Digitizing the Restaurant with Network-Enabled Smart Tables

By
Andrew Chen (andrew6)
Eric Ong (eong3)
Can Zhou (czhou34)

Final Report for ECE 445, Senior Design, Fall 2020
TA: Sophie Liu

December 9, 2020
Group 3

Abstract

The primary goal of our design is to pose solutions to challenges brought by new restaurant regulations due to the COVID-19 pandemic. We aim to propose a more contact-free seating process with tables that adapt to customer seating preferences and a monitoring system for restaurant employees to determine if seats are vacant, occupied, or in need of cleaning. We also include a mechanism for contactless payment directly with the dining table that leverages our occupancy monitoring system. Ultimately, this minimizes contact during the dining experience such that the sole point of employee to patron contact is when food is served to the customer.

Table of Contents

1. Introduction 1

2. Design Details and Procedure 2

2.1. Height Adjustment Module ... 3

2.2. Customer Presence Module ... 6

2.3. Table Network Module ... 8

2.4. Payment Module ... 12

3. Verification 14

3.1. Table Adjustment Module ... 14

3.2. Customer Presence Module ... 15

3.3. Table Network Module ... 15

3.4. Payment Module ... 16

4. Costs 17

4.1. Labor ... 17

4.2. Components ... 17

4.3. Total ... 18

5. Conclusions 19

5.1. Summary ... 19

5.2. Ethics ... 19

5.3. Future Work ... 20

6. References 21

Appendix 22

A. Requirements and Validation Table per Project Module ... 22

B. Finalized Prototype ... 25

C. Restaurant Dashboard ... 26

D. Payment ... 27

E. Schematics ... 28

F. Board Design ... 29

G. Resistance Values Collected for Custom FSR ... 30

H. Network Topology ... 31

1. Introduction

Since the onset of COVID-19, there has been a shift in the restaurant industry to support more contactless service due to new regulations that require fewer personnel in-house and tighter occupancy caps. Many restaurants, however, still lack contact-free interaction in almost every stage of the typical dining experience including: initial seating, ordering, food delivery, and payment [1], [2]. Restaurants unable to adapt to this new contactless norm have suffered as a consequence: more than 26,000 restaurants have closed since this past July [3]. The purpose of this project is to showcase potential solutions to minimize contact during these interactions.

We propose three modular improvements to dining tables that encourage minimal contact between customers and staff in the restaurant setting. Firstly, we have designed a table system that includes an automatic height adjustment mechanism so restaurants may serve customers with a greater variety of physical needs while retaining a low furniture count. Secondly, we constructed a table network system that monitors occupancy status of all seating in the restaurant which will cut down the number of staff needed to directly monitor customers and also be useful for analytics in the future. Finally, we offer a payment method directly integrated into the table, further reducing the number of interactions between waiters and patrons. Our design culminated in a prototype unit that demonstrates these technologies.

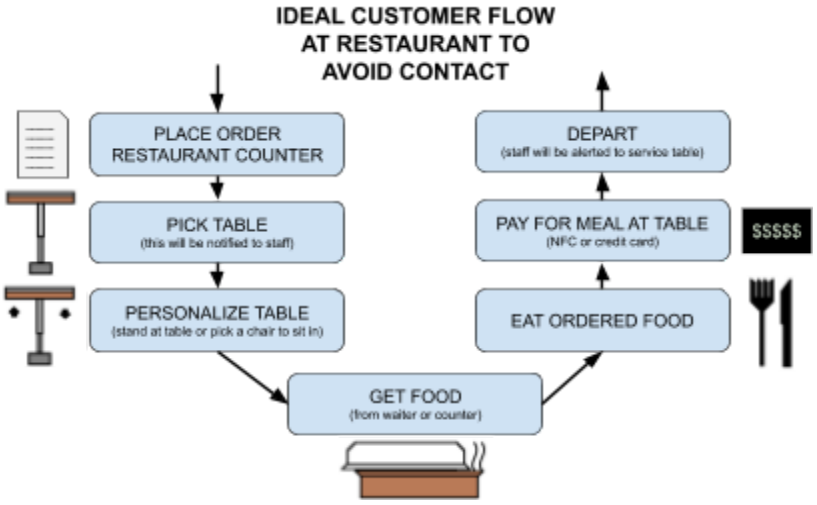

Fig. 1: Ideal Diner Interaction with Table System

2. Design Details and Procedure

Our design can be broken down into four main functional groups, or modules. We had a group of components dedicated to adjusting the table height, a group used to determine the presence of a customer at the table, a network component used to connect to a central computer owned by the restaurant running our dashboard software application, and a group of components that facilitate table-based monetary transactions. These modules and the parts that make up each of them is shown in Figure 2.

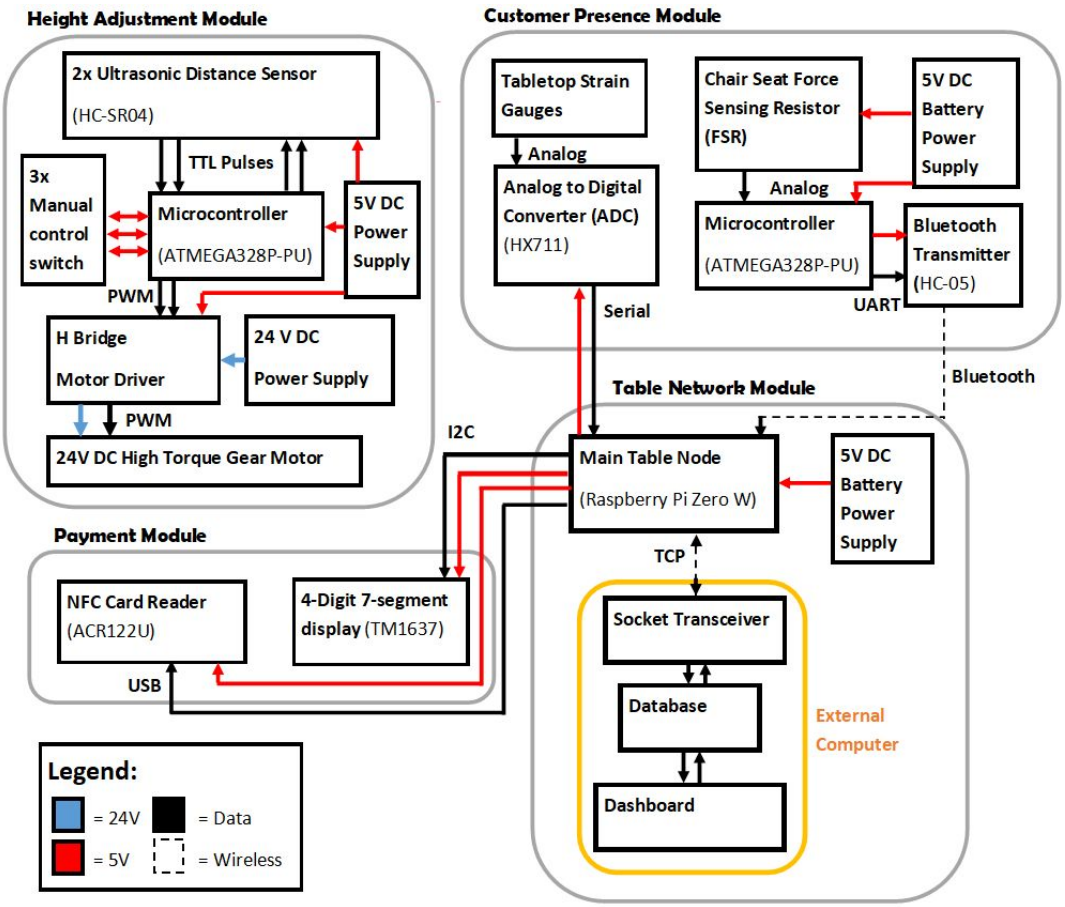

Fig. 2: Table System Block Diagram

For the sake of organization and bookkeeping, we decided to reorganize our groups from our design document into ones that better fit the functionality of each of the components we used. Notably, we have powered separate modules with their own dedicated power sources, and have considered the main node, our table, to be part of the networking group rather than belonging to its own individual group.

2.1. Height Adjustment Module

The prototype we built adjusts its height with assistance from two ultrasonic sensors pointed at the floor on opposite sides of the tabletop. A microcontroller, which is also situated on the top portion of our table, is employed to make decisions with this data, and sends PWM signals to a motor driver circuit in the base of our table. The motor driver circuit uses an h-bridge to route voltage to the terminals of a gear motor that drives the mechanical height adjustment of our table. For safety purposes, we included button switches to provide manual control to the height adjustment of our table. We designed a PCB to connect the sensors, motor components, and safety features of the Height Adjustment Module together. When designing this PCB, we used the public SparkFun repository for our microcontroller library [4] and the public Eagle libraries archive for our linear regulator library [5]. The schematics of this board is included in Appendix E, and the final design of our board is provided in Appendix F. Due to the physical distance between our components, and to encourage more efficient verification of individual hardware, we used screw terminal blocks to connect parts to the table module PCB. More specific interactions between our hardware for the Table Adjustment Module are detailed in Table 1.

Table 1: Table Adjustment Module Interconnects

Connection From	Connection To	Purpose
9V Battery	Linear Regulator	To provide sufficient voltage for conversion to a stable, continuous 5V power stream for the rest of the components in this module
Linear Regulator	Table Adjustment Module Board	To route power to each of the components involved in this module
Table Adjustment Module Board	Microcontroller	To read data from the attached sensors and make a decision on what PWM signal to output that would turn the motor driving the height of the table
Table Adjustment Module Board	Front Ultrasonic Distance Sensor	To determine the height of customers, if any, sitting at the table
Table Adjustment Module Board	Rear Ultrasonic Distance Sensor	To determine the height of the table itself
H-bridge Motor Driver	Gear Motor	To drive the rotation of the lead screw that supports the nut that in turn supports the entire hollow tabletop compartment of our prototype

Table 1: Table Adjustment Module Interconnects (continued)		
Connection From	Connection To	Purpose
Table Adjustment Module Board	Latching Enable Button Switch	To provide a safety mechanism that disables automatic operation of the table
Table Adjustment Module Board	Momentary Up Button Switch	To provide additional support to our safety mechanism for this module, allowing the table to have an option to manually ascend
Table Adjustment Module Board	Momentary Down Button Switch	To provide additional support to our safety mechanism for this module, allowing the table to have an option to manually descend

Our model measures the distance between the tabletop and any customer seating underneath it with an ultrasonic sensor on the user's side of the table. Our table adjusts downwards such that this ultrasonic distance sensor reaches a height of eight inches from the legs of any customer sitting at the table if it is physically possible to do so, determined by the other distance sensor. Our height adjustment has a tolerance of one inch, so the table will halt with a distance between seven and nine inches. We use the sensor on the opposite side of the table to keep track of the height of the table itself, which prevents the table from adjusting upwards from its maximum height or downwards from its minimum height. When the height adjustment to eight inches is not possible, we adjust as far downwards as possible. When no customer seating is detected under the table, our prototype adjusts to a standing height.

To build a prototype to showcase our technology developed for this project, we consulted the ECEB Machine Shop, interfacing with Greggory Bennett to design our table and Jordan Spezia to build and fit the mechanical parts of our prototype. During early development of our project, we had planned to use a linear actuator to perform the height adjustment operation of our table, but upon recommendation and discussion with the shop we instead moved towards a lead-screw system operated by a gear motor. This was to ensure that the height adjustment portion that also acted as the support beam for the table would not be too powerful and tear the tabletop in half when raised.

To go into further detail, the table was separated into two parts by this system, the bottom portion containing the motor and the top portion that moves up and down based on the position of a large nut driven by the rotation of the lead screw. This provided us with a safer option that would put less stress on the metal used for our table, though linear actuators could potentially be a preferable option for tables with more supporting legs [6]. Figure 3 details the

design of the mechanical component of our Table Adjustment Module. A picture of the final build is provided in Appendix B.

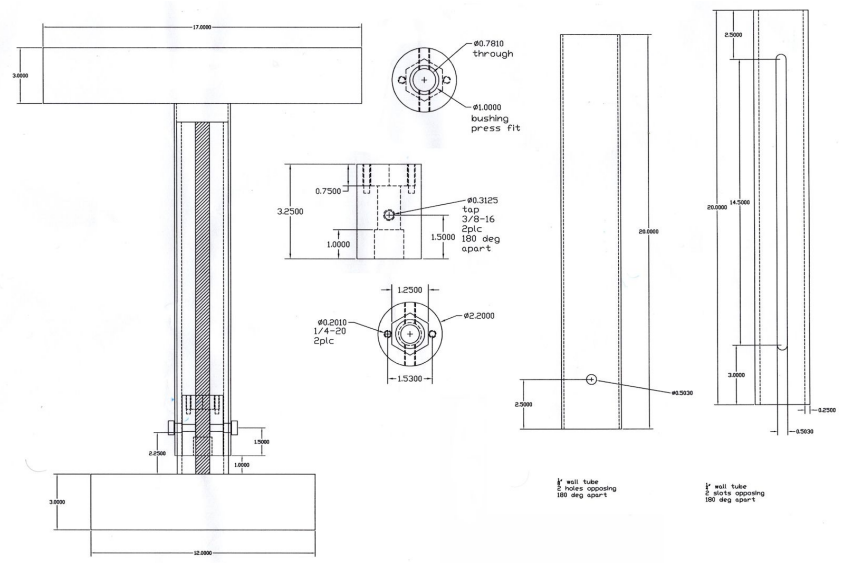

Fig. 3: Diagrams of the Physical Prototype from the ECEB Machine Shop [6]

We had originally planned to measure and compare the different voltages to the rotational speeds of our motor, however due to time constraints and limited access to laboratory equipment we were unable to record these measurements. Despite this, the voltage that we sent to our motor provided adequate speed per our specifications all the while causing little to no motion intrusion on the table, thus allowing for a smooth dining experience for the customer.

When designing our prototype, we considered the balance between how modular and integrated we wanted our functional modules to be. During this process, we considered the height adjustment attribute of our table in relation to the network aspect of it. We determined that restaurant staff would not be interested in how a table is configured relative to more commercially beneficial information that could be used for analytics, such as the amount customers spend in a table regularly, the demand for picking one table as opposed to other vacant tables, and the total number of occupied seats at a given time during the day. With this in mind, we opted to keep table height adjustment segregated from the network-enabled portion of our design, though it would be simple to integrate the two using serial communication from the table module PCB we designed.

Due to our partnership with the ECEB Machine Shop, much of the placement of our components while designing our project was predetermined by where sections were provided.

For example, during discussion of the parts integral to our design, we explained the necessity of connecting components from the base of our table to the circuit contained in the top of the table. Specifically, we needed to send PWM signals from our microcontroller in the upper box of our prototype to the H-bridge circuit in the bottom box, which would reroute a PWM signal to the motor held in the same location. Rather than run wires from a hole in the top of the table to a hole in the bottom of the table, we collaboratively pursued an option to thread wires through an internal notch in the shaft of our prototype, though this had limited space. Due to the placement of these wires, we had to plan the placement of our designed PCB such that it facilitated movement in this fragile configuration. Similarly, spatial limitations resulted in the decision to use the rear ultrasonic sensor.

We had also initially planned to use a motor encoder to determine the current height of the table since it would be stored internally in the table and would be minimally susceptible to external tampering. However, due to the physical constraints of our prototype, we were unable to route a sufficient amount of wires from the top to the bottom of our table model. As such, we had to improvise and therefore opted to instead use a second ultrasonic sensor to determine the current height of the table, instead. By doing this, we saved development time adapting to a new sensor while accomplishing the same task, albeit externally instead of contained within the metal shell of our model. Though these spatial limitations forced us to make certain design decisions, the constrained locations for our components also allowed us to be decisive with the layout of our electronics and organize the components in a more intuitive way.

2.2. Customer Presence Module

This determines the vacancy status of our model by detecting customers sitting at the table with a force sensing resistor and objects on the tabletop with load cells. More specific interactions between our hardware for the Customer Presence Module is detailed in Table 2.

Table 2: Customer Presence Module Hardware Interconnects

Connection From	Connection To	Purpose
9V Battery	Linear Regulator	To provide sufficient voltage for conversion to a stable, continuous 5V power stream for the rest of the components in this module
Linear Regulator	Chair Module Board	To route power to the chair module parts

Table 2: Customer Presence Module Hardware Interconnects		
Connection From	Connection To	Purpose
Chair Module Board	Microcontroller	To read and interpret data from the FSR
Chair Module Board	Bluetooth Transmitter	To relay the information from the microcontroller in the chair module to the Raspberry Pi Zero W in the Table Network Module
Raspberry Pi Zero W (Table Network Module)	Analog to Digital Converter (ADC)	To provide a reference voltage to the ADC and power both the strain gauges and the ADC. This will also ultimately receive the digital output from the load cells.
Strain Gauges	Analog to Digital Converter (ADC)	Send raw analog readings from the load cells, and to provide power to the load cells from the ADC, which is powered by the Raspberry Pi Zero W
Analog to Digital Converter (ADC)	Raspberry Pi Zero W (Table Network Module)	Send converted digital data to the Raspberry Pi Zero W for operations in the Table Network Module

We designed a PCB for the Chair Module to organize the connection between the FSR, Chair Module microcontroller, bluetooth transmitter, and provided battery power. The schematic for this board is included in Appendix E, and the final board design is as shown in Appendix F.

In order to detect customers sitting at a table we built our own force sensing resistor (shortened as "FSR") to act as a pressure plate. When force is applied on the resistor, the resistance measured across it is decreased; We use this to detect a customer sitting down via the increased voltage difference through the resistor. Our custom FSR was constructed by combining two copper plates, formed out of copper tape, and separating them with two pieces of semi-conducting foam, all enclosed in PVC and gaffer's tape for comfort and safety. The two copper ends get closer when a force is applied, decreasing resistance between the leads and thus allowing us to The internal construction of this resistor is as shown in Figure 4. We decided to use a custom FSR due to the limited selection of budget-friendly weight detection options online. We could have used a system of high weight strain gauges, but this would be intrusive to the user as these units are made of metal and require specific mounted placement. We sought a more modular option that could be used without custom seating. For the purpose of future reference, we will refer to the chair-based components of the Customer Presence Module as the "Chair Module".

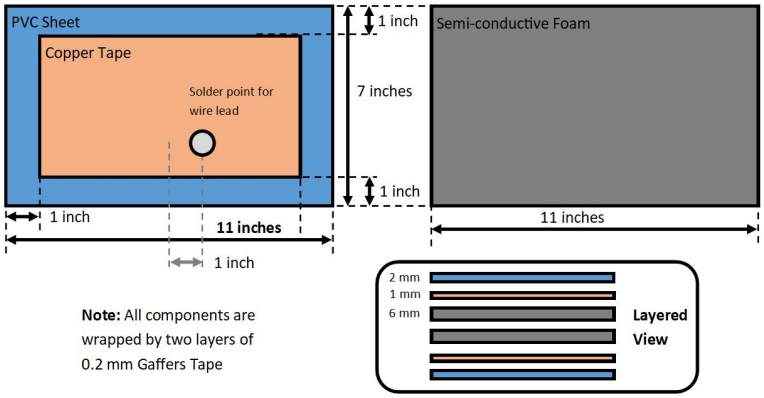

Fig. 4: Force Sensing Resistor Internals

Communication from the Chair Module to the Table Network Module was sent over Bluetooth, utilizing the HC-05 Bluetooth module as a sender and the Raspberry Pi Zero W as the receiver. We picked the HC-05 because of its low cost and known compatibility with our microcontroller, and we chose to send the data via Bluetooth since it is a reliable form of communication at a low range and uses a relatively low amount of power. The Bluetooth module is wired up to a microcontroller which in turn is then connected to the FSR to detect customers. The microcontroller continuously polls for the status of the FSR and sends a ‘0’ over Bluetooth if it determines the chair it is placed on is unoccupied. When sufficient force is applied downwards to the FSR from a customer sitting down on it, the microcontroller detects this increased voltage and declares the seat as occupied, sending a ‘1’ over Bluetooth for the duration that voltage is determined to be greater than a set threshold.

We also used load cells to measure the weight on the tabletop surface. "Load cells" are also known as "strain gauges", and these terms will be used interchangeably in this report. We used the load cells to determine if there was an excess amount of weight on the table, and this allowed us to detect whether a table needs a waitress to service it by cleaning its surface. We wired the load cells to an Hx711 analog-to-digital converter (this will be referred to as an "ADC" going forwards) and sent the converted information to our Table Network Module. To connect the ADC to our Table Network Module we used the serial pins of the module's Raspberry Pi Zero W. For future reference, the Raspberry Pi Zero W will also be referred to as the "RPi Zero W". We initially had four load cells but were only able to accurately measure weight with two, which we deduced was due to faulty internal wiring.

2.3. Table Network Module

There are two methods of wireless communication used in this project. In particular, we communicate wirelessly from our chair module to the RPi Zero W in the table via Bluetooth, and from the RPi Zero W to our dashboard running in our external computer via TCP/IP. For our demonstration, we used a personally owned laptop, but the dashboard may run on any laptop or desktop computer that can connect to the same local area network as the RPi Zero W's used in each table.

The Table Network Module contains both distinct hardware and software components. In terms of hardware, the Table Network Module consists of our RPi Zero W, paired with its power source as shown in Table 3, which is connected to an external computer running our dashboard software. The component diagram for the table is shown in Figure 5.

Table 3: Table Network Module Hardware Interconnects

Connection From	Connection To	Purpose
9V Battery	Linear Regulator	To provide sufficient voltage for conversion to a stable, continuous 5V power stream for the rest of the components in this module
Linear Regulator	RPi Zero W	To route power to each of the components involved in this module

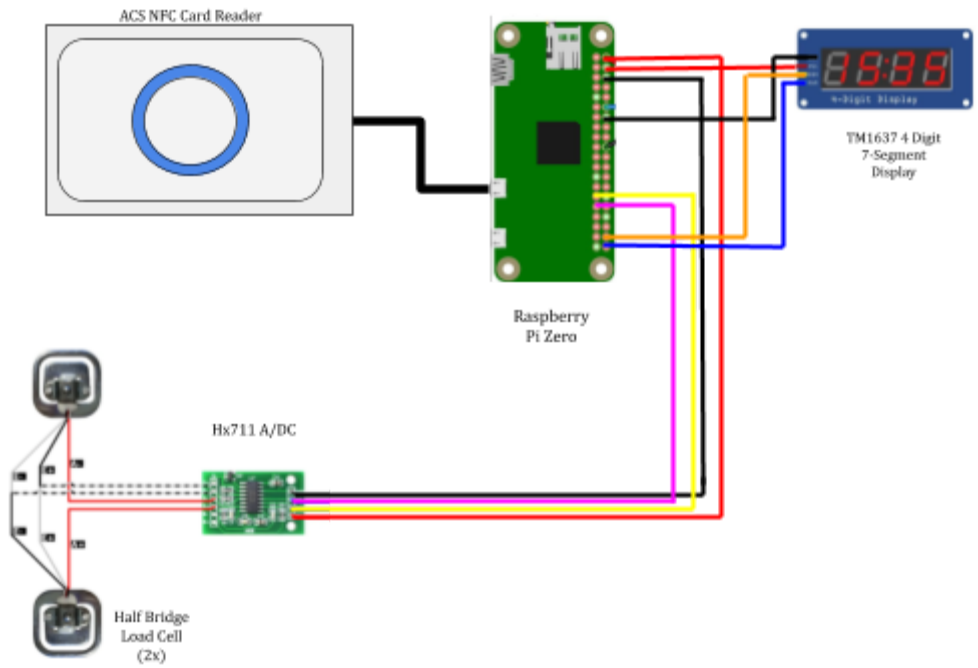

Fig. 5: Circuit Diagram for Internal Network-enabled Table Components

The RPi Zero W reads in data from the FSR via Bluetooth and from the load cell via serial communication. The RPi Zero W takes in this data, makes a decision for table state, and then forwards that data to the dashboard over UDP. This process will be handled by the Python program running on each RPi Zero W shown in Figure 7. The current table status is then decided by the logic included in Figure 6.

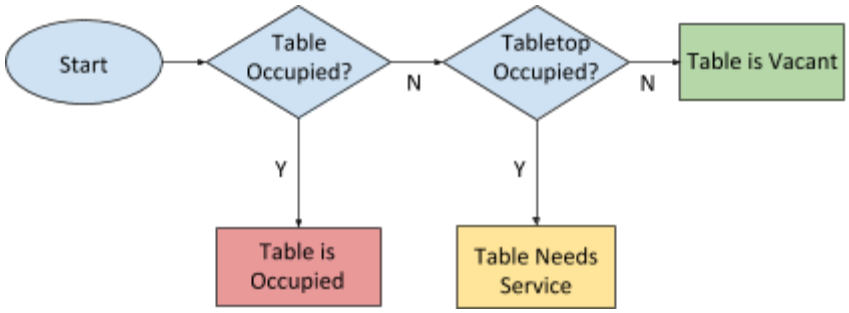

Fig. 6: Table Status Decision Tree

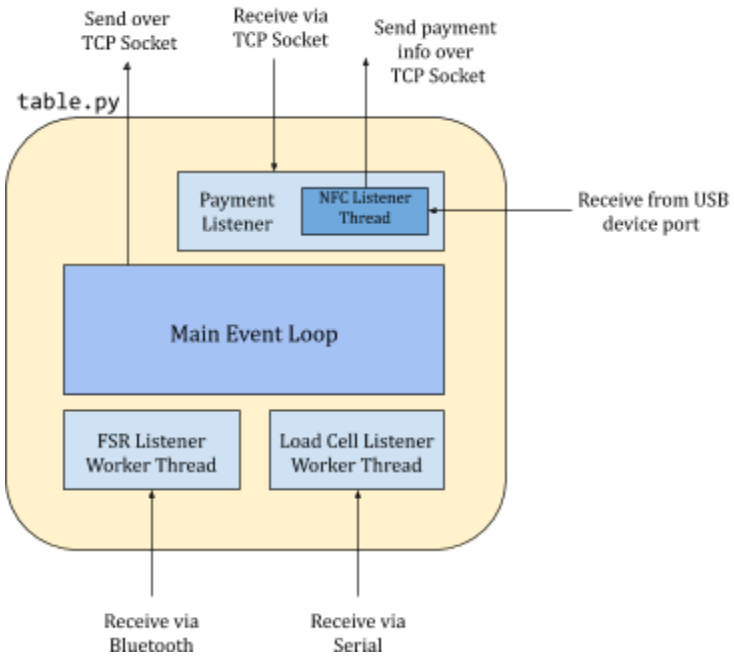

Fig. 7: Software running on Raspberry Pi (on each physical Table)

Table-to-dashboard communication in turn is simply over TCP/IP from the RPi Zero W to whatever computer the restaurant staff have running the dashboard. We initially considered using Zigbee and using a single XBee coordinator for the dashboard, XBee routers for the tables, and XBee edges for the seats. This would have in theory worked as well, but Zigbee modules are a lot more expensive than Bluetooth and so we decided to go with the latter. For the table-to-dashboard communication we also decided to go with a socket-based approach, meaning all these devices would need to be on the same network. At first we attempted to use an ad-hoc network but were running into issues getting the RPi Zero W to connect to this network (hosted on one of our group member's computers) despite being on the same channel and frequency. Next we tried using the UIUC network, but eventually settled on using a mobile hotspot. The reason for this was there were issues connecting all devices to the UIUC network in ECEB, but fortunately using a mobile hotspot allowed for us to have the dashboard talk to the table with no issues. All message passing between devices were then done over the network with Python using socket.io since it is a well-known, reliable socket communication library.

The dashboard is the main way dining staff will interface with our overall design. The dashboard is a software application that may run on any browser-enabled computer that is able to connect to the same network as the RPi Zero W table nodes. The dashboard software architecture is pictured in Figure 8. The dashboard we created for our demonstration was designed to be flexible with different restaurant configurations. Therefore, on setup, it is blank so users may upload their own restaurant floor plan onto it to aid with different seating arrangements. This

also allows restaurants to easily adapt to changes in their layout. A picture of the dashboard GUI is provided in Appendix C. The table node shown in the picture changes color accordingly with the status of the table. The database will store the status of each table and will be updated in real-time at a rate of approximately one update per three seconds.

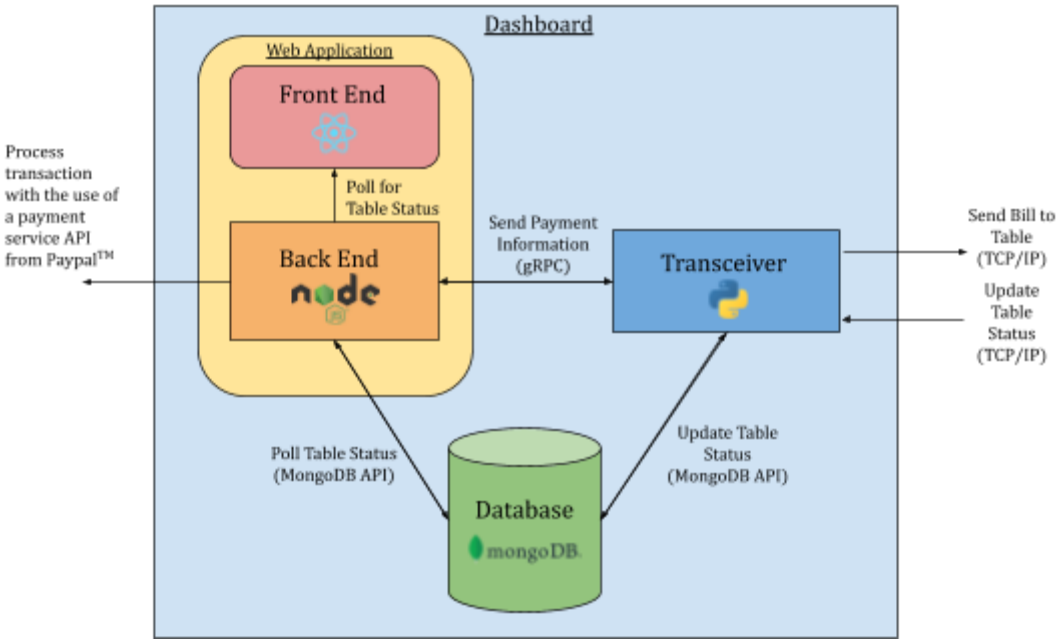

Fig. 8: Dashboard Software running on Restaurant Staff's Computer

All incoming and outgoing communication with the table RPis is handled by a multi-threaded Python application, which we dubbed as the "Transceiver". The messages updating table statuses are stored in the MongoDB database also running on the same computer. Finally, the web application will have a GUI for the dining staff to monitor each table's status which will be queried from the database.

Our dashboard uses MERN stack since it allows us to create dynamic web pages and is also an industry-standard methodology. React.js is a powerful frontend framework developed by Facebook that enables us to move objects and update their appearances on the web page all without ever having to refresh the page. MongoDB is a NoSQL document-based database that allows for inserting various data-types without being restricted to a single schema. This was very helpful in the beginning when we were not quite sure what exact data and data formats we would be storing.

2.4. Payment Module

In order for users to send bills to the table independently, the user needs an interface to interact with. The interface we used in the Table Network Module allows restaurant employees to contactlessly bill customers. On the patron end of this transaction, we have outfitted our prototype with an NFC Card Reader as an example payment method and a four-digit seven segment display to inform customers when and how much they need to pay. The connections to these components are detailed in Table 4. Visuals for the Paypal API dashboard and hex display are provided in Appendix D.

Table 4: Payment Module Hardware Interconnects

Connection From	Connection To	Purpose
NFC Card Reader	RPi Zero W (Table Network Module)	Powered by the RPi Zero W, we use the NFC card reader to relay NFC card information to our Table Network Module to showcase
RPi Zero W (Table Network Module)	4-digit 7-segment Display	Powered by the RPi Zero W, we send numbers to display on our panel to facilitate payment from the customer side

In order to send a bill to a table, the restaurant staff would input the dollar amount in the dashboard's GUI and have this value be sent over to the table. This would create a bill with the help of Paypal's payment API, and would require the customer's payment data to complete a transaction. In our model, this request is made first through gRPC from the web application backend to the transceiver, and the transceiver forwards the request to the desired table. When the table receives the bill request it lights up the hex display connected to the RPi Zero W and polls for customer payment information from the NFC reader, both of which are shown previously in the table circuit diagram in Figure 5. A customer will then pay with his/her card and the card information will follow the reverse path to respond to the dashboard's billing request. When the card information arrives at the web application backend, we utilize Paypal's payment gateway API to process the transaction, and the bill is then complete.

Initially, we were planning to use both a magnetic stripe reader and NFC reader in our payment solution but we realized that it will be more cost effective if we cut the magnetic stripe reader as most modern credit cards have NFC technology.

3. Verification

As opposed to the lower-level component-based verification of operation that we used in our Design Document, it made more sense to shift to a module-based approach during later development of our prototype. This was in part due to organizational concerns, as we had components that required significantly different procedures for verification and different hardware for testing, but also due to the interdependence of the requirements for our components. As such, we sought to verify multiple low-level requirements per a higher level application, such that if these tests worked, we would know that all low-level requirements were satisfied. If these tests failed, we could fall back to our old verification processes to troubleshoot. A table detailing the requirements, steps, and results of verification is included in Appendix A. In sections 3.1 to 3.4, we elaborate on the verification process per module.

3.1. Table Adjustment Module

The components used in this module relied heavily on the reliability of the other components. This was made especially apparent to us while testing the ultrasonic sensors with our microcontroller, which had trouble sending simultaneous outputs to both the front and rear distance sensors. Because of this, we could test that our microcontroller used in this module was able to parse 10 inputs to each sensor per second, and verify that each sensor could send handle 10 inputs from the microcontroller per second, yet fail when both sensors were attached to the microcontroller and both were tested simultaneously. With this in mind, despite the sufficient verification of our lower level components, it became necessary to additionally audit the integration of multiple parts on top of this. Therefore, to verify our Table Adjustment Module components, we took advantage of the serial communication pins of our microcontroller and output to the Arduino IDE. We did this by connecting our Table Adjustment Module PCB to an Arduino Uno as shown in Figure 9.

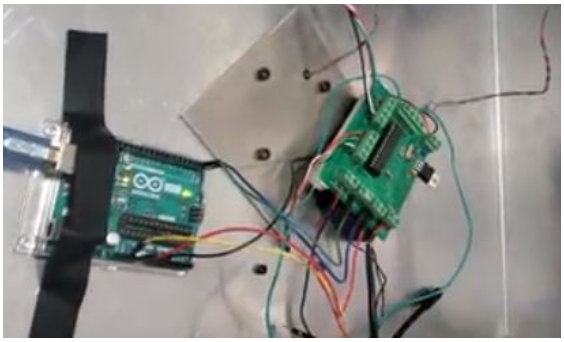

Fig. 9: Configuration between Table Adjustment PCB and Arduino IDE

With the displayed configuration, we had access to the serial monitor included as a part of the Arduino IDE and were able to view what signals were received by the microcontroller and confirm that it was accurate in the uploaded program.

3.2. Customer Presence Module

To collect the resistance values of our FSR, we attached an ohmmeter across its copper leads and applied even force downwards with a flat plastic board. We employed a digital scale under the components to measure the force downwards. The resistance measurements were taken over a wooden table on a hard floor. The resistance values we collected as related to applied force is provided in Appendix G. Initially, we declared there to be a customer presence if a force exceeding 20 kg (measured voltage 4.7 V) was applied, and sent whether or not a presence was detected to the Table Network as elaborated in Section 1.3.3. However, to achieve more consistent results, as force in practice is not always evenly distributed, we changed this voltage threshold to be 3 V for demonstration purposes. This voltage was observed when the resistor had about 15 pounds of force applied downwards onto it, and compensated for softer surfaces for the resistor to be placed on and for uneven amounts of force applied to it.

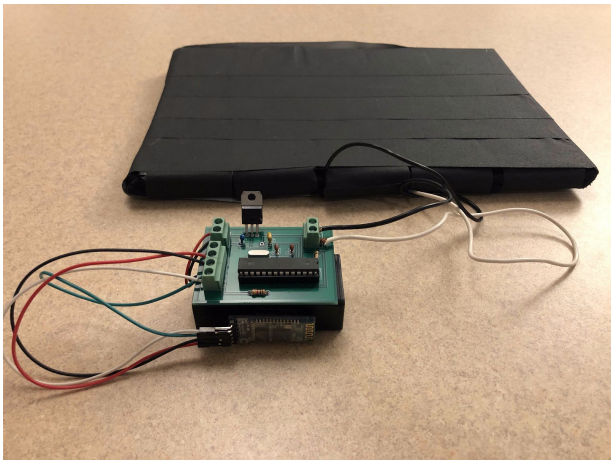

Fig. 10: Force Sensor Resistor with Chair Module PCB

Verification for the load cell can be found in Appendix A under "Customer Presence Module".

3.3. Table Network Module

Verification can be done by connecting all devices on the same network and pinging each other. Bluetooth connection can be verified by the steps defined under the Table Network Module in Appendix A.

The overall network topology of how devices communicate is shown in Appendix H.

3.4. Payment Module

Our verification of the payment module was done by connecting the 4-digit 7 segment display and the NFC Reader to our RPi Zero W. We then programmed the GPIOs to print out digits to the display and we verified that display is printing what we requested seen in Figure 11. To verify the functionality of the NFC reader, we will need to write a python program that will constantly poll for NFC contact and will extract the information from it. We verified it that the payment information has been parsed and the NFC waits for NFC contact shown on Figure 12.

Fig. 11: 4-digit 7 Segment Display Printing a Total Bill Amount and Confirming a Successful Transaction

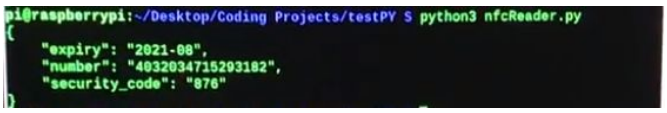

Fig. 12: Linux Terminal Printing the Content Read From an NFC card

4. Costs

4.1. Labor

An entry-level Electrical Engineer's average hourly wage is $31.12 [7] and an entry-level Software Engineer's average hourly wage is $35.61 [8]. Each member of our team worked on both hardware and software. Taking the average of both hourly wages into consideration, each engineer on a team using our design can be estimated to be making $33.37 per hour. Considering the time frame for a build to be 16 weeks, and assuming each person works on it for 10 hours, the cost of labor for three engineers contracted for the assignment would be $40,044. In addition, our table was made by the ECE machine shop, paid for by the department, which we estimate would incur a labor $25 per hour, for approximately 6 hours. Our two PCBs were quoted at $23 each, for a total of $46 including labor, manufacturing, and shipping. With all these numbers in mind, the total labor cost comes out to a grand total of $40,240. The calculations described and related costs are shown in Table 5.

Table 5: Involved Labor and Related Costs

Labor Description	Math	Total Cost
Engineering Labor	$3 people \cdot \$33.37/person\ hr \cdot 10 hr/wk \cdot 16 wk \cdot 2.5$	$\$40,044$
Machine Shop	$\$25.00/hr \cdot 6 hr$	$150
PCB Production and Labor	$2\ pcbs \cdot \$23/pcb$	$46
Total Labor Cost	$\$40,044 + \$150 + \$46$	**$40,240**

4.2. Components

Table 6: Components Used and Related Prices

Component Name	Component Description	Manufacturer	#	Total Cost
B07RT54H9V	24V 5A DC Power Supply	Arcity	1	$22.98
B07BXBS93X	9V Battery Holders	LAMPVPATH	4	$8.99
L7805CV	Linear Voltage Regulators	MCIGICM	4	$6.99
BTS7960	H Bridge Motor Driver Circuit	SongHe	1	$8.75
A17092900ux0369	24V DC High Torque Gear Motor	uxcell	1	$32.99

Table 6: Components Used and Related Prices (continued)				
Component Name	Component Description	Manufacturer	#	Total Cost
A18050400ux0104	Latching Push Button Switch	uxcell	1	$7.79
B07FS9G4ZJ	Momentary Push Button Switch	WGCD	2	$9.98
HC-SR04	Ultrasonic Distance Sensor	Excelity	2	$9.99
ATMEGA328P-PU	Microcontrollers with Sockets	Fii Tech	2	$12.25
Arduino Uno	Microcontroller programmer	Arduino	1	$22.79
HC-05	Bluetooth Module	HiLetgo	1	$8.99
B07RTHD45H	Screw Terminals	QSU	40	$10.99
L6LAC003-DT-R	Power Cords	AmazonBasics	2	$7.42
WP2PINBB	QD Connector Pairs	BTF-LIGHTING	2	$8.88
N/A	Antistatic Semiconducting Foam Sheet	Multicomp	2	$14.99
N/A	Roll of Copper Tape	Samyoung	1	$7.99
N/A	Roll of Gaffer's Tape	YYXLIFE	1	$7.97
TM1637	4-Digit Tube LED Segment Display	Comimark	1	$5.49
RPi Zero W	Single board computer	Raspberry Pi	1	$14.00
ACR122U	USB NFC Reader	ACS	1	$42.75
OTG4HUB	MicroUSB to USB Port Hub	LoveRPi	1	$6.99
Total Component Cost				**$279.96**

4.3. Total

As shown in equation 4.1.1, taking into account the calculated labor costs and the total cost of components, we estimate that this project would cost about $40,520. It must be noted that the total price of the components was higher than one might have with contracts with the companies; we purchased the parts for our prototype through retail sites and secondhand electronics shops online due to speed constraints. We have reason to believe that all of these components had a significant markup from the vendors we bought them from.

(eq 4.1.1)

$$Total\ Cost = Labor + Components = \$40,240 + \$279.96 = \$40,519.96 \approx \$40,520$$

5. Conclusions

5.1. Summary

Overall, our project culminated in a working prototype that showcased possible improvements in the restaurant setting. Looking at each of our working modules, we consider this project to be a success. By programming a dedicated algorithm to take advantage of our chosen sensors, we were able to model a table that adjusts to any chair that will fit under it. With it interpreting information from our customer presence module, we were able to successfully share the vacancy status of our prototype table with an external device that one might use as an employee in a restaurant setting. Finally, we explored and modeled a method of contactless payment that could potentially see success in a regulated environment. The modules we constructed offer solutions to a changed dining landscape.

5.2. Ethics

Per the IEEE Code of Ethics, which demand a responsibility "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment" in Article 1, we made efforts to comply to safe and ethical design in our project [9].

On the physical level of our prototype, we had to consider various factors that could potentially harm users: the moving parts and the high power used in our prototype specifically pose a security threat. One particular concern is in the automatic adjusting height mechanism of the table: an error could cause the motor driving the system to keep adjusting downwards and possibly injure a user. To address this, we implemented a safety control and a manual override system in order to adjust the height of the table should it be desired.

Since we used high voltage and current in our circuits, the erroneous discharging of our electronics to the user could be potentially fatal. We used a 24 V (5 A) DC power supply in the base of our table, 9V battery holders, and various 5 V (1.5 A) connections elsewhere in the table. All of these voltages are potentially life-threatening if passing through a user. Therefore, we made efforts to segregate the electronic components from areas that would be easily accessible externally. For the top of our prototype, we used non-conductive foam to form a physical barrier from our power wires and the metal shell that most of our model was made of. For our custom FSR, we used thick sheets of PVC to increase resistance between the copper

tape plates and any object or person on top of the sensor. We also used gaffers tape as a second, waterproof layer for this component.

Finally, since the table we built involves a payment and billing mechanism, there is a possibility that private information has the capacity to be accessed by a malicious actor. For our demonstration, we used fake card information and a private network. For future consideration of our product and integration into a commercialized environment, it is paramount that security measures are put into place to protect private information from the transactions. This is elaborated on in Section 5.3, Future Work.

5.3. Future Work

Considering the exposed sensors used during project development, one point of future improvement we recommend going forward with this design is in upgrading the sensors for more seamless integration. Our prototype utilized hobbyist components, and collaborating with sensor manufacturers and furniture manufacturers for specialized hardware would result in a higher quality product at a reduced cost.

One other area of improvement to pursue for our project is in the security of using NFC communication for payment. In our demonstration and proof of concept, we showcased the modules relying on a personal mobile hotspot. For future products based on this design, some possible avenues to take include using a hidden personal network, the development of a restaurant's own proprietary ad-hoc network, and the integration of encryption and decryption mechanisms in the communication between restaurant tables and the central hub that manages them. Potential cryptography solutions include using AES encryption, having IPSec enabled between the tables and hub, and using a OTP system for all allowed connectections to the network.

References

[1] PYMNTS, 'Restaurants That Aren't Ready For A Mobile Future Risk Losing Out', *PYMNTS*, 2020 [Online]. Available: https://www.pymnts.com/news/mobile-payments/2019/restaurants-mobile-ordering-food-delivery-ritual/ [Accessed 17-Sep-2020].

[2] MRM Staff, 'The Future of Dining: Industry Expert Insights', *MRM*, 2020 [Online]. Available: https://modernrestaurantmanagement.com/the-future-of-dining-industry-expert-insights/ [Accessed 17-Sep-2020].

[3] Yelp, 'Increased Consumer Interest in May Correlates with COVID-19 Hot Spots in June, According to the Yelp Economic Average', *Yelp*, 2020 [Online]. Available: https://www.yelpeconomicaverage.com/yea-q2-2020.html [Accessed 17-Sep-2020].

[4] Sparkfun Electronics, *Github* 2020 [Online]. Available: https://github.com/sparkfun/SparkFun-Eagle-Libraries [Accessed 28-Sep-2020].

[5] C. Focant, "tango.lbr." Autodesk, San Rafael, California, 06-Aug-2008. http://eagle.autodesk.com/eagle/download/800

[6] G. Bennett, private communication, Oct. 2020.

[7] Payscale, 'Average Electrical Engineering Salary', Payscale, 2020 [Online]. Available: https://www.payscale.com/research/US/Job=Electrical_Engineer/Salary [Accessed 7-Oct-2020]

[8] Payscale, 'Average Software Engineering Salary', Payscale, 2020 [Online]. Available: https://www.payscale.com/research/US/Job=Software_Engineer/Salary [Accessed 7-Oct-2020]

[9] IEEE, 'IEEE Code of Ethics', *IEEE*, 2020 [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 17-Sep-2020].

Appendix

A. Requirements and Validation Table per Project Module

Table of Requirements, Validation, and Results per Project Module		
Requirement	**Verification**	**Result**
Height Adjustment Module		
The microcontroller must determine the distance to any customer seating while eliminating outliers at a rate of 10 samples a second, and sensors will work up at least up to our maximum table height of 40 inches	1. Connect serial connection from height adjustment PCB to Arduino 2. Display the serial output via the Arduino IDE 3. Put the table in automatic mode for at least one second 4. Visually confirm that 20 readings, 10 per each of two sensors, are managed per second 5. Mathematically confirm both medians are correct 6. Remove objects under the table to make it automatically adjust to the maximum height 7. Verify that the collected information from the sensors is accurate when the table is at a height of 40 inches	**Satisfied**
The height adjustment adjust upwards and downwards at a rate of an inch vertically between 10 and 20 seconds	1. Set table to manual adjustment 2. Hold tape measurer parallel with shaft of the table 3. Time the ascent and descent of the table and verify travelling an inch takes between 10 and 20 seconds	Ranges from 17-19 seconds depending on operation; **Satisfied**
The height adjustment system must have safeties implemented to disable automatic adjustment within an inch of travel	1. Connect serial connection from height adjustment PCB to Arduino 2. Display the serial output via the Arduino IDE 3. Put the table in automatic mode for at least one second 4. Enable the manual safety mode and time how long it takes to stop 5. Confirm on the monitor that the microcontroller has switched to manual control 6. Visually confirm that table has stopped moving	Table halts within 1 second, less than a 10th of an inch; **Satisfied**

Table of Requirements, Validation, and Results per Project Module (continued)		
Requirement	**Verification**	**Result**
Customer Presence Module		
Voltage difference can be measured when applying a force on the FSR to denote that a person is sitting on it.	1. Wire up FSR to microcontroller. 2. Print the output from the microcontroller to a terminal via UART. 3. Verify that values change according to how much pressure is applied to the FSR.	**Satisfied**
Load Cell can detect when there is an excess amount of weight on the tabletop.	1. Wire up the load cell to the Hx711 A/DC and the A/DC to the Raspberry Pi. 2. Place a large smooth surface on top of the load cells. 3. Run a program on the Raspberry Pi that prints the output weight sensed by the load cell. 4. Verify the weight value changes according to how much weight is placed on top of the surface atop the load cells.	**Satisfied**
Table Network Module		
Data from FSR sent via Bluetooth can be read at an approximate rate of one message every five minutes.	1. Pair Bluetooth device with Raspberry Pi. 2. Flash a program on the microcontroller connected to the Bluetooth device that will write data to the connected serial pin. 3. Run a program on the Raspberry Pi to read data from the paired Bluetooth device serial port at *the same baud rate*. 4. Validate via inspection that the message received was the one sent over the flashed program.	Data from FSR can be sent via Bluetooth to Raspberry Pi at a rate of one message every three seconds; **Satisfied**

Table of Requirements, Validation, and Results per Project Module (continued)		
Requirement	**Verification**	**Result**
Table Network Module		
All table nodes will be able to send information to a single central server node via sockets over a network.	1. Connect all necessary devices onto the same local area network. 2. Obtain the IP addresses of each device (e.g. 'ifconfig' command). 3. Attempt to ping each device from one device. If this step fails, go back and diagnose that step 1 was performed correctly. 4. Write a sender program to send a message over TCP from one device on the network to another. 5. Write a receiver program to receive a message over a TCP socket. 6. Run the sender and receiver programs on *different* devices but ensure they read/write to the *same* port. 7. Validate via inspection that the sent/received message match.	**Satisfied**
Payment Module		
Seven-segment display is able to output integer values	1. Connect the seven-segment display to the RPi Zero W. 2. Program the GPIOs to print an arbitrarily decided two to four digit number 3. Verify that the seven-segment display shows the number as requested	**Satisfied**
NFC Reader is able to poll for an NFC card and interpret transmitted information given by said card	1. Connect the NFC Reader to the Rpi Zero W 2. Run the terminal-based NFC reading program to print the data from the card 3. Quickly bring an NFC card near the reader 4. Confirm that information is transmitted by the card by observing the output of the reading program 5. Confirm that the transmitted information is as expected	**Satisfied**

B. Finalized Prototype

C. Restaurant Dashboard

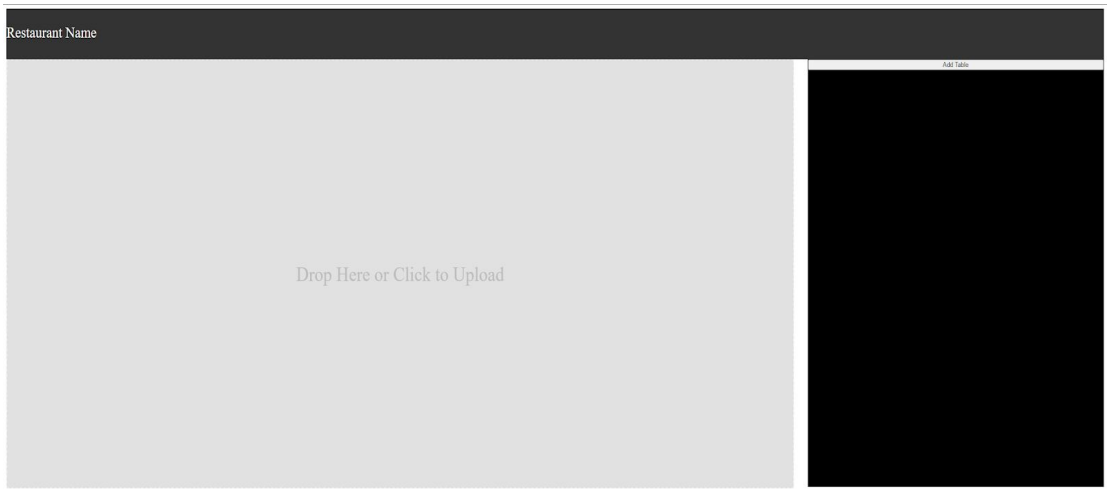

Fig. 1: Empty Dashboard

Fig. 2: Dashboard with Floor Plan and Table Node

D. Payment

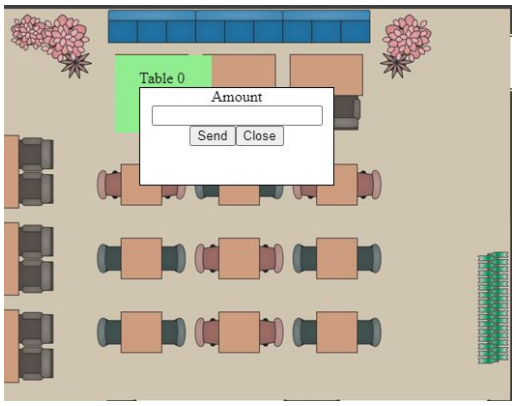

Fig. 1: Payment Interface

Fig. 2: 4-digit 7 Segment Display

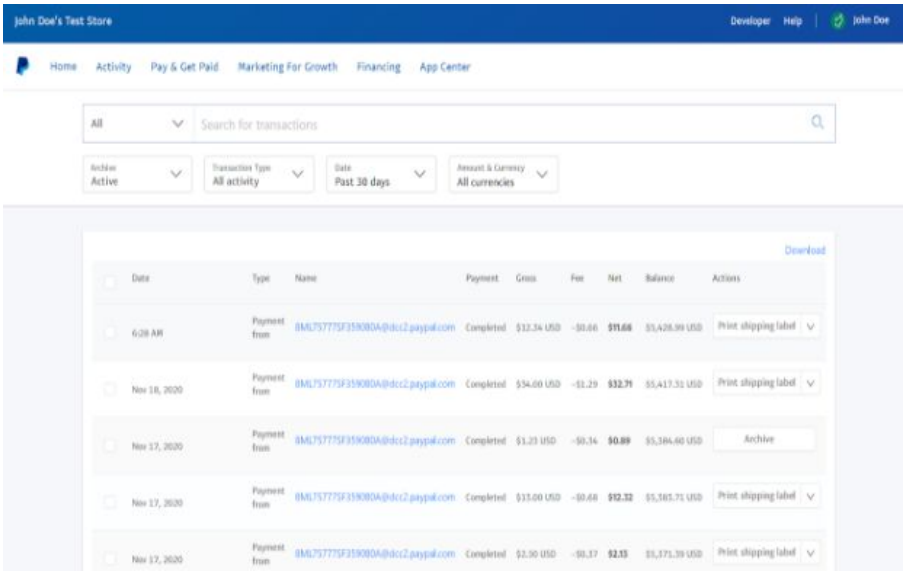

Fig. 3: Paypal Dashboard

E. Schematics

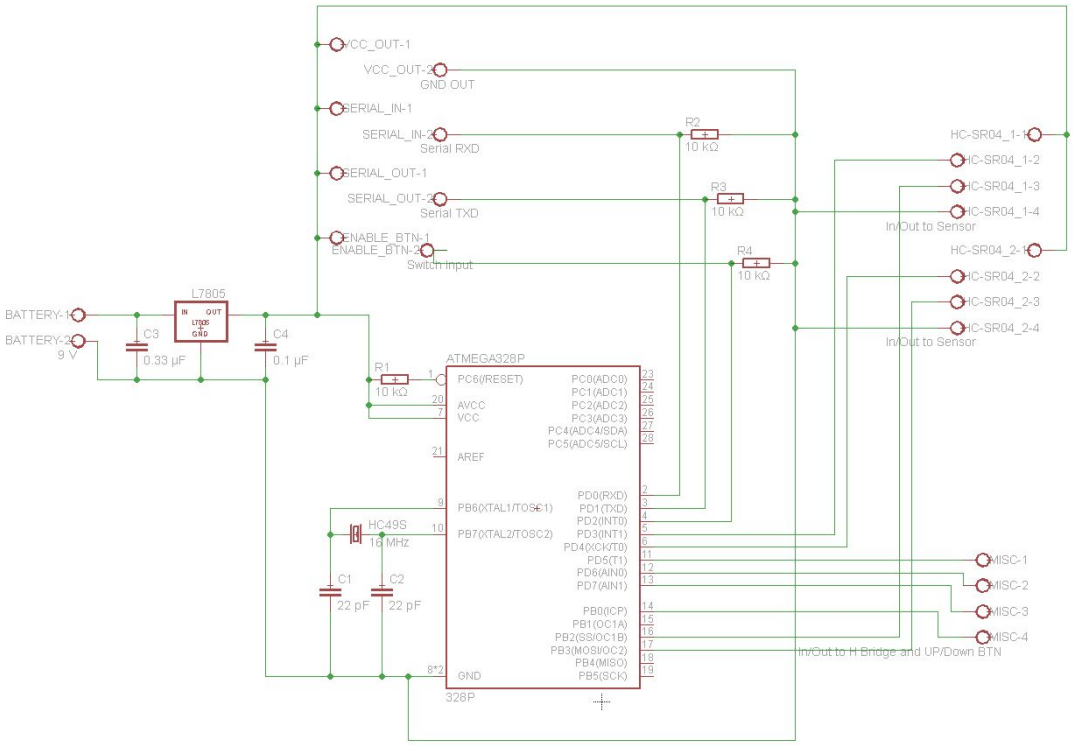

Fig. 1: Schematic of our Table Height Adjustment Module PCB

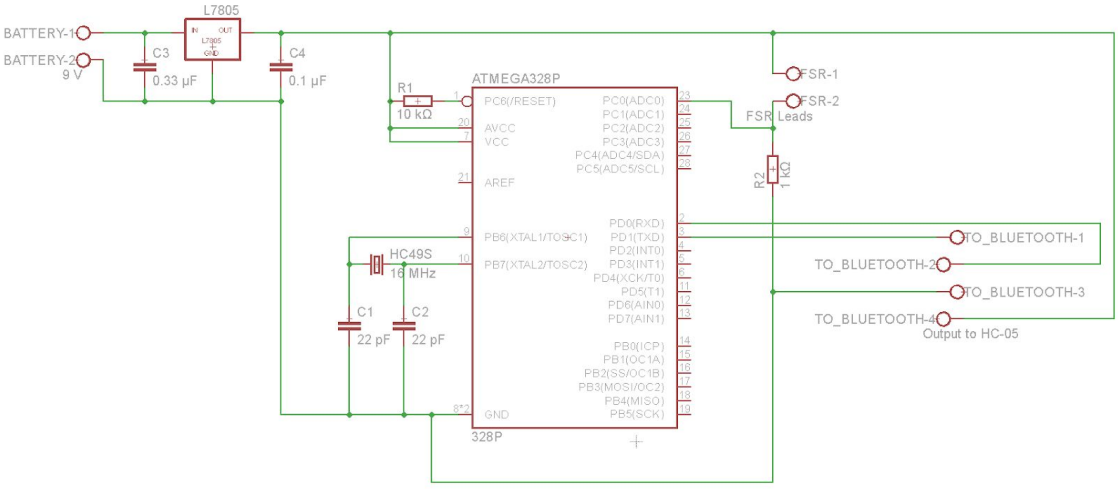

Fig. 2: Schematic of our Chair Module PCB

F. Board Design

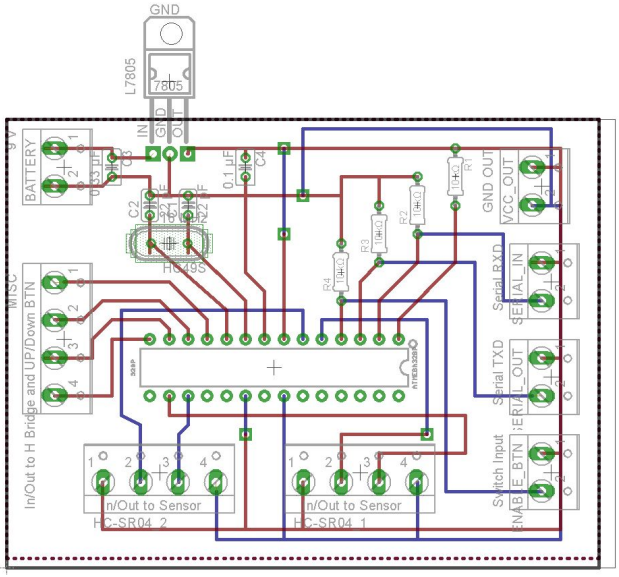

Fig. 1: Table Module PCB

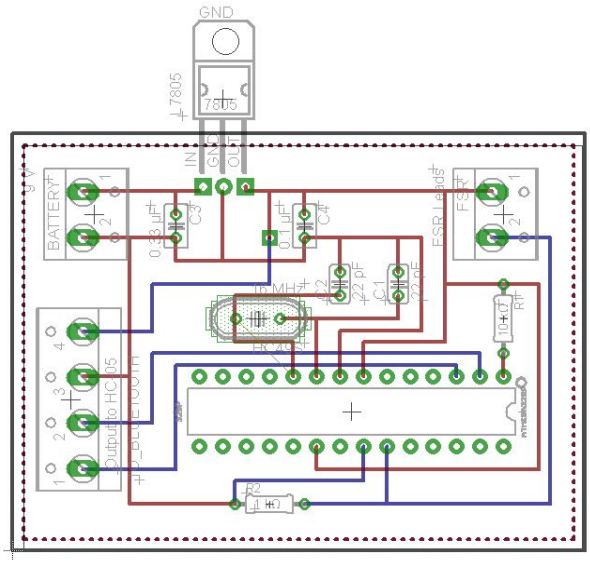

Fig. 2: Chair Module PCB

G. Resistance Values Collected for Custom FSR

Table of Resistance Values per Applied Force for FSR			
Weight on the FSR (in lb)	Resistance Measured by Voltmeter (in Ω)	Weight on the FSR (in lb)	Resistance Measured by Voltmeter (in Ω)
0	40 to 60 kΩ	31	82
17	330	32	80
18	130	33	92
19	130	34	71
20	200	35	150
21	155	36	81
22	134	37	70
23	142	38	62
24	136	39	66
25	132	40	45
26	140	41	43
27	109	42	42
28	108	43	44
29	90	44	40
30	400	45	49

H. Network Topology