HotBox

Senior Design Documentation

Spring 2021

Group 14

August 3rd, 2021

Chaitanya Vemuri - Computer Engineering
Haafiz Shafau - Computer Engineering
Austin Tillotson - Computer Engineering
Ahmed Kazzoun- Computer Engineering

Table of Contents

List of Figures

List of Tables

1. Executive Summary

For our senior design project, we developed a smart electric food heater named the HotBox. In 2020, there were over 60% of U.S. consumers who ordered either take out or delivery food at least once a week, and due to COVID-19 online ordering was the only primary source of income for a lot of restaurants as well. Takeout and deliveries have been on the rise ever since the arrival of the pandemic. Apps and services like DoorDash and UberEATS are preferred and more sought out rather than going in store and ordering due to the risk of infection. Restaurants and other food places have begun their own online ordering system and allow consumers to order from their devices and pick up or have it delivered to them. With contactless ordering being in demand, the traffic of a restaurant will be high (independent of the quality of the restaurant) for picking up the orders in the store. The more orders that restaurant receives, the more traffic that will occur in the restaurant. And a lot of restaurants limit a certain number of people in the store to lower the risk of infection as well. Our device's goal is not only to keep the food warm, but also to provide restaurants a way to limit consumers inside by notifying consumers when their food is ready and in the box warming.

The general idea is to have a box that consists of the following functions:

- Ability to warm a reasonable size of food in different materials that contain the food (i.e., plastic bag, cardboard) and retain the heat as well.
- Ability to display online orders outside the box to identify each order.
- Ability to lock and secure orders inside the box for security.
- Ability to link to restaurant computers for usage of the box.
- Ability to scan in codes to identify orders even further.

2. Project Description

This section of the document contains a description of our project. Included are the following subsections: motivation, goals and objectives, functions, requirement specifications, house of quality table, and block diagram. The motivation explains why we decided to develop this project. The goals and objectives explain what we hoped to achieve, show, and learn from doing this project. The functions section explains what the functionality of the project is. The requirement specifications are what our project achieved in order for us to consider it having accomplished the goal it was designed to do. The house of quality table shows the correlation between our project's marketing and engineering requirements. The block diagram shows an overview of what our hardware and software do on a base level for the project.

2.1 Motivation

For this project we wanted to focus on the main problem with takeout food places which is receiving your food cold. As a customer that pays for the food, we should at least be entitled to warm food. Especially with the times we are in now, contactless food pick up is almost sought after in a lot of stores and restaurants. And it's not always a guarantee that you'll receive your food on time and it'll be hot and ready to consume. However, with the HotBox, a smart electric food heater and warmer that sends out notifications to the customer picking up the order when the food is in the box and keeping warm ready to be picked up. With this project, not only will the food be warm and ready when you pick it up, it hopes to reduce a majority of the traffic and congestion in the restaurant/store itself by letting the customer know when it is ready to be picked up. Main goal of our project is to heat and warm the food to an appropriate temperature for the food to taste good. So the box will sense when food enters the box, and will be able to warm most supported materials and hold temperature for a reasonable amount of time. It also sends a notification out to the customer's devices app when the food is present in the box and heating.

2.2 Objective

The ultimate goal of Senior Design is to create and implement a project idea into a functioning physical prototype that can be demoed and presented. The Senior Design lectures help guide us on our path to this destination and the bootcamp gave us a strong starting point, having us form a foundation to build off of.

There are many smaller goals and objectives that will lead us to the end goal. Senior Design is an opportunity to display all that has been learned over each of our college careers; to show that we have acquired all the necessary skills and overall knowledge needed to become successful engineers in the workplace. As such, one goal for this semester is to display these abilities and to strengthen them. Senior Design will present many challenges that we have never encountered before. And with every challenge is an opportunity to grow. Being that we are taking Senior Design II over the summer semester, and thus having less time to build the physical project, it will be important to have well thought out and thorough documentation to make this process as painless as possible. Success is paramount for these two semesters and that cannot be achieved alone. While individual contribution will be of the utmost importance, working as a team will be the only way to succeed. Our objective must consist of strong teamwork and communication or every goal along the way will be all the more difficult to achieve.

For the documentation, we strive as a group to not procrastinate and be diligent in making new progress each week. Each time we meet, we will set specific goals that must be completed prior to the next meeting. Meetings must be effective and efficient, a time to discuss what must be done, what challenges will come, and how to deal with them. A plan must always be in place with a shared goal attached. Problems that arise

should be discussed and dealt with in a reasonable manner, to ensure a positive working environment is always present. These are our group objectives.

As for the project itself, we aim to create something that is not only useful but something that would improve our resumes. We all hope to find a career and make a living with the last four or so years of hard work. This project is the final step of this journey and the potentially the last thing we will have to place on our resumes from college.

2.2.1 List of Objectives and Goals

- Display engineering skills learned
- Overcoming challenges presented
- Write thorough documentation
- Teamwork
- Communication
- Avoid procrastination
- Have Effective meetings
- Always have a plan
- Discuss problems encountered
- Keep a positive group environment
- Build useful project
- Build a project that strengthens resume

2.3 Function

Keeping food warm: the main functionality of the HotBox would be to keep takeout food warm whilst waiting for a delivery driver to come pick it up. This would involve a hot plate for the restaurant to place the food on top of and a circuit to notify the customer while the food is being heated. The box will also be insulated and sealed to keep temperatures from varying. The box will be reasonably sized to fit most reasonable meals.

Web/App Notifications: Another functionality of the device would be to notify the driver when the food is being heated. This would ensure the food will still be warm when it gets to the customer, assuming a normal travel distance, and notifies the restaurant that the HotBox is empty and can be used for another order. The web app will have a GUI showing which boxes are empty and send the order number to an empty box for the employee to put the food in. This would help the driver in finding which HotBox holds the food that they are delivering.

Weight/Temperature: The circuit should also be able to detect the weight of the food being placed and the temperature of the plate by itself so it is limited to a certain degree to avoid burning the food or letting it get cold.

2.4 Requirements Specifications

This section contains the hardware and software specifications for the project.

2.4.1 Hardware Requirements Specifications

The following table displays the hardware specifications our project must abide by.

Table 1. Hardware Requirement Specifications

#	Requirements	Specifications
1.1	Accommodatable size	Box size should be large enough to accommodate different fast foods. (Minimum size : 10" x 10" x 10”)
1.2	Ability to set temperature and vary temperature as needed, as well as retain it for short to long periods o f time.	Heat foods kept in different materials in a temperature controlled unit. (“Within a range of 110°F and 140°F”)
1.3	Support different materials in heated temperatures	The box should support different materials such as cardboard ,particulate, and foil inside the box while idle and heating. It should be able to withstand the heating temperatures of the box at different levels.
1.4	Box Security	The box should be secure and locked if food is present or if nothing is present in the box to regulate heating temperatures. The box can only be unlocked if it is unlocked remotely through the restaurant computer or if the correct order was scanned in through the Barcode scanner.
1.5	Order Validation	The box should be equipped with a scanner for barcodes for customers to identify and unlock the box for their respective order.
1.6	Power System	Able to power multiple different voltages by a single 120V AC wall line. (“MCU - 5V DC, Lock - 12V DC, Heating Pad 24V DC.”)

2.4.2 Software Requirements Specifications

The following table displays the software requirements our project must abide by.

Table 2. Software Requirement Specifications

#	Requirements	Specifications
2.1	The microcontroller will also have to be able to connect to a web app in order to send notifications to the customer.	The web app must be able to send notifications to the client so they know when their food is being placed on the plate to be kept warm and when the delivery driver picks up their food. Using Android Push notifications or SMS to send messages to the customer.
2.2	The LCD on the box should flash the order number in order to signal to the driver to which box to pick up.	The order number would be pulled from the restaurant and flash. The software team would have to create the connection from the order and the box.
2.3	There should be a connection between a web app and the boxes themselves in order to tell which one is empty or ready to be picked up.	A web app would have to be developed in order to connect to multiple boxes to send the order number to the box and reply saying that it is empty.
2.4	Control temperatures in box	The restaurant computer will have the ability to vary the temperatures of the box.
2.5	Barcode Scanner	After the consumer has scanned the 2D barcode, the box will direct the consumer to the box where their food is located. It will also prompt the user if the order is incorrect or not present in the box.
2.6	Box Database	The box will have a database for orders that were placed to keep track of orders that enter and leave the box.

2.5 House of Quality

The following is a figure of our House of Quality. This figure shows the marketing and engineering requirements for our project and what correlation each has on each other in respect to our project, from strong positive correlations to strong negative or no correlation.

			Engineering Requirements					
			Dimensions	Production Cost	Power Consumption	Web app Communication	Temperature Regulation	Security
			(-)	(-)	(-)	(+)	(+)	(+)
Marketing Requirements	Size	(-)	↑↑	↑↑	↑		↑↑	
	Ease of Use	(+)	↓			↑↑	↑	↑
	Reliable	(+)			↑	↑↑	↑↑	↑↑
	Easy to Maintain	(+)	↑			↑		↑
	Quality	(+)		↑↑	↑	↑	↑↑	↑
	Temperature	(+)	↓		↑		↑↑	
Targets for Engineering Requirements			15"/13"/12"	<$150	100 watts	WIFI	140F - 250F	QR Code

Correlations	
Strong Positive	↑↑
Positive	↑
Strong Negative	↓↓
Negative	↓
No Correlation	

Figure 1. House of Quality

2.6 Block Diagram

The following figure is our block diagram for the project. The figure displays the main operations of the HotBox, color-coordinated by which teams/members worked on it.

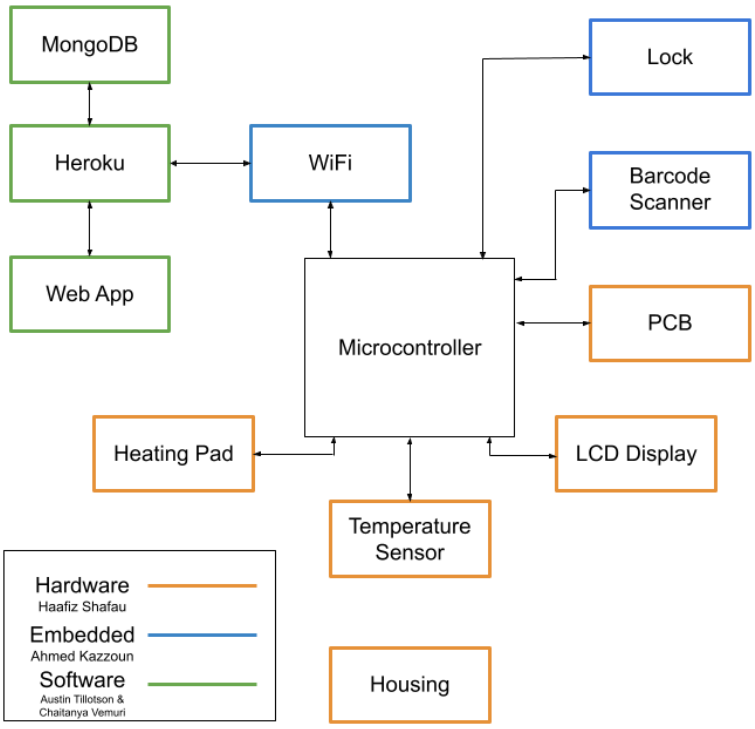


Figure 2. Project Block Diagram

3. Research related to Project Definition

This section of the document will contain all the research we did for our project. The subsections for this include similar existing projects, microcontrollers, hardware research, and software research. The similar existing projects section will contain projects we found that could be related to ours in some way. These projects will be useful in understanding the requirements and process we will need in order to develop our project and ensuring that our project has something unique to offer that you cannot get from these other projects. The microcontroller section will contain research that was specifically done on microcontrollers, including microcontrollers that could be used for what our project will require and which would be best. The hardware research section will contain all relevant research related to hardware parts that our project will need, from the material for the casing to the specifics on how to heat the box. The software research section will contain all relevant research related to the software side of our project, from which languages would fit best for our project to what Stack would best be used for our demo application. The hardware and software research sections will also contain subsections related to part selection, explaining what parts we decided to go with and why we chose that part of the other options.

3.1 Similar Existing Projects

Heating and keeping food warm is not a new problem and many projects have been made to try and solve this problem. We plan on analyzing these past designs, figuring out what went right and wrong, the technologies used, and other implementations to aid us when building our design. After extensive research on past projects, we were able to figure out how we want to implement and build our design as well.

3.1.1 HOTLOGIC Food Warming Tote/Lunch Box

The HOTLOGIC Food Warming Tote/Lunch Box is a portable lunch box that plugs into the wall and heats up the food using a heating plate. It can even cook food if desired as well which means the box reaches ideal temperatures to keep food warm, even better hot. The box is also compatible with a variety of materials including glass, plastic, metal, aluminum, and cardboard.

This design to keep food warm portably was smart however we did not want our implementation to be portable since the Hotbox will be stationary. The HOTLOGIC lunch box heating plate is similar to what we wanted to implement however the heating plate heats and warms to a set temperature that is unable to be changed which is not practical for our design. The interior of the HOTLOGIC Lunch Box which is lined with aluminum to retain heat dissipation is something we do want to implement in our design.

Figure 3. HOTLOGIC Food Warming Tote/Lunch Box. Permission requested from HOTLOGIC

3.1.2 Programmable Temperature Controller + Hotplate

When we were researching into the heating component of the box, we came across the Programmable Temperature Controller + Hotplate (PTCH) made by BrittLiv. It implemented a similar heating pad/plate design that we designed into our box.

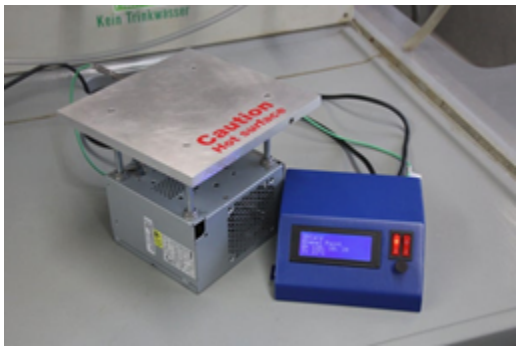


Figure 4. Programmable Temperature Controller + Hotplate. Permission requested from BrittLiv

The project was designed to solve heating tasks in a laboratory environment where exact temperature control is very crucial in chemical processes. It was able to run temperature ramps and read multiple different temperature programs from a simple SD card. This project was also low in cost as the overall cost of the controller was $45 and the hot plate $50. The programmable controller consisted mainly of the electronics cased in plywood. The hot plate mainly consists of an aluminum plate and cartridge heaters and a thermocouple used for the heating.

3.2 Microcontroller

The main starting point for the electronics for the box is the microcontroller chip. For our project, there are certain criteria we need our chip to be able to meet for the box to function the way we plan. Some of the criteria are listed below.

- We need a sufficient amount of **I/O pins** for other hardware components to be added
- We need the chip to be **cost** efficient
- We need the chip to be able to have a wide range of **support** to work with other parts
- Programming Language of the chip
- Architecture of the chip
- Speed

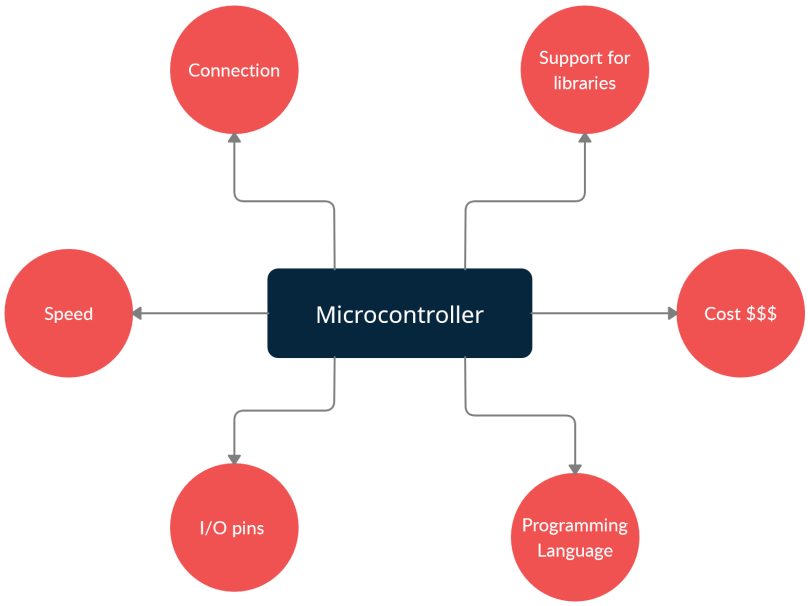


Figure 5. Microcontroller Criteria

Since our box will feature different sensors and components that will communicate with the microcontroller chip, I/O is especially important. We need a chip that has a sufficient amount of pins for all these features and functions to work. And the way the chip will communicate to the components is through the different standard communication protocols available on the chip.

3.2.1 Communication Protocols

Communication protocols are essential to understand when it comes to utilizing the microcontroller. In this section we will take a look at the three protocols **UART**, **SPI**, and **I2C** available to us and go over the pros and cons of each one to further understand each one.

3.2.1.1 UART

The Universal Asynchronous Receiver/Transmitter (UART) protocol is an asynchronous protocol meaning there is no clock necessary when transmitting and receiving data. Instead, there is a start and stop bit added when transmitting data, so the receiver knows when to start reading and when to stop reading the data as well. UART can either be utilized in a full or a half-duplex setting. Below is an example of UART being done in a full-duplex setting.

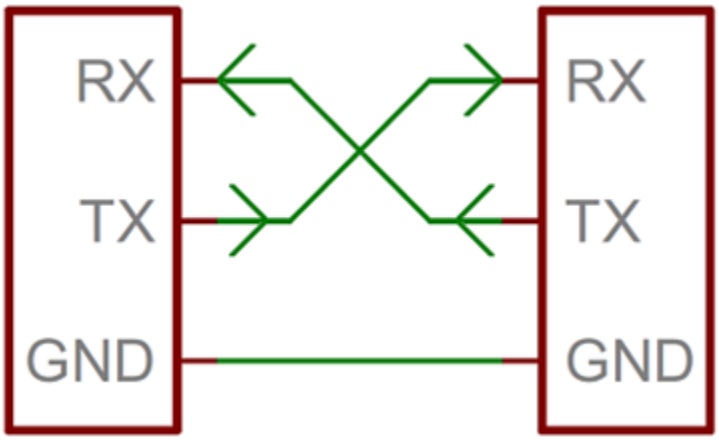


Figure 6. Full-duplex UART example

Below is a chart showing some of the main characteristics of data transmission of the UART protocol.

Table 3. Characteristics of Data Transmission for UART Protocol

Start Bit	The bit that starts the transfer of data. Usually, transmission of data occurs when the transmitting UART switches from the high voltage level it was defaulted at, to a low in one clock cycle.
Data Frame/Size	The actual data being transmitted and/or received.
Parity Bit	Optional bit for error checking.
Stop Bit	The bit that signals the end of the transmission, driving the transmission from a low voltage to a high.
Baud Rate	Rate at which information or data is transferred in a communication channel.

Advantages of UART:

- Only need 2 wires for usage. (Transmitter Tx and Receiver Rx wire)
- No clock signal is needed.
- Parity bit present to allow for error checking.
- The way the data packet is structured can be changed to whatever is desired as long as both ends are set up for it.
- Very popular protocol method and is well documented.

Disadvantages of UART:

- Does not support multiple slave or multiple master systems.
- Data frame size limited to a maximum of 9 bits.
- Baud rates of each UART must be within 10% of each other.
- Slower than the other two protocols (I2C and SPI)

UART is generally the most straightforward protocol out of the other 3 due to no clock being present. Devices that will be connected using UART include the 2D Barcode Scanner and the Wi-Fi Module.

3.2.1.2 I2C

The Inter-Integrated Circuit (I2C) protocol is a synchronous communication protocol meaning all the connected sensors and devices share the same clock signal. I2C configurations usually consist of multiple devices or sensors connected to a single master. The master device is the source of all communication as well as the main clock signal that the rest of the devices will use. Like UART, I2C only requires two wires for usage.

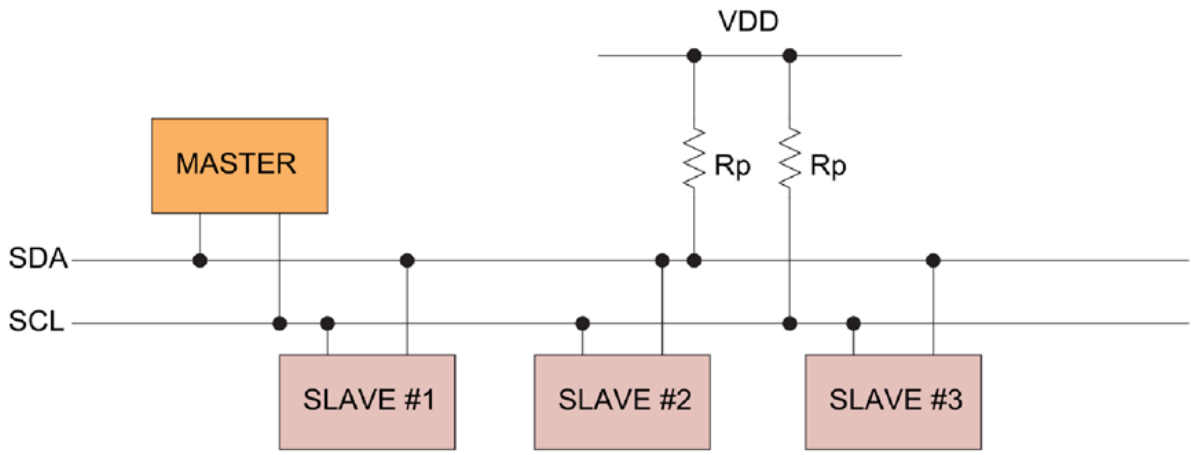


Figure 7. I2C Operation Diagram

Below is a table of the main characteristics of data transmission regarding the I2C communication protocol.

Table 4. Characteristics of Data Transmission for I2C Protocol

SDA (Serial Data)	The line for sending and receiving data.
SCL (Serial Clock)	The line that holds the clock signal.
Start Condition	The SDA line flips from a high voltage to a low **before** the SCL line flips from high to low.
Stop Condition	The SDA line flips from a low voltage to a high **after** the SCL line flips from low to high.
Address Frame	7 or 10 bit sequence that is unique to each device connected to the master that identifies when communication occurs.
Read/Write Bit	Bit to specify whether master is sending or receiving data.
ACK/NACK Bit	Acknowledge/No-acknowledge bit.

Advantages of I2C

- Only two wires needed for usage.
- Can have multiple masters and devices connected to the master.
- Bit confirmation for successful transfers (ACK/NACK)
- Less hardware than UARTs

Disadvantages of I2C

- Slower speed when compared to SPI.
- Data frame size limited to 8 bits.
- More hardware when compared to SPI.

I2C is a very convenient protocol because multiple devices can be connected to just one singular master clock so it saves a lot of time and wiring as well. Devices that will utilize the I2C protocol will be the LCD Display and the temperature sensor.

3.2.1.3 SPI

SPI (Serial Peripheral Interface) is another synchronous communication protocol where one or more devices are connected to a single master clock. However, unlike I2C that can be run in either full or half duplex, SPI can only be run in full duplex which means we need two wires for communication between the master and the other device. This configuration also doesn't allow multiple master configurations since there is only one master clock.

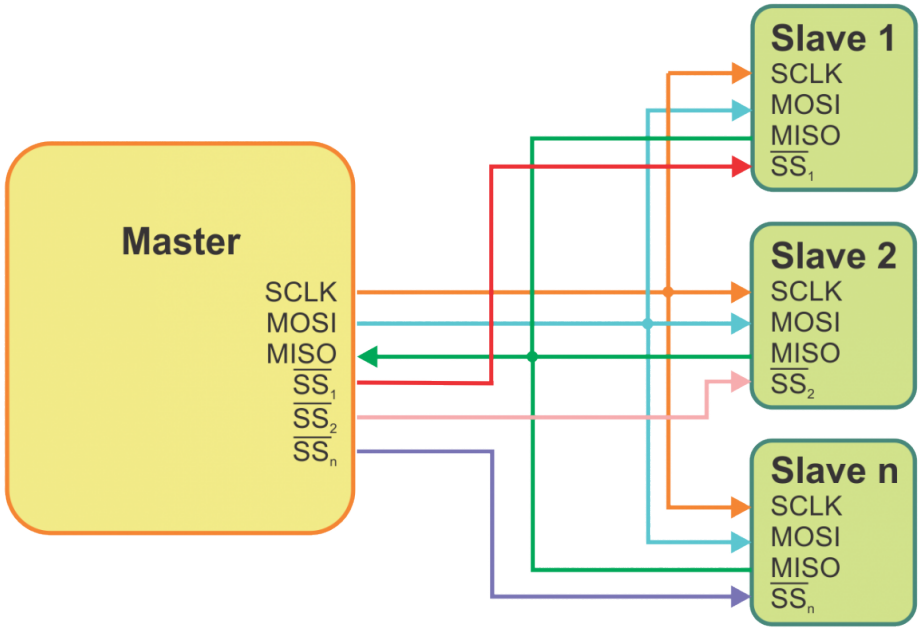


Figure 8. SPI Operation Diagram

Below is a table of the main characteristics of data transmission regarding the SPI communication protocol.

Table 5. Characteristics of Data Transmission for SPI Protocol

MOSI (Master Output/ Slave Input)	Line for the master to transmit data to the other device.
MISO (Master Input/ Slave Output)	Line for the device to transmit data to the master.
SCLK (Clock)	Clock signal
SS/CS (Slave Select/ Chip Select)	Line for the master to choose which device to transmit data to.

Advantages of SPI

- Since there is no start or stop bits, communication can be done continuously without interruption.
- No need to address other devices like I2C
- Higher data transfer rate when compared to I2C
- Data can be sent and received at the same time due to separate MISO and MOSI lines.

Disadvantages of SPI

- Uses four wires compared to only two in SPI and UART.
- No ACK/NACK for successfully received data.
- No error checking
- Only allows a singular master device.

3.3 Hardware Research

This subsection contains all the relevant research that was done for the hardware side of the project. This research is primarily focused on finding parts that could be used and implemented in our design to create the HotBox. Their parts are then all compared at the end of their relevant sections. The section ends with a selection summary of the parts, explaining what parts we chose for the project and why we chose them over other options.

3.3.1 Relevant Hardware Parts

In this subsection, we will be discussing some of the hardware components that are relevant to our product. We will be looking at many components like pre-built containers that will hold the food when it is warm. We will be also researching the different materials of those containers. Also, we will research some of the electronic components that will go in our product like infrared sensors and heating pads.

3.3.1.1 Cambro GoBox® Half Size Top Loader Insulated Food Pan Carrier

When researching a good case to use for our project, we came across this product. In order to house electronics and keep food warm without any heaters, a case that can support high temperatures is needed. This case insulates food and eventually, we can modify the case to house electronics and wires for the heating pad we plan to install.

Made using an innovative EPP material(see section 3.3.1.1.1), this Cambro food carrier is ideal for new catering companies or delivery services. It is light enough to hold above your head with one hand.

The Cam GoBox features a professional grade construction made from expanded polypropylene (EPP), an insulated material with a high level of durability. This high-tech

material is extremely lightweight while maintaining optimal impact and chemical resistance, and it is also eco-friendly.

The insulation quality of this carrier is ideal for transporting your food for extended periods of time. It will hold cold food below 41 degrees Fahrenheit or hot food above 135 degrees Fahrenheit for up to four hours at a time. The walls have recesses on the inside so you can get a good hold of your food pans while removing, and the straight walls allow you to stack food pans with ease.

It is able to fit up to an 8" deep half size food pan and it can withstand heavy-duty use and abuse without being damaged

This table shows all of the product details of this container such as the weight and the dimensions of the container. This table is important to show the specifications of the container to see if it can handle the heat we will make and could store the electronics.

Table 6. Cambro GoBox Details

Weight	1.54 lb
Width	15 2/5 Inches
Height	12 2/5 Inches
Depth	13 Inches
Material	EPP Polypropylene (Expanded Polypropylene Foam)
Capacity	23.6 qt
Color	Black
Interior Height	10 Inches

Interior Width	13 Inches
Interior Depth	10 4/5 Inches
Application	Heavy Duty
NSF Listed	Yes
Loading Style	Top Load
Hazardous Material	No
Power	Non-Electric
Product Line	Cam GoBox

3.3.1.1.1 Expanded Polypropylene (EPP)

Our project requires a container for the food to be stored in. We needed to find the best type of container material for the exterior while housing some of the electronics of the device. In this section we will be discussing the advantages and disadvantages of using this material.

Expanded Polypropylene (EPP) is a highly versatile closed bead foam that provides a unique range of properties, including outstanding energy absorption, multiple impact resistance, thermal insulation, buoyancy, water and chemical resistance, exceptionally high strength to weight ratio and 100% recyclability. EPP can be made in a wide range of densities, from 15 to 200 grams per litre, which are transformed by moulding into densities ranging from 18 to 260 grams per litre. Individual beads are fused into final product form by the steam chest moulding process resulting in a strong and lightweight shape.

EPP was first developed in the 1970's as a result of research into new forms of polypropylene. The material's first applications were for automotive products in Japan in 1982. The demand for EPP has only increased since then. It is needed because the auto makers try to improve energy management functions whilst reducing weight and improving environmental benefits. The first application for this material was for an energy absorbing component in a bumper system of an automobile. EPP is now widely used for numerous other automotive parts and systems, including seating and other interior components.

Manufacturing this material is complex however. It requires both technical expertise and specialised custom equipment. Polypropylene resin is combined with other ingredients in a multi-step proprietary process. Under tightly controlled conditions, extruded pellets expand to become consistently shaped beads of expanded polypropylene foam. Other specialised manufacturing techniques may be employed to produce variations in the final product form.

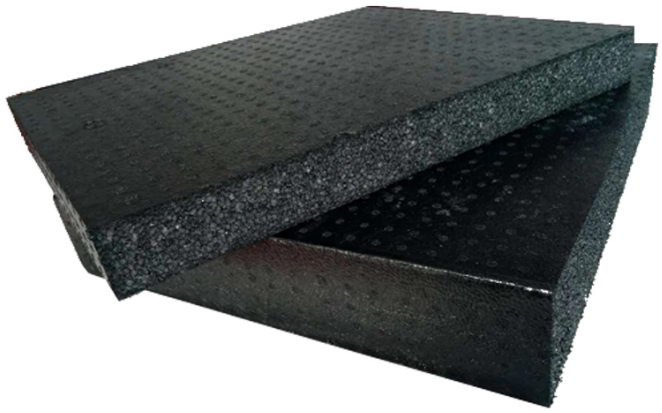
EPP has been recently approved for use in conjunction with food products. Its thermal insulation properties and structural strength make it appropriate for containers such as food delivery containers and beverage coolers and the like. EPP does not support microbial growth and can be made sterile with steam cleaning.

This table below shows the physical properties and chemical properties of this specific material. As shown, this material is resistant to many harmful things that could damage the product if it wasn't made specifically for that. The table is from … that conducted individual tests to test all of these properties.

Table 7. Properties of Expanded Polypropylene

Physical Properties	**Resistance to Chemicals** Exposure Media 7 days immersion at 22°C KEY; 1 = no change 2 = slight swelling
EPP density range, from 20 g/l through 200g/l	Petrol / Gasoline 2
Tensile strength (kPa) 270 to 1930	Gas-oil 2
Tensile elongation (%) 21 to 7.5	Kerosene 2
Compressive strength (kPa)	Mineral oil 2
25% strain 80 to 2000	Toluene 2
50% strain 150 to 3000	Ethyl alcohol 1
75% strain 350 to 9300	Methyl ethyl ketone 2
Compressive set (%)	10% Sulfuric acid 1
25% strain, 22H, 23°C 13.5 to 10.5	10% Nitric acid 1
Burning rate (mm/min) 100 to 12	10% Hydrochloric acid 1

This figure shows a small piece of the material EPP. As seen, the material looks like a stronger foam. This “foam” is resitenat to many chemical properties as shown in the table above and will be a good product for our project.



3.3.1.2 End Loading Insulated Food Pan Carrier

Another potential case that the device will be built on is the End Loading Insulated Food Pan Carrier. This carrier features a strong sealing performance and superior insulation, it can keep food hot or cold for over 8 hours without any modifications. Two hidden handles and stackable design facilitates movement, making the food pan carrier take up less space. The 95-quart capacity allows you to place a large number of hot meals or cold drinks and other items. It can be used in restaurants, canteens and other places.

This food pan carrier has a seamless double shell and is filled with high-quality commercial-grade polyurethane materials for achieving optimal insulation. The door is designed with an inlaid sealing ring so that the sealing performance can reach its maximum potential. It will maintain the desired temperature for over 8 hours.

Coming in with a net weight of 35 pounds, this 95-quart food pan carrier is equipped with 12 levels that can hold 6 full-size food pans of hot or cold food, allowing you to place different sizes of food pans or modify it with heating pads for our project.. It can store various hot dishes or cold drinks and is suitable for canteens, restaurants, and other places. The overall dimensions of this box is 27" x 19" x 23.5" (L x W x H) and the interior dimensions is 21" x 13.5" x 20" (L x W x H).

The manual insulated food pan carrier is equipped with two hidden handles for convenient movement. The stackable and interlocking design will minimize the needed space. The maximum angle of opening the door is 270 degrees, so you can easily take the items out.

This insulated pan carrier can keep multiple pans of food at safe temperatures for hours without electricity and has zero energy consumption. In addition, this feature allows for energy conservation and environmental protection. Implementing an electrical component in this device could prove simple to implement.

3.3.1.3 Case Material Comparison

The following table is a comparison of the features and materials of the above cases.

Table 8. Case Material Comparison Table

Features	Cambro GoBox® Half Size Top Loader Insulated Food Pan Carrier	End Loading Insulated Food Pan Carrier
>30 qt		✔
EPP material	✔	
<30 pounds	✔	
Can keep food warm for over 8 hours	✔	✔
Has a door that has a hinge		✔
No electricity involved	✔	✔

3.3.1.4 Barcode Scanner

A barcode is used to encode information in a visual pattern readable by a machine. Barcodes are used for a variety of reasons including tracking products, prices, and stock levels for centralized recording in a computer software system.

A barcode scanner usually consists of three different parts including the illumination system, the sensor, and the decoder. In general, a barcode scanner “scans” the black and white elements of a barcode by illuminating the code with a red light, which is then converted into matching text. More specifically, the sensor in the barcode scanner detects the reflected light from the illumination system (the red light) and generates an analog signal that is sent to the decoder.

The decoder interprets that signal, validates the barcode using the check digit, and converts it into text. This converted text is delivered by the scanner to a computer software system holding a database of the maker, cost, and quantity of all products sold.

Because barcode scanners are variable and include diverse capabilities, some are better suited for certain industries due to reading distance and to work volume capacity. There are multiple types of barcode scanners.

Restore Factory Setting

3.3.1.4.1 Pen-type Reader

Pen barcode readers resemble small wand-type sticks that resemble a small pen. The pen-style barcode reader consists of an LED light and a photodiode in its tip. The user passes this tip over a barcode and the LED light illuminates the black and white bars. The photodiode measures the reflection of light and is able to determine the width and color (white or black) of each bar. This information allows for a digital reading of the barcode, and information is transmitted to another unit for processing.

Pen or wand barcode readers are designed for durable, inexpensive use by a single user who can quickly scan packages or other items. However, because of the shakiness and imprecision of human use, the user may need some practice to perfect the scan.

3.3.1.4.2 Laser Scanner

Slightly more advanced than a pen scanner, a laser barcode scanner is capable of more exact light readings which prevent false positives or scanner errors. In a laser scanner, a laser beam is shot at a mirror inside the actual unit. This mirror makes a movement so that the laser sweeps across the barcode in a straight line. This light then reflects back to a diode, which measures the level of reflection. This reflection is translated into a digital signal readout of the barcode. Laser scanners can either be mounted in a scanning unit or be part of a handheld unit.

3.3.1.4.3 CCD Reader

A charge coupled device (CCD), also known as an LED scanner, features hundreds of tiny LED lights arranged in one long row. These lights are shot directly onto a barcode, and a sensor then measures not the reflection, but voltage of the ambient light directly in front of each lightbulb. This voltage measurement provides a digital snapshot of the barcode. CCD units can be very expensive, but are highly accurate and versatile pieces of equipment.

3.3.1.4.4 2D Camera

Some barcodes do not consist of white and black bars, but white and black spaces in a two-dimensional (2D) target. These 2D barcodes cannot be read by standard machinery, but they do allow for versatility of information coding as they can hold and provide much more data than a standard barcode. To read these barcodes, a 2D camera image scanner is necessary. This camera consists of hundreds of tiny lights like the CCD scanner, but these are arranged in multiple rows. The lights flash onto the barcode and take a digital picture of the barcode, which is then sent to software as a digital signal. The software then decodes the information.

3.3.1.4.5 GM65 barcode scanner

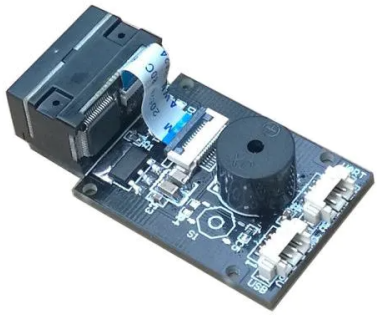
The GM65 barcode scanner identifies a product by its barcode. Then we enter the quantity of the product on the Nextion touch screen display and send it via WiFi to the server, where we enter it in the database. This device allows you to automate the process of inventory of goods.

The secret of the device is a combination of a digital camera and an image processing module. It has an algorithm that recognizes barcodes and QR codes in the field of view of the camera, and if it does not have enough external lighting, the built — in LED backlight comes to the rescue. A light marker in the form of a red stripe is provided for precise pointing at the scanner's barcode.

We bring the barcode at a distance of about 20 cm to the lens, and a characteristic buzzer signal is heard, as at a supermarket checkout. To speed up the process, you need to align the barcode plane perpendicular to the scanner. The maximum deviation angle is 60 degrees.

The barcode scanner works directly with the computer without any applications: it pretends to be a typical USB keyboard and outputs the data in the form of a string of text. This is the default operating mode.

You can configure the scanner using UART commands, but it is much easier to use service QR codes: switch reading modes, control the LED and buzzer, save and reset settings by simply targeting the corresponding QR code in the device instructions. This allows you to change the configuration on the fly.



3.3.1.4.6 Barcode scanner Comparison Table

The following table displays the advantages and disadvantages of each of the barcode scanners against each other.

Table 9. Barcode Scanner Comparison Table

	Advantages	Disadvantages
Pen-type Reader	• Durable • Inexpensive	• Not meant for products • Meant for smaller scale use
Laser Scanner	• **Fast**: Laser scanners can decode extremely fast. • **Accurate**: Laser scanners scan most 1D barcodes with ease. Just point and scan for quick results. • **Powerful**: Unlike a CCD scanner, laser scanners can read large barcodes and from longer distances. Some laser scanners can read barcodes from over 50 feet away.	• **Ambient Light**: Because a laser scanner functions by reflecting light back, high ambient light may cause difficulties when trying to scan. • **Laser Limitations**: Most laser barcode scanners cannot read barcodes from a computer or mobile device display. Furthermore, if the barcode is damaged, faded or too small in scale, the laser barcode scanner may not be able to capture the scan.
CCD Reader	• **Durable**: CCD scanners have no internal moving parts so they are typically a bit more durable than a laser scanner. • **More economical**: CCD scanners, on average, are usually less costly than other barcode scanners. • **Scan In Direct Sunlight & On Device Screens**: Due to this scanning process, CCD scanners are able to operate in bright sunlight. Most CCD scanners are also able to read scans off a computer or other devices' screens.	• **Barcode Size Limitations**: CCD scanners are unable to read barcodes that are wider than the imager line. • **Scan Distance Limitations**: Most CCD scanners are only able to read barcodes from a few inches away.

2D Camera	• **Versatility**: A 2D barcode scanner can scan virtually every barcode symbology. • **Powerful**: 2D scanners, because of their image capture processing, can read barcodes from nearly every surface type including computer or mobile device screens, and even scan through glass surfaces.	• **Price**: 2D barcode scanners are typically more expensive than linear imagers or laser barcode scanners.
GM65 barcode scanner	• Low power consumption of the module, operating current less than 150mA, integrated design, small size; • Support TTL232 and USB interface; • Supports all common 1D barcodes and common 2D codes to directly identify the phone screen.	• Requires WIFI connection

3.3.1.5 BoxLock Smart Lock

To have a secure device, we decided to research some smart locks for our product. The BoxLock Smart Lock is a great way to secure our device by implementing a way to unlock over WIFI and access barcodes. Here are some of the features that are included.

BoxLock Home is the first smart padlock designed to protect your deliveries from porch pirates and package thieves. Through BoxLock's APIs, operators can designate a lock to open remotely for a set period of time by pressing the button on the top of the lock.

This lock can communicate over 2.4GHz WiFi and Bluetooth Low Energy. With BoxLock 2M's "Press to Open" remote operators and network operations centers can open their BoxLock devices without the need for a barcode, mobile software or a key fob.

The lock measures 6.5 by 2.6 by 2.0 inches (HWD) and sports a hardened steel shackle, a tamper-resistant zinc interior casing, and a weatherproof fiber-reinforced outer shell. It has an internal rechargeable battery rated to last up to 30 days between charges, and comes with a USB charging adapter and cable, and a printed master barcode that you can use to open the lock if you don't have your phone handy.

The bottom part of the lock contains a barcode scanner that's activated by pressing the black button at the top of the lock. Inside is a Wi-Fi radio (2.4GHz) that connects the lock to your home network and a Bluetooth radio for close-range communications. The BoxLock will work with just about any size storage box with a hasp fastener; if you don't have one there are several links on the BoxLock website for purchasing plastic and wood boxes of all shapes and sizes.

Setting up the BoxLock is easy. You can download the app and create an account using your full name, email address, and zip code, and a password. Press Add a BoxLock and follow the steps to configure the lock. You can then power up the lock to initiate search mode, select a Wi-Fi SSID from the list. The app asked for permission to pair with my phone's Bluetooth, asked for permission to send notifications, and setup was finished.

Figure 9. BoxLock Smart Lock

3.3.1.6 Electric Heating Technology

A lot of time and research went into the heating technology of the box. Most present heaters either use gas or electric and in 2021, electric would be the most practical and efficient means of energy when it comes to heating. Electric heating is also much better for the environment as well compared to heaters fueled with gas. Doing research into electric heaters, we found out that this part of the box would come out to be the most expensive and hard part of the project. There are a couple of options for heating when it comes to electric heating and in this section, we will go over the options we researched, their advantages and disadvantages, and go over the option we chose for our project and why.

3.3.1.6.1 Flexible Heaters

Flexible Heaters are devices that can generate heat electrically or chemically with flexible mechanical properties. Due to them being somewhat flexible in nature they can be bent and contoured (to an extent) and be adhered to different surfaces. This makes it ideal for curved and linear flat surfaces. They can be made with a variety of things including Flex PCBs, printed electronics, wires, etc. Each method meet a different specific set of standards for heating and performance as well as size, shape, and weight requirements.

3.3.1.6.2 Types of Flexible Heaters

There are three main types of flexible heaters on the market: Polyimide, Silicone, and wired flexible heaters. Main difference between all of these is the material and the construction of the heater themselves. Each of these types of flexible heaters offer different benefits to the end user and we will look at each to see the one that aligns most with our needs.

3.3.1.6.2.1 Polyimide Flexible Heaters

Polyimide flexible heaters are great for heating rigid or curved surface areas. Polyimide flex heaters consist of an etched circuit between two pieces of polyimide film. They are relatively slim in form and have great tensile strength and resilience. Meaning that they are very difficult to tear even with the small form factor of the heaters. Figure **X.** shows an example of a polyimide flex heater.

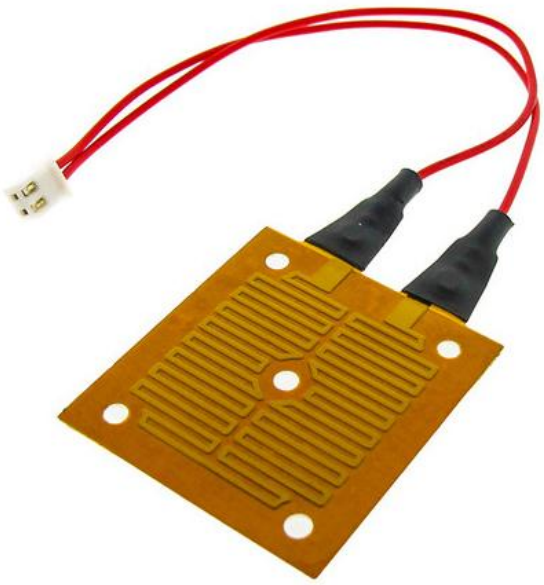


Figure 10. Polyimide Heater Example. Permission requested

Below as well shows some current specifications for polyimide heaters that are important for us to list off as well.

Table 10. Polyimide Heater Specifications

Maximum Watt Density	60 W/in^2
Maximum Operating Temperature	400F (200C)
Minimum Operating Temperature	-319F (-195C)
Wattage Tolerance	+- 10%
Dielectric Strength	1000 VAC

3.3.1.6.2.2 Silicone Flexible Heaters

Silicone flex heaters are great for large, flat surface areas making them great for industrial use. Their key point is their chemical resistance due to the silicone material used. The silicone flex heater consists of a circuit that is attached to power leads, then vulcanized between two layers of silicone rubber. Silicone is both an advantage and a disadvantage being scratch resistant and repelling most chemicals making it very good for scientific and industrial usage, however they are also thick and do not always have the best electrical efficiency. Figure **X.** below shows an example of a silicone flex heater.

Figure 11. Silicone Flexible Heater Example. Permission requested

Below as well shows some current specifications for silicone heaters that are important for us to list off as well.

Table 11. Silicone Heater Specifications

Maximum Watt Density	60 W/in^2
Maximum Operating Temperature	392F (200C)
Minimum Operating Temperature	-70F (-56.6C)
Wattage Tolerance	+- 10%
Dielectric Strength	1000 VAC

3.3.1.6.2.3 Wire Heaters

Wire Heaters are composed of wires that are usually sewn into textiles or carbon fiber in some applications. Because of this, they are usually harder, and more difficult to install, however they are cheap to produce. Products that use wire heaters include heated blankets, heated gloves, and heated jackets.

3.3.1.6.2.4 Flexible Heaters Comparison

Below is a table showing the characteristics between the three main types of flexible heaters. The highlighted column/s is the heater that would be most ideal for our project as well.

Table 12. Flexible Heaters Comparison

Characteristics	Polyimide	Silicone Rubber	Wire
Temp Threshold	-320F – 400F	-70F – 392F	-58F – 203F
Voltage	5V – 115V	12V – 600V	1V – 400V
Attachment Style	Adhesive	Vulcanization/ Adhesive	Sewn
Ideal in	Areas with tight spaces and harsh conditions, and do not require a lot of flexibility	Areas that require chemically resistant solution	Products that need a low price point on the market

3.3.2 Strategic Hardware Components and Part Selections

In this section of the paper, we talk about all the main parts and components and how they will work together in the project. The main research for different parts and components has been done and using that research we will create the PCB and other schematics. In this section, most of the hardware conflicts and complications will be worked out and dealt with for a complete and working hardware side for this project.

3.3.2.1 Power System

Due to the nature of the box and the varying different components, the power system for the components is going to require multiple different voltages. It is very important to talk about the varying voltages before talking about the rest of the strategic parts and components. There are a total of 4 different voltages needed in the box for all the components to fully work. For our heating pad, it requires 24V DC. The lock requires 12V DC. some of the components such as the microcontroller and Wi-Fi module require 5V and 3.3V. We will need to take the raw 120V AC coming from the wall, and convert it to 24V DC, then to 12V DC, and then from 12V DC to 5V DC and 3.3V DC. To do this we will be using the following to yield us the correct voltages needed. It is also important to note that we must also keep in mind of the wattage requirements needed for correct functionality.

- AC to DC Transformer (120V AC - 24V DC)
- Buck Converters
    1. 24V DC to 12V DC
    2. 12V DC to 5V DC
- DC to DC Voltage Regulator (3.3V)

3.3.2.1.1 AC to DC Transformer

In order to power the majority of our devices and components, we must take the raw 120V AC voltage coming out of the wall and transform it to something that is actually usable which begins with the 24V DC output. It is much safer, easier, and efficient to step down voltages rather than stepping up lower ones to higher voltages. We need to take the Alternating Current that oscillates and yield a Direct Current which will then be usable throughout the project. An AC to DC transformer does exactly this by using a transformer to lower the high AC voltage and then uses a full wave rectifier to yield the correct current conversion to DC. We will use this 24VDC voltage in the heating plate bed as well as convert this to be used in other devices and components.

Below is a table of some of the notable specifications of this transformer.

Table 13. AC to DC Transformer Specifications

Rated Max Power	350W
Input Voltage	110 ~ 240 VAC
DC Output Voltage	24 VDC
Item Weight	1.6 pounds
Product Dimensions (LxWxH)	8.4 x 4.5 x 1.1 inches

3.3.2.1.2 Buck Converters

To convert the rest of the voltages, we decided to use buck converters. Reason for this is because it is far more efficient due to the varying currents in the system. Comparing it to other converters, a voltage divider would be inefficient in our system due to all the varying currents. A linear regulator would be better than the voltage divider because it can take varying currents, however a lot of the current dissipates due to heat.

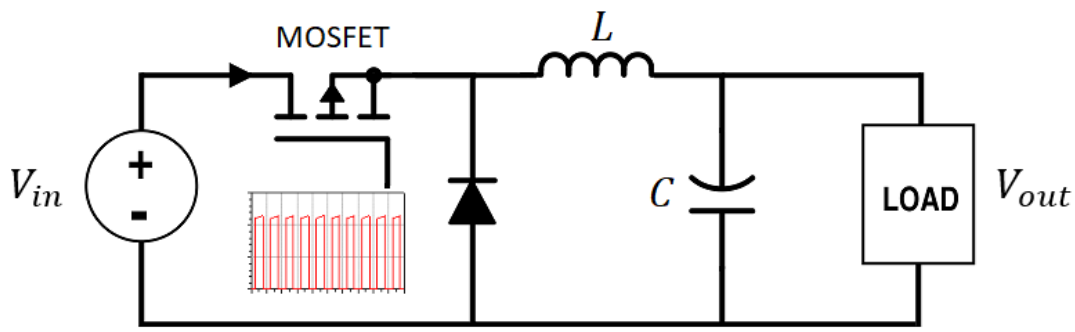


Figure 12. Buck Converter Circuit Configuration

Figure X. shows how the buck converter operates. It uses a switching device (MOSFET in this figure) that charges the capacitor as well as the inductor for a brief period of time and the charging current causes the inductor to produce a different voltage while charging, thus producing the net voltage drop across the load.

3.3.2.1.3 1D & 2D Barcode Scanner

For order verification, the importance of a working barcode scanner is essential. The DyScan Barcode Scanner scans both 1D and 2D digits which means we will have flexibility when scanning in customer order numbers. All the orders will be in the database and when the barcode is scanned and present in the database, the box will be

open. To power the barcode module, a supply voltage 3.3V will be necessary. For communication, either UART or USB which is embedded on the module itself. For our project, we decided that UART would be the best means of communication for us.

3.3.2.1.4 Pin Description

The Waveshare Barcode scanner is integrated with a built-in breakout board which makes it simple for us to connect and use the functionalities of the module. Present on the breakout board is a LED and a buzzer as well as a tactile push down switch which provides a trigger pin on the board. For our usage, we will use the pin header located on the board to gain access to the necessary pins for the barcode to function in the box. The pins include the TTL serial pins, power pins, and trigger pins. Below is Figure **13**. showing the pin layout on the breakout board as well as Table **14.** with the descriptions of the pins that will be used.

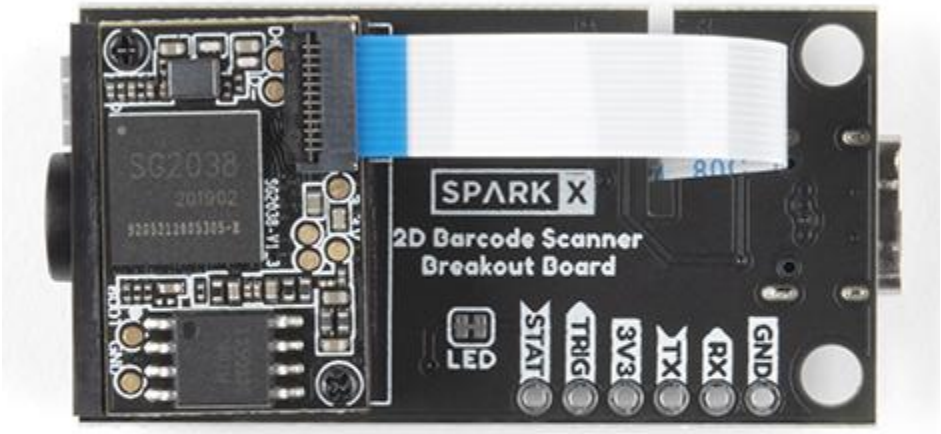


Figure 13. 2D Scanner Breakout Board. Permission requested

Table 14. Pin Descriptions of 2D Scanner Breakout Board

Pin#	Signal	Type	Definition
1	VCC	p	Power Supply, 5V
2	GND	p	Ground
3	RXD	Input	TTL-RS232 receive
4	TXD	Output	TTL-RS232 send
5	TRIG	Input	Trigger pin

3.3.2.2 Heating Technology

For the heating technology, we decided to opt in using a heating plate type of technology to heat up the box and the food. Rather than purchasing heating cartridges, which would yield us high temperatures, they would be rather expensive in practical use. And after researching the price of purchasing a slab of material and countersinking holes into it, it also was too expensive in practicality. After doing more research, we found out that a heated bed would work the best for the project. Below in Fig X. is an example of utilizing a heating bed with a solid-state relay (SSR), a power supply, and a microcontroller. We will be doing the same concept in our project as well.

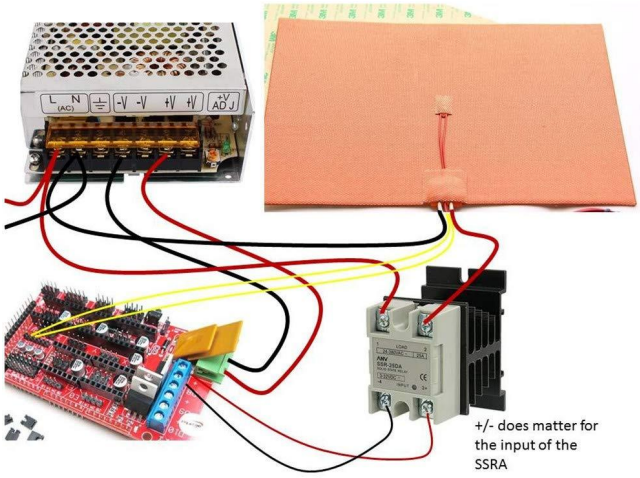


Figure 14. Heating Bed Wiring Diagram.

Heated Beds typically are used in 3D printers to heat up the surface the print will rest on and in doing so ensure the print adheres to the surface as well as keeping the print warm. That way the print will not come loose due to unavoidable warping forces, resulting in a nice and even print job. Which means that the heated bed needs to be able to heat amply across the entire surface for short to long durations of time to ensure a successful and well-done print. For our project, we need the plate to be able to hit our target temperatures in a reasonable amount of time as well as hold that temperature for short to long durations. We also need to make sure that the plate is a material that will not get affected by different materials at different temperatures in any way. Luckily, heated beds come in various materials, and the main one we were searching for is aluminum. A lot of heated beds also come with a thermistor attached to the back of the heated bed which makes temperature regulation much easier and simpler. Below are

some of the options we looked at before making our final decision, as well as depictions for the three options.

3.3.2.2.1 Creativity 3D Printer Heated Bed

The creativity heated bed features an upgraded full gold processed hot bed plate compared to the one present in the Ender3 3D printer which is a spray tin plate. This one is insulated with cotton which improves heat conduction as well as heat-resistance performance. It features high sensitivity, rapid response, good stability, and high reliability. It is compatible with various types of 3D printers (Ender 3, Ender 3X, Ender 3 Pro) but for our usage, only the size matters not so much the 3D printer. We plan on planning the box around the size of the heating plate and this heating plate comes in two options, 235mm x 235mm (9.25 x 9.25 inches) or 310mm x 310mm (12.20 x 12.20 inches) and is 3mm (.118 inches) thick.

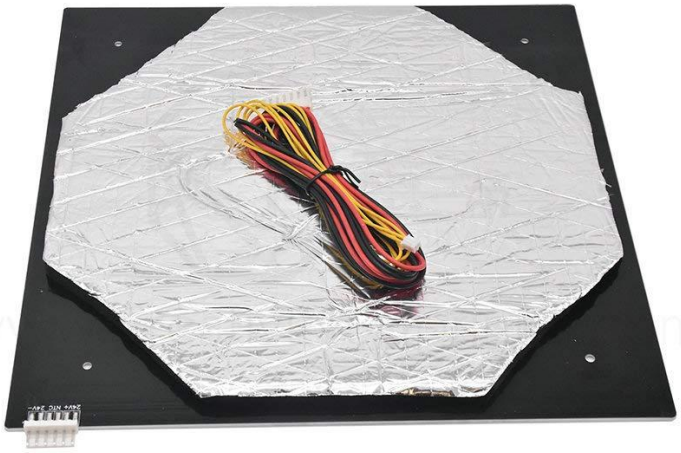


Figure 15. Creativity 3D Heated Bed. Permission requested

The Fig X. above showcases the Creativity Heated Bed. The plate requires a supply voltage of 24VDC and features open circuit detection to prevent harm to the rest of the project.

Below are the specifications of the Creativity 3D Printer Heated Bed summarized in tabular form.

Table 15. Creativity 3D Printer Heated Bed Specifications

Specifications	
Supply Voltage	24V
Operating Current	<14.6Amps
Manufacturer	Creativity
Dimensions (LxW)	235mm x 235mm
Thickness	3mm
Maximum Temperature	130C (266F)

3.3.2.2.2 SIMAX3D CR10 Aluminum Heated Bed

The CR10 aluminum heating bed is very similar to the Creativity heating bed with a few minor differences. The material of the plate is just made up of purely aluminum which is fine for our usage. Some more notable features include rapid heating up to 100C (212F), max 130C (266F).

Figure 16. SIMAX3D CR10 Aluminum Heated Bed.

In the figure above shown is the SIMAX3D CR10 Aluminum Heated Bed. On top of the plate lies a protective film to protect the plate from any scratches before usage. The protective film should be taken off before using the product to prevent any mishaps.

Below are the main specifications of the heating bed.

Table 16. SIMAX3D CR10 Aluminum Heated Bed Specifications

Specifications	
Supply Voltage	24V
Work Power	220W
Manufacturer	SIMAX3D
Dimensions (LxW)	310mm x 310mm (12.20 x 12.20 inches)
Thickness	3mm
Maximum Temperature	130C (266F)

3.3.2.2.3 RICHOOSE 3D Printer Silicone Heated Pad

The third and final option we decided to look at was the 3D Printer silicone heated pad manufactured by RICHOOSE. This heating pad is made out of Arlon silicone rubber substrate so is resistive to other chemicals and moistures that could otherwise affect the pad. It also allows the pad to distribute heat evenly across the pad as well and provides excellent heating performances for long durations of time at high temperatures.

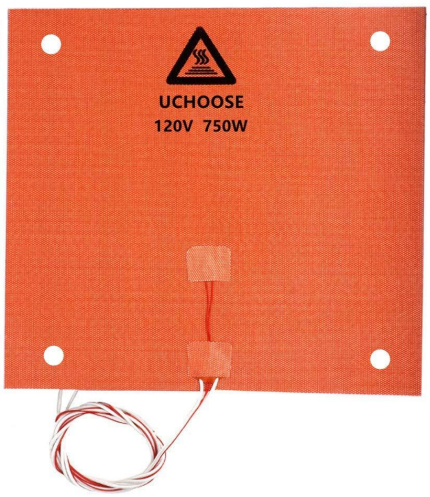


Figure 17. RICHOOSE Heating Pad.

The figure above showcases the RICHOOSE Heating Pad. The pad can adhere to multiple surfaces using the 3M adhesive, but aluminum is the surface recommended for the best heating results. The dimensions of the pad are the same of the other two heating pads, 310mm x 310mm (12.20 x 12.20 inches). The thickness of the pad is about 1.5mm however it relies more on the surface it is adhering to. Below shows a table rounding up the rest of the specifications of the pad.

Table 17. RICHOOSE 3D Printer Silicone Heated Pad Specifications

Specifications	
Supply Voltage	120V
Work Power	750W
Manufacturer	RICHOOSE
Dimensions (LxW)	310mm x 310mm (12.20 x 12.20 inches)
Thickness	1.5mm (.059 inches)
Working Temperature	-62C – 235C (-79.6 – 455F)
Mounting process	Adhesive + Screws

3.3.2.2.4 Heating Technology Comparison

In this section we take a side by side look of all the heating technologies researched and come to a decision about which choice we decided to go with. All the important specifications to us are listed in a tabular for easy comparison to each other and the highlighted column is the one we decided to go with. The specifications are important to note as it will determine which ones best suit our final end product.

Table 18. Heating Technology Comparison Table

Specifications	Creativity 3D Printer Heated Bed	SIMAX3D CR10 Aluminum Heated Bed	RICHOOSE 3D Printer Silicone Heated Pad
Supply Voltage	24V	24V	120V
Max Operating Current	<14.6 Amps	<9.16 amps	<6.25 amps
Thermistor Included?	Yes	Yes	Yes
Dimensions	310mm x 310mm	310mm x 310mm	310mm x 310mm

Thickness	3mm	3mm	1.5mm
Mounting Type	Screws	Screws	Adhesive + Screws
Working Temperature	130C (266F)	130C (266F)	-62C – 235C (-79.6 – 455F)
Price	$20.99	$39.99	$36.99

The main reason we chose the Creativity 3D Printer Heated Bed was due to a couple reasons. One was the fact that we did need a super powerful heater for our use, so the RICHOOSE was out of the question despite it being able to reach extreme temperatures. The fact that the RICHOOSE also needed 120V to be supplied to it would have been overkill again for our project. The reason the Creativity was chosen over the SIMAX3D option was because of the price as well as the added insulated cotton for improved heat conduction and heat resistance as well.

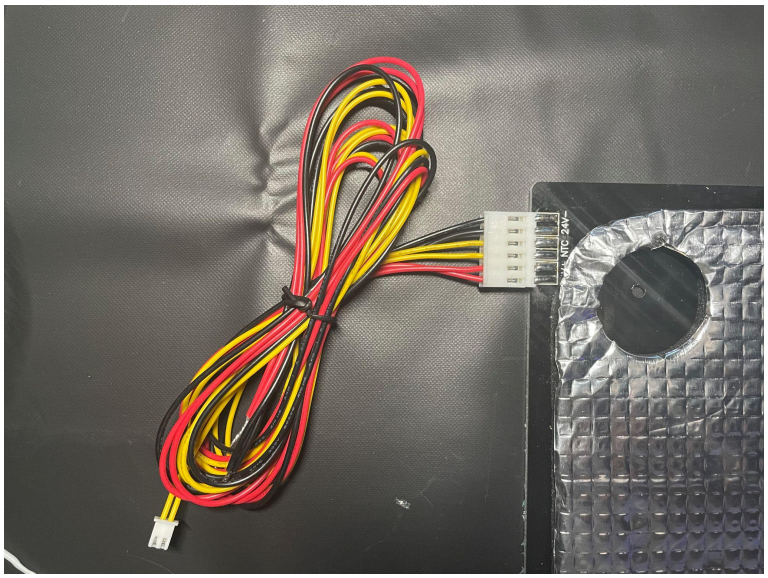


Figure 18. Heating Pad Connections

The figure above shows the connections that will be used on the heating pad. The heating pad will be connected using a 5V relay so that the microcontroller will be able to control the heating of the pad, the NTC thermistor will be connected to the microcontroller itself in order to track and monitor the temperature of the heating pad as well. Due to the pad requiring a supply voltage of 24V, we must find an appropriate power source in order to power on the heating pad.

3.3.2.4 RGB backlight positive LCD 20x4

This LCD display has many features. Instead of just having blue and white, or red and black, this LCD has black characters on a full color RGB-backlight background. That means you can change the background color to anything you want - red, green, blue, pink, white, purple yellow, teal, salmon, chartreuse, or just leave it off for a neutral background. This LCD is the most daylight readable character LCD we have and is very beautiful and easy to read no matter what color/brightness you have for the backlight.

This table below shows some of the specifications of the RGB backlight positive LCD 16x2. The table shows some essential information like the dimensions of the actual LCD and the amount of pins and ports. It also shows the languages you can program in and the electronic details like the voltage and current required.

Table 19. RGB backlight positive LCD technical features

Dimensions	38mm x 84mm / 2.3" x 3.9"
Design	Black text on multi-color background
Connection port	Connection port is 0.1" pitch, single row for easy breadboarding and wiring
Display Length	20 characters wide, 4 rows
RGB LED	Single RGB LED backlight included can be dimmed easily with a resistor or PWM and uses much less power than LCD with EL (electroluminescent) backlights
Electronic details	Each R, G, & B LED has a 200 ohm resistor in series so you can power the backlight from 3V or 5VDC. R forward voltage is ~2.2V, G & B are ~3.4V
Languages	Built in character set supports English/Japanese text
Pins	Can be fully controlled with only 6 digital lines. (Any analog/digital pins can be used) and 3 PWM pins for the backlight

Extra Features	Up to 8 extra characters can be created for custom glyphs or 'foreign' language support. Comes with 10K necessary contrast potentiometer and strip of header

This RGB backlight positive LCD will be used in our project to display to the user whether or not their food is heated and ready to be picked up. This will be powered up and controlled by our arduino and PCB. When not being used, it will be turned off to save the LCDs

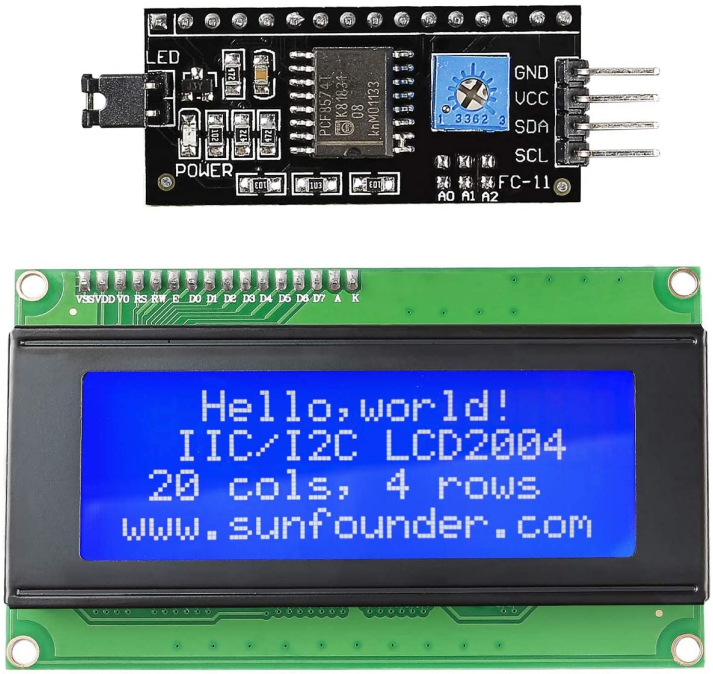


Figure 19. RGB backlight positive LCD Display

3.3.2.5 Adafruit Accessories Lock-style Solenoid

To have increased security on our device, we decided to use this particular lock. This lock is a solenoid lock that can be programmed by a MCU. This lock is small enough for a cabinet or lock.

Solenoids are basically electromagnets: they are made of a big coil of copper wire with an armature (a slug of metal) in the middle. When the coil is energized, the slug is pulled into the center of the coil. This makes the solenoid able to pull from one end.

This solenoid in particular is nice and strong, and has a slug with a slanted cut and a good mounting bracket. It's basically an electronic lock, designed for a basic cabinet or safe or door. Normally the lock is active so you can't open the door because the solenoid slug is in the way. It does not use any power in this state. When 9-12VDC is applied, the slug pulls in so it doesn't stick out anymore and the door can be opened.

The solenoids come with the slanted slug as shown above, but you can open it with the two Phillips-head screws and turn it around so it's rotated 90, 180 or 270 degrees so that it matches the door you want to use it with.

To drive a solenoid you will need a power transistor and a diode. You will need a good power supply to drive a solenoid, as a lot of current will rush into the solenoid to charge up the electro-magnet, about 500mA, you cannot power it with a 9V battery.

This figure is from the official website of the Adafruit Accessories Lock-style Solenoid. It shows the electronic schematics of the actual lock itself. In this figure, you can see where the arduino gets connected from and the MCU controlling it.

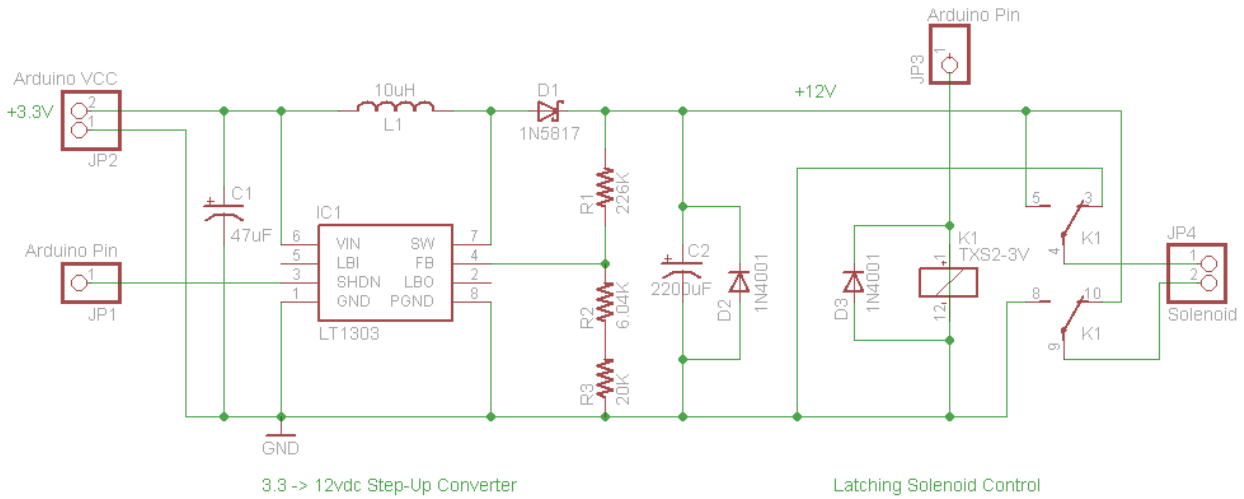


Figure 20. Solenoid Lock Internal Schematics

This table shows some of the actual technical features of the Adafruit Accessories Lock-style Solenoid. It is very important to have this because it shows the critical things like power required and how to use it properly. It also shows dimensions to make sure it can fit our product.

Table 20. Adafruit Accessories Lock-style Solenoid Technical Features

Power needed	Power required: 12VDC (you can use 9-12 DC volts, but lower voltage results in weaker/slower operation). Draws 650mA at 12V, 500 mA at 9V when activated

Usage	Designed for 1-10 seconds long activation time
Dimensions	0.92" x 2.65" x 1.08"
Max Dimensions (when opened)	1.64" x 2.1" x 1.08"
Wire Length	8.75”
Weight	147.71g
Extra Features	Slanted-cut Slug can be reversed to match application. Designed for a basic cabinet, safe, or door. Good strong mounting bracket

This figure shows how the lock is going to look. It is a really simple design. The solenoid lock is only accompanied by the connection wires which will be used to connect to the arduino. This lock will be in the front of our product and will wire to the electronics bay of our product.

Figure 21. Adafruit Solenoid Lock

3.3.2.6 Arduino ATMEGA2560

For our product, we decided to use an Arduino MEGA2560. The Arduino MEGA 2560 is designed for projects that require more I/O lines, more sketch memory and more RAM. When programmed, it is recommended for 3D printers and robotics projects.

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Arduino Mega 2560 is programmed using the Arduino Software (IDE), an Integrated Development Environment common to all the boards and running both online and offline.

Figure 22. Arduino Mega2560

3.3.2.7 ESP32-DEVKITC-32D

The ESP32-DEVKITC-32D is a WIFI card that will be used for our project. The purpose of a WIFI card is to communicate with the restaurant's web application to send notifications to the users to notify them when the food is ready to be heated. This is a brief description of the particular model we chose.

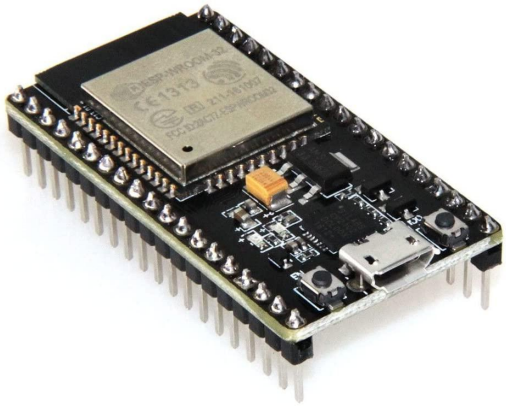


Figure 23. ESP32-DEVKITC-32D

The ESP32 is a single 2.4 GHz WiFi and Bluetooth module that integrates the TSMC ultra-low power technology. It's designed for performance, versatility, and reliability in a wide array of applications. There is a IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI

The ESP32 Development Board is made with the official WROOM32 module. There is a built in USB-to-Serial converter, automatic bootloader reset, Lithium Ion/Polymer charger. And just about all of the GPIOs brought out so we can use it with any sensor. That module contains a dual-core ESP32 chip, 4 MB of SPI Flash, tuned antenna. And all the passives we need to take advantage of this powerful new processor. The ESP32 has both Wi-Fi and Bluetooth Classic/LE support.

3.3.2.8 ELEGOO 4 Channel DC 5V Relay Module

A 4 Channel Relay Breakout is an easy way to use our Arduino to switch high voltages and high current loads. The board is 5V logic compatible and uses 4 digital outputs to control 4 individual relays. Each relay has the common, normally open, and normally closed pin broken out to a convenient 5.0mm pitch screw terminal. The contacts on each relay are specified for 250VAC and 30VDC and 10A in each case, as marked on the body of the relays.

Figure 24. 4-Channel 5V Relay

This is a 4-channel relay interface board, which can be controlled directly by a wide range of microcontrollers such as Arduino, AVR, PIC, ARM, PLC, etc. It is also able to control various appliances and other equipment with large current. This is widely used for all MCU control, industrial sector, PLC control, smart home control.

The following is the pinout table for the 4 Channel Relay Module.

Table 21. 4-Channel Relay Module Pinout

Pin Number	Pin Name	Description
1	GND	Ground reference for the module
2	IN1	Input to activate relay 1
3	IN2	Input to activate relay 2
4	IN3	Input to activate relay 3
5	IN4	Input to activate relay 4
6	Vcc	Power supply for the relay module
7	Vcc	Power supply selection jumper
8	JD-Vcc	Alternate power pin for the relay module

From the picture below, when the signal port is at low level, the signal light will light up and the optocoupler 817c (it transforms electrical signals by light and can isolate input and output electrical signals) will conduct, and then the transistor will conduct, the relay

coil will be electrified, and the normally open contact of the relay will be closed. When the signal port is at high level, the normally closed contact of the relay will be closed.

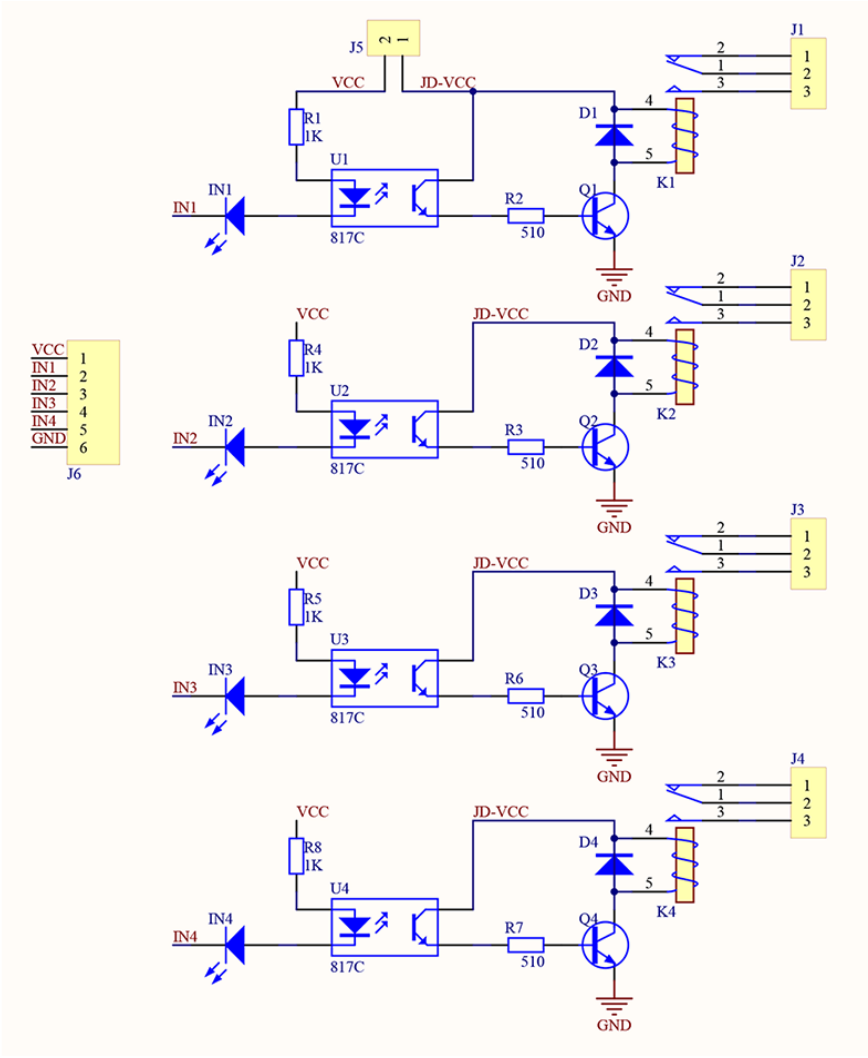


Figure 25. Relay Circuit Diagram

3.3.4 Part Selection Summary

Below are the specific parts we have chosen to create our product as discussed in section 3.3.3 and 3.3.4. These parts were specifically chosen because they are compatible with each other and work well. Each part connects to the MCU that we chose and is powered and controlled by that.

Table 22. Selected Parts Table

Item	Part #	Manufacturer	Cost (USD)
LOCK	a19042500ux0028	uxcell	$11.49
RELAY	EL-SM-006	ELEGOO Store	$7.99
LCD DISPLAY	I2C LCD1602	SunFounder Store	$8.99
ESP32	3-01-1287	HiLetgo	$10.99
12V DC ADAPTER	B015OCV5XO	Ksmile	$4.50
DIGITAL MULTIMETER	Vpro850L	AIRMARK	$11.88
MEGA 2560	LYSB01H4ZDYCE-ELECTRNCS	ELEGOO Store	$15.99
HEATING BED	B087C22PHM	Creativity	$18.99
AC/DC TRANSFORMER	LRS350W-24	NiuGuy	$36.90
TEMP SENSOR	N/A	*From local so no manufacturer*	$4.96
BARCODE SCANNER	DE2120	Sparkfun	$44.95
LINING	SmartSHIELD -3mm	Insulation MarketPlace LLC	$14.95

3.4 Software Research

This section includes all the research that was done for the software aspects of the project. This includes relevant software such as stacks to the various programming languages available to use for the given tasks to be done.

3.4.1 Relevant Software - Stacks

This section of the document will cover some stacks commonly used in web development, which will be important to know for the development of the demo web application used for testing the HotBox project. The stacks that will be covered are the LAMP and MERN stacks. Each section of the specific stack will cover what the stack is, a small description of each of its components, and the benefits of using the stack for a web application. The components of these stacks may be covered more in detail later when more research is done for exactly which components will be used in the final decision.

3.4.1.1 LAMP Stack

LAMP stack is one of the many stack options available to use for web applications. The LAMP stack has been around since the beginning of web development, being one of the first stacks as well as being open source. Being it is still often used today, the LAMP stack is a time-tested stack that has continued to be relevant through the years.

LAMP stands for Linux, Apache, MySQL, and PHP. These are the components that are used to build the LAMP stack. Linux is the stack's operating system, a well known open source operating system that is often used for its flexibility and configuration options. Apache is the stack's web server. Apache has been the most used web server on the public internet for many years. The server "processes requests and serves up web assets via HTTP", making it accessible via a URL for anyone that is in the public domain. MySQL is the database of the stack (IBM Cloud Education, 2019). This database is open source and stores information queried from the SQL language. PHP is the backend programming language for the stack that is also open source. With the help of Apache, PHP code can result in dynamic web pages for the application.

This stack has a strong appeal for its simplicity. Compared to other stacks, LAMP does not take much work to produce a working website. Along with it being time-tested, the stack is proven to be a stable option for web development. Due to the open source nature of the LAMP stack, it is quite flexible, allowing the choice of the components that best fit the needs of the application. LAMP is efficient to use, due to how long it has been around. The stack can be built off what others have done and customize parts already available, allowing development to take less time than working from scratch. Another appeal of the LAMP stack is that we are familiar with it, having used it in a previous class to create a small web application project.

3.3.1.2 MERN Stack

MERN stack is a variation of the MEAN stack. MERN stands for MongoDB, Express, React, and Node. MongoDB is a nonSQL database and is the database for the stack, designed to be easy to work with. As the official MongoDB website states about the MERN stack, the database “was designed to store JSON data natively” (MongoDB). The MERN stack’s query language is built off JSON and JavaScript, making MongoDB work well with the components of this stack. Express is the web server framework for the stack and a package for Node. React is a JavaScript framework and the client-side framework for the stack. Node is the web server of the stack. The MERN stack works using a 3-tier architecture using only JavaScript and JSON, having all of its components work together for a fast and responsive web or mobile application.

MERN stack has many advantages to make it appealing for use. MERN is composed entirely of JavaScript and JSON, making the learning curve of this stack much lower compared to some other stacks that use multiple languages. Due to Node, MERN stack applications can be run locally, making development and testing very efficient. React allows the frontend portion of the application to run on a remote server, making it quick and reactive, not needing to wait for a response from the main server to update. The MERN components, especially MongoDB and Node, work well together, all resulting in an application that is fast for the user and quicker to develop in general. MERN is a common stack choice for web developers today, making it a great tool to learn. Along with some of us also having experience with this stack from a previous class to create a larger web application project, the MERN stack has many benefits going for it.

3.4.1.3 MEAN Stack

MEAN Stack is an alternative version of the MERN stack. The only difference between the two is the choice of client-side framework present. The MEAN stack uses Angular instead of React. Angular has been a very popular framework for quite a while, along with MEAN being one of the most popular stacks.

The main appeal for this stack is its past popularity. Similarly to React, Angular is a commonly used framework and one worth learning for web development. Unlike the previous stacks, however, members do not have experience with Angular.

3.4.1.4 Stack Comparison

Below is a table comparing the aforementioned stacks, what components the stack consists of and the benefits each stack offers.

Table 23. Stack Comparison Table

Stack	Components	Benefits
LAMP	Linux, Apache, MySQL, PHP	- Time-tested - Open-source components - Uses most common web server - Flexible - Previous experience
MERN	MongoDB, Express, React, Node	- Modern - Single language design - Local server development - Resume building skills - Previous experience
MEAN	MongoDB, Express, Angular, Node	- Modern - Single language design - Local server development - Resume building skills - Most commonly used stack

Both LAMP and MERN share the benefit of members having previous experience with them. LAMP has the benefits of its flexibility of each component being open source. LAMP is more time-tested than MERN and MEAN, which have only been around in recent years in comparison. LAMP uses Apache, the most common web server, making it very accessible to everyone. MERN and MEAN are more modern with components that have all been developed more recently and work extremely well together. MERN and MEAN have the benefit of local server development, something LAMP lacks. MERN and MEAN are a single language designed stack, using JavaScript, the most commonly used language for web development. This makes MERN and MEAN have a lower learning curve while also leaving the developer to learn JavaScript, a skill any web developer would need on their resume. MEAN has been the most used web development stack in recent years, however, recent studies have shown a larger interest in React over Angular, making MERN appear to be the future. The interest between the two is not too major, however, so both are solid options for both the project and for building a resume.

3.4.2 Programming Languages

This section contains the languages that were researched for the project. These programming languages are the options we have to create the software design for the embedded software and demo application. Each subsection gives a small history of the language, why it is used, and the positives and negatives for each of them.

3.4.2.1 Embedded Languages

This section contains the programming languages that were researched and considered for the embedded software. This portion of the document contains a section at the end which compares all the following languages and a table containing this information.

3.4.2.1.1 C

C is a computer programming language that is used in a wide variety of software, from simple applications to embedded system programming. C was devised out of Bell Labs by Dennis Ritchie and the early 1970s as a language for the Unix operating system in order to implement utilities. C was directly an improvement of the B language, which was slow in nature, but has become a staple language since.

C is a powerful language and generally one of the first taught languages as it provides a strong background for programming. This is due to the fact that most data structures, functions, etc are created from scratch, forcing the programmer to learn as they code. C is a language we are all familiar with, having used it at least once in a coding class during our major. While it is often one of the most difficult languages in comparison to others, the familiarity of the language is one of its strongest appeals. C was also used in our embedded classes, making it a prime choice for the embedded programming of this project.

3.4.2.1.2 C++

C++ is a superset of the C language, meant to bring object oriented programming into the language, a style normal C is unable to do, resulting in the original name of C with Classes. In *History of C++* found on cplusplus.com, it states that the history of C++ goes back to 1979, created by Bjarne Stroustrup during his work on his Ph.D. thesis. C++ was officially published and became commercial in 1985 (Albatross).

C++ is a commonly used language for embedded programming. Along with the additions it contains compared to C, C++ is a strong alternate option to choose for our embedded language of choice for the project. However, due to the popularity of C in our courses, experience with C++ is not as strong.

3.4.2.1.3 Python

Python is a commonly used, object-oriented language created by Guido van Rossum, initially released in 1991. Python Institute states that Guido van Rossum’s goals for Python were to be an open source language that would be easy and intuitive while being just as powerful as other languages. He wanted the code to be as understandable as that of plain English and suitable for everyday tasks (Python Institute). These goals were accomplished, resulting in Python to often be the go to language in many industries and a go to language for embedded programming.

Python is a simple and easy to understand language, used universally in the programming world. This makes it a must learn language for any programmer. Unfortunately, Python is not the language of choice for our courses and our experience

with it is minor at best. Yet it is a skill that should and must be learned, making it not only a great option for embedded programming but also a great opportunity to learn a valuable skill.

3.4.2.1.4 Arduino IDE

Arduino Integrated Development Environment (IDE) is a language composed of functions from C and C++. While Arduino is composed of C and C++, it is not those languages. Arduino has limitations compared to C and C++. One major limitation is that the majority of the standards libraries for C and C++ do not work in Arduino. Another limitation is that all files related to the code must be in the same folder. These limitations result in Arduino not being as powerful or as flexible as C and C++. However, it comes with the benefit of being designed specifically for the Arduino boards. The Arduino IDE simplifies the process of coding the board, making it great for beginners and results in it having a lower learning curve. For complicated embedded programs, Arduino IDE's limitations might make it difficult to code, but for simpler programs, Arduino IDE will be perfect. This language will be mainly considered if an Adruino board is chosen for the project.

3.4.2.1.5 Assembly

Assembly language is the root of all programming. Going back to the 1940s, assembly was used on the first modern computers by programmers. Due to the limits of the first computers, this was the only option at the time. Assembly is a difficult language, requiring much effort to produce the same results of even the simplest logic in modern languages. Assembly instead has become a background, the language that compilers of various languages compile to. While assembly is occasionally still used, modern compilers can create assembly code far more efficiently than most programmers, making assembly best left to the compilers.

Assembly might be the final language given to hardware to run, it is a very difficult language to work with. However, some courses did force us to write in assembly to make simple code, leaving us with experience of it. However, it would be a poor option to use, even for the simplest of tasks, making it not an ideal choice for our project.

3.4.2.1.6 Embedded Language Comparison

The following table is a comparison of the embedded language options. The following pros and cons were gathered from above research, common or person knowledge, the sections from Data Flair on each language:

Table 24. Embedded Language Comparison Table

Language	Pros	Cons
C	- Powerful and efficient - Lot of experience - Commonly used - Portable	- Lacks OOP - Difficult to debug - Use of pointers

C++	- Portable - Object-oriented - Commonly used - C compatible	- Lacks Garbage Collector - Use of pointers
Python	- Extensive Libraries - Simple - Object-oriented - Open-source - Portable	- Speed limitations - Lack of previous experience
Arduino IDE	- Designed for Arduino boards - Simpler than C/C++	- Lacks standard libraries
Assembly	- Some experience	- Difficult to work with - Complex to code simple logic - Outdated for programmers

From the above table and research, it is clear assembly as the main programming language is a last resort and should only be used if necessary, and as such, will not be a choice. Python offers many benefits. Not only is it one of, if not the, most popular languages today and a strong language to learn for the industry, it is also much more simple than that of C and C++, having a learning curve that could likely be accomplished in the time frame. C and C++ are very similar in nature. C++ mainly has the advantage over C in having more options to it, specifically containing the ability to use object-oriented programming. However, for the embedded programming that this project will require, this extra functionality will likely not be needed. C has the major benefit of our members having the most experience with it on both the hardware and software ends. While it might be more difficult to use, familiarity between the members could avoid a lot of headache during the process of building the project. The Arduino IDE is important to consider due to its specific use for Arudino boards. Arduino boards are very well known and commonly used, making it a likely choice to use as the microcontroller. If this ends up being the case, Arduino IDE should be considered as it was designed specifically for what the embedded language needs to fulfil.

3.4.2.2 Website Application Languages

Many languages are available to use for web development. The most common four are JavaScript, Python, Java, and C++. Unlike the above section, it is predetermined that JavaScript will be used for the web application for the project. This is due to the amount of experience we have with this language. We all have had a class that specifically used JavaScript based stacks to produce websites. Due to the limited time available to create both the project and the web application and the fact JavaScript is the leading language for web development, using it provides both saved time from learning new languages and valuable experience in an already commonly used language. The decision comes down to what JavaScript based languages are available to choose from.

3.4.2.2.1 React

React is an open-source, frontend, JavaScript library and part of the aforementioned MERN stack. React was developed and is maintained by Facebook and a community of independent programmers. It was released in 2013. React is used for single-page web applications or mobile applications. React is made up of small segments of code called components, which are used together to create complex and reactive applications. The use of components allow for creating the complex applications with manageable code due to the smaller sizes of implemented components. React uses DOM to render out the html of the code to be viewed by the user. React uses html style code with the use of JSX and JavaScript, allowing programmers that know the languages to have a much easier time working with the library.

React is a very common library used for front-end web and mobile development. Our group has experience with React from a previous class that was used specifically to make a web application, making it an ideal choice for our web application.

3.4.2.2.2 Angular

Angular is an open-source web application framework using TypeScript, a superset of JavaScript and is part of the aforementioned MEAN stack. Angular was developed by a team at Google and initially released in 2016. Angular.io states that Angular uses a root component to connect to a hierarchy of components with DOM. Each component is a class with logic and data and associates with an HTML template to display. These components have a decorator that adds metadata, which can be modified with the assistance of directives and binding markups (Angular).

Angular is a relatively new and sophisticated framework with many options available to the programmer. We do not have experience with this framework, however, it is a relevant framework in web development and thus is a powerful framework to learn.

3.4.2.2.3 React vs Angular

Table 25. React vs Angular Table

Name	Pros	Cons
React	- Previous experience - JavaScript based - Fast - Strong community	- Uses JSX - Constantly changing
Angular	- Larger functionality - Strong community	- Uses TypeScript - No previous experience - Steep learning curve - Performance hit on complex apps

React and Angular are two of the most commonly used frameworks for today's web development. Angular has many more options to it, but with the options comes more complexity and a steeper learning curve. Many of the extra options it contains are mandatory to use, forcing the programmer to learn them all. In comparison, React is much simpler, primarily using JavaScript and an html style code. JSX needs to be known and used by the programmer, but this does not vary far from normal html code, making it much easier to grasp. React applications are fast and reactive, leaving the user with a great experience. While we have previous experiences using React for web development, we do not have experience with Angular, meaning we would have to attempt to learn a framework with a steeper learning curve, taking up valuable time that could be allocated elsewhere.

3.4.2.3 Database Software

This section will contain the options we have to use as the database for our web application. The three main options are MongoDB, Firebase, and Amazon Web Services(AWS). The main requirements of the final selection of a database include being able to connect using NodeJS, simple to use, and having a decent amount of storage in the free tier of the database. The main usage of the database in the web app would be to store information on each of the HotBoxes such as if the box is open or not, the order number of the order that is being put in the box, and many other pieces of information. The following few sections will elaborate on the various pros and cons of each database software.

3.4.2.3.1 MongoDB

MongoDB is a global cloud database service for modern applications. It provides automated deployment, the ability to make post-deployment modifications to the database, and the ability to make clusters in order to test new ideas. It is also a post-relational database with JSON-like document data models and schemas, specifically made with horizontal scalability in mind. It is very easy to make changes on the fly to the database if necessary. MongoDB is most known for high performance and best-in-class security. Furthermore, MongoDB can be operated in the cloud, using MongoDB Atlas, or on-premise. The Atlas M0(free tier), which is what we would be using if we selected MongoDB as our database, has a storage limit of 512 MB and uses shared RAM along with a shared vCPU. The free tier provides more than enough for the scope of this web application. MongoDB uses its own query language, MQL, which was designed from the ground up for doing advanced queries and updates on the JSON-like document structures that are stored in each cluster. It uses the same JSON syntax as MongoDB documents, so it is very easy to assemble complex queries by hand or programmatically. MongoDB also supports ACID transactions and schema validations, so it is also very similar to its relational database predecessors. The limitations of MongoDB include high memory usage, in that it stores key names for each value pair, limited data size, in that the size of a document must not exceed 16 MB, and the lack of functionality of joins resulting in data redundancy.

3.4.2.3.2 Firebase

Firebase Realtime Database is a NoSQL, cloud-hosted database with data being synced across all clients in real time. Data is stored in the structure of JSON in the database as well. While Firebase is a database, it is also an application development platform. Firebase is explicitly designed for mobile application development, so its entire user interface and onboarding flows are built around that use case, including hosting, authentication, data-driven triggers, and analytics. As a result, it has more features that are purely for mobile app database management. The free tier that Firebase provides(Spark Plan) offers 1 GB of stored data, 50000 document reads, 20000 document writes, 20000 document deletes, all of which apart from the storage acts as a daily quota. It also provides a network egress of 10 GiB(Gibibyte) per month. This is more than enough for the current scope of this project. Should the project scale further later on, Firebase offers higher tiers that could potentially suit future needs.

3.4.2.3.3 Amazon Web Services

Amazon Web Services(AWS) is a cloud platform that provides a multitude of functionalities ranging from infrastructure technologies, such as compute, storage, and databases, to emerging technologies such as machine learning and artificial intelligence. As mentioned earlier, the main focus of AWS for our project would be to use it for its database services. AWS DynamoDB is the free tier of their database service. It is a database system that supports data structures and key-valued cloud services, allowing users the benefit of auto-scaling, in-memory caching, and backup and restore options for internet-scale applications. The advantages of DynamoDB include its performance and scalability, persistence of event stream data, and its storage of inconsistent schema items. In terms of DynamoDB’s performance and scalability, the main highlight is that it gives the user the ability to auto-scale by automatically tracking how close usage is to the upper bounds of the limits. The second advantage, the persistence of event stream data, refers to DynamoDB streams allowing developers to receive and update item-level data before and after changes in that data. This is because the streams provide a time-ordered sequence of changes made to data within the last 24 hours, subsequently allowing the developers to make a full-text search data store such as Elasticsearch. The third advantage has the largest impact on our project, the storage of inconsistent schema items. DynamoDB follows a NoSQL data model, so it handles less structured data more efficiently than a relational data model. As a result, it handles higher volumes of queries and subsequently high performance queries for item storage in inconsistent schemas. This would improve with a larger scale for our project since most consumers will have multiple “HotBoxes” along with NoSQL being a better fit for our project since that is what most of the members have experience with.

3.4.2.3.4 Database Comparison

The following table shows the advantages and disadvantages of each of the databases that are available for us to use for the project.

Table 26. Database Comparison Table

Database Name	Advantages	Disadvantages
MongoDB	- Schema-less so any data can be stored. - High Speed, very easy to access documents via indexing. - Is horizontally scalable, so it is easy to distribute data to multiple machines.	- Does not support joins, must be coded in manually which can affect performance. - Has high memory usage as it stores key names for each value pair, causing data redundancy. - Limited data size, documents cannot exceed 16 MB.
Firebase	- Concise documentation, making it accessible to everyone. - Provides many services beyond just database services. - Has an accessible UI and provides ease of integration.	- Limited querying capabilities, very difficult to make complex queries. - Cannot implement relations between data items. - Was designed with mobile app development in mind, so not in line with current scope of project.
AWS DynamoDB	- Provides access to control rules without bottlenecks other people's workflow. - Provides automatic data management in the form of backups. - Easily links to AWS hosting since they are part of the same product.	- Indexing will be more expensive since it requires additional read and write capacity to be provided. - Can be difficult to analyze data in the database since it is a key-value database so general data structures for analytics may not work.

3.4.3 Software Selection Summary

This section of the document contains a discussion of what software we decided to implement into our project. The section contains each selection for a specific software portion of the project that needs to be implemented and why we decided to use it over the various other options we had to choose from.

3.4.3.1 Website Stack

This section will contain all the development technologies we decided to implement for our website demo for our project. All these technologies were explored and compared in the Software Research section of the document. A stack contains four components that will be discussed below. These components are the database, the front-end framework, the back-end framework, and the web server. While there are commonly used stacks,

as aforementioned in the stacks portion of that research section, the following is a stack of components we decided would best implement our application.

The following table displays our chosen stack with its components. The sections following this table will explain why each component was chosen.

Table 27. Website Stack Table

Component Type	Selected Technology
Database	MongoDB
Front-end Framework	React
Back-end Framework	Express/NodeJS
Web Server	Heroku

3.4.3.1.1 Database

The chosen database for our demo site is MongoDB, specifically MongoDB Atlas. The reasons for selecting MongoDB Atlas as the database are as follows:

- Previous experience
- Industry standard for Database As A Service
- Single Object Structure is clear

The first reason for selecting MongoDB was the fact that all members of the group have previous experience. We all created our own website application in the last year that had similar software requirements. Obviously, the main challenge with this project on the software side will be linking the embedded software to the database, so it is beneficial for us to choose a database that we are already familiar with. Taking a new database that we have never used would force us to learn how to use it, therefore the learning curve could prove to be a problem.

The second reason for selecting MongoDB was that it is an industry standard for database as a service. MongoDB Atlas was specifically designed for developers to not worry about operational tasks such as patching, backups, and configuration. Many companies use mongoDB, albeit higher tiers, as their database, so having a good understanding of it by using it here would be beneficial.

The third reason for using MongoDB was because it had a clear single object structure. The single object structure is beneficial because it makes it very easy to search the collection for specific details and display it for whatever use is desired. However, this can be both an advantage and a disadvantage, since this can result in a lack of organization in the specific collection thereby causing potential confusion when searching the database.

There are several reasons we did not opt to use our other options, which were Firebase or AWS DynamoDB. One of the reasons for not using Firebase was that it is mainly geared towards mobile application development. While it does provide more of a full package in terms of authentication and hosting, it was specifically built for mobile development which is not in line with the vision for the web app we are trying to create. As for the reason to not use DynamoDB, while it does fit the vision for the project, the entire project, which encompasses the web app, the hardware, and other parts, we are very limited in terms of time. From the time we actually start working on the project, we have a span of 2 and a half months to build the entire project, so it is much more convenient to stick with what is known in the form of MongoDB.

For the reasons above, the entire group has decided to go with MongoDB for the database for our project.

3.4.3.1.2 Front-end Framework

The chosen front-end framework for our demo site is the JavaScript library React. Our group decided to to with React because:

- Previous experience
- Easy testing development
- Reactive application
- Industry standard

Our first and main reason for choosing React was our previous experience. We used this JavaScript library in a previous class to create a website application. This application was on a similar scale to the demo application we will have for this project. Thus, the experience we have with React fits perfectly with what we will need to use it for in this project.

Another strong benefit React had going for it was its easy testing development. React allows for the developer to fully test the front-end portion of the application on their local system. This saves time from needing to upload the code to the server every time it needs to be tested, specifically what needs to be done using the components of the LAMP stack.

React is used for its reactive applications. While this might not be very important with the size our application will be, it was considered for the fact of where it will be used. In the chance that our project ends up needing to be demoed on campus, the campus WiFi could make it difficult to run the application, knowing how slow the WiFi can be there. React will allow our application to run quickly and effectively, only needing the initial loading of the site to work. This may save time and stress during the demo, and that is valuable.

Another reason for choosing React is the fact it is an industry standard. React is the leading front-end framework used for web development today, making it an important skill to know and have for the resume of any software based designer or engineer.

Reasons we did not go with the other option, in this case, Angular, is due to its large learning curve. Being our project needs both a web application and embedded code, it is important to ensure we have time to complete both. Having to spend time learning a framework with a large learning curve, while also an industry standard, is valuable time we might require elsewhere.

For the reasons presented above, our group decided to use React for the front-end framework of our website application stack.

3.4.3.1.3 Back-end Framework

The chosen backend framework for our demo site is NodeJS/Express. The backend framework of our website will be the main driver that creates our models and routes for each data structure. This involves the structure of how we store information on each HotBox, such as the state of the box, the order number that must be sent to the box, and a unique identifier that links the web app and the box to the database. Node JS is an industry standard that offers a variety of packages that can satisfy any need.

The main reasons we opted to use NodeJS are as follows:

- Previous Experience
- Flexibility
- Industry Standard

The first reason we opted to use NodeJS is because all members of the group have prior experience with it. As mentioned in previous sections, all members of the group have used a MERN stack before, which uses NodeJS. The previous project was of a similar scale, so translating that experience from then to now should be relatively simple. The benefit of having that previous experience is that we do not have to spend time learning how to set up the framework and looking for specific packages that we will need. Since the second part of the course will be during the 12 week summer semester, time is of the essence. So removing the learning curve period is extremely valuable to us as a group.

The second reason we opted to use NodeJS as our back-end framework is that it is extremely flexible. Using NodeJS as a backend provides all the benefits of full stack JavaScript development. It has an extremely large number of free tools, provides better speed and performance, and provides better efficiency and overall developer productivity. The packages that NodeJS provides can fit almost all situations that any developer would need, whether that is requiring connecting to MongoDB, using Express, or even connecting to MySQL. Along with having a large range of packages, there is a lot of documentation detailing how to to install and use these packages. Having this documentation only serves to help and improve our workflow for this project.

The third reason we opted to use NodeJS as our back-end framework is that it is an industry standard. Even though it was introduced back in 2009, it is still an extremely popular tool that people opt to use over other options. NodeJS has made itself available to many hosting service providers over the years, so it has remained relevant for all of those years. It also offers the ability to cache data, meaning it stores data for some future requests, making the application of NodeJS even faster. It is also an open source technology, so it is constantly evolving in order to fit any need.

For the reasons stated above, our group has decided to go with NodeJS as our back-end framework.

3.4.3.1.4 Web Server/Hosting

In terms of the web server/hosting, we are primarily looking for 3 things. The first being github integration, the second being easy to set up, and the third being a free service. For these reasons we have decided to go with Heroku Web Hosting. Heroku is a platform as a service(PaaS) that enables developers to build, run, and operate applications entirely in the cloud. It satisfies all the requirements that we specified and more. Heroku has GitHub integration, it is easy to set up, has a free service, and we have all used it before in a previous project.

Github integration will be extremely important since this project has multiple members, meaning that the project files will be constantly updated, especially between the frontend programming and the backend programming. Heroku uses Git, which is a popular code version control system that is integrated with Github, to deploy code from github to Heroku and back. Using Git will be very important to our project's success since it will keep everything organized and will allow other members to constantly see what changes have been made, or even what specific requirements have been fulfilled which prevents two people from working on the same task unnecessarily. The other side of this entails actually uploading things to a github repository that is specifically made for this project. Having a project github repo will be important to keep all of our code and other resources collated to stay organized. This, combined with the easy deployment of code to Heroku, makes it easy to select Heroku for hosting our web app.

A simple set up will also be vital to our project's success, specifically since the rest of the project is so complex. As we are on a tight schedule, since the entire project should be completed by the end of July, it is important that we don't get caught up on struggling to set up a web server. Heroku is very easy to use and has a lot of documentation to help anyone that is struggling to work with it. A brief overview of the process is described as follows. First, we need to set up the Heroku CLI which is the command line interface that helps manage, scale, and run applications. Second, we need to create a heroku app from the command line, using the "heroku create" command to create an app on Heroku. Finally, we deploy our code using "git push heroku main" which pushes our code to the aforementioned heroku app. The entire process takes around 10 minutes to get a basic application set up, and any further updates to our application can

be added using Git and updating the heroku web app. This is an extremely short process, which is exactly what we specified in our requirements for a web host/server.

Having a free service is also very important. Our web app is not very complicated and it does not have any requirements that would result in the need for us going past the free plan most services offer. Heroku's free plan includes having a monthly run time of 550 hours, 512 megabytes of RAM, and many add ons that can be used for almost anything. Heroku is primarily built for students, which is perfect since this Senior Design project is for students. Additionally, all members of the group have used Heroku before in a previous project, so we all have experience using it and know that it is free to use.

For the reasons specified above, we have decided to continue forward with Heroku for our web hosting/server.

3.4.3.2 Embedded Software Language

There are a vast variety of languages to use for the embedded software for the project. However, for our project, we believe the best option will be the Arduino IDE. This IDE, using a C based language, is built specifically for the Arduino boards. Having decided to use an Arduino board, using the IDE designed for them is a perfect fit.

The limitations of the IDE should be noted. At the time of writing the documentation, we cannot know exactly what the code will need to contain and whether the limitations will get in the way of producing the algorithms that will be needed. As such, there is a chance that, should the Arduino IDE have too many limitations to achieve our goals, we will swap to C or C++ as needed. This switch will be possible due to the fact that the IDE uses C/C++ at a base level. Most code that works in the Arduino IDE can be put into a C or C++ file and run without any adjustments. This will make the swap mostly painless and greatly increase the options we have with the added libraries that the Arduino IDE does not support. But since the IDE is designed specifically for the Arduino boards and made simple to implement for their boards, this will be our first option in order to save time in the development process.

4. Related Standards and Realistic Design Constraints

This section of the document will contain all the standards and design constraints we will have to deal with during the design and development of our project. The related standards that apply to our project are things we must abide by to ensure our project could be marketable and manufacturable as a design, even if we are not planning to do that. The realistic design constraints will be problems and constraints caused by some standards that we must address to ensure our product not only functions properly but is also safe to use and not a hazard to people or the environment.

4.1 Software Standards

This section of the document will cover the form and interpretation of the code written for this project. Javascript is a programming language that conforms to the ECMAScript specification and uses curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. ECMAScript is commonly used for client-side scripting on the web, and is being increasingly used for writing code using Node.js.

ECMAScript Javascript supports a structure of programming that is very similar to C, in that they are imperative and structured. Both use keywords such as “let”, “const”, and “var” for both block scoping and function scoping. It also has a very similar control-style, with while, for, do/while, if/else, and switch statements. Functions can accept and return any type, including “undefined” if no argument is provided. ECMAScript is also weakly typed, which means that it has loose typing rules and can result in unpredictable outputs. It can also perform implicit type conversion during runtime, but this has drawn criticism from many developers since it can introduce many unexpected issues. It is also dynamically typed, meaning that a type is associated with a value instead of an expression. A language is dynamically typed if the type of a variable is checked while the application is running.

In terms of actual coding standard, there are not very many specifications that a developer must follow. Unless this code must be refactored to match a different set of standards, we will be following a generic set of standards. For spacing, we will be aiming to space out all code, so indentation with only tabs, trying to not leave whitespaces at the end of every line, and using braces for if/else/for/while/try statements. We will also not be leaning on the automatic semicolon insertion functionality offered by Javascript since it can cause several issues in output, so we will be making sure to include all statement terminators wherever necessary. By following all of these standards, we hope to keep our code organized and readable for others.

4.2 Realistic Design Constraints

This section will contain design constraints that we will have to abide by or deal with in the research and building process of our project. There are a wide variety of issues that must be addressed or accounted for in making an acceptable and safe functioning product. These constraints include economic, time, environmental, social, political, ethical, health, safety, manufacturability, sustainability, and Covid 19 constraints. Each of these constraints will be addressed in the following subsections.

4.2.1 Economic and Time constraints

The first of the constraints of this project are economic and time constraints. These constraints come from the fact that, as students doing this project for a class course, we have a limited amount of time in order to get everything done, from the documentation to the product itself. Everything for this project comes out of our pockets so we also must keep the product affordable.

Our time constraint is separated into two semesters. The first semester is a time constraint on the documentation. This included the time available to research parts for our product and create achievable requirement specifications. This also results in a limited amount of time to produce the documentation for the entire project. Parts will also need to be acquired in time for the building process, which should be done during this time period.

The second semester is a time constraint on the actual building process of the product. We have a limited amount of time to build a prototype, test the prototype, and fix and adjust it as necessary. We also have the added time constraint of taking this semester over summer, giving us less total time to get our product built as compared to if we took it in fall. This time constraint will be one of the major constraints of our group and must be accounted for and prioritized to ensure the project gets finished.
The economic constraint of our project is the monetary constraints we have. The project must be affordable to develop and build. The overall product should also remain at an affordable cost in order to fit the needs we are wanting our project to fit. Added to this constraint is the fact our project is not sponsored. As such, all expenses of the project come out of our pockets. This results in a significantly smaller budget to work with then we might otherwise have had.

The main aspects of the project that will be affected by this constraint are part selection. Some parts that could have been ideal for our project might result in having too large an expense to go with, resulting in aspects of the project being more difficult, if not impossible, to achieve. This could affect our stretch goals if specific parts are needed to complete them and end up being too expensive. This project should not stress our everyday livelihood due to expenses, so we must be sure to only get parts that are necessary for the project and be careful not to have issues that could result in us needing to replace parts later on.

The time and economic constraints will play a vital role in our success of the project. Without proper time management, the time constraint would result in an incomplete project. Without proper monetary management, the project could become too expensive either for us to afford building or too expensive to achieve the goals we set out to achieve with our project. These constraints can make or break our project, and thus, will be important to keep track of throughout the project.

4.2.2 Environmental, Social, and Political constraints

Environmental, Social, and Political constraints come around if a product is going to be manufactured and deployed. While we currently have no plans for this, this subsection will cover the constraints would the above happen.

Environmental constraints involve the effect of the product and the environment on each other. Our product resembles a low heat oven, as its use is to keep food warm, not to cook the food. Although the product will not reach the same temps of an oven, it would follow many of the environmental constraints of an oven. As such, the HotBox must not cause damage to the environment. Potential issues that could occur would be sparks from the electronics or wires and failure to hold heat inside the box, potentially resulting in damage to the surrounding environment by fire. The product should not pollute the surrounding air with any dangerous chemicals. The product should not produce dangerous levels of radiation that could harm anything in its surrounding. This product must be properly designed so that it will not cause any damage to its surrounding.

Social constraints involve the effect of the product on people. An individual must be able to easily use the product without fear of damaging the product or getting hurt by the product. This product deals with electronics and heat. An individual should be safe from any risk of electrocution. The product should have ample safety in keeping the individual safe from any potential burns from the produced heat of the product. Warnings of misuse should be contained on the product or in the documentation manual so that the individual using the product knows what to avoid for their safety. Proper use of the product must never risk harm to the individual.

Political constraints involve the effect of the government on the product. If the product were to be manufactured and deployed, the government could place constraints on the product. These constraints could range from tariffs to standards specific to the product. At the time of writing the documentation, there are no plans of manufacturing and publicly deploying the product, so there are no current political constraints on this product.

While some of these constraints may never apply to the product, it is important to keep them in mind. Even if the product is to be kept private, environmental and social constraints are still important to keep in mind in development. Regardless of where the product is used, these constraints are important for the safety of anyone who could encounter the product and the environment itself.

4.2.3 Ethical, Health, and Safety constraints

Ethical, Health, and Safety constraints exist from standards on manufacturing, product design, and the public view on the production of products. These standards were discussed above in the standards section.

Ethical constraints are constraints placed by the public view and some standards. These constraints include how parts are procured and how the product is manufactured. The product is currently only manufactured by us privately, resulting in no constraints on the actual manufacture process. The parts needed for the product should be acquired via ethical means. This includes ordering parts from accepted manufacturers, not from manufacturers with questionable ethics. These constraints generally will not apply to our product unless it were to be manufactured and deployed publicly.

Health and Safety constraints for the product are vast. Some previously mentioned constraints stem from these constraints. The product must be built from safe materials. The product should not produce any gas or other substance that could cause harm to an individual using the product. The product must have safe and proper wiring to ensure the electronics of the product will not cause sparks or electrocute an individual using the product. The product should maintain the heat inside safely, not letting it leak into the surrounding or transfer into the outside material that could risk burning an individual. The documentation or manual of the product should make clear any specific safety concerns specific individuals could encounter with the product. The documentation should provide how to safely use the product to avoid harm. The documentation should provide warnings of what is improper use of the product and warnings of potential injury should the product not be used properly. The product must not harm an individual that properly uses the product according to the documentation.

There are some specific constraints that need to be considered in our design to ensure the project is safe to use. The above will be expanded into specific constraints that are as follows:

4.2.3.1 Overheating

The function of our project is to keep food warm after being cooked. The heat that the HotBox produces must only be enough to keep food warm. The project will also be designed around heating food, not cooking food. Should the project produce heat greater than the project was designed for, it could result in the overcooking or burning of the food and damage to the internal portions of the project, potentially damaging the functionality and/or safety of the product.

4.2.3.2 Fire

The HotBox will make use of electricity to power the electronics and heat to warm the food. Both come with the risk of causing fires should they be improperly managed. Electronics must be properly insulated and use of safe wires should be used to transfer the electricity to the electronics. Improper consideration of this could result in sparks that could risk producing fire. The heat that will be produced by the heating pad inside

the box must be accounted for and limited. If the heat inside the project is not insulated from the outside or the electronics are not insulated from the heat, there is a risk fire could be produced.

These constraints are important to ensure the safety of any individual that uses the product. Should these constraints not be followed, individuals could be at risk using the product. Should an individual be harmed by the product, the designers or manufacturers will be held accountable. The product should be produced ethically to have favorable views from the public. The product should be designed according to the standards applicable to the product to ensure it is safe to use and does not risk the health of an individual using the product.

4.2.4 Manufacturability and Sustainability constraints

Manufacturability and sustainability constraints will impact our project in the development process. Manufacturability constraints are the manufacturability limitations. These limitations restrict the parts and components that can be used in our project's design to parts and components that can be manufactured. For us, this will limit the parts we can use to those that are actively manufactured and open to purchase. This also restricts our electronics to be under manufacturing standards so they can be sent and manufactured after the design process. Being we will be building our project ourselves, we will be limited to what is available to us to use at the campus, development tools we may already own, or tools we can afford to purchase for the project.

Sustainability constraints are constraints that will affect how long our project can remain functional and safe. Our project is expected to handle daily use. The project must maintain a constant heat that could affect the surrounding materials, parts, and electronics. These factors need to be considered in our design and development of the project. Without it, the project could not function as expected or not function as expected for an extended period of time. Improper consideration of these limitations could also result in hazards our project should not have. Importantly, the design needs to consider how the heat could affect the electronics of the project. Either the electronics need to be designed to withstand the heat the parts need to endure or the design needs to account for how to keep the heat away or minimized so the electronics will not encounter this issue.

Manufacturability and sustainability requirements may either compliment each other or oppose each other. Manufactured parts that improve sustainability are considered positive correspondence. However, often the opposite is the case and manufacturing and sustainability oppose each other. Ideally we will find manufacturing that improves the sustainability of the project but this might be unavoidable.

4.2.5 Covid 19 constraints

In January, 2020, the US reported its first case of the disease Coronavirus Disease 2019, stemming from the virus SARS-CoV-2. More commonly known as Covid 19, this

disease would rapidly grow in cases and officially be declared a pandemic in March 2020. This would lead to many the closing of businesses, public spaces, and importantly for us, college campuses. Covid 19 has brought about many challenges to overcome for this project, resulting in constraints not usually present for the Senior Design project. Covid 19 has resulted in unprecedented times for everyone. Every day presents new challenges and constraints and each day is more difficult than it once was. The following are specific constraints that Covid 19 has caused for us, constraints that may not usually have been present or are worse than the constraints usually would have been.

One constraint Covid 19 has brought about was the weakening of the economy. Over this year time, the economy has greatly fallen and inflation has increased. This brings a constraint of budget. Not only is the usual budget constraints aforementioned present, Covid 19 has caused the price of many products to increase, effectively creating a greater constraint for the budget, making our available budget effectively be less than it could be. Covid 19 has also made it difficult to get or keep a job, resulting in the aforementioned budget potentially being smaller than it would have been had this not been going on.

Another constraint Covid 19 has caused, related to the economy, is shipping and product availability. Due to social distancing standards and closing of businesses, the production of products available has greatly decreased. While this situation has improved over the year since it began, it is still a struggle going on and a constraint that we must deal with. Many businesses have been overwhelmed during this time. While availability has decreased, the need has not. This has resulted in shipping also being overwhelmed. Many products are not always available, and when these products are, shipping may take longer than usual due to the demand. This could be a minor or major constraint, depending on whether our parts are affected by unavailability or just potentially longer shipping times.

Potentially the greatest constraint we will encounter is social distancing. While Florida has had far less aggressive constraints on this, it is still nonetheless a risk for us to meet up, and thus is avoided. In order for us to meet, we have to be careful to have proper social distance prior to meeting, affecting other aspects of our lives. This will specifically make the building of the project more difficult. The vaccine may help with this, but it is unsure when it will be available for everyone and thus, it is unknown how much this could constrain our process of building and testing the project.

5. Project Hardware and Software Design Details

This section contains more specific details on how each portion of the project will be designed or built. This section contains both the hardware designs and the software designs.

5.1 Housing Design

In this section we will mainly be going over the housing design of the box and how it will be constructed. The material of the housing of the box will be plywood due to the inexpensiveness and the ease of manipulating and editing the wood to meet our project needs. The wood we purchased also had excellent thermal retention so that would aid with the insulation of the box. The dimensions of the box heavily rely on the dimensions of the heating pad we decided to use. Our heating pad is a square base coming in at 310mm x 310mm (12.20 x 12.20 inches) and 3mm (.12 inches) thick. This is perfect to fit the majority of different food sizes. The heating pad will sit on the bottom of the inside of the box. The bottom of the heating pad features insulated cotton lined with aluminum that helps with thermal conductivity inside the box. We want the height of the box to have enough clearance for the customer to pick their food from the box, however not too much clearance in order to heat adequately through the box. The shorter the height of the box, the less time, energy and space needed to heat the overall interior of the box, making it more efficient. The box is made up of three main compartments; the interior heating space, the upper shelf for other electrical hardware components, and the power system storage unit.

The interior of the box will be lined with a semi thick aluminum lining in order to improve heat dissipation and heat retention inside the box. Inside the interior portion, the only components present are the heating plate, used for the main heating of the box, and the LM75A temperature sensor that is used to regulate the inside temperature of the interior of the box. In order to keep the contents of the interior safe and secure, we installed a door as well. The door features two regular stainless steel hinges that open the door outward toward the customer for them to take their food. The solenoid lock is also on the door with the latch present on the box itself in order to catch the lock and secure the contents inside. The door is also insulated as well to prevent any heat from escaping.

Right above the interior of the box, we have the upper slide out shelf portion that holds the rest of the main hardware components. The upper portion is vented in order to prevent overheating and is properly sealed off to prevent heat dissipation from the interior portion of the box. On the shelf, the main components present will be the 4 channel relay, PCB microcontroller, the LCD display, and the barcode scanner. For the barcode scanner and the LCD display, we placed an acrylic transparent plexiglass on

the front of the upper portion in order for customers to be able to scan in QR codes/barcodes and view their order information for validation on the display.
The back of the box features a small compartment that houses the overall wiring and power system of the project. The AC-DC transformer and the necessary DC-DC converters to operate the hardware are also present inside. This compartment is also vented properly in order to prevent overheating which could damage our system. The excess wiring that we had was also kept in this compartment for simplicity and ease.

5.2 Part Schematics

In this section we will cover all the different part and component schematics present on the PCB and go over what went into the design of each individual part. Although the overall schematic is not final and is changing, this will give us a main idea of what the schematic will look like for each part.

5.2.1 Barcode Scanner

As we mentioned in the previous section about the barcode scanner, we will only be using four of the necessary pins on the breakout board to fully operate the scanner. The barcode scanner will be communicating with the microcontroller using UART. These pins are listed below:

- VCC (5V)
- GND
- RXD (Input receive)
- TXD (Output transmit)

When implementing the scanner into our designs, we came across a problem that needed to be solved. One was that the ATmega328 is developed to operate at a 5V logic on the TX and RX pins used for communication while the RX and TX pins on the scanner operate at a 3.3V logic level. Since we have the ESP32 WiFi module present, which operates at a 3.3V logic level, we are able to use the barcode scanner without having to level shift the connections. And we use the UART communication interface to communicate between the wifi module and the scanner.

Below is the PCB schematic for the barcode scanner. Due to the inability of placing the physical barcode scanner on the pcb, we decided to use a 4 pin header and have wires connect to them for ease of use.

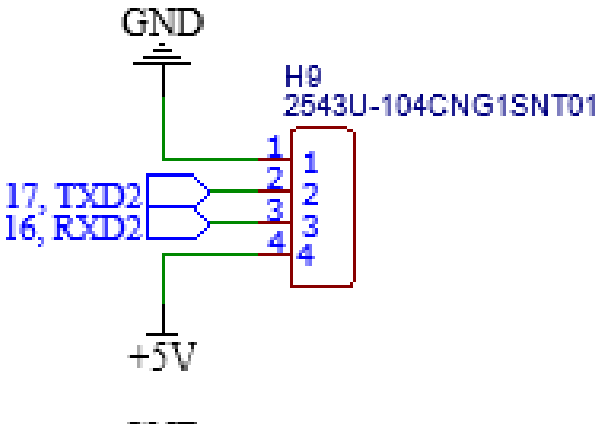


Figure 26. Barcode Scanner Schematic

5.2.2 4-Channel 5V Relay

For the 4-Channel 5V Relay, we will have to implement a header onto the pcb just like we did for the barcode scanner. The 4-channel relay has 6 pins that will be connected to the pcb, four of which are the input signal pins for communication to the relay, pins listed below:

1. GND
2. IN1
3. IN2
4. IN3
5. IN4
6. VCC 5V

Below is the PCB schematic for the 4-channel Relay, as you can see there are two unused input pins just in case for any added parts or features needed in the project. In the schematic as well we have the electronic lock signal as well as the heating pad signal connected to the relay.

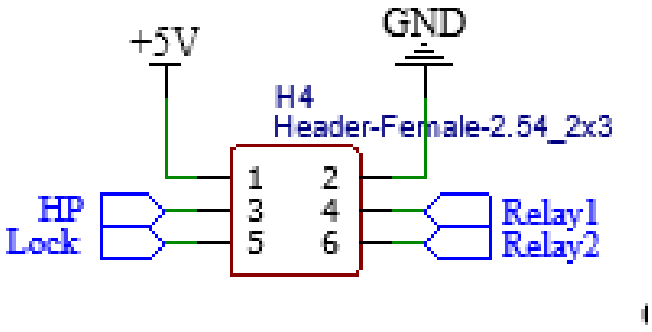


Figure 27. 4-channel 5V Relay Schematic

5.2.3 Wireless Communication

For communication to other devices and servers, the ESP32 will be responsible. It's key for us to get this executed as best as possible because this is one of the key features of the box. For the supply voltage we have a choice between 3.3V and 5V for the ESP32. The barcode scanner is also wired together for communication through UART as well. The pins on the ESP32 that we will be using are listed below:

1. SCL
2. SDA
3. GND
4. TXD0
5. RXD0
6. TXD2
7. RXD2
8. 5V

Below is the schematic for the ESP32 in our project

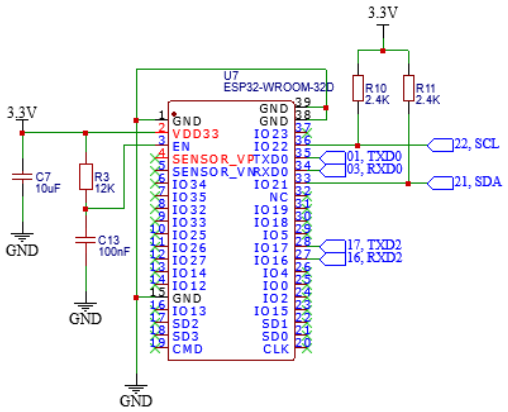


Figure 28. ESP32 Schematic

The ESP32 will be run together with the ATMega2560 and will communicate with each other using either I2C or UART. Just like the Barcode scanner, the ESP32's I/O pins operate at a 3.3V logic level meaning we will have to use a logic level converter for this to fully function. The figure below shows the schematic of the logic level conversion:

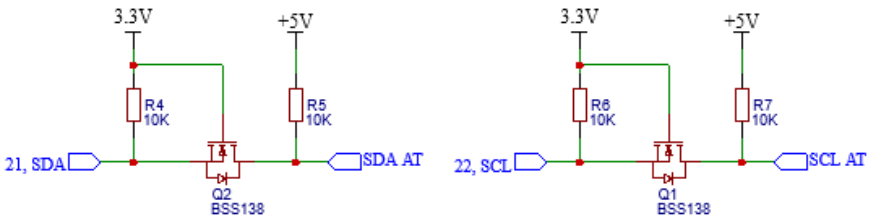


Figure 29. MOSFET Logic Level Conversion Schematic

5.2.5 Temperature Sensor

To regulate the temperatures inside the box in order to control the heating, we must have a temperature sensor. The temperature sensor will be housed inside the heating portion of the box and since the ATMega will not be housed inside with the temperature sensor, we will require a header for it on the PCB design and wire it. The temperature sensor will communicate to the ATMega via I2C and the pins used will be listed below:

- VCC
- GND
- SDA
- SCL

Below is the temperature sensor schematic, as well as a schematic for the pull up resistors required for the I2C communication between the sensor and the ATMega to fully function properly.

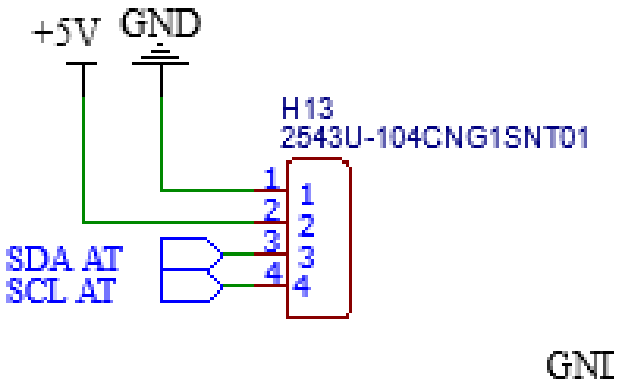


Figure 30. Temperature Sensor Schematic

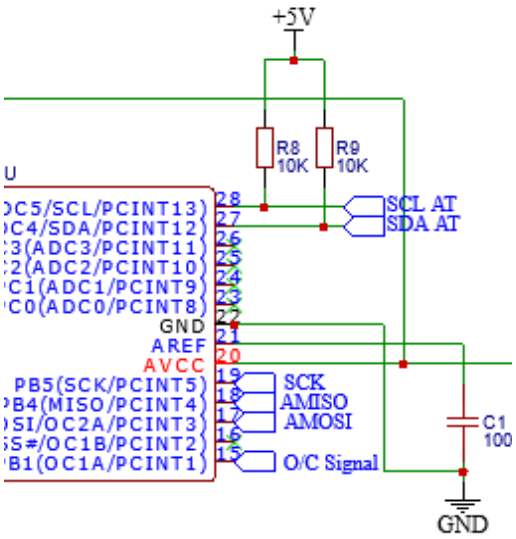


Figure 31. Pull-Up Resistors

5.2.6 LCD Display

In order for customers to confirm their order is the correct order, an LCD display will be present on the box where users can view their order information. The display will communicate with the ATMega to pull different orders from the database through the wi-fi module. The display will communicate to the ATMega via I2C similarly to the temperature sensor meaning only 4 pins are required for use. Below lists the pins used as well as the schematic for the display.

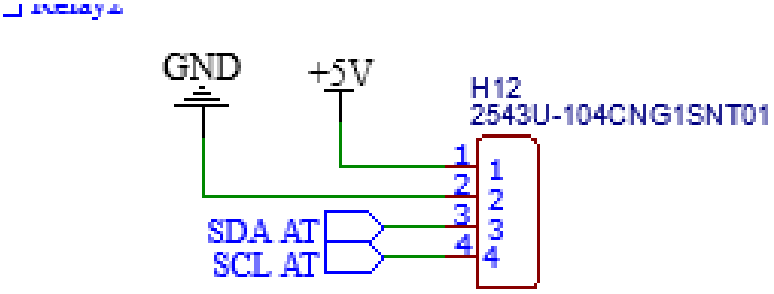


Figure 35. LCD Display Schematic

5.2.8 External Clock

The ATMega2560 clock can be sourced from a few options including External Clock, Crystal Oscillator, Low-frequency Crystal Oscillator, or a Calibrated RC Oscillator. The ATMega is equipped with its own internal RC oscillator at 8MHz; however a more accurate clock can be connected through the ATMega clock pins, XTAL1 and XTAL2 as long as the frequency of the oscillator is between 0.4 - 16 MHz. Below is a table of the pins with their descriptions.

Table 28. XTAL pin Descriptions

Pin	Description
XTAL1	Input pin to the inverting oscillator amplifier as well as the input to the clock operating circuit.
XTAL2	Output pin from the inverting oscillator amplifier.

We decided to implement a 16MHz Full Swing Crystal Oscillator into our project and the recommended range for the capacitors needed are between 12-22pF. We decided to use 20pF for both the capacitors. The schematic for the external clock configuration is below.

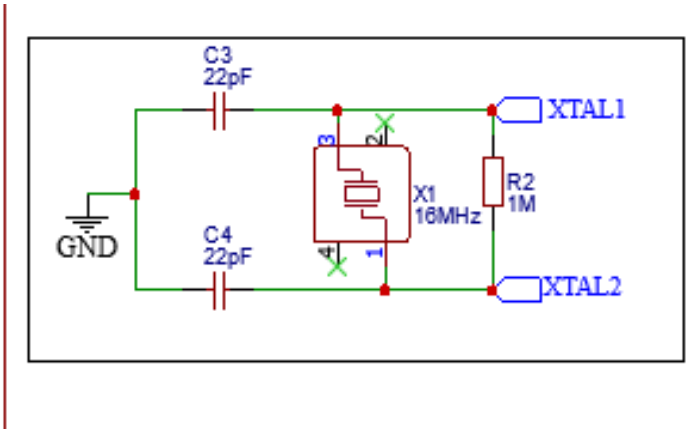


Figure 33. External Clock Configuration Schematic

5.2.9 Reset Button

Present on the ATMega2560 is also an optional reset input pin. In order to activate the pin, a low level for longer than the minimum pulse length (2.5µs) with or without the clock signal operating. We decided to include a button onto our design for resetting the PCB just in case. Below is the schematic for the configuration for the Reset Button.

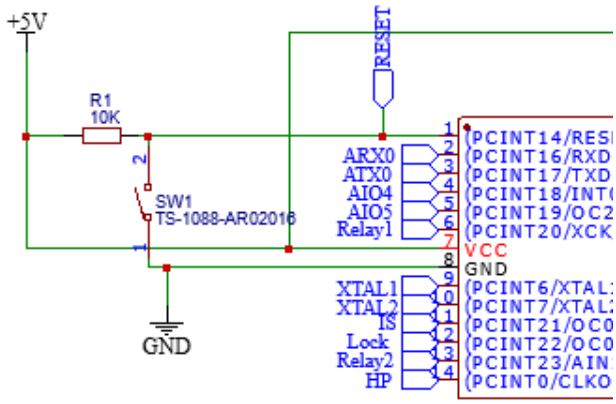


Figure 34. Reset Button Schematic

5.2.9 PCB Microcontroller

The microcontroller is the main unit communicating to all the hardware components and essentially the component controlling the project. Initially, when choosing our microcontroller, we were looking for a controller that could not only support our components, but other components in the future. We decided to go with the Atmega2560 at first, which featured 4-UART, 1-I2C and 5-SPI peripherals for communication. This was perfect for us because a lot of the components we decided on using have open source libraries which are supported with the Atmega2560. Also it featured 4-UART interfaces which bodes well for our system because UART is a key communication method with various devices such as the wifi module and barcode scanner. However, with all our necessary components installed and connected, we realized we have too much space and decided to minimize the project by going with the Atmega328 which has 28 pins instead of 54 pins on the Atmega2506. Atmega328 still features the same communication protocols, so this did not cause any problems when switching controllers.

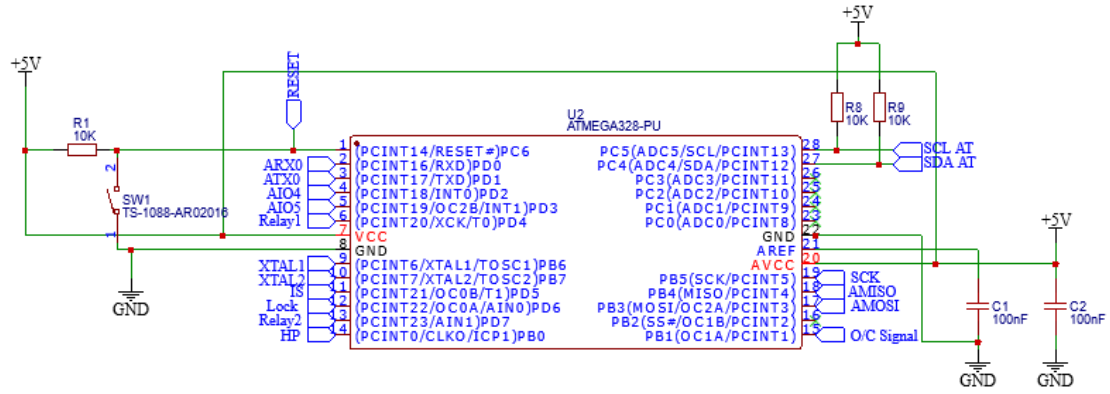


Figure 35. ATMega328 Schematic

5.3 Final PCB Design Schematic

After going through each part's schematic, below showcases the final PCB Design Schematic for our project.

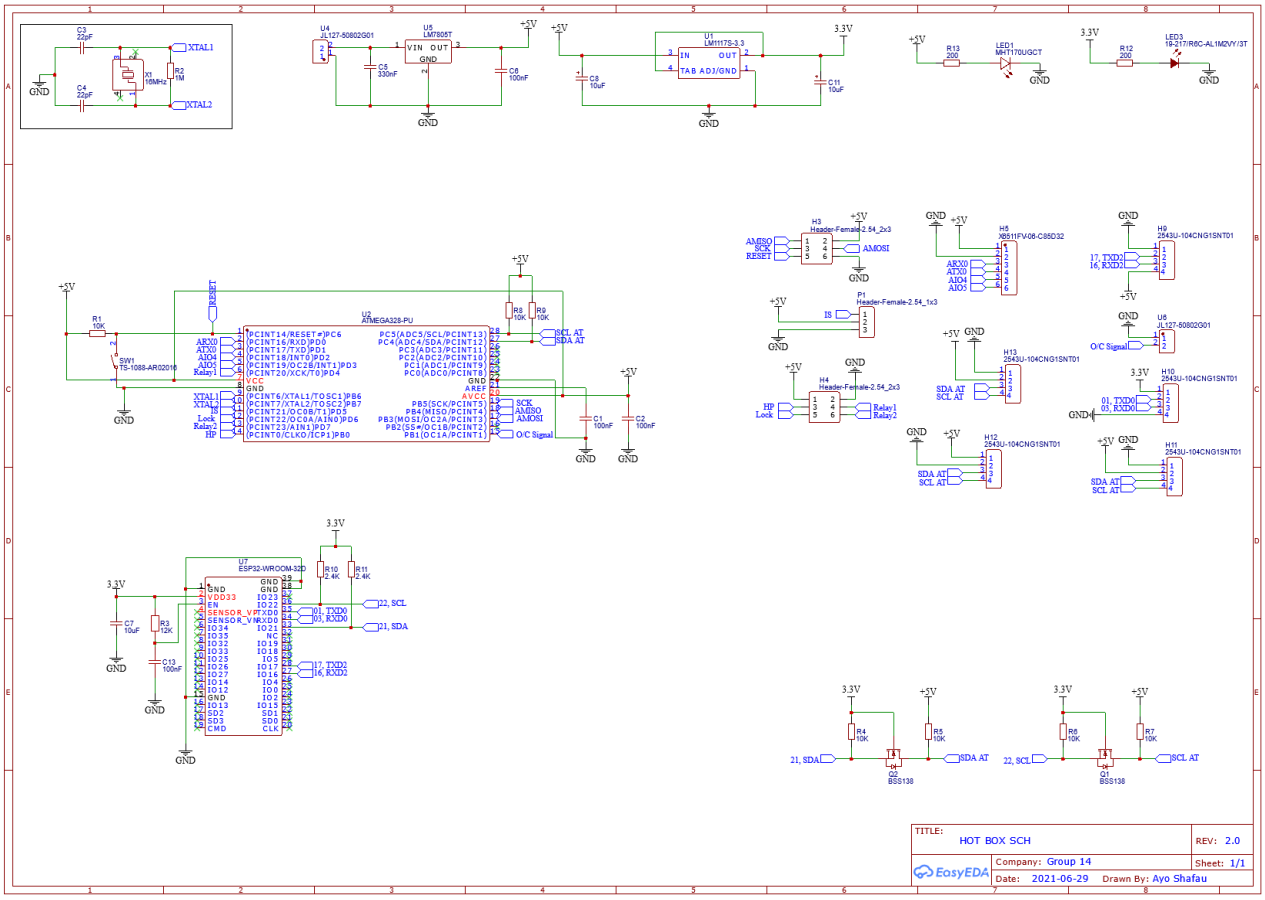


Figure 36. Overall PCB Design Schematic

5.4 Software Design

This section of the document will contain all the relevant software designed needed to make the HotBox function. All these designs are being written prior to actually making and testing any of them. As such, the designs in this section are more of a plan of how to create the code than a for sure design. Everything in this section is subject to change during the actual development of the project, depending on how everything goes during the development.

5.4.1 Demo Website

The HotBox revolved around the communication of a web application and the MCU in order to communicate what the box needed to do. We decided this web application would purely be for demo purposes. If this was to be manufactured, the web application would need to be expanded to make it easier to implement with the restaurant or the functions of the web application would need to be implemented to the restaurant's personal website for this communication.

For demo purposes, we will have multiple temperature options to show off the differentiating heat dependent on the heat option chosen by the user. This option was placed in the order form of the site, where a user would place an order. Once the form was submitted, the application communicated the information to the MCU, giving the box the order would be placed in and its order number so the user knew it was their order. The box was also told when to lock or unlock by the site or barcode scanner, ensuring the security the box was sought to have. The MCU will use software segments built to do all these tasks. The application just needs to communicate what the MCU needs done.

The demo site will need to have a user interface. At a minimum, the frontend interface should be easy to follow and easy to use. Due to the shortened time period we have over the summer semester, the application will be focused on having the minimum requirements done. Stretch goals will be worked on when all the embedded software is functioning. This frontend will be designed using React. React was used in a previous class and is rather simple to use. React will easily allow for making the basic interface but also can easily be used to expand into an advanced interface if time permits. Figma was used to plan out the user interface for the web application, giving a visual to aim for when coding up the frontend. (Below may or may not be this Figma design…)

The web app consisted of two pages, the status page and order page. The status page displayed all the boxes in the system, their current status, and their current or previous order number. This page also consisted of admin controls for the site, which could be accessed below the status page by inputting the admin pin. This would unlock admin control features, consisting of four buttons, an add, delete, lock, and close button, which would add a box, delete a requested box, lock a requested box, and close the admin modal respectively. Below are figures showing off these aspects of this page:

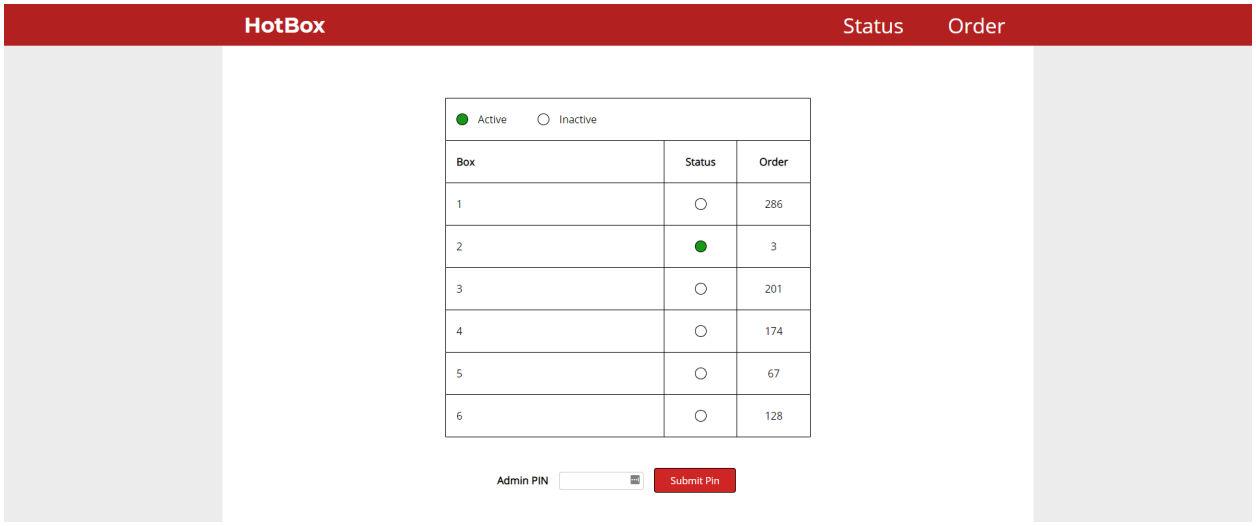


Figure 37. Status Page

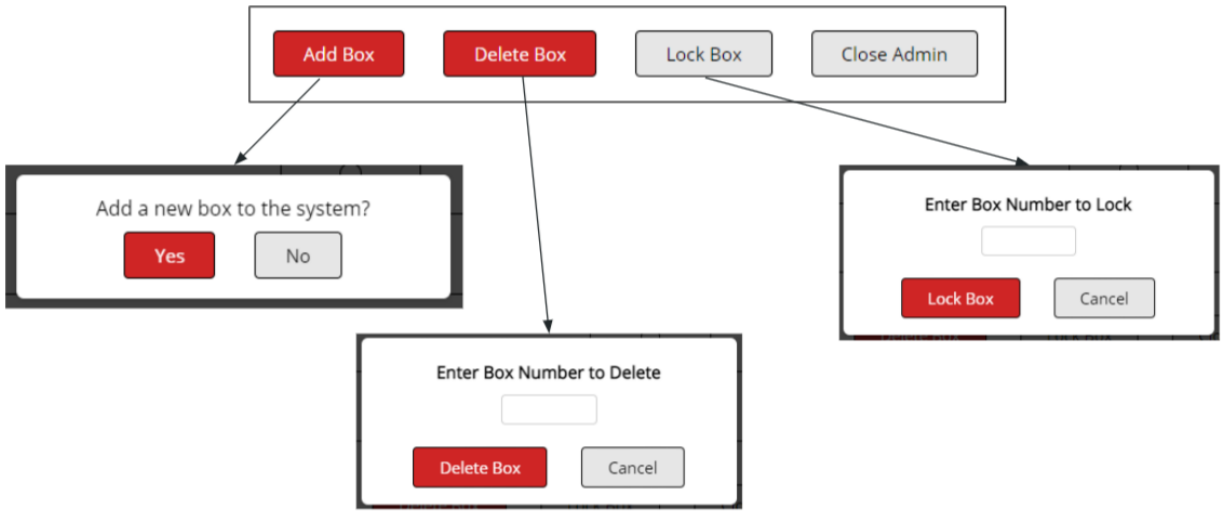


Figure 38. Admin Modal Commands

The order page allows a user to place an order. Due to the demo purpose of the site, the actual food item to choose is not an option and the box number requirement is only an option for demo purposes. These would be adjusted if the HotBox was implemented, as the site of that location would need to implement this site's controls. Submitting the order will give the user confirmation that the order was placed and send an email containing the box number their order would be placed in and the order's number. Below are figures showing off the aspects of this page:

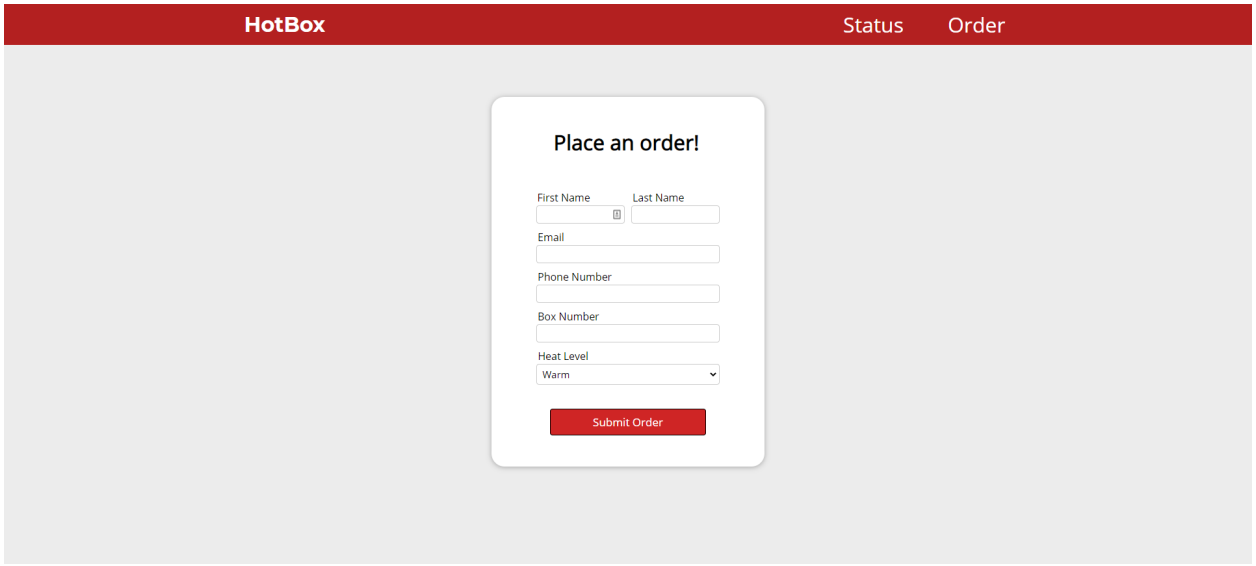


Figure 39. Order Page

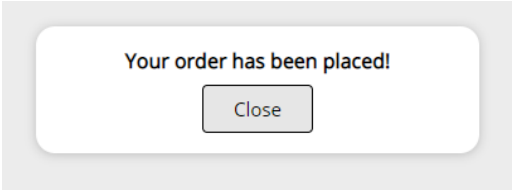


Figure 40. Order Page Confirmation

5.4.1.1 Backend

This section will mainly detail the specifics of the backend portion of the demo website and how it connects to the database. When designing the backend of the website, there are a few things to consider. As we have already decided which technologies to use, all of which are found in section 3.4.3, the first thing to consider is how we will connect our backend to MongoDB. NodeJS has two packages that we will use, the MongoDB driver and mongoose. MongoDB has an official NodeJS package that we will be using and mongoose is a MongoDB object modeling tool that is designed to work in an asynchronous environment. We will be using mongoose to connect to the existing cluster that we will have on MongoDB.

5.4.1.1.1 Schema Structure

The next thing to consider is the structure of the schema that will be occupying the database. The schema is defined as a uniform data structure that will store all the information we need about each box. All of the schema will be stored in a folder called “models” in order to have a comprehensive file structure. As we have not actually started programming, the actual structure may change later on, but for now the basic structure involves the following:

- An order number with type Number

- A BoxID of type String to uniquely identify each box
- A BoxNumber of type Number to identify each box on the frontend.
- A boolean value that defines if a HotBox is empty or not
- A Temperature value of type Number to send to the heating plate.

For the first point, it is important that we store the order number to identify which order is corresponding to each HotBox. The order number will be displayed on the box, so we need to at least store it. We will also display the order number on the web app so the user can see which order they are working on, this is something that is already implemented in most restaurant systems. The method with which we send the order number will be detailed in a later section. Obviously the order number will be taken from the online order, but the proprietary part of our product is the actual box, so we will not need to develop an ordering portion for our web app.

The second point involves a unique hex code that links a HotBox to the web app. There are many packages available that create randomized codes, such as bcrypt, so this is relatively simple to implement. This is mainly for verification or for when we are trying to use a GET request to retrieve information about a box. There are other alternatives for connecting a box to the web app, such as using the order number, but until we actually develop the application we will still use the hex code to link the box and web app.

The third point involves having a boolean value that shows whether or not a box is empty or not. This is mainly for the user of the web application, that is the employee that stocks the box with the food. Ideally, fast food joints would have multiple units of the HotBox, so having a system that displays which boxes are empty or full would speed up the overall process of filling the box. This is especially important since the HotBox really shines when there are multiple orders or a large influx of customers.

The following paragraph describes what a schema would look like when we actually program our backend. The first line would import mongoose, which is a NodeJS package that acts as a driver for MongoDB to interact with a NodeJS project. The next line defines the schema as a mongoose schema, so it is compatible with the format MongoDB requires. More information on the format for MongoDB can be found in section 3.4.2.2.1. The next section defines variables in the schema along with their individual parameters, all of which are written in JSON format in order to properly parse them to MongoDB. For example, a parameter that is strictly a whole number would have a parameter of being of type “Number” and a parameter of “required” being true. The first parameter just requires that the input must be a number while the second parameter means that this variable is required when querying the database with a POST or GET request. The final two lines simply export the schema and enable it to be used in other files.

5.4.1.1.2 Routing

Since we have all of our schema being stored in a folder called “models”, we will also store all of our routing files in a folder called “routes”. This is done to maintain a comprehensive file structure in an effort to stay organized. The purpose of the routing files is to set up all of the HTTP requests that any operation will require. The general HTTP requests will be for creating, reading, updating, or deleting data from our database. These are known as CRUD operations. Specifically, HTTP uses GET, POST, PUT, PATCH, and DELETE requests for RESTful APIs. In the following paragraphs, we will be detailing how we will use each request, what their responses are going to look like, and setting up commands that will be used throughout the entire web application.

In order to match our software requirements, mentioned in section 2.4.2, we will most likely have multiple route files to accommodate different parts of the web app. These parts include the connections the box will have to the database and the connections the web app will have to the database. The main requirements we will focus on are the following:

- Sending the order number to the box along with a QR code to the customer
- Sending box information to the database
- Sending box information from the database to the web app

For the first requirement, we have to use a combination of a POST request and a GET request to satisfy it. Firstly, we will need to create an order, which generates the order number and a QR code programmatically. Since our product is specifically the heating of the food, we will not need to store any information about food ordered, but we will need to store the generated order number in our database. The QR code would be sent to the customer through their email as well. After storing the order number, we would send it to the web app and the HotBox, since both would have to display it. The web app should show the order number that corresponds with each box so the employee can track each order. The HotBox should also be able to display the order number on the LCD so the delivery drivers can check which order they are picking up. The input will be in JSON format, as that is what MongoDB will store our information as, and the output will also equally be in JSON format for the web app connection, but the connection to the HotBox may require a different type of output in order to properly display the number on the LCD display. A positive response will respond with a code of 500, which will be easily seen with a testing environment such as Postman.

For the second requirement, we want to send information about the box to the database. To this end, the box will send HTTP requests to the database, such as POST, GET, and PATCH requests. We need to send whether or not the box is empty or not to the database. This information will later be displayed on the web app, which will be discussed in the next paragraph. This is the only main requirement that will involve HTTP requests with the HotBox, the rest will be using embedded programming to control the temperature of the HotBox, which will be covered in a separate section.

For the third requirement, we want to send information about the box from the database to the web application. This will require more GET and POST requests in order to retrieve the information that is necessary. The first bit of information we retrieve from the database is the state of the HotBox. As mentioned in the previous paragraph, the state of the box is whether or not the box has food in it. This must be sent to the web application using a GET request, as in retrieving the information from the database. The information will be sent in JSON format, the schema as mentioned in section 5.6.1.1.1 is structured so this value is a boolean variable. The second bit of information that we retrieve from the database would be the order number that will correspond to each box. The main idea is that the web app would display which box corresponds to each order. This is simply a GET request as well, with a 500 return code meaning an errorless response.

The following describes what one of the routing files would look like when we actually program the web app. The first line imports Express, which is a popular backend framework, into the file so we can use it when creating GET or POST requests. The next line imports the schema that will be relevant to this file, this is the same schema displayed in the previous section. The third line is there so we can reference the express API with the term “app”, so for example “app.get” or “app.post”. The next section is an example of what an HTTP GET request will look like in Javascript. The first parameter is how we will call this specific function, defined as ‘/FUNCTION_NAME’. So the web app will generally have the URL as “mywebapp.com/goals” when accessing this code. All functions will be asynchronous, meaning that all HTTP requests will be processed separately and in different threads. All HTTP requests are split into two parts, the request and the response. The request is the query that we send to the database, so in our web app, we would request for a customer’s order number. The response would be what is returned, so continuing the example, this would be the customer’s order number that we retrieved from our database. So this shows how we query the database and store it in a variable to be accessed in other files. Next is a try-catch block where we either send what was requested or an error code if it fails. However, this is only for a GET request. For a POST request, we are physically updating the database with a supplied payload. We also must recreate the entry in the database with “req.body” meaning the payload that was supplied. The following try-catch block is almost the same as the GET request, except we also must save the updated schema and then send the response. The final line simply exports the app which enables it to be used elsewhere.

5.4.2 Embedded Implementation

This section will detail the embedded programming required for the HotBox. The main requirements we have for the embedded implementation for the HotBox would be modulating the temperature of the box and unlocking and locking the solenoid lock. To do all of this, we decided the most efficient way to program this is by using the Arduino IDE. This allowed us to use many libraries that helped us complete our goal. We programmed both the AtMega3560 and the ESP32 using the Arduino IDE.

For the ESP32 we coded it to use HTTP requests. Using HTTP requests, we can update the status of the box using POST requests. POST requests are used to send data to a server to create or update a resource. By using POST requests, we can send JSON objects to the temperature module and update it. This will have to be connected to the web app, with the user updating the temperature via either a drop-down or a manual text entry. We tested the POST requests by using Postman, by checking if the actual temperature was updated once that portion of the box was built.

For the MCU, this controlled all of the main components of the box. This controlled the temperature regulation of the heating pad. It also controlled whether the correct barcode was scanned or not. If it was, the lock of the box would disengage and the box would be unlocked

5.4.3 Barcode Scanner

The HotBox must have a form of security. This security is meant to ensure that only the customer or the delivery driver of that order is allowed to access the contents of the box. Otherwise the box will be locked via a lock which will only be unlocked by a proper barcode given to said order.

The scanner that will be used to complete this task is the Waveshare 2D Barcode Scanner. This scanner will communicate with our MCU in order to lock or unlock the box.

The following is a flowchart showing the general flow of how the code for this will function:

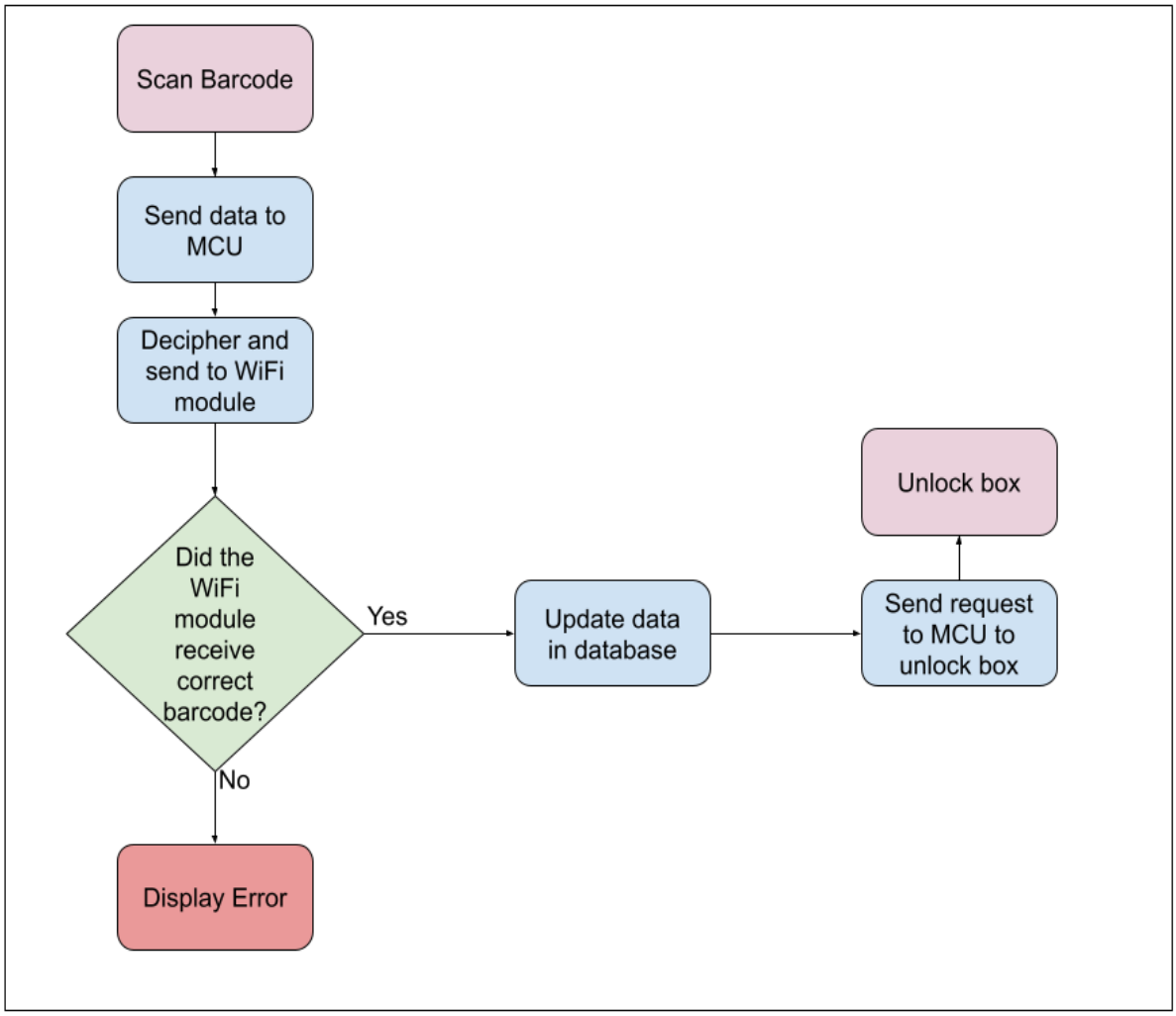


Figure 41: Barcode Scanner Flowchart

Here will be a more in depth explanation of the above flowchart. When a customer or delivery driver comes to pick up their order, they will have the barcode scanner scan their barcode. The scanner will send that data to the MCU, which will in turn decipher the request. This request will be to send the data to the web application, so the MCU will request the WiFi module to send the data over. If the WiFi module does not get the request, an error will be displayed. Otherwise, this data will be sent over to the web application. If the web application does not send a response after a certain period, an error will be thrown. Otherwise, the web application will have used this data to determine what box the order was placed in. The application will send this information back to the WiFi module which will send the request to the MCU to unlock the box. The MCU will unlock the box and end this request.

The Arduino library will be used to allow the scanner to get the information needed from the customer's barcode. This will in turn send a request to the MCU. The other operations contained in this flowchart will have been done in its specific section. The

MCU should already be programmed to receive and send requests and the ability to lock or unlock the box will be covered in that section.

5.4.5 File Structure

The file structure for our web app is very important to maintain, as it must be standardized so other group members can look at our code and know where they need to call certain functions. Generally, all MERN stacks have the same file structure. The backend is always in the root directory while the frontend is placed in its own folder in the root directory. We will also have a models folder which will store all the schemas, which are the definitions of the data structures stored in our database. This folder will be stored in the root directory along with the routes folder, which will contain all of the routing files. The root directory will also contain a folder called “node_modules”, which is installed by default when installing NodeJS. Any additional packages that are installed will also be added into this folder. The remaining 3 files found here are the “package.json”, the “package-lock.json”, and the “server.js”. The server.js file will contain the connection to the MongoDB database while the package.json file is used to give information to npm that allows it to identify the project and its required dependencies. The package-lock.json file keeps track of the exact version of all the packages installed. The “.env” file is a file that is usually highlighted in the gitignore file, meaning that The picture below displays the root directory and its file structure, which will likely be the same when actually programming the web app.

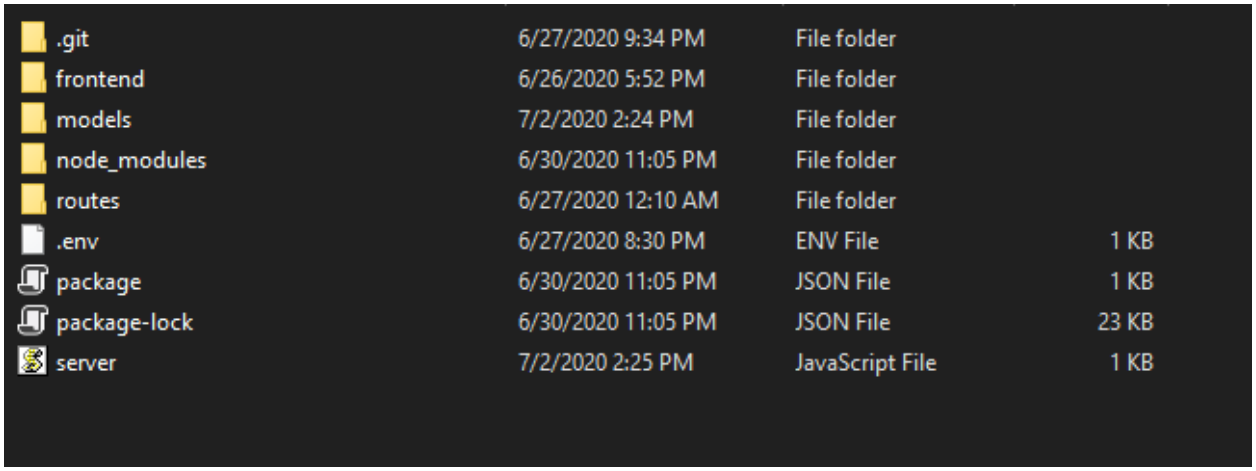


Figure 42: Sample Backend File Structure

Going into the frontend folder, it contains many of the same files as the root directory such as the package.json and the package-lock.json. Since we are using React, there is a command that automatically builds this folder, so the default folders that are created are a “public” folder, a “src” folder, and a “node_modules” folder. The src folder contains all of the CSS files and the Javascript files that control the UI of the web app. Any image files that are displayed on the web app will also be stored in the public folder. The public folder contains images, stylesheets, and fonts from Javascript. It also contains the manifest.json file, a json file that is installed by default when adding React to a web app.

A web app manifest is included in a React app so anyone can install our React application on their device. The sample file structure for the frontend folder is located in the following figure.

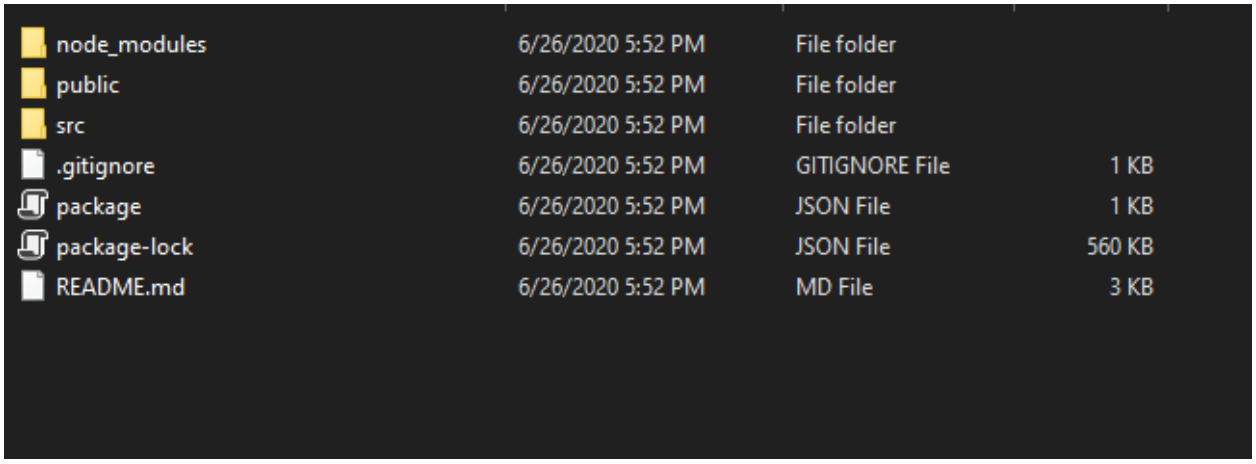


Figure 43: Sample Frontend File Structure

6. Project Prototype Construction and Coding

This section will go into detail of the materials and software that will be used to build the prototype. The definition of prototype is a first, typical or preliminary model of something, especially a machine, from which other forms are developed or copied. The reason this is important is because although it will be our first model of our product we need to use this to perfect the design. When developing the prototype, we need to observe any negatives with our designs to eventually perfect our final design. This section will go into detail with the type of PCB we will be creating and the other facilities and software that we will use in our project.

6.1 PCB Vendor and Assembly

In this section, we will discuss possible Printed Circuit Board Vendors and Assemblies. There are many vendors that we can use to print our circuit board and we will discuss the advantages and disadvantages of them here. Most of them are similarly priced with a few exceptions.

6.1.1 JLCPCB (JIALICHUANG Printed Circuit Board)

The company JLCPCB is a great vendor for PCBs. They are a worldwide leading PCB prototype enterprise and a high-tech manufacturer specializing in quick PCB prototype and small-batch PCB production. This is our top vendor for our PCB.

With over 14-year continuous innovation and improvement based on customers' needs, they have been growing fast, and becoming a leading global PCB manufacturer, who provides the rapid production of high-reliability and cost-effective PCBs and creates the best customer experience in the industry. They are the largest PCB prototype enterprise in China and a high-tech manufacturer specializing in quick PCB prototype and small-batch PCB production.

Some benefits for using JLCPCB is their advanced PCB technology allows them to provide high precision boards suitable for industrial, military, aerospace, and medical applications. They are continuously investing top-level base materials and advanced equipment for fully automated production lines.

Since 2006, JLCPCB has continuously driven to become more efficient and reduce costs. They promise to offer customers the most economic PCBs forever. JLCPCB makes the cheapest but top quality PCBs possibly because of scale effect, extremely high production efficiency and less manpower cost.

Their customer service is also exemplary. They have an easy-to-use online ordering system, professional and efficient customer service, digital manufacturing technology, full-automatic production lines, and stable logistics partners make every step to deliver the PCBs faster.

With all these advantages, there are a few disadvantages. Although they advertise that their delivery system is fast and reliable, their headquarters are located in China. This means that the delivery will not be as fast as some vendors that are based in the US. Also, China is known to sometimes have shortages of critical materials to build certain products like the current shortages of GPUs. This could affect the time we get our PCB. Another disadvantage is that the parts are cheap. They will not be as high quality as some other vendors that charge a little bit extra.

Below is an example of one of the PCB from JLCPCB. As you can see, the quality of the product does not look cheap although it might be cheap. This picture is from one very satisfied user.

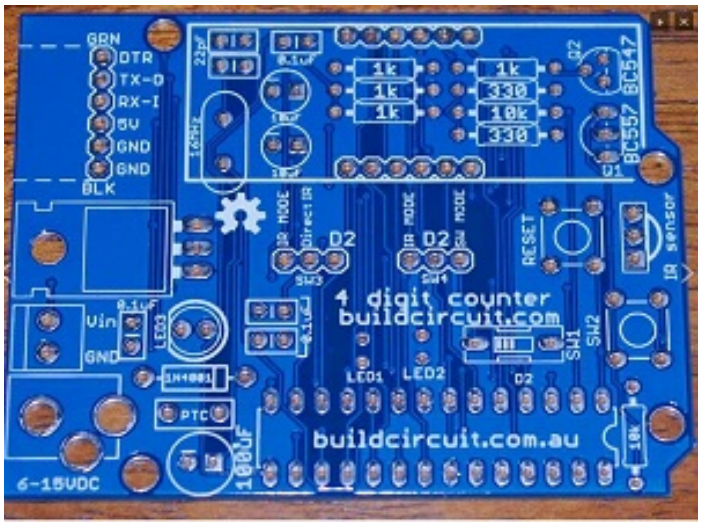


Figure 44: JLCPCB Example

Table 29. JLCPCB Features

Customers	800,000 +
Orders Daily	20,000 +
Total Factory Area	450,000㎡
Production Capacity/Month	620,000㎡
PCBs Produced/Year	6 Million +
Countries Covered	170 +

Employees	3000 +
Years Founded	14
On-time delivery	99.97%
Quality Complaint Rate	0.23%
Online Service	24/7

6.1.2 Jabil

Jabil Inc. is an American worldwide manufacturing services company. Located in the Gateway area of St. Petersburg, Florida, it is one of the largest companies in the Tampa Bay area. Jabil has 100 plants in 30 countries, and 260,000 employees worldwide. Additionally, Jabil gives back to multiple nonprofits and charity causes here in the Tampa Bay area and beyond.

This company provides electronic manufacturing services and solutions throughout the world. The Company operates in two segments, which include Electronics Manufacturing Services (EMS) and Diversified Manufacturing Services (DMS). Its EMS segment is focused on leveraging information technology (IT), supply chain design and engineering, technologies centered on core electronics, sharing of its manufacturing infrastructure and the ability to serve a range of markets. Its DMS segment is focused on providing engineering solutions and a focus on material sciences and technologies.

Jabil is involved in design engineering services. The company has industrial design services that concentrate on designing the look and feel of plastic and metal enclosures that house printed circuit board assemblies and systems. Mechanical design services of Jabil include dimensional design and analysis of electronic and optical assemblies. Computer-assisted design from Jabil includes printed circuit board assembly design testing and verification and other consulting services.

Jabil has combined an unmatched breadth and depth of end-market experience, technical and design capabilities, manufacturing know-how, supply chain insights and global product management expertise to enable success for the world’s leading brands.

This PCB company has many advantages. The biggest advantage is the fact that the headquarters is located in St. Petersburg, Florida. This means that the delivery time will be extremely fast compared to other PCB companies. However, there are also some disadvantages, such as the price. The price overall is not as good as JLCPCB for the quality of the PCB..

6.1.3 JLCPCB vs. Jabil Comparison Table

Table 30. JLCPCB vs. Jabil Comparison

Features	JLCPCB	Jabil
Customer service	24/7	Standard 9-5
Amount of Employees	3000+	260,000+
Countries Covered	170+	30 +
Delivery Time	Estimated 1-2+ weeks	Estimated 3-4+ weeks
Quality	Cheap but is lower in quality	More expensive but higher in quality
Price	$29	$49
Shipping costs	$2	$5
Location	China	St. Petersburg, FL

6.1.4 PCBWay

PCBWay is a manufacturer specializing in PCB prototyping, low-volume production and PCB Assembly service all under one roof. They offer quick turn PCBs at a very budgetary price. This is another PCB manufacturer that is located in China and not in the US.

PCBWay advertises a high delivery rate. They claim that through the years, they are proud to have been keeping an on-time delivery rate of 99%. This is very similar to JLCPCB as well. They understand that apart from PCB quality, the other most important factor is the shortest possible lead-time, which is crucial for us engineers' works, especially in the stage of prototyping.

PCBWay offers 10 quantities of 2 layer PCB for a price of $5 plus shipping. They offer a feature called turn-around or quick turn time. They pride themselves as one of the leading suppliers of PCB prototyping and low-volume production in the world. Their turn time can be as short as 24 hours after your Gerber files are reviewed and approved by our engineers, greatly saving you time in your work. There are other time schedules to fit the budget. They offer 48-hours and 72-hours shipping too.

They claim that being quick in delivery never compromises on the quality. With their stringent test procedures, we can be assured that we will be satisfied with our PCB for your high-tech work. Apart from the quality, they believe that they can beat almost every other fabricator on prices. Finally, they claim that If we find any other fabricators who offer lower prices, they will lower their prices and do a price-match.

All of these advantages are good, however there are some disadvantages. After doing more research, we have found some reviews of the company and their products were not the best. There are a few reviews that indicate that the product is not how it is advertised. Another disadvantage is that this company, just like JLCPCB is located in China. Although they advertise fast shipping and delivery, it is not confirmed with these companies.

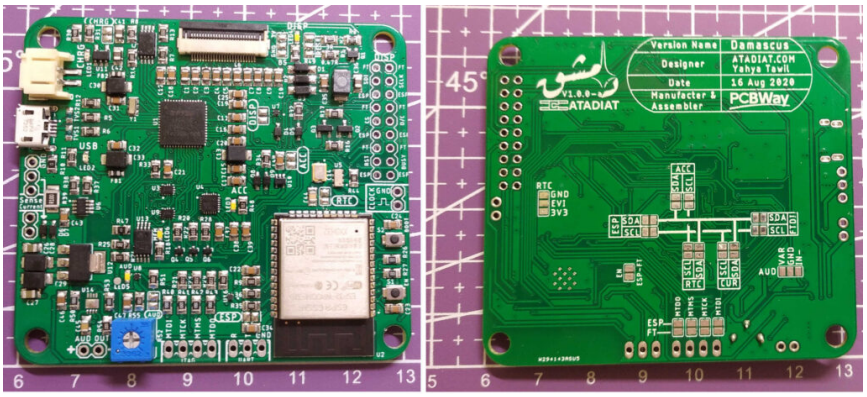


Figure 45: PCBWay Example

6.1.5 JLCPCB vs. PCBWay Comparison Table

The following is a comparison table of the above PCB options we can choose from for our project.

Table 31. JLCPCB vs. Jabil Comparison

Features	JLCPCB	PCBWay
Customer service	24/7	24/7
Amount of Employees	3000+	500+
Countries Covered	170+	170+
Delivery Time	Estimated 1-2+ weeks	Estimated 1-2+ weeks
Quality	Cheap but is slightly lower in quality	More expensive but slightly higher in quality
Price	$29	$40
Shipping costs	$2	$5
Location	China	China

6.1.5 Conclusion and Final Decision

After researching multiple different PCB vendors and assemblers, we have come to a decision between 2 companies: PCBWay and JLCPCB. In this section we will be discussing the advantages and disadvantages of each and our final decision at the end.

The two companies have a lot of similarities. Both JCLPCB and PCBWay are located in China. They both offer great customer service compared to other vendors. There are only a few things they differ in like the costs of productions. One costs almost $50 while the other costs around $30. The quality of each according to reviews are really similar so to spend that much on the board will not be smart. However, PCBWay says that they will match any competitors' prices. This could be very useful when making a decision.

Finally, we believe that JLCPCB will be the best vendor to use for our PCB. There are many reasons why we decided to go with this company. The customer service is great and better according to some reviews. We are afraid because of previous owners, that PCBWay will not be as advertised. We also have had many recommendations to use JLCPCB from other students who have completed their Senior Projects and from UCF faculty. This is why we are going to use JLCPCB as our vendor

6.2 Final Coding Plan

This subsection of the document will discuss the overall plan on developing all the software as a whole for the project. This section will have three subsections for each of the software designs: embedded software, frontend design, and backend design.

6.2.1 Frontend Plan

The overall plan for the frontend is to implement a user interface for the web application that can send and receive all the requests it needs in order to allow the project as a whole to function properly. This will be done by implementing the designs from a previous section.

The first part of this plan will be designing the frontend UI prototype in Figma. This will be done early in Senior Design II. This design will be presented to the group to allow everyone to have a say in how its design should be. While the demo application will be primarily focused on its functionality rather than how it looks, having a base to go off of and a goal to make it look like will help drive the design. This design will need to contain a UI that allows a user to select what food they wish to order, and thus, have heated till pickup and the ability to actually pick up the food. The rest of the frontend design will be implementing the API and communicating using the API and backend to do the above tasks.

Once the Figma design has been developed and approved by the group, the actual development process will begin. We will be using React in order to create the frontend portion of the design. The basic UI design should not take long and should be done relatively early to allow more time to be spent on the more difficult portions of connecting the frontend to the API. This design portion will also contain developing the functions the frontend will need in order to store information that will be passed to the API.

After the basic UI is complete and the API endpoints have all been connected, the resting process will begin. This testing process will be further explained in a following section but in general, this period will contain all the testing needed to ensure the software is functioning as desired. Software often needs a lot of debugging so we will ideally get to this step as early as possible so we can have ample time to find and fix all the bugs that could get in the way of a working product.

Finally, if the above was all done and there is still remaining time prior to the demo and presentation of the product, we will be able to clean up the UI and make it look as nice as possible. This will be where we really focus on matching the original Figma design and adding the more difficult aspects that are unnecessary and in general a waste of time should all the important aspects not already be completed. This final portion is a stretch goal and is fully dependent on time constraints.

6.2.2 Backend Plan

The overall plan for writing code for the backend is to set up the connection to the database and ensuring that all API endpoints are functional. The backend is the first thing that must be done from a software standpoint in order for the project to progress. Without it, the frontend will eventually be stunted in its progression because it is waiting on the backend to have fully developed API endpoints. For essentially everything else in the project to function, the backend must be developed first.

The first thing that needs to be done is to set up the file directory where we will be storing our web app. Since we are using a MERN stack, we will be installing all the components of that stack into the file directory. To that end, we will first need to install npm and NodeJS to the file directory. NodeJS is the backend framework for our application and npm is a package manager that will be dealing with all of the installations of various packages that will be used in our project. We will also need to install Express, since we are using that for the web application framework for NodeJS. Express will be used for simplifying the writing of the server code when deploying our application to the web. Since we are writing in Javascript, we will also need to have some files prepared before we actually start programming our backend. These include a package.json file and a server.js file. The package.json file will hold all of the metadata relevant to our project, such as the dependencies and scripts of our project. The server.js file will hold all of the connections to the database and the localhost port opening when testing our application locally.

The second thing that needs to be done is to connect the database to the backend. Since we are using MongoDB as our database, we need to research what is required to connect it to a NodeJS application. From previous experience, we know that we will need to use mongoose, an npm package, to connect our NodeJS app to the database. So first we will need to install that using npm on the command line. The specific command is “npm install mongoose”. Once this is done, we will need to create the cluster that will hold our database on MongoDB so we can get the database URL. This database URL will be placed in our server.js file. This is all that is needed to connect our database to the backend that will be created.

Next, we will need to write schemas for data that will be placed in the database. All of these schemas will be placed in the models folder located in our file directory. The structure of the file directory is important to maintain, as exporting and importing in Javascript is dependent on the file location. The first schema we will need to write is for the order number of the box. As described in section 5.6.1.1.1, the order number is typically taken from an ordering application, but the proprietary part of our product is the actual box so we do not need to develop an ordering system. Schema for MongoDB is written in the MongoDB Query Language, or MQL, which is very similar to JSON formatting. An order number is a relatively simple data structure to program for. All that is needed is to specify the type being of type “number” and the required value set to “true”. The type being “number” means that the order number cannot be a decimal, which is fine since order numbers are typically whole numbers. Along with that, the

required value being set to “true” means that the order number must be in the input, otherwise the input payload won’t be put in the database.

We will also need to write a structure for a value to connect a HotBox to the web app. This will be written as a string that will uniquely identify each box. Since all the boxes will theoretically have different orders in each box, it will be important to uniquely identify each box beyond the order number that will be displayed on the box. Using an ID also adds another layer of security beyond the barcode scanner that will be implemented. This code is randomized and stored in the database. As for its schema structure, it will be stored as type “String” and its required value will be set to “true”. There’s no specific type for hex values so type “String” will have to do.To add on to this, we will also have a BoxNumber that will be used for display purposes and for some inputs on the frontend to make it easier for a user to reference each box. This will be of type Number and will automatically be set and incremented each time a box is added to the database.

The final schema structure that we must have is a boolean value that tells us if the box is empty or not. Ideally, this process is automatic, where this value is constantly being updated. This is very simple to program, with only two parameters to add. One for a bool value with it’s default value set to “false” and the other being a required value being set to “true”. All of these schemas should be exported so they can be used in other files. This should be all the schemas that will be needed to be made.

The next step would be to set up all the routing files, all of which will be stored in the routes folder. The routing files will contain all of the API endpoints that will send HTTP requests to and from the web app. This includes updating the order number for each box, updating if the box is empty or not, and sending all this information to the web app as well. The HTTP requests that we will be mainly using for this are POST and GET requests. POST requests update the database and GET requests fetch information from whatever source is needed.

To update the order number, we will be using a POST request to update the information. Realistically, we would also need to implement a GET request so we could retrieve the order number from the order, but this will not be needed since we are not developing the ordering portion of the entire process. Once we get the order number, we will need to first store that in the database and then send that order number to box. The order number needs to be displayed on the LCD so it needs to be sent from the database.

Updating the box information mainly entails sending the state of the box to the database. This mainly uses a POST request that updates the boolean value of the schema in the database. Once the barcode scanner gets a positive reading, meaning that the person retrieving the food from the box opens it, we should update the state of the box from true to false. Similarly, this process should also be reversible so the box’s state can be updated from the web app as well, so when a worker puts in an order into the box, they can also update its state. After writing all the API endpoints in the routing files, we then need to export all of them properly so they can be used elsewhere.

Overall, after implementing all the features described above, this should cover everything needed for the backend. Obviously, when actually programming the backend, other issues may come and other needs may also come up so we'll need to be able to adjust and add any functionalities that may be required.

6.2.3 Embedded Software Plan

Arguably the most important software of the project is the embedded software. Without it, the project has no hope of functioning in the slightest. The development of the embedded software will be a major focus for the first half of Senior Design II. While each portion by itself should not be overly complicated to design, having every bit of software work together could slowly turn into a complex problem. Thus, ample time should be allocated for this portion of the design.

The first thing to be done is research. While plenty of research was done for the development of this document, coding specific research is hard to do until the actual coding process begins, as we cannot know every little requirement we might need or issue we will run into. We must first get the MCU software completed so that it is ready to work with the other embedded software. We will likely also add the needed code to communicate with a server at this point, as the communication between the MCU and web application server could be one of the harder development processes but also a vital part of the project as a whole.

Once we have the MCU setup and coded, we can begin adding all the small software bits that will be required to run the project as a whole. These bits include software such as the barcode scanner and reed switch. Ideally these small designs should not be too complex, especially with the help of the Arduino libraries, and we can move onto the next step as soon as possible.

This final step will be testing and debugging all the embedded software. It is beyond important that the embedded software works properly and has no bugs. Unlike the demo site, a bug in the embedded software could stop the entire project from functioning. For instance, if our reed switch does not work, it is highly possible that any requests made would be denied as most requests will require the door of the HotBox to be a specific state. The way embedded software builds on top of each other creates major reliances between them. We will need to create a well oiled machine of software to create the product we have invisionsed. Once we get done with this step, the majority, if not all, of the software implementation for the project will be done.

7. Project Prototype Testing Plan

The following sections will detail how we plan on testing both the hardware and software side of our product. The hardware testing will cover the testing of the wifi connectivity for a HotBox, the barcode scanner, the lock testing, temperature sensor, heating pad, and the LCD display. Confirming that all of these are working are imperative to the success of our project, so we need to make sure that our testing can account for as many issues as possible. In terms of the software, the following sections will explain how we will test all of the API endpoints in terms of the tools that will be used and what those results could possibly mean. Ensuring that the software side of the project is important since that will be one of the first things finished, so the hardware can safely test their own modules with the full frontend and backend.

7.1 Hardware Testing

In this section, we will discuss the hardware testing of essential parts and components used to create the HotBox. Each component will be tested individually in both wiring and software to ensure proper functionality.

7.1.1 Wi-Fi Testing

To test the wi-fi connectivity of the HotBox, we will be testing the wi-fi module which is the ESP32. We will first test the ability to transmit data between the microprocessor (ATMega2560) and the ESP32 using UART. The next test will be to successfully connect to the internet by connecting to a host. Once that test is complete, we will then test the ability to use GET and POST commands and successfully read and update data from a server.

For the first test, we will be trying to send a simple string from the ESP32 to the ATMega and have the string printed out on the monitor of the ATMega console. Then we will repeat the same thing except backwards, sending a string of data from the ATMega to the ESP32 and have it viewed on that console as well. This will ensure that the UART connectivity between the two is successful.

For the second test, we will have the ESP32 connect to the internet. There will be an available 2.4 GHZ connection that will be made. Once the connection is made to the internet, a PING will be made to a website and print the ping time. If the ping was successful, it will print the ping time onto the ESP32 console. This will ensure that the wi-fi module will be able to connect to the server to transmit and receive data between the website and other devices as well.

Lastly, for the third test we will be testing the GET and POST commands. We will start off by creating a POST command to write into an API a JSON string. Once done, we will then create a string that outputs onto the console the successful POST command. Then using the GET command, we will try and get the POST command we created and print

the same information that was POST onto the console from the API. This will ensure that the POST and the GET commands are working properly for usage.

7.1.2 Barcode Scanner Testing

The barcode scanner is essential when it comes to customer validation and unlocking the HotBox. How the scanner works is within the structure, a red beam of light is flashed when the scanner is triggered, the module will then extract the alphanumeric digits from the barcode, and compare it to the values present in the database. For Testing, we will use serial communication to transmit and receive data from our scanner module to a PC/database. Before scanning can be performed, we must set up the barcode scanner to use serial TTL communications. By doing so, this will place the module in the correct mode to receive and transmit sequential data. In order to set it to the correct mode, we scan in the barcode located in the settings manual for the Waveshare 2D barcode. Below is an image of the barcode.

Figure 46. TTL/RS232 mode barcode configuration

7.1.3 Lock Testing

The lock will be our main point of security for the HotBox, meaning testing will be very crucial for this step. For this test, we test a 12VDC solenoid lock to ensure that it locks and unlocks properly when 12VDV is applied to it. In the main application it will be connected to a 5V relay. Other tests include making sure the lock is secure and in the correct position when the box is closed and unbreachable once it is in the locked position fastened into the custom fasteners. Below are some images showing the lock with a ~12VDC applied to the unlock state of the lock as well as 0V when it is in the lock state.

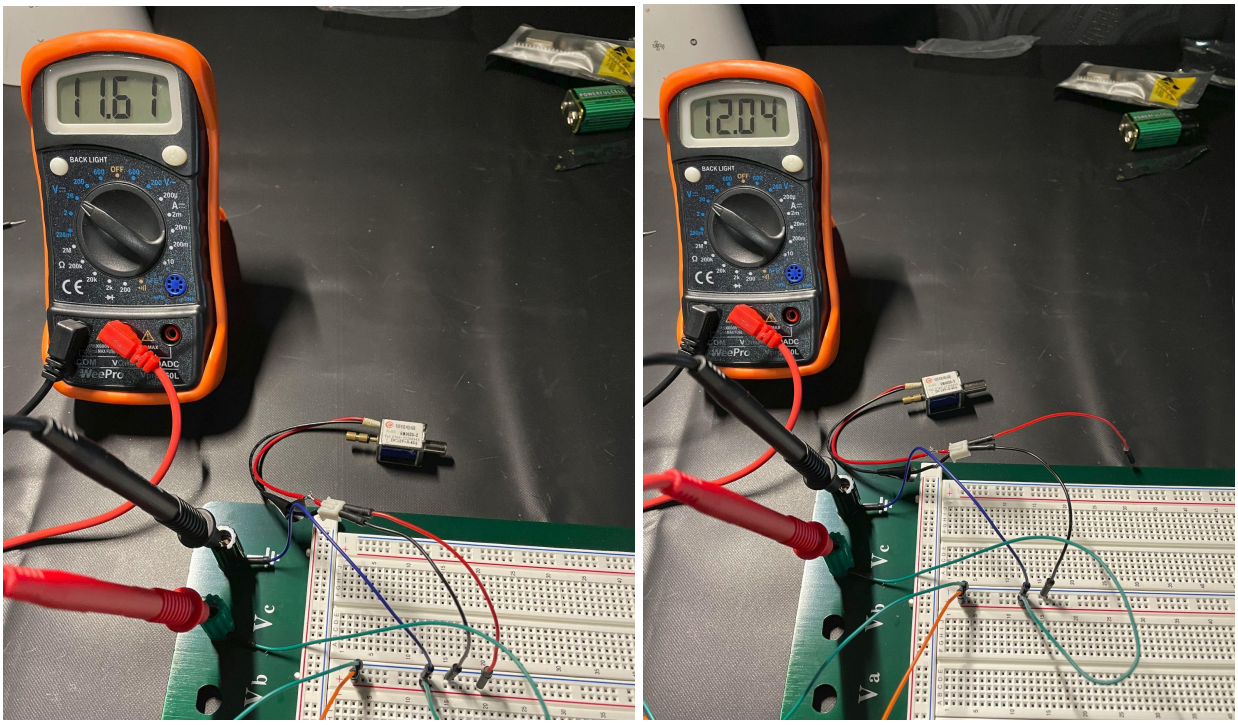


Figure 47. and 48. Lock in unlocked and locked state

7.1.4 Temperature Sensor Testing

Present in the HotBox, is two forms of temperature sensing, both work the same way. Inside the box, and on the heating pad. However for simplicity purposes, we will only be testing the thermistor which will be located on the heating pad. For testing, we decided to sense the temperature of a simple room to make sure that it is able to read and pick up accurate temperatures. We took a simple NTC thermistor and connected it to one of the ATMega2560 input pins and had the temperatures output onto the Arduino console. Below shows the testing wiring as well as the results. As you can see the thermistor successfully reads the temperatures of the environment successfully.

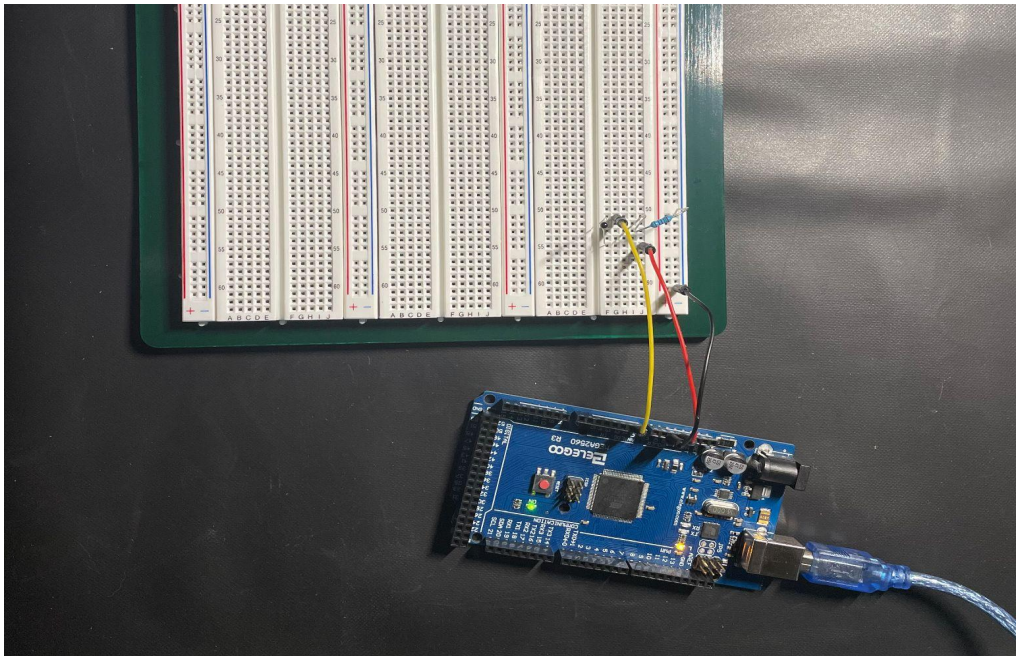


Figure 49. Temperature Sensor Testing Wiring

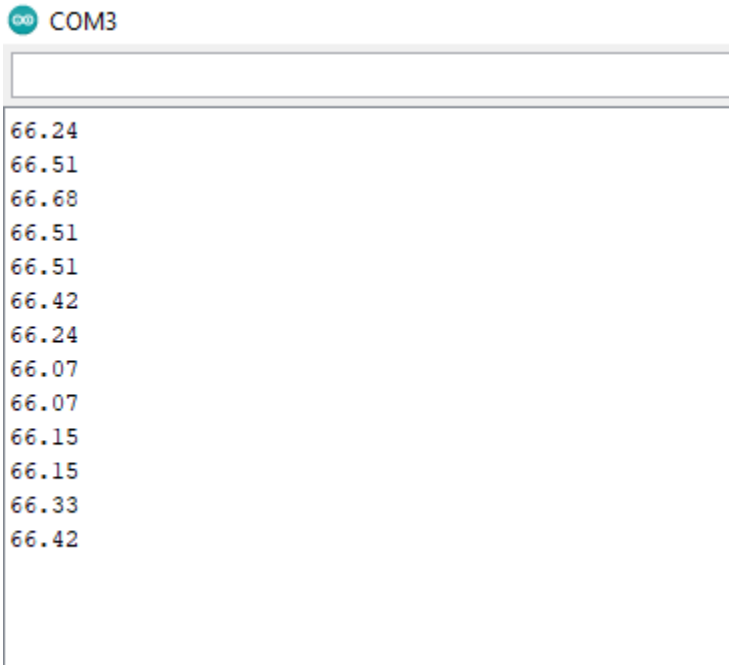


Figure 50. Temperature Sensor Testing Results

7.1.5 Heating Pad Testing

The heating pad is another crucial part of the box that needs to be tested because it is responsible for heating the food. The Heating Pad takes in 24VDC so we will use an AC/DC transformer to power on the heating pad and make sure it starts heating. We will then wire it with a relay and connect the relay to the ATMega where we will test if a

signal from the ATMega can power on and off the heating pad. We will test this in conjunction with the temperature sensor in order to ensure it heats up to temperatures that meet the HotBox's requirements.

Figure 51. Heating Pad Testing Results

7.1.6 LCD Display Testing

The LCD Display will be used to show customers order information of the order present inside the box. For testing, we will make sure that a string of text appears onto the LCD display. Because we are using I2C, we saved a lot of time and space with the wiring because I2C only requires 4 wires.

To begin testing,we first wire up the LCD Display and we first must figure out the I2C address of the LCD display, to do this we write a simple code to figure out the address. Below are the test results for finding the I2C address of the LCD Display.

Figure 52. I2C Address for LCD Display

Once the address is figured out we can then go ahead and output a sample string onto the LCD display to see if it is functioning properly. Below shows the wiring and testing of the LCD display.

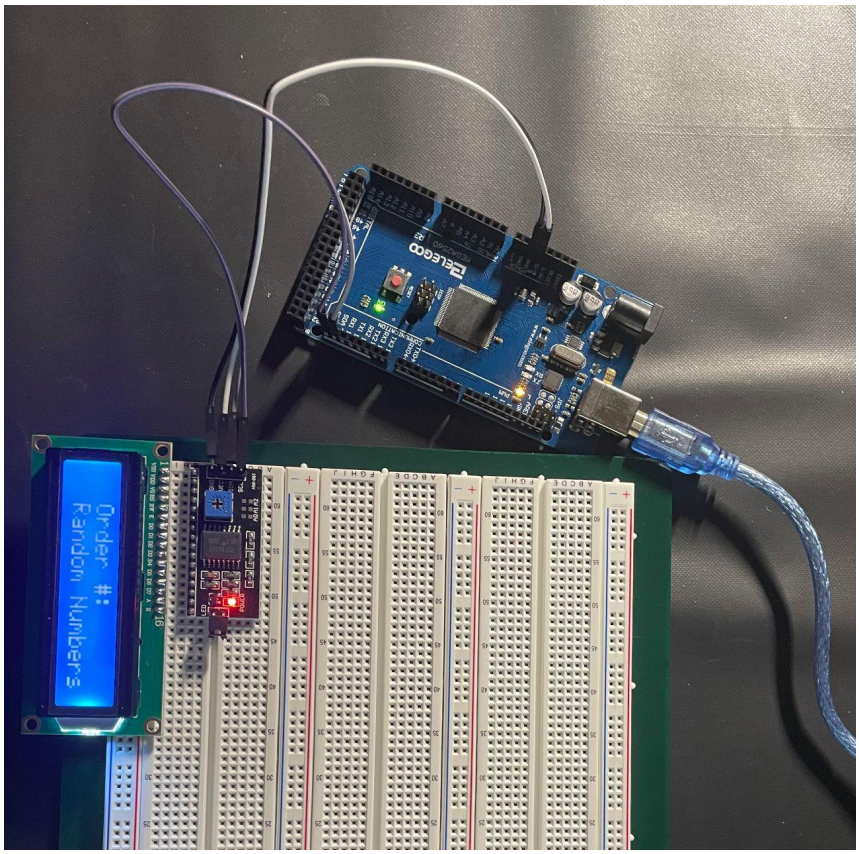


Figure 53. LCD Display Wiring and Testing

7.3 Software Test Environment

In terms of testing the software, there are several options that we can use to test our project. The first thing that we would test is the connection between our input, whether that be from the restaurant applications that send orders to us or data that we receive from the HotBox, to our database. Another thing that would need testing is the connection between our database and the companion web application, mainly testing the interface that displays the information. The following few sections will cover various applications that are valuable in assisting us for creating a software testing environment.

7.3.1 Postman

Postman is an application mainly used for testing the backend of our application. It allows for a developer to easily create, share, test, and document APIs. For our specific purposes, we would use it to send sample web API requests to ensure that our backend software is properly connected to our database. Postman allows users to send POST,

GET, PATCH, and DELETE requests and permits them to add an additional JSON payload that specifies specific details that should be sent along with the requests. Postman has also benefited from being able to be automated. It is very important to be constantly able to check test cases every time an update is made to the backend, so having an automated testing environment makes it very easy to catch bugs early on.

7.3.4 Testing Tools Comparison

Table 32. Testing Tools Comparison Table

Environment	Benefits	Limitations
Postman	- Most complete API development/testing environment. - Easy to create test suites that contain multiple integration tests. - Easy to move test cases from one system to the next.	- Does not provide codeless testing. - Lack of development environment compatibility. - Can only test 1 API continuously.
Swaggerhub	- Very beneficial for documentation purposes. - Can execute API class from the documentation which helps testing. - Is able to deploy to different host servers such as AWS.	- Mainly used for documentation, thus has limited testing capabilities. - Is mostly written in YAML and RAML.
Selenium	- Integrated with Agile, Azure DevOps, and continuous delivery. - Supports mobile testing. - Supports 10 different programming languages.	- Has no reporting capabilities, so it is difficult to understand what's wrong. - Has a steep learning curve. - Requires third party tools to reach full potential.

7.4 Software Specific Testing

This section of the paper will discuss the testing procedures that will be applied to the software side of the project. This includes the testing of the embedded software, the frontend functionality of the web application, and the backend functionality of the web application. Each of these three topics will be discussed in their own subsections. These procedures are only an initial plan prior to the actual development process and are subject to change depending on what is needed or changed.

7.4.1 Frontend Testing

Frontend testing can be simple but also very difficult. A well known fact for software developers is that something will work as intended to the developer but might not work in cases the developer did not account for. As such, the frontend must be tested both for its proper usage as well as improper usage. The site should not break if a user goes around clicking parts of the site that were not intended to be used. However, since the web application is meant to be a demo site and not a final product of an attached web application, some latency will be allowed to focus on it demoing the product.

The frontend will go through a series of tests both during development and post development. The site will be run in different browsers and in different browser sizes. This will test the site for flexibility, ensuring it is not built to only be run on Chrome in full screen mode for example. The decision of how flexible the site should be will be chosen during development dependent on overall time, but the site at the least should not break from the above testing. The site will likely be meant only for a computer and thus might not work properly in a phone browser. Development of mobile browser compatibility will be a stretch goal should time permit.

All functionality of the frontend will go through a series of tests. This will include ensuring the frontend is properly sending and receiving data through the API endpoints, checking that each function the frontend may contain correctly runs as per designed, and ensuring the site as a whole looks the way we want it to.

As stated, the web application is being designed purely to test the functionality of the product. We will focus on the site's functionality first and foremost and view any extra additions to the site as stretch goals to clean up the site. The main functionalities of the website are to display food options for the user to choose from, to communicate with the server and hardware to run the correct settings and functions related to this choice, and to allow communication with the user to display where their food is being held when they are ready to claim it.

7.4.2 Backend Testing

This section will be used to explain the testing procedure for the backend. This includes testing each API endpoint, ensuring that the database is updated when using each API endpoint, and ensuring all of the schemas are correctly being recreated in the database. Unless these things are all working, the entire product will not function at all. So ensuring that these are working perfectly is of vital importance.

Beginning with the testing of each API endpoint, we used Postman to test each endpoint. Postman allows developers to send JSON payloads through an endpoint by connecting to the host url. It is fully customizable, so sending POST or GET requests are both viable and can have any JSON payload attached. Granted, the payload must be accurate according to the corresponding schema that is attached to that data structure. Otherwise, we will simply receive an error code that could cause the

application to crash. It was our job to account for errors in the case that unexpected requests are made. We will also need to effectively add comments so our frontend developer can understand what each endpoint does.

Testing if the database is updated after using an endpoint is also important. If the database is not being updated, that means something with either the database connection or the actual endpoint is not functioning correctly. To do this, we can once again use Postman to send JSON payloads through each endpoint and check what is being updated in the database. It is important to ensure that the database is being correctly updated since many GET requests will be made to the endpoint to retrieve data. Should any of the information be wrong or corrupted, it can halt several other processes such as displaying the data on the web app or displaying the data on one of the HotBoxes. Halting other processes can result in the entire application crashing, which is something that must be avoided, so it is vitally important to make sure the database is being updated correctly. The picture below shows how we use Postman to test each endpoint. The input is shown in the body, set in JSON format, the output is shown at the bottom along with a server response code. The example shows the testing for the “GetStatus” endpoint, which takes the ID of the box as a parameter from the URL and returns the value of the “Empty” field located in the document of the box matching that ID. The red square at the top represents the input, the red square in the bottom left represents the output, and the final square shows the status code after running the operation.

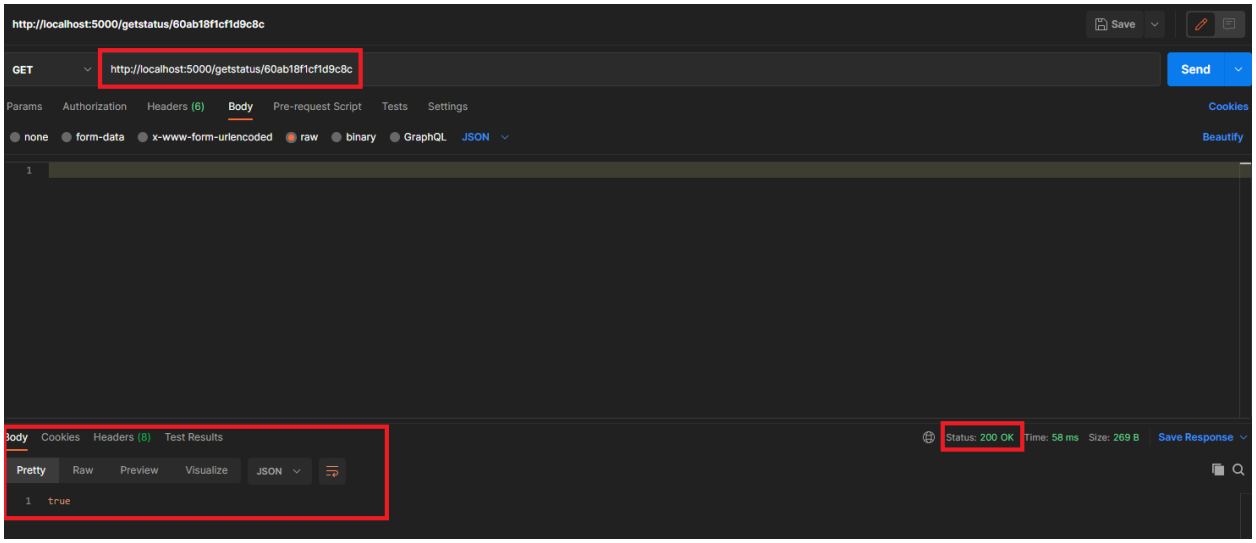


Figure 54. Postman Endpoint Testing

Ensuring that the schemas are being accurately created in the database is extremely important. The purpose of a schema is to structure the data in the database and provide a standard for other endpoints to retrieve or add data to the database. So, if the structure is not what is expected, this can throw off many other functionalities of our web app. To test this, we will be using Postman to send POST requests to our database. Once we do that, we will compare what is posted in the database with our schema

structure that was saved in the models folder. If the schema structure matches what was posted to the database, then we can assume that it is working correctly. If it does not match, we will have to go back and check 2 things, the routing of the endpoint and the export of the schema. Either one of those not functioning could cause several crashes since anything trying to access that document in the database would either try to get something that is not there or try to post something that does not exist.

After following these testing procedures, our backend should be functioning properly. As development progresses, other problems may come up as they often do when programming so it is our job to understand how those issues are coming about and what we can do to prevent them.

7.4.3 Embedded Software Testing

This subsection will discuss the testing procedure used for each of the embedded software designs discussed in a previous section. Each design will be of utter importance to run correctly in order for the product to function as a whole. Should any of the designs not work properly, there is a high chance the product could fail to do what it was intended to do.

7.4.3.1 Temperature Modulation

This design of the embedded software will be used to adjust the temperature of the product to the heat required to keep the selected food warm till pickup. First we ensured that the modulation can be controlled by the requests made by the web application. Once we have ensured the MCU is receiving and acknowledging these requests, we can then choose to test specific temperatures. We can simply use a thermometer or an embedded hardware to check and track the temperature of the box. We will be checking if the temperature modulation correctly communicates with hardware to heat the box to the requested temperature and ensure the product maintains a relatively close temperature throughout the process.

7.4.3.2 Barcode Scanner

This design of the embedded software will be used to unlock or lock the box. The barcode scanner should be able to scan a code and send the data to the MCU and eventually to the web application through the ESP32 WiFi module. This can be tested in a variety of ways. The main functionality we are checking for here is that the barcode scanner is properly scanning the code and requesting the MCU to send the data to the web application. We set up a console log or just manually checked the data that is sent to check if it is equivalent to what was encoded into the scanned code.

7.4.3.3 ESP32 WiFi Module

The ESP module is programmed by the Arduino IDE. We needed to download libraries to get it set up and be able to program with the Arduino IDE. Using the IDE, we set up

HTTP commands to send to the server and database. These requests return a JSON containing all the information of an order. An example of a HTTP GET request was tested below.

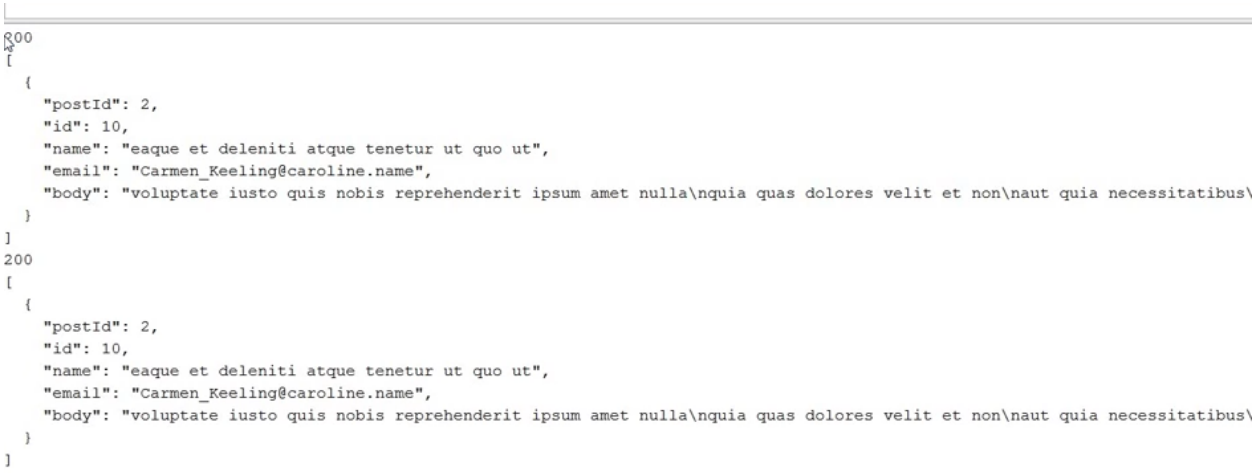


Figure X. HTTP GET test

The ESP32 code behaves similarly to normal Arduino code; there is an initializing phase where the ESP32 connects to the local internet. After this, it goes into a loop until it receives an order from the server. We get orders from the server by making HTTP requests from the ESP32 to the hosted server. The server sends data in the form of JSONs because it is more efficient to interpret. When the ESP32 receives this data, it communicates to the MCU through the Serial Monitor like the example shown above.

8. User Manual

In this section, we will be explaining the operation of the HotBox. We will be breaking it down into two parts, the consumer part as well as the administrator side. But before that we will talk about the process before it gets to that stage. The HotBox is a stationary heating box and should be placed in an area that has a wall outlet to power the box. Once plugged in, the operation of the box will be available. Assuming the connection of the box has been configured to the server and the network of the restaurant, all operations of the box will be accessible to the admin of the restaurant. A list of operations are listed below

- Lock/Unlock the Box
- View whether contents are present in the box
- Send order information to the box
- View temperatures of the box
- Set temperatures of the box.

Once an order is placed at a restaurant with a HotBox present, the order is then prepared and the information is sent and placed in the box by the admin. Once the food is placed inside the box, the admin locks the box to start the heating process and send a notification to the customer. The information of the order is then displayed on the outside of the box in addition with the box number on an LCD screen for consumers to view. The box will be programmed to be on a low power ‘warm’ preset; that way once food is placed inside the box, it will not take a long time to ramp the heat up inside the box. Once food is placed inside the box, it will automatically be locked once the door closes. After this happens, the box will be heated to whatever preset the customer selected when they were placing their order. There are 3 temperature presets that the customer can choose from (Warm - 110F, Warmer - 125F, Hot - 140F)

Once the order information is sent to the box and the box securely locks the food inside the box, the box sends out a notification to the consumer alerting them that their food is ready for pick up through email. Included in the email is a QRcode that is for unlocking the box. The box will heat the food to keep it warm until the consumer comes in and opens the box. In order to unlock the box, the consumer must scan his barcode associated with his order, either on the restaurant app/web page or on his order receipt (email). The box can also be opened by the admin in case of barcode malfunction.. All orders that have been made will be present in the database so one of the states will be available on the screen.

- **Successful Scan**: *Order is ready and inside box #...* the corresponding box will unlock and the consumer will be able to open the box and take their food. Once the order is taken from the box, the order information is sent to a different database with completed orders.

- **Unsuccessful Scan**: *Invalid Scan, Please see admin for more details.* This means that the order is not in the database for current orders for the box. This could mean that the respective order has already been taken out of the box (old order) or it's a invalid receipt. The admin can give the consumer more information regarding the issue if this prompt shows.

This concludes the user manual section for the HotBox.

9. Administrative Content

This section of the document is to cover the milestones and budget management of the group. This section is meant to show the plan we came up with to go about the task of creating the document containing all the research and design plans for the actual project. The section shows how we planned to manage our time ahead of actually writing the document. Also included in the section is our budget and finances for the project. This section will include all the parts that will be purchased for the project and the expected prices for each of those parts.

9.1 Milestone Discussion

The following table is the plan we created during the Divide and Conquer portion of the document. This contains the estimated start and end dates we planned to go about completing each of the tasks. These start and end dates were adjusted based on how they were actually done. The portion following the table will discuss the changes made and why they ended up how they are. The table includes both Senior Design I and Senior Design II tasks that needed to be completed.

Table 33. Initial Project Milestones

Task	Start Date	End Date	Status
Senior Design I			
Form Group	1/10	1/14	Completed
Senior Design Bootcamp	1/21	1/21	Completed
Project Discussion and Decision	1/21	1/24	Completed
Initial Project Document - Divide and Conquer	1/24	1/29	Completed
Final Divide and Conquer Document	1/31	2/12	Completed
Emergency Group Meeting	3/16	3/16	Completed
60 Page Documentation Draft	3/16	4/2	Completed
60 Page Draft Meeting	4/5	4/7	Completed
100 Page Documentation Report	4/5	4/16	Completed
Final Revisions and Proof Readings	4/18	4/27	Completed
Final Document	-	4/27	Completed
Acquire Project Parts	4/28	-	Completed

Senior Design II			
Assemble Prototype	5/2	7/26	Completed
CDR Presentation	-	6/18	Completed
Midterm Demo	-	7/7	Completed
8-Page Committee Paper	-	7/16	Completed
Form Committee Panel	-	7/20	Completed
Final Presentation and Demo	-	7/26	Completed
Committee Meeting	-	7/27	Completed
Website and Deliverables	-	8/3	Completed

While having everything going according to plan would be great, this did not happen for us. We found the semester to be full of obligations and responsibilities, from other classes to work, affecting the progress of our meeting to plan and produce the project documentation.

The largest change was removing an optional task containing a time to have completed 40 pages of the document. This task was originally added to keep us on track. The gap between the final Divide and Conquer and 60 page draft was well over a month, making it hard to know whether we were on track. This task, which was to be completed by the second week of March, was to measure this. However, we found that, due to aforementioned responsibilities, this was not possible and the start date for working towards the 60 page draft was pushed back drastically.

Instead of the above task, we decided to have a meeting when this task was due. During this meeting, we created a table of contents based on the table of contents example given in the documentation guidelines. This was created to be a guide for us, to know what needs to be written and around how many pages should be written for each section. We also discussed each individual's role in completing this document. This included each member was responsible for around 30 pages of the document. The group contained two pairs, the hardware pair and software pair. We discussed that each pair was responsible for writing the sections related to their roles, being the hardware sections and software sections.

The milestones that ended up in the final milestone table were important tasks to be tracked. These tasks were based on due dates for portions of the documents, meetings the Professor had with us, and other tasks we viewed as important to track in order to complete the document on time.

Our group was formed on January 13th, finalized into a group in the class on the 14th. Our group contained members who we already knew from previous classes, adding chemistry and cooperation to those already present. Our group was composed of four computer engineers, however, two were focused on the software side, one was focused on the hardware side, and one was flexible, allowing the group to have a good split of hardware and software focused individuals.

Our group attended the Senior Design bootcamp on January 21st. This meeting focused on giving us a foundation to build off of. This meeting was used to create the first iteration of the milestone table.

The next couple Milestones were done following the bootcamp. During this time, we discussed ideas for the project and decided on one, which was used to write up the first Divide and Conquer documentation. We had trouble coming up with ideas, leaving few options to choose from. The main two ideas we had to decide between was the HotBox project and a secure messenger device. Some members realized that the secure messenger device would be too small to have the required microboard leaving us with the HotBox project as our only real option.

We had a meeting with Dr. Richie about the Divide and Conquer document and had the project approved. We followed the meeting up by creating a House of Quality table for the final Divide and Conquer document.

The next milestone was the 60 page document draft document. As aforementioned, we had major delays on starting this task, reducing the effective time the group had to complete the task. This resulted in an emergency meeting on March 16th. In this meeting, we had an important discussion on everything. First and foremost, we once again went over the expectations of each member, specifically for the software members. Up till this point, it had not been very clear what software would be needed for the project. This was clarified to be a website application to demo the project and code that would adjust what temperature the HotBox should maintain for specific food items. The hardware members also had some discussions on what they should research. We also came up with a goal of writing a page a day. At this point, if every member wrote at least a page a day, we would make the deadline. This would also help with making the entire document seem less daunting and more manageable. We also decided we would have a meeting every week from then on, in order to keep up communication and to keep everyone accountable for what they needed to have done. Without this meeting, we may never have managed, so it was a very important part of our process.

Up till the 60 page submission deadline, we continued to have one to two meetings a week to discuss anything that came up. In general, we discussed specifics for the hardware and software, in order to know what to write about for that week. During this period, some members excelled in the page per day goal while some fell short. By the time the submission came around, although we did not reach 60 pages of original content based on the requirements given, we technically had 61 pages for the

submission. With this submission came a meeting with Dr. Richie to discuss how everything was going.
This meeting was successful. Dr. Richie stated that what we had was good so far. He had a few comments for us, specifically needing introductory paragraphs for sections. He also gave us some much needed pointers on how to go about writing the second half of the document, as multiple of us were finding it difficult to find stuff to write about at this point. We got clarification on a few questions we had for him, including the best way to get in contact with him if needed and exactly what needs to be written.

We held a team meeting following the Dr. Richie one to go over everything discussed and set a plan for the 100 page submission. This meeting included a new page per day goal for each member, based on how much they had written so far and how much they had to write from here on out to reach the end goal of 30 pages per person. We also went over the remaining requirements that had to be written. Dr. Riche had mentioned to make sure we did not hit 120 pages and stop before the requirements were meant. To ensure we avoid this, a list of everything remaining was made that we could cross off as we continued. We also started to finalize the parts we would need so we could start purchasing them earlier rather than later.

The 100 page submission was due over Spring Break. Unfortunately for how the semester worked out, Spring Break ended up being a work week for a lot of us. Due to the workload that was required to be completed during the week, we found little time available to commit to the paper. This resulted in us falling short of the 100 pages by the time it was due, putting us well behind schedule.

We held a meeting the Tuesday after Spring Break to discuss what needs to be done in order to complete the paper in the remaining week. We made the decision to try and finish all the writing by Sunday night. This would give us all of Monday to ensure we had all the requirements but importantly give us plenty of time to format the paper for the final submission. Being we were significantly behind schedule, this was going to require every member all in to get the paper done. Any effort less than the most we could do would leave us behind the 120 page mark by the due date. Procrastination had taken its toll over the project and now was the time to get on it.

The final week plan seemed to be falling apart. The time remaining for writing had passed and barely anything more had been done. This led to a member making an important reminder and request to get to work. This was not something to wait last minute, yet this is exactly what we were doing and it needed to be addressed now. If the last push was ignored, we would have no chance at making the deadline and all the effort spent would not result in what we would want.

The final days before the due date were a grind to say the least. Many members worked all day every day for those final days to catch up on the writing that needed to be done. Slowly but surely though, the pages were made and we ended up just around the page count that was desired. The final day and a half was reserved for formatting the paper and ensuring all the requirements were meant. We did not want to find ourselves having

reached the page goal we wanted but did not have some of the major requirements needed for the document as a whole. After all this was done, the paper was finally submitted and one step of the project was complete.

Senior Design II began with us being delayed a couple weeks. This was due to a member being out of the country and overall needing some time off after the process of developing the SD1 Document. We got started once this member returned to the country.

The first deliverable of the class was the CDR Presentation, to show off what our project would be, how we planned to accomplish it, and our current progress of completing it. Being that we were still very early in the development process, this presentation was mostly a progress report of what we had done so far. Around this time, we concluded we were about 50% completed, although this percentage included the Senior Design I deliverables, we were somewhat behind overall but had plenty of time left to go.

We continued developing the project up till the next deliverable, which was the Midterm Demo. By this time, we were only weeks away from the project needing to be done. The main focus of this demo was to show off what we had completed so far, what we had worked on but not yet completed, and what we were behind on. This led to a meeting where we were advised on anything that might need to be cut. In this meeting, we were confident we could finish everything we had sought to accomplish with the time we had left and continued to work on developing the project.

The final couple weeks prior to the Committee meetings consisted of hard work and stress. The team worked overtime, especially the hardware side, on getting the box developed. At this time, we had configured almost everything for the project but nothing on the physical product had been put together. The software team was focused on adding the final touches to their side and keeping constant communication with the hardware team to ensure the site and endpoints had everything needed for the box to function. The main stress at this time was finding members for our Committee. There was an influx of about 50% more groups over the Summer semester than normal, and with no increase in the professors available, we struggled greatly to find professors for our Committee. We eventually finalized our Committee well over the original due date and only days before the actual meeting.

The final two days consisted of the team working day and night on finishing the project. These days consisted of many problems that needed to be fixed, hindering us from finalizing the project. Fortunately, what felt like neverending issues popping up stopped occurring the final day of development and we were able to complete the development and record our demo. The final day before the meeting was spent recording our presentation and finalizing our 8-page Committee paper. We had issues with the internet and some mistakes were made in the video editing of the presentation, leading us to finish later than we wanted. We were able to get it done within the hour before it needed to be due and received our meeting details within that hour.

Finally came the meeting. This meeting was very successful, as we answered all the questions our Committee had and answered enough questions with our presentation, demo, and 8-page paper that the meeting only needed twenty of the thirty minutes allocated. With this, the team agreed some rest was in order and decided to take the rest of the day to relax.

We met up the next day to discuss the final deliverables for the class, which was creating a website that contained all the deliverables for Senior Design and updating our old document with any changes from the development process in Senior Design II. We spent the final week completing this, making a deadline to have everything done a couple days before the final due date so that everything could be added to the site and give us time for any issues that might end up occurring with the website.

With this, we had completed everything needed for Senior Design and could finally enjoy a nice break from all the hard work we had done over the seven or so months.

9.2 Budget and Finance Discussion

This section of the document will discuss our estimated budget and finances for the project. This project is a personal project and has no sponsors to assist with the expenses. As such, all costs for the project will be coming out of our pockets and will heavily restrict the budget we will have to work with. Early on in the semester we all decided we wanted to try and keep the project to a maximum cost of $400 or $100 per group member. It is possible we will need to spend more and are prepared to so if necessary, but it is our goal to not spend more than that. Along with personal spending, our budget is meant to keep the project affordable. While we do not have initial plans to push the project further than being just a project, we still want to try and develop it around the idea that it could be manufactured, and as such, should be affordable to produce for those that would make use of having it.

The following is a table containing the cost of the components needed to create the HotBox. This the pricing of creating a single box:

Table 34. Budget and Finance

Parts	Quantity	Regular Cost
Lock	**2**	$18.89
Barcode Scanner	**1**	$46.12
Relay	**1**	$7.99
LCD Display	**2**	$10.43
Wifi Module	**3**	$24.97

Transformer	**2**	$34.88
MEGA 2560	**1**	$33.76
Heating Bed	**1**	$20.99
24v to 5 converter	**1**	$10.09
Temperature sensor	**1**	$9.57
24 to 12 converter	**1**	$10.63
Uart converter	**1**	$13.33
Plexiglass	**13”x5”**	$9.99
Foil Insulation	**1**	$12.19
Pack of Pin Headers	**1**	$3.99
Infrared thermometer	**1**	$18.89
Logic Level Shifter	**1**	$15.68
Heavy Duty Plug	**1**	$12.99
TTL UART to usb	**1**	$7.99
Transformer	**2**	$35.99
DC power jack	**1**	$4.50
Pack of breadboard jumper wires	**1**	$13.34
400 Grit sandpaper	**1**	$4.99
40 Grit sandpaper	**1**	$5.19
Fiberwood	**¼ log**	$8.02
Paint, paint tape, screws, caulk, wood, glue, handle	**1 of each**	$69.83
JCLPCB	**2**	$63.15
Total: $542.39		

9.3 Project Tools

This section contains all the tools that were used or will be used in the design and development process of the project.

Discord

Discord is a free voice, video, and text application. Discord offers many options to make it easy to collaborate between ourselves. It offers private message groups for smaller groups and channels that can support larger groups or be used to better organize information in the channel. Discord groups and channels also have voice channels or the ability to call in order to communicate in voice should typing not cover the material that is to be discussed. Discord has been our main source for communication. Early on, we made a group for early discussions on finding members for our group and talking about ideas we had for the project. While we used the group for far longer than we should have, we eventually made a channel for our group. This group contained three text channels, general, hardware, and software. General was used for general discussion about the project. Hardware was focused on the hardware aspects of the project, mainly used by the hardware pair of our group. Software was focused on the software aspects of our project, mainly used by the software pair of our group. These text channels helped keep information separate and organized. Our channel also had a few voice channels that we could use in order to speak to one another for more effective communication, importantly for our weekly meetings. Discord also has the ability to pin messages. This was useful for keeping track of important messages, valuable information, links to our document, etc. Due to the difficulties of meeting up in person during our Senior Design, Discord helped keep strong communication among the members.

GroupMe

GroupMe is a communication application that allows the creation of groups based on a specific topic or classes. GroupMe is commonly used by students for our classes to talk amongst students in the class. GroupMe was used early on to help find members for our group. The application was also used to speak to other students in the class that were not part of our group. This ability was useful for asking questions about other portions of the class, including finding lecture links and keeping track of due dates that we might have forgotten about.

Google Drive

Google Drive is a file storage and synchronization service developed by Google. Google Drive was used for nearly everything related to writing this document. Google Drive was desirable for this document because it allows for easy sharing among members and allows us to easily all work on something at the same time without worry of conflicting with one another. Google Drive contains many tools that were used for the document.

Our project contained multiple subfolders in our Senior Design drive. These folders made it easy to organize different aspects from each other. For example, we had a

folder to hold all our flowcharts to make them easy to find and adjust as needed without them getting mixed with everything else. Google Drive also made it easy to keep older versions and backups of our work. We made viewable only docs for each of our separate submissions, from our initial Divide and Conquer to each of the page milestone submissions. We also kept at actively updated backup in the chance something happened that could result in a catastrophic loss of work.

Google Docs was primarily used for writing the 120 page document. The ability for all of us to easily be able to look over and work on the same document at the same time was beyond helpful. It made collaboration very easy and quickly allowed us to review each other's work and provide feedback during the writing process. While Google Docs does have its issues, specifically with page breaks that sometimes do not make sense, the benefits heavily outweigh any inconveniences it has.

Google Drawings was used to create many of the figures found in the document. Google Drawings provided all the tools we could find wanting for developing these figures. Like all Google Drive tools, Google Drawings made it easy to collaborate and work on the same figure at once, saving time and effort and overall increasing productivity.

Google Sheets were also used for some of our figures and tables. We specifically used Google Sheets to create our House of Qualities table. The House of Qualities table was created during a meeting with all of us collaborating on it at the same time. Google Sheets made it very simple to both design the House of Qualities and simultaneously work on it.

Google Drive and the tools it contains helped immensely with the process of creating this document, saving us time and effort with the ability to easily share and collaborate on the document and its aspects.

Figma

Figma is a vector graphics editor commonly used for designing frontend web app interfaces. Figma makes it easy to share ideas and collaborate on designs. Figma also contains a vast variety of tools to create any design one could imagine. Figma will be used to create a user interface for the web application. This design will be built through collaboration, making sure everyone on the team likes the design of the application. The visual of the design will make coding up the frontend of the application much easier. It will provide a direct reference when attempting to replicate the design over in code.

Visual Studio Code

Visual Studio Code is a freeware source-code editor developed by Microsoft. VSCode is a very powerful editor with many tools available to download and use. VSCode is able to edit many languages. Along with having an option to live share code with teammates, it may prove very useful for collaboration in the software development process. VSCode could potentially be used for all the software development we will do, allowing us to

streamline what we use to develop. At the very least it will likely be the main editor used for the development of the web application.

GitHub

GitHub is an Internet hosting provider for software development and version control. GitHub is used by millions of software developers for their code and many open-source software can be found and used from GitHub, including some of the libraries we will be using in our project. Not only will GitHub be used to access some of these open-source software available to us, we will also use GitHub ourselves for its version control aspect. A vital portion of software development is ensuring changes we make do not break everything. Version control and its ability to create branches for new code will allow us to test changes before pulling them into the main code, and should something break, version control can be used to restore to a point before that happened, although ideally we will avoid this potential issue at all costs. Needless to say, GitHub will be a vital part of our software development for the project, storing the code and allowing us to easily manage how to pull changes.

Git

As mentioned previously, we will be using GitHub to manage and collate all of the code that will be written for the web app. To that end, we will be using Git to control this from the command line. Git is a free and open source distributed version control system designed to work in tandem with GitHub, hence the naming. Git is specifically for version control, it makes it easier for developers working on a project to track what changes have been made to any code so people do not work on the same thing. Git can also be used to deploy Node.js apps to Heroku, which will be its primary function in this project. Beyond that, Git will be mainly used for version control and so everyone always will have the latest version of the project on their local machine. Git is popularly used in the industry, so using it here is a great experience for the future.

9.4 Project Roles

This section covers each member of the group, what they studied and their roles in the project as a whole. This section will also contain a table with a simplified version of the information on each member and their contribution to the team.

Ahmed Kazzoun

Ahmed Kazzoun is a computer engineering student. He was part of the hardware team for the project. Ahmed is the most flexible member of the team, having strong backgrounds in both the hardware and software sides of electrical engineering. He used this experience to be the main communicator between the two teams, being able to comprehend what both sides are doing and effectively merge the two. Along with communication, his main focus will be working in the hardware team to build the functioning physical prototype for the project and work.

Austin Tillotson

Austin Tillotson is a computer engineering student. He was part of the software team for the project. Austin maintained that his strongest skills lied in coding, putting him on the software team. He will focus on creating the embedded software with the software team and focus on the frontend side of the demo web application. Austin also took the role of organizing meetings and creating agendas for the meeting. He wanted to ensure each weekly meeting was focused and discussed what needed to be done that week or in the near future. His focus during the development of the paper was to hold each member accountable for their part in the writing process, keeping track of each member's contribution. He also was the final judge on the formatting of the paper.

Chaitanya Vemuri

Chaitanya Vemuri is a computer engineering student. He was part of the software team for the project. Chaitanya was the strongest member for software development on the team, making him a prime candidate to work on the software team. His role for the project will be to develop the embedded software with the software team. Specifically he will be focused on creating the communication between the hardware and software via signals from the web application. He will also focus on creating the backend for the web application. He showed interest in working on the frontend of the application as well and will help with the frontend should time permit.

Haafiz Shafau

Haafiz Shafau is a computer engineering student. He was part of the hardware team for the project. Haafiz was by far the most experienced member when it came to hardware so he led the charge for the hardware team. Haafiz's role in the project is to design and develop the physical prototype for the project and do the hardware testing for the prototype with the help of the hardware team. He also designed the electronic diagrams and schematics needed for the project design and documentation. Haafiz was also the project manager of the group. He was the one who came up with the project idea and took the lead in developing what the project would require from both the hardware and software.

The following is a table containing the above information of each member in a more simplified environment:

Table 35. Project Roles Table

Ahmed Kazzoun	- Computer Engineer - Hardware team - Communicator between teams - Physical prototype developer
Austin Tillotson	- Computer Engineer - Software team - Meeting organizer - Embedded software developer - Frontend developer
Chaianya Vemuri	- Computer Engineer - Software team - Embedded software developer - Backend developer
Haafiz Shafau	- Computer Engineer - Hardware team - Physical prototype developer - Hardware designer and tester

9.5 References

Adafruit. *“RGB backlight positive LCD 16x2 + extras - black on RGB” Derived:* *https://www.adafruit.com/product/398*

Albatross. *History of C++*. cplusplus: http://www.cplusplus.com/info/history/

Angular. *Introduction to Angular Concepts*. Angular: https://angular.io/guide/architecture#:~:text=Angular%20is%20a%20platform%20and,you%20import%20into%20your%20apps

Arduino. *ATmega640/1280/1281/2560/2561 Datasheet*. Retrieved from Microchip: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf

AWS DynamoDB. *Amazon DynamoDB FAQs.* AWS: https://aws.amazon.com/dynamodb/

BrittLiv. “*Programmable Temperature Controller + Hot Plate.”* Derived: https://www.instructables.com/Programmable-Temperature-Controller-Hot-Plate/

Campbell, Scott. “*Basics of the I2C Communication Protocol.*” Derived: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Campbell, Scott. “*Basics of the SPI Communication Protocol.*” Derived: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol

Campbell, Scott. “*Basics of UART Communication.*” Derived: https://www.circuitbasics.com/basics-uart-communication/

Centers for Disease Control and Prevention. “*Personal and Social Activities.”* Derived: https://www.cdc.gov/coronavirus/2019-ncov/daily-life-coping/personal-social-activities.html

Circuit Basics, “Basics of the SPI Communication Protocol,” Available: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol

CostWay. *“End Loading Insulated Food Pan Carrier Hot and Cold” Derived:* End Loading Insulated Food Pan Carrier Hot and Cold

Creativity. “*Heated Bed 24V Black Parts Heatbed Hot HotBed 3D Printers Part Heat 235mmx235mm Aluminum Plate 3m 3DprinterAccessories.”* Derived: http://www.creativity3dprinter.com/HeatedBed_24VHeatbed_235mmx235mmHeatbed

Dataflair. *Pros and Cons of MongoDB.* Dataflair-Training: https://data-flair.training/blogs/advantages-of-mongodb/

Epec, LLC. "*Polyimide / Kapton® Flexible Heaters."* Derived: https://www.epectec.com/flexible-heaters/polyimide-kapton-heaters.html

Epec, LLC. "*Silicone Rubber Heaters."* Derived: https://www.epectec.com/flexible-heaters/silicone-rubber-heaters.html

Espressif Systems. *ESP32 Datasheet*. Retrieved from Expressif: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

Firebase. *Support and Upgrade Guide*. Firebase: https://firebase.google.com/support/guides

Heroku. *Heroku Dev Center.* Heroku: https://devcenter.heroku.com

IBM Cloud Education (2019, May 9). *Lamp Stack Explained*. IBM: https://www.ibm.com/cloud/learn/lamp-stack-explained

"Interface an I2C LCD with Arduino." Derived: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/

JLCPCB. *"Why JLCPCB?"* Derived: https://jlcpcb.com/aboutUs

MongoDB. *Firebase vs MongoDB.* MongoDB: https://www.mongodb.com/firebase-vs-mongodb#:~:text=MongoDB%20is%20a%20more%20robust,purely%20a%20cloud%20database%20service

MongoDB. *Mern stack*. MongoDB: https://www.mongodb.com/mern-stack

Postman. *Postman Learning Center*. Postman: https://learning.postman.com

Python Institute. *About Python.* Python Institute: https://pythoninstitute.org/what-is-python/

Resendes, Stephanie. "*26 Online Ordering Statistics Every Restaurateur Should Know in 2020."* Derived: https://upserve.com/restaurant-insider/online-ordering-statistics/

Sanruskmv, ProjectHub *"GM65 barcode scanner"* Derived: https://create.arduino.cc/projecthub/sanruskmv/wireless-product-tracking-arduino-mkr-barcode-scanner-3708d9

Section (2020, Jan 23). *A Brief History of C Programming*. Section: https://www.section.io/engineering-education/history-of-c-programming-language/

Selenium Testing Tool. *About Selenium*. Selenium: https://www.selenium.dev

Sparkfun. *2D Barcode Scanner Module -DE2120*. Retrieved from sparkfun: https://www.sparkfun.com/products/16410

SwaggerHub. *Why Swagger*. SwaggerHub: https://swagger.io

Texas Instruments. "*LM75A Digital Temperature Sensor and Thermal Watchdog With Two-Wire Interface"* Derived: https://www.ti.com/lit/ds/symlink/lm75a.pdf

Tom. "*3D printing guides: Everything about heated beds!"* Derived: https://toms3d.org/2014/12/21/3d-printing-guides-everything-about-heated-beds/

Vogt, Mary. *" What Are Flexible Heaters?"* Derived: https://www.loomia.com/blog/flexible-heaters

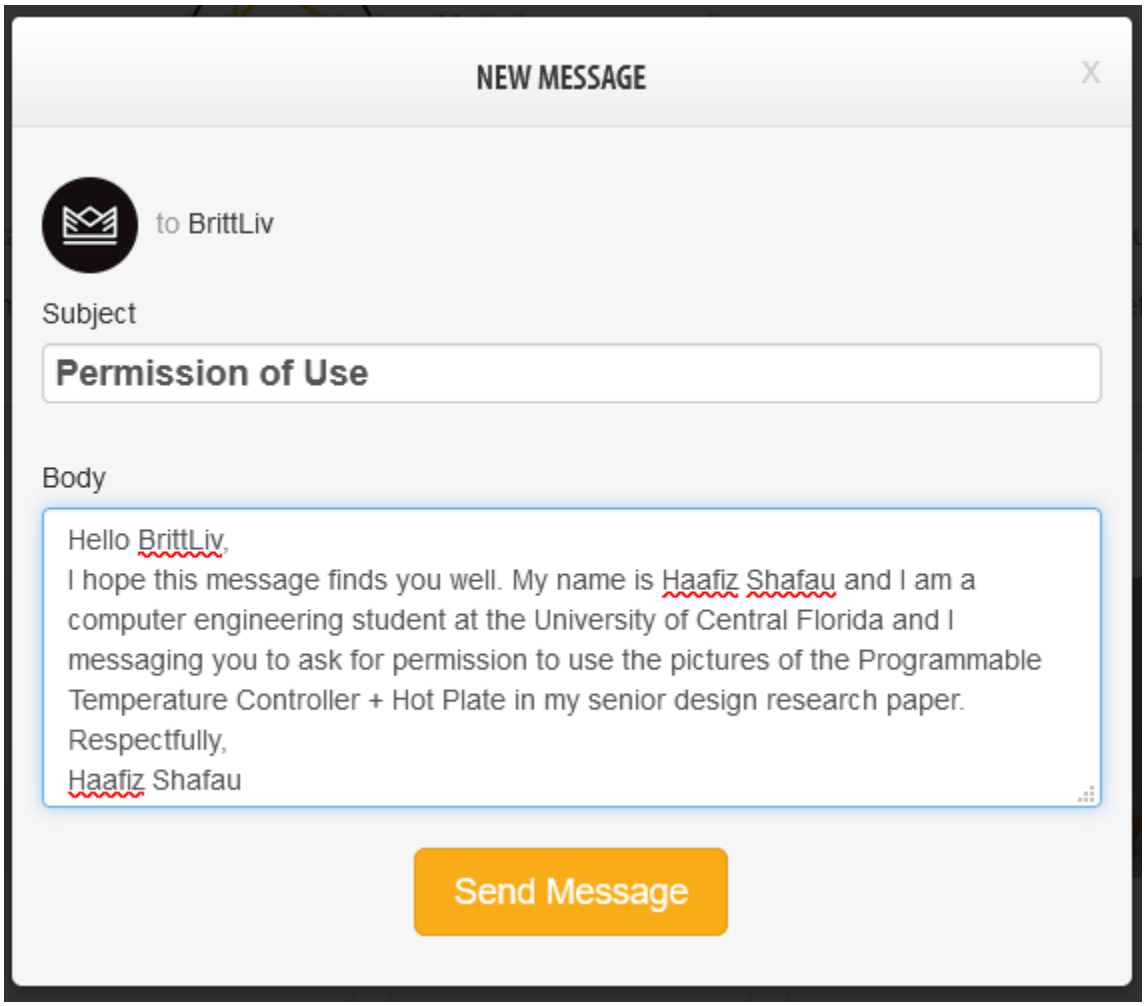
Wasp. *"Barcode Scanners: How Do They Work?"* Derived: https://www.waspbarcode.com/buzz/how-barcode-scanners-work
Relay Circuit Diagram

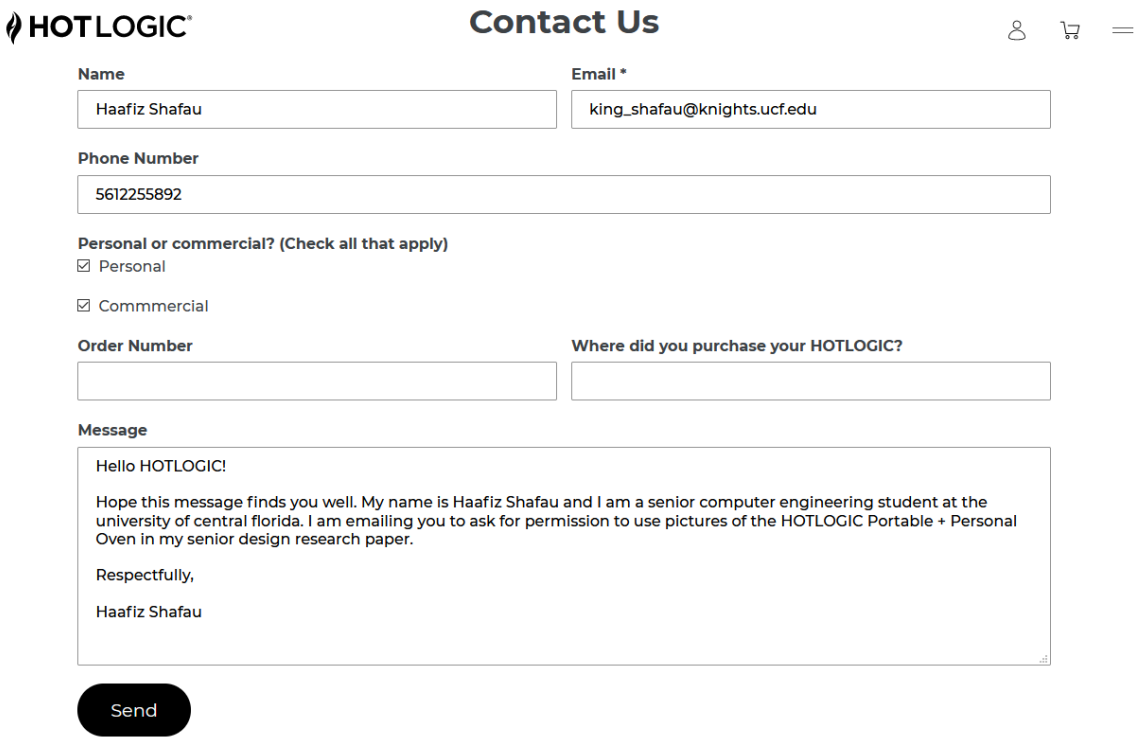
Waveshare. *"Barcode Scanner Module User Manual.* Derived: https://www.waveshare.com/w/upload/d/dd/Barcode_Scanner_Module_Setting_Manual_EN.pdf

WebstaurantStore. *"Cambro EPP260SW110 Cam GoBox"* Derived: https://www.webstaurantstore.com/cambro-epp280sw110-cam-gobox-black-half-size-top-loader-insulated-food-pan-carrier-15-3-8-x-13-x-12-3-8/214EPP280BLK.html

Wilhelm, Alex. "*How COVID-19 accelerated DoorDash's business."* Derived: https://techcrunch.com/2020/11/13/how-covid-19-accelerated-doordashs-business/

9.6 Permission Requests




HOTLOGIC
Contact Us
Name
Haafiz Shafau
Email *
king_shafau@knights.ucf.edu
Phone Number
5612255892
Personal or commercial? (Check all that apply)
Personal
Commmercial
Order Number
Where did you purchase your HOTLOGIC?
Message
Hello HOTLOGIC!
Hope this message finds you well. My name is Haafiz Shafau and I am a senior computer engineering student at the university of central florida. I am emailing you to ask for permission to use pictures of the HOTLOGIC Portable + Personal Oven in my senior design research paper.
Respectfully,
Haafiz Shafau
Send

Permission of Use

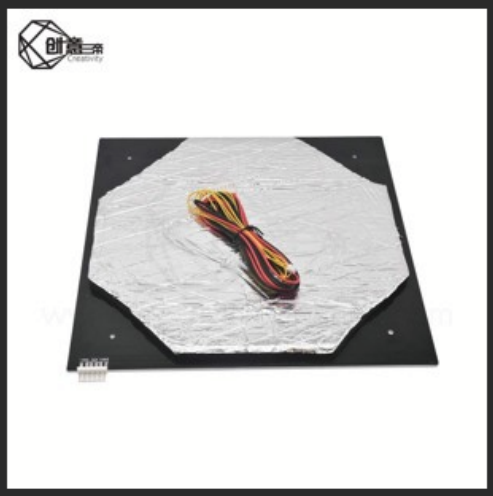
HS

Haafiz Shafau <king_shafau@Knights.ucf.edu>
12:11 AM

To: sales@Creativity3dprinter.com

Hello Creativity3D!

Hope this message finds you well. My name is Haafiz Shafau and I am a senior computer engineering student at the university of Central Florida. I am emailing you to ask for permission to use pictures of the Heated Bed 24V Black Parts Heatbed Hot HotBed 3D Printers Part Heat 235mmx235mm Aluminum Plate 3m 3DprinterAccessories in my senior design research paper.



Respectfully,
Haafiz Shafau

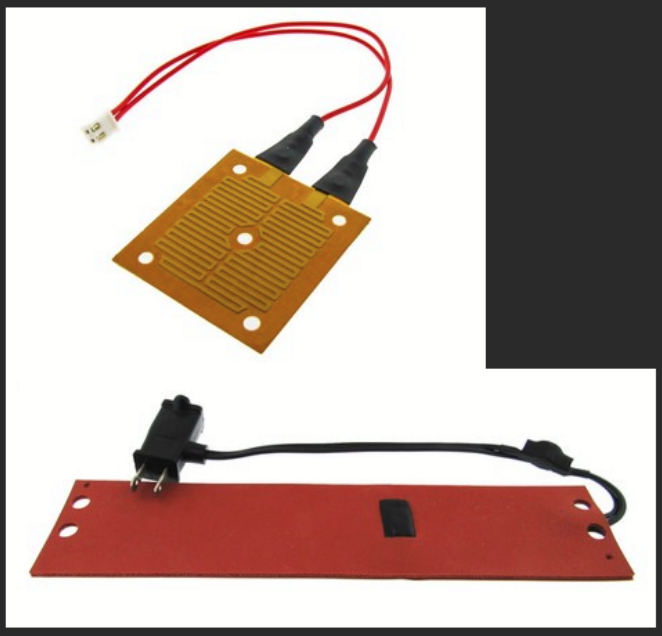
Website Information Request

HS **Haafiz Shafau <king_shafau@Knights.ucf.edu>**
12:24 AM

To: sales@epectec.com

Hello Epec!

Hope this message finds you well. My name is Haafiz Shafau and I am a senior computer engineering student at the University of Central Florida. I am emailing you to ask for permission to use pictures of the Polyimide / Kapton® Flexible Heater and Silicone Rubber Heater in my senior design research paper.



Respectfully,
Haafiz Shafau

Permission of Picture Use

HS

Haafiz Shafau <king_shafau@Knights.ucf.edu>
1:07 AM

To: support@sparkfun.com

Hello Sparkfun!

Hope this message finds you well. My name is Haafiz Shafau and I am a senior computer engineering student at the University of Central Florida. I am emailing you to ask for permission to use pictures of the 2D Barcode Scanner Breakout (SPX-16441) in my senior design research paper.

Respectfully,
Haafiz Shafau

To H help@webstaurantstore.com Bcc

Cc

Permission of Use

Hello WebstaurantStore!

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the Cambro EPP260SW110 Cam GoBox and the specific features that this product shows.

Thanks, Ahmed

Are the auto-complete suggestions above helpful? Yes No

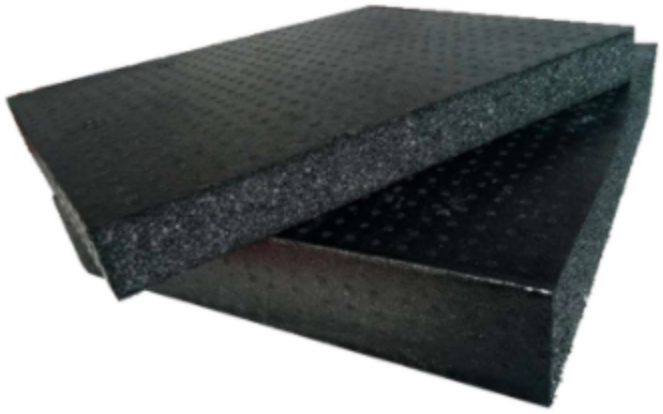
To R reception@bpf.co.uk Bcc

Cc

Permission of Use

Hello bpf.co.uk!

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the Expanded Polypropylene (EPP) and the specific features that this product shows.



Thanks, Ahmed

CONTACT US

Our support team is here to help! Drop us a note below and we'll get back with you to help answer or resolve your issue.

NAME *

Ahmed | Kazzoun

First Name *Last Name*

EMAIL *

akazzoun2000@knights.ucf.edu

MESSAGE *

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the BoxLock Smart Lock and the specific features that this product shows.

Thanks, Ahmed

› SEND

To S service@costway.com Bcc

Cc

Permission of Use

Hello CostWay!

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the End Loading Insulated Food Pan Carrier Hot and Cold and the specific features that this product shows.

Thanks, Ahmed

Contact Form

Fields marked with an * are required

Name *

Ahmed Kazzoun

Email *

akazzoun2000@knights.ucf.edu

Message *

Hello LastMinuteEngineers!

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the Passive Infrared (PIR) Sensor and the specific features that this product shows.

Thanks, Ahmed

What is thirteen minus 6? *

7

Send

Contact

PLEASE NOTE: Adafruit **does not** have a retail store, **orders cannot be picked up. NO VISITORS ARE PERMITTED**. Our factory is **not available** for visits.

If you need technical support for your purchase from Adafruit, please visit the Adafruit customer support forums where we have dedicated technical support engineers that will be able to assist you quickly. While we answer all questions in the customer support forums, we **do not** provide one-on-one email or telephone consulting. Our engineers will answer questions, troubleshoot, handle replacements and assist you in the Adafruit customer support forums.

Full Name: Ahmed Kazzoun *

Email Address: akazzoun2000@knights.ucf.edu *

Confirm E-Mail: akazzoun2000@knights.ucf.edu *

Category: Press / media inquires

Please supply a detailed description of your press/media inquiry in the form below, or email press@adafruit.com call: 646-248-7822 directly for urgent matters.

At this time the Adafruit team and our founder Limor "Ladyada" Fried is booking new speaking engagements in exchange for donations to help NYC during the COVID-19 pandemic.

Message

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the Adafruit Accessories Lock-style Solenoid and its schematic on this page and the specific features that this product shows.
https://forums.adafruit.com/viewtopic.php?t=58311
https://www.adafruit.com/product/1512

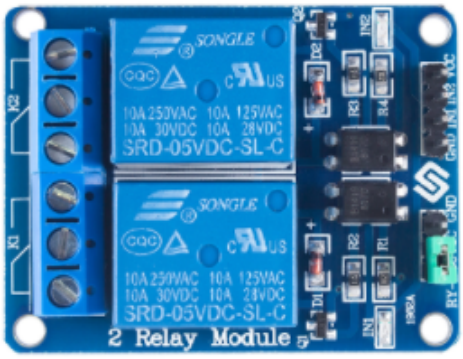
Thanks, Ahmed

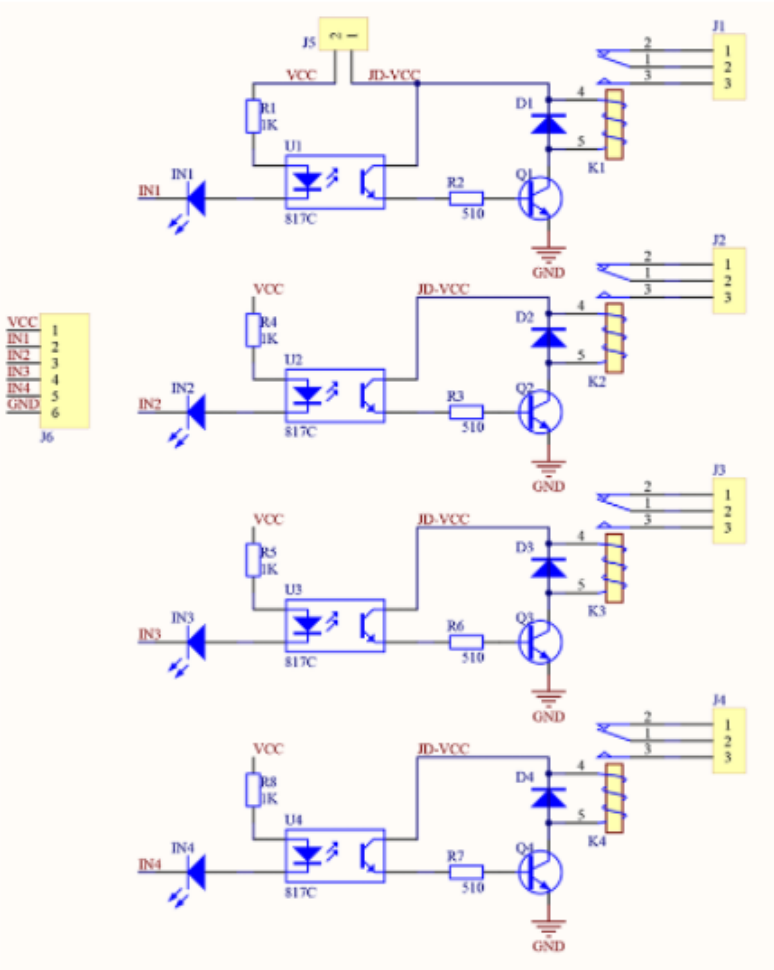
Send Message

Permission of Use

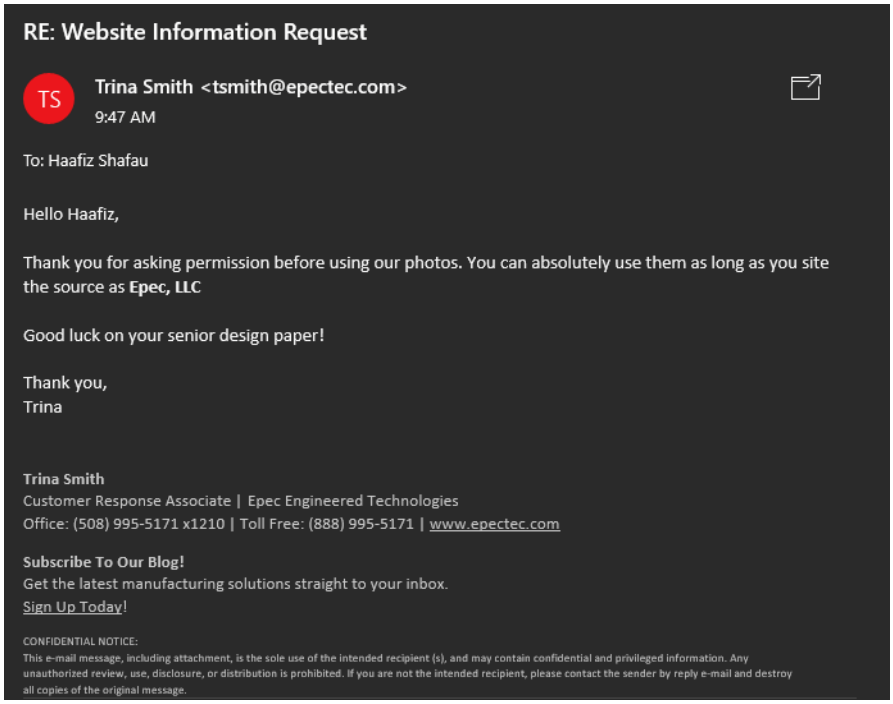
Hello SunFounder!

I hope this email finds you. My name is Ahmed Kazzoun and I am a senior Computer Engineering Student at the University of Central Florida. I am emailing you to ask for permission to use the picture of the 2 Channel 5V Relay Module and the electronic diagram of it and the specific features that this product shows.





Thanks, Ahmed



RE: Website Information Request

TS

Trina Smith <tsmith@epectec.com>
9:47 AM

To: Haafiz Shafau

Hello Haafiz,

Thank you for asking permission before using our photos. You can absolutely use them as long as you site the source as **Epec, LLC**

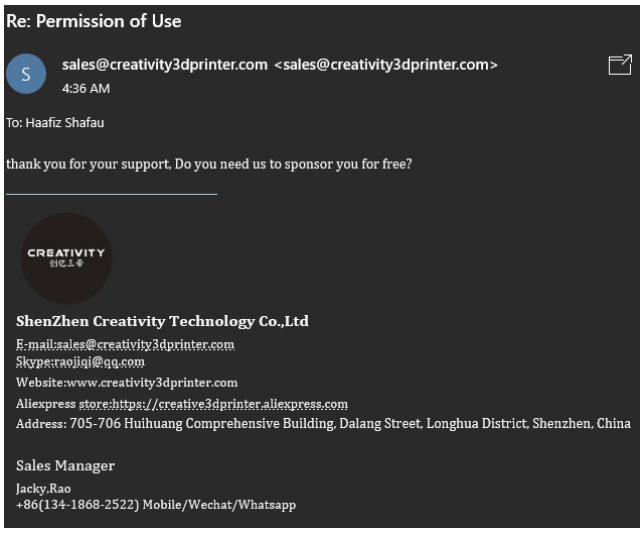
Good luck on your senior design paper!

Thank you,
Trina

Trina Smith
Customer Response Associate | Epec Engineered Technologies
Office: (508) 995-5171 x1210 | Toll Free: (888) 995-5171 | www.epectec.com

Subscribe To Our Blog!
Get the latest manufacturing solutions straight to your inbox.
Sign Up Today!

CONFIDENTIAL NOTICE:
This e-mail message, including attachment, is the sole use of the intended recipient (s), and may contain confidential and privileged information. Any unauthorized review, use, disclosure, or distribution is prohibited. If you are not the intended recipient, please contact the sender by reply e-mail and destroy all copies of the original message.



Re: Permission of Use

S

sales@creativity3dprinter.com <sales@creativity3dprinter.com>
4:36 AM

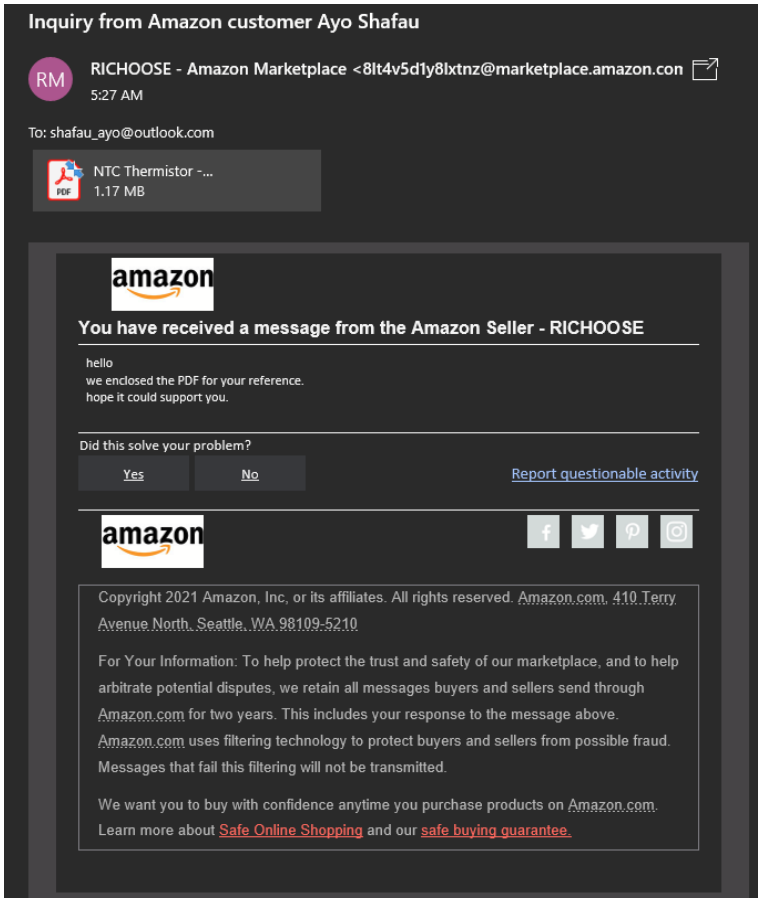
To: Haafiz Shafau

thank you for your support, Do you need us to sponsor you for free?

CREATIVITY

ShenZhen Creativity Technology Co.,Ltd
E-mail:sales@creativity3dprinter.com
Skype:raojiqi@qq.com
Website:www.creativity3dprinter.com
Aliexpress store:https://creative3dprinter.aliexpress.com
Address: 705-706 Huihuang Comprehensive Building, Dalang Street, Longhua District, Shenzhen, China

Sales Manager
Jacky,Rao
+86(134-1868-2522) Mobile/Wechat/Whatsapp


Inquiry from Amazon customer Ayo Shafau
RM
RICHOOSE - Amazon Marketplace <8lt4v5d1y8lxtnz@marketplace.amazon.con
5:27 AM
To: shafau_ayo@outlook.com
NTC Thermistor -...
1.17 MB
amazon
You have received a message from the Amazon Seller - RICHOOSE
hello
we enclosed the PDF for your reference.
hope it could support you.
Did this solve your problem?
Yes
No
Report questionable activity
amazon
Copyright 2021 Amazon, Inc, or its affiliates. All rights reserved. Amazon.com, 410 Terry Avenue North, Seattle, WA 98109-5210
For Your Information: To help protect the trust and safety of our marketplace, and to help arbitrate potential disputes, we retain all messages buyers and sellers send through Amazon.com for two years. This includes your response to the message above. Amazon.com uses filtering technology to protect buyers and sellers from possible fraud. Messages that fail this filtering will not be transmitted.
We want you to buy with confidence anytime you purchase products on Amazon.com. Learn more about Safe Online Shopping and our safe buying guarantee.

Permission of Use

HS **Haafiz Shafau <king_shafau@Knights.ucf.edu>**
10:45 AM

To: contact@simax3dp.com

Hello SIMAX3D!

Hope this message finds you well. My name is Haafiz Shafau and I am a senior computer engineering student at the University of Central Florida. I am emailing you to ask for permission to use pictures of the 310x310mm CR10 Aluminum Heated Bed 24V, Hot Bed, Heating Plate Platform, B for CR10, CR 10S Pro, CR-X with 1Meter Cable in my senior design research paper.

Respectfully,

Haafiz Shafau