Clever Coasters

Rubba Ashwas, Mitchell Crozier, and Teodotas Kursevicius

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

***Abstract* —The goal of this project is to provide restaurants servers with a more simple and efficient way to ensure higher customer satisfaction through a portable and compact electronic coaster system. The design uses the CC2650MODA microcontroller with built-in Bluetooth antenna, a Velostat pressure sensor, and its own self-charging circuit to keep the system powered. With additions such as an intuitive user-interface, LED indicators and the ability to calibrate different cup sizes, the Clever Coaster system provides a friendly user experience to both the servers and customers of a restaurant.**

***Index Terms* — bluetooth, battery chargers, internet of things, pressure sensors, microcontroller**

I. Introduction

The Clever Coaster system was designed with cost and ease in mind, while also tailoring to the fact that different customers require different attention. Some want their water refilled when its halfway, while some never even take a sip of their drink until their meal arrives. Whichever the case, the Clever Coaster System integrates weight sensors and wireless communication to provide a tailor-made experience for every restaurant goer.

The goal of this project is to create a system of low-power, cost-efficient smart coasters that can wirelessly connect to a device at the table, which then connects to a main display where information about their current state is provided. The coasters should be easily chargeable in large numbers, ensuring that a minimal number of wires are needed. The coasters should last through at least a full day of service before needing to be re-charged. Also, they should seamlessly connect to the device at any table they are placed. The coasters should have the ability to detect if a drink placed on them is empty or full.

II. System Hardware

A. Microcontroller

Our microcontroller of choice was the CC2650MODA chip. We picked this chip for several different reasons, the biggest of which is that it has a built in antenna, meaning we didn't have to design our own antenna which need to be incredibly precise in order to function as intended, and could even violate FCC rules if we messed it up. Another reason is the ability to upload code easily through the two JTAG pins and our development board without needing to add a USB interface to each PCB we make, saving space and money. The final reason is the low power consumption of the chip, allowing us to run the coasters for at least a day before needing a recharge.

B. Weight Sensor

As shown in Fig.1 below, we decided on choosing Velostat, as it provided more flexibility and worked as needed during the component testing phase. In the component testing phase, we looked only at how the resistance changed as pressure/weight was applied to the surface of the Velostat or FSR. However, we need to be able to have the microcontroller detect this change through the measurement of output voltage. To do this, we can set up a simple voltage divider circuit as shown below. The voltage measured between the resistor R_1 and the Velostat will be connected to an input port pin on the microcontroller, which will then need to convert this voltage reading to a digital value. This can be done through the microcontroller's internal Analog to Digital Converter. As more weight and pressure is applied to the sensor, the voltage will drop because the resistor will simultaneously drop.

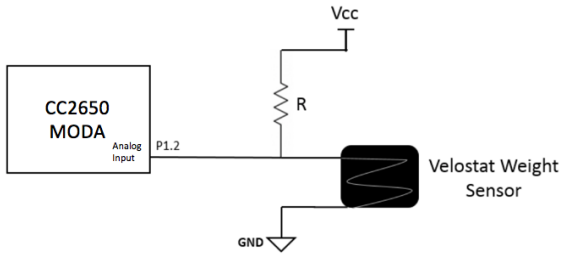

Figure 1: Voltage Divider connected to analog input pin

Equation (1) below shows the relationship between the change in resistance to the change in output voltage. The analog input reads the voltage and can compare it to our calibrated cutoff values.

$$V_{out} = V_{DD} \frac{Resistance\ of\ Velostat}{Resistance\ of\ Velostat + R_1} \qquad (1)$$

III. Device Power

The Clever Coasters have several requirements to meet regarding their power source. Primarily, they must be wireless, which means that a battery is required in the

design. The battery powers the microcontroller, wireless communication chips, the PCB, sensors, and LEDs. The battery must also have enough capacity to last the entire business day before being charged to avoid being an inconvenience to customers and employees. The coasters must be easy to recharge to not be an annoyance to waiters who must manage them.

A. Charging Method

Three charging methods and one non-charging method were considered for the Clever Coasters design. The non-charging method is simply the use of disposable batteries that waiters would have to replace when they run out. This is not very realistic because it would require the restaurant to stock and properly dispose of many batteries on a regular basis. Furthermore, it would be time consuming for waiters to be constantly switching batteries out of dead coasters and annoying for customers if a battery died during their meal. Waterproofing is another concern for a system that must be opened a regular basis. Because of those reasons, the non-charging version was dismissed early.

The next possibility considered was a plug-in charger. This would require a place for coasters to be held and a large number of wires to connect them to the charger. The benefits of this system is that everyone knows how to use something like a USB charger and it would be fairly easy to implement in the design while maintaining the waterproofing. Unfortunately, the system doesn't scale well and was also dismissed.

Another option was to create a charging dock where multiple coasters are stacked and charge at the same time. This is very easy for wait staff to use and ensures a highly reliable charge every day. It does pose a small safety concern for a potential of the dock to be short-circuited, but this can be prevented by circuits that limit current. Additionally, this design requires each individual coaster have a charging circuit within it.

The last option was to use induction charging. This would require an even more complex charging circuit within the coasters because it would have to have a coil and be capable of converting high frequency AC to DC. This system is also extremely inefficient because of the large EM fields it must generate regardless of how many coasters are placed on it. Not only that, but depending on the implementation, it might not be able to charge very quickly. [1] In the end, this option is the most complex.

Ultimately, the charging dock solution was selected as the one that will be used for the Clever Coasters system. It provides a good balance of design complexity to ease of use, safety, and efficiency.

B. Battery Type

The type of battery used in the Clever Coasters was another consideration. The types of batteries considered were disposable alkaline, reusable alkaline, lead-acid, nickel-cadmium, nickel-metal-hydride, lithium-ion, and lithium-ion polymer.

Since a system where the batteries are replaced by hand when they run out was ruled out, disposable alkaline batteries were ruled out as well.

Likewise, reusable alkaline batteries were dismissed as an option because of several bothersome factors. First, their energy density is not as high as that of other batteries, they suffer from poor cycle life so the lose a lot of capacity over time, and they have high internal resistance. Their positive qualities are a fast recharge time and low self-discharge, but those are not enough to overcome the drawbacks.

Lead-acid batteries are even worse in the energy density department, but they do have very low internal resistance, high overcharge tolerance, and a low self-discharge. The real problems with them come in the form of a long charge time and a potential maintenance requirement. Over all, they're not the best choice.

The two nickel-based batteries are very similar to each other. NiCd batteries have better cycle life, extremely fast recharge and a better overcharge tolerance. NiMH, on the other hand, have a higher energy density and are somewhat more environmentally friendly since they don't contain cadmium. Both, however, require very precise charging and discharging as a form of frequent maintenance and are therefore disqualified.

The last two battery types that were considered are lithium-ion and lithium-ion polymer. These two are almost identical except that the form factor of the LiPo batteries is easier to work with and Li-ion have worse overcharge tolerance. Aside from that, they have both have high energy density, good cycle life, no maintenance requirement, a fast recharge time, and a low tolerance to overcharge. Because it was good in almost all of the important categories, the LiPo was chosen for Clever Coasters with the understanding that charging would have to be carefully regulated. [2]

C. LiPo Charging Theory

Lithium-ion polymer batteries are charged in a two-stage process. First, they undergo current limited charging in which the current being pushed into the battery is limited by a regulator to be less than the rated amount. Typically, charging at 1C (or one-hour times the capacity of the battery) is considered safe and conservative. During this

process, the battery increases in voltage until it reaches the target voltage which would be 4.2V in the case of the LiPo.

At this point, the battery is at about two thirds of its full charge. Now, in order to prevent over-charging the battery, the constant voltage stage of charging begins. Here, the voltage across the battery terminals is kept at 4.2V and the current is slowly decreased. This behavior arises from the internal resistance of the battery. Ideally, the battery cell should reach 4.2V when measured without resistance, but while charging, some of the 4.2V comes from the current passing through the internal resistance. In this way, the current decreases until it is just drip charging a negligible amount of current into the battery. [3]

D. Charging Circuit

The purpose of the charging circuit is to control the rate of charge of the battery when it is placed in the external charging dock. The particular circuit used here is inspired by a circuit found in TI literature and is a very simple design for a single cell lithium battery. It uses a single linear regulator to control the charge as described in the LiPo Charging Theory section.

The rectifier on the left is used to deliver the right voltage to the regulator regardless of the orientation that the coaster is placed in the charging dock. The diode at the output of the regulator is used to protect the regulator from a current reversing when the coaster is taken out of the charging dock and the input goes to zero. Finally, the voltage divider on the right is used for the feedback pin to receive an input and control the output voltage. The resistors are chosen based on the desired Vout.

This design uses the LP2951 regulator and charges at a constant current of 100mA when in the current limited stage of charging. [4] There are also capacitors at the input, output, and feedback pins to reduce noise, but these are not shown in the Fig 2. below.

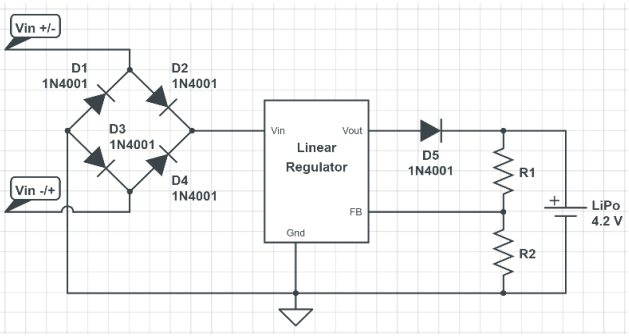

Figure 2: Charging Circuit

IV. System Board Design

Since the coasters need to be compact and small, our PCB design had to be as simplified as possible to ensure it would fit the dimensions of the 3D-printed coaster housing. Therefore, only the microcontroller, charging circuit, and voltage divider were included in the circuit design. Since the CC2650MODA has a built-in antenna, more board space was available to focus on the other aspects of the design.

A. Schematic Design

As shown in Fig.3 below, the main sub-systems included on the design were the voltage divider, charging circuit, and microcontroller. Multiple pins are provided to connect external components such as LEDs and ability to load code onto the microcontroller with an external debugger.

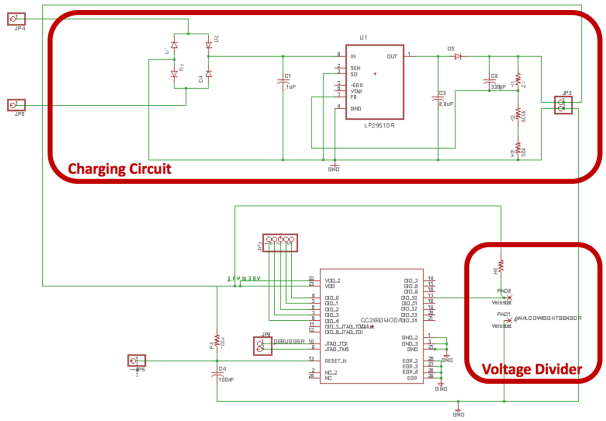

Figure 3: Schematic of System Design

B. PCB Design

The PCB layout of the coaster design was challenging, since the microcontroller and Velostat sensors needed to be placed on the edges of the board. In Fig.4 below, the dotted line represents where the copper plane would lie. The built-in antenna (top of the microcontroller) needed to be clear of copper according to TI's datasheet for the component [6]. Aside from the microcontroller and linear regulator, all other components are through-hole, to allow for more simple hand soldering. There are also more through-hole components available in the lab to utilize, which would help avoid the issue of waiting for components to ship.

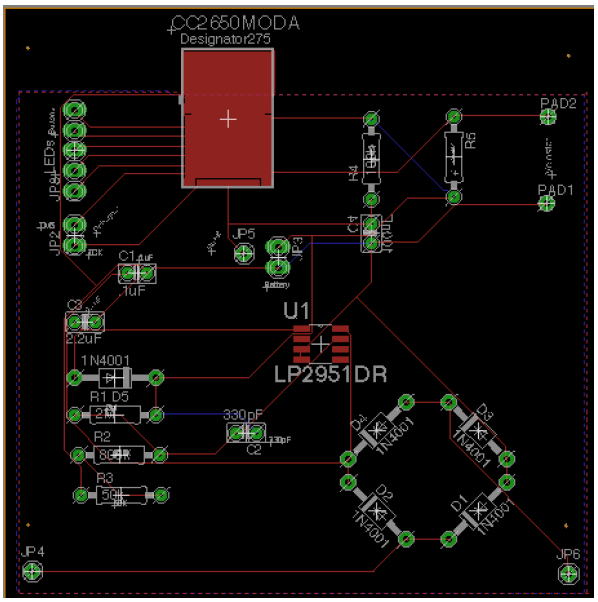

Figure 4: PCB Layout

V. Software

The software for our system was split into several components. We needed to program the actual microcontroller to read the analog input from our weight sensor, software that helps the coaster communicate information to the servers, and also an interface for the waiter to interact with.

A. Microcontroller Software

The microcontroller for the Clever Coasters must be able to communicate over Bluetooth Low Energy, measure the weight of a cup, and be calibrated. This functionality must be programmed into the chip in such a way that these functions can work together simultaneously and seamlessly.

The main thing that allows for multiple, simultaneous functionality in the microcontroller is the TI-RTOS, or Real-Time Operating System. The TI-RTOS runs its programs in a task context. Each task is given different priorities and they can be preempted by another task of a higher priority. Therefore, it is important to give long, persistent tasks low priorities while giving the highest priority to short, important tasks. [5]

There are also hardware and software interrupts which preempt the task context and execute immediately. Ideally, these are not used to do any processing because they block other tasks from taking place. Instead, we use them to start a high priority task which will get done in the normal task context, but allow new interrupts to happen without blocking.

When Clever Coasters first turns on, it initializes its services and begins advertising on Bluetooth. When a BLE enabled device connects to it, it allows that device to see the state of the weight sensor, to subscribe to that state, and to calibrate the coaster by sending the right signals. When a signal is sent to the coaster over BLE, it is placed into a buffer, interrupts the current task, and sends a message that the buffer has new information. The coaster can then process the information. In this case, the information is usually a calibration signal that tells the coaster to store the value it is detecting with the weight sensor as a specific threshold that can later be used to identify whether a cup placed on it is full, half full, empty, or not placed on it at all.

As the microcontroller spends most of its processing power on the BLE and message passing tasks, the sensor controller monitors the weight sensor independently. It is a very low power controller that periodically checks the sensor and does some basic calculations to determine if there is a need to notify the main application of a change. Since we have thresholds set for the different levels of liquid in the cup, the sensor controller doesn't send any messages until a threshold is passed. This means noise is not being processed by the relatively more power-hungry main processor.

Overall, the processor does not use much energy and can remain idle on very low power. With the proper use of interrupts, the sensor controller, and even a separate core for the RF communication which further decreases the amount of time the processor spends outside of idle. [6]

B. Android App

The Android app is designed to be the middleman between the microcontroller and the employee webpage. It uses three main communication protocols to achieve this, NFC, WiFi and Bluetooth. NFC or Near Field Communication is an extremely short range protocol that an NFC enabled device can use to read and write data on NFC tags through electromagnetic signals. The tags themselves don't use any power, which means we can write the tags inside each coaster and whenever they come in range of the android device, it will automatically read all the information stored in the device from the coaster without additional power needing to be supplied by the coaster itself. One of the pieces of information stored on the NFC tag will be the coaster's bluetooth address, and the id of the characteristic to be read to get the status of the coaster. Using these pieces of data, the app will automatically connect to the coaster through bluetooth and perform actions based on what value it reads from the characteristic data. The actions that it performs will all involve updating the database, which is hosted on a server

in a different location so we need to use WiFi to access it. Below is a high level overview of how the app uses NFC, WiFi and Bluetooth to accomplish these tasks. The NFC tag containing all the information the app needs to connect means that the app can begin polling as soon as the tag is scanned.

The app makes use of a number of different third party libraries to make developing the application easier, such as Retrofit[7] to easily connect to and send request to the REST API running on a server the group set up, Gson[8] to both marshal and unmarshal the JSON data our server sends and receives, RxJava[9] to eliminate blocking behavior for tasks that could take time to complete such as waiting for a response from the server. We also heavily made use of the built-in default android libraries for NFC and Bluetooth low energy.

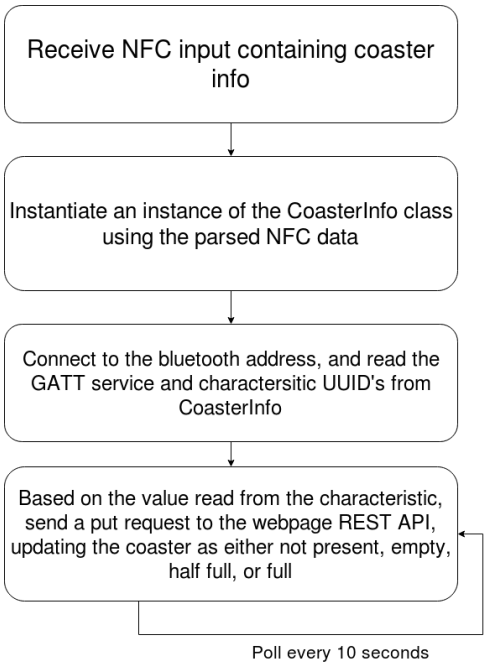

Figure 5: NFC flowchart

C. Website

Our webpage was designed to be a simple display that listed available coasters, what table they were at, and the status of the cup for employees to be able to tell at a glance which tables need service. Since the webpage is relatively simple, instead of renting server space with a service like AWS, we went with a server hosted on one of our desktops, with the port opened and a free dns host to give us our domain name. This lets us send and receive data to any internet connected device, instead of requiring them to be on the same local network.

The webpage uses several different technologies to run, in what is commonly called the MERN stack, which stands for MongoDB[10], Express[11], React[12], NodeJS[13]. MongoDB is our database of choice because it requires less setup than a traditional relational database such as MySQL, and the document style database of Mongo leads to easier integration into other parts of our project like the Android app, which as mentioned earlier uses Gson to convert the data sent from our database into Java objects. Express is a tool that simplifies how to handle incoming traffic to our server, for example if we receive a put request, we update our database with the information contained in the body of the request and send the updated record back as a response to let the requestor know the update occurred. React is a component based front end framework that allows us to easily create user interfaces without needing to write a huge amount of html and css code. Node is the backbone of the webpage, built around handling events asynchronously so it is able to handle a large number of requests without getting behind. The webpage also uses a third party library called Mongoose[14] to create a schema for database that the data being sent to our server must follow, such as the table number for this coaster and its status, etc.

VI. Conclusion

We believe that our project will benefit the end users by providing an intuitive and cost-efficient system available for continuous usage. Separating our project into multiple subsystems, Software, Power, Microcontroller, Sensors, allowed testing to happen in isolated chunks. This made debugging and errors easier to detect and fix. Choosing the right microcontroller was critical in allowing our design to be simplified, which is why we ended up changing the microcontroller to one with a built-in antenna. The biggest challenge faced was how we would get the coasters to seamlessly communicate information between devices, so not having to design our own antenna from scratch really helped ease the process of this design requirement.

Acknowledgement

The authors would like to thank Lei Wei with his continued assistance throughout this project.
The Team

Rubba Ashwas will be graduating with a Bachelor's of Science in Computer Engineering with Cum Laude Honors distinction. She has been an active member on campus as an electrical networks tutor and former president of the Tau Beta Pi engineering honor society. In the past, she has completed a computer vision REU, and interned at both Harris and Microsoft. Upon graduation, she will be taking 6 months off to travel and then starting a full-time position at Microsoft as a Program Manager.

Mitchell Crozier will be graduating this semester with a Bachelor's of Science in Computer Engineering. He likes to take on a lot of personal software projects, and loves to attend hackathons to build android applications. This past summer, he interned as a Software Engineer at Vencore, and will be looking for a full-time position upon graduation.

Teodotas Kursevicius will graduate with Honors and receive a Bachelor's of Science in Computer Engineering and Minors in Mathematics and Engineering Leadership in May of 2018. He is currently the Vice President of the Society of Sales Engineers and is proud of his 1st and 3rd place wins in the National Sales Engineering Competition. He has had two internships with Microsoft as a Software Engineer and plans to work there full-time in after graduation on the Word for Mac team.

REFERENCES

[1] F. Mishriki, "Design Considerations in Modern Wireless Power Transfer Systems: Coil Geometry.," PowerbyProxi, 27 Nov 2016. [Online]. Available: https://powerbyproxi.com/2014/09/design-considerations-modern-wireless-power-transfer-systems-coil-geometry/>. [Accessed 27 Apr 2017].

[2] "What's the Best Battery," Battery University, 21 Mar 2017. [Online]. Available: http://batteryuniversity.com/learn/archive/whats_the_best_battery. [Accessed 27 Apr 2017].

[3] Chester Simpson, "Battery Charging," Texas Instruments Incorporated, 2011. [Online]. Available: http://www.ti.com/lit/an/snva557/snva557.pdf. [Accessed 27 Apr 2017].

[4] "LP295x Adjustable Micropower Voltage Regulator with Shutdown," Texas Instruments, [Online]. Available: http://www.ti.com/lit/ds/symlink/lp2951.pdf. [Accessed 27 Apr 2017].

[5] "CC2640 and CC2650 SimpleLink Bluetooth low energy Software Stack 2.2.1 Developer's Guide," Texas Instruments Incorporated, Oct 2016. [Online]. Available: http://www.ti.com/lit/ug/swru393d/swru393d.pdf. [Accessed 17 Nov 2017].

[6] "CC2650MODA SimpleLink Bluetooth low energy Wireless MCU Module," Texas Instruments Incorporated, July 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2650moda.pdf. [Accessed 17 Nov 2017].

Retrofit by Square Inc [Computer software]. (2013).

[7] Retrieved from http://square.github.io/retrofit/

[8] Gson by Google [Computer software]. (2008). Retrieved from https://github.com/google/gson

[9] RxJava by Netflix [Computer software]. (2013). Retrieved from https://github.com/ReactiveX/RxJava/tree/1.x

MongoDB by Mongo [Computer software]. (2008).

[10] Retrieved from https://www.mongodb.com/

Express by StrongLoop[Computer software].

[11] (2008). Retrieved from https://expressjs.com/

React by Facebook [Computer software]. (2017).

[12] Retrieved from https://reactjs.org/

Node by NodeJS Foundation [Computer software].

[13] (2017). Retrieved from https://nodejs.org/en/

Mongoose by Learnboost [Computer software].

[14] (2013). Retrieved from http://mongoosejs.com/