The World Leader in Thermal Management Solutions. (n.d.). Lairdthermal.com. Retrieved December 5, 2023, from <https://lairdthermal.com/thermal-technical-library/white-papers/electronic-enclosure-cooling-thermoelectric-vs-compressor-based-air#:~:text=A%20compressor%20system%27s%20components%20will>
4. "IEEE Standard for Measuring Electroacoustic Performance of Communication Devices," in IEEE Std 269-2019 (Revision of IEEE Std 269-2010) , vol., no., pp.1-203, 13 Feb. 2020, doi: 10.1109/IEEESTD.2020.8998539.
5. IEEE Guide for Array and Battery Sizing in Stand-Alone Photovoltaic (PV) Systems. (n.d.). <https://doi.org/10.1109/ieeestd.2008.4518937>
6. IEEE Standard for Sensor Performance Parameter Definitions. (2018). IEEE Std 2700-2017 (Revision of IEEE Std 2700-2014), 1–64. <https://doi.org/10.1109/IEEESTD.2018.8277147>
7. IEEE Recommended Practices for Modulating Current in High-Brightness LEDs for Mitigating Health Risks to Viewers. (2015). IEEE Std 1789-2015, 1–80. <https://doi.org/10.1109/IEEESTD.2015.7118618>
8. USB Port and power delivery: An overview of USB port interoperability | IEEE Conference Publication | IEEE Xplore. (n.d.). [ieeexplore.ieee.org](https://ieeexplore.ieee.org/document/7138710). Retrieved December 5, 2023, from <https://ieeexplore.ieee.org/document/7138710>
9. Smoot, J. (2022, September 27). Decoding the USB Standards from 1.0 to 4.0. DigiKey. <https://www.digikey.com/en/articles/decoding-the-USB-standards-from-1-to-4>
10. IEEE Standard for System, Software, and Hardware Verification and Validation. (2017). IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/Cor1-2017), 1–260. <https://doi.org/10.1109/IEEESTD.2017.8055462>
11. An overview of Bluetooth Wireless Technology/sup TM/ and some competing LAN standards | IEEE Conference Publication | IEEE Xplore. (n.d.). [ieeexplore.ieee.org](https://ieeexplore.ieee.org/abstract/document/1029081). Retrieved December 5, 2023, from <https://ieeexplore.ieee.org/abstract/document/1029081>
12. ChatPDF - Chat with any PDF! (n.d.). [www.chatpdf.com](https://www.chatpdf.com/c/HooCGjy1xF3vOaJcDio7). Retrieved December 5, 2023, from <https://www.chatpdf.com/c/HooCGjy1xF3vOaJcDio7>
13. Martínez, I. (2023, July 30). privateGPT. GitHub. <https://github.com/imartinez/privateGPT>
14. GitHub. (2023). GitHub Copilot · Your AI pair programmer. GitHub. <https://github.com/features/copilot>

15. power - Powering the Pi 4 - safe voltage levels and current requirements. (n.d.). Raspberry Pi Stack Exchange. <https://raspberrypi.stackexchange.com/questions/99961/powering-the-pi-4-safe-voltage-levels-and-current-requirements>
16. Massachusetts Institute of Technology. (2019). MITApp Inventor | Explore MITApp Inventor. Mit.edu. <https://appinventor.mit.edu/>
17. React Native. (2022). React Native · A framework for building native apps using React. Reactnative.dev. <https://reactnative.dev/>
18. (2023). Tequipment.net. https://www.tequipment.net/Digilent/PmodKYPD/General-Accessories/?Source=googleshopping&utm_source=google&utm_medium=organic&utm_campaign=surfaces-across-google#description
19. How To Make DIY Music Reactive RGB LED Strip (WS2812B). (n.d.). Wwww.youtube.com. Retrieved September 19, 2022, from <https://www.youtube.com/watch?v=5WP2Tjt9o2U>
20. DIY Automatic Musical Christmas Light Show (Bluetooth + MSGEQ7 + Arduino). (n.d.). Wwww.youtube.com. Retrieved December 5, 2023, from <https://www.youtube.com/watch?v=pgVAOd0k9A4>
21. How I made a simple christmas light show with an Arduino. (n.d.). Wwww.youtube.com. Retrieved December 5, 2023, from <https://www.youtube.com/watch?v=MafNmsmBW8s>
22. H, Y. L. (2020, October 13). Music to LED strip tutorial (using Fourier Transform). Medium. <https://medium.com/@yolandaluqueh/music-to-led-strip-tutorial-using-fourier-transform-3d203a48fe14>
23. Schatzmann, P. (2023, June 26). A Simple Arduino Bluetooth Music Receiver and Sender for the ESP32. GitHub. <https://github.com/pschatzmann/ESP32-A2DP>