Smart Electric Food Heater HotBox

Ahmed Kazzoun, Chaitanya Vemuri, Austin Tillotson, Haafiz Shafau

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The HotBox uses various different modern technologies to solve a problem in the food industry. The project is designed to house different kinds of warm-hot ready-made food and keep them properly warm for customer health and enjoyment. Once an order is in the system and is placed in the vacant box, the box securely locks the food inside. A notification (email) is then sent to the customer, letting them know that their food is warming and ready for pick up. The customer will also receive a QR code/barcode in the notification for order validation. The customer must scan their QR code/barcode in order to unlock the box and take their ready warm food. Majority of food that is ordered online and picked up in stores is cold and unenjoyable to the customer. And due to the COVID-19 pandemic, online ordering was a necessity to a lot of restaurants to reduce in person traffic. However, due to this, a lot of restaurants had increased traffic due to more online orders. With our system, we wish to provide customers with warm safe food and limit in store traffic with our notification system so you know when your food is hot and ready to be eaten.**

I. Introduction

For our senior design project, we will be developing a smart food heating system named 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 restaurants 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. This report details how we designed, created and configured our project in order to meet the requirements and specifications we wish to meet in order to meet our end goal design. The hardware and software requirements and specifications requirements will be detailed and expounded on.

For the paper, we divided it into two main sections, main research and the implementation and process of creating the design itself. In the first half of the paper, we outline all the main research that went into the project. The research portion of the paper consists of the possible hardware components that would make up the system, the software that was necessary for the hardware components and the system to work together successfully. The criteria and structure of the project as well as the specifications that we put in place to make sure our system met them in order for it to perform successfully are also covered. In the second half of the paper, we go over the implementation and design process of the HotBox. This section holds the main hardware design and the final software implementation. We fully discuss the design process of the box and the integration of all the final chosen hardware components necessary such as the Microcontroller. We also discuss the frontend and backend creation and utilization in the project and explain how we connected all these components to yield us a working system.

II. System Concept

The goal of our smart heating box is to heat customers' food until they arrive to pick it up. This is because of the common complaint that food gets cold after being created while waiting for the customer to pick it up. Our device is a wooden box capable of holding different types of foods. When a customer orders food, a confirmation email will be sent to them with a confirmation and a QRcode to scan whenever they go to the store and pick it up. This barcode unlocks the box and they can now open the door and leave with their food. The box's temperature can be set by the customer while placing an order. The temperature of the box will not go above this temperature by using a temperature sensor.

A. Member Responsibilities

Each person has a key part to create our project that laid the foundation. In the table below, the group members' names and what their roles are in the project are shown. After careful planning and setting milestones for our project, we set out to fulfill the project's specifications. Although we each have individual tasks, we always will help each other whenever we need it.

Group Member	Main Responsibility
Ahmed Kazzoun	Embedded Hardware
Chaitanya Vemuri	Web Application (Backend)
Austin Tillotson	Web Application (Frontend)
Haafiz Shafau	Housing and Hardware

Table 1: Group Members and Their Responsibilities

B. Block Diagram

Although each of us is responsible for different parts of the project, we must all have a good understanding of the project. This is important so everyone can have a vision of how the project should be like.. The block diagram for the project is included below.

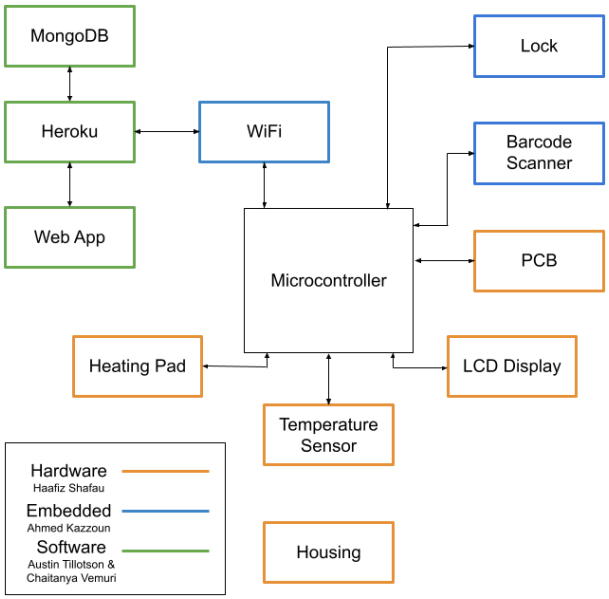

Fig. 1. HotBox Block Diagram

C. 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.

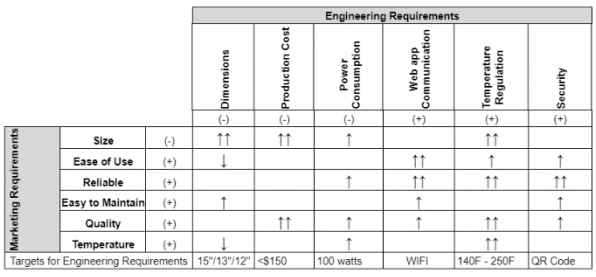

Fig. 1. HotBox House of Quality

D. Design 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 to get everything done. Everything for this project comes out of our pockets so we also must keep the product affordable. We also 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.

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.

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.

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.

III. Hardware Components

In this section, we look at and study previous projects, relevant parts, and different technologies that are available for our project. At the start of this project, we needed to do some significant research on these parts to find the best combination of efficient and compatible parts.

A. WI-FI Module

The ESP32 is the Wifi Module we decided to use. It is an upgrade from another ESP8266 Wi-Fi module. The ESP32 supports bluetooth and 2.4 GHz bandwidth. The purpose of a WIFI card is to communicate with the restaurant's web application to send notifications to the users that notify them when the food is ready to be heated. It is designed for performance, versatility, and reliability in a wide array of applications. There are 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.

We will use the ESP32 to communicate with our server. We coded the ESP32 module to make certain HTTP requests to the server which affects the MongoDB. Because of the discussed security features the ESP32 provides, the security of the WiFi is satisfactory.

B. 4-Channel Relay Module

A 4 Channel Relay Module 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. 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.

C. Barcode Scanner

A barcode scanner will be used to scan barcodes from customers when they are ready to pick up their food from the restaurant. Barcode scanners record and translate barcodes that have a striped image with alphanumeric digits underneath it. Scanners can read different types of barcodes that provide various types of properties and functionalities. The barcode scanner that we will be using will send information to a database through a wired connection to the ESP32 module. Barcodes will be sent out in emails whenever a user places an order. We made a collective decision that using QR codes is the most efficient and most reliable way to scan barcodes. When a user scans a barcode, the box will determine if it is the correct barcode for that order. If it is not a correct barcode, the box will not unlock. If the barcode is correct, the box will unlock and the customer will be able to leave with their food.

D. Solenoid Lock

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. The size of this lock is small enough for a cabinet which makes it perfect for our project. This solenoid in particular is nice and strong, and has a slug with a slanted cut and a mounting bracket. It is basically an electronic lock, designed for a basic cabinet, safe, or door. Normally the lock is active so you cannot 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 does not stick out anymore and the door can be opened.

The lock is only engaged when the Box gets new order information loaded in from the server and ESP32. It stays locked and secure until a customer is ready to pick up their order by scanning the barcode that will be sent through an email. When it unlocks, the customer is free to leave with their order.

E. Temperature Sensor

The temperature sensor is used inside the interior of the box to measure the temperature of the box at a given time. There are a couple aspects to be noted that this sensor accomplishes. First, it ensures the box does not reach temperatures that will overheat or cook the food, as it is meant only to keep the food warm. Second, it ensures the box does not exceed the temperature that the customer set when they made an order. In the application, a customer can choose a preset of what kind of heating they want for their food. These presets have a set temperature that ensures the food will stay warm but not overheat**.** If the temperature detected from the sensor exceeds the required amount, the heating pad will turn off and wait until it reaches a certain temperature until it turns back on.

F. Heating Pad

The heating pad is the main component of our project. This is where the heat will resonate from. It is also where the food will rest on top of. So by convention, the food should warm up really quickly. The specific heating plate

we chose supports what we need and requires 24V to operate.

The Creativity Heated Bed features an upgraded, full gold processed hot bed plate. This one is insulated with cotton, which improves heat conduction and heat-resistance performance. It features high sensitivity, rapid response, good stability, and high reliability.

G. Power System

Due to the nature of our project, we must carefully look at the nature and design of the hardware components we use. The power system must be able to power all the devices and components present in and around the box. Each component has varying voltage ratings and current consumption. The microcontroller and the components connected to it operate at a 5V DC, however, the solenoid lock operates at 12V DC. In terms of current, multiple components will not be able to just be powered by our arduino, due to the max current of the arduino being 500mA over USB or 800mA through the Wall Wart power. This means some components will require an external source in order for them to properly function inside the box. Failure to properly connect components to the proper source ratings and requirements can damage the sensors and can cause our system to fail as a whole. All these components with varying requirements must be powered from a single wall outlet cable for the ease of the user. This means we will have to take our raw 120V AC wall voltage and split it into different voltages that are ideal for the components in the system. The 120V AC wall voltage will first be converted to 24V DC 20A using an AC-DC transformer. This 24V DC voltage will be used to power the heating bed. The 24V DC source will then be converted into 12V DC 1A for the solenoid lock, and 5V DC 10A for the microcontroller and the rest of the components.

H. LCD Display

The LCD display is going to be used in our project to display the Box Number of the current box. This is important because, when a customer orders from the website, they get an email with their barcode and the exact box number that currently has their order. This is to differentiate if there are multiple boxes in a restaurant. The LCD display will also show when the box is unlocked and stopped heating when a customer scans their barcode.

IV. Hardware Design

This section will contain all the design choices we made for the HotBox. We will discuss the housing design of the box, our PCB design, and all the embedded parts that were used to create the box.

A. 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 relies 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 so 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.

Part	Length (in)	Width (in)	Height (in)	Volume (in^3)
Outer Box	14”	14”	14”	
Interior	13”	14”	12”	
Electronics Compartment	14.5”	14.5”	2.5”	
Transformer Compartment	14”	5-⅜”	7-¼”	
Door	13-⅛”	½”	11”	

Table.2. Box Dimensions

B. 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.

For the custom PCB, we designed the schematic on EasyEDA and used JLPCB for the manufacturing and SMT assembly. We opted in having not only the microcontroller on board, but the wifi module as well which significantly reduced our final project size. We designed the custom PCB to include parts for each individual component connected to the microcontroller and the wifi module as well.

C. Wireless Communication

For communication to other devices, the backend of 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. The ESP32 will be run together with the ATMega328 and will communicate with each other using UART. 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. For the supply voltage, we have a choice between 3.3V DC and 5V DC for the ESP32.

D. LCD Display

An LCD display will be present on the front of the box where users can view the order information of the box, allowing a user to find their respective box. The display will communicate with the ATMega328 to display the corresponding order numbers respective of the box’s current order. The display will communicate to the ATMega via I2C similarly to the temperature sensor, meaning only 4 pins are required for use.

E. Relay Module

For components that have no smart features and/or no ways of directly communicating with the microcontroller, a relay is necessary. For the relay module, we opt in using a 4 channel 5V relay module that allows us to connect components to the relay and use our microcontroller to send a 5V signal to turn it on or off. In our design we connect the heating pad and solenoid lock to the relay and wire the correct voltages together with the components to ensure proper functionality.

F. Barcode Scanner

For customer order validation we decided to use a barcode scanner to scan in the codes the customer receives when an order is ready to be picked up. The barcode we opted in using was the Waveshare Barcode Scanner Module. This barcode scanner is able to read various common 1D/2D barcodes such as a barcode, QR code, etc. And due to its small form factor, it fits into our system perfectly. The module features UART and USB as its two main communication interfaces. The module requires a 5V DC source to operate, however, the logic level is at 3.3V. Normally this would be a problem because the ATMega328 operates at a 5V logic level, but 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.

V. Software Design

This section will cover the software design choices and overall flow of our project. This project contained embedded software for the physical project and frontend/backend software for the demo web application. Understanding how all the software communicates and functions is vital for having a full understanding of the project. This section will begin by covering the embedded software of the project and end with the highest written code, the demo web application.

A. Microcontroller software design

Our ATmega2560 uses the Arduino Software IDE. It is a great piece of software which allows users to write code and upload it straight to the device through USB. To assist in the development process, Arduino includes many libraries which assists in the development of the required embedded functionality. Another benefit of the Arduino IDE is that the serial data being sent from the Arduino board is done directly within the IDE. This is a huge help when dealing with the UART protocol.

The code is constructed in a way where it gets initialized once it gets plugged into the power and then runs a loop. Inside the loop, there are several infinite loops that define the states the box is in. For example, after the board gets initialized, the board sits in a loop until an order from the ESP32 comes through and loads the box. The code also controls the main components of the box like the Relay switch which controls the lock and the heating pad.

B. ESP32 Module Software Design

The ESP module is also programmed by the Arduino IDE. We needed to download libraries to get it set up and be able to program with the Arduino IDE. The reason we chose to do this is because, as discussed above, the Arduino software is very helpful when using multiple unique libraries compared to the other software applications out there.

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.

C. Web Application

The webapp design was chosen to demonstrate the functionality of the project as if the HotBox was being used in a restaurant setting. The webapp project contains just under 200 commits with hundreds of lines of JavaScript code. The app is made up of three main portions of code; the frontend, backend, and database.

D. Frontend

The frontend, designed using React, contains Status and Order pages for the user to use. The Status page implements a status table displaying the status of each box in the system and the order that is or was last contained in each box. This page also contains admin controls to add, delete, or lock a box, which can only be accessed using an admin pin. The Order page contains an order form designed to demonstrate how it would function in a restaurant setting. When completed, the user will receive an email with their order's QR code which is used to unlock the box. The frontend communicates all this information to or from the database using the backend API endpoints via fetch requests.

E. Backend

The backend was designed using NodeJS and Express, communicating with the frontend and the database by using all 8 of the API endpoints we created. Some major examples of endpoints include "GiveOrder", "AddBox", and "LockBox". The "GiveOrder" endpoint handles creating an order and sending that order to an available box. Along with this, it also sends an email to the customer that contains their order number, the box it is stored in, and a QR code that will unlock the corresponding box. To create the QR code, we use the npm package "qrcode" which provides a qr code generator. The process occurs as follows, once the "GiveOrder" endpoints is called, we randomly generate an order number, convert it to a string, and then convert it to a data URL. This allows us to send it as an attachment in our email which is promptly sent once the endpoint is called. An example of what the QR code would look like is shown in the following image:

To send the email to a customer, we use the "nodemailer" npm package which allows us to use an email account and programmatically send emails to anyone. As mentioned earlier, we add the QR code as an attachment so users can open the box with it.

Another major endpoint we used was the "AddBox" endpoint. This endpoint takes no input, but updates the database with a new document that is linked to a specific box. The BoxID field is a randomly generated hex value that is used for the backend to reference specific boxes. Once the new document, or "boxModel", is created, we save it to the database and return all the values in that document. On the website, this is simply shown as the "Add Box" button after unlocking admin privileges.

Another endpoint that we use in the backend is the "LockBox" endpoint. As mentioned in the previous paragraph that details the frontend, this endpoint is only in use once the admin code is inputted by a user. This was a late addition to the web application, but we deemed it a necessary one. Essentially, the endpoint takes a box number and finds that box in the database. Once it does, it replaces the value of the "Empty" field to false, therefore updating the table shown on the status page and triggering the embedded code to lock the box..

For the remaining endpoints, they are all rather small and only do single tasks, so we will describe them all in this paragraph. The "GetOrder" endpoint is used to take an existing order from the database and send it to a box. This is mainly for the embedded side of the programming. The "GetStatus" endpoint is used to return the status of the box, as in returning the value of the Empty field of a specific box. It takes the BoxID from the URL parameters and searches the database for that matching BoxID. Once it finds a match, it returns the value of the Empty field for that document. The purpose of the "DeleteBox" endpoint is to delete a box from the database by taking the BoxNumber from the URL and searching the database with the same method as the GetStatus endpoint. The "BoxStatus" endpoint updates the Empty field for a specific box. This is used for the table of boxes on the Status page of the website. The "BoxStatusEmbedded" has mainly the same functionality, but is used for communication between the embedded programming and the database.

F. Database

For the database, we opted to use MongoDB as it is a NoSQL database, along with the fact that all the authors had prior experience with it. In our database, we have two collections, the boxes collection and the counters collection. The boxes collection stores all of the information of our boxes in documents while the counters collection is used for incrementing the "BoxNumber" field in the boxes collection. For each document in the boxes collection, we have a structure that they must all follow. When the "AddBox" endpoint is called, an example of the document created in the database is shown in the following image:

The schema is as follows: the "Empty" field that shows if that box has anything in it represented by a bool, the "Temperature" the box is set to which defaults to 0 when creating a box, the "BoxID" field which is a randomly generated hex value that is used for the backend to reference each box, and the "BoxNumber" field that is used for the frontend to reference each box. The "BoxNumber" field is automatically incremented using the "mongoose-sequence" npm package by creating the aforementioned counters collection to keep track of the most recent box number. To interact with the database, we use the "mongoose" npm package which is a MongoDB object modeling tool that works in an asynchronous environment.

VI. Conclusion

The main goals of the HotBox are to offer solutions for customers picking up cold food, reducing traffic and waiting times for customers and delivery drivers picking up food, and adding a contactless option for picking up food during the pandemic. To achieve this, we had to put everything we have learned over our undergraduate careers to the test and demonstrate mastery of these design concepts to produce a functional project that achieves the goals we set out. We created a multi-tier product that can take orders from a customer, direct an employee to which box the food should be set in, and allow the customer to pick up their food in a contactless manner from a secure box, all while ensuring the food is warm when they retrieve it.

Acknowledgement

The authors wish to acknowledge the assistance and support of Dr. Richie, for the amount of assistance guiding and educating us through our project.

The authors also wish to acknowledge the support of Dr. Wei, who was always available and willing to answer any questions or concerns we had during Senior Design as a whole.

The authors want to extend admiration and congratulations to all the groups involved in this 2021 Senior Design. Despite Covid-19, online classes, and the general state the world is in from it, these groups have pushed through these hardships to produce impressive work. It has been an honor to work alongside and be part of this group of future engineers.

We offer our thanks to the professors that were willing to take time out of their busy schedules to review the project we have dedicated so much time and effort towards designing and developing over these last six months.

References

[1] 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

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

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

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

[5] 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

[6] Arduino. ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet. Retrieved from Micorchip: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

[7] Wasp. "Barcode Scanners: How Do They Work?" Derived: https://www.waspbarcode.com/buzz/how-barcode-scanners-work

The Authors

Chaitanya Vemuri is a 21-year old Computer Engineering student graduating in the beginning of August. Chaitanya's career goals include working as a Software Engineer in the aerospace industry with an intent to get his masters degree within the next 5 years.

Austin Tillotson is a 23-year old Computer Engineering student. Austin will graduate in Fall 2021 due to a remaining elective to take. Austin's career goal is to work as a Software Engineer/Developer. Austin plans to get a Masters but does not know what to Master in yet.

Haafiz Shafau is a 21-year-old Computer Engineering student graduating in the beginning of August. He plans on finding a career in the computer hardware industry. He plans on obtaining his masters in the near future.

Ahmed Kazzoun is a 21-year-old Computer Engineering student graduating in the beginning of August. Ahmed's career goals include getting in the space industry as a Software Engineer. He intends to get his masters at a later date.