**Electrical Engineering Practicum: A Method for Improving Restaurant Beverage Refill Rates Using an Intelligent Coaster System**

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Taylor Kramer
Spring, 2020

Technical Project Team Members
Daniel Ayoub
William Define
Adam El-Sheik
James Garcia-Otero

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature  Date 05/07/2020
Taylor Kramer

## Abstract

The ECE Capstone group of Electro-Magnetic Friends built a Smart Coaster system to improve the consistency of drink refill rates in a restaurant. The Smart Coaster makes use of force sensing to determine the relative weight of the glass placed on the coaster. The coaster then determines when drinks are nearly empty, and wirelessly notifies the wait staff over WiFi to refill the empty drink. The Smart Coaster also makes use of wireless charging, a technology which eliminates the need for any open ports. Combining wireless charging and ingenious mechanical design, the Smart Coaster is water resistant at a rating of IP54.

The project involved the design and construction of three main components: the coaster itself, the inductive charging station, and the wait station desktop app. While the Smart Coaster may seem like a relatively simple concept, the project involved wireless communications over WiFi, embedded and network programming, iterative mechanical design, and force sensing. In this report, we detail the motivation for this project and any prior art. We also report on the standards used by the Smart Coaster and the tools which made all of this possible. Lastly, we go into the technical details, test plans, costs, and future work.

### Background

During a restaurant outing, the most frequent point of interaction between servers and patrons is the process of drink refills. This is a multi-element system which requires the server to repeatedly check each table for near-empty drinks to refill. As a result, the customer experience for drink service is highly variable and correlated with the quality of server. If an un-observant waiter fails to notice an empty drink near a thirsty patron, the patron will spend time and energy attempting to find and flag down the waiter. This puts strain on the customer experience and reflects negatively upon the establishment. Although a diligent waiter may be able to notice empty drinks in a timely manner, this requires additional effort from the waiting staff. This project provides an opportunity to increase server efficiency while at the same time increasing customer satisfaction.

Similar attempts to address drink refill issues have occurred in academia. The most similar attempt was from students at Saarland University in Germany in 2005 [1]. Students designed a beer mat to detect weight of empty drinks, and incorporated coaster-flipping detection to be used in “voting” games. Although the Saarland University project is very similar, the smart coaster proposed in this project incorporates induction charging which sets it apart from competitors, and takes into consideration scalability. Additionally, University of Virginia students have previously attempted to solve automatic drink detection and serving through the use of a robotic arm with computer vision [2]. This previous project is similar in purpose to the proposed project, however it introduces ethical issues and creates opportunities for confusion with varied lighting sources. The designed Smart Coaster is scalable, easy-to-integrate, and aims to increase worker efficiency without removing the need for a waiter or introducing image security concerns.

Our project differs from prior work in this area, because we focus our efforts into a design that benefits every user in the process, while at the same time attempting to limit any negative outcomes which could result from a device in this area.

Key users and stakeholders in this process include the patrons, waitstaff, and the restaurant owners. Patrons have all privacy protected, since the only data being used is cup weight data, rather than visual camera feeds which other implementations may use. In addition, patrons are given all capabilities to indicate when they would like to use the system, removing unethical automatic order of unwanted drinks. Waitstaff are assisted by this technology, reducing the time required to check every table for empty drinks and the multiple trips back and forth this would require. Since the device does not directly replace a waiter, the waiter's job is not directly in competition either. In addition, the restaurant owners are included in the recipients of positive results, since patrons using these devices will have consistent and efficient drink refills, ensuring service quality throughout all waitstaff. This will lead to more positive patron experiences, and encourage positive reviews and return visits. Return visits to the restaurant may also occur due to the novelty of the smart coaster device.

Our team chose this project because it incorporates a wide range of engineering principles and product design. Although the basis of this project is in electrical and computer engineering design, it requires understanding and consideration at a system level. It also brings into play a significant amount of user experience investigation and product design. This project solves a real-world problem, and has the opportunity to positively impact multiple parties. Design for the smart coaster uses background knowledge from a variety of engineering courses. The induction portion of this project calls into play knowledge from Electromagnetic Fields, which was taken by three members of our group. One member also has coursework experience in Microwaves and RF. Also important is the foundational knowledge, hardware design, and embedded programming learned from the FUNdamentals series and Intro to Embedded classes, taken by all members of the group. Three members of the group have also taken Computer Networks, which is important to the networking component of the smart coaster system. Digital Signal Processing and Electromagnetic Energy Conversion were also taken by a team member and come into play with the sensing and power requirements of this project. Two team members also have experience with 3D printing and physical component design, which contributes to producing a successful product.

*Design Constraints*

One of the largest constraints in terms of the manufacturability of this system is the size. A standard coaster is typically around 3 to 4 inches in diameter and is typically not very tall or wide. As such, we designed all of our components to be as small as possible, fitting into a form that is suitable for placement on a dinner table while comfortably holding a drinking glass.

A major constraint was that the smart coaster needed to also be designed such that it can handle typical usage, which includes being frequently moved around and being in the presence of liquids. Since the coaster is split into a base enclosure and a top cover which is able to be cleaned separately without compromising the main contents of the device, it is able to withstand heavy usage.

The small size design constraint also resulted in a constraint for our PCB to include surface-mount components. Although this reduced the space required for a component to be placed on the PCB, this increased the time required to solder components, and also necessitated multiple trips to 3W to professionally attach the surface-mount MCU, adding cost.

Due to cost constraints, all components used were relatively inexpensive and common pieces. This meant that there were no significant time constraints regarding purchasing of electrical components. With regard to the initial PCB however, there were some time constraints due to setbacks with the delivery timeline of the first PCB.

*Economic and Cost Constraints*

Cost constraints of this project included designing a coaster which is able to be cost-effective for a restaurant. For this stage of design, we set a constraint for designing a system which has manufacturing totaling under $100 per coaster. With further iterations of design and more interaction with potential buyers, we would set further cost restraints which could scale appropriately with mass production and implementation. In the end, the cost to purchase all the components for a single coaster is $37.86. With the cost of manufacturing a PCB and services for soldering all the components, the total increases to around our cost constraint. If we were to manufacture 10,000 units, the total for all the components would be reduced to $16.28, which makes a coaster cost around $75 to manufacture in bulk.

**External Standards**

1. Wireless: The coaster uses the 802.11 (WiFi) standard to communicate between the smart coasters and the laptop Python terminal application [3]

2. PCB: We used IPC standards for laying out our PCB. IPC standards regulate what part spacing and track spacing to use on the PCB. [4]

3. Python: The server application uses PEP8 style guide for indenting/naming python code. [5]

4. Water Resistance: The coaster follows the IP54 solids and liquids ingress rating. The IP54 standard states that product will be protected from limited dust ingress and be protected from water spray in any direction. [6]

5. UART: We used UART serial communication standards to program the MCU.[7]

6. Qi Standard: A Wireless Charging Standard by an industry group called the Wireless Power Consortium. The standard is applicable for electrical power transfer over distances of up to 40 millimeters or 1.6 inches. The standard specifies that the electrical energy transfer is primarily accomplished using magnetic induction. Additionally, this standard specifies an output of 5-15 W of power transmission for electronics that can operate in this region. We used this standard for our inductive charging. [8]

7. Food: The smart coaster follows the NSF/ANSI 51 standard for food equipment materials, which establishes sanitation requirements for materials used in commercial food equipment. [9]

## Tools Employed

To accomplish a project of this magnitude, we had to make use of several external tools and applications. These tools collectively made our lives and the design much easier.

1. Python - Python is a relatively high level and extraordinary popular programming language. We used it to write our laptop terminal server application. [10]

2. Vim - Vim is a text editor. We used it to write Python for our laptop terminal server application. [11]

3. Code Composer Studio (CCS) - CCS is an IDE for TI embedded processors. We used it as an IDE for C to program our board. [12]

4. C - C is a low level programming language. We used it to program our smart coaster. [13]

5. CCS UniFlash - Tool for programming flash memory on the MCU. It allowed us to write our program to flash memory. The MCU reads and run the program from flash memory on startup. [14]

6. NI Multisim - NI Multisim is a circuit simulation program. We used it to design/verify our circuit. [15]

7. NI Ultiboard - NI Ultiboard is a PCB layout program. We imported our circuit from Multisim. Then we did our PCB layout in Ultiboard. [15]

8. AutoDesk Inventor Professional - Inventor is a professional-grade 3D mechanical design, documentation, and product simulation tool. It works efficiently with a powerful blend of parametric, direct, freeform, and rules-based design capabilities. We used it to design the body of the coaster. [16]

9. Ultimaker Cura - Ultimaker is a slicing engine. It takes a 3D Object or CAD file, slices it horizontally, and encodes the slices into g-code, a format understood by most 3D printers. The g-code is ran on a 3D printer which prints out one slice at a time until the 3D object is formed. We used it to slice our AutoDesk Inventor file, which was first converted into an STL format, to 3D print our coaster components. [17]

## Ethical, Social, and Economic Concerns

*Environmental Impact*

The production of this device will have a slight negative impact on the environment, due to the manufacturing processes of electronics. Future versions of this project could attempt to cut down on this environmental impact, focusing on electrical components which are safer to manufacture, or storing renewable energy to power the charging circuits used to wirelessly charge the coasters.

*Sustainability*

As a reusable and rechargeable drink coaster, this project will eliminate the need for restaurants to use and discard disposable paper or cork drink coasters. This cut in disposable coasters will reduce restaurant waste, and enable the drink support process to be more sustainable.

*Health and Safety*

Since the smart coaster will be used in the presence of food and drink, we needed to ensure that all the materials used for manufacturing, specifically the 3D printing material, is food safe. PLA plastic was used to print out all the components users would come into contact with, and while most forms of PLA are food safe, we also ensured that there were no additives that would cause the coaster to not be safe around food. This would be important in the case of spills, which is likely to occur in an environment where a coaster would be present.

*Manufacturability*

Considering all the components that we needed to fit in this form factor (battery, induction coil, embedded processor and associated PCB, and force sensor), we had to design our components to be as small as possible. This led to us designing a 4-layer PCB, which was part of our main design, and could contribute added complexity in the manufacturing process. Our final device presented at the capstone fair ended up being much larger, due to issues we had with our PCB, leading us to design a custom two-layer header board which reduced complexity but required us to use the launchpad. Use of the launchpad in the design increased cost of a coaster, and increased the amount of 3D filament required to create the coaster. Full implementation of our main PCB with the standard coaster components however is more manufacturable in the end, due to smaller size.

*Ethical Issues*

This project has limited ethical issues. Since this is an assistive technology, waiters' jobs will not be directly replaced. With the increase in waiter efficiency, however, it is possible that an individual restaurant may require fewer waiters at a time. Indirectly, with the increase in efficiency, this could result in a slight replacement of jobs, but the effect would be minimal when compared to a technology intended to fully replace a waiter. The Smart Coaster is intended to alleviate concerns from both sides and seeks to improve the waiters' ability to do their jobs.

The Smart Coaster system includes a human waiter in the refill process which alleviates concerns regarding alcoholic drinks. While other drink refill systems may automatically fill customers' drinks, the Smart Coaster system maintains the waiter-customer interaction, enabling waiters to check intoxication levels to ensure customer safety.

In order to protect the customer's economic interests, the Smart Coaster system also includes a user-interaction button which can disable the device and cancel requests. In such a way, the customer can self-select into the automatic drink refill. This is especially important for drinks which are pay-per-refill. When the customer begins their restaurant visit, they accept the coaster and depress the button. At this point, the device is now enabled and updating refill states to the wait-staff station. The user can disable the device at any time, pressing the button to cease

sending refill states after they have refilled a desired number of times. This button also allows restaurants to reduce wasted drink refills, since a customer can disable the refill device when they know they are about to leave and no longer want more drink refills.

*Intellectual Property Issues*

This project is not patentable due to prior patents whose claims together encompass material providing full coverage of our project.

Patent 1: US7353136B2 [18]

This patent covers the primary purpose and functionality of our project with an independent claim (claim 1) for "an electronic drink coaster, comprising a coaster containing a weight detector for sensing a weight or weight change of a container with drinking fluid on the coaster, and further comprising a processor for processing signals from the weight detector, and a communications port for wirelessly relaying information relating to the drinking fluid on the coaster, wherein a bar or restaurant is notified to provide a new drink when the drinking fluid is consumed." The only portion of our project which this patent does not cover is the inductive charging portion.

Patent 2: US8629654B2 [19]

The inductive charging portion of our project, however, can be covered through this patent, which describes the use of inductive charging coils for electronic devices, and has an independent claim (claim 1) regarding the use of inductive charging with a primary transmission base unit (which refers to the project's charging station) and a receiver unit (the project's coaster) to charge portable devices or batteries.

Patent 3: US9432758B1 [20]

Additionally, there is also a similar patent for electronic drink coasters which do not necessarily operate wirelessly. The relevant claim in this patent is an independent claim (claim 1) regarding the design of the coaster itself. In this case, key portions of the claim include a flexible coaster case top which deflects upon the presence of pressure, use of a circuit board, and the use of audio signals as an input to the device. If we were to use a flexible case covering for our project, which we tested in one of our design iterations, this patent is relevant to that design decision. Since the flexible cover

Although the third patent explored does not affect patentability in light of its claims, the first two patents explored together provide enough coverage to result in this project not being patentable. Considering the extent of the coverage provided by the first patent, inclusion of wireless charging to improve portability and waterproofing of the device is simply not enough to make this a patentable project on its own.

## Detailed Technical Description of Project

*System Design*

The Smart Coaster system is split up into three components: the coaster, the charger, and the wait staff station. The wait staff interacts with all three components, ensuring functionality, while the customers only see the user-facing end of the coaster itself.

The coaster includes a series of electronics enclosed by a 3D-printed casing. The 3D printed case has an inner enclosure which is waterproofed and protects the electronics from liquids and use. The integration of wireless charging allows the coaster to function without any externally-facing ports which could be ruined by water. This inner enclosure is then topped with a 3D-printed cover, which covers the top and sides of the coaster, including an opening to allow button access. This cover is the surface coming into touch with potential spills and drips, and can be cleaned separately to allow full restaurant usability.

The charging station includes a pre-purchased inductive charging transmission board, fitted into a 3D-printed casing. This charging station may be plugged into a wall outlet anywhere in the restaurant, and each individual circuit will charge one coaster at a time.

At the beginning of a customer transaction, the waiter arrives to the table with an appropriate number of Smart Coasters. Each coaster is labelled with a sticker signifying table number and coaster number. Each of these coaster number combinations is mapped in the system to the coaster's appropriate IP address, allowing the system to know which address is associated with which table and coaster number. As the waiter takes drink orders, he or she also notes which coaster each drink belongs to.

Once all orders are taken, the waiter heads to the waitstaff station and begins a new transaction. He or she selects for each coaster the appropriate drink type according to what each customer ordered. This allows the waiter to later know which drinks to bring refills of, without having to ask the customer for a reminder. The specification of drinks also allows the system to register refills based off of typical glass type. For instance, a single malt whiskey will have different full versus refill-required sensor readings than a standard 16oz restaurant glass of water.

The coaster uses a force sensing resistor to obtain a voltage reading which changes along with the weight placed on top of the coaster. The coaster then communicates this voltage reading to the server code, which is where all of the threshold values are stored. Based on the drink type and thresholds, a full or empty signal is communicated to the waitstaff station python terminal. Here, the coaster states are only updated once a change of state is detected.

Since the waitstaff station is more centrally located, and waiters can observe the status of multiple tables at once, the waiter will know drink refill states with much more reliable timeliness. Once the waiter sees a "refill required" signal appear on the terminal for a particular coaster, he or she will read the drink associated to that coaster number, and bring the appropriate drink to the table and replace the customer's empty cup.

At the end of the day, waitstaff will collect the coasters from each table and charge them using the wireless charging stations. A full charge will last the whole restaurant day, and transaction periods can be open or closed using the waitstaff python terminal.

*Electrical Design*

In this section of the report, we go into the details of the entire electrical design and layout of the Smart Coaster. The electrical design can be broken into two main components: the schematic and the layout, each having their own subsequent design decisions and iterations. The electrical design was the most iterative component of the project, as it had to be integrated seamlessly with the software, mechanical, and system design components of the coaster.

*Schematics*

The schematics of the Smart Coaster can be divided into five main sections: RF Design, Force Sensing, Power, MCU, and TI Header Schematic. Below are the specifics behind the electrical design and the reasons for why certain decisions were made.

*RF Design*

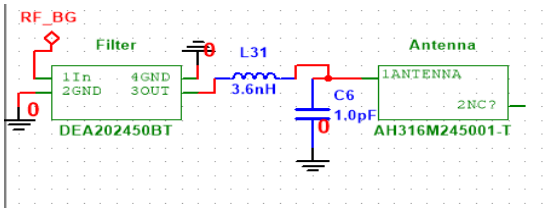


**Figure 1 RF Schematic**

To implement the WiFi connectivity of the Smart Coaster, an RF section, operating at a frequency of 2.4 GHZ, had to be incorporated into the schematic. This section of the board consisted of a chip antenna, 2.45GHz bandpass filter, and an LC matching circuit. The exact components used came directly from a Texas Instruments example for the CC3200 chip [3]. The interconnecting copper traces had to be considered as transmission lines due to the high operating frequency of this circuitry. The exact width of the copper traces was calculated to have a characteristic impedance of 50 Ohms. This calculation involved the dielectric constant of the board's material and the distance between cooper planes. The RF section was proven to work on the second iteration of the layout, as well be detailed later in the layout section.

*Force Sensing*

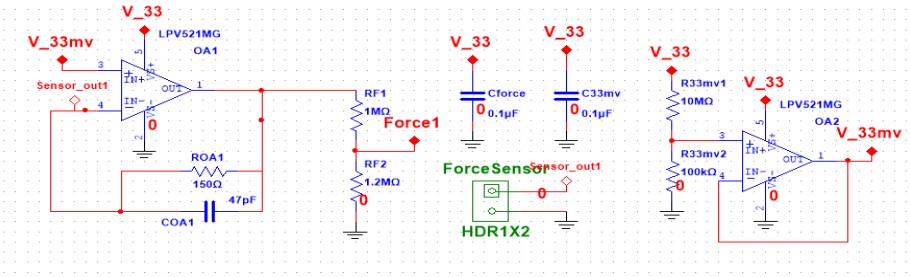


**Figure 2 Force Sensing Schematic**

As part of the electrical design, we needed to incorporate circuitry for the force sensing capabilities of the Smart Coaster. The force sensing circuitry made use of a force-sensitive resistor (a circuit element whose resistance changes when a force is applied to it) to output a voltage (Force1 in Figure 2) whose value was linear to the weight. This was a crucial element of the design as this is ultimately what read the weight of the glass placed on the coaster and fed the information to the ADC which processed it in software.

The circuit primarily used a non-inverting op-amp where $V_{out} = V_{in}(1 + R_1/R_2)$. $V_{in}$ is a constant 33mV as generated by a voltage divider and unity gain op amp (OA2 in Figure 2). $R_1$ is a feedback resistor (ROA1 in Figure 2) with a final value of 30.1kΩ. This feedback resistor changed the overall gain of the non-inverting op-amp, which had the effect of increasing or decreasing the sensitivity of the force sensor. Increasing the value of this resistor had the effect of making the weight sensing more sensitive. $R_2$ is the force sensitive resistor. Since $1/R_2$ is proportional to $V_{out}$, the conductance of $R_2$, and therefore the weight, is also proportional to $V_{out}$. The LPV521 op-amp was chosen as it works from rail-to-rail, and thus does not require a negative bias voltage. A capacitor was included in the negative feedback loop of the op-amp circuit to allow resilience from small spontaneous fluctuations in the weight placed on the coaster. A voltage divider on Force1 ensures that the CC3200 ADC never sees voltages over 1.8V as this would permanently damage it. The force sensing circuitry worked as expected.

*Power*

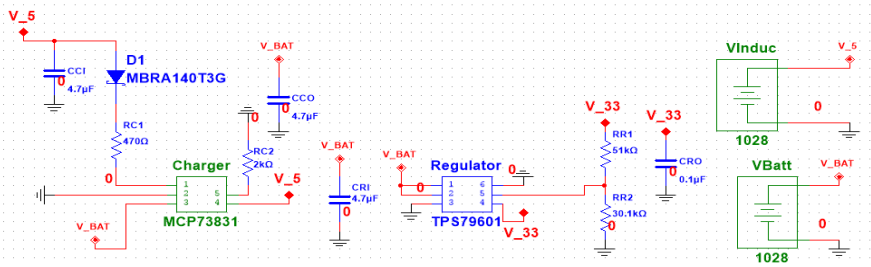


**Figure 3 Power Schematic**

The power section the schematic was arguably the most important part of the entire electrical design, as without it nothing would work. It was also the most precarious circuitry as there were three different voltages that had to be considered: a 5V supply coming from the inductive

charger, a 3.7V supply coming from the battery, and a 3.3V operating voltage to the MCU. All of this had to be handled correctly and effectively to allow for proper operation and charging/discharging of the battery.

The power section included ports for both the inductive power receiver and battery and provided over charge protection. It incorporated a 3.3V regulator to provide a steady voltage and current supplies to the microcontroller. The inductive power receiver was rated to transmit 500mA at 5 volts to the battery charging IC: the MCP73831. The MCP73831 prevented overcharging, protected against overdraw, and was placed in between the inductive charging port and the battery. This chip was successfully demonstrated to charge the battery.

On the first board, we used the TLV33P as our regulator. This regulator provided only 300mA max output current [21]. Despite our best efforts at incorporating many bypass capacitors, a ground plane and power plane, this first board had a current sourcing issue. We discovered that in the initialization of the WiFi subsystem, the board was drawing over 500mA for a short flicker of time. Because the board could not source this current, the initialization would fail, throwing the MCU into a hard fault state. In response to this issue, we upgraded to the TPS79601 regulator, which has 1A max output current. Our power section worked well on our header board, so this final design was adequate in sourcing the proper current.

*MCU*

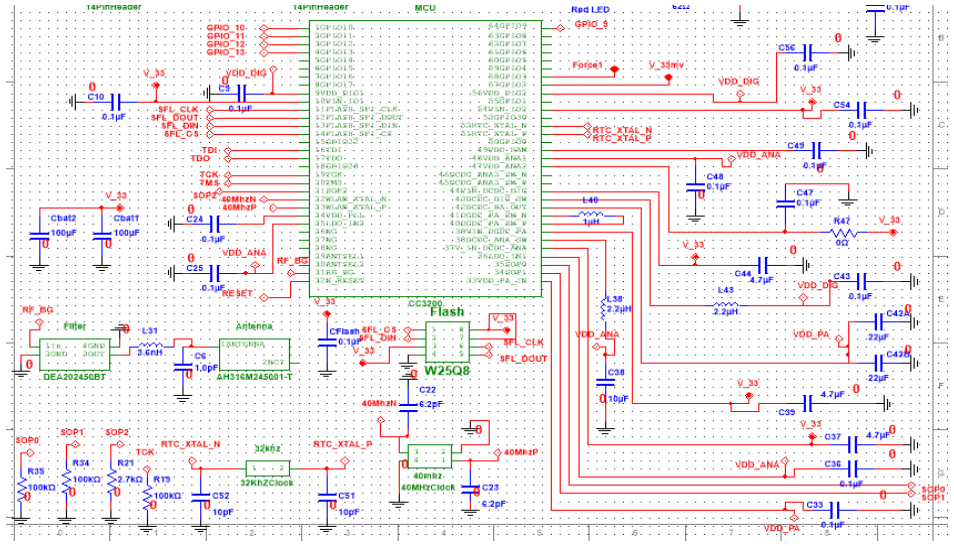


**Figure 4 MCU Schematic**

The MCU section of our board was the most complicated as the CC3200 requires a large number of bypass capacitors, pulldown resistors, external inductors, 2 crystal clocks, and flash memory. We debugged the board via 4 pin SPI-By-Wire and a Reset pin. Most of the circuitry here was recommended by Texas Instrument's documentation, which we mostly followed closely [21]. All internal voltages, SPI-By-Wire signals, and GPIO ports were accessed easily through header pins. Mistakenly, we did not include UART communication which made flashing the code onto the external flash memory difficult. The final board was able to communicate with the debugger and consequently the computer. It successfully ran the networking code and sensed when a glass

was empty or full. However, in order to get the code to run independent of the debugger, the code must exist in an external flash memory. Unfortunately, our efforts to flash the external memory ultimately failed, and the program was unable to work on startup. We also suspect that we may have unintentionally pushed the board into instability or fried the MCU in that process, which deemed the PCB as unusable.

*TI Header Schematic*

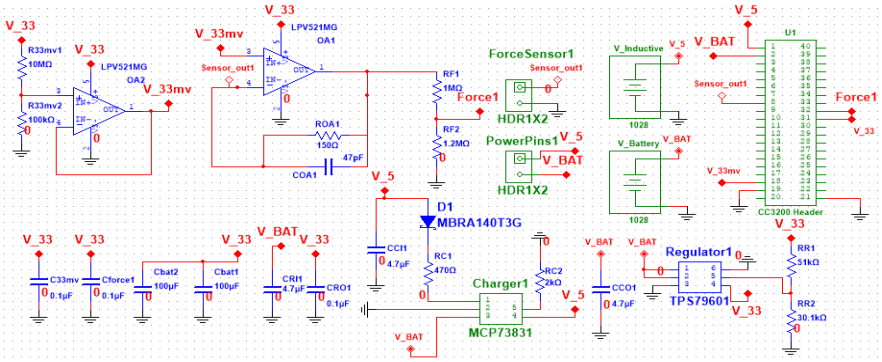


**Figure 5 Header Schematic**

The Header schematic consisted of the force sensing and power section of our board placed on a header designed for the CC3200 launchpad. This header was prepared as a backup in the event that our custom PCB didn't work. The header was responsible for powering the launchpad and for the force sensing capabilities. It was ultimately used it in our demo as our custom PCB ran into issues, and it worked reliably.

*Layout*

This project consisted of three 4-layer layouts. The earliest board represented our first attempt to make a custom MCU board. It was able to run simple programs such as flashing lights but ultimately was not able to source enough current to initialize the WiFi subsystem. We suspect this was due to poor layout of the bypass capacitors and interrupted power planes. The second board fixed the current sourcing issues of the first board, but ultimately was not able to run a program from flash memory, as the UART interface for flash memory was mistakenly left out. A header board for the TI LaunchPad was developed in conjunction with the second board in the case the second board did not work as well.

*First Board*

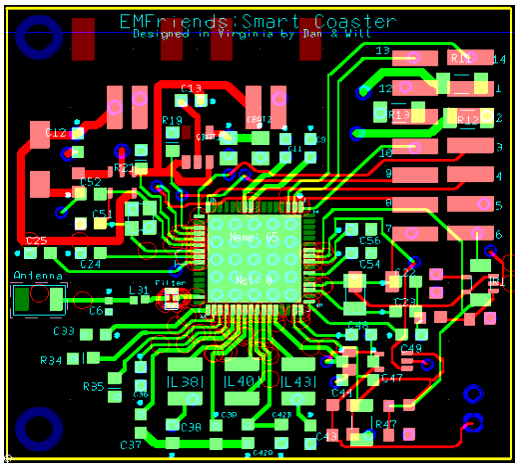


**Figure 6 First Board Layout**

Our first board could run simple code such as the flashing of lights, but contained a number of issues. The RF traces were not tested because the WiFi subsystem was never initialized due to faulty current sourcing issues. We determined that there were current sourcing issues because the code responsible for setting up the radio failed unpredictably. We deduced that this activity required more current than our board could supply. To test this theory, we decided to power up the board using an external power supply, one that could supply as much current as needed. When we did this, the board was able to make it further into the WiFi subsystem initialization, but still failed unpredictability. The current sourcing issue may have been caused by a regulator that could not provide enough current, an interrupted power plane, or bypass capacitors placed too far from pins. We strived to fix all of these issues on our second board. This board also lacked a number of features to make programming easier. The board lacked a reset pin which had to be soldered manually, internal voltage pins, a bottom silk-screen, and flash chip for storing code.

*Second Board*

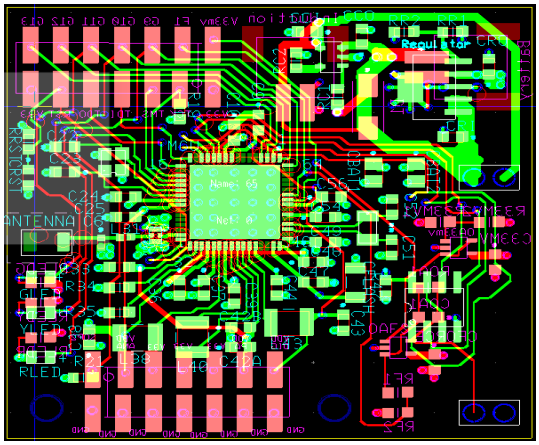

**Figure 7 Second Board Layout**

The second board fixed the problems in the first board but encountered new problems. The three potential current sourcing issues (capacitor location, interrupted power plane and regulator) were all fixed. The inconveniences (no reset pin, no internal voltages, no bottom silk-screen, no flash) were fixed. This board was successfully able to source enough current to initialize the WiFi subsystem. It was also able to communicate wirelessly over WiFi and transmit information about the amount of water in the class, proving that the custom RF trace we added worked as intended, which we were very excited about. Unfortunately, the board lacked UART communication so the program could not be flashed to memory despite having the flash chip. The flash chip recommended by TI was not in stock on DigiKey either. We used another flash chip with a similar footprint, but it may not have been designed to interface with the CC3200. We were not able to solder UART pins manually, and any effort to do so may have inadvertently fried the MCU. Despite solving the current sourcing issue, we were unable to get this board to run the code on powerup. It did work perfectly as intended from a computer, but after efforts to solder on UART pins, the board or the MCU was fried. We completely soldered two different boards and did have any luck with it. We even went back to 3W to change the MCU, but we suspect another part of the board was pushed to instability during our attempts to fix UART. If we had two more weeks or an additional layout, we are confident that we could have gotten it working.

*Header Board*

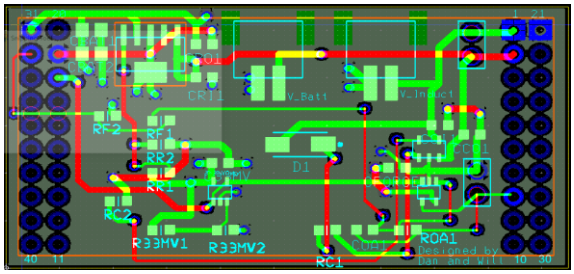


**Figure 8 Header Board Layout**

The Header Board was used in our demo to great success. The force sensing and power sections worked well with the launchpad. Since the power section worked fine, this was likely not responsible for the second board's failure.

*Inductive Charging*

Our coaster could be recharged via induction. We used transmission and receiving boards from Adafruit rather than build our own inductive chargers. The inductive charging pair communicated via the Qi protocol. We build a charging platform that contained the transmission PCB and connected to a wall outlet. The coaster contained the receiving chip and could charge through the wall of the coaster. We confirmed with a voltage meter that the battery charged when the inductive charger supplied power to the coaster. Despite charging a full battery, the inductive charger never overcharged and broke the battery showing that the overcharge protection circuit worked as expected.

*Mechanical Design*

The design of the coaster went through multiple iterations with regards to how the weight would be reliably measured by the force sensor. The basis of all these iterations is that there is a solid enclosure which would house all of the electronics. This will always be represented by the two models at the bottom in each figure. The box has as little I/O as possible with the intention of it being water sealed, and this included just a hole for the force resistor to slot through and a hole for a button. This box would then be fit into a cover which served to focus the weight of a glass onto the force sensor. The cover clips onto the box via the tabs shown on the box's base.

*Iteration 1*

The first design featured 1 solid cover that the enclosure box would sit into. The enclosure was printed with PLA plastic but the cover was printed with TPE plastic. TPE is a flexible plastic, and the idea was that it would add an extra layer of protection from the electronics, be easy to remove for cleaning, and most importantly, still be able to focus the weight onto the force resistor. However, after testing, we found that certain glasses and mugs, specifically with a rim around the bottom, would not register at all since the cover was very flexible. As seen in the figure below, if the rim around the bottom of the beverage vessel was able to surround the square intent on the piece below, no force would be registered. This provided as an unsuitable solution for the coaster cover.

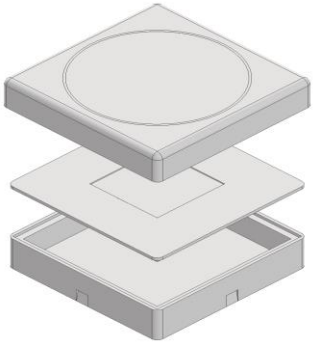

**Figure 9 Coaster Design Iteration 1**

*Iteration 2*

The second design featured a cover with 2 separate pieces. One piece was a circular disk which would indicate where a beverage vessel should sit on the coaster in order to register a reliable weight. The other piece was the rest of the cover which simply held and centered the disk in place. All components in this design were printed with PLA plastic. This design eliminates the influence of friction between the cover and the sides of the enclosure box. We decided to scrap this design because when the larger piece clips into the enclosure box, the circular disk is pressed against the force resistor. When no glass was present, the force resistor would register a large value, which reduces the range of weight that can be detected by the system. Furthermore, this design was difficult to assemble/disassemble and we figured that we could achieve similar results by combining both pieces, essentially reverting back to the first design but with hard plastic instead.

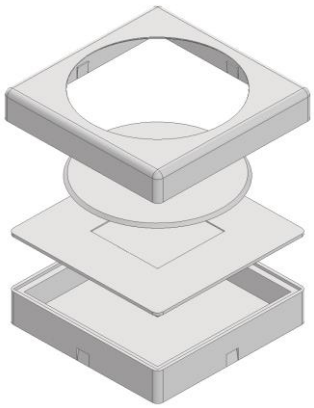

**Figure 10 Coaster Design Iteration 2**

*Iteration 3*

The final design was, just as described, iteration 1 where the cover was printed with PLA plastic instead of TPE plastic. It's much larger due to the use of a launchpad instead of our custom PCB. It includes mounts in the base for the launchpad to sit on so that it doesn't move around the coaster. Furthermore, this design features a hole for a pushbutton, which wasn't accounted for in previous iterations. This design yielded decent results, but we had to make some modifications post-print in order to get it to function well. These modifications include removing the tabs that lock the cover in and making the arch in the cover larger for the pushbutton. The former was required because, when the cover clipped in, a lot of force was applied onto the FSR, which was the same issue with the previous iteration. The latter was required due to a measurement error in the design. The arch for the pushbutton was too small, which prevented the cover from moving freely up and down. The produced inconsistent results depending on where the beverage vessel was placed on the coaster. This problem was fixed by widening the hole with clippers, a file, and an x-acto knife.

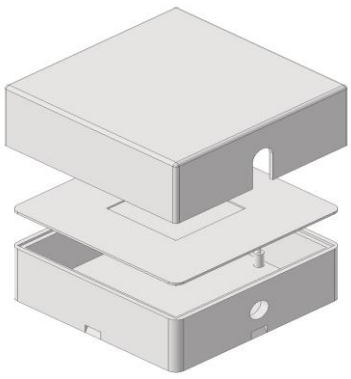

**Figure 11 Coaster Design Iteration 3**

*Induction Charger*

A small case was designed for the induction transmitter. The unit purchased for this project had the coil and PCB separate, which was not ideal to work with. The case features tabs for the PCB to clip into as well as cutouts for the inputs for power (USB and Micro-USB). The narrow part of the case is for the induction coil wires to pass through, and the large cutout in the back is for the large coil plate to sit in. The design also features 6 stubs where an optional cover can be placed.

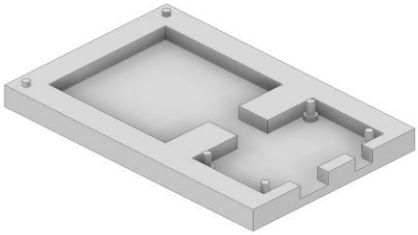

**Figure 12 Induction Charger Design**

*3D Printing*

All components for the coaster were sliced with Ultimaker Cura 4.3.0 and printed with a Ultimaker 3 3D printer. All the Inventor files for this design were converted into an STL format. The models were then printed with the following slicing settings:

- Material: Ultimaker Translucent and White PLA
- Layer Height: 0.32mm
- Wall Line Count: 2 Layers (includes Top and Bottom Layers)
- Infill Density: 10%, Triangle pattern
- Print Temperature: 200°
- Support: Everywhere (60° overhang angle)
- Build Plate Adhesion: Skirt

PLA plastic was used for printing because it was the most readily available plastic that was available to use. The layer height was chosen because most models lacked detail in the z-axis, and therefore a thicker layer height wouldn't compromise any details or features. It allowed the models to be printed with significantly higher speed. Furthermore, thicker layers provide a more structurally sound model, however, in practice, this introduced some errors in the final print. Mainly, some layers didn't adhere well to one another which produced some lumps on the bottom side of the coaster and tolerances that needed to be larger than accounted for. The wall lines were chosen because all walls in the models were thin. As a result, 2 wall lines would provide more than enough structure to the final model. Lastly, the infill was chosen as a low

percentage because, again, all the models featured thin walls, and therefore having minimal infill would not compromise the structure of the coaster.

*Software Design*

*Embedded Design*

The coaster program was a splice of the ADC TI SDK example code and the UDP Client TI SDK example code. First, it would connect to a hardcoded local area network (LAN) (we hardcoded a custom name and password for the LAN). It would also wait and try again every 5 seconds if it was unable to connect. Then it had an infinite while loop. In the while loop the coaster would check if it were still connected to the network. If not, it would wait 5 seconds then try to connect. Then it would read in voltage from the ADC. Then it would try to send that voltage in a packet to a hardcoded IP address.

*Server Design*

The server program was a python UDP server with some additional logic. First, it would ask what kind of drink was on the coaster. Based upon the answer it would store in global variables some thresholds associated with that coaster. Then it would listen for packets on a hardcoded IP address. After receiving a packet, it used a running average of it and the previous 2 packets, 2 thresholds hardcoded and based upon what drink is on the coaster, and the previous state (whether or not the drink was on the coaster) to decide whether or not the drink needs a refill. All of these plus a couple heuristics were required to make the server accurate because the underlying data was super noisy and the cup may not even be on the coaster.

*Software Issues*

Figuring out the handshake to use/designing the communication was tricky, because the laptop's IP address is not a constant. Whenever a laptop connects to WiFi, it gets assigned an IP address. However, in Windows 10, when you set up a hotspot for things to connect to, on that LAN, a laptop always assigns itself the same IP address. We hardcoded that IP address in the coaster to send packets to. As a whole, there were very few issues in the design/implementation of the software for this project.

**Project Timeline**

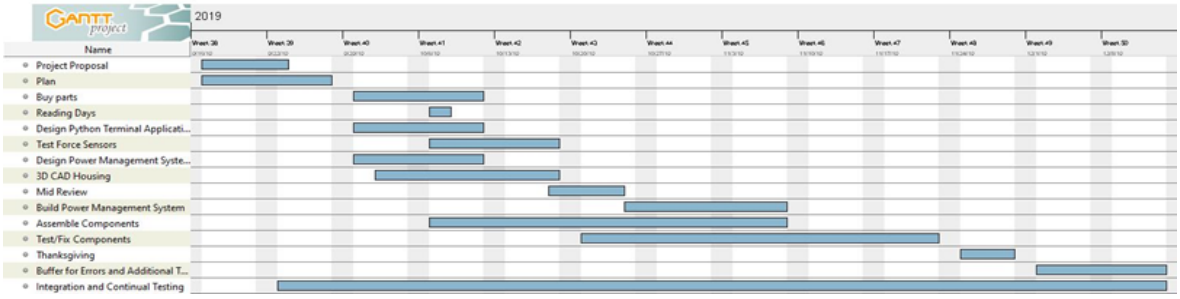


**Figure 13 Proposed Gantt Chart**

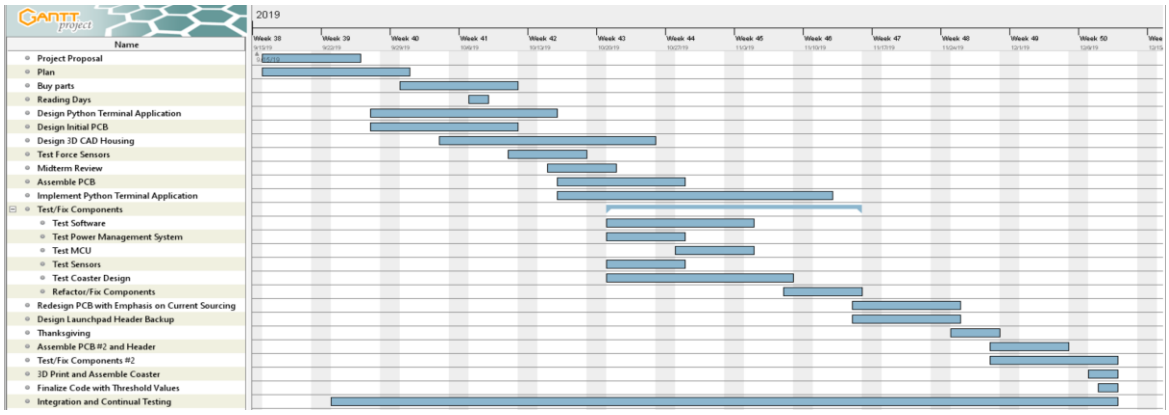


**Figure 14 Final Gantt Chart**

The main differences between the proposed Gantt chart and the final of events that occurred is that some events were pushed back and many new, smaller tasks were added towards the end. The reason for the delays were due to errors in parts ordering and unfortunately missing some deadlines. However, even though some events were pushed back, we managed to complete the project. The addition of many smaller tasks to the end of the project were due to unforeseen issues that we encountered, mainly revolving around the MCU and its intricacies.

Given how modular our design was, there were a handful of tasks that were easy to parallelize. For example, the code, physical design, and electrical design could all be done at the same time, with a bit of communication between members who were working on their respective sections. This is reflected in both Gantt charts. Testing was easily done in parallel. While specific portions of the PCB were being assembled in tested, the code was entirely written and being tested with the launchpad, and various tests were done with different force sensors and the preliminary designs for the coaster. A great example of how parallelizing tasks became extremely crucial was with the design and testing of the header board. When we deemed our PCB unusable, the header board was fully tested and ready to go and an updated design for the coaster was ready to be printed. It took around 3 days to fully make this transition, which gave us time to exhaustively test the PCB before determining that it wouldn't function properly. Tasks that were done in serial were those that couldn't be done in parallel otherwise. For example, designing a header board for the launchpad was not in the scope of this project until we discovered issues with our own PCB and needed a backup. Another example is establishing threshold values for the coaster. This couldn't be done until everything was packaged into the coaster otherwise the values would be incorrect.

In terms of distributing the work, Dan and Will were mainly responsible for the PCB, James was responsible for the code, and Adam and Taylor were responsible for the physical design. Each member also helped with another aspect of the project such that almost everyone was up to date. This allowed communications to occur seamlessly, which allowed much of the coaster to be developed completely in parallel.

## Test Plan

To accomplish a capstone project of this scale, we had to develop and abide by systematic test plans. These test plans became a strategy for debugging in the scenario when things did not work from the first time, a scenario we unsurprisingly found ourselves in many times this semester. This project can be simplified down to four fundamental disciplines or tasks: power, sensors, the microcontroller, and the RF circuitry. For each of these disciplines, a strategic test plan was developed. Each test plan was developed as a flow diagram, with different aspects to test based on the success or failure of previous steps. In this section, we will go into the details of these test plans and their role in the implementation of the Smart Coaster.

*Power*

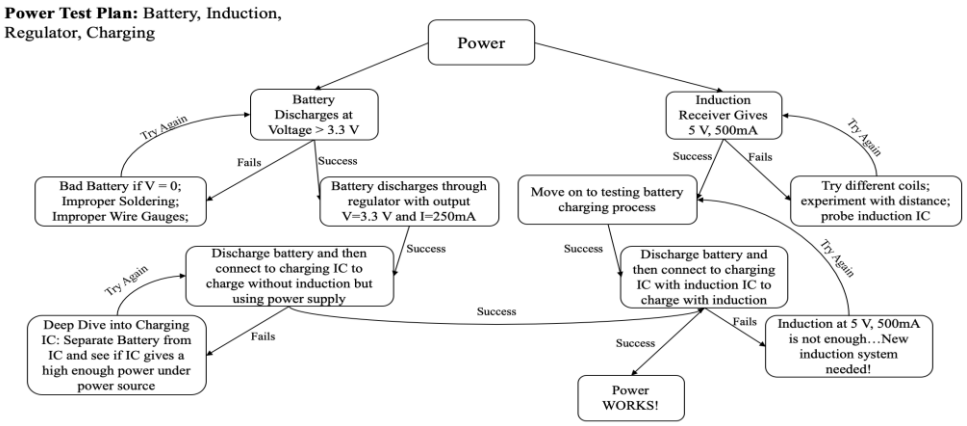


**Figure 15 Power Test Plan**

Before anything else can be tested, the power circuitry and design had to be functioning effectively. For this reason, the test plan for power was developed and referred to first. In this test plan, the term “power circuitry” refers to anything concerning the battery, the inductive coils, the regulators, and the charging of the battery. Please refer to the figure above for a more compressive picture to the test plan being described here. The test plan starts with the two main power sources: the battery and the inductive charging. From there, the test plan details a systematic check of the voltages produced by the sources and points to either a new area to test or repeat depending on (a) success or (b) failure. This test plan was developed for the mid-semester design review and was followed quite closely throughout the implementation of the coaster. In the end, we were able to successfully demonstrate working power circuitry.

*MCU*

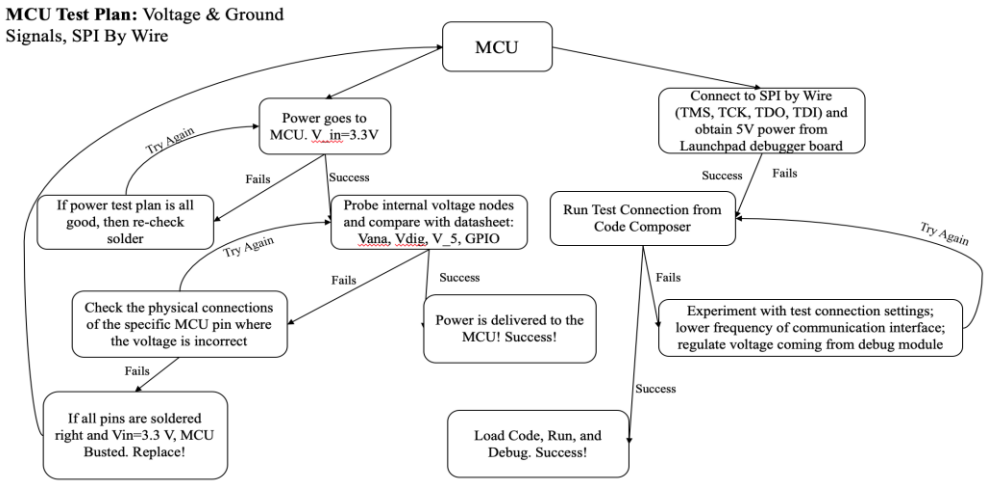


**Figure 16 MCU Test Plan**

After ensuring proper power supplies, the most natural next step was to test the schematic and layout of the microcontroller. Here the microcontroller test plan refers to any circuitry or layout traces that deal with the voltages into and out of the MCU, the GPIO pins, and finally the JTAG communications with a debugger. Similar to the power test plan, this test plan also develops a systemic approach based on a success or failure of a previous step. This test plan was followed quite closely in the iterative testing and soldering of the MCU and its components. For example, the MCU would almost always have VIN of 3.3 V, but would not always connect to the debugger or successfully run a simple program. In cases such as these, the test plan would recommend a probing of the internal voltage nodes, followed by a physical check of all solder joints. This was a step we did many times throughout the semester, and it often illuminated many mishaps and inaccuracies. Through this test plan, we discovered a fatal current sourcing issue in the first layout, an issue that would throw the initialization of the WiFi subsystem on the MCU into an infinite loop. Because of this discovery early on in the semester, we were able to layout and populate two new boards, which did not have this current sourcing problem. In another situation early in the semester, all internal voltage nodes were accurate, but JTAG communication was failing. We learned that we had forgotten to pin out the RESET, a problem we fixed in a later PCB layout.

*Sensors*

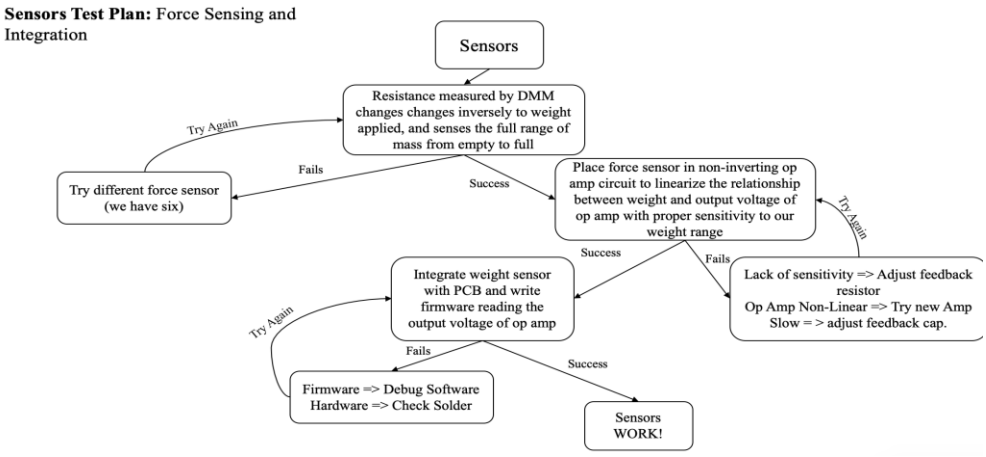


**Figure 17 Sensors Test Plan**

After the testing and iterative development of the MCU and the power circuitry, came the testing of the sensors. This test plan involved the forcing sensing and its integration into the overall system. Again, this test plan was followed quite closely throughout the semester. The first step in this plan was determining the right force sensitive resistor to use. This involved asking questions about the relative size of the sensing area and how to focus all the weight of a large glass on the pad. Then, the feedback resistor which determines the sensitivity of the weight sensing had to optimized. This resistor was changed many times to increase or decrease sensitivity of the weight sensing. As the mechanical 3D printed prototypes came in, the force sensing resistors and its interface with both the MCU and the coaster case had to be rechecked and considered. Following completion of the test plan through multiple iterations, we ultimately reported successful force sensing.

*RF*

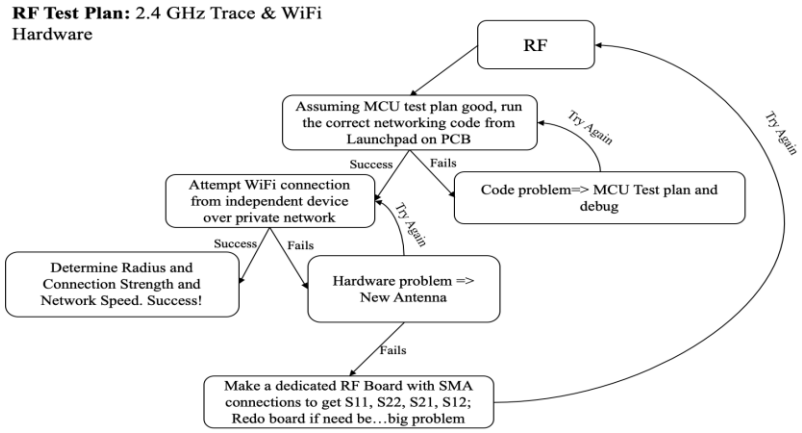


**Figure 18 RF Test Plan**

Once the power, MCU, and sensors were functioning, the last step to consider was the RF trace. As part of the system requirements, we had to demonstrate reliable communication over WiFi, a technology which operates at 2.4GHz. The layout of this WiFi trace needed an LC matching network, a microwave bandpass filter, and 50 Ohm characteristic impedance lines. Because of all of these special considerations, a test plan, similar to the first three was developed. After successful completion of all test plans, we reported successful RF communications with a local network and were able to send packets of information over a UDP client-server architecture.

*Coaster Design*

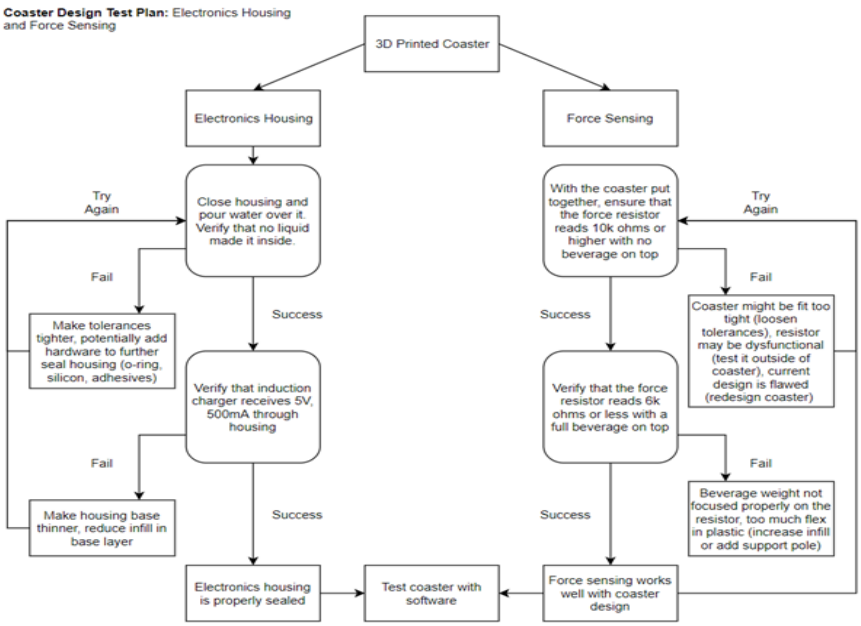


**Figure 19 Coaster Design Test Plan**

With all the electronics tested, the last step is to test the integration of everything in the coaster. This consisted of two branches, the first tests that the coaster fits tightly together and is water resistant, the second tests that the design can register reliable values from the force sensor. Once the coaster passes these tests, the software can be run to verify that all the subsystems are integrated properly.

## Final Results

We proposed 4 separate domains for our project to be judged on in the proposal: sensors, power, communication, and design. **Sensors:** The coaster reliably tells when the drink is empty. **Power:** The coaster has inductive charging and when we probed the voltages, the voltage on the battery goes up while the coaster was on the charger. **Communication:** Because we used WiFi, multiple coasters would work simultaneously. It would require trivial modification of the code to do so (Every coaster would get a different number in its program that it sends along with weight

to differentiate them). **Design:** Our device is water resistant and can be used multiple times. However, the final coaster is not aesthetically pleasing because it is too large. We were not able to get the PCB with an MCU to work so we used the launchpad with a header board instead.

We succeeded in 3 out of 4 domains in our proposal, and we completed part of the 4th domain.

**Costs**

The research and development of this coaster ran a total of $463.95 in parts and a total of roughly $680 including the services for soldering components and PCB orders. The cost to purchase all the components for a single coaster is $37.86. With the cost of manufacturing a PCB and services for soldering all the components, the total increases to around $100. If we were to manufacture 10,000 units, the total for all the components would be reduced to $16.28, which makes a coaster cost around $75 to manufacture in bulk.

**Future Work**

Future work in this project would include manufacturing a fully-functioning version of the final 4-layer PCB with an MCU. Our team ran into difficulties regarding this PCB during the testing process, and needed one more design iteration to produce a fully-functional and sustainable 4-layer PCB. Our difficulties came with the implementation of UART with the MCU, and this was something we did not have very much prior experience on from other coursework. Advice regarding avoiding pitfalls on this matter would be to test that subsystem separately early-on, in addition to asking advice from the professor on transitioning from using a launchpad to using an MCU integrated with the PCB.

As a result of reducing the size of our PCB back down to a small 4-layer board, future implementations of this device would include a reduced coaster size. This would substantially improve appearance, usability, and storage of the device, making it more appealing to patrons and restaurant owners alike.

This project could be expanded upon by adding further capabilities, such as game interaction for large sports bars, or adding user capability to change drink orders. In addition, a further project could focus on scale of this product, and look into reducing cost per coaster in order to make it a more purchasable product at mass scale.

This project may also serve as inspiration for other innovations in the food and service industry, encouraging other students to focus on non-restaurant applications of similar food and service products, investigating where technology can improve the food services industry.

## References


[1] A. Butz and M. Schmitz, “Design and applications of a beer mat for pub interaction,” Jan. 2005.

[2] K. Hutchinson, J. Gustafson, B. Houska, and M. Syed, “Aquarius Drink Filling Robot,” *The Oculus: The Virginia Journal of Undergraduate Research*, vol. 17.

[3] V. Beal, “What is 802.11 Wireless LAN Standards? Webopedia Definition.” [Online]. Available: https://www.webopedia.com/TERM/8/802_11.html. [Accessed: 16-Sep-2019].

[4] “Standards | IPC.” [Online]. Available: http://www.ipc.org/ContentPage.aspx?pageid=Standards. [Accessed: 16-Dec-2019].

[5] “PEP 8 -- Style Guide for Python Code,” *Python.org*. [Online]. Available: https://www.python.org/dev/peps/pep-0008/. [Accessed: 16-Dec-2019].

[6] M. Parker, “IP67 vs IP68: Waterproof IP ratings explained,” *Trusted Reviews*, 07-Sep-2018. [Online]. Available: https://www.trustedreviews.com/opinion/what-is-ip68-ip-ratings-explained-2947135. [Accessed: 16-Sep-2019].

[7] “Serial Communication - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication/uarts. [Accessed: 16-Dec-2019].

[8] “Qi Wireless Charging Standard | Electronics Notes.” [Online]. Available: https://www.electronics-notes.com/articles/equipment-items-gadgets/wireless-battery-charging/qi-wireless-charging-standard.php. [Accessed: 16-Sep-2019].

[9] “NSF/ANSI 51 & NSF/ANSI 61: Covering your product certification needs from food to drinking water.” NSF.

[10] “Welcome to Python.org,” *Python.org*. [Online]. Available: https://www.python.org/. [Accessed: 16-Dec-2019].

[11] “welcome home : vim online.” [Online]. Available: https://www.vim.org/. [Accessed: 16-Dec-2019].

[12] “CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) | TI.com.” [Online]. Available: http://www.ti.com/tool/CCSTUDIO. [Accessed: 16-Dec-2019].

[13] “The GNU C Reference Manual.” [Online]. Available: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html. [Accessed: 16-Dec-2019].

[14] “UNIFLASH Uniflash Standalone Flash Tool for TI Microcontrollers (MCU), Sitara Processors & SimpleLink devices | TI.com.” [Online]. Available: http://www.ti.com/tool/UNIFLASH. [Accessed: 16-Dec-2019].

[15] “NI Multisim and NI Ultiboard Professional Product Features - National Instruments.” [Online]. Available: http://www.ni.com/product-documentation/5609/en/. [Accessed: 16-Dec-2019].

[16] “Inventor | Mechanical Design And 3D CAD Software | Autodesk.” [Online]. Available: https://www.autodesk.com/products/inventor/overview. [Accessed: 25-Sep-2019].

[17] “Ultimaker Cura: Powerful, easy-to-use 3D printing software,” *ultimaker.com*. [Online]. Available: https://ultimaker.com/software/ultimaker-cura. [Accessed: 16-Dec-2019].

[18] C. A. Vock, P. Youngs, and A. Larkin, “Electronic drink coaster,” US7353136B2, 01-Apr-2008.

[19] A. Partovi and M. Sears, “System and method for inductive charging of portable devices,” US8629654B2, 14-Jan-2014.

[20] J. C. Kirk, “Electronic coaster,” US9432758B1, 30-Aug-2016.

[21] “CC3200 Application notes, User guides, Solution guides, White papers, Design files, More literature, Blogs.” [Online]. Available: https://www.ti.com/product/CC3200/technicaldocuments. [Accessed: 26-Sep-2019].